EAP Working Group
INTERNET-DRAFT
Category: Experimental

<draft-puthenkulam-eap-binding-04.txt>
27 October 2003

Jose Puthenkulam
Victor Lortz
Intel Corporation
Ashwin Palekar
Dan Simon
Microsoft

The Compound Authentication Binding Problem

Abstract

There are several motivations for using compound authentication methods using tunnels, but man-in-the-middle attacks have been found in these protocols under certain circumstances. They occur when the inner methods used inside a tunnel method are also used outside it, without cryptographically binding the methods together. At the time of writing this document, several protocols being proposed within the IETF were vulnerable to these attacks, including IKE with XAUTH, PIC, PANA over TLS, EAP TTLS and PEAP. This document studies the problems and suggests potential solutions to mitigate them. We also provide a reference solution for an EAP tunneling protocol like PEAP.

Table of Contents

1. Introduction

As authentication protocols evolved over time, weaker ones have either been replaced or modified to meet the increasing security needs in new environments. The development of secure tunneling techniques like [TLS] and [IPSEC] have generated a lot of interest in securing legacy or weaker authentication protocols using tunnels. We call such compositions of multiple authentication protocols that are used in concert to accomplish a specific purpose, Compound Authentication Methods. They may be used for several purposes, including user authentication for granting network access or establishing a security association for confidentiality and integrity protection of data traffic.

We highlight certain man-in-the-middle vulnerabilities associated with the composition of compound authentication methods that are being proposed or already in use today. These include authentication methods that are commonly encapsulated within an independently authenticated tunnel that provides additional protective properties. Some examples of compound authentication methods using tunnels include EAPMD5 using [EAPTTLS], [EAPMSCHAPV2] using [PEAP], PANA over TLS [PANATLS], HTTP authentication over TLS, [XAUTH] with ISAKMP as part of IKE, and secure legacy authentication support for IKEv2 [SLA]. These may also include multiple authentication methods used in sequence when multiple credentials are used in different stages of authentication.

Editor's Note: Some of the above mentioned protocols and others mentioned in this document have recently been or are being updated with solutions for this problem. This document lays out the general principles for arriving at such solutions.

1.1. Scope

This document identifies the man-in-the-middle attacks that are possible when one-way authenticated tunnels are used to protect communications of one or a sequence of authentication methods; and characterizes the solution(s) that address the attack. The context studied is the use of compound authentication for granting network access using a client and authentication server setup.

Intuitively, certain attacks may also be possible with sequences of authentication methods even if they are not used within tunnels, but we do not address those scenarios here. This is because such authentication sequences are open-ended and unless a protocol method defines the security constructs of the sequence in totality, its difficult to analyze the attack scenarios. Also at the time of writing, the authors are not aware of such open-ended sequences with clearly defined security constructs.

constructs. If they do exist further study may be needed to see if the attacks we describe apply to them and also whether the solutions make sense.

We study the root causes of the attack and develop solutions using a mix of policy based and cryptographic means. We also describe a reference solution as a logical extension to an EAP based tunneling protocol. However the same principles could be used in other classes of authentication protocols as well.

1.2. Motivations for Compound Methods using tunnels

They are the following:

[1] Support for legacy authentication methods in new environments:

These new environments have brought new requirements including identity privacy, mutual authentication, authentication data confidentiality, integrity protection, replay and dictionary attack protection and strong cipher keys for the authenticated link, especially in the context of wireless networks. Secure tunneling techniques like TLS and ISAKMP with strong security properties provide a way to accomplish this in an extensible manner, providing longer life for easy to use legacy methods.

ex: Using legacy PAP that has no key derivation in 802.11 WLANs, by running it inside an EAP-TTLS tunnel that provides the additional protection needed.

[2] Consistent security properties for authentication methods:

Any time a new credential type is introduced, it becomes necessary to construct a complete cryptographic protocol with all the necessary security properties, which is not an easy task to accomplish. The availability of well-reviewed tunneling protocols like TLS and ISAKMP provide the opportunity to construct simpler methods for new credentials that can use the protection of the tunnel to do authentication securely. On a wireless network supporting multiple authentication methods, this enables consistent security for all credential types. This is possible because these protocols allow cipher suites to be selected and configured independent of the authentication methods used inside.

ex: EAP-SIM running inside a PEAP tunnel that uses a TLS tunnel for privacy protection.

[3] Support for Multiple credentials

New system capabilities in certain cases require authentication of more than one credential between two peers. This sometimes requires authentication methods with different security properties to be sequenced. By running the sequence inside a tunnel, it enables providing a protection layer for all the methods.

[4] Deployment aspects for securing legacy methods:

The ease of deployment of secure tunneling techniques using server authentication alone as in TLS and also IPSec (though they allow mutual authentication), making them even more popular candidates for securing the use of legacy authentication methods.

ex: The TLS tunnel in PEAP can be established with server authentication using a server certificate. No client certificates that are unique per user are required.

1.3. Problem Statement

This document suggests that compound methods have evolved from need, but their composition today as it is practiced, has some man-in-the-middle vulnerabilities in certain circumstances. These were not known at the time these compositions were suggested but have been found recently [MITM]. As compound methods have widespread application, this document studies the problem, its circumstances and and suggests solutions for mitigating them using a mix of protocol extensions and policy based techniques.

1.4. Requirements language

In this document, several words are used to signify the requirements of the specification. These words are often capitalized. 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].

1.5. Terminology

This document frequently uses the following terms:

Authenticator
: The end of the link initiating authentication. In IEEE 802.1X the term authenticator is defined and it has the same meaning here. It is also referred to as a NAS (Network Access Server).

Peer
: The end of the link, which is being authenticated by the Authenticator. In IEEE 802.1X, this end is known as the Supplicant. We also sometimes refer to the Peer as a Client of the Authentication Server.

Authentication Server
: It is an entity that provides an Authentication Service to an Authenticator. This service verifies from the credentials provided by the Peer, the claim of identity made by the Peer. It is also referred to as the backend authentication server.

Legacy Authentication Methods
: These are mostly one-way authentication methods widely used today that are either based on passwords or secure tokens or challenge-response methods that have no key derivation, no message authenticity or integrity protection, no identity privacy protection, use weak algorithms and are vulnerable to many well known attacks.
ex: PAP - Password Authentication Protocol

Legacy Credentials
: Passwords are typically referred to as legacy credentials. They may also include cases where the password is stored in a secure token device and not known to the user.

Sequenced Methods
: Authentication methods which are used in sequence one after another where each one completes and the next one starts. The authentication is complete after the final method in the sequence is completed.

Tunneled Methods
: The first method sets up a tunnel and subsequent method(s) run within the tunnel.

Binding Phase
: The protocol stage where validation of the peers participating in the compound authentication method is carried out after all the methods have completed or one or more successive individual methods have completed.

Compound MAC
: A Message Authentication Code (MAC) computed using keying material from multiple methods used in a compound authentication method. This is computed during the binding phase.

Compound Keys
: Session keys computed using keying material from multiple methods used in a compound authentication method. This is computed during the binding phase and can be used for ciphering at the lower layers or as application specific keys.

2. Problem Description

2.1. Overview

The attack described in this document can be mounted against a number of proposed IETF protocols, including IKE with [XAUTH], [PIC], [PANATLS], [SLA], [EAPTTLS], and [PEAP]. Each of these protocols supports tunneling of legacy authentication methods such as CHAP [RFC1994], EAP-MD5 [RFC2284], One-Time-Password (OTP) [RFC1993], Generic Token Card (GTC) [RFC2284], and SecurID [SECURID] in order to provide a number of benefits, including well-understood key derivation, fast reconnect, mutual authentication, replay and dictionary attack protection and privacy support.

The attack is enabled when compound authentication techniques are used, allowing clients and servers to authenticate each other with one or more methods encapsulated within an independently authenticated tunnel. The simplest attacks occur when the tunnel is authenticated only from the server to the client, and where tunneled authentication techniques are permitted both inside and outside a tunnel using the same credential. The tunnel client, having not proved its identity, can act as a man-in-the-middle, luring unsuspecting clients to authenticate to it, using any authentication method suitable for use inside the tunnel. The attacks are possible even though the tunnels created within these protocols utilize well-analyzed protocols such as ISAKMP [RFC2408] and TLS [RFC2246], because mutual authentication (supported within both protocols) is not used.

