802.11i

**摘要：**本修正案定义了IEEE802.11的安全机制，其中包括WEP的定义，以便与原始标准IEEEStd802.11,1999版向后兼容。该修订定义了TKIP和CCMP，它们提供了比WEP提供的更强大的数据保护机制。它将安全关联的概念引入IEEE802.11，并定义了称为4次握手和组密钥握手的安全关联管理协议。此外，它还规定了IEEE802.11LAN如何利用IEEE802.1X进行身份验证。

**关键词：**AES，认证，CCM，CCMP，机密性，对策，数据真实性，EAPOL-Key，4次握手，组密钥握手，IEEE802.1X，密钥管理，密钥混合，Michael，RC4，重放保护，健壮的安全网络，RSN，安全，安全协会，TKIP，WEP

**IEEE标准**文档是在IEEE协会和IEEE标准协会（IEEESA）标准委员会的标准协调委员会内开发的。IEEE通过美国国家标准协会批准的共识开发流程制定其标准，该协会汇集了代表不同观点和兴趣的志愿者，以实现最终产品。志愿者不一定是研究所的成员，无偿提供服务。虽然IEEE负责管理流程并制定规则以促进共识开发流程的公平性，但IEEE不会独立评估，测试或验证其标准中包含的任何信息的准确性。

使用IEEE标准完全是自愿的。对于因发布，使用或依赖此标准或任何其他IEEE标准文件而直接或间接导致的任何性质的任何性质的人身伤害，财产或其他损害，IEEE不承担任何责任，无论是特殊的，间接的，后果性的还是补偿性的。

IEEE不保证或代表此处所含材料的准确性或内容，并明确否认任何明示或暗示的担保，包括任何适销性或适用于特定用途的暗示担保，或此处包含的材料的使用是免费的从专利侵权。IEEE标准文档“按原样”提供。

IEEE标准的存在并不意味着没有其他方式来生产，测试，测量，购买，营销或提供与IEEE标准范围相关的其他商品和服务。此外，在标准被批准和发布时表达的观点可能通过现有技术的发展和从标准的用户收到的评论而发生变化。每个IEEE标准至少每五年进行一次审查，以进行修订或重申。当一份文件超过五年并且没有得到重申时，可以合理地得出结论，其内容尽管仍具有某些价值，但并不完全反映现有技术水平。提醒用户检查以确定他们拥有任何IEEE标准的最新版本。

在发布和提供此文档时，IEEE不会为任何人或实体建议或代表任何人或实体提供专业或其他服务。IEEE也不承担任何其他人或实体对另一个人所欠的任何义务。任何使用此标准的人以及任何其他IEEE标准文件，都应依赖主管专业人员的建议，以确定在任何特定情况下合理谨慎的行使。

**解释：**有时可能会出现与特定应用相关的标准部分含义的问题。当解释的需要引起IEEE的注意时，研究所将采取行动准备适当的答复。由于IEEE标准代表了相关利益的共识，因此重要的是确保任何解释都得到了利益平衡的同意。出于这个原因，IEEE及其协会和标准协调委员会的成员无法对口译请求作出即时回应，除非此事已经得到正式审议。在讲座，座谈会，研讨会或教育课程中，提供IEEE标准信息的个人应明确表明他或她的观点应被视为该个人的个人观点，而不是IEEE的正式立场，解释或解释。

无论是否隶属于IEEE，欢迎任何有兴趣的人士对修订IEEE标准的评论。对文件变更的建议应采用拟议的文本变更形式，并附上适当的支持性意见。

**介绍**

[此介绍不是IEEEStd802.11i™-2004，IEEE标准信息技术-系统之间的电信和信息交换-本地和城域网特定要求-第11部分：无线媒体访问控制（MAC）和物理层的一部分（PHY）规范-修订件6：媒体访问控制（MAC）安全增强。）]

在本修订中定义了超出由基本标准IEEEStd802.11,1999版的有线等效保密（WEP）机制提供的那些特征和能力的IEEE802.11介质访问控制（MAC）的增强安全服务和机制。此修订保留了WEP功能，以便向后兼容现有的IEEE802.11设备，但WEP已弃用，以支持本修订中提供的新安全功能。

**专利**

需要注意的是，本标准的实施可能需要使用专利权所涵盖的主题。通过公布本标准，对与此相关的任何专利权的存在或有效性不作任何立场。IEEE不负责识别可能需要许可证才能实施IEEE标准的专利或专利申请，或者对提请其注意的那些专利的法律效力或范围进行调查。专利持有人或专利申请人已提交保证声明，表明其将根据这些权利授予许可，无需赔偿或以合理的费率和非歧视性，合理的条款和条件向希望获得此类许可的申请人提供。IEEE未对专利持有人或专利申请人提供的许可协议的费率，条款和条件的合理性作出任何陈述。更多信息可从IEEE标准部门获得。

# 1.概述

## 1.2目的

更改1.2的第五个子弹如下：

- 描述了通过无线介质（WM）传输的用户信息的隐私保密性以及符合IEEE802.11标准的设备的认证的要求和程序。

# 2.规范性参考文献

# 3.定义

**删除“3.40隐私”定义。**

**更改“有线等效隐私（WEP）”的定义如下：**

**3.49有线等效保密（WEP）：**IEEE802.11规定的可选加密机密性算法，可用于提供主观上等同于不使用加密技术的有线局域网（LAN）介质的机密性的数据机密性。增强隐私保密性**。**

**按字母顺序将以下定义插入第3条，必要时重新编号：**

**3.63附加认证数据（AAD）：**未加密但受密码保护的数据。

**3.64认证，授权和计费（AAA）密钥：**请求方与认证服务器（AS）之间联合协商的密钥信息。该密钥信息通过安全信道从AS传输到Authenticator。成对主密钥（PMK）可以从AAA密钥导出。

**3.65身份验证和密钥管理（AKM）套件：**一套AKM套件选择器。

**3.66认证服务器（AS）：**向认证者提供认证服务的实体。该服务根据请求者提供的凭证确定请求者是否有权访问由身份验证者提供的服务。（IEEEP802.1X-REV6）

**3.67Authenticator：**位于点对点LAN网段一端的实体，便于对连接到该链路另一端的实体进行身份验证。（IEEEP802.1X-REV）

**3.68认证者地址（AA）：**IEEE802.1X认证者媒体访问控制（MAC）地址。

**3.69授权：**明确允许。

**3.70bigendian：**对于给定的多八位字节数字表示，最重要的八位字节具有最低地址的概念。

**3.71密码套件：**一组一种或多种算法，旨在提供数据机密性，数据真实性或完整性和/或重放保护。

**3.72具有CBC-MAC的3.72计数器模式（CTR）[具有消息认证码（MAC）的密码块链接（CBC）]（CCM）：**使用CTR和使用CBC-MAC的数据源真实性提供机密性的对称密钥块密码模式。

**3.73解封装：**从受保护的帧中恢复未受保护的帧。

**3.74解封装：**通过解封装封装帧生成明文数据的过程。

**3.75EAPOL-Key确认密钥（KCK）：**用于完整性检查EAPOL-Key帧的密钥。

**3.76EAPOL-Key加密密钥（KEK）：**用于加密EAPOL-Key帧中的密钥数据字段的密钥。

**3.77封装：**从未受保护的帧构造受保护的帧。

**3.78封装：**从明文数据生成加密有效载荷的过程。这包括密文以及数据接收器所需的任何相关密码状态，例如初始化矢量（IV），序列号，消息完整性代码（MIC），密钥标识符。

**3.79 4次握手：**由此修订定义的成对密钥管理协议。它确认双方共同拥有成对主密钥（PMK）并分发组临时密钥（GTK）。

**3.80组：**无线网络中的实体，例如接入点（AP）及其关联站（STA），或独立基本服务集（IBSS）网络中的所有STA。

**3.81组密钥握手：**本修订定义的组密钥管理协议。它仅用于向与本地站（STA）已经形成安全关联的对等体发布新的组临时密钥（GTK）。

**3.82组主密钥（GMK）：**可用于导出组临时密钥（GTK）的辅助密钥。

**3.83组临时密钥（GTK）：**由广播/组播源分配的随机值，用于保护来自该源的广播/多播媒体访问控制（MAC）协议数据单元（MPDU）。GTK可以从组主密钥（GMK）导出。

**3.84组临时密钥安全关联（GTKSA）：**通过组密钥握手或4次握手从成功的组临时密钥（GTK）分发交换产生的上下文。

**3.85 IEEE802.1X身份验证：**由IEEE802.1X协议传输的可扩展身份验证协议（EAP）身份验证。

**3.86密钥计数器：**256位（32字节）计数器，用于伪随机函数（PRF）以生成初始化向量（IV）。每个站（STA）存在一个对该STA是全局的单个密钥计数器。

**3.87密钥数据封装（KDE）：**EAPOL-KeyData字段中除信息元素之外的数据的格式。

**3.88密钥管理服务：**在健壮的安全网络（RSN）内分发和管理加密密钥的服务。

**3.89小端：**对于给定的多八位字节数字表示，最不重要的概念八位字节的地址最低。

**3.90活跃度：**对等体实际参与此通信实例的演示。

**3.91消息完整性代码（MIC）：**由对称密钥加密函数生成的值。如果输入数据被改变，在不知道对称密钥的情况下不能正确地计算新值。因此，秘密密钥保护输入数据免于不可检测的改变。传统上称为a消息认证码（MAC），但首字母缩略词MAC已经被保留用于其他含义修正。

**3.92 Michael：**临时密钥完整性协议（TKIP）的消息完整性代码（MIC）。

**3.93 nonce：**一个不能与给定密钥一起重用的值，包括所有时间内系统的所有重新初始化。

**3.94成对：**彼此相关联的两个实体，例如，接入点（AP）和相关联的站（STA），或独立基本服务集（IBSS）网络中的一对STA。这个术语习惯了描述仅在成对关联中的两个实体之间共享的键的关键层次结构。

**3.95成对主密钥（PMK）：**此修订中使用的最高顺序密钥。PMK可能是源自可扩展认证协议（EAP）方法或可以直接从预共享密钥（PSK）获得。

**3.96成对主密钥安全关联（PMKSA）：**由成功的IEEE产生的上下文对等体和身份验证服务器（AS）之间或预共享密钥之间的802.1X身份验证交换（PSK）。

**3.97成对临时密钥（PTK）：**从成对主密钥（PMK），认证者地址（AA），请求者地址（SPA），Anthenticatornonce（ANonce）和Supplicantnonce派生的值（SNonce）使用伪随机函数（PRF）并将其分成多达五个密钥，即时间加密密钥，两个时间消息完整性代码（MIC）密钥，EAPOL-密钥加密密钥（KEK），EAPOL-Key确认密钥（KCK）。

**3.98成对临时密钥安全关联（PTKSA）：**成功导致的上下文对等体和认证体之间的4次握手交换。

**3.99密码：**用于证实用户身份的秘密文本字符串。

**3.100每帧加密密钥：**为每个媒体访问控制构建的唯一加密密钥（MAC）协议数据单元（MPDU），由某些IEEE802.11安全协议使用。

**3.101每帧序列计数器：**对于临时密钥完整性协议（TKIP），使用的计数器作为推导每帧加密密钥的随机数。对于CCM[具有密码块链接的计数器模式（CTR）（CBC）和消息认证码（MAC）]协议（CCMP），每帧初始化矢量（IV）。

**3.102预鲁棒安全网络关联（pre-RSNA）：**一对站（STA）使用的关联类型，如果它们之间建立认证或关联的过程不包括四次握手。

**3.103预鲁棒安全网络关联（pre-RSNA）设备：**无法创建强大安全网络关联（RSNA）的设备。

**3.104预共享密钥（PSK）：**通过外部方法分发给系统中的单元的静态密钥这项修正案的范围总是采用一些带外手段。

**3.105伪随机函数（PRF）：**一种hash各种输入以导出伪随机函数值的函数。要为伪随机值添加活​​泼度，nonce应该是输入之一。

**3.106健壮安全网络（RSN）：**一种安全网络，仅允许创建强大的安全性网络协会（RSNA）。RSN可以通过RSN信息元素中的指示来识别组密码套件指定的Beacon帧的（IE）不是有线等效保密（WEP）。

**3.107鲁棒安全网络关联（RSNA）：**一对站使用的关联类型（STA）如果在它们之间建立认证或关联的过程包括4次握手。请注意，一对设备存在RSNA本身并不能提供强大的安全性。当网络中的所有设备都使用RSNA时，会提供强大的安全性。

**3.108 robust-security-network-association-（RSNA-）能力设备：**能够的站（STA）创建RSNA。由于配置，这种设备可以使用pre-RSNA。请注意，RSNA能够做到并不意味着完全符合RSNA协议实施一致性声明（PICS）。已升级为支持临时密钥完整性协议（TKIP）的传统设备可以是RSNA，但如果它不支持CCM，则不符合PICS[计数器模式（CTR）具有消息认证码（MAC）的密码块链接（CBC）]协议（CCMP）。

**3.109 robust-security-network-association-（RSNA-）设备：**一个站（STA）支持RSNA和dot11RSNAEnabled设置为TRUE。

**3.110鲁棒安全网络关联（RSNA）密钥管理：**包括密钥管理的密钥管理4次握手，组密钥握手和STAKey握手。

**3.111 securitynetwork（基础服务集）：**一种基本服务集（BSS），其中启动BSS的站（STA）通过在Beacon帧中包括鲁棒安全网络（RSN）信息单元来提供关于BSS的安全能力和配置的信息。

**3.112 selector：**指定IEEE802.11管理消息信息单元中的列表成分的项。

**3.113 STAKey：**用于保护基础设施基本服务集（BSS）中的直接站到站（STA到STA）通信的对称密钥。

**3.114 STAKey握手：**STAKey密钥管理协议，用于向具有成对临时密钥安全关联（PTKSA）和接入点（AP）的站（STA）发出新的STAKey。

**3.115 STAKey安全关联（STAKeySA）：**基础结构基本服务集（BSS）中直接站到站（STA-to-STA）通信的安全上下文。STAKeySA包括STAKey。

**3.116请求者：**位于点对点LAN网段一端的实体，由连接到该链路另一端的Authenticator进行身份验证。（IEEEP802.1X-REV）

**3.117请求者地址（SPA）：**请求者的媒体访问控制（MAC）地址。

**3.118 temporal-encryption-key（时间加密密钥）：**成对临时密钥（PTK）或组临时密钥（GTK）的一部分，用于直接或间接加密媒体访问控制（MAC）协议数据单元（MPDU）中的数据。

**3.119 temporal-key：**时间加密密钥和时间消息完整性代码（MIC）密钥的组合。

**3.120时间消息完整性代码（MIC）密钥：**用于确保介质访问控制（MAC）服务数据单元（MSDU）或MAC协议数据单元（MPDU）的完整性的临时密钥部分。

**3.121过渡安全网络（TSN）：**一种安全网络，允许创建预先强大的安全网络关联（RSNA前）和RSNA。可以通过Beacon帧的鲁棒安全网络（RSN）信息元素中的指示来识别TSN，其中使用的组密码套件是有线等效保密（WEP）。

# 4.缩略语和首字母缩写词

**在字母顺序中将以下缩写插入第4条：**

AA 身份验证员地址

AAA 身份验证，授权和记帐

AAD 附加认证数据

AES 高级加密标准

AKM 身份验证和密钥管理

AKMP 认证和密钥管理协议

ANonce Authenticator-nonce

ARP 地址解析协议

AS 认证服务器

CBC 密码块链接

CBC-MAC 密码块链接和消息验证码

CCM 点击率与CBC-MAC

CCMP CTR与CBC-MAC协议

CTR 计数器模式

EAP 可扩展身份验证协议（IETFRFC3748）

EAPOL LAN上的可扩展身份验证协议（IEEEP802.1X-REV）

GMK 组主密钥

Gnonce 组nonce

GTK 组临时密钥

GTKSA 组临时密钥安全关联

ICMP Internet控制消息协议

KCK EAPOL-Key确认键

KDE 密钥数据封装

KEK EAPOL-Key加密密钥

LFSR 线性反馈移位寄存器

MIC 消息完整性代码

NTP 网络时间协议（IETFRFC1305[B12]7）

OUI 组织上唯一的标识符

PAE 端口访问实体（IEEEP802.1X-REV）

PMK 成对主密钥

PMKID 成对主密钥标识符

PMKSA 成对主密钥安全关联

PN 包号

PRF 伪随机函数

PRNG 伪随机数发生器

PSK 预先关键

PTK 成对临时密钥

PTKSA 成对临时密钥安全关联

RADIUS 远程认证拨入用户服务（IETFRFC2865[B14]）

RSC 广播/多播发送序列计数器

RSN 强大的安全网络

RSNA 强大的安全网络关联

SNAP 子网访问协议

SNonce Supplicantnonce

SPA 申请人地址

TKIP 临时密钥完整性协议

TSC TKIP序列计数器

TSN 过渡安全网络

TTAK TKIP混合发送地址和密钥

UCT 无条件转让

# 5.一般说明

**5.1架构的一般描述**

**5.1.1无线LAN系统的不同之处**

**5.1.1.4与其他IEEE802®层的交互**

**在5.1.1.4末尾插入以下段落：**

在强大的安全网络协会（RSNA）中，IEEE802.11提供保护数据帧的功能，IEEE802.1X提供认证和受控端口，IEEE802.11和IEEE802.1X协作提供密钥管理。RSNA中的所有站（STA）都具有处理这些服务的相应IEEE802.1X实体。该修订定义了RSNA如何利用IEEE802.1X访问这些服务。

