1. 引言

本文档定义了扩展认证协议，一个支持多路认证方法的认证框架。EAP 通常直接运行在数据链路层，例如 PPP 协议或者是 IEEE802,不需要 IP 地址。EAP 自身支持消除重复和转发，但是它依赖于底层正确的排序。EAP 本身不支持分片，然而特别的 EAP 方法可能支持这个。

EAP 架构的优势之一就是它的灵活性。EAP 是用来选择一个专门的认证方法，通常是在认证方在得到更多的信息以后决定使用什么专门的认证方法。与其让认证方不断更新来支持每个新的认证方法相比，EAP 更倾向于使用后台认证服务器，它可以实现一些或所有认证方法，此时认证方工作于传递模式。

1.1 要求说明书

1.2 术语

本文档经常使用下列词语：

认证方：启动 EAP 认证的链路终端。

被认证方：回应认证方的链路终端。（也就是客户端）

客户端：在 IEEE802.1X 中，链路终端回应被认证方。在本文档中，这个链路终端被称为被认证方。

后台认证服务器：后台认证服务器是一个提供认证服务给认证方的实体。当被使用时，这个服务器通常为认证方使用 EAP 方法。（也就是 **AAA** 服务器）

AAA：认证，授权和计费。支持 EAP 的 AAA 协议支持包括 RADIUS 和 Diameter。在这个文档中，AAA 服务器和后台认证服务器这两个术语是相同的意思。

EAP 服务器：终止和被认证方进行 EAP 认证方法的实体。在没有后台认证服务器时，EAP 服务器是认证方的一部分。在认证方工作在传递模式的情况下，EAP 服务器相当于后台认证服务器。

简单丢弃：这意味着执行操作没有做进一步处理的能力，只是将数据包简单的丢弃。该执行应该提供记录错误的能力，如丢弃包的内容；并在统计处记录下该事件。

成功认证：在本文档中，成功认证是一个 EAP 消息的交换，同样也是认证方决定允许被认证方访问和被认证方决定访问的结果。认证方的决定通常包括认证和授权两个方面；被认证方可能已经成功的向认证方得以认证，但是访问可能由于政策原因被认证方拒绝。

消息完整性检查：主要的哈希函数用于认证和数据完整性保护。这常常被称为消息验证码。

加密分离：两个密钥（x 和 y）是独立的加密，如果对手知道了在协议交换中所有的信息，也不能进行破密，即从 X 中计算出 Y，或者从 Y 中计算出 X。特别是，这个定义允许对手知道所有以明文形式发送的随机数，和在协议中使用的所有可预见的计数器的值。如果密钥是分开加密的，没有捷径来从 Y 中分离出 X 或从 X 中分离出 Y，若对手想得到密钥，他必须执行的计算，相当于执行一个穷尽搜索。

主会话密钥（**MSK**）：建钥资料是从 EAP 客户端和服务器间获取，通过 EAP 方法输出。MSK 至少是 64 字节长度。在现有的实现中，一个 AAA 服务器作为一个 EAP 服务器来传送 MSK 到认证方。

扩展的主会话密钥：附加的建钥资料从 EAP 客户端和服务器间获取，通过 EAP 方法输出。EMSK 至少 64 字节的长度。EMSK 不与认证方或其它第三方共享。EMSK 是为将来使用的，现在还没有定义。

结果标志：如果在方法的最后信息被发送或者接收以后，它提供结果显示：

1) 被认证方知道它是否认证了服务器，还有服务器是否认证了它。

2) 服务器知道它是否认证了被认证方，还有被认证方是否认证了它。

在这种情况下，成功的认证足够获得访问批准，于是被认证方和认证方将会知道另外一方是否提供或者接收访问。也可能不经常是这种情况。一个认证的被认证方可能被拒绝访问，由于缺少授权或者其它的原因。既然 EAP 交换是在被认证方和服务器间运行的，其它节点（例如 AAA 代理）也可能影响到授权的决定。这在 7.16 中被详细的讨论。

1.3 适用性

EAP 被设计用来使用在网络访问认证上，IP 层连接到达不了的地方。不建议将 EAP 用于其它用途，例如块数据传输。

由于 EAP 不需要 IP 连接，它仅仅为认证协议的可靠传输提供了足够的支持，其余的什么也没有。

EAP 是锁步协议，它仅仅支持每次传输一个数据包。因此，EAP 不能够有效的传输块数据，不像传输层协议例如 TCP 或 STCP。

虽然 EAP 为转发提供了支持，但是在它假定底层保证有序传输的基础上，所以不支持乱序接收。

由于 EAP 不支持分片和重组，EAP 认证方法产生的有效载荷大于 EAP MTU 需要提供分片支持的最小值。

虽然认证方法例如 EAP-TLS 支持分片和重组，在本文档中 EAP 方法不支持。因此，如果 EAP 数据包的大小超出了 EAP 链路的 MTU，这些方法将会遇到困难。

EAP 认证是由服务器（认证方）发起的，而很多认证协议是由客户端发起的，因此，为了运行 EAP，认证算法增加一两个额外的信息是必要的。

凡基于证书的认证都是支持的，由于证书链的分片，额外的往返包的数量可能增多。一般来说，一个分片的 EAP 数据包由于有分片，将需要很多的往返包来发送。例如，一个认证链的大小是 14960 个字节，将需要 10 个往返来发送一个 1496 字节大小的 EAP MTU。

EAP 运行在底层，此时会有很多重要的包发生丢失，或者在认证方和认证服务器之间的通信时，重要的包丢失也发生，EAP 方法需要很多往返包来解决这些问题。在这种情况下，建议 EAP 方法使用较少的往返包。

2 扩展认证协议

EAP 认证交换过程如下：

1、 认证一方向另一方首先发送一个身份请求，并等待对方发来响应。

2、 对方在接到身份请求要求时，它应发送一个身份响应包，他们的 ID 必须相同

3、 认证方检查包类型是否是身份响应包，且 ID 是否与发送请求时一致。

4、 在收到身份响应时，认证方根据身份，检查通信实体数据库，以查找对应的认证方式，如找到则开始发送响应的认证请求。

5、 对方在收到认证请求时，首先检查它自己是否支持认证者所要求的认证请求，如不支持，则发送一个 NAK 否认包，如支持，则发送认证响应。

6、 认证者检查响应包，如是 NAK 否认包，则重发对方允许的认证请求。如是上一次的认证请求的响应，则检查它是否是它所期待的响应以决定认证成功或失败，从而决定链路的打开或关闭。

7、 如上一步成功，向对方发送认证确认包，以使对方把链路打开，通知上层网络控制协议进行网络通信协议参数的协商，随着各个网络控制协议打开之后，PPP 就把数据链路交给通信网络协议，这时双方才可以进行实际的通信。

优点：

- EAP 协议能够支持多种认证方法。
- 网络访问服务器设备不需要理解每个认证方法，同时可能为后台认证服务器作传递代理。支持传递是可选的。认证方可能认证本地的被认证方，同时可能为非本地被认证方和不能当地实施的认证方法作为传递方。
- 被认证方和后台认证服务器相分离，简化了证件管理和政策的制定。

缺点：

- 在 P2P 中使用，EAP 需要增加一个新的认证类型到 PPP LCP，因此 PPP 需要作出改变才能够使用它。它与之前的 PPP 认证模型不同，在链路建立阶段不指定认证方法。同样，交换机或无线接入点为使用 EAP 协议需要支持 IEEE802.1X。
- 被认证方和后台认证服务器分开，它使安全性分析复杂化，密钥分配也复杂化了。

2.1 支持序列

EAP 会话可能利用各种方法。一个典型的例子，在身份请求后跟着一个 EAP 认证方法，例如 MD5-挑战。然而，在一个 EAP 会话中，认证方和被认证方必须使用一种认证方法，之后认证方必须发送成功或失败数据包。