Since the initial authentication only authenticates the server to the client, or provides only an indication of group membership (where group pre-shared keys are used within IKE), and because the keys derived within ISAKMP and TLS are not subsequently included within the tunneled authentication methods, there is no demonstration that the ISAKMP/TLS endpoints are the same as the tunneled authentication endpoints.

Where authentication techniques are enabled both inside and outside the tunnel, such as when they are in use for multiple purposes (e.g. dialup or web authentication) then an attacker can induce an unsuspecting client to authenticate, then tunnel the authentication within [XAUTH], [PIC],[PANATLS],[EAPTTLS] or [PEAP].

For the purposes of the attack, it makes no difference whether the authentication method used inside the tunnel supports mutual authentication or not. The vulnerability exists as long as both sides are not required to demonstrate participation in the previous "tunnel authentication" as well as subsequent authentications, and as long as keys derived during the exchange are not dependent on material from *all* of the authentications.

Thus it is the lack of client authentication within the initial security association, combined with key derivation based on a one-way tunnel authentication, and lack of "cryptographic binding" between the security association and the tunneled inner authentication method that enables the vulnerability. In addition, it is necessary for the same authentication credentials to be used inside and outside of tunnels.

Despite the prevalence of man-in-the-middle vulnerabilities within these compound authentication protocol proposals, it should be noted that these vulnerabilities do not imply any design flaw in (a) either in the tunnel creation protocols (such as IKE [RFC 2409] or TLS [RFC 2246]), or (b) in some cases the authentication methods themselves (such as EAP-AKA or EAP-MSCHAPv2).

IPsec VPN implementations which require strong mutual authentication within the tunnel prior to permitting subsequent authentication are not vulnerable to this attack. For example, when L2TP over IPsec [RFC3193], or IPsec tunnel mode [DHCPIPsec], is used with certificate authentication or unique pre-shared keys, the attack is not possible. By requiring strong mutual authentication via certificates or a unique pre-shared key, the tunnel server obtains the ability to verify the identity of the tunnel client. The tunnel server may then subsequently apply access control in order to limit authentication within the established tunnel.

However, where group pre-shared keys are used (as is common in IKE Main

Mode implementations that support clients with dynamically allocated IP addresses), followed by one-way authentication mechanisms such as CHAP [RFC1994], the vulnerability is exposed. Since group pre-shared keys

only determine group membership but authenticate neither the client nor the server, it is not possible for the server to enforce access controls on individual members of the group. Since CHAP is a widely used authentication method, an attacker can easily gain access to a client willing to engage in CHAP authentication. This allows any client with knowledge of the group pre-shared key to act as a man-in-the-middle for another member of the group.

The concept of EAP tunneling has been introduced by recent work-in-progress such as [PIC], [PANATLS], [EAPTTLS], and [PEAP]. However, these proposals have not yet been published as RFCs, and are not yet widely deployed.

Note that man-in-the-middle vulnerabilities are not a necessary consequence of "credential reuse". For example, they need not necessarily occur where the same authentication credentials are used in accessing the network via multiple media. For example, L2TP [RFC2661], when used in "compulsory tunneling", assumes that the same credentials are used for both PPP and VPN access. PPP dialin users are then permitted to access a VPN by tunneling PPP packets from the network access server (NAS) to the VPN server. This is mainly because, on physically secure networks, unlike wireless networks, its harder to become a man-in-the-middle.

2.2. Attack scenario

This section describes how man-in-the-middle vulnerabilities can be exploited, as well as discussing the underlying causes of the attacks. We use a single example to highlight the problem. But the weakness of the composition of tunnel-based compound authentication methods should become apparent even in a broader context using this example.

The major scenario for the attack is a one-way authenticated tunnel encapsulating subsequent authentication methods. In this scenario, the client and server first establish a tunnel, then include within the tunnel one or more authentication method(s). The attacker first poses as a valid client to the server and establishes a tunnel that is authenticated only on the server end, obtaining tunnel keys. Since these keys protect a conversation between an attacker and a server, the strength of the key derivation is not relevant. For the purposes of exploiting the vulnerability, the tunneling protocol can be any protocol that does not provide mutual authentication.

Once the attacker has established a tunnel to the server, it seeks to induce clients to connect to it. This can be accomplished by having the attacker masquerade as a legitimate 802.11 access point (AP) or Ethernet switch implementing [IEEE8021X], a PPP Network Access Server (NAS), a SIP server supporting CHAP, or a VPN server using a protocol such as

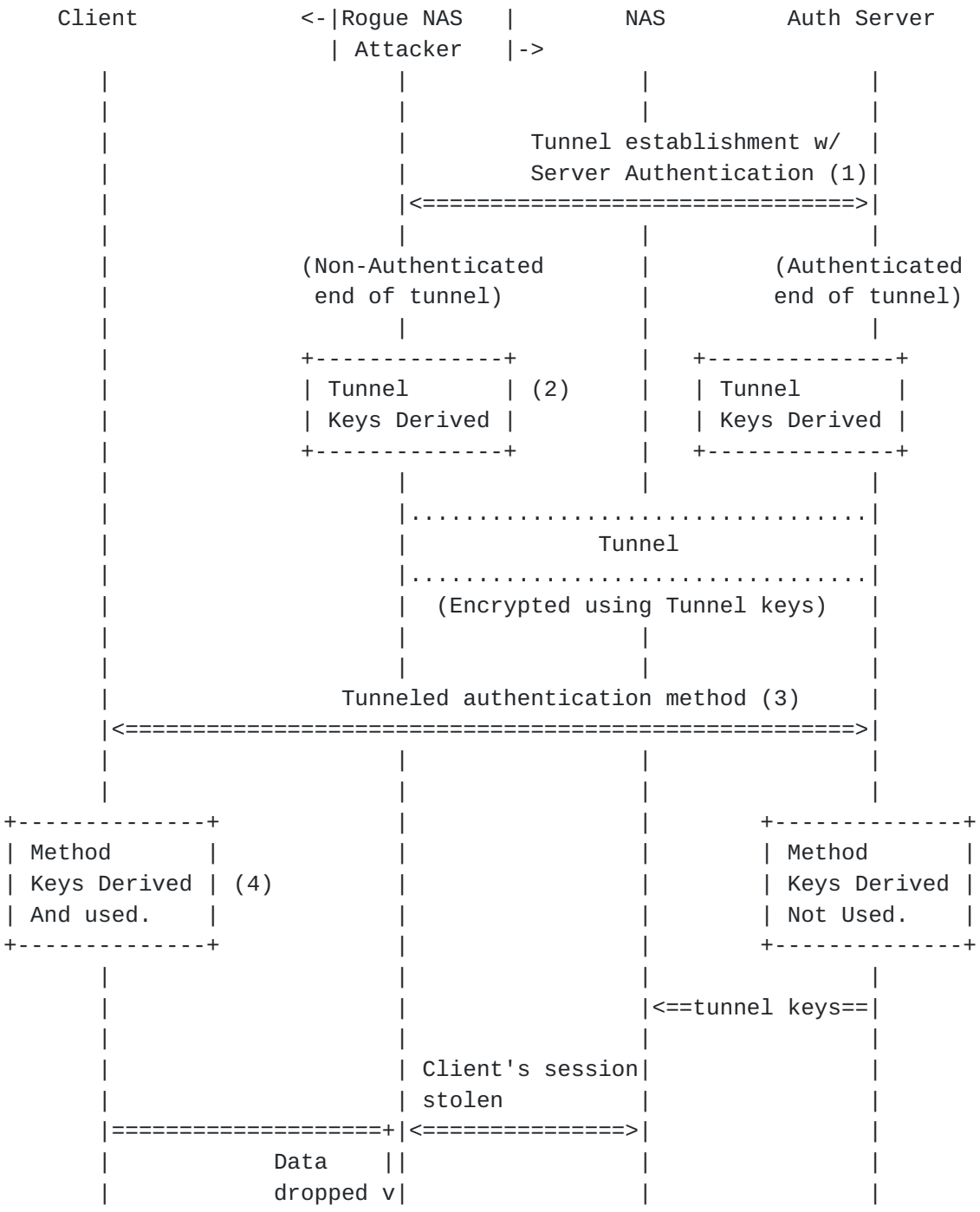


Figure 1 - Man-in-the-middle attack on a one-way authenticated tunnel

PPTP [RFC2637]. For the purposes of the attack, it is necessary that the client be induced to authenticate to the attacker using an authentication mechanism permitted within the tunnel. It is also necessary that the credentials within the client protocol and the tunneled authentication protocol be the same, so that the authentication mechanism remains valid when encapsulated within the tunnel.