**在5.1.1.4之后，插入5.1.1.5：**

**5.1.1.5与非IEEE802协议的交互**

RSNA利用非IEEE802协议进行身份验证和密钥管理（AKM）服务。其中一些协议由其他标准组织定义，例如Internet工程任务组（IETF）。

**5.2IEEE802.11架构的组件**

**5.2.2分配系统（DS）概念**

**在5.2.2.1之后，插入5.2.2.2：**

**5.2.2.2RSNA**

除有线等效保密（WEP）和IEEE802.11认证外，RSNA还定义了许多安全功能。这些功能包括：

-增强的STA认证机制

-密钥管理算法

-加密密钥建立

-增强型数据封装机制，称为CTR[计数器模式]，带有CBC-MAC[带有消息认证码（MAC）的密码块链接（CBC）]协议（CCMP），以及可选的临时密钥完整性协议（TKIP）

**RSNA依赖于IEEE802.11架构外部的几个组件。**

第一个组件是IEEE802.1X端口访问实体（PAE）。PAE存在于RSNA中的所有STA上，并控制与媒体访问控制（MAC）之间的数据转发。接入点（AP）始终实现Authenticator-PAE并实现EAP-Authenticator角色，并且STA始终实现Supplicant-PAE并实现可扩展认证协议（EAP）对等角色。在独立的基本服务集（IBSS）中，每个STA实现认证者PAE和请求者PAE以及EAP认证者和EAP对等角色。

**PAE-端口访问实体，可以支持EAP认证**

第二个组件是认证服务器（AS）。AS可以验证RSNA本身的元素，即非APSTA;和AP可以提供RSNA元素可以用来相互认证的材料。AS通过IEEE802.1X Authenticator（认证者）与每个STA上的IEEE802.1X Supplicant（请求者）进行通信，使STA能够通过AS进行身份验证，反之亦然。RSNA依赖于使用支持AS和STA的相互认证的EAP方法。在某些应用中，AS可以集成到与AP相同的物理设备中，或者集成到IBSS中的STA中。

**5.3逻辑服务接口**

**在5.3中的架构服务列表中更改项目g），如下所示：**

**g）隐私保密**

**5.3.1站点服务（SS）**

**更改5.3.1中SS的列表中的项目c），如下所示：**

**c）隐私保密**

**5.4服务概述**

**5.4.2支持分发服务的服务**

**5.4.2.2关联**

**在5.4.2.2的第二段之后插入以下段落：**

在强大的安全网络（RSN）中，这是不同的。在RSNA中，IEEE802.1X端口确定何时允许跨IEEE802.11链路的数据流量。单个IEEE802.1X端口映射到一个关联，每个关联映射到IEEE802.1X端口。IEEE802.1X端口由IEEE802.1X受控端口和IEEE802.1X非受控端口组成。阻止IEEE802.1X受控端口在两个STA之间传递通用数据流量，直到IEEE802.1X身份验证过程通过IEEE802.1X非受控端口成功完成。AKM成功完成后，将启用数据保护以防止未经授权的访问，并且IEEE802.1X受控端口将解锁以允许受保护的数据流量。IEEE802.1X请求者和认证者通过IEEE802.1X非受控端口交换协议信息。预计大多数其他协议交换将使用IEEE802.1X受控端口。但是，给定协议可能需要绕过授权功能并使用IEEE802.1X非受控端口。

**注-**有关受控端口和非受控端口的讨论，请参阅IEEEP802.1X-REV。

**5.4.2.3重新关联**

**在5.4.2.3的末尾插入以下段落：**

在重新关联期间，没有提供设施来移动RSNA。因此，旧的RSNA将被删除，并且需要构建新的RSNA。

**将5.4.3的标题更改如下：**

**5.4.3访问控制和机密性控制服务**

**将5.4.3的第二段改为如下：**

提供两种服务以使IEEE802.11功能符合有线局域网（LAN）假设：认证和隐私机密性。使用身份验证而不是有线媒体物理连接。隐私数据机密性用于提供封闭有线媒体的机密方面。

**在5.4.3的末尾插入以下段落：**

在不支持RSNA的无线LAN（WLAN）中，定义了两种服务，即身份验证和机密性。使用IEEE802.11认证代替有线媒体物理连接。定义WEP加密以提供封闭有线媒体的机密性方面。

RSNA使用IEEE802.1X认证服务以及TKIP和CCMP来提供访问控制。IEEE802.11站管理实体（SME）通过IEEE802.1X EAPOL-Key帧的交换来提供密钥管理。RSN密钥管理与TKIP和CCMP一起提供机密性和数据完整性。

**5.4.3.1认证**

**用以下内容替换5.4.3.1的文本：**

IEEE802.11身份验证在IEEE802.11 STA之间的链路级别运行。IEEE802.11不提供端到端（消息源到消息目的地）或用户到用户认证。

IEEE802.11尝试通过身份验证服务控制LAN访问。IEEE802.11身份验证是SS。在扩展服务集（ESS）和IBSS网络中，所有STA可以使用该服务来向与其通信的STA建立其身份。如果在两个STA之间尚未建立双方可接受的认证级别，则不应建立关联。

IEEE802.11定义了两种认证方法：开放系统认证和共享密钥认证。开放系统身份验证允许任何STA进入DS。共享密钥身份验证依赖于WEP来演示WEP加密密钥的知识。IEEE802.11认证机制还允许定义新的认证方法。

RSNA还支持基于IEEE802.1X或预共享密钥（PSK）的身份验证。IEEE802.1X身份验证利用EAP相互验证STA和AS。该修订未规定必须实施的EAP方法。有关IEEE802.11 IBSS中IEEE802.1X身份验证和PSK使用情况的说明，请参见8.4.4。

在RSNA中，IEEE802.1X请求者和认证者通过IEEE802.1X非受控端口交换协议信息。阻止IEEE802.1X受控端口在两个STA之间传递通用数据流量，直到IEEE802.1X身份验证过程通过IEEE802.1X非受控端口成功完成。

尽管开放系统认证在RSNA-IBSS中是可选的，但开放系统认证算法用于基本服务集（BSS）和IBSS-RSNA。RSNA不允许使用共享密钥身份验证。

附件D中提供了管理信息库（MIB）功能，以支持标准化认证方案。

STA可以在任何给定时刻与许多其他STA进行认证。

**5.4.3.2解除认证**

**将5.4.3.2的案文改为如下：**

当要终止现有的OpenSystem或SharedKey身份验证时，将调用deauthentication服务。取消身份验证是SS。

在ESS中，由于身份验证是关联的先决条件，因此取消身份验证的行为将导致站点解除关联。验证方（非APSTA或AP）可以调用解除验证服务。取消身份验证不是请求;这是一个通知。任何一方均不得拒绝取消认证。当AP向关联的STA发送解除认证通知时，该关联也应终止。

在RSNESS中，需要开放系统身份验证。在RSNESS中，取消身份验证会导致终止未经身份验证的站点的任何关联。它还导致该STA被禁用的IEEE802.1X受控端口，并删除成对临时密钥安全关联（PTKSA）。解除认证通知通过MAC层提供给IEEE802.1X。

在RSNA中，解除身份验证还会破坏STA中存在的任何相关PTKSA，组临时密钥安全关联（GTKSA）和STAKey安全关联（STAKeySAs），并关闭关联的IEEE802.1X受控端口。如果未启用成对主密钥（PMK）高速缓存，则deauthentication还会破坏从中派生已删除PTKSA的成对主密钥安全关联（PMKSA）。

在RSNIBSS中，开放系统身份验证是可选的，但STA需要识别Deauthentication帧。取消身份验证会导致IEEE802.1X受控端口被禁用，并删除PTKSA。

**更改5.4.3.3的标题和文本如下：**

**5.4.3.3隐私保密**

在有线局域网中，只有物理连接到线路的站点可能听到可以发送或接收LAN流量。使用无线共享介质时，情况并非如此。任何符合IEEE802.11标准的STA可以听到可以接收范围内的所有类似PHY IEEE802.11流量并且可以发送到范围内的任何其他IEEE802.11 STA。因此，单个无线链路（没有隐私机密性）与现有有线LAN的连接可能严重降低有线LAN的安全级别。

为了将无线LAN的功能安全性提升到有线LAN设计中隐含的水平，IEEE802.11提供了加密保护消息内容的能力。此功能由隐私保密服务提供。隐私保密是SS。

IEEE802.11规定了可选的隐私算法WEP，旨在满足有线LAN“等效”隐私的目标。该算法不是为最终安全而设计的，而是“至少与线路一样安全”。有关详细信息，请参阅第8章。

IEEE802.11使用WEP机制（参见第8章）来执行消息的实际加密。提供MIB功能以支持WEP。

请注意，只能为数据帧和某些身份验证管理帧调用隐私。所有站点最初都是“明确地”启动，以便建立身份验证和隐私服务。

IEEE802.11提供三种加密算法来保护数据流量：WEP，TKIP和CCMP。WEP和TKIP基于RC48算法，CCMP基于高级加密标准（AES）。为STA提供了一种装置，用于选择要用于给定关联的算法。

所有IEEE802.11 STA的默认隐私保密状态是“明确的”。如果未调用隐私保密服务，则所有消息都应以未加密的方式发送。如果该默认策略对于一方或另一方不是不可接受的，则LLC实体之间的数据帧不能成功通信。发件人，不得发送数据帧;如果策略对接收方不可接受，它应丢弃任何接收的数据帧。未加密的未受保护的数据帧在配置为强制性隐私机密性的站点接收，以及使用接收站不可用的密钥的保护加密数据帧，在没有指示逻辑链路控制（LLC）的情况下被丢弃（或者没有指示分发服务在在AP处接收的“ToDS”帧的情况。这些帧在WM[如果没有帧检查序列（FCS）错误的情况下接收]上被确认，以避免在重试时浪费WM带宽。

**在5.4.3.3之后，插入5.4.3.4至5.4.3.6：**

**5.4.3.4密钥管理**

增强的机密性，数据身份验证和重播保护机制需要新的加密密钥。本修正案中定义的程序通过称为4次握手和组密钥握手的协议提供新密钥。

**5.4.3.5数据来源真实性**

数据源真实性机制定义了一种装置，通过该装置，接收数据帧的STA可以确定哪个STA发送了MAC协议数据单元（MPDU）。RSNA中需要此功能以防止一个STA伪装成不同的STA。该机制是为使用CCMP或TKIP的STA提供的。

数据源真实性仅适用于单播数据帧。协议不保证广播/多播数据帧的数据源真实性，因为这不能使用对称密钥来实现，并且公钥方法的计算成本太高。

**5.4.3.6重播检测**

重放检测机制定义了一种装置，通过该装置，从另一个STA接收数据帧的STA可以检测数据帧是否是未授权的重传。该机制是为使用CCMP或TKIP的STA提供的。

**5.6 ESS和IBSSLAN之间的差异**

**在5.6的末尾插入以下段落：**

在IBSS中，每个STA必须实施自己的安全策略。在ESS中，AP可以跨所有STA实施统一的安全策略。

**5.7支持服务的消息信息内容**

**更改5.7.5的标题和文本如下：**

**5.7.5隐私保密**

对于STA调用WEP隐私机密性算法（由MLME-SETKEYS控制，见第10章，或相关的MIB属性，见第11章），隐私机密性服务导致MPDU加密选择机密性算法并设置WEP帧headerProtectedFrame位适当（见第7章）。

**5.7.6认证**

**更改5.7.6中的第一段如下：**

对于STA使用开放系统或共享密钥认证与另一STA进行认证，认证服务使得交换一个或多个认证管理帧。帧的确切顺序及其内容取决于所调用的身份验证方案。对于所有这些认证方案，在管理帧体内识别认证算法。

**5.7.7取消认证**

**更改5.7.7中的第一段如下：**

对于STA使使用开放系统或共享密钥身份验证建立的活动身份验证无效，将发送以下消息：

**5.8参考模型**

**将图11替换为以下内容：**

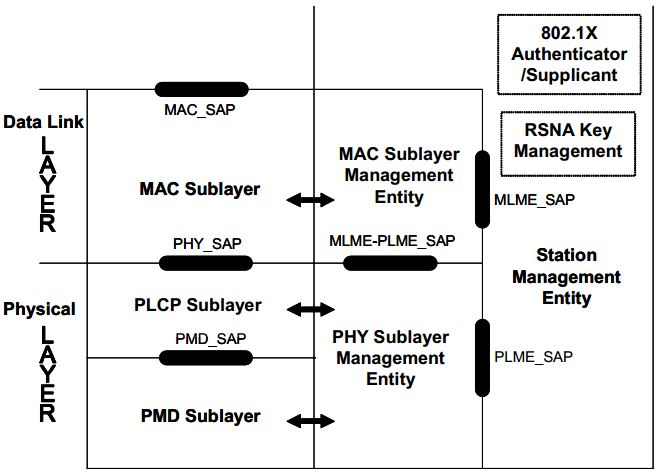


图11-本标准涵盖的ISO/IEC基本参考模型的部分

**在5.8结尾插入以下段落：**

IEEE802.1X请求者/认证者与图11中所示的SME之间存在接口。该接口在IEEEP802.1X-REV中描述。

**在5.8之后，插入5.9到5.9.5并根据需要重新编号：**

**5.9IEEE802.11和IEEE802.1X**

RSNA依靠IEEE802.1X提供认证服务，并使用8.5中定义的IEEE802.11密钥管理方案。IEEE802.1X访问控制机制适用于STA与AP之间的关联以及IBSSSTA与STA对等体之间的关系。AP的SME执行Authenticator，可选择执行Supplicant和AS角色。在ESS中，非APSTA的SME执行请求者角色。在IBSS中，STA的SME承担Supplicant和Authenticator角色，并且可以担任AS角色。

**5.9.1IEEE802.1X的IEEE802.11使用**

IEEE802.11依靠IEEE802.1X通过使用IEEE802.1X受控/非受控端口模型来控制DS和STA之间的MAC服务数据单元（MSDU）的流动。IEEE802.1X认证帧在IEEE802.11数据帧中传输，并通过IEEE802.1X非受控端口传递。阻止IEEE802.1X受控端口在两个STA之间传递通用数据流量，直到IEEE802.1X身份验证过程通过IEEE802.1X非受控端口成功完成。Supplicant和Authenticator都负责实现端口阻塞。一对STA之间的每个关联创建一对唯一的IEEE802.1X端口，并且相对于这些端口单独进行身份验证。

IEEE802.11依赖于IEEE802.1X和第8章中描述的4次握手和组密钥握手来建立和更改加密密钥。验证完成后建立密钥。密钥可能会因各种原因而发生变化，包括IEEE802.1X身份验证计时器到期，密钥泄露，危害危险或策略。