一旦被认证方已经发送和初始请求一样类型的回应包，认证方在一个给定的方法完成最后一轮之前，不能发送一个不同类型的请求包，也不能在初始认证方法完成之后，发送请求任意类型的额外方法；被认证方收到上述请求包时，必须把它们作为无效的，简单丢弃。因此，不支持身份重新查询。

被认证方在初始 non-NaK 回应被发送之后，不能为回复请求再发送一个 NaK。由于攻击者可能发送伪造的 EAP 请求包，认证方收到意外的 NaK 时应该丢弃它并且记录这件事情。

在一个 **EAP** 会话中不支持多路认证方法，因为它们容易受到拦截式攻击，并与现有的实现不兼容。

当使用一个特别的 **EAP** 认证方法时，其它的方法在它之中运行（一个隧道方法），此时上述在一个 **EAP** 会话中禁止多路认证方法就不适合了。这种隧道方法是一种特别 EAP 的认证方法。可提供后向兼容，因为一个不支持隧道方法的被认证方，能够用 NaK 回复初始 EAP 请求。为了解决安全性的缺陷，隧道方法必须支持保护对抗拦截式攻击。

2.2 EAP 多路传输模型

从概念上讲，EAP 的操作包括以下部分：

a) 底层。底层负责在认证方和被认证方之间转发和接收 **EAP** 帧。EAP 已经在各类的底层上运行，例如 PPP，有线 IEEE802 局域网，IEEE802.11 无线局域网，UDP 和 IKEv2，TCP。底层的表现在第 3 节中讨论。

b) **EAP** 层。EAP 层通过底层收到和转发 **EAP** 数据包，完成重复检测和转发，同时从 EAP 被认证方和认证层传递和接收 EAP 信息。

c) **EAP** 被认证方和认证层。基于代码字段，EAP 层多路分离输入 EAP 数据包到 EAP 被认证方和认证层。通常情况下，在一给定主机上运行的 EAP，将支持被认证方或者认证方的功能，同样对于一个主机来说，也可以同时担当 EAP 被认证方和认证方。此时，EAP 被认证方和认证层都将会出现。

d) EAP 方法层。EAP 方法通过 EAP 被认证方和认证层，实现认证算法和接收转发 EAP 信息。由于 EAP 本身不支持分片，现在这是 EAP 方法的责任，这将在第 5 节中被讨论。

EAP 多路传输模型在下图一种说明。注意这里没有要必须遵守这个模型，只要线上行为与它保持一致就行。

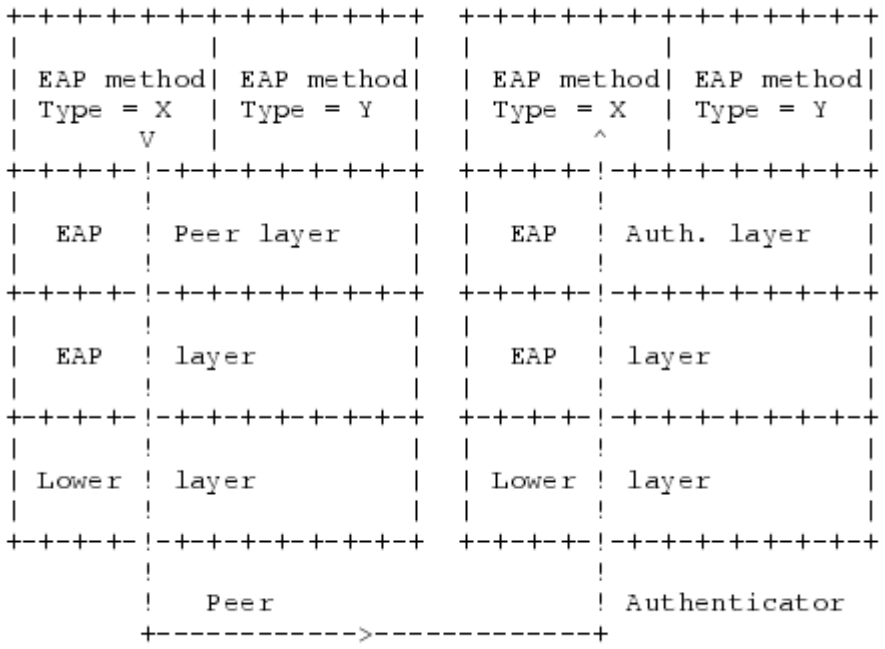


Figure 1: EAP Multiplexing Model

在 EAP 中，代码字段功能很像在 IP 中的协议号码。假定 EAP 层根据代码字段多路分离传入的 EAP 数据包。接收到的 EAP 数据包（1 请求，3 成功和 4 失败）被 EAP 层传到 EAP 的被认证方层。带有代码=2（回应）的 EAP 数据包被传送到 EAP 认证层。

在 EAP 中，类型字段功能很像 UDP 或 TCP 中的端口号码。假定 EAP 被认证方和认证层根据它们的类型多路分离输入的 EAP 数据包，然后把它们传送给相应类型的 EAP 方法。在一台主机实施的 EAP 方法可能注册来接收从被认证方或认证方来的数据包，或者都接收，这样根据它所支持的功能角色。

通知回复仅仅被作为确定被认证方接收到通知请求，不是它已经处理了通知请求，或者向使用者显示了这个信息。它不能假定通知请求或回复的内容对另外一种方法是有效的。通知类型将在 5.2 节讨论。

NaK（类型 3）或者扩展的 NaK（类型 254）都是用于协商的目的。被认证方回应一个 **NaK** 响应或者扩展 **NaK** 响应，给不想接收类型的初始 **EAP** 请求。它不能假定 NaK 响应的内容是另一种方法有效。NaK 类型将在 5.3 节被讨论。

带有成功或失败代码的 EAP 数据包不包括类型字段，同时也不被传送给一个 EAP 方法。成功和失败将在第 4.2 节中讨论。

鉴于上述考虑，成功，失败，NaK 响应和通知请求/回应信息不能用与承载注定要发送到其它 EAP 方法的数据。

2.3 传递行为

当作为一个传递认证方时，认证方检查代码，身份，字段长度，在 4.1 节描述的那样。它把从被认证方收到的、目的地址是它自己认证层的 EAP 数据包，转发给后

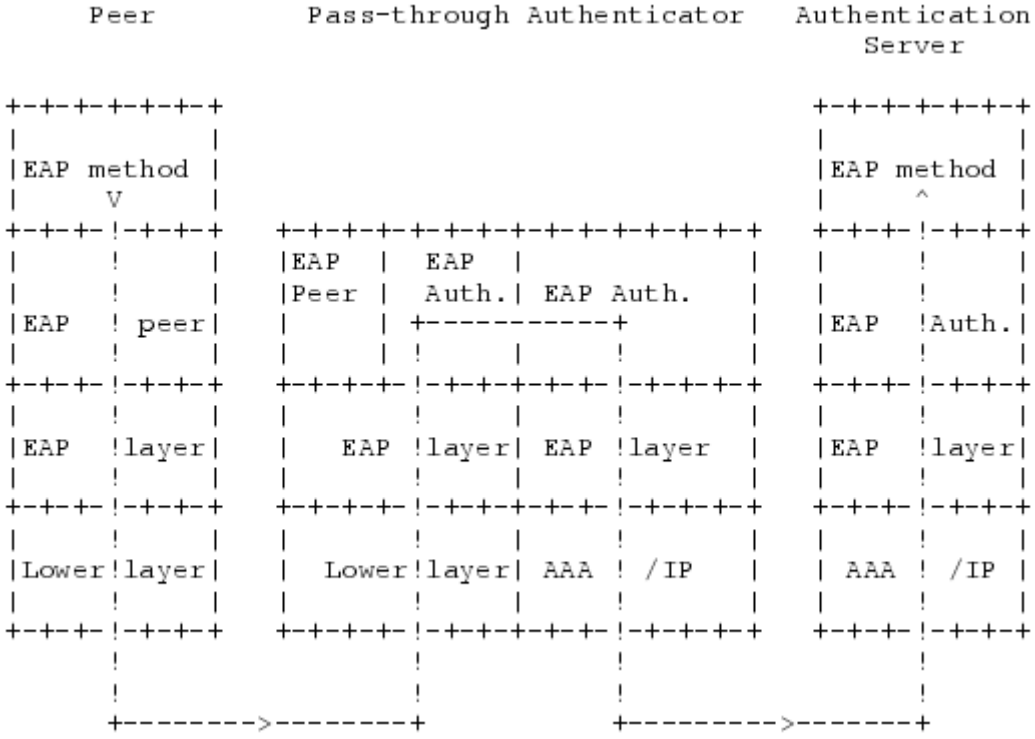
台认证服务器；从后台认证服务器接收到的、目的地是被认证方的数据包被转发到传递认证方。