Figure 1 illustrates the attack, for the case where the attacker acts as a rogue NAS or Access Point. In step 1, the attacker creates a tunnel with the authentication server. This can occur directly in [PIC] or [XAUTH] or through a NAS using EAP [RFC2284]. In step 2, tunnel keys are derived, using server-only authentication via protocols such as ISAKMP [RFC2408], IKE [RFC2409] with group pre-shared keys or TLS [RFC2246]. Since the tunnel is between the attacker and the server, both the server and attacker possess the keys.

In the third step, the client connects to the rogue NAS or AP, and the attacker tunnels the authentication method between the client and server. The tunneled authentication method may or may not derive keys, but if it does, then in the fourth step, the method keys are derived on the client and the server. Since the attacker does not obtain the method keys, it is not able to decrypt traffic sent between client and server. However, while the client may use the method keys, the server will typically use only tunnel keys, which have been obtained by the attacker.

In the last step, the attacker obtains access to the server, using the successfully tunneled authentication and the tunnel keys. The attacker does not have access to client data, since it is encrypted with the method keys. As a result, it will typically discard the data sent by the client, who will eventually disconnect due to a lack of response. Since the attacker has accomplished its mission, continued interaction with the client is not necessary unless reauthentication is required.

The attack described is also possible, if the tunnel server is not the final authentication server. Now in that case the vulnerability is even more serious because the inner method could be running to an untrusted network, and it assumes, that the tunnel server and the final authentication server have a trust relationship which it cannot validate. EAPTTLS typically suggests this model and care must be taken while deploying this intermediate tunnel termination model to prevent such man-in-the-middle attacks.

2.3. Conditions for the Attack

The following are some causal conditions that are necessary for this attack to be possible. We label these conditions using a prefix CC and refer to them during the solution description.

[CC-A] Client and Authentication server policy allowing client credentials to be used both within one-way server-authenticated tunnels and outside them.

ex: An authentication server supports EAP-MD5 using EAP-TTLS and also supports the same method with the same credentials when used without EAP-TTLS. This can occur when clients are being upgraded at different times or when upgrades are not universal. If the identities used were different in each case, then the server would be able to detect fraudulent use and prevent the attack.

[CC-B] The ability for a man-in-the-middle to pose as a legitimate client to the authentication server as well as a legitimate authenticator to the client and perform both functions simultaneously.

ex: A 802.11 WLAN client that also has the capabilities to function as an AP and functions as one entity with malicious intent. This is relatively easy to accomplish given the low cost of equipment and tools. This may be relatively more difficult to accomplish on dial-up RAS servers.

[CC-C] The inability of the authentication server to validate that the client authentication occurring inside a tunnel originated at the same endpoint as the tunnel itself.

ex: When an EAP method runs inside of EAP-TTLS or PEAP, the tunnel endpoint that is not authenticated really does not prove that it originated the inner EAP conversation as link is protected only by the tunnel keys.

[CC-D] The data link being authenticated is always confidentiality- and/or integrity-protected using tunnel keys instead of the keys derived from the client method running inside the tunnel.

ex: The keys from any EAP method running inside of EAP-TTLS or PEAP are not used today.

3. Potential Solutions

This section describes potential solutions to the man-in-the-middle attacks prevously described. This includes a discussion of solution criteria, concepts, mechanisms, approaches and their scope. Some of the limitations of these solutions are also captured.

3.1. Solution Criteria

The following are some criteria for a potential solution.

Backwards compatibility
: A solution MUST NOT require modification of existing authentication methods. Since tunneling is used in order to prolong the life of legacy authentication techniques, requiring replacement of existing methods across the board is likely to be unacceptable.

Simplicity
: A solution SHOULD add minimal round trips to the authentication exchange and be modest in its computational cost.

Protocol compatibility
: Given that tunneling techniques are used with well-established security protocols such as IKE [RFC2409], ISAKMP [RFC2408], TLS [RFC2246], and RADIUS [RFC2865], a solution MUST NOT require changes to these protocols.

Forward evolution
: The solution SHOULD be compatible with authentication methods supporting mutual authentication and key derivation. It is acceptable if the solution cannot be uniformly applied for providing security for tunneling of one-way authentication methods that do not derive keys, such as CHAP, EAP-MD5,OTP, Generic Token Card, or SecurID. As described earlier, these methods are already vulnerable to connection hijacking.

Architectural compatibility
: Solutions MUST NOT require fundamental architectural changes to established technologies such as network access authentication. Since these technologies are already widely deployed, such changes would be likely to be unacceptable.

3.2. Solution Concepts

There are several concepts at our disposal for mitigating this attack. As the root problem was missing proof that both peers actually performed all the individual methods in the compound authentication, one solution is to provide just that. The other solutions include restrictions on the implementation of the compound authentication methods so as to avoid the causal conditions described in section 2.3 (CC-A...CC-D).

So here we list all the concepts and some of the limitations of each.

[S1] Provide proof that the unauthenticated tunnel endpoint is a real peer and not the man-in-the-middle attacker using cryptographic binding. This prevents condition CC-C.

This solution works for all key-deriving methods used inside a tunnel. And this is the primary solution described in this document. This will not work for non-key-deriving methods without breaking at least one of our solution criteria. So we only consider this solution for key-deriving methods.

[S2] Guarantee that the same peer credential is never usable inside and outside a tunnel using server and client policy. This prevents condition CC-A.

This solution actually works for all methods, but is sometimes hard to deploy, due to legacy deployments and since clients and servers need to be synchronized for proper policy enforcement. An additional problem with this solution is the manageability issues due to the multiple credentials that have to be managed by the same client and server.

[S3] Prevent an Access Point (or Base station)/Client to be ever manipulated to perform both functions and become an attacker. This prevents condition CC-B.

This solution is possible to accomplish in a very limited context, like under the watchful eyes of law enforcement. But due to the wide availability of hacking tools, it is extremely expensive to implement in the real world.

[S4] Provide a mechanism for all method secrets in a compound authentication to be used for deriving sessions keys to protect subsequent communication. This prevents condition CC-D.

This solution also, just like [S1], will only work for only key-deriving methods. For non-key-deriving methods, this won't work as the only choice is to use tunnel keys to meet the solution criteria.

[S5] When using the same credentials inside and outside the tunnel, modify the authentication method inside the tunnel for the server/client to distinguish the authentication inside the tunnel. This also prevents condition CC-C.

This solution partially violates the backwards compatibility solution criteria, but is feasible where such a requirement does not exist. Though this is not a general solution, in situations, where the interest is in preserving the legacy credentials as opposed to preserving the legacy authentication method, this solution may make sense.

For realizing concepts S1 and S4 we use cryptographic means. S2 is realizable just using policy techniques on the client and server ends. The solution S5 uses modified legacy authentication methods when they are used within the tunnel. Thus they require specification of tunneled variations of legacy authentication methods to distinguish them from the legacy methods. Thus we have solutions that can prevent all the causal conditions except CC-B, as solution S3 is not viable.

3.3. Solution Mechanisms

The solution mechanisms include a mix of policy based techniques and using cryptographic means. Assuming that compound methods are of interest, the policy based techniques have significant relevance for non-key deriving (possibly legacy) authentication methods where cryptographic binding is not necessarily practical. These involve the following mechanisms:

[1] Providing separate credentials for a user identity when using the same authentication method inside and outside tunnels.

This enables the server to know when a credential that is not intended for use inside a tunnel is being used. This maybe done by an attacker and appropriate action can be taken. Though this is restrictive and worsens usability, it maybe easy to deploy.

[2] Enforce client and server policy to always use authentication methods inside tunnels.

This could have more significant deployment issues, but is a better option if possible, as it enables the benefits of compound methods to be realized more effectively.

[3] Provide a modified authentication method inside the tunnel for the same credentials associated with a single user identity.