**5.9.2基础设施功能模型概述**

本子条款总结了RSN的系统设置和操作，有两种情况：使用IEEE802.1 XAS时和使用PSK时。对于ESS，AP包括认证者，并且每个关联的STA包括请求者。

**5.9.2.1AS的AKM操作**

使用IEEE802.1XAS时执行以下AKM操作：

a）在使用IEEE802.1X之前，IEEE802.11假定Authenticator和AS已建立安全信道。Authenticator和AS之间通道的安全性超出了本修正案的范围。在关联之前，必须将身份验证凭据分发给请求者和AS。

b）STA通过被动监控Beacon帧或通过主动探测发现AP的安全策略（如图11a所示）。如果使用IEEE802.1X身份验证，则当AP的身份验证器发送EAP-Request（如图11b所示）或STA的Supplicant发送EAPOL-Start消息时，EAP身份验证过程将启动。EAP身份验证帧通过Authenticator和Supplicant的未受控端口在Supplicant和AS之间传递。这在图11b中示出。

c）请求者和AS相互认证并生成PMK。PMK通过安全信道从AS发送到Authenticator。见图11b。

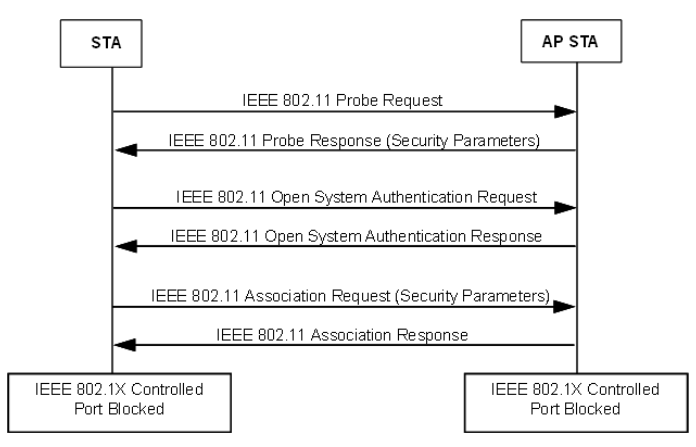


图11a-建立IEEE802.11关联

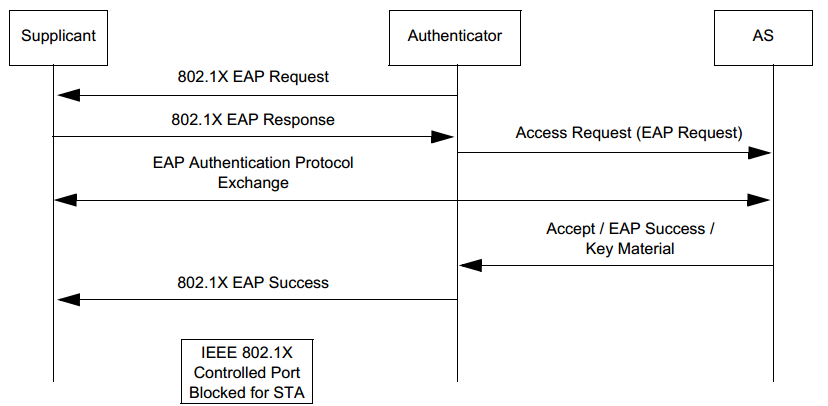


图11b-IEEE802.1XEAP身份验证

使用EAPOL-Key帧的4次握手由Authenticator启动，以执行以下操作：

-确认实时对等方拥有PMK。

-确认PMK是最新的。

-从PMK导出新的成对瞬态密钥（PTK）。

-将成对加密和完整性密钥安装到IEEE802.11中。

-将组临时密钥（GTK）和GTK序列号从Authenticator传输到Supplicant，并在STA中安装GTK和GTK序列号，如果尚未安装，则在AP中。

-确认密码套件选择。

成功完成4次握手后，Authenticator和Supplicant相互验证;并且IEEE802.1X受控端口未被阻止以允许一般数据流量。见图11c。

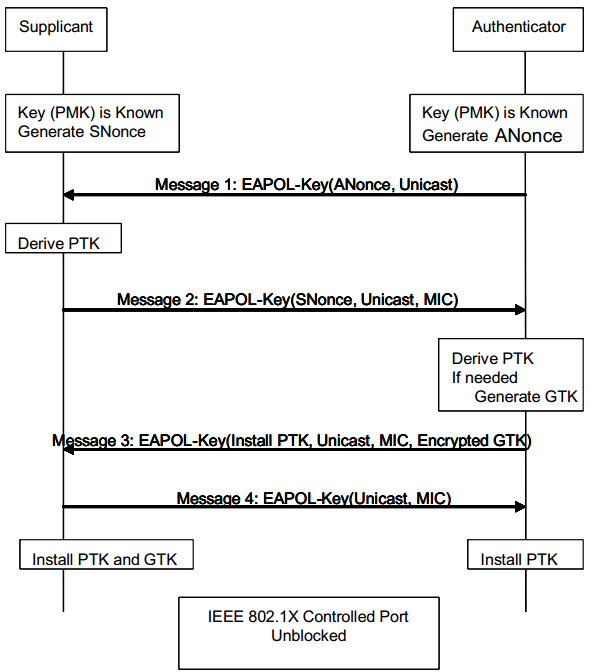


图11c-建立成对和组密钥

如果验证器稍后更改GTK，则它使用组密钥握手将新的GTK和GTK序列号发送给请求者，以允许请求者继续接收广播/多播消息，并且可选地，发送和接收单播帧。EAPOL-Key帧用于执行此交换。见图11d。

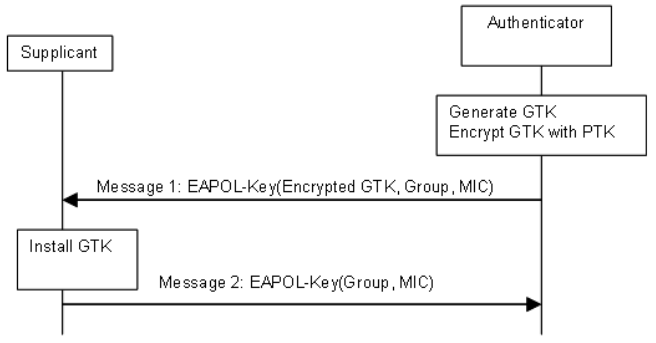


图11d-后续组密钥的交付

5.9.2.2使用PSK进行操作，当PMK为PSK时，执行以下AKM操作：

- STA通过被动监控Beacon帧或通过主动探测发现AP的安全策略（如图11a所示）。STA与AP关联并协商安全策略。PMK是PSK。

- 当存在AS时，使用EAPOL-Key帧的4次握手与IEEE802.1X认证一样使用。见图11c。

- GTV和GTK序列号从Authenticator发送到Supplicant，就像在AS情况下一样。参见图11c和图11d。

**5.9.3 IBSS功能模型描述**

本小节总结了IBSS中RSNA的系统设置和操作。IBSSRSNA在8.4.7中规定。

**5.9.3.1密钥用法**

在IBSS中，两个STA之间的单播数据帧用成对密钥保护。密钥是PTK的一部分，PTK是在4次握手期间派生的。

在IBSS中，广播/多播数据帧由诸如名为B1的密钥保护，该密钥由发送广播/多播帧的STA生成。为了允许其他STA解密广播/多播帧，必须将B1发送到IBSS中的所有其他STA。B1在EAPOL-Key帧中发送，在PTK的EAPOL-Key加密密钥（KEK）部分加密，并且受到PTK的EAPOL-Key确认密钥（KCK）部分的修改保护。

在IBSS中，STA的SME通过删除与该STA相关联的PTKSA来响应来自STA的Deauthentication帧。

**5.9.3.2 IBSS 4次握手示例**

在这个例子中（见图11e），有三个STA：S1，S2，S3。S1发送的广播/组播帧受B1保护;类似地，B2用于S2，B3用于S3。

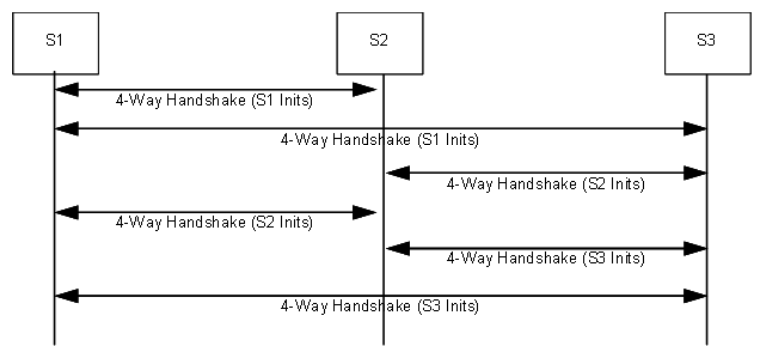


图11e-IBSS中的样本4次握手

对于站S2和S3来解密来自S1的广播/多播帧，必须将B1发送到S2和S3。最初使用4次握手并使用组密钥握手进行GTK更新。

从S1到S2的4次握手允许S1向S2发送广播/组播帧，但是不允许S2向S1发送广播/组播帧，因为S2具有不同的发送GTK。因此，S2需要启动到S1的4次握手以允许S1解密S2的广播/多播帧。类似地，S2还需要发起对S3的4次握手以使S3能够从S2接收广播/多播消息。

以类似的方式，S3需要用S1和S2完成4次握手以将B3传送到S1和S2。

在这个例子中，有六个4次握手。通常，NSTA请求者需要N（N-1）个4次握手。

**注-**原则上单个4次握手的KCK和KEK可用于两个方向的组密钥握手，但使用两个4次握手意味着认证器密钥状态机不需要在IBSS和ESS之间有所不同。

组密钥握手可用于将GTK发送到正确的STA。4次握手用于导出成对密钥并发送初始GTK。因为在IBSS中，在任何两个STA请求者和认证者之间存在两个4次握手，所以在任何两个STA之间使用的成对密钥来自具有较高MAC地址的STA认证者发起的4次握手（参见8.5.1for地址比较的概念）。用于组密钥握手的KCK和KEK是由发起组密钥握手的相同认证者发起的4次握手导出的KCK和KEK。

在IBSS中，当两个4次握手成功完成时，两个站之间存在安全链路。Supplicant和Authenticator4次握手状态机相互作用，因此在两个4次握手完成之前，不会设置IEEE802.1X变量portValid。

如果第四个STA进入范围并且其SME决定发起与三个对等体的安全关联，则其认证者与其他三个STA请求者中的每一个发起4次握手。类似地，IBSS中的原始三个STA认证器需要向第四STA请求者发起4次握手。STA从信标或探测响应帧获知对等STA是启用RSNA的并且对等的安全策略（例如，认证和密钥管理协议（AKMP）是PSK还是IEEE802.1X认证）。启动可能由于多种原因而开始：

1. 第四STA从MAC地址接收信标或探测响应帧，其尚未完成4次握手。

b）STA的SME从MAC地址接收MLME-PROTECTEDFRAMEDROPPED.indication原语，该原语尚未完成4次握手。这可以是由任何STA发送的多播/广播数据帧。如果SME想要建立与对等STA的安全关联，但不知道对等体的安全策略，则它应该向对等STA发送探测请求帧以找到其安全策略，然后建立与该对等体的安全关联。同伴STA。

c）STA的SME接收发送给STA的4次握手的消息1，因为发起者从该STA接收广播数据帧，信标帧或探测响应帧。如果STA收到4次握手，想要建立与对等STA的安全关联，但不知道对等体的安全策略，则应该向对等STA发送探测请求帧以找到其安全策略之前建立与对等STA的安全关联。

**5.9.3.3 IBSSIEEE802.1X示例**

使用IEEE802.1X身份验证时，每个STA都需要包含IEEE802.1X身份验证器和AS。STA从信标或探测响应帧获知对等STA是启用RSNA的并且对等的安全策略（例如，AKMP是PSK还是IEEE802.1X认证）。

每个STA的请求者将向其想要认证的每个其他站发送EAPOL-Start消息，并且每个STA的认证者将以其想要使用的凭证的身份进行响应。

当从MAC地址接收受保护的数据帧（通过MLMEPROTECTEDFRAMEDROPPED.indication原语指示），IEEE802.1X消息，信标帧或探测响应帧时，启动EAPOL-Start和EAP-Request/Identity消息。STA尚未完成IEEE802.1X身份验证。如果SME想要建立与对等STA的安全关联，但不知道对等体的安全策略，则它应该向对等STA发送探测请求帧以找到其安全策略，然后建立与该对等体的安全关联。同伴STA。

尽管图11f示出了串行化的两个IEEE802.1X交换，但它们可以交错出现。

**5.9.4认证者到AS协议**

Authenticator-to-AS身份验证定义超出了本修订的范围，但是，为了提供安全保证，协议必须支持以下功能：

a）Authenticator和AS之间的相互认证

b）Supplicant/AS认证的通道

c）以提供密钥源认证的方式将生成的密钥从AS传递给认证者的能力，确保密钥传输的完整性，并保护密钥对所有其他方的机密性。

合适的协议包括但不限于远程认证拨入用户服务（RADIUS）（IETFRFC2865[B14]）和Diameter（IETFRFC3588[B15]）。

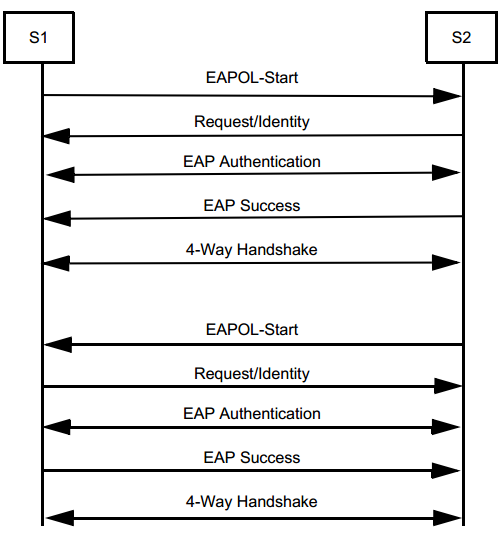


图11f-使用IEEE802.1X身份验证的示例

**5.9.5PMKSA缓存**

Authenticator和Supplicant可以缓存PMKSA，其中包括IEEE802.1X状态。可以出于任何原因随时从缓存中删除PMKSA。

STA可以在（重新）关联请求帧中提供PMK或PSK密钥标识符的列表。每个密钥标识符命名为PMKSA;PMKSA可能包含单个PMK。Authenticator在4次握手的消息1中指定所选的PMK或PSK密钥标识符。选择要包括在（重新）关联请求帧和4次握手的消息1中的密钥标识符超出了本修订的范围。

**6.MAC服务定义**

**6.1MAC服务概述**

**6.1.2安全服务**

**更改6.1.2的文本如下：**

IEEE802.11中的安全服务由认证服务和WEP，TKIP和CCMP机制提供。提供的安全服务范围仅限于站到站数据交换。由IEEE802.11 WEP，TKIP和CCMP实现提供的隐私机密性服务是MSDU的加密。出于本标准的目的，WEP，TKIP和CCMP被视为位于MAC子层内的逻辑服务，如参考模型中所示，图11. WEP，TKIP和CCMP服务的实际实现对于LLC和MAC子层之上的其他层。

WEP，TKIP和CCMP在IEEE802.11中提供的安全服务如下：

a）保密;

b）认证;

c）与层管理相结合的访问控制。

在认证交换期间，双方A和B如第8章所述交换认证信息。

WEP，TKIP和CCMP提供的MAC子层安全服务依赖于来自非第2层管理或系统实体的信息。管理实体通过一组MIB属性将信息传递给WEP。管理实体通过一组MAC子层管理实体（MLME）接口和MIB属性将信息传递给TKIP和CCMP;特别是，8.7中定义的TKIP和CCMP的决策树由MIB属性驱动。

**在6.1.3之后，插入6.1.4并根据需要重新编号：**

**6.1.4MAC数据服务架构**

MAC数据平面架构（即涉及传输全部或部分MSDU的过程）如图11g所示。在传输期间，MSDU经历以下部分或全部过程：在省电模式期间帧递送延迟，序列号分配，分段，加密，完整性保护和帧格式化。IEEE802.1X可能会阻止受控端口的MSDU。

在接收期间，接收的数据帧经历MPDU报头+循环冗余码（CRC）验证，重复删除，解密，碎片整理，完整性检查和重放检测的过程。如果未启用受控端口或MSDU不代表IEEE802.1X帧，则IEEE802.1X受控/非受控端口将丢弃MSDU。TKIP和CCMPMPDU帧顺序执行在解密之后发生，但在MSDU碎片整理之前;因此，如果MPDU故障到达，则碎片整理将失败。

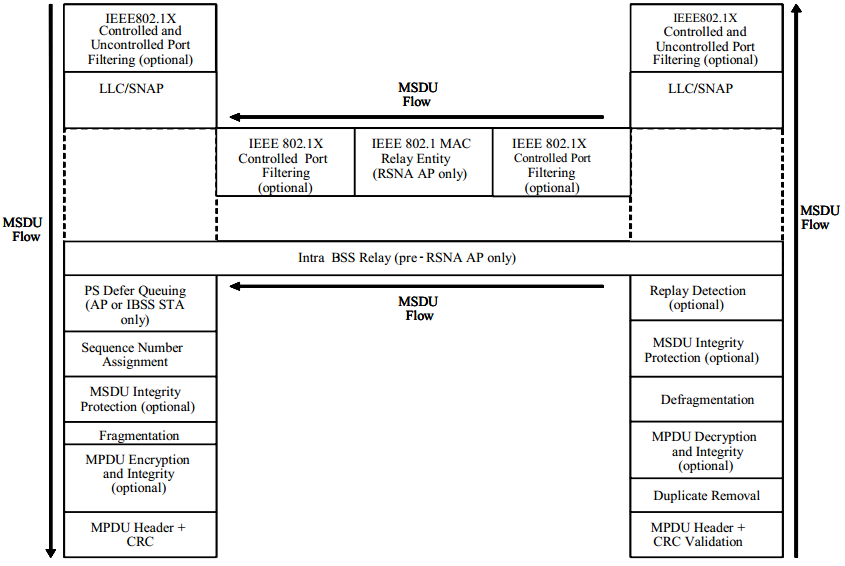


图11g-MAC数据平面架构

**7.帧格式**

**7.1MAC帧格式**

**7.1.3帧字段**

**7.1.3.1帧控制字段**

**更改7.1.3.1的文本如下：**

帧控制字段由以下子字段组成：协议版本，类型，子类型，目的DS，源DS，更多片段，重试，电源管理，更多数据，有线等效加密（WEP）保护帧和订单。帧控制字段的格式如图13所示。