一个主机收到 EAP 可能仅仅对其做下列三件事情中的一件：执行它，丢弃它，继续传送。继续传送的决定通常是根据检查后的代码，身份和字段长度。一个传递认证方必须能够把从被认证方接收的带有代码=2（回应）的 EAP 数据包转发给后台认证服务器。它也必须能够吧从后台认证服务器收到 EAP 数据包，将代码=1（请求），代码=3（成功），代码=4（失败）的 EAP 数据包传递给被认证方。

除非认证方本地完成一个或多个支持认证方角色的认证方法，EAP 方法层包头字段（类型、类型数据）作为转发决策的一部分不被检查。当认证方支持本地认证方法，它可能检查类型字段来决定是否执行这个数据包或者转发这个数据包。符合传递认证方的操作，必须默认转发任何类型的 EAP 数据包。

接收到代码=1（请求），代码=3（成功），以及代码=4（失败）的 EAP 数据包被 EAP 层多路分离，并且被传送到被认证方层。因此，除非一个主机实现 EAP 被认证方层，否则这些数据包将会被简单丢弃。同样的，接收到代码=2（回复）的 EAP 数据包也会被 EAP 层多路分离，并且被传送到认证方层。因此，除非主机实现 EAP 认证方层的功能，否则这些数据包将会被丢弃。传递性认证方层的行为在这个说明中没有被定义，并且像 RADIUS 和 Diameter 这些 AAA 协议不支持传递行为。

转发模型在图 2 中被说明：



图表 2:传递性认证方

在本章节中，认证方作为一个传递，它必须确定认证结果，基于后台认证服务器发送的接收/拒绝指示；结果必须不能被和接收/拒绝指示一起发送的 EAP 数据包的内容确定。

2.4 P2P 对等操作

由于 EAP 是 P2P 对等协议，一个独立和同步的认证可能发生在相反的方向上（取决于底层的能力）。链路的两个终端可能同时作为认证方和被认证方。这种情

况下，两个终端都必须实施 EAP 认证方和被认证方层。另外，在两个被认证方实现的 EAP 方法，必须同时支持认证方和被认证方的功能。

虽然 EAP 支持 P2P 的对等操作行动，但是一些 EAP 的实现，方法，AAA 协议和链路层可能不支持这个。一些 EAP 方法可能支持不对称的认证，其中被认证方需要一个类型的证书，认证方需要另一个类型的证书。使用上述 EAP 方法并支持 P2P 对等操作的主机，因此将需要预备两种证书类型。

AAA 协议像 RADIUS/EAP 和 Diameter EAP 仅仅支持传递认证操作。像在 2.6.2 中说明的那样，一个 RADIUS 服务器对访问请求回应一个封装了 EAP 请求，成功或失败的访问拒绝包。因此这里不支持传递式对等操作。

即使是一种方法被用于支持相互认证和结果显示，一些注意事项可能会指示，需要两个 EAP 认证（每个方向上一个）。主要有：

[1] 在底层支持双向会话密钥派生。底层像 IEEE802.11 可能只支持单向派生和传输临时会话密钥。例如，组密钥握手定义在 IEEE802.11i 是单向的，因为在 IEEE802.11 结构模型中，只有访问点发送多播/广播信息。在 IEEE802.11ad hoc 模型中，被认证方可能发送多播/广播流量，请求两个单向组密钥交换。由于设计的限制，这同样意味了单播密钥派生的需要和 EAP 方法交换，发生在每个方向。

[2] 在底层中支持打破僵局。底层像 IEEE802.11adhoc 不支持打破僵局，当两个主机相互初始认证时，将仅仅转送一个特别的认证。这意味着，虽然 802.11 支持双向组密钥握手，然后两个认证，每个方向一个，仍可能会发生。

[3] 被认证方政策补偿。EAP 方法可能支持结果显示，使被认证方能够用这种方法向 EAP 服务器显示，它成功的认证 EAP 服务器，同时使服务器将表明它也认证了被认证方。然而，传递认证方将不会得知被认证方已经接收了 EAP 服务器提供的证书，除非这个信息通过 AAA 协议提供给了认证方。认证方应该解释接收数据包中的密钥属性，作为被认证方成功认证服务器的信息显示。

然而，即使相互认证发生了，在初始 EAP 交换过程中，EAP 被认证方的访问策略不被满足。例如，EAP 认证方可能不会证明授权同时担任认证方和被认证方的职能。因此，被认证方可能需要在相反方向上加额外的认证，即使被认证方提供指示，EAP 服务器已经成功的认证了被认证方。

3 底层行为

3.1 底层需求

EAP 对底层有如下的假设：

[1] 不可靠的传输。在 EAP 中，认证方转发还没有接收到回应的请求，因此 EAP 不能假定底层是可靠的。由于 EAP 定义了它自己的转发行为，当 EAP 运行在可靠的底层时，转发是有可能同时发生在底层和 EAP 层。

注意，EAP 成功和失败数据包不被转发。没有可靠的底层和不可忽略的错误比例，这些数据包可能丢失，从而导致时延。因此降低 EAP 成功或失败数据包的丢包率是合适的，就像 4.2 节中说的那样。

[2] 底层错误检测。当 EAP 不能假定底层是可靠的，它将依赖于底层错误检测 EAP 方法可能不包括一个 MIC，或者如果 EAP 方法包括 MIC，也可能不计算 EAP 数据包的所有字段，像代码，身份，长度或类型字段。因此，没有底层

错误检测，不可预料的错误将会进入 EAP 层或者 EAP 方法层的包头字段，导致认证失败。

例如，EAP TLS，它仅仅通过类型数据字段计算出它的 MIC，把 MIC 确认失败当作致命性的错误。没有底层错误检测和其它类似的方法，底层将不会可靠。

[3] 底层安全。EAP 不会请求底层去提供安全服务，像是每个数据包加密，认证，完整型和重播保护。然而，当这些安全服务可获得时，支持密钥派生的 EAP 方法，能够被用于提供动态密钥材料。这将把 EAP 认证绑定到随后的数据上，同时保护数据不被修改、欺骗或重放。详细请见 7.1 节。

[4] 最小 MTU（最大传输单元）。EAP 能够在底层运行，同时提供一个 1020 字节甚至更大的 EAP MTU（最大传输单元）。

EAP 不支持路径 MTU 发现，同时也不支持分片和重组，但是不包括在本说明书中定义的方法：身份 1，消息 2，NaK 回应 3，MD5 挑战 4，一次性密钥 5，通用令牌卡 6，扩展的 NaK 响应类型 254 类型。

通常，EAP 被认证方从底层获取关于 EAP MTU 的信息，并设定 EAP 帧大小为合适的值。当认证方运行在传递模式时，认证服务器没有直接的方法来确认 EAP MTU，因此依赖于认证方提供这项信息，例如通过 MTU 帧属性，在 2.4 节描述的那样。