This is not a general solution, but can be applied in situations where the authentication method is modifiable for use inside the tunnel and there is significant interest in preserving the credential being used for user convenience. The mechanisms include some form of protected signal to the server from the client indicating the authentication to be running inside a tunnel. It also includes client and server policy of always expecting the modified method to be used only inside a tunnel and not outside it. It is also important that the man-in-the-middle attacker should not be able to accomplish the same modification to the method that is normally run outside the tunnel and thus spoof the server. It should be noted as mentioned earlier this solution violates the backward compatibility criterion in section 3.1, but can be used where such a criterion is not relevant.

The mechanisms to provide cryptographic proof involve combining some knowledge derived from each authentication method involved in a compound authentication to prove that both parties are the real peers. This knowledge may be in the form of keying material available to both parties. This information can be used as part of a two-stage solution.

[Stage 1] Binding Phase Exchange with Compound Keyed MACs

Here we execute an additional 2-way handshake to the tunneling protocol or base protocol, we call it the Binding Phase Exchange÷. This phase is entered only if the server knows that all the individual authentication methods inside the tunnel have completed. (There maybe some situations where binding maybe carried out after each inner method completes. However that is beyond our scope at the moment.)

In case of the tunnel server endpoint not being the final authentication server, it has possession of the inner method keys if they are available. The keys from all the inner methods and the tunnel keys are used to compute keyed compound MACs as described in section 4.2. The validation of the compound MACs protects against the man-in-the-middle attack, as the attacker is unable to get any of the inner method keys. Here the server sends a Binding request B1 with a B1_MAC and the client validates it. If the message is valid the client responds with the B2 message that also has a B2_MAC associated with it. If the server successfully validate the B2_MAC, then it can be certain that there was no man-in-the-middle.

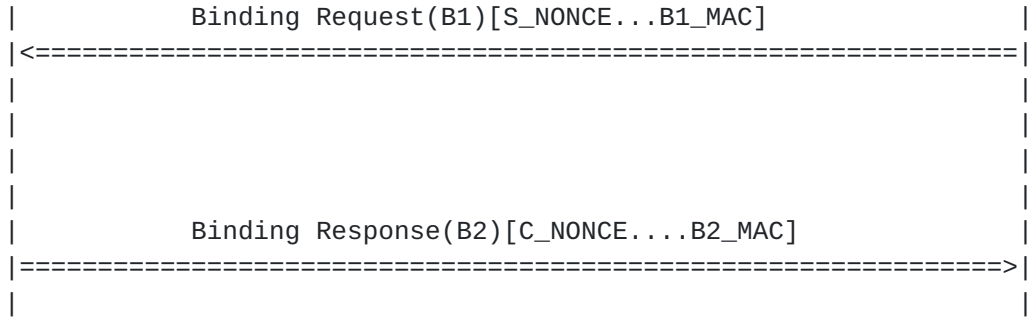


Figure 2: Binding Phase Message Flows

For clarity, we name all the keys and nonce values used in the two stages. The input keys for the binding phase are TSK and the several ISKs, one from each inner method. These keys should be available before the binding phase exchange can be carried out. The details of the derivation of these keys are in section 4.2

TSK Tunnel Session Key. This is part of the base keying material available from the tunneling protocol. It should be at least 256 bits and derived ensuring key separation for binding, non-binding, transmit and receive encryption and integrity purposes.

ISK Inner Session Key for the inner authentication method running inside the tunnel. Each inner method that derives keys will have an non-zero ISK. It could be of variable length depending on the particular method. But should be at least 64 bits. The key derivation process in section 4.2 is capable of handling variable length ISKs as they are input as strings to the PRFs.

S_NONCE 256 bit random number used for computing the Compound keyed MAC values for the B1 message (B1_MAC) and the B2 message (B2_MAC).

C_NONCE 256 bit random number used for computing the Compound keyed MAC value for the B2 message (B2_MAC).

The output keys generated as part of the cryptographic binding process are the MAC keys CMK_B1 and CMK_B2 and the CSK. CMK_B1 and CMK_B2 are needed for computing the B1_MAC and B2_MAC repectively in stage 1. The CSk is needed for stage 2.

CMK_B1 Compound MAC key derived for the B1 message MAC (B1_MAC) computation. It is 128 bits in length. This is derived with the S_NONCE which is a 256 bit random number.

CMK_B2 Compound MAC key derived for the B2 message MAC (B2_MAC) computation. It is 128 bits in length. It is derived using two nonces the S_NONCE and the C_NONCE both of which are 256 bit random numbers and also additional parameters that representative of all the inner methods.

CSK Compound Session Key, which is the bound key generated for use as the base keying material for the link layer. It is 192 bytes in length. The different portions of the CSK are partitioned into 32 byte chunks and specified for different uses including export as the session key for subsequent protocol layers.

Here we descibe the stage 1 message exchange related aspects.

B1. Binding Request

This message is a request sent from the tunnel server to the tunnel client to perform binding. Among other parameters it includes a 256 bit random number, ie. the S_NONCE and a compound keyed MAC, B1_MAC computed by the server over the entire B1 message. There is a compound MAC server key CMK_B1, derived for computing the B1_MAC on the server and another equivalent one derived on the client for validating it after receiving the message. The S_NONCE is used in the derivation of this CMK_B1 in addition to the bound keying material (ISK1...ISKn) from all the inner methods inside the tunnel and the tunnel keys (TSK). So if the client and server have all the keying material from all the methods, the B1_MAC validation on the client should succeed and the response message B2 will be sent.

In the case of invalid B1_MAC, the client need not send a response and can disconnect as a potential man-in-the-middle could be present and be modifying packets. Also this message as it will be enacapsulated in the underlying protocol, the retry policies on the server can be specified as part of that protocol which initiates binding.

B2. Binding Response

This is only sent as a response to B1 and includes a B2_MAC and also additional parameters including a 256 bit random number called the C_NONCE. The B2_MAC is a compound keyed MAC calculated over the entire B2 message. The derivation of B2_MAC is explained in section 4.2. A client MAC key is derived for computing this B2_MAC. To prevent against replay attacks, the CMK_B2 is derived using the S_NONCE and the C_NONCE and can only be derived after the B1 message is received.

If the B1 message has an invalid B1_MAC, this CMK_B2 MAC key derivation is not possible and the B2 message cannot be safely constructed. So no response is sent. The server that does not receive a B2 response can timeout and disconnect or perform a retry. It is expected to use a new S_NONCE for every retry.

If the Binding Phase successfully completes with the server validating the B2 message, then the compound authentication is complete. More details this binding phase exchange is described in section 4.1.
The CMKs should provide enough strength for the MACs so that it cannot be broken in the time taken to authenticate. ie. seconds. Its important to remember that the binding phase exchange when performed as the final step to to complete authentication should include the protected success/failure indication using the Result TLV. This is described in section 4.1. Also other meta information about the methods exchanged can be used for policy validation or fraud detection purposes. However the protection from the attack is mainly from the MACs and not from the other parameters.

[Stage 2] Compound Session Keys Generation

In stage 2 we generate combined keys that are session keys for link layer confidentiality and integrity protection needs.These are computed using from all the inner method keys if they are available and the tunnel keys. The resultant keys called Compound Session Keys (CSKs) are provided to the link layer for ciphering and integrity protection. These keys provide the assurance that all packets are being exchanged between the real peers and no man-in-the-middle is actually decrypting the conversation. The Compound Session Key derivation is specified in section 4.2.

Though from our analysis, compound MACs used in the Stage 1 Binding Phase Exchange provide the necessary protection, we argue that for additional protection one could also use Stage 2. The main reason for Stage 2 is to allow each tunnel packet privacy protection to also be cryptographically bound to the inner methods it carries.

Performing Stage 1 prevents a man-in-the-middle from gaining authenticated access. It also prevents false authenticated states which could result in a Denial-of-Service attack that could result if only Stage 2 is done.

Only implementing Stage 2 does not require additional round trips, but it enables the man-in-the-middle to authenticate, although not to obtain keys used for subsequent link-level authentication and encryption of the data. This implies that the client will only discover an attack when it discovers that it is unable to decipher the incoming data stream. As a result, Stage 2 is probably not sufficient by itself.