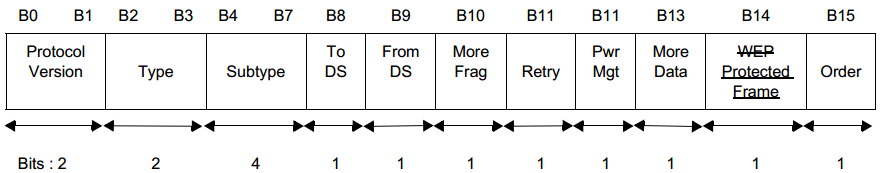


图13-帧控制字段

**更改7.1.3.1.9的标题和文本如下：**

**7.1.3.1.9WEP受保护帧字段**

WEP受保护的帧字段长度为1位。如果“帧主体”字段包含WEP已处理的加密封装算法的信息，则“受保护的帧”字段设置为1。WEPProtectedFrame字段仅在Data类型的数据帧内以及Management，subtypeAuthentication类型的管理帧中设置为1。所有其他帧中的WEPProtectedFrame字段都设置为0。当WEP位保护帧字段设置为1时，帧体字段按照8.2.5在数据帧中的定义进行扩展，帧体字段使用加密封装算法进行保护，并按照第8章的规定进行扩展。仅WEP允许作为子类型认证的管理帧的加密封装算法。

**7.2各帧类型的格式**

**7.2.2数据帧**

**更改7.2.2中字母列表后面的第一段，如下所示：**

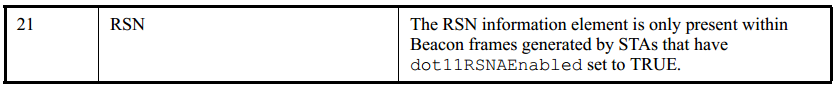
帧体由MSDU或其片段以及WEPIV和ICV安全头和尾部组成（当且仅当帧控制字段中的WEP受保护帧子字段被设置为1时）。在子类型NullFunction（无数据），CF-Ack（无数据），CFPoll（无数据）和CF-Ack+CF-Poll（无数据）的数据帧中，帧体为空（长度为0个八位字节）。

**7.2.3管理框架**

**7.2.3.1信标帧格式**

**在表5中插入订单21信息字段：**

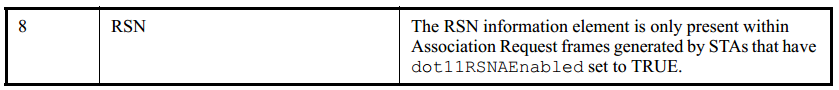
表5-信标帧体



**7.2.3.4关联请求帧格式**

**在表7中插入订单8信息字段：**

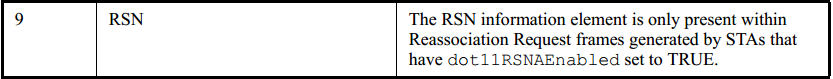
表7-关联请求帧体



**7.2.3.6重新关联请求帧格式**

**将订单9信息字段插入表9：**

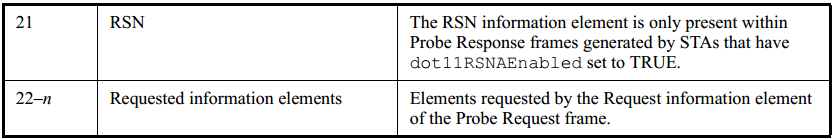
表9-重新关联请求框架主体



**7.2.3.9探测响应帧格式**

**在表12中插入订单21和22-n信息字段：**

表12-探针响应框体



**7.2.3.10认证帧格式**

**在7.2.3.10的第一句之后插入以下文字：**

只有身份验证算法设置为开放系统身份验证的身份验证帧才可以在RSNA中使用。如果在RSN关联之前调用共享身份验证，则RSNASTA不应关联。

**7.3管理框架主体组件**

**7.3.1固定字段**

**7.3.1.4能力信息字段**

**更改7.3.1.4中的第六和第七段如下：**

如果在BSS内交换的所有数据类型帧需要WEP加密数据机密性，则AP在发送的信标，探测响应，关联响应和重新关联响应管理帧内将Privacy子字段设置为1。如果不需要WEP加密数据机密性，则在这些管理帧内将AP设置为“隐私”子字段设置为0。

在RSNA中，ESS中的非APSTA在发送的关联和重新关联请求管理帧内将Privacy子字段设置为0。AP忽略收到的关联和重新关联请求管理帧中的隐私子字段。

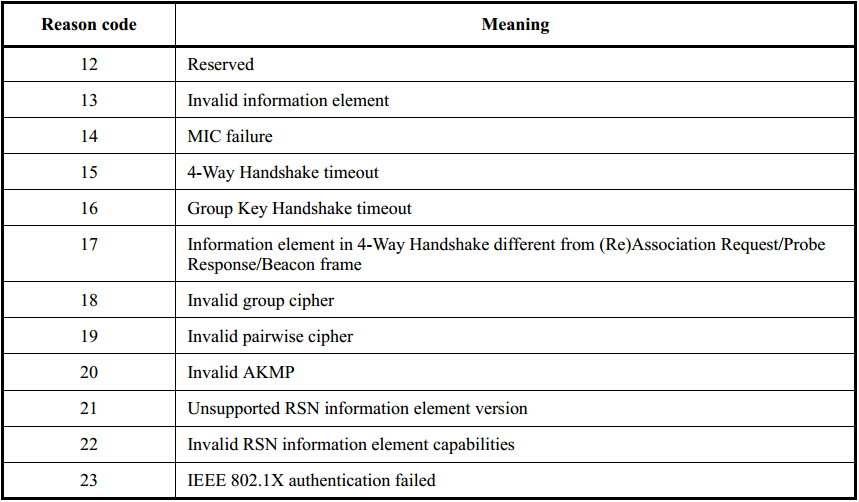
如果在IBSS内交换的所有数据类型帧需要WEP加密数据机密性，则IBSS内的STA将发送的信标或探测响应管理帧中的隐私子字段设置为1。如果不需要WEP加密数据机密性，则IBSS集中的STA将在这些管理帧内将Privacy子字段设置为0。

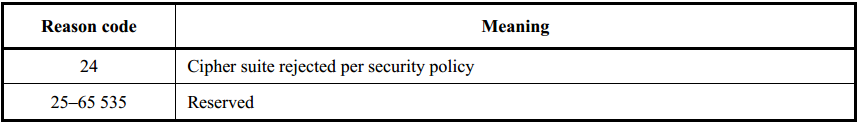
在信标和探测响应帧中包括RSN信息元素的STA应在包含RSN信息元素的任何帧中将Privacy子字段设置为1。

**7.3.1.7原因代码字段**

**插入原因代码12到24并更改表18中的最终保留原因代码行，如下所示：**

表18-原因代码

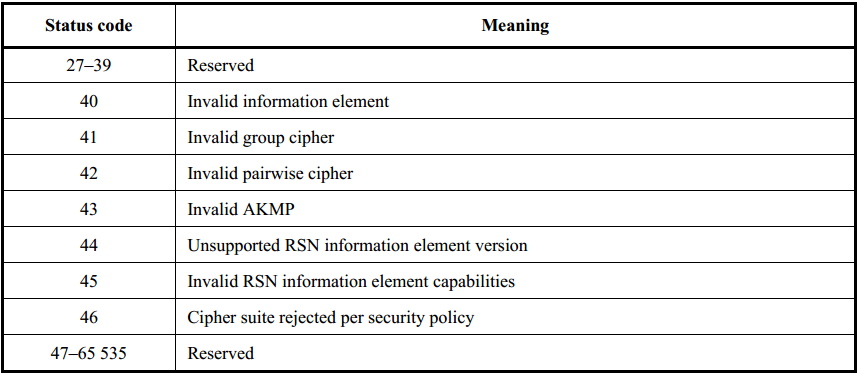




**7.3.1.9状态代码字段**

**插入原因代码27到46并更改表19中的最终保留原因代码行，如下所示：**

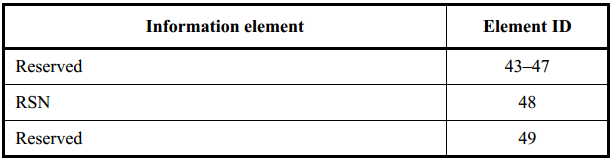
表19-状态代码



**7.3.2信息要素**

**替换表20中的元素标识符（ID）43到49，如下所示：**

表20-元素ID



**在7.3.2.24之后，插入7.3.2.25至7.3.2.25.5并根据需要重新编号表格和图表：**

**7.3.2.25RSN信息单元**

RSN信息元素包含认证和成对密码套件选择器，单组密码套件选择器，RSN能力字段，PMK标识符（PMKID）计数和PMKID列表。见图46ta。所有实现RSNA的STA都应支持该元素。RSN信息元素的大小受信息元素大小的限制，即255个八位字节。因此，成对密码套件，AKM套件和PMKID的数量是有限的。

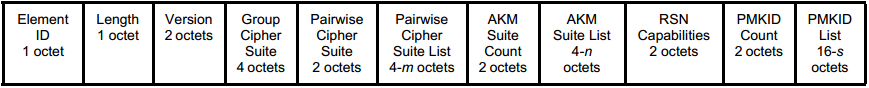


图46ta-RSN信息单元格式

在图46ta中，m表示成对密码套件计数，n表示AKM套件计数，s表示PMKID计数。

所有字段都使用7.1.1中的位约定。RSN信息元素应包含最多并包括Version字段。Version字段后面的所有字段都是可选的。如果没有任何可选字段，则不应包括任何后续字段。

元素ID应为48十进制（30十六进制）。

Length给出信息元素中信息字段（元素ID和长度字段后面的字段）中的八位字节数。

Version字段表示RSNA协议的版本号。STA支持的Version字段值的范围应该是连续的。“版本”字段的值0和2或更高版本是保留的。RSN版本1在本修订中定义。

注-以下代表示例信息元素：

802.1X身份验证，CCMP成对和组密码套件（不允许WEP-40，WEP104和TKIP）：

30，//信息元素id，48表示为十六进制值

14，//以八位字节为单位的长度，20表示为十六进制值

0100，//版本1

000FAC04，//CCMP作为组密码套件

0100，//成对密码套件计数

000FAC04，//CCMP作为成对密码套件

0100，//验证计数

000FAC01//802.1X验证

0000//没有能力

802.1X身份验证，CCMP成对和组密码套件（不允许WEP-40，WEP104和TKIP），支持预身份验证：

30，//信息元素id，48表示为十六进制值

14，//以八位字节为单位的长度，20表示为十六进制值

0100，//版本1

000FAC04，//CCMP作为组密码套件

0100，//成对密码套件计数

000FAC04，//CCMP作为成对密码套件

0100，//验证计数

000FAC01//802.1X验证

0100//预身份验证功能

802.1X身份验证，将GTK用于成对密码套件，WEP-40组密码套件，可选的RSN功能字段省略：

30，//信息元素id，48表示为十六进制值

12，//以八位字节为单位的长度，18表示为十六进制值

0100，//版本1

000FAC01，//WEP-40作为组密码套件

0100，//成对密码套件计数

000FAC00，//使用组密钥作为成对密码套件

0100，//验证计数

000FAC01//802.1X验证

802.1X身份验证，将CCMP用于成对密码套件，CCMP组密码套件，预身份验证和PMKID。

30，//信息元素id，48表示为十六进制值

26//以八位字节为单位的长度，38以十六进制值表示

0100，//版本1

000FAC04，//CCMP作为组密码套件

0100，//成对密码套件计数

000FAC04，//CCMP作为成对密码套件

0100，//验证计数

000FAC01//802.1X验证

0100//预身份验证功能

0100//PMKID计数

0102030405060708090A0B0C0D0E0F10//PMKID

**7.3.2.25.1密码套件**

GroupCipherSuite字段包含BSS用于保护广播/多播流量的密码套件选择器。

PairwiseCipherSuiteCount字段指示PairwiseCipherSuiteList字段中包含的成对密码套件选择器的数量。

PairwiseCipherSuiteList字段包含一系列密码套件选择器，用于指示RSN信息元素中包含的成对密码套件。

套件选择器的格式如图46tb所示。

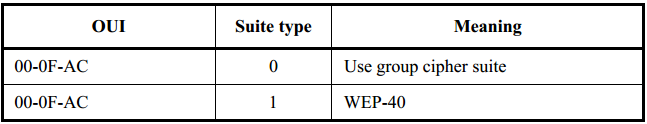


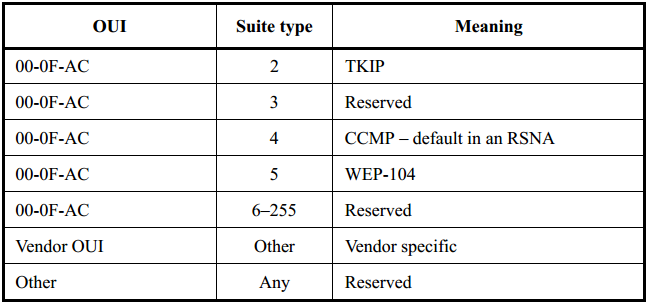
图46tb-Suite选择器格式

组织唯一标识符（OUI）字段的顺序应遵循7.1.1中MAC地址的排序约定。

表20da提供了由该修订定义的密码套件选择器。

表20da-Cipher套件选择器





密码套件选择器00-0F-AC：4（CCMP）应为默认密码套件值。

密码套件选择器00-0F-AC：1（WEP-40）和00-0F-AC：5（WEP-104）仅作为转换安全网络（TSN）中的组密码套件有效，以允许预先RSNA设备加入BSS。

不支持将CCMP用作具有TKIP作为成对密码套件的组密码套件。

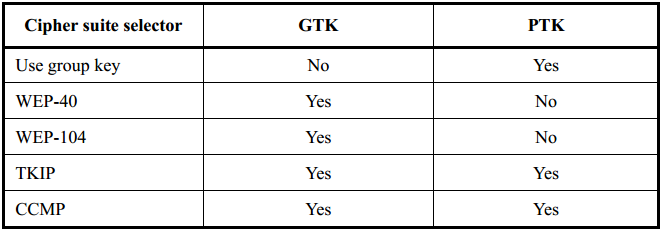
注-如果STA可以支持CCMP，则不需要较弱的数据机密性协议。

密码套件选择器00-0F-AC：0（使用组密码套件）仅作为成对密码套件有效。如果成对密码套件不支持任何成对密码套件，则AP可以为选择器00-0F-AC：0（使用组密码套件）指定。如果AP指定00-0F-AC：0（使用组密码套件）作为成对密码选择，则这将是AP通告的唯一成对密码选择。

如果启用了CCMP，则AP支持成对密钥，因此套件选择器00-0F-AC：0（使用组密码套件）不应是有效选项。

表20db表示应使用每个密码套件的情况。

表20db-Cipher套件用法



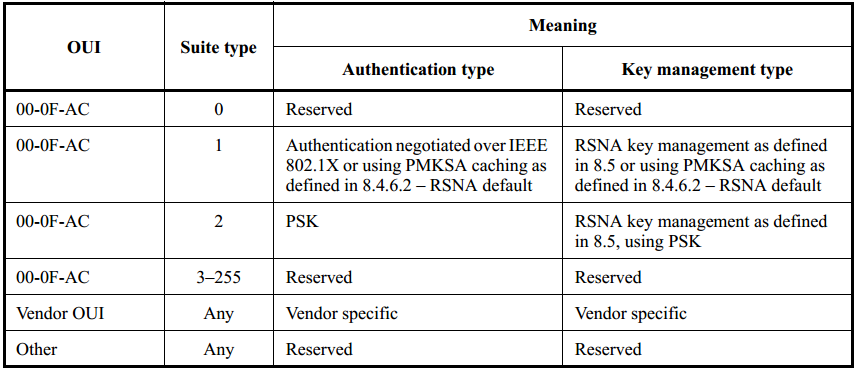
**7.3.2.25.2AKM套件**

AKMSuiteCount字段指示AKMSuiteList字段中包含的AKM套件选择器的数量。

AKMSuiteList字段包含RSN信息元素中包含的一系列AKM套件选择器。在IBSS中，只能指定一个AKM套件选择器，因为IBSS中的STA必须使用相同的AKM套件，并且因为没有机制在IBSS中协商AKMP（见8.4.4）。

每个AKM套件选择器指定一个AKMP。表20dc给出了该修订定义的AKM套件选择器。

表20dc-AKM套件选择器



AKM套件选择器值00-0F-AC：1（通过IEEE802.1X协商的验证）具有（8.5中定义的RSNA密钥管理或使用8.4.6.2中定义的PMKSA缓存），当AKM套件选择器时应为默认值字段未提供。