虽然例如 EAP-TLS 方法支持分片和重组时，但 EAP 方法本来是设计在 PPP 中使用，此时一个 1500 字节的 MTU 是为保证控制帧可能没有分片和重组特征。

EAP 方法能够假定一个最小的 1020 字节的 EAP MTU 没有其它的信息。EAP 方法应该包括支持分片和重组，如果它们的有效载荷比这个最小的 EAP MTU 大。

EAP 是一个锁步协议，当处理分片和重组时，它意味必然的低效率。因此，如果底层支持分片和重组（例如在 EAP 通过 IP 传输的地方），那么分片和重组发生在底层比在 EAP 中会更好。这能够通过提供一个人为的大 EAP MTU 给 EAP，导致分片和重组在底层中实现。

[5] 可能的重复。当底层可靠时，它将提供 EAP 层一个不重复的数据包流。然而，当底层提供无重复是令人满意的时，它就不是个请求。身份字段向认证方和被认证方都提供检查重复的能力。

[6] 排序保证。EAP 不要求身份单调的增加，因此为正确运行而依赖于底层正确的排序。EAP 本来是定义运行在 PPP 上，同时第一节中有排序请求:

> “P2P 对等协议是被设计用来在两个被认证方之间的简单链路上传送数据包设计。这些链路提供了全双工的同步的双向操作，同时被假定是按顺序传送数据包。”

EAP 的底层传输协议必须维持源地址和目的地之间的给定优先级水平的排序。

重新排序，如果这个发生的话，通常会导致 EAP 认证失败，导致 EAP 认证重新开始。在重新排序这种环境下，EAP 认证失败将会十分平常。建议运行 EAP 的底层能够提供排序保证；在原始的 IP 或 UDP 传送中运行 EAP 是不被推荐的。在 RADIUS 中封装 EAP 满足了排序的需要，由于 RADIUS 是锁步协议，因此需要按顺序发送数据包。

3.2 在 PPP 中 EAP 的用法

为了通过 PPP 的链路建立通信，在链路建立阶段，PPP 链路每个的终端首先发送 LCP 数据包来配置数据链路。链路建立完毕后，进入到网络层协议阶段前，**PPP** 提供一个可选的认证。

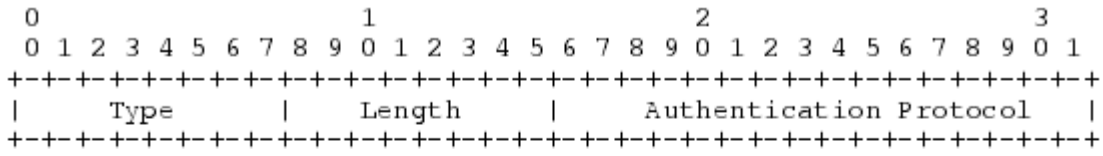
默认情况下，认证不是强制的。如果要求链路的认证，在链路建立阶段期间，必须要明确指定认证协议配置选项。

如果被认证方的身份在认证阶段中显示了，服务器能够为接下来的网络层协商使用选项中的身份。

当在 **PPP** 中实施时，**EAP** 不能在 **PPP** 链路控制阶段选择一个具体的认证方法，而是把这个推迟到认证阶段。这允许了认证方在确定专门的认证策略前获得更多的信息。当 PPP 认证方仅仅通过认证交换时，这也同样允许使用实际上能够实现各种认证方法的后台服务器。PPP 链路建立和认证阶段，和认证协议配置选项，被定义在 PPP 协议中。

3.2.1 PPP 配置选项格式

与 EAP 协商的 PPP 认证协议配置选项格式如下所示。字段从左端传送到右端。准确的说，一个 EAP 数据包被封装到一个 PPP 数据链路层帧的信息字段，此时协议字段显示了类型 hex C227。



Type

3

Length

4

Authentication Protocol

C227 (Hex) for Extensible Authentication Protocol (EAP)

3.3 在 IEEE802 中使用 EAP

通过 IEEE802 封装 EAP 被定义在 IEEE802.1X。在 IEEE802 封装的 EAP 不包含 PPP，同时 IEEE802.1X 不支持对链路或网络层的协商。因此，在 IEEE802.1X 中，不可能商议非 EAP 的认证方法，例如 PAP 或 CHAP。

3.4 底层标志

底层标志的可靠性和安全性依赖于更低的层。由于 EAP 是媒体独立的，底层安全性的存在与否不会被考虑到 EAP 信息处理中。

为了提高可靠性，如果被认证方收到底层的成功标志消息，其被定义在 7.2 节，它可能包括一个成功数据包已经丢失，和表现为好像它已经收到一个成功数据包。这也包含了在某些情况下选择忽略成功数据包，像是在 4.2 节描述的那样。

关于在 PPP 底层可靠性和安全性问题的讨论，IEEE802 有线网络和 IEEE802.11 无线局域网能够在安全性思考中找到，7.12 节。

在 **EAP** 认证结束后，被认证方通常将会通过认证方转发和接收数据。需要提供保证，实体转发的数据是同一个成功结束 EAP 认证的数据。为了达到这个目的，

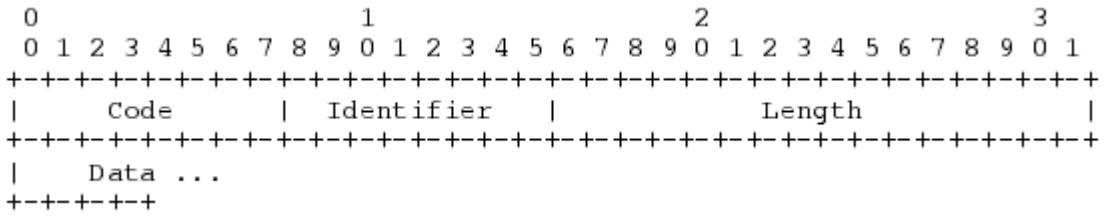
底层必须提供每个包的完整性，认证和重放保护，并在 EAP 认证阶段，把这些数据包的服务同派生的密钥相互绑定。除此之外，随后的数据流也有可能被修改，欺骗和重演。

底层加密套接字的密钥材料本身由 EAP 提供的，加密套接字协商和密钥激活是由底层控制的。在 PPP 中，加密套接字在 ECP 中被协商，因此不可能使用从 EAP 认证中派生的密钥，直到 ECP 结束。因此，一个初始的 EAP 交换不能被 PPP 加密套接字保护，尽管 EAP 再次认证可能被保护。

在 IEEE802 媒体中，初始的密钥激活通常发生在 EAP 认证之后。因此一个初始的 EAP 交换通常不被底层的加密套接字保护，尽管 EAP 的重新认证或预认证交换可能被保护。

4 EAP 数据包格式

EAP 数据包格式由下图所示。这些字段从左到右传送。



代码

代码字段是一个字节，确定了 EAP 数据包的类型。

EAP 代码的代表含义：

1 请求

2 回应

3 成功

4 失败

由于 EAP 仅仅定义了代码 1-4，EAP 数据包其它的代码被认证方和被认证方简单丢弃。

身份

身份字段是一个字节，用于匹配请求的回应。

长度

长度字段是两个字节，显示了长度， EAP 数据包包含代码，身份，长度和数据字段的字节数。超出长度字段的字节应该被当作数据链路层的填补，一旦接收必须被忽略。当消息长度字段设置的值大于接收字节数时，必须被简单丢弃。