3.4. Thwarting the attack

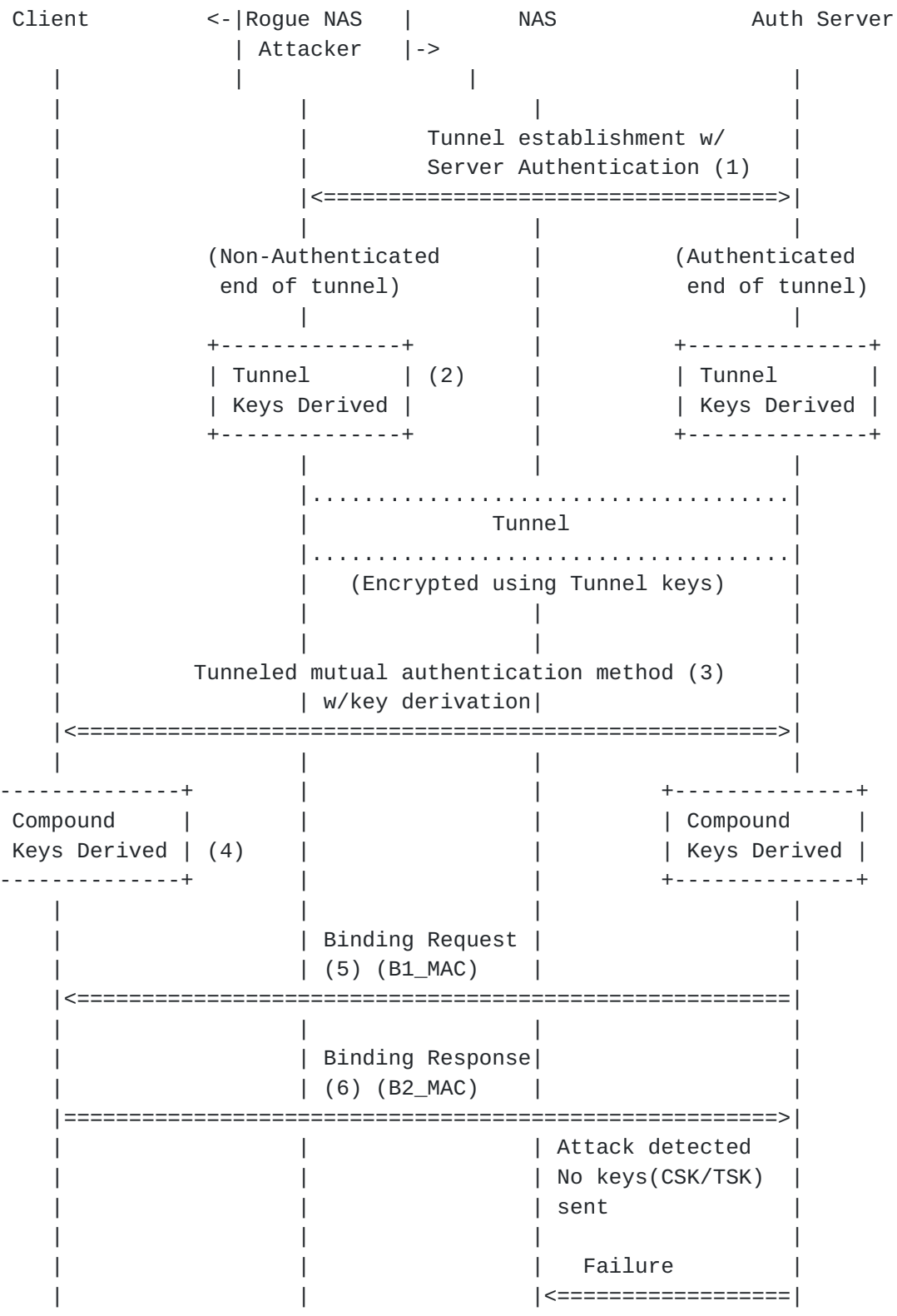


Figure 3 - Man-in-the-middle attack thwarted by compound MAC and compound session keys

Figure 3 shows how by adding the Binding Phase Protocol Exchange the attack is prevented as the man-in-the-middle attacker cannot provide the correct B1_MAC that the client will validate against in the B1 message in step (5). Also the B2 message cannot be successfully sent by the attacker and the server will fail a false B2 message, that does not possess a valid B2_MAC. Hence the compound session keys (if stage 2 is used) or tunnel keys are not sent to the Authenticator by the server. Thus the man-in-the-middle attack is prevented.

3.5. Solution approaches

Stages 1 or 2 can be implemented in different ways:

EAP
: In this approach, the binding phase exchange of a compound keyed MACs is supported within EAP, by implementing the exchange as a new EAP method occuring after authentication is complete. Tunnel keys are provided by the tunneling protocol to the EAP layer in order to enable computation of the compound keyed MACs and compound session keys. Since existing EAP implementations already enable EAP methods to provide keying material to the EAP layer, such an interface typically already exists. This approach is general in that it applies to any tunneling technique.

Tunneling method
: In this approach, the tunneling method acquires keying material derived by the underlying authentication methods, in order to enable computation of the compound keyed MACs and compound session keys. Since existing tunneling techniques typically do not provide an interface for receiving keying material from authentication methods, this approach will typically require some re-architecture of existing implementations. It also has the disadvantage of requiring changes to each tunneling method, and as a result is not as general as an EAP-based solution. Given the prevalence of the attacks described within this document, it would represent a considerable burden on the security community to review changes to each individual tunneling approach. However, such an approach may be able to take advantage of properties of the underlying tunnel technology, such as the ability to have more than one packet in transit at a time.

EAP methods
: In this approach, keys derived from previous EAP methods are incorporated into the authentication calculations of subsequent methods. Since existing interfaces only support the export of keys by EAP methods, not importation, some

rearchitecture is required in this approach. In addition, this approach requires modification of existing EAP methods, which will create deployment barriers. However, this approach may be applied even to methods which support only one-way authentication and do not generate keys. As mentioned

earlier the use of signals to indicate the explicit intention to run inside a tunnel can be another approach to mitigating the problem. However the signals have to be protected from spoofing and that is not easy without some keying material being available.

Based on the pros and cons of each approach, we recommend a solution that applies to all EAP methods either in the Tunneling method or in the EAP base protocol.

3.6. Solution scope

The policy based mechanisms we described in the previous section will work for any tunneling protocol, but it can be a bit restrictive.

The cryptographic mechanisms work for all key deriving methods and can be implemented in the EAP base protocol or the tunneling protocol as extensions. This is possible because all EAP methods deriving keys will be able to provide keying material required for the binding process. Also the PRF that is used for key derivation in the binding phase can be selected by the tunneling protocol fairly independently. The cryptographic binding process will also provide exportable keying material through the CSKs for subsequent protocol layers or application use in a transparent manner.

When a mix of key deriving and non-key deriving methods are used inside the tunnel the nature of protection largely relies on the key deriving methods. If the non-key deriving methods are not properly managed through policy mechanisms as described earlier and if they play a significant role in the compound authentication success/failure condition then the vulnerability still exists. But the attacker may not be able to take control of the network access session.

However it important to note that, downgrade attacks are possible with modified EAP or tunneling protocols as attackers can always try to get the server to connect to a client tunnel that does not support cryptographic binding or the policy mechanisms. So appropriate server and client policies are needed to either not support older versions of the protocols that do not have the enhancements or have counter measures in place to deal with potential fraud.

4. Reference Solution in Tunneling Protocol

Here we describe a reference solution using cryptographic binding that can be implemented in a tunneled EAP-based authentication protocol like PEAP to thwart man-in-the-middle attacks. This involves incorporating a binding phase handshake after all the inner EAP methods inside the tunnel complete, to provide cryptographic proof that the peers indeed took part in all the authentication methods. We use compound keyed MACs for these messages using keys derived from the individual methods combined with the tunnel keys. We also derive cryptographically bound session keys that can be exported by the tunnel method as Master Session Keys (MSK) for ciphering at the link layer. Though we limit our discussion to PEAP, the principles used here can be applied to other compound authentication protocols as well. Here for simplicity though we assume the tunnel server endpoint is the final authentication server, the solution works for either case. The only requirement is that tunnel server get all the inner method session keys if they are derived, from the final authentication server.

4.1 Binding Phase Precepts

The cryptographic binding MUST take place between the tunnel client and the tunnel server. The tunnel client and server MUST have access to both tunnel method and inner authentication method Master Session Keys (MSKs). (However an attacking tunnel client will not be able to meet this condition.) The tunnel method and inner method MSKs MUST be fresh to ensure liveness of the binding phase handshake. Though the binding phase is intended to be carried out after all the inner authentication methods complete, some tunneling protocols MAY opt to do it after every inner authentication method completes. This requires the tunnel method to distinguish between the final binding and the intermediate ones, using the RESULT TLV described in section 4.4.