**注-**选择器值00-0F-AC：1仅指定IEEE802.1X用作身份验证传输。IEEE802.1X选择认证机制。

当PSK与RSNA密钥管理一起使用时，使用AKM套件选择器值00-0F-AC：2（PSK）。

注-选择器值00-0F-AC：1和00-0F-AC：2可以由Authenticator同时启用。

**7.3.2.25.3 RSN功能**

RSN功能字段指示请求或通告的功能。如果RSN能力字段在RSN信息单元中不可用，则每个RSN能力字段的值应取为0。

RSNCapabilities字段的长度是2个八位字节。RSNCapabilities字段的格式如图46tc所示，并在该图之后进行描述。

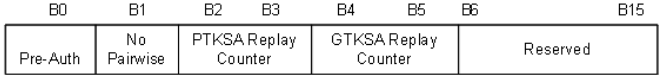


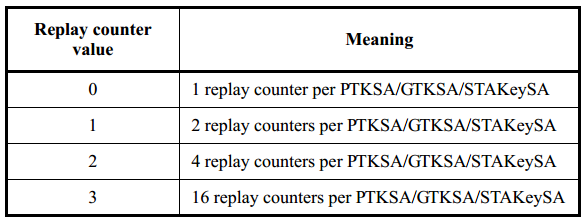
图46tc-RSN功能字段格式

-位0：预认证。AP将RSNCapabilities字段的Pre-Authentication子字段设置为1以表示它支持预身份验证（参见8.4.6.1），并在不支持预身份验证时将子字段设置为0。非APSTA将预认证子字段设置为0。

-位1：不成对。如果STA可以与成对密钥同时支持WEP默认密钥0（见8.5.1），则STA将RSNCapabilities字段的NoPairwise子字段设置为0.如果STA不能同时支持WEP默认密钥0key（见8.5.1），则STA将RSNCapabilities字段的NoPairwise子字段设置为1.NoPairwise子字段描述非APSTA的能力。IBSS和AP中的STA将NoPairwise子字段设置为0.NoPairwise子字段仅应在TSN中设置，并且当STA选择的成对密码套件为TKIP时。

-第2-3位：PTKSA重播计数器。STA将RSNCapabilities字段的PTKSAReplayCounter子字段设置为dot11RSNAConfigNumberofPTKSAReplayCounters中包含的值。dot11RSNAConfigNumberofPTKSAReplayCounters的最低有效位（LSB）放在第2位。见8.3.2.6和8.3.3.4.3。表20dd中定义了PTKSA/GTKSA/STAKeySA重播计数器子字段中值的含义。每个STAKeySA的重放计数器数量与每个PTKSA或GTKSA的重放计数器数量相同。

表20dd-PTKSA/GTKSA/STAKeySA重播计数器使用情况



-比特4-5：GTKSA重播计数器。STA将RSNCapabilities字段的GTKSAReplayCounter子字段设置为dot11RSNAConfigNumberofGTKSAReplayCounters中包含的值。dot11RSNAConfigNumberofGTKSAReplayCounters的LSB放在第4位。见8.3.2.6和8.3.3.4.3。表20dd中定义了GTKSA重放计数器子字段中值的含义。

-比特6-15：保留。RSN功能字段的其余子字段是保留的，在发送时应设置为0，并在接收时忽略。

**7.3.2.25.4PMKID**

PMKID计数和列表字段仅应用于AP的（重新）关联请求帧中的RSN信息元素。PMKIDCount指定PMKID列表字段中的PMKID数。PMKID列表包含STA认为对目的地AP有效的0个或更多个PMKID。PMKID可以参考。

a）通过与目标AP进行预身份验证获得的缓存PMKSA

b）来自EAP身份验证的缓存PMKSA

c）源自目标AP的PSK的PMKSA

有关PMKID的构造，请参见8.5.1.2。

注-如果STA不想使用该PMKSA，则STA可以选择不在PMKID列表字段中插入PMKID。

**用以下文字全部替换第8条：**

# 8.安全

**8.1框架**

该修订为IEEE802.11网络定义了两类安全算法：

-用于创建和使用RSNA的算法，称为RSNA算法

-预RSNA算法

注-该修订不禁止STA同时运行RSNA前和RSNA算法。

**8.1.1安全方法**

RSNA前安全性包括以下算法：

-WEP，见8.2.1

-IEEE802.11实体认证，如8.2.2中所述

RSNA安全性包括以下算法：

-TKIP，见8.3.2

-CCMP，见8.3.3

-RSNA建立和终止程序，包括使用IEEE802.1X认证，如8.4所述

-密钥管理程序，见8.5

**8.1.2RSNA设备和RSNA功能**

支持RSNA的设备可以创建RSNA。当dot11RSNAEnabled为真时，支持RSNA的设备应在信标，探测响应和（重新）关联请求帧以及4次握手的消息2和消息3中包括RSN信息单元。预RSNA设备无法创建RSNA。

**8.1.3RSNA建立**

STA的SME通过以下四种方式之一建立RSNA：

1. 在ESS中使用IEEE802.1XAKM时，支持RSNA的STA的SME如下建立RSNA：

1）它根据AP的信标或探测响应帧将AP识别为RSNA。

2）它应调用开放系统认证。

3）它在关联过程中协商密码套件，如8.4.2和8.4.3所述。

4）它使用IEEE802.1X进行身份验证，如8.4.6和8.4.7中所述。

5）它通过使用8.5定义的协议执行密钥管理算法来建立临时密钥。

6）它对商定的临时密钥和密码套件进行编程，以保护MAC并调用保护。有关RSNA数据保护机制的说明，请参见8.3.2和8.3.3。

b）如果RSNA基于ESS中的PSK，则STA的SME建立RSNA，如下所示：

1）它根据AP的信标或探测响应帧将AP识别为RSNA。

2）它应调用开放系统认证。

3）它在关联过程中协商密码套件，如8.4.2和8.4.3所述。

4）它通过使用8.5定义的协议执行密钥管理算法来建立临时密钥。它使用PSK作为PMK。

5）它通过将协商的密码套件和建立的临时密钥编程到MAC中然后调用保护来保护数据链路。

c）如果RSNA基于IBSS中的PSK，则STA的SME执行以下过程序列：

1）它从对等体的信标或探测响应帧中将对等体识别为具有RSNA能力。

注意-STA可以通过发送探测请求帧来响应来自未识别的STA的数据MPDU，以找出未识别的STA是否具有RSNA能力。

2）它可以选择调用开放系统身份验证。

3）每个STA使用8.5中的过程来建立临时密钥并协商密码套件。它使用PSK作为PMK。注意，两个对等STA可以同时遵循该过程。见8.4.9。

4）它通过编程协商的密码套件和建立的临时密钥然后调用保护来保护数据链路。

d）在IBSS中使用IEEE802.1XAKM的支持RSNA的STA的SME建立RSNA如下：

1）它从对等体的信标或探测响应帧中将对等体识别为具有RSNA能力。

注意-STA可以通过发送探测请求帧来响应来自未识别的STA的数据MPDU，以找出未识别的STA是否具有RSNA能力。

2）它可以选择调用开放系统身份验证。

3）每个站使用IEEE802.1X与与其他STA的认证者相关联的AS进行认证，如8.4.6和8.4.7中所述。因此，两个认证同时发生。

4）每个STA的SME通过使用8.5中定义的协议执行密钥管理算法来建立临时密钥。因此，两个这样的密钥管理算法在任何两个STA的请求者和认证者之间并行发生。

5）两个STA使用来自其中一个交换机的PTK和成对密码套件的商定的临时密钥部分来保护链路。每个STA使用由其发起的交换机建立的GTK来保护它发送的多播和广播帧。

安全关联建立的时间应小于MIB变量dot11RSNAConfigSATimeout。安全关联设置在由SME启动时启动，并在调用MLME-SETPROTECTION.request原语时完成。STA对超时采取的动作是策略决策。一些选项包括重试安全关联设置或尝试其他STA。当其中一个设置安全关联的STA无法正确响应设置安全关联时，此超时允许恢复。当安全关联超时导致两个安全关联之一失败时，它还允许在IBSS中进行恢复。

**8.1.4RSNA假设和约束（提供信息）**

RSNA假设如下：

a）每个STA可以生成加密质量的随机数。这种假设是基本的，因为加密方法需要随机源。有关建议的硬件和软件方法，请参阅H.6，以获得适合此目的的随机性。

b）使用IEEE802.1X身份验证时，使用的特定EAP方法执行相互身份验证。这种假设是IEEE802.11LAN中RSN设计所固有的，并且如果不将STA暴露给中间人攻击，则无法将其移除。EAP-MD5是不满足此约束的EAP方法的示例（请参阅IETFRFC3748）。此外，由于恶意内部人员可伪装成服务器并建立中间人攻击，因此使用无法区分服务器和客户端凭证的EAP身份验证方法会降低该方法的安全性，从而降低PSK的安全性。特别地，相互认证要求意味着未指定的先前注册过程（例如，长期认证密钥或通过诸如认证机构的第三方建立信任），因为STA必须能够将ESS或IBSS识别为一个值得信赖的实体，反之亦然。STA与所选AP使用的AS共享认证凭证，或者在PSK的情况下，与所选AP共享认证凭证。服务集标识符（SSID）提供未受保护的指示，指示所选AP的认证实体与STA共享凭证。只有成功完成IEEE802.1XEAP或PSK身份验证后，才能验证AP连接到授权网络或服务提供商的任何此类指示。

c）相互认证方法必须强大，这意味着当基于认证所暴露的信息时，模仿攻击在计算上是不可行的。这种假设是RSN设计所固有的。

d）AP和AS之间具有可信赖的信道，可用于交换加密密钥而不暴露于任何中间方。

e）IEEE802.1XnAS永远不会向除了STA当前正在与之通信的AP之外的任何一方公开公共对称密钥。这是一个非常强大的约束。这意味着AS本身永远不会受到损害。它还意味着IEEE802.1X AS嵌入在AP中，或者AP在物理上是安全的，AS和AP完全位于同一管理域中。该假设遵循以下事实：如果AP和AS不是共同定位的或者不直接共享成对密钥加密密钥，则不可能向移动STA保证其密钥由AS分配给AP，使用前没有受到损害。

f）类似地，STA从不与第三方共享与对等方共享的公共对称密钥。这样做会破坏用于检测MPDU重放和伪造的密钥的效用。

g）STA的Supplicant和Authenticator为该对之间的每个会话生成不同的新PTK。这种假设是基本的，因为任何PTK的重用都会使得受该密钥保护的所有数据受到损害。

h）发射机选择的目的地STA是正确的目的地。例如，地址解析协议（ARP）和因特网控制消息协议（ICMP）是确定不受ESS的其他成员攻击的安全的目的地STAMAC地址的方法。该问题的可能解决方案之一可能是STA仅发送或接收其最终目的地址（DA）或源地址（SA）是AP并且AP用于提供网络层路由功能的帧。但是，此类解决方案不属于本修正案的范围。

**8.2RSNA前安全方法**

除了开放系统身份验证之外，所有RSNA前安全机制都已被弃用，因为它们无法满足其安全目标。新实现应支持pre-RSNA方法，仅用于帮助迁移到RSNA方法。

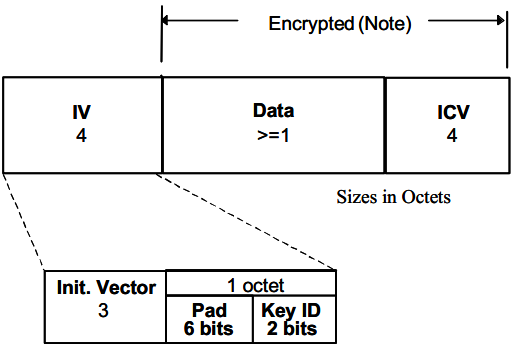
**8.2.1有线等效隐私（WEP）**

**8.2.1.1WEP概述**

WEP-40被定义为一种保护（使用40位密钥）无线LAN的授权用户之间交换的数据的机密性以防止随意窃听的手段。WEP的实施是可选的。在现场实现中，相同的算法已被广泛用于104位密钥而不是40位密钥;这被称为WEP-104。无论使用40位还是104位密钥，WEP封装和解封装机制都是相同的。因此，随后，WEP可以指WEP-40或WEP-104。

**8.2.1.2WEPMPDU格式**

图4310描绘了由WEP算法构造的加密帧体。



注-加密过程将原始MSDU扩展了8个八位字节，4个用于IV字段，4个用于ICV字段。ICV仅在数据字段中计算。

图43-扩展的WEPMPDU的构造

WEPICV字段的长度应为32位。扩展的框架体应以32位IV场开始。该字段应包含三个子字段：包含初始化矢量（IV）的3字节子字段，2位密钥ID子字段和6位Pad子字段。7.1.1中定义的排序约定适用于IV字段及其子字段和ICV字段。密钥ID子字段内容选择四个可能的秘密密钥值之一用于解密该帧体。使用键映射键时，将忽略KeyID字段值。

8.2.1节将进一步讨论这些位的解释。Pad子字段的内容应为0.KeyID子字段占用IV字段的最后一个八位字节的2个最高有效位（MSB），而Pad子字段占用该八位字节的6个LSB。

**8.2.1.3WEP状态**

WEP仅使用加密密钥;它不执行数据身份验证。因此，它没有数据完整性密钥。WEP使用两种类型的加密密钥：密钥映射密钥和默认密钥。

密钥映射密钥是对应于不同的发送器地址-接收器地址<TA，RA>对的未命名密钥。如果配置为<TA，RA>对，则实现应使用密钥映射密钥。换句话说，密钥映射密钥应用于WEP封装或解密由TA发送到RA的MPDU，而不管其他密钥类型的存在。当存在地址对的密钥映射密钥时，MPDU中的WEP密钥ID子字段在发送时应设置为0，在接收时应忽略。

默认键是四元素MIB数组中名为dot11WEPDefaultKeys的项，由名为dot11WEPDefaultKeyID的相关数组索引的值命名。如果没有为WEPMPDU的<TA，RA>对配置密钥映射密钥，则WEP将使用默认密钥来封装或解封装MPDU。在发送时，所选择的密钥是由索引dot11WEPDefaultKeyID给出的dot11DefaultKeys数组的元素-值0,1,2或3-分别对应于dot11WEPDefaultKeys的第一，第二，第三或第四元素。发送器在发送的MPDU的WEP密钥ID子字段中编码的值应为dot11WEPDefaultKeyID值。接收方应使用MPDU的密钥ID子字段索引到dot11WEPDefaultKeys以获得正确的默认密钥。所有WEP实现都应支持默认密钥。

注-许多实现还支持104位WEP密钥。它们的使用方式与40位WEP密钥完全相同：如104.1.4.3所述，在104位密钥之前加上24位WEPIV来构建128位WEP种子。然后，RC4流密码消耗所得到的128位WEP种子。

这种基于104位密钥的结构不能提供比40位结构更多的保证，并且它的实现和使用绝不被这一修正所宽恕。相反，104位结构仅用于记录事实上的实践。

所有WEP密钥的默认值应为null。WEP实现应丢弃MSDU并生成MA-UNITDATA-STATUS.indication，其传输状态指示帧可能未用空密钥封装以响应用空密钥封装MPDU的任何请求。

**8.2.1.4 WEP程序**

**8.2.1.4.1 WEPICV算法**

WEPICV应使用在明文MPDU数据（PDU）字段上计算的7.1.3.6中定义的CRC-32来计算。

**8.2.1.4.2WEP加密算法**

WEP实现应使用来自RSASecurity，Inc。的RC4流密码作为其加密和解密算法。RC4使用伪随机数生成器（PRNG）生成密钥流，该密钥流与明文数据流进行异或（XOR）以生成密文或从密文恢复明文。

**8.2.1.4.3WEP种子构建**

WEP实现应通过将加密密钥连接到IV来构造称为种子的每MPDU密钥。

对于WEP-40，WEP密钥的比特0-39对应于种子的比特24-63，并且IV的比特0-23分别对应于种子的比特0-23。7.1.1中的位编号约定适用于种子。种子应是RC4的输入，以加密或解密WEP数据和ICV字段。

注-对于WEP-104，WEP密钥的比特0-103对应于种子的比特24-127，并且IV的比特0-23分别对应于种子的比特0-23。

封装MPDU明文数据的WEP实现应为其WEP保护的每个MPDU选择一个新的IV。IV选择算法未指定。用于选择用于构造种子的加密密钥的算法也未指定。

解封装MPDU的WEP实现应使用来自所接收的MPDU的Init向量子字段的IV。有关解封装器如何选择用于构造每个MPDU密钥的密钥的规范，请参见8.2.1.4.5。

**8.2.1.4.4 WEPMPDU封装**

WEP应对明文MPDU应用三次转换以实现WEP封装。WEP通过明文数据计算ICV，并在MPDU数据之后附加。WEP使用RC4加密MPDU明文数据和ICV，种子按照8.2.1.4.3的规定构建。WEP将IV和密钥标识符编码到IV字段中，并加到加密的数据字段之前。