数据

数据字段是 0 字节或者更多的字节。数据字段的格式被代码字段确定。

4.1 请求和回应

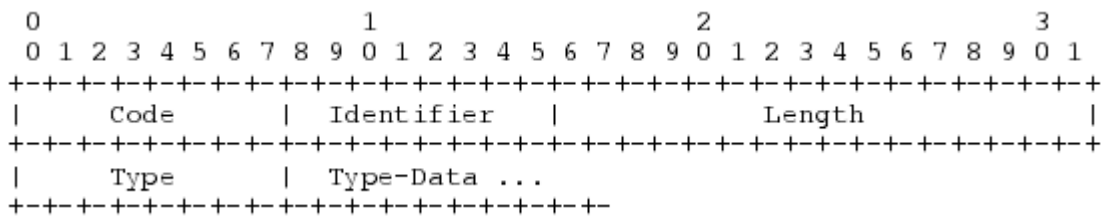
描述

请求数据包被认证方发送到被认证方。每个请求都有一个类型字段，其中显示了什么在被请求。额外的请求数据包必须等到有效的回应数据包被接收到，或一个可选的重试计数器到期，或者底层失败显示被接收到后，才能被发送。

转发的请求在发送时必须带有相同的身份，目的是为了与新请求包相互区分。数据字段的内容依赖于请求类型。被认证方必须发送一个回应包，用来回应有效的请求数据包。回应包只有在回应有效的请求是才能被发送，绝对不能定时转发。

如果一个被认证方接收到有效的重复请求包时，此时它已经发送了一个回应包，它必须重新发送自己的初始回应，同时不能再次处理请求包。请求必须按照它们接受的顺序进行处理，同时只有在处理结束之后才能检查下一个请求。

请求包和回应包的格式如下所示，这些字段从左到右发送。



代码

1 请求

2 回应

身份

身份字段是一个字节。当等待回应时，如果请求数据包由于超时被转发，身份字段必须相同。任何新的请求必须修改身份字段。

回应的身份字段必须和当前未解决的请求相符合。认证方接收到一个回应包，如果它的身份不能符合当前未解决的请求包，必须简单丢弃。

为了防止在新请求和转发请求之间混淆，每个新请求的身份值应该和之前请求身份值不同，但是不需要在整个会话过程中是独一无二的。一个可以达到此目的的方法就是开始时为身份赋予初值，随后每个新的请求包的身份值依次递增。最好是开始的身份值是随机的，而不是从零开始，因为它会导致序列攻击什么的，变得很麻烦。

由于身份空间对每个字段来说是独一无二的，认证方没有限制同时只能有 256 个认证会话。同样的，当再认证时，一个 EAP 会话可能持续一段很长的时间，它也不会被仅仅限制在 256 个往返行程。

注意事项：被认证方负责转发请求信息。如果请求信息从其它地方获得（例如从后台认证服务器），那么认证方为了它，将需要存储请求的拷贝。在用任何方法处理它们之前，被认证方负责探测和处理重复请求信息，包括把它们传送到外面。在用任何方法执行它们之前，认证方同样负责丢弃身份值不符合的回应信息，包括为了验证，把它们传送到后台认证服务器。由于认证方能够在收到被认证方的回应包之前转发数据包，因此认证方能够接收到多路的回应包，每个都有一个相符的身份值。直到一个新的请求被认证方接收到，身份的值不会被更新，因此认证方转发回应包给后台认证服务器，每次一个。

长度

长度字段是两个字节，显示了包含代码，身份，类型和类型数据字段的 EAP 数据包的长度。长度字段范围之外的字节应该被当作数据链路层的填充，同时一旦收到必须被忽略。当消息长度字段设置的值大于接收字节数时，必须被简单丢弃。

类型

类型字段是一个字节。这个字段显示了请求或回应的类型。一个特别的类型必须为每个请求包或回应包描述。初始的类型说明书在本文档的第 5 小节中讨论。

回应包的类型字段要么必须与请求相符合，要么必须与遗产或扩展的 Nak 相符合，它显示了被认证方不接受的请求类型。被认证方在初始 non-NaK 回应被

发送后，必须不能发送 NaK 作为请求的回应。EAP 服务器接收到的没有满足这些要求的回应，都要被简单丢弃掉。

类型数据

类型数据字段随着请求、相关回应的类型而有不同的值。

4.2 成功和失败

在 EAP 认证方法完成，并同时显示被认证方已经成功向认证方认证后，成功数据包被认证方发送给被认证方。认证方必须传送代码字段为 3 的 EAP 数据包。如果认证方不能认证被认证方（对一个或多个请求来说是不可接收的回应），那么在 EAP 方法不成功认证之后，必须发送一个代码字段为 4 的失败 EAP 数据包。为了允许人为错误，认证方在发送失败回应之前，可能希望发出多路请求包。成功和失败数据包必须不能包含额外的数据。

如果给定方法的说明没有明确允许这个方法在那个端点完成，成功和失败数据包必须不能够被 EAP 认证方发送。被认证方 EAP 接收到成功或者失败数据包时，若发送端不是明确被允许，则必须丢弃。默认情况下，EAP 被认证方必须简单丢弃一个包装的成功数据包（一旦连接，成功数据包立即发送）。

注意事项：因为成功和失败数据包不被承认确认，它们不能被认证方转发，也可能丢失了。一个被认证方必须考虑到这个注意事项中的情形。

像是在 2.1 节描述的那样，在一次 EAP 会话中，只有单独一个 EAP 认证方法被允许。EAP 方法可能实现结果标志。认证方发送了一个失败结果标志给被认证方后，不管从被认证方传来的回应消息，它必须随后发送一个失败数据包。在认证方发送一个成功结果标志给被认证方，并且从被认证方接收到成功结果标志后，它必须随后发送一个成功数据包。

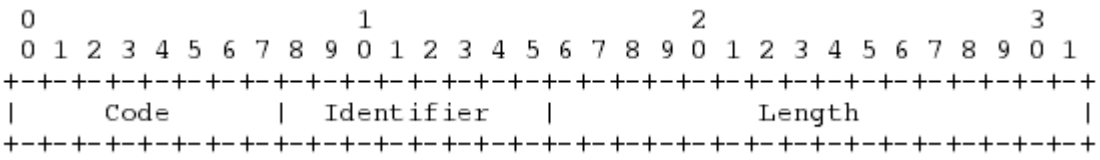
在被认证方，一旦方法没有不成功(例如认证方发送了失败结果标志，或者被认证方决定它不想要继续这个会话了，可能在发送一个失败结果标志后)，被认证方必须终止这个会话并且对底层显示失败信息。被认证方必须简单丢弃成功数据包，并且可能简单丢弃失败数据包。因此，失败数据包的丢失不会导致超时。

在被认证方，成功结果标志被双方交换后，一个失败数据包必须被丢弃。被认证方可能，如果 EAP 成功数据包没有被接收，EAP 成功数据包的丢失和认证成功。

如果认证方还没有发送一个结果标志，被认证方将会继续这个会话，一旦这个方法结束，被认证方会等待一个成功或失败数据包，同时不能够丢弃其中的任何一个数据包。如果成功或失败数据包都没有接收到，那么被认证方应该终止这个会话，以防止在数据包是 EAP 失败包的时候超长超时。

如果被认证方想要向认证方认证，并且这样做失败了，那么认证方必须发送一个失败数据包，并且不能通过发送一个成功数据包而批准访问。然而，认证方可能忽略，被认证方在限制的访问已经获得的情况下得到了认证，在这种情况下，认证方必须发送一个成功数据包。

当被认证方成功向认证方认证，但是认证方没有发送结果标志，认证方可能通过发送一个失败数据包拒绝访问，此时被认证方不能当时获得网络访问的资格。成功和失败数据包的格式如下所示。字段从左到有传输。



代码：

3 成功

4 失败

身份：

身份字段是一个字节，用于匹配回复响应。身份字段必须和回应数据包的身份字段相符合。

长度：

4

4.3 转发行为

由于认证过程往往包含用户输入，对于转发策略和认证超时必须慎重做决定。默认情况下，EAP 运行在不可靠的底层，EAP 转发计时器应该被动态的估计。建议最多转发 3-5 个。