The PRF selected for generating compound keyed MACs and the compound session keys SHOULD be based on what is supported in the tunneling protocol. This enables easier implementation of the binding phase handshake. For the case of PEAP, as TLS is used, the P_SHA-1 PRF is that is part of TLS is selected.

The tunnel server MUST do proper version negotiation upfront to prevent downgrade attacks. If the tunnel server wishes cryptographic binding to be mandatory, then using policy it MUST ensure that it only permits connections to a tunnel client version that supports cryptographic binding. At the same time, it MUST ensure that for non-key deriving methods that MAY be used, separate credentials are provided for use inside and outside the tunnel.

In the case where the tunnel server is not the final authentication server, this binding phase MUST be started only after the tunnel server gets the inner method keys from the final authentication server. This also assumes that the tunnel server and authentication server have a security association that enables them to share these keys and the tunnel server is capable of holding the authentication state till the binding phase is complete.

4.2 Binding Phase Handshake

This 2-way handshake for cryptographic binding SHOULD be added as a tunnel protocol extension. This handshake is started when the server makes a determination that the last inner authentication methods have completed. The highest version number supported is needed in both the Binding Request (B1) and Binding Response (B2) messages to provide additional protection through policy enforcement.

The version number will prevent roll back attacks between tunnel protocol versions that support cryptographic binding, but not for other implementations that do not support it.

The server having possession of all the inner method keys and also the tunnel keys, and including a 256 bit server Nonce (S_NONCE), computes a compound MAC server key (CMK_B1). This and other cryptographic derivations are described in the next section. Then the server sends a Binding Request Message (B1) that has a server-computed compound keyed MAC (B1_MAC) associated with it. This message body includes the version number of the protocol extension, a 256 bit S_NONCE, a RESULT_TLV that provides protected success/failure indication. The B1_MAC is computed over this entire message body by setting the MAC field bits to zero.

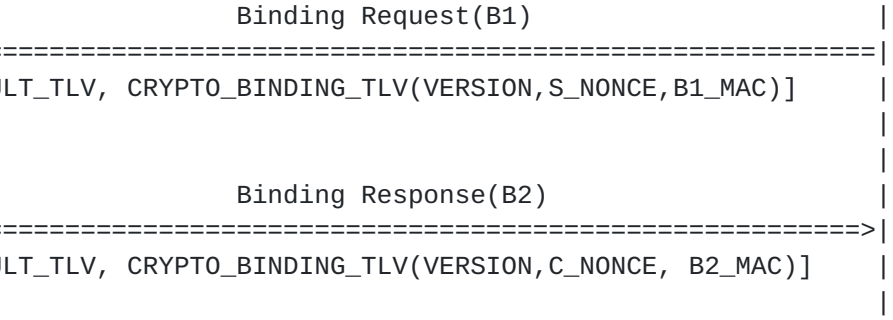


Figure 4: Binding Phase Message Flows (Success Case)

The client on receiving the B1 message will first compute its own CMK_B1 using the S_NONCE and its own knowledge of all the inner method keys and the tunnel keys. Then it validates the B1_MAC sent from the server by recomputing it over the B1 message body with MAC bits set to zero. If the B1 message is valid, it responds with a Binding Response (B2) message that includes a compound keyed MAC (B2_MAC). Before it responds, it first computes a MAC key (CMK_B2), for computing the B2_MAC, using the tunnel keys, all the inner method keys, the S_NONCE received from the server and a locally derived 256 bit client Nonce (C_NONCE). The B2 message body is similar to the B1 message, and includes a version number of the protocol extension, a 256 bit C_NONCE, a RESULT TLV that provides acknowledgment for the protected success/failure indication.

A client side MAC (B2_MAC) is computed for this message body using the CMK_B2, with the MAC bits set to zero, and is included in the B2 message. Figure 4 shows both the messages in the success case.

There are several conditions that can cause invalid B1 or B2 messages, they include the following:

- Invalid B1_MAC or B2_MAC
- Incompatible protocol version
- Invalid RESULT_TLV

A man-in-the-middle attack can cause these conditions to occur. More details on detecting these conditions are described as part of the message formats in section 4.3. Here we describe how they are handled as the failure authentication cases.

[1] Client receiving an invalid B1 message

If the client finds the Binding Request (B1) message to be invalid, it does not send any response. It can make a decision to drop the connection at this time, as it is not certain, that its talking to the right server. Any other Binding Requests (B1) messages if they arrive subsequently, and if they are valid, are responded to using proper Binding Response (B2) messages. This is to allow for packet loss during the server retry time period. The retry of packets is handled by the EAP layer; and is transparent to the EAP methods. The packets should not be different because the media is unreliable.

[2] Client never receiving a B1 message

In this case, client having waited for sufficient time (retry time period) will time out and can drop the connection.

[3] Server receiving an invalid B2 message

If the server receives an invalid Binding Response (B2) message, it can decide to drop the connection, as its not certain that it is talking to the right client. Any further Binding Responses are ignored, even if the server had sent out more than one B1 messages.

[4] Server never receiving a B2 message

In this case, the server will time out and can do a retry, up to 3 times with new B1 messages. It must use a new S_NONCE every time it sends a new message.

4.3 Compound MAC and Session Key derivation

The input for the cryptographic binding we use includes the 128-octet Tunnel Session Key (TSK), the 64-octet Tunnel Initialization Vector (TIV), and the inner method provided session keys, ISK1, ISK2,..ISKN, that belong to the N authentication methods used inside the tunnel. These keys may all be of varying sizes, so a known size in multiples of 32 bits between 64 and 256 bits is chosen of each of the methods based on available keying material.

The Compound MAC for the client (the B2_MAC) and the server (the B1_MAC) are derived from two different MAC keys called CMK_B2 and CMK_B1 respectively. A compound session key (CSK) is also derived on both the client and server for cryptographic purposes. If the binding phase is implemented, that alone prevents the man-in-the-middle attacks, so the CSKs are really not needed and the tunnel session keys can still be used for ciphering purposes, just as the TSK is in a normal EAP-TLS session.

The CMK_B1, CMK_B2 and CSK are derived as follows for the PEAP protocol. PEAP tunnel session key (TSK); TSK is calculated using the EAP-TLS algorithm (RFC2716 section 3.5), and is 128 octets. This includes the first 64 octets of MSK (Master Session Key) and next 64 octets of EMSK (Extended Master Session Key).

ISK1..ISKn are the EAP inner session keys (MSK) obtained from methods 1 to n.

In some cases, ISKi, for some i, may be the null string (""), when the corresponding method does not derive keying material.

We use the P_SHA-1 PRF specified in the TLS specification [RFC2246] for PEAP, though this PRF selection decision can be made independently for each tunneling protocol.

Compound MAC Key derivation algorithm:

Take the second 256 bits (32 octets) of MSK portion of the TSK.
output)

IPMK0 = TSK

for j = 1 to n do

    IPMKj = PRF(IPMK(j-1),"Intermediate PEAP MAC key", ISKj);

Where j denotes the last inner method that runs inside the tunnel. Each

IPMKj output is 32 octets. IPMKn is the intermediate combined key used to derive compound session and compound MAC keys.

The Compound MAC Key for the server (the B1_MAC) is derived CMK_B1

CMK_B1 = P_SHA1(IPMKn,"PEAP Server B1 MAC key" | S_NONCE);

The Compound MAC Key for the client (the B2_MAC) is derived from MAC key called CMK B2.

CMK_B2 = P_SHA1(IPMKn,"PEAP Client B2 MAC key" | C_NONCE | S_NONCE);

The compound MAC keys (CMK_B1 and CMK_B2) are each 20 octets long.

("|" denotes concatenation)

The pseudorandom function PRF(k,label,string) is computed as P_SHA-1(k,label|string) (or substitute, if necessary, any other PRF that produces output of sufficient length). Here the outputs are taken to as many bits as are necessary (typically 256 bits for a key).