图43a描绘了WEP封装过程。在加密之前，应计算ICV并将其附加到明文数据，但IV编码步骤可以以便于实现的任何顺序发生。

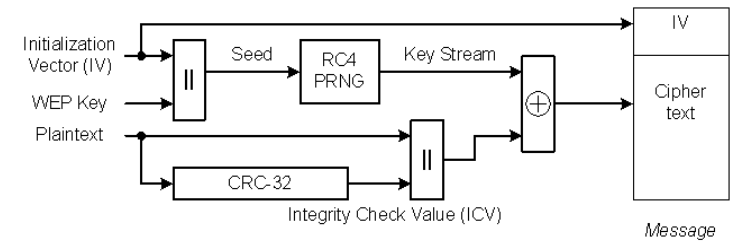


图43a-WEP封装框图

**8.2.1.4.5 mWEPMPDU解封装**

WEP应对WEPMPDU应用三次转换以解封其有效载荷。WEP从接收的MPDU中提取IV和密钥标识符。如果<TA，RA>对存在密钥映射密钥，则应将其用作WEP密钥。否则，从接收到的MPDU中的WEPIV字段的密钥ID子字段中提取密钥标识符，标识要使用的默认密钥。

WEP使用构造的种子来解密WEPMPDU的数据字段;这会产生明文数据和ICV。最后，WEP重新计算ICV并逐位将其与来自MPDU的解密ICV进行比较。如果两者在位上相同，则WEP从MPDU中移除IV和ICV，这被认为是有效的。如果它们在任何位位置不同，则WEP会向MAC管理生成错误指示。具有错误MPDU的MSDU（由于无法解密）不应传递给LLC。

图43b描绘了WEP解封装的框图。与封装不同，解封装步骤应按指示的顺序进行。

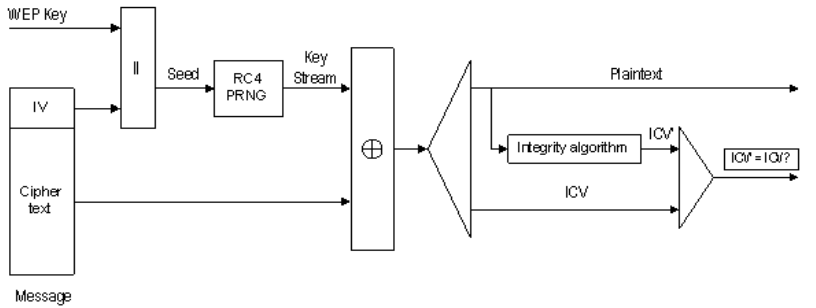


图43b-WEP解封装框图

**8.2.2预RSNA认证**

**8.2.2.1概述**

在ESS中，非APSTA和AP必须在关联之前完成IEEE802.11认证交换。这种交换在独立的BSS网络中是可选的。

子类型认证的所有管理帧应该是单播的，因为在STA对之间执行IEEE802.11认证，即，不允许广播/多播认证。子类型Deauthentication的管理帧是建议性的，可以作为组寻址帧发送。

不推荐使用共享密钥身份验证，除了与RSNA前设备的向后兼容性之外，不应实现共享密钥身份验证。

**8.2.2.2开放系统认证**

OpenSystem身份验证是一种空身份验证算法。如果接收方STA处的dot11AuthenticationType设置为开放系统认证，则可以认证任何请求开放系统认证的STA。STA可以拒绝与另一个请求STA进行认证。开放系统认证是RSNA前设备的默认认证算法。

开放系统身份验证使用双消息身份验证事务序列。第一条消息断言身份并请求身份验证。第二条消息返回验证结果。如果结果是“成功”，则应声明STA相互认证。

在8.2.2.2.1和8.2.2.2.2的描述中，发起认证交换的STA被称为请求者，并且交换中的初始帧被寻址的STA被称为响应者。

**8.2.2.2.1开放系统认证（第一帧）**

-消息类型：管理

-消息子类型：身份验证

-信息项目：

* 认证算法识别=“开放系统”
* STA标识声明（在标题的SA字段中）
* 身份验证事务序列号=1
* 身份验证算法相关信息（无）

-消息方向：从请求者到响应者

**8.2.2.2.2开放系统认证（最终帧）**

-消息类型：管理

-消息子类型：身份验证

-信息项目：

* 认证算法识别=“开放系统”
* 身份验证事务序列号=2
* 身份验证算法相关信息（无）
* 7.3.1.9中定义的请求认证的结果

-消息方向：从响应者到请求者

如果dot11AuthenticationType不包含值“OpenSystem”，则结果代码不应取值“success”。

**8.2.2.3共享密钥验证**

共享密钥身份验证旨在将STA认证为知道共享密钥的成员或不知道共享密钥的成员。

当且仅当选择了WEP时，才能使用共享密钥身份验证。

该机制使用通过独立于IEEE802.11的安全信道传递给参与STA的共享密钥。此共享密钥在只写MIB属性中设置，旨在将密钥值保持在STA内部。

除非dot11PrivacyOptionImplemented属性为true，否则STA不应启动共享密钥身份验证交换。

在8.2.2.3.1至8.2.2.3.5的描述中，发起认证交换的STA被称为请求者，并且交换中的初始帧被寻址的STA被称为响应者。

**8.2.2.3.1共享密钥认证（第一帧）**

-消息类型：管理

-消息子类型：身份验证

-信息项目：

* STA标识声明（在标题的SA字段中）
* 验证算法识别=“共享密钥”
* 身份验证事务序列号=1
* 身份验证算法相关信息（无）

-消息方向：从请求者到响应者

**8.2.2.3.2共享密钥认证（第二帧）**

在共享密钥认证序列中发送第二帧之前，响应者应使用WEP生成一串八位字节以用作认证质询文本。

-消息类型：管理

-消息子类型：身份验证

-信息项目：

* 验证算法识别=“共享密钥”
* 身份验证事务序列号=2
* 身份验证算法相关信息=身份验证结果
* 7.3.1.9中定义的请求认证的结果代码。

如果状态代码不是“成功”，则这应该是事务序列的最后一帧;并且未指定挑战文本字段的内容。

如果状态代码为“成功”，则以下附加信息项应具有有效内容：

认证算法相关信息=挑战文本

该认证结果应具有128个八位字节的固定长度。该字段应填充由WEPPRNG生成的八位字节。质询字段的实际值并不重要，但该值不应是静态值。

-消息方向：从响应者到请求者

**8.2.2.3.3共享密钥认证（第三帧）**

请求者应将挑战文本从第二帧复制到第三帧。第三帧应在使用共享密钥通过WEP封装后发送，如8.2.1中所定义。

-消息类型：管理

-消息子类型：身份验证

-信息项目：

* 验证算法识别=“共享密钥”
* 身份验证事务序列号=3
* 身份验证算法相关信息=来自第二帧的质询文本

-消息方向：从请求者到响应者

**8.2.2.3.4共享密钥认证（最终帧）**

响应者应按8.2.1中的描述对第三帧进行WEP解封装。如果WEPICV检查成功，响应者应将挑战文本字段的解密内容与第二帧中发送的挑战文本进行比较。如果它们相同，则响应者应在序列的最后一帧中以成功的状态代码进行响应。如果WEPICV检查失败或挑战文本比较失败，则响应者应在最终帧中以不成功的状态代码进行响应。

-消息类型：管理

-消息子类型：身份验证

-信息项目：

* 验证算法识别=“共享密钥”
* 身份验证事务序列号=4
* 身份验证算法相关信息=身份验证结果
* 7.3.1.9中定义的请求认证的结果代码。

这是一个固定长度的项目，其值为“成功”和“不成功”。

-消息方向：从响应者到请求者

**8.2.2.3.5共享密钥MIB属性**

为了发送子类型认证的管理帧，认证事务序列号字段值为2，MAC应根据以下决策树进行操作：

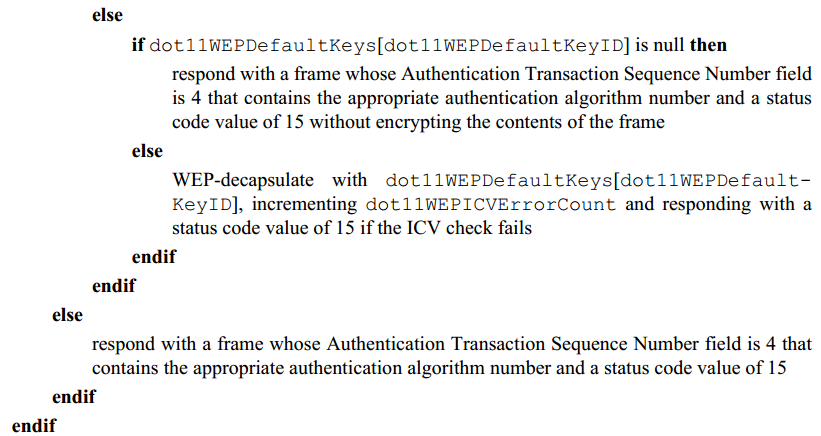
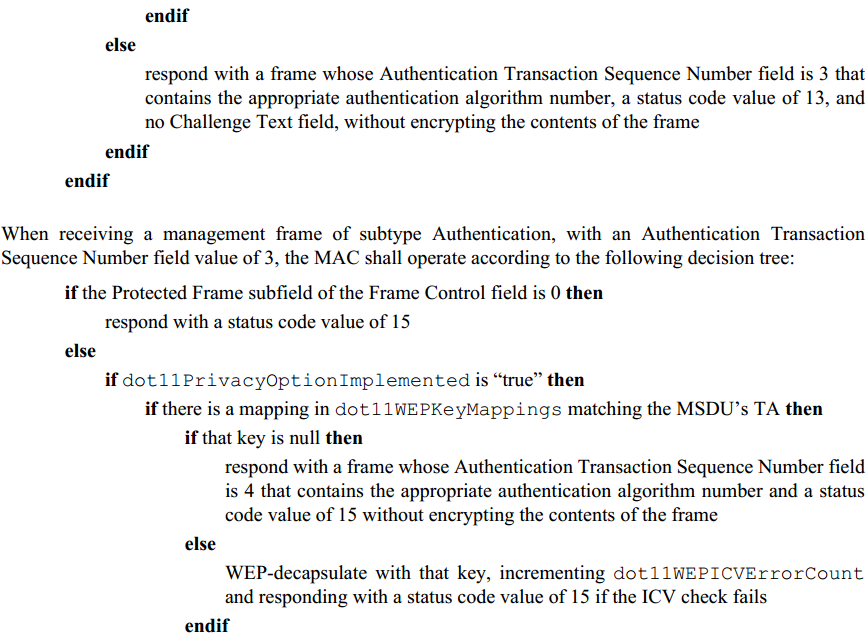
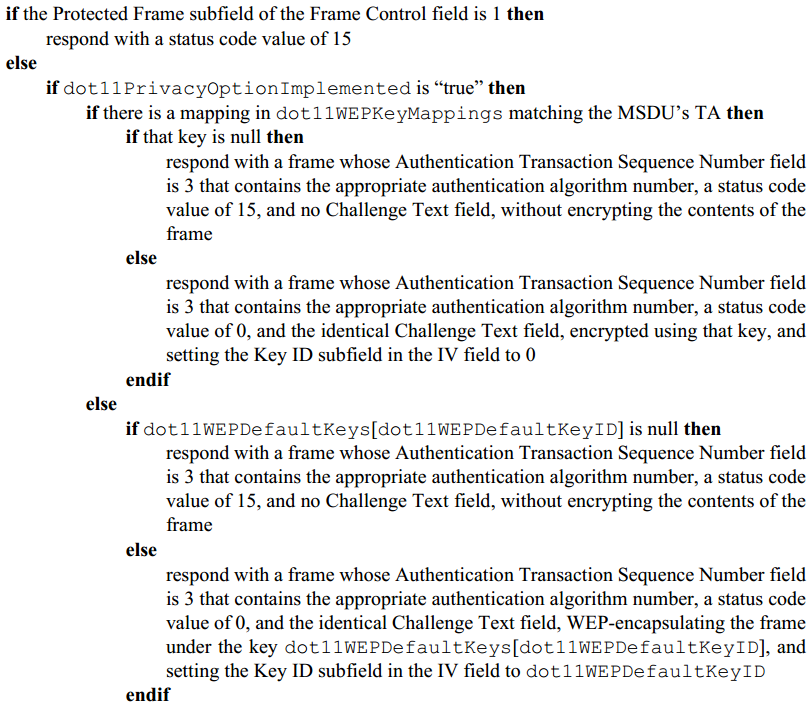
**如果**dot11PrivacyOptionImplemented为“false”，那么MMPDU在ChallengeText字段中以0个八位字节的序列传输，状态码值为13

**否则**

使用WEPPRNG生成的128个八位字节的序列发送MMPDU，其密钥的值未指定且超出了本修订的范围和随机选择的IV值（请注意，这通常由选择IV的相同机制选择）挑战文本字段中传输数据MPDU的值，状态代码值为0（使用的IV是无关紧要的，不会传输）。请注意，此过程的密钥/IV选择中涉及加密问题，因为质询文本未加密发送，因此提供了来自PRNG的已知输出序列。

Endif

为了接收子类型认证的管理帧，认证事务序列号字段值为2，MAC应根据以下决策树操作：



如果属性dot11PrivacyOptionImplemented为false，则属性dot11PrivacyInvoked不应采用true值。将dot11WEPKeyMappings设置为包含多于dot11WEPKeyMappingLength条目的值是非法的，并且应对机密性服务的操作具有特定于实现的影响。请注意，dot11WEPKeyMappings可能包含从零到dot11WEPKeyMappingLength条目（包括0和dot11WEPKeyMappingLength条目）。

在验证序列期间，不应更改aPrivacygrp中的属性值，因为可能会导致意外操作。

**8.3RSNA数据机密性协议**

**8.3.1概述**

该修订定义了两种RSNA数据机密性和完整性协议：TKIP和CCMP。在声称符合RSNA标准的所有IEEE802.11设备中，CCMP的实施必须是强制性的。对于RSNA，TKIP的实施是可选的。TKIP的设计目标是该算法应该在大多数仅支持WEP的设备的能力范围内实现，因此供应商可以对许多此类设备进行现场升级以支持TKIP。

注-使用任何机密性算法取决于本地策略。TKIP的机密性和完整性机制不如CCMP的机密性和完整性机制。TKIP旨在在广泛的RSNA前设备的硬件限制范围内运行。TKIP适用于固件设备，硬件兼容的现场设备升级。RSNA设备仅在与无法或未配置为使用CCMP进行通信的设备通信时才应使用TKIP。

**8.3.2临时密钥完整性协议（TKIP）**

**8.3.2.1TKIP概述**

TKIP是一个密码套件，增强了RSNA前硬件上的WEP协议。TKIP修改WEP如下：

a）发射机计算MSDUSA和DA上的密钥加密消息完整性码（MIC），MSDU优先级（见8.3.2.3）和MSDU明文数据。TKIP在分割成MPDU之前将计算的MIC附加到MSDU数据。接收器在解密，ICV检查以及将MPDU的碎片整理成MSDU之后验证MIC，并丢弃任何接收到的具有无效MIC的MSDU。TKIP的MIC可以防御伪造攻击。

b）由于TKIPMIC的设计限制，对手仍有可能损害消息的完整性;因此，TKIP也实施了对策。这些对策限制了成功伪造的可能性以及攻击者可以了解密钥的信息量。

c）TKIP使用每MPDUTKIP序列计数器（TSC）对其发送的MPDU进行排序。接收机丢弃无序接收的MPDU，即没有接收到序列号增加的MPDU。这提供了重播保护。TKIP将发送方到接收方的TSC值编码为WEPIV和扩展IV。

d）TKIP使用加密混合函数将临时密钥，TA和TSC组合到WEP种子中。接收器从接收的MPDU恢复TSC，并利用混合功能计算正确解密MPDU所需的相同WEP种子。密钥混合功能旨在抵御针对WEP密钥的弱密钥攻击。

TKIP定义了额外的MIB变量;见附件D.