当运行在可靠的底层时，认证方转发计时器应该被设定一个无穷大的值，因此在 EAP 层转发才不会发生。被认证方可能坚持维持一个超时值，为了避免无限的等待请求包。

在认证过程需要用户输入，测量往返次数可能取决于用于用户的反应而不是网络的特点，因此动态的 RTO 估计可能不会有帮助。相反的，转发计数器应该被设定，为用户响应提供足够的时间，在某些情况下需要较长的超时时间，例如当令牌卡参与。

为了给 EAP 认证方提供合适的超时值，一个提示可以被后台认证服务器传递给被认证方。

为了动态的估计 EAP 转发计数器，建议使用 SRTT,RTTVAR,RTO 估计算法，在 RFC2988 中描述，包括使用 Karn’s 算法，和以下潜在的修改：

a) 为了避免在分布式系统中固定计时器发生同步的行为，转发计时器通过使用 RTO 值和随机在-RTOmin/2 和 RTOmin/2 之间增加一个值，同抖动一起计算。也可能使用另一种计算来增加抖动。这些必须是伪随机的。

b) 当 EAP 在一个单一链路上传输时（不同于通过互联网），RTOinitial，RTOmin 和 RTOmax 这些较小的值被使用。推荐的数据是 RTOinitial=1 秒，RTOmin=200 毫秒，RTOmax=20 秒。

c) 当 EAP 在一个单一链路上传输时（不同于通过互联网），预算可能在每个认证方的基础上完成，而不是每个会话的基础上。这使得转发估计能最大的使用链路层上的信息。

d) 反馈计数器很多次后，EAP 可能清除 SRTT 和 RTTVAR，因为很有可能，当前的 SRTT 和 RTTVAR，在这种情况下是伪造的。一旦 SRTT 和 RTTVAR 被清零，它们应该用下一个 RTT 样本初始化，具体描述在 RFC2988。

5 初始 EAP 请求/回应类型

这一小节定义了在请求/回应交换中使用的 EAP 类型初始设置。更多的类型可能被定义在将来的文件中。类型字段是一个字节，并且确定了 EAP 请求或回应数据包的结构。前 3 个类型被当作是特殊的事件类型。

剩下的类型定义了认证交换。Nak（类型 3）或者扩展的 Nak（类型 254）仅仅对回应数据包有效，它们不能发送在请求包中。

所有的 EAP 实现必须支持类型 1-4，同时也应该支持类型 254。

1 身份
2 通知
3 Nak
4 MD5-挑战
5 一次性密钥
6 GTC 通用令牌卡
254 扩展类型
255 实验性的使用

EAP 方法可能支持基于共享密钥的认证。如果共享的密钥是由用户输入的口令，可以实现支持输入非 ASCII 类型的口令。在这种情况下，输入应该用一个合适的 stringprep 配置文件来处理，同时用 UTF-8 编码。

5.1 身份

描述

身份类型是用来询问被认证方的身份。通常情况下，认证方将宣布这作为初始请求。可选的显示信息可能被包括用来提示被认证方一种情况，即希望和用户互动。类型为 1（身份）的回复应该回复类型为 1（身份）的请求。

默认情况下，EAP 不应该假定身份请求或回应能大于 1020 字节。

建议身份回应主要用于路由选择目的和选择使用哪种 EAP 方法。EAP 方法应该包括一个专门为获得身份的方法，因此它们不需要依靠身份响应。身份请求和回应被明文发送，因此攻击者可以窥探身份信息，甚至更改或欺骗身份交换。为了解决这些威胁，最好是 EAP 方法中包含支持每个数据包认证、完整性和重放保护、保密性的身份交换。身份响应对这个方法来说可能不是合适的身份；它可能被缩短或模糊以提供机密性，或者被修改用于路径目的。当被认证方被配置为仅仅接收支持保护身份交换的认证方法，被认证方可能提供一个简短的身份回应。进一步关于身份保护的讨论，请看 7.3 节。

注意事项：被认证方可能通过用户输入获得身份。在无效的身份或认证失败的情况下，认证方被建议重新尝试身份请求。在终止认证之前，建议身份请求应该至少尝试 3 次。通知请求可能被用于显示无效的认证请求，优先于传输一个新的身份请求。

类型
1
类型数据

这个字段可能包含在请求中可显示的信息，包括 UTF-8 编码的 10646 属性。请求包中什么也没有，仅仅优先于空的一部分字段被显示。如果身份是未知的，身份回应字段应该是在长度上为 0 字节。身份回应字段不应该被无效的终止。在所有的情况下，类型数据字段的长度应该从请求/回应数据包的长度字段中获得。

安全需求（看 **7.2** 节）

认证方法: None
加密套接字协商: No
相互认证: No
完整性保护: No
重放保护: No
机密性: No
密钥派生: No
密钥强度: N/A
字典式攻击保护.: N/A
快速链接: No
Crypt. binding: N/A
会话独立: N/A
分片: No
频道绑定: No

5.2 通知

描述

通知类型是可选的，用于从认证方向被认证方传递显示消息。当这里没有未完成的请求，或优先完成一个 EAP 认证方法时，认证方可能随时向被认证方发送通知请求。被认证方必须对通知请求回复通知响应，除非 EAP 认证方法专门禁止使用通知消息。在任何情况下，一个 NaK 回应必须不能当作 NaK 请求的回应被发送。注意，默认的通知请求的最大长度是 1020 字节。默认情况下，这最多为可读信息留了 1050 字节。

EAP 方法在它的说明书中可能显示，通知消息不能在方法实施过程中发送。在这种情况下，被认证方必须简单丢弃从端点取得的通知请求。

被认证方应该显示这个信息给用户，或者如果没有显示的话应该记录下来。通知类型被打算用于提供一些重要性质的通知，但不是错误显示，因此不改变被认证方的状态。在很多情况下，通知不被需要。

类型

2

类型数据

请求包中的类型数据字段包含一个显示的信息，在长度上大于零字节，也包含 UTF_8 编码的 ISO10646 属性。信息的长度取决于请求数据包的字节长度。消息不能够被无效的终止。回应必须作为类型字段为 2 的请求包的回应。回应包的类型数据字段在长度上是零字节。回应包应该被立即发送。

安全声明（看 **7.2** 节）

Auth. mechanism: None
Ciphersuite negotiation: No
Mutual authentication: No
Integrity protection: No
Replay protection: No
Confidentiality: No
Key derivation: No
Key strength: N/A

Dictionary attack prot.: N/A
Fast reconnect: No
Crypt. binding: N/A
Session independence: N/A
Fragmentation: No
Channel binding: No

5.3 NaK

5.3.1 遗留的 NaK

描述

遗留的 NaK 类型只有在回应信息中是有效的。当期望的认证类型没有被接收到时，它就会当作回应请求发送出去。认证类型是 4 或者更高。回应包含被认证方想要的一个或多个认证类型。类型 0 被用于显示发送者没有其它的选择，因此认证方在接收到一个包含值为 0 的 NaK 回应后，不应该发送另一个请求。

由于遗留的 NaK 类型仅仅在回应包中是有效的，并且在功能上很大限制，它不能被用作一般的错误显示目的。

代码

2 用于回应

身份

身份字段是一个字节，用于匹配请求的回应包。遗留的 NaK 响应的身份字段必须符合请求数据包的身份字段。

长度

》=6

类型

3

类型数据

当被认证方接收到不想接收认证类型（4-253，255）的请求包时，或者不支持扩展类型的被认证方接收到类型值为 254 的请求包，NaK 响应（类型 3）必须被发送。NaK 响应的类型数据字段必须包含一个或更多的字节，显示想要获取的认证类型，每个类型一个字节，或者值为 0，来显示没有预期的其它选择。支持扩展类型的被认证方，接收到不想接收类型（4-253，255）的请求，可能在 NaK 回应中包含数值 254，来显示对扩展认证类型的渴望。如果认证方能够适应这个优先权，它将回应一个扩展的类型请求（类型 254）。