Compound Session Key derivation:

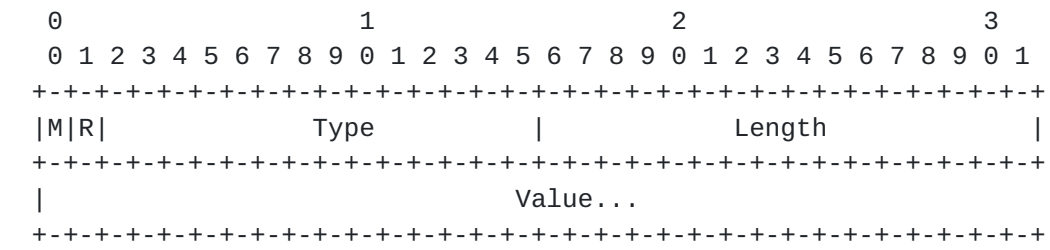
The following function provides the necessary keying material.

```
CSK = PRF(IPMKn, "PEAP compound session key",
          C_NONCE|S_NONCE, OutputLength);
```

where the PRF is the same one used for CMK derivation. The output is taken to 128 octets. The first 64 octets are taken and the MSK and the second 64 octets are taken as the EMSK. The MSK and EMSK are described in [RFC2284bis].

4.4 Binding Message Formats

The Binding Messages are represented using TLVs defined below. The generic TLV format is defined in [PEAP] in section 3.1 and duplicated below.



M

0 - Non-mandatory TLV
1 - Mandatory TLV

R

Reserved, set to zero (0)

Type

A 14-bit field, denoting the attribute type. Allocated AVP Types include:
0 - Reserved
1 - Reserved
2 - Reserved
3 - Result TLV (Acknowledged Result)

Length

The length of the Value field in octets.

Value

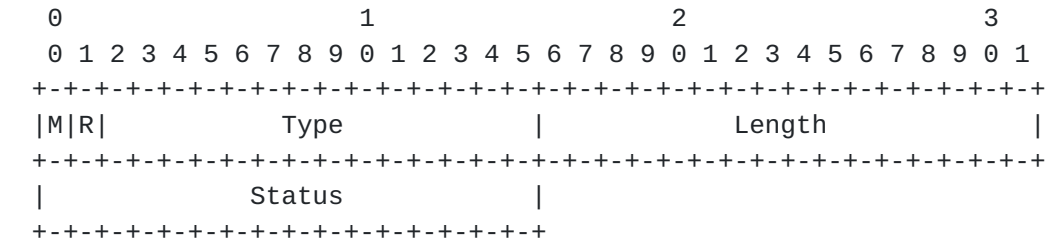
The value of the object.

We also use the Result TLV that also defined in [PEAP] to indicate the final success acknowledgement. The following is the duplicated definition of the Result TLV.

Result TLV:

This is defined in [PEAP] and is duplicated below.

The Result TLV provides support for acknowledged Success and Failure messages within EAP. It is defined as follows:



M

1 bit. Value = 1 - Mandatory TLV

R

1 bit. Reserved, set to zero (0)

TLV Type

14 bits. Value = 3 - Success/Failure

Length

2 octets

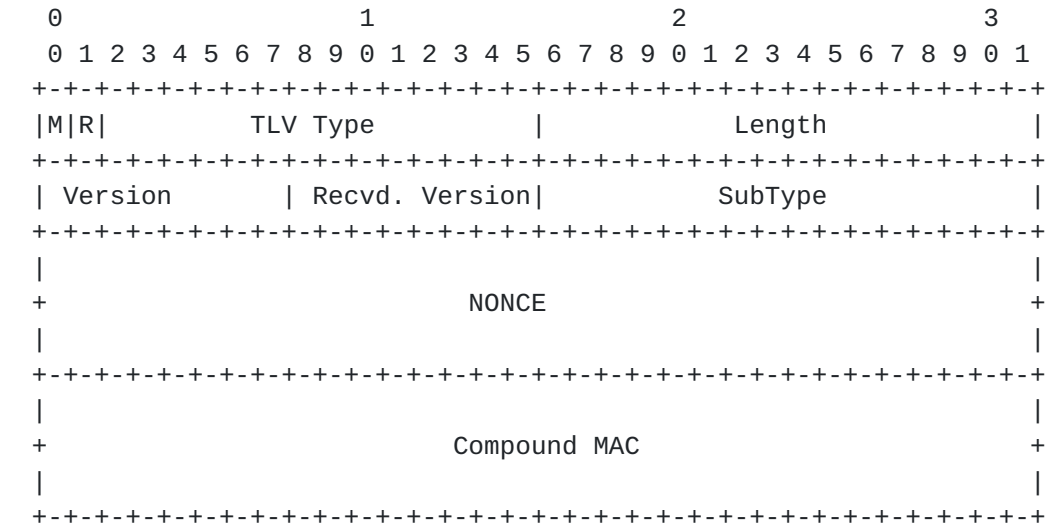
Status

The status field is two octets. Values include:
1 - Success
2 - Failure

The Cryptographic Binding is performed using the Binding TLV. It is described below. Both the Binding Request (B1) and Binding Response (B2) use the same packet format. However the SubType indicates whether it is B1 or B2. The Binding TLV and other TLVs are carried in the EAP-TLV packet defined in [PEAP]. The Binding TLV can be used to perform cryptographic Binding after each EAP method is complete. If this is the final method, then the EAP-TLV packet must also include the Result TLV along with the Binding TLV.

Binding TLV (Also called Tunnel Authenticity/Integrity Check TLV):

This message format is used for the Binding Request (B1) and also the Binding Response. This uses TLV type CRYPTO_BINDING_TLV. The format is as given below.



M

1 bit. Value = 1 - Mandatory TLV

R

1 bit. Reserved, set to zero (0)

TLV Type

14 bits. Value = CRYPTO_BINDING_TLV. See IANA Considerations.

Length

2 octets. Value = 52 (excludes the 16 bit EAP-TLV header)

Version

1 octet. Version of tunnel protocol extension for binding.
Initially set to 0.

Received Version

1 octet. PEAP version number received to prevent downgrade attacks.

SubType

2 octets.
0 - Binding Request
1 - Binding Response

Nonce

32 octets. 256 bit Random number that is never repeated and is used for compound MAC key derivation at each end. It is a S_NONCE for the B1 message and a C_NONCE for the B2 message.

MAC

16 octets (128 bits). This can be the Server MAC (B1_MAC) or the Client MAC (B2_MAC). It is computed over the entire Binding TLV packet using the HMAC-SHA1-128 that provides 128 bits of output using the CMK_B1 or CMK_B2 with the MAC field zeroed out.

4.5 IANA Considerations

IANA has assigned the EAP type number 33 for TLVs.

This protocol extension defines a new TLV types for cryptographic binding that are defined below.

CRYPTO_BINDING_TLV 5

It also uses the Result TLV (RESULT_TLV) defined in the draft[PEAP]

4.6 Security Considerations

This section describes the limitations of the solutions provided in this document and also provides guidelines for ensuring proper implementation.

For our cryptographic binding based solution to work, the tunnel session keys (TSKs) and all the inner methods keys (ISKs) need to be guaranteed to be fresh and derived for new for every compound authentication session.

For the case where the tunnel server is not the final authentication server, the inner method keys must be securely delivered to the tunnel server from the final authentication server where they are derived.

The key derivation we use always ensures the usage of tunnel keys and therefore prevents a weak inner method key to be used by itself, which could have opened up some dictionary attack possibilities on the compound MAC based on weak inner method keys alone.

5. Normative references

[TLS] [RFC2246]

[EAP-MD5] [RFC2284]

[ISAKMP] [RFC2408]

[IKE] [RFC2409]

[PEAPV0] Kamath, V., Palekar, A., Wodrich,M., "Microsoft's PEAP version 0 (Implementation in Windows XP SP1)", October 2002 (work in progress)

[RFC1661] Simpson, W., "The Point-to-Point Protocol (PPP)", STD 51, RFC 1661, July 1994.

[RFC1938] Haller, N. and C. Metz, "A One-Time Password System", RFC 1938, May 1996.

[RFC1994] Simpson, W., "PPP Challenge Handshake Authentication Protocol (CHAP)", RFC 1994, August 1996.

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

[RFC2284] Blunk, L., Vollbrecht, J., "PPP Extensible Authentication Protocol (EAP)", RFC 2284, March 1998.

[RFC2865] Rigney, C., Rubens, A., Simpson, W., Willens, S., "Remote Authentication Dial In User Service (RADIUS)", RFC 2865, June 2000.