**8.3.2.1.1TKIP封装**

TKIP通过几个附加功能增强了WEP封装，如图43c所示。

a）TKIPMIC计算保护MSDU数据字段和相应的SA，DA和优先级字段。MIC的计算是在SA，DA，优先级和MSDU数据字段的有序级联上执行的。MIC附加到MSDU数据字段。TKIP在附加MIC之前丢弃任何MIC填充。

b）如果需要，IEEE802.11将具有MIC的MSDU分段为一个或多个MPDU。TKIP为每个MPDU分配一个单调增加的TSC值，注意从同一个MSDU生成的所有MPDU具有相同的扩展IV值（见8.3.2.2）。

c）对于每个MPDU，TKIP使用密钥混合函数来计算WEP种子。

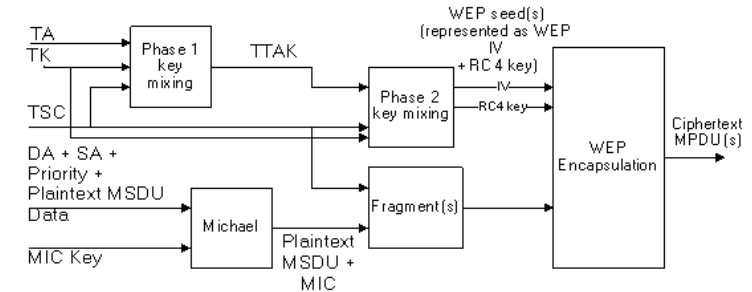


图43c-TKIP封装框图

d）TKIP将WEP种子表示为WEPIV和RC4密钥，并将每个MPDU传递给WEP以生成ICV（见7.1.3.6），并加密明文MPDU，包括全部或部分MIC，如果有的话。WEP使用WEP种子作为WEP默认密钥，由与临时密钥相关联的密钥标识符标识。

注-当TSC空间耗尽时，实现可用的选择是用新的临时密钥替换临时密钥或结束通信。重新使用任何TSC值会危及已发送的流量。注意，重传的MPDU在不损害安全性的情况下重用TSC。然而，TSC足够大，TSC空间耗尽不应成为问题。

在图43c中，TKIP混合发送地址和密钥（TTAK）表示由TKIP混合功能的阶段1产生的中间密钥（见8.3.2.5）。

**8.3.2.1.2TKIP解封装**

TKIP通过以下附加步骤增强了WEP解封装过程：

a）在WEP解封装接收到的MPDU之前，TKIP从WEPIV和扩展IV中提取TSC序列号和密钥标识符。TKIP丢弃违反排序规则的接收MPDU（见8.3.2.6），否则使用混合函数构建WEP种子。

b）TKIP将WEP种子表示为WEPIV和RC4密钥，并将这些密钥与MPDU一起传递给WEP进行解封装。

c）如果WEP指示ICV检查成功，则实现将MPDU重新组装成MSDU。如果MSDU碎片整理成功，则接收方验证TKIPMIC。如果MSDU碎片整理失败，则丢弃MSDU。

d）MIC验证步骤在MSDUSA，DA，优先级和MSDU数据字段（但不是TKIPMIC字段）上重新计算MIC。然后将计算的TKIPMIC结果与接收的MIC逐位比较。

e）如果接收到的和本地计算的MIC值相同，则验证成功，TKIP应将MSDU传送到上层。如果两者不同，则验证失败;接收方应丢弃MSDU并采取适当的对策。

图43d描绘了该过程。

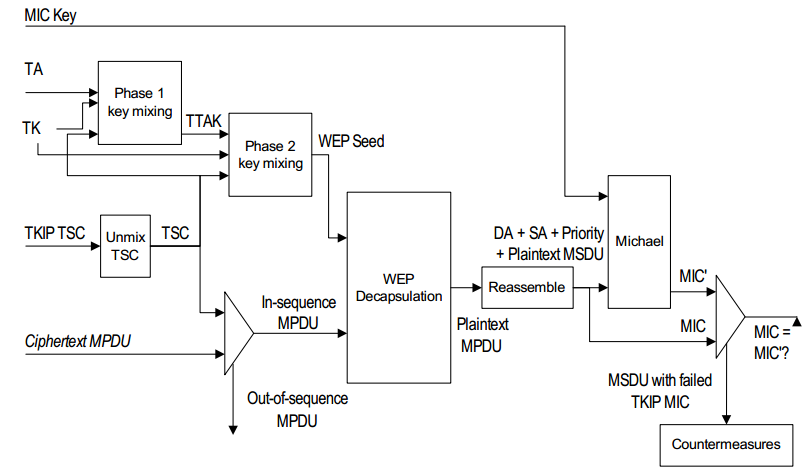


图43d-TKIP解封装框图

**8.3.2.2TKIPMPDU格式**

TKIP重用RSNA前WEPMPDU格式。它将MPDU扩展4个八位字节以适应WEPIV的扩展，由扩展IV字段表示，并将MSDU格式扩展8个八位字节以适应新的MIC字段。TKIP在WEPIV字段之后和加密数据之前立即插入扩展IV字段。TKIP将MIC附加到MSDU数据字段;MIC成为加密数据的一部分。

一旦将MIC附加到MSDU数据，则添加的MIC八位字节被认为是用于后续分段的MSDU的一部分。

图43e描绘了使用TKIP时加密的MPDU的布局。注意，该图仅描绘了MSDU可以封装在单个MPDU中的情况。

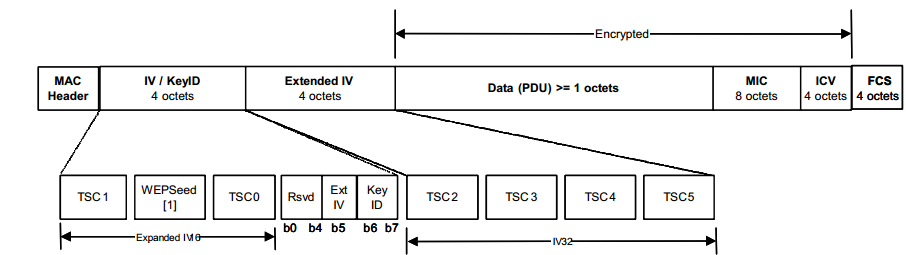


图43e-扩展的TKIPMPDU的构建

密钥ID八位字节中的ExtIV位指示是否存在扩展IV。如果ExtIV位为0，则仅传输非扩展IV。如果ExtIV位为1，则原始IV之后的4个八位字节的扩展IV。对于TKIP，应设置ExtIV位，并提供扩展IV字段。对于WEP帧，ExtIV位应为0。KeyID字段应设置为MLME-SETKEYS.request原语提供的密钥索引，用于封装帧的密钥。

TSC5是TSC中最重要的八位位组，TSC0是最不重要的。八位字节TSC0和TSC1形成IV序列号，并与TKIPPhase2密钥混合一起使用。八位字节TSC2-TSC5用于TKIP阶段1密钥散列，并且位于扩展IV字段中。当较低的16位序列号翻转（0xFFFF→0x0000）时，扩展的IV值，即整个48位TSC的高32位，应递增1。

注-这种结构的基本原理如下：

-对齐字边界可简化旧设备的实施。

-添加4个八位字节的扩展IV消除了TSC耗尽作为重新生成的原因。

-密钥ID八位字节更改。位5表示存在扩展IV。接收器/发送器将密钥ID之后的4个八位字节解释为扩展IV。接收/发送STA还使用八位字节TSC0和TSC1的值来检测必须更新高速缓存的TTAK。

扩展IV字段不应加密。

WEPSeed[1]不用于构造TSC，但设置为（TSC1|0x20）和0x7f。

TKIP应加密在同一时间密钥下从一个MSDU生成的所有MPDU。

**8.3.2.3TKIPMIC**

IEEE802.11WEP设计中的缺陷导致其无法实现保护数据流量内容免受偶然窃听者攻击的目标。最重要的WEP缺陷之一是缺乏消除伪造消息和其他主动攻击的机制。为了防御主动攻击，TKIP包括一个名为Michael的MIC。该MIC仅提供针对消息伪造的弱防御，但它构成了大多数传统硬件可以实现的最佳状态。TKIP使用不同的MIC键，具体取决于传输方向，如8.6.1和8.6.2所述。

附件H包含TKIPMIC的实施。它还为MIC提供测试向量。

**8.3.2.3.1TKIPMIC的动机**

在定义MIC的详细信息之前，查看此机制运行的上下文非常有用。原始WEP设计支持的主动攻击包括：

-比特翻转攻击

-数据（有效载荷）截断，连接和拼接

-碎片攻击

-针对密钥的迭代猜测攻击

-通过修改MPDUDA或RA字段进行重定向

-通过修改MPDUSA或TA字段进行模拟攻击

MIC使得任何这些攻击都难以成功。

所有这些攻击都与TKIPMIC保持在MPDU级别。然而，MIC适用于MSDU，因此它阻止成功的MPDU级攻击。TKIP将MIC应用于发射机的MSDU，并在接收机的MSDU级别进行验证。如果MIC检查在MSDU级别失败，则实现应丢弃MSDU并调用对策（见8.3.2.4）。

图43f描绘了不同的对等层通信。

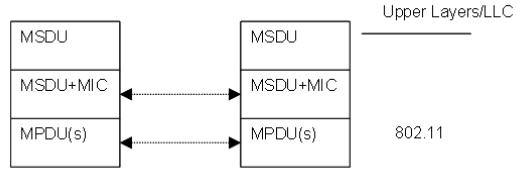


图43f-TKIPMIC与IEEE802.11处理的关系（资料性）

该图描绘了一种体系结构，其中MIC在逻辑上附加到原始MSDU以响应MA-UNITDATA.request原语。TKIPMIC计算结束

-MSDUDA

-MSDUSA

-MSDU优先级（保留供将来使用）

-整个未加密的MSDU数据（有效负载）

DA字段，SA字段，三个保留八位字节和一个八字节优先级字段仅用于计算MIC。优先级字段应为0并保留供将来使用。使用7.1.1中描述的约定将图43g中的字段视为八位字节流。

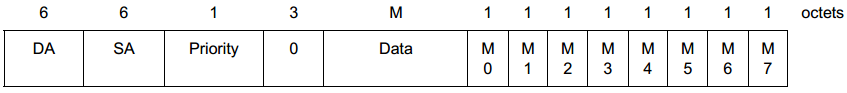


图43g-TKIPMIC处理格式

TKIP在MSDU有效载荷的末尾附加MIC。迈克尔的MIC大小为8个八位字节。然后，IEEE802.11MAC应用其正常处理来将该MSDU-with-MIC作为一个或多个MPDU的序列发送。换句话说，可以将具有MIC的MSDU划分为一个或多个MPDU，在每个MPDU上计算WEPICV，并且甚至可以在划分之后将MIC划分为位于两个MPDU中。TKIPMIC增强但不替代WEPICV。由于TKIPMIC是一种弱结构，TKIP通过加密保护MIC，这使得TKIPMIC伪造更加困难。WEPICV有助于防止错误检测可能导致对策被调用的MIC故障。

接收器反转此过程以重新组装MSDU;并且，在逻辑上重新组装MSDU之后，IEEE802.11MAC在将MSDU递送到上层之前验证MIC。如果MIC验证成功，则MAC递送MSDU。如果MIC验证失败，MAC应丢弃MSDU并调用对策（见8.3.2.4）。

注意-TKIP通过MSDU而不是MPDU计算MIC，因为这样做可以增加预先存在的WEP硬件的实现灵活性。

应该注意的是，单独的MIC不能提供完全的伪造保护，因为它无法防御重放攻击。因此，TKIP通过TSC测序和ICV验证提供重放检测。此外，如果TKIP与GTK一起使用，则内部STA可以伪装成属于该组的任何其他STA。

**8.3.2.3.2TKIPMIC的定义**

迈克尔生成64位MIC。迈克尔密钥由64位组成，表示为8字节序列，k0...k7。它被转换为两个32位字，K0和K1。在本小节中，八位字节和32位字之间的所有转换都应使用7.1.1中给出的小端符号约定。

Michael对每个MSDU进行操作，包括优先级字段，3个保留八位字节，SA字段和DA字段。MSDU由八位字节m0...mn-1组成，其中n是MSDU八位字节的数量，包括SA，DA，优先级和数据字段。消息在末尾填充一个具有值0x5a的单个八位字节，然后是4到7个零八位字节。选择零八位字节的数量使得填充的MSDU的总长度是四的倍数。填充不与MSDU一起传输;它用于简化最终块的计算。然后将MSDU转换为32位字M0...MN-1的序列，其中N=（n+5）/4，并且其中a表示将a舍入到最接近的整数。通过构造，MN-1=0并且MN-2≠0。（48页）

从键值（K0和K1）开始迭代计算MIC值，并为每个消息字应用块函数b，如图43h所示。算法循环总共运行N次（i取0到N-1的值），其中N如上所述，组成填充的MSDU的32位字的数量。该算法产生两个单词（l和r），使用minimalsignificant-octet-first约定将其转换为8个八位字节的序列：

—M0=l&0xff  
—M1=(l/0x100)&0xff  
—M2=(l/0x10000)&0xff  
—M3=(l/0x1000000)&0xff  
—M4=r&0xff  
—M5=(r/0x100)&0xff  
—M6=(r/0x10000)&0xff  
—M7=(r/0x1000000)&0xff

这是MIC值。MIC值作为要发送的数据附加到MSDU。

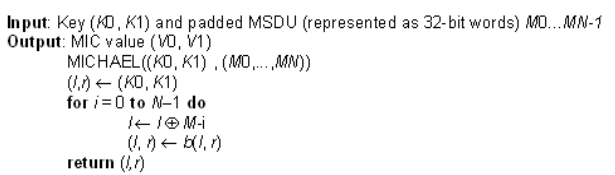


图43h-Michael消息处理

图43i定义了迈克尔块函数b。它是Feistel型结构，具有交替添加和XOR操作。它使用<<<表示旋转左运算符的32位值，>>>表示旋转右运算符，XSWAP表示交换2个最低有效八位字节位置的函数。它还使用一个单词中两个最重要的八位字节的位置。

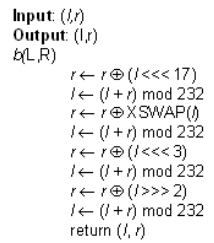


图43i-Michael块功能

**8.3.2.4TKIP对策程序**

TKIPMIC以安全性为代价，支持RSNA前设备的可实现性。迈克尔仅提供针对主动攻击的弱保护。接收到的MSDU中的MIC故障表示可能的主动攻击。对MIC的成功攻击意味着攻击者可以注入伪造的数据帧并对加密密钥本身进行进一步的有效攻击。如果TKIP实施检测到可能的主动攻击，TKIP应采取本子条款规定的对策。这些对策实现了以下目标：

-应将MIC故障事件记录为与安全相关的事项。MIC故障几乎是主动攻击的明确指示，并需要系统管理员进行跟进。

-MIC故障率必须保持在每分钟两次以下。这意味着在60秒内检测到两个MIC故障事件的STA和AP必须禁用使用TKIP的所有接收，持续60秒。减速使得攻击者很难在短时间内进行大量伪造尝试。

-作为附加安全功能，PTK以及在Authenticator的情况下，应更改GTK。

在验证MIC之前，接收方应检查所有相关MPDU的FCS，ICV和TSC。在检查MIC之前，应丢弃具有无效FCS，不正确的ICV或小于或等于TSC重放计数器的TSC值的任何MPDU。这避免了不必要的MIC故障事件。在MIC之前检查TSC使得基于对策的拒绝服务攻击更难以执行。虽然FCS和ICV机制足以检测噪声，但它们不足以检测主动攻击。FCS和ICV提供错误检测，但不提供完整性保护。

应使用单个计数器或计时器来记录MIC故障事件。这些失败事件定义如下：

-对于身份验证者：

-检测收到的单播帧上的MIC故障。

-收到MichaelMIC失败报告框。

-对于请求者：

-检测接收到的单播或广播/多播帧上的MIC故障。

-尝试发送迈克尔MIC失败报告框架。

MIC故障的数量是独立于特定密钥上下文而产生的。任何单个MIC故障，无论是由请求方还是认证方检测到，以及是否由组MIC密钥故障或成对MIC密钥故障引起，都应视为MIC故障事件的原因。

Supplicant使用单个MichaelMICFailureReport框架向Authenticator报告MIC故障事件。迈克尔MIC故障报告是EAPOL-Key帧，其以下密钥信息字段位设置为1：MIC位，错误位，请求位，安全位。请求者使用当前PTK保护此消息;Supplicant使用PTK的KCK部分来计算IEEE802.1XEAPOLMIC。

IEEE802.11MAC使用MLME-MICHEALMICFAILURE.indication原语来尝试向本地IEEE802.1X请求者或认证者指示MIC故障。请求者使用MLME-EAPOL.request原语发送包含MichaelMIC失败报告的EAPOL-Key帧。当接收到该EAPOL-Key帧的IEEE802.11MAC确认（ACK）时，MLMEEAPOL.confirm原语向请求者指示。

应记录第一次MIC故障，并启动计时器以实施对策。如果请求者检测到MIC失败事件，它还应通过发送迈克尔MIC失败报告帧向AP报告该事件。

如果在最近的先前故障的60秒内发生后续MIC故障，则其IEEE802.1X实体充当请求者的STA将取消认证（如11.3.3中所定义）本身或取消认证具有安全关联的所有STA。其IEEE802.1X实体充当认证者。对于IBSSSTA，应采取Supplicant和Authenticator动作。此外，设备不应接收或发送任何TKIP加密数据帧，并且在检测到至少60秒后，不得在任何对等体之间接收或发送除IEEE802.1X消息之外的任何未加密数据帧。第二次失败。如果设备是AP，则应在此60秒期间禁止使用TKIP进行新的关联;在60秒期限结束时，AP应恢复正常操作并允许STA（重新）关联。如果设备是IBSSSTA，则应在此60秒期间禁止使用TKIP的任何新安全关联。如果设备是请求者，它应首先发送迈克尔MIC失败报告帧，然后撤销其PTKSA并取消自我验证。

aMICFailTime属性应包含记录MIC故障时的sysUpTime值。

**8.3.2.4.1认证者的TKIP对策**

Authenticator使用的对策如图43j所示，描述如下：

a）对于接收具有MIC错误的帧的Authenticator的STA，

1）丢弃框架。

2）增加MIC故障计数器，dot11RSNAStatsTKIPLocalMIC-Failures。

3）生成MLME-MICHAELMICFAILURE.indication原语。

b）对于接收MLME-MICHAELMICFAILURE.indication原语或MichaelMICFailureReport帧的Authenticator，