安全性声明（看 **7.2** 节）

Auth. mechanism: None
Ciphersuite negotiation: No
Mutual authentication: No
Integrity protection: No
Replay protection: No
Confidentiality: No
Key derivation: No
Key strength: N/A
Dictionary attack prot.: N/A
Fast reconnect: No

Crypt. binding: N/A
Session independence: N/A
Fragmentation: No
Channel binding: No

5.3.2 扩展的 NaK

描述

扩展的 NaK 类型只有在回应消息中是有效的，它只有在回应类型 254 的请求包时被发送，此时认证类型是不可接收的。扩展的 NaK 类型使用它自己的扩展类型格式，同时响应包括被认证方期望的一个或更多的认证类型，所有都在扩展类型格式中。类型 0 是用于显示发送者没有另外的可选项。扩展类型的普遍格式在 5.7 节中描述。

由于扩展的 NaK 类型只有在回应包中是有效的，并且在功能上很大限制，它不能被用作一般错误显示目的。

代码

2 用于回应

身份

身份字段是一个字节，用于符合请求的响应。扩展 NaK 响应的身份字段必须符合请求数据包的身份字段。

长度

》=20

类型

254

供应商 **ID**

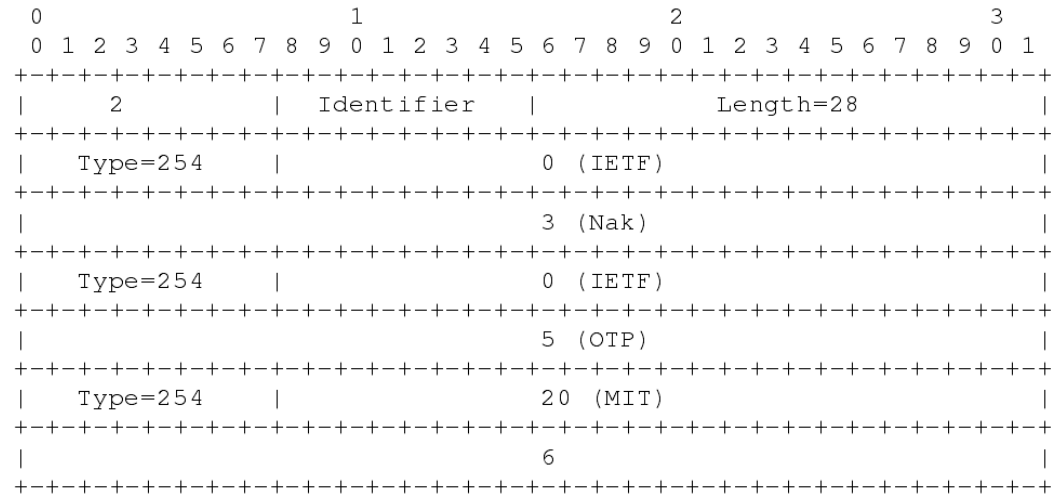
0

供应商类型

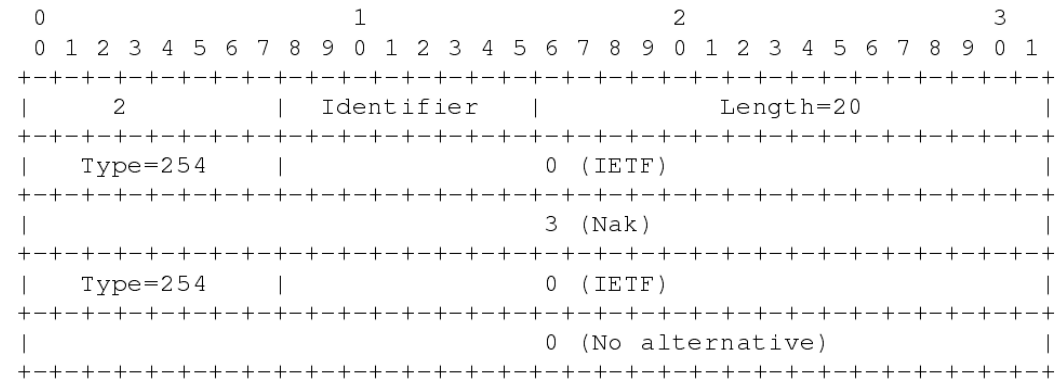
3（NaK）

供应商数据

扩展的 NaK 类型只能在请求包含扩展类型 254 时被发送。NaK 回应的供应商数据字段必须包含一个或更多的认证类型（4 或者更大），所有的都在扩展的格式中，每个类型 8 个字节，或者值为零，也在扩展的类型格式中，来显示没有其它的选项。期望的认证类型可能包括专门供应商和 IETF 类型的混合。例如，扩展的 NaK 响应显示优先 OTP，和一个 MIT 扩展类型 6 将显示如下：



一个扩展的 NaK 响应显示没有其它的选项将显示如下：



安全性声明（见 7.2 小节）

Auth. mechanism: None
Ciphersuite negotiation: No
Mutual authentication: No
Integrity protection: No
Replay protection: No
Confidentiality: No
Key derivation: No
Key strength: N/A
Dictionary attack prot.: N/A
Fast reconnect: No
Crypt. binding: N/A
Session independence: N/A
Fragmentation: No
Channel binding: No

5.4 MD5-挑战

描述

5.5 MD5-挑战类型与 PPP CHAP 协议相类似。请求包含一个给被认证方的挑战信息。回应包必须作为响应请求而发送。回应既可能是 MD5-挑战 4，Nak 3,或者扩展的 Nak 254。Nak 回复显示了被认证方期望的认证类型。EAP 被认证方和 EAP 服务器必须支持 MD5-挑战方法。仅仅支持传递性的认证方必须允许同支持 MD5-挑战

的后台认证服务器通信，尽管 EAP 认证方本身不需要支持 MD5-挑战。然而，如果 EAP 认证方能被配置认证本地的被认证方，那么可以申请支持 MD5-挑战方法。

注意到在 MD5-挑战类型中使用身份字段是不同于在 RFC1994 中描述的那样。EAP 允许转发 MD5-挑战请求数据包，当 RFC1994 声明身份和挑战字段在每次挑战发送时就要改变一次。

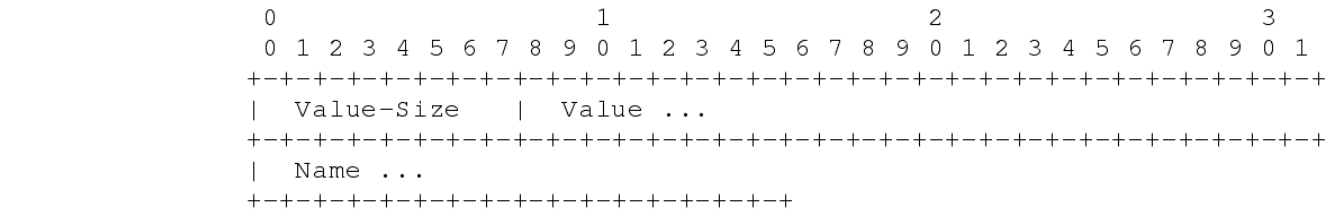
RFC1944 把共享密钥当作字节串，同时不会说明它是怎么进入系统的。EAP MD5-挑战可能支持输入非 ASCII 类型的密钥

类型

4

类型数据

类型数据字段的内容如下总结。查阅这些字段的使用，可以查看 PPP 挑战握手认证协议。



安全性声明（查看 7.2 节）

Auth. mechanism: Password or pre-shared key.
Ciphersuite negotiation: No
Mutual authentication: No
Integrity protection: No
Replay protection: No
Confidentiality: No
Key derivation: No
Key strength: N/A
Dictionary attack prot.: No
Fast reconnect: No
Crypt. binding: N/A
Session independence: N/A
Fragmentation: No
Channel binding: No