[RFC3193] Patel, B., et al., "Securing L2TP Using IPsec", RFC 3193, November 2001.

[EAPTTLS] Funk, P., Blake-Wilson, S., "EAP Tunneled TLS Authentication Protocol", Internet draft (work in progress), February 2002.

[DHCPIpsec] Patel, B., et al., "DHCPv4 Configuration of IPsec Tunnel Mode", Internet draft (work in progress), draft-ietf-ipsec-dhcp-13.txt, June 2001.

[PEAP] Palekar et al., "Protected EAP Protocol (PEAP)", Internet draft (work in progress), draft-josefsson-pppext-eap-tls-eap-07.txt, October 2003.

[PIC] Sheffer, Y., Krawczyk, H., Aboba, B., "PIC, A Pre-IKE Credential Provisioning Protocol", Internet draft (work in progress), draft-ietf-ipsra-pic-06.txt, September 2002.

[IPSRAREQ] Kelly, S., Ramamoorthi, S., "Requirements for IPsec Remote Access Scenarios", Internet draft (work in progress), draft-ietf-ipsra- reqmts-05.txt, September 2002.

[GETCERT] Bellovin, S., and Moskowitz, B., "Client Certificate and Key Retrieval for IKE", Internet draft (work in progress), draft-bellovin-ipsra-getcert-00.txt, June 2000.

[SECURID] Josefsson, S., "The EAP SecurID(r) Mechanism", Internet draft (work in progress), draft-josefsson-eap-securid-01.txt, February 2002.

[PANATLS] Ohba, Y., et al., "PANA over TLS", Internet draft (work in progress), draft-ohba-pana-potls-00.txt, September 2002.

[XAUTH] Pereira, R., Beaulieu, S., "Extended Authentication within ISAKMP/Oakley", Internet-draft (work in progress), draft-beaulieu-ike-xauth-02.txt, September, 2000.

[IEEE802] IEEE Standards for Local and Metropolitan Area Networks: Overview and Architecture, ANSI/IEEE Std 802, 1990.

[IEEE8021X] IEEE Standards for Local and Metropolitan Area Networks: Port based Network Access Control, IEEE Std 802.1X-2001, June 2001.

[IEEE80211] Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Specific Requirements Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications, IEEE Std. 802.11-1997, 1997.

6. Informative references

[RFC1510] Kohl, J., Neuman, C., "The Kerberos Network Authentication Service (V5)", RFC 1510, September 1993.

[RFC2246] Dierks, T. and C. Allen, "The TLS Protocol Version 1.0", RFC 2246, November 1998.

[RFC2486] Beadles, M., Aboba, B., "The Network Access Identifier", RFC 2486, January 1999.

[RFC2486bis] Blunk, L, Vollbrecht, J., Aboba, B., Carlson, J., Levkowetz, H., "Extensible Authentication Protocol", RFC 2486bis-06.txt (work in progress), September 2003.

[RFC2401] Atkinson, R., Kent, S., "Security Architecture for the Internet Protocol", RFC 2401, November 1998.

[RFC2402] Kent,S., Atkinson, R., "IP Authentication Header", RFC 2402, November 1998.

[RFC2406] Kent,S., Atkinson, R., "IP Encapsulating Security Payload (ESP)", RFC 2406, November 1998.

[RFC2407] Piper, D., "The Internet IP Security Domain of Interpretation of ISAKMP", RFC 2407, November 1998.

[RFC2408] Maughhan, D., Schertler, M., Schneider, M., and Turner, J., "Internet Security Association and Key Management Protocol (ISAKMP)", RFC 2408, November 1998.

[RFC2409] Harkins, D., Carrel, D., "The Internet Key Exchange (IKE)", RFC 2409, November 1998.

[RFC2412] Orman, H., "The Oakley Key Determination Protocol", RFC 2412, Nov. 1998.

[RFC2433] Zorn, G., Cobb, S., "Microsoft PPP CHAP Extensions", RFC 2433, October 1998.

[RFC2637] Hamzeh, K., et al., "Point-to-Point Tunneling Protocol (PPTP)", RFC 2637, July 1999.

[RFC2661] Townsley, W., Valencia, A., Rubens, A., Pall, G., Zorn, G., and Palter, B., "Layer Two Tunneling Protocol L2TP", RFC 2661, August 1999.

[RFC2716] Aboba, B., Simon, D.,"PPP EAP TLS Authentication Protocol", RFC 2716, October 1999.

[RFC2759] Zorn, G., "Microsoft PPP CHAP Extensions, Version 2", RFC 2759, January 2000.

[RFC2866] Rigney, C., "RADIUS Accounting", RFC 2866, June 2000.

[DECEPTION] Slatalla, M., and Quittner, J., "Masters of Deception." HarperCollins, New York, 1995.

[KRBATTACK] Wu, T., "A Real-World Analysis of Kerberos Password Security", Stanford University Computer Science Department, http://theory.stanford.edu/~tjw/krbpass.html

[KRBLIM] Bellovin, S.M., Merritt, M., "Limitations of the kerberos authentication system", Proceedings of the 1991 Winter USENIX Conference, pp. 253-267, 1991.

[KERB4WEAK] Dole, B., Lodin, S., and Spafford, E., "Misplaced trust: Kerberos 4 session keys", Proceedings of the Internet Society Network and Distributed System Security Symposium, pp. 60-70, March 1997.

[PPTPv1] Schneier, B, Mudge, "Cryptanalysis of Microsoft's Point-to- Point Tunneling Protocol", Proceedings of the 5th ACM Conference on Communications and Computer Security, ACM Press, November 1998.

[IEEE8023] ISO/IEC 8802-3 Information technology - Telecommunications and information exchange between systems - Local and metropolitan area networks - Common specifications - Part 3: Carrier Sense Multiple Access with Collision Detection (CSMA/CD) Access Method and Physical Layer Specifications, (also ANSI/IEEE Std 802.3-1996), 1996.

[MITM] Nokia Research Center, Man-in-the-middle attacks in tunneled authentication protocols, http://www.saunalahti.fi/~asokan/research/mitm.html, October 2002

[MITM1] N. Asokan, Valtteri Niemi and Kaisa Nyberg, Man-in-the-middle in tunneled authentication protocols, Technical Report 2002/163, IACR ePrint archive, October 2002. Available from http://eprint.iacr.org/2002/163

[SLA] Harkins, D., Piper, D., and Hoffman, P., "Secure Legacy Authentiation for IKEv2", draft-hoffman-sla-00.txt, December 2002 (work in progress)

[IKEv2] Kaufman, C. (Editor), "Internet Key Exchange (IKEv2), draft-ietf-ipsec-ikev2-05.txt, February2003 (work in progress)

Acknowledgments

We wish to thank N. Asokan, Valtteri Niemi, Kaisa Nyberg and Henry Haverinen of Nokia for initially making us aware of the problem [MITM].

Special thanks to Bernard Aboba, N.Asokan, Jesse Walker, Farid Adrangi, Nancy Cam-Winget, Joe Salowey, Jari Arkko, Mohan Parthasarathy, Jukka Ylitalo, Michael Richardson, DongGook Park and Hao Zhou for their valuable comments and suggestions.

Also additional thanks to Bernard Aboba for his expert advice and editorial assistance with the early versions of this document.

Authors' Addresses

Jose Puthenkulam
Intel Corporation

2111 NE 25th Avenue, JF2-58
Hillsboro, OR 97124

EMail: jose.p.puthenkulam@intel.com
Phone: +1 503 264 6121
Fax: +1 503 264 4230

Victor Lortz
Intel Corporation
2111 NE 25th Avenue, JF3-206
Hillsboro, OR 97124

EMail: victor.lortz@intel.com
Phone: +1 503 264 3253
Fax: +1 503 264 4230

Ashwin Palekar
Dan Simon
Microsoft Corporation
One Microsoft Way
Redmond, WA 98052

EMail: {ashwinp, dansimon}@microsoft.com
Phone: +1 425 882 8080
Fax: +1 425 936 7329

Intellectual Property Statement

Full Copyright Statement

EAP Issues

Open issues relating to EAP, including the issues described in this document, are tracked on the following web site:

http://www.drizzle.com/~aboba/EAP/eapissues.html

Expiration Date

This memo is filed as <draft-puthenkulam-eap-binding-04.txt>, and expires on April 26, 2004.