5.6 一次性密钥（OTP）

描述

请求包含一次性密钥挑战格式，其被描述在 RFC2289。响应必须作为请求的回应被发送。响应是类型 5（OTP），Nak（3），或者扩展 Nak（254）。Nak 响应显示了被认证方期望的认证类型。EAP 的一次性密钥方法被打算用于一次性密钥系统，不能被用于为明文提供支持。

类型

5

类型数据

类型数据字段包含了 OTP 挑战，作为请求中的可显示的信息。在响应中，这个字段被用作从 OTP 字典（RFC2289）中的 6 个字。消息必须不能被无效的忽略。字段的长度从请求/回应数据包的长度字段中得到。

注意：RFC2289 不明确用户如何秘密的传递密钥，或者密钥短语怎样被变换成字节的。EAP OTP 可能支持输入非 ASCII 性质的密钥短语。查看说明书的第 5 小节，输入怎样被处理和变换成字节的。

安全性声明（查看 **7.2** 节）

Auth. mechanism: One-Time Password
Ciphersuite negotiation: No
Mutual authentication: No
Integrity protection: No
Replay protection: Yes
Confidentiality: No
Key derivation: No
Key strength: N/A
Dictionary attack prot.: No
Fast reconnect: No
Crypt. binding: N/A
Session independence: N/A
Fragmentation: No
Channel binding: No

5.7 通用令牌卡

描述

通用令牌卡类型被定义为各种需要用户输入信息的令牌卡。请求包含可显示的信息，同时回应包中包含认证必须的令牌卡信息。通常，令牌卡信息是由用户从令牌卡设备上读取得到并作为 ASCII 文本输入的。响应包必须作为回应请求包被发送。回应包的类型必须是类型 6（GTC），Nak（3）或者扩展的 Nak（类型 254）。Nak 响应显示了被认证方期望的认证类型。EAP GTC 方法被打算用于支持挑战/响应认证的令牌卡。

类型

6

类型数据

请求中的类型数据字段包含一段长度大于零的可显示的信息，它的长度取决于请求数据包的长度字段，因此该消息

能被无效终止。应答必须作为类型字段为 6 的请求的回应。应答中包含用于认证的令牌卡信息，通常它的长度也可以从报文的长度字段中计算得到。

EAP GTC 可能支持输入非 ASCII 性质的响应。查看第 5 节的说明，输入是怎样处理的和变换成字节的。

安全性声明

Auth. mechanism: Hardware token.
Ciphersuite negotiation: No
Mutual authentication: No
Integrity protection: No

Replay protection: No
Confidentiality: No
Key derivation: No
Key strength: N/A
Dictionary attack prot.: No
Fast reconnect: No
Crypt. binding: N/A
Session independence: N/A
Fragmentation: No
Channel binding: No

5.8 扩展的类型

描述

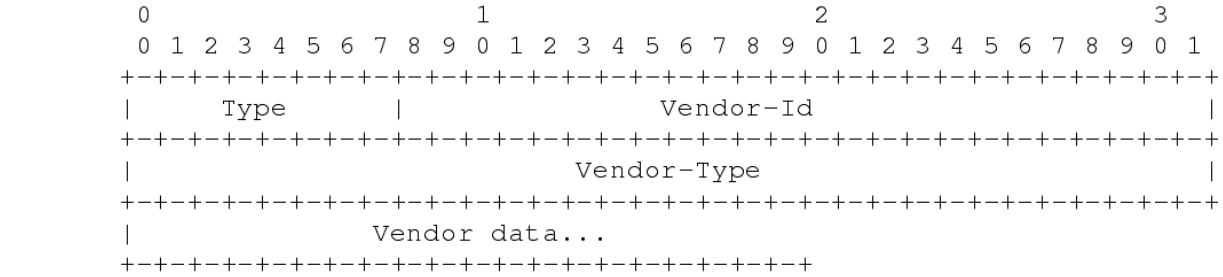
由于很多存在使用的 EAP 都是供应商专门使用的，扩展的方法类型能允许供应商支持它们自己的扩展类型，

适合于普遍应用。

扩展的类型也被用于扩展全球方法类型空间，除了一开始的 255 个值。供应商 ID 为 0 的将原始的 255 可能的类型映射成 2^32-1 个可能的类型空间。（类型 0 只用在 NaK 回应包中，来显示没有接收的选择）

支持扩展属性的操作必须平等对待少于 256 的 EAP 类型，不论它们是否表现为单个字节或在扩展类型供应商 ID 为 0 中，作为 32 位供应商类型。没有准备解释扩展类型的被认证方必须发送一个 NaK，其在 5.3.1 节中描述，同时商量一个更合理的认证方法。

扩展的类型格式如下所示。字段从左到右传输。



类型

254 为扩展的类型

供应商 **ID**

供应商 ID 是 3 字节，代表了 SMI 网络管理民营企业代码字节的供应商顺序，由 IANA 分配。供应商 ID 为 0 的由 IETF 预留使用，提供扩展的全球 EAP 类型空间。

供应商类型

供应商类型字段是 4 字节，同时代表了供应商专门的方法类型。

如果供应商 ID 为 0，那么供应商类型字段是可扩展的，也是 EAP 类型已存在的命名空间的超集。前 256 个类型被预留用作和特别字节的 EAP 类型兼容，这已经被计划使用了，或者在将来中已经被计划使用了。因此，EAP 类型值从 0 到 255 在语义上是相等的，不论它们表现为特别一个自己的 EAP 类型或者当供应商 ID 为 0 时候的供应商类型。这条规则只有一个例外：扩展的 NaK 和一流的 Nak 数据包共享相同的类型，但必须被区分开来，因为它们有不同的格式。

供应商数据

供应商数据字段被供应商定义。当供应商 ID 为 0 时，供应商数据字段将会被用作传输 EAP 方法类型的内容，其被 IETF 定义。

5.9 实验

描述

实验类型没有固定的格式或内容。在实验新 EAP 类型时使用。这种类型被打算用作实验和测试目的。在被认证方之间使用这个类型不能保证可互操作性。

类型

255

类型数据

没有定义

6 IANA 的考虑

本节对 IANA 关于对相关的 EAP 协议注册值提供了指导，与 BCP 26 相符合。

在 EAP 中有两个名字空间需要注册：包代码和方法类型。

EAP 没有被打算作为通用的协议，不应该用于和注册不相关的目的。

接下来的属于将会被使用，其被定义在 BCP26：命名空间，分配值，注册。

接下来的术语将会被使用，其被定义在 BCP26：私人使用，先到先服务，专家审查，需要的规范，IETF 共识，标准行为。

一个指派的专家应该查看注册请求，重要的 IESG 领域主管应该确定制定的专家。目的是任何分配将会被发表的 RFC 陪伴。但是为了允许利益的分配，优先于 RFC 被支持发表，指派的专家～～～貌似这段没有用

6.1 数据包代码

数据包代码的范围从 1 到 255，其中 1-4 已经被分配了。因为一个新的数据包代码对互操作性有一定的影响，一个新的数据包代码请求标准行动，并且从 5 开始被分配。

6.2 方法类型

初始的 EAP 方法类型空间的范围是从 1-255，在 EAP 中资源是不足的，因此必须小心的分配。方法类型 1-45 已经被分配了，还有 20 个可回收使用。方法类型 20 和 46-191 可能根据专家的意见和规范要求进行分配。

方法类型块的分配应该请求 IETF 的意见。EAP 类型值 192-253 被预留，分配时需要标准的行动。

方法类型 254 被分配用于扩展类型。当供应商 ID 字段非零时，扩展的类型仅仅被用于为一个供应商 EAP 的特定功能，此时无互操作性被认为是有用的。当供应商 ID 是零时，方法类型 254 也能够被用于提供扩展的 IETF 方法类型空间。在类型值 1-191 被分配后，方法类型值 256-4294967295 可能被分配，同时也是根据制定专家的建议和规范的要求。

方法类型 255 被分配用于测试，像在一个永久类型被分配前，测试新的 EAP 方法。