1）如果是迈克尔MIC失败报告框架，则增加dot11RSNAStatsTKIPRemoteMICFailures。

2）如果这是过去60秒内的第一次MIC故障，请初始化对策计时器。

3）如果自最近一次MIC失败以来已经过去不到60秒，则认证者应使用TKIP对所有STA进行解除认证并删除所有PTKSA。如果当前GTKSA使用TKIP，则应丢弃该GTKSA，并构造新的GTKSA，但不会使用60秒。认证者应拒绝使用TKIP作为一个或多个密码构建新的PTKSA，持续60秒。在此期间结束时，应重置MIC故障计数器和定时器，并像往常一样接受PTKSA的创建。

4）如果Authenticator使用IEEE802.1X认证，则Authenticator应将IEEE802.1XAuthenticator状态机的状态转换为INITIALIZE状态。这将重新启动IEEE802.1X状态机。如果Authenticator使用的是PSK，则省略此步骤。

请注意，如果请求者的STA是ESSSTA，则可以使用MIC失败的原因代码进行解除身份验证。Authenticator不应将deauthenticate记录为MIC故障事件，以防止通过解除认证的拒绝服务攻击。请求者的STA必须通过迈克尔MIC失败报告帧报告MIC失败事件，以便AP记录事件。

使用TKIP对所有STA进行去认证的要求将包括那些使用CCMP作为成对密码的用户，如果他们也使用TKIP作为组密码。

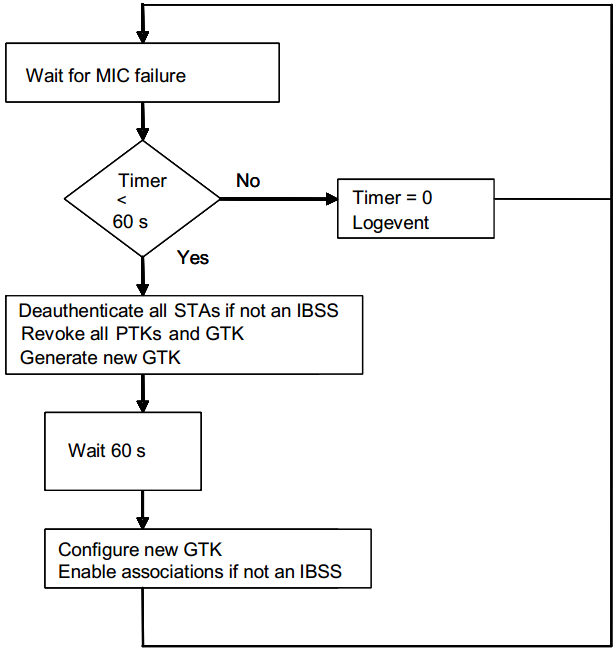


图43j-认证器MIC对策

**8.3.2.4.2申请人的TKIP对策**

请求者使用的对策如图43k所示，描述如下：

a）对于接收具有MIC错误的帧的请求者STA，

1）增加MIC故障计数器，dot11RSNAStatsTKIPLocalMIC-Failures。

2）丢弃违规的框架。

3）生成MLME-MICHAELMICFAILURE.indication原语。

b）对于从其STA接收MLME-MICHAELMICFAILURE.indication原语的请求者，

1）向AP发送迈克尔MIC失败报告帧。

2）如果这是过去60秒内的第一次MIC故障，请初始化对策计时器。

3）如果自最近一次MIC失败以来已经过去不到60秒，则删除PTKSA和GTKSA。从AP解除认证并等待60秒（重新）与同一AP建立TKIP关联。TKIP关联是任何使用TKIP作为其成对或组密码套件的IEEE802.11关联。

c）如果非APSTA接收到具有原因代码“MICfailure”的去认证帧，则不能确定该帧是否未被伪造，因为它不包含MIC。STA可以尝试与该AP或另一AP进行关联。如果帧是真实的，那么尝试与请求使用TKIP的同一AP相关联的尝试将失败，因为AP将进行对策。

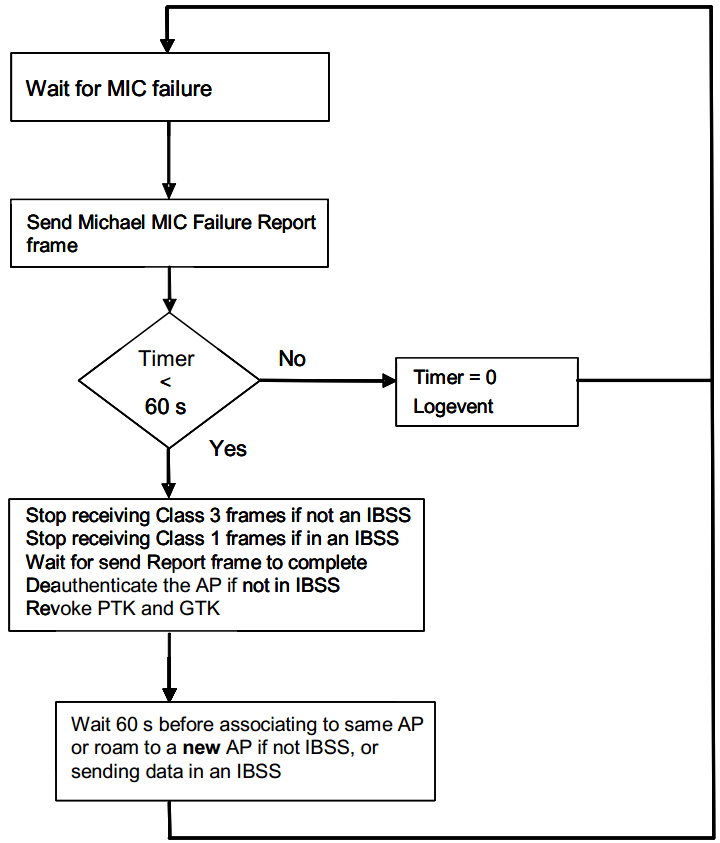


图43k-请求者MIC对策

**8.3.2.5TKIP混合功能**

附件H定义了TKIP混合功能的C语言参考实现。它还提供混合功能的测试向量。

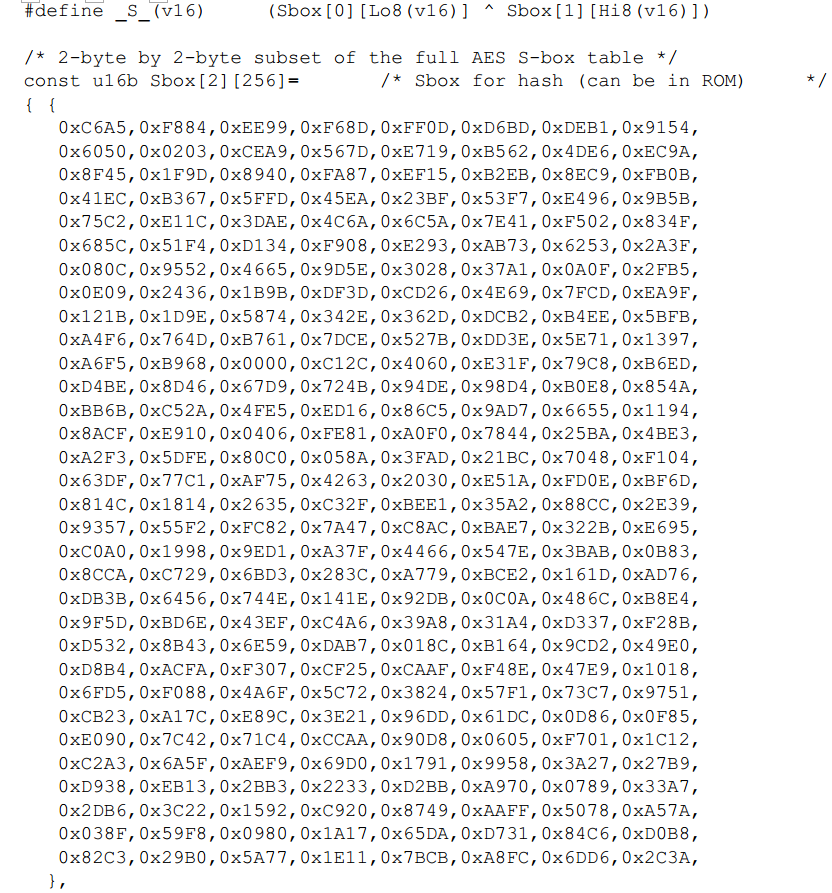
混合功能有两个阶段。阶段1将适当的临时密钥（成对或组）与TA和TSC混合。STA可以缓存该阶段的输出以重新使用与相同临时密钥和TA相关联的后续MPDU。阶段2将阶段1的输出与TSC和临时密钥（TK）混合以产生WEP种子，也称为每帧密钥。WEP种子可以在使用之前进行预先计算。两阶段过程可概括如下：

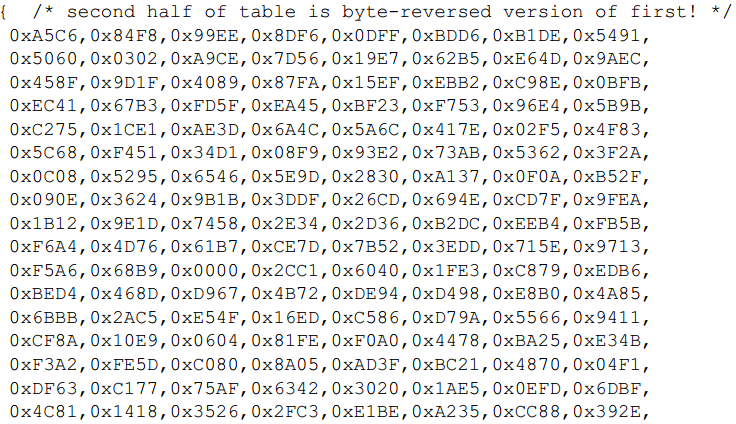


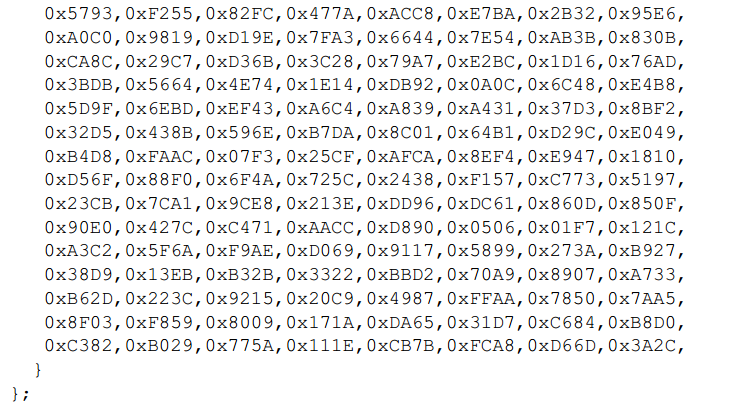
**8.3.2.5.1S-Box**

阶段1和阶段2都依赖于本子条款中定义的S盒。S盒用一个16位值替换另一个16位值。该功能可以实现为查表。

注-S盒是非线性替代。表查找可以组织为具有65536个条目的单个表和16位索引（表的128K八位字节）或具有256个条目和8位索引的两个表（两个表的1024个八位字节）。当使用两个较小的表时，高阶八位字节用于从一个表获得16位值，低阶八位字节用于从另一个表获得16位值，以及S-box输出是两个16位值的XOR（⊕）。第二个S-box表是第一个八位字节交换的副本。







**8.3.2.5.2阶段1定义（图431）**

时间密钥混合函数的阶段1的输入应该是临时密钥（TK），TA和TSC。临时密钥的长度应为128位。在阶段1中仅使用TSC的32个MSB和所有临时密钥。输出TTAK的长度应为80位，并由16位值的数组表示：TTAK0TTAK1TTAK2TTAK3TTAK4。

阶段1算法的描述将以下所有值视为8位值的数组：TA0..TA5，TK0..TK15。TA八位位组顺序根据7.1.1中的约定表示，前三个八位位组表示OUI。

在阶段1规范中使用XOR（⊕）操作，逐位和（＆）操作以及加法（+）操作。还使用循环计数器i和数组索引临时变量j。

一个函数Mk16用于阶段1的定义。函数Mk16从两个8位输入构造一个16位值，如Mk16（X，Y）=（256⋅X）+Y.

两个步骤组成了阶段1算法。第一步是从TSC和TA初始化TTAK。第二步使用S盒迭代地将密钥材料混合到80位TTAK中。第二步将PHASE1\_LOOP\_COUNT设置为8。

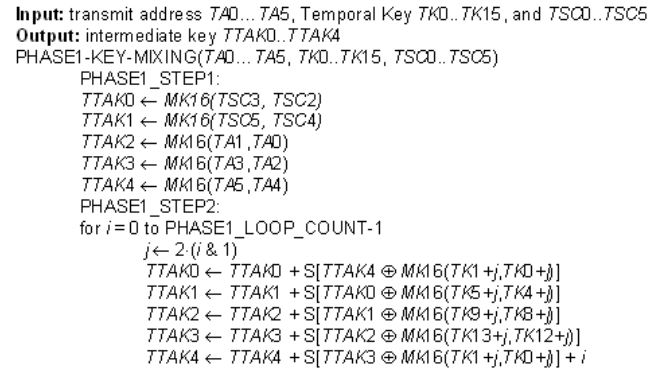


图43l-阶段1键混合

**注意：**

1-TA混合到散列函数的阶段1中的临时密钥中。通过高速缓存阶段1的输出，实现可以实现显着的性能提升。对于来自相同临时密钥和TA的216=65536个连续帧，阶段1输出是相同的。考虑STA仅与AP通信的简单情况。STA将使用其自己的地址执行阶段1，并且TTAK将用于保护发送到AP的流量。STA将使用AP地址执行阶段1，并将用于解包从AP接收的流量。

2-当需要更新TSC换行的低16位和高32位时，需要更新来自阶段1的高速缓存的TTAK。

**8.3.2.5.3第2阶段定义（见图43m）**

时间密钥混合函数的阶段2的输入应该是阶段1（TTAK）的输出以及临时密钥和TSC。TTAK的长度为80位。在阶段2中仅使用TSC的16个LSB。临时密钥是128位。输出是WEP种子，它是每帧密钥，长度为128位。构建的WEP种子具有符合WEP规范的内部结构。换句话说，WEP种子的前24位应以明文形式作为WEPIV发送。这样，这24位用于将TSC的低16位从发送方（加密方）传送到接收方（解密方）。TSC的其余部分应在扩展IV字段中传送。临时密钥和TTAK值表示为阶段1.WEP种子被视为8位值的数组：WEPSeed0...WEPSeed15。TSC应被视为8位值的数组：TSC0TSC1TSC2TSC3TSC4TSC5。

指定阶段2混合函数的伪代码使用一个变量：PPK，其长度为96位。PPK表示为16位值的数组：PPK0..PPK5。伪代码也使用循环计数器，即。如本子条款所述，从16位PPK值到8位WEPseed值的映射显然是小端，以匹配用于此应用程序的最常用处理器的端序架构。

在阶段2的规范中使用XOR（⊕）操作，加法（+）操作，AND（＆）操作，OR（|）操作和右移位（>>）操作。

算法规范依赖于四个函数：

-第一个函数Lo8引用16位输入值的8个LSB。

-第二个函数Hi8引用16位值的8个MSB。

-第三个函数RotR1将其16位参数向右旋转1位。

-第四个函数Mk16已在阶段1中使用，由Mk16（X，Y）=（256⋅X）+Y定义，并构造来自两个8位输入的16位输出。

**注-**STEP2中的旋转和加法操作使第2阶段对处理器的endian体系结构特别敏感，尽管由于在大端处理器上运行此算法而导致的性能下降应该很小。

第2阶段包括三个步骤：

-STEP1复制TTAK并引入TSC。

-STEP2是96位双射混合，采用S盒。

-STEP3引入最后一个临时密钥TK位并分配24位WEPIV值。

WEPIV格式带有3个八位字节。阶段2的步骤3确定这三个八位字节中的每一个的值。选择该结构是为了排除使用已知的RC4弱键。接收方可以通过连接第三个和第一个八位字节来重建发起方使用的TSC的16个LSB，而忽略第二个八位字节。TSC的剩余32位从扩展IV字段获得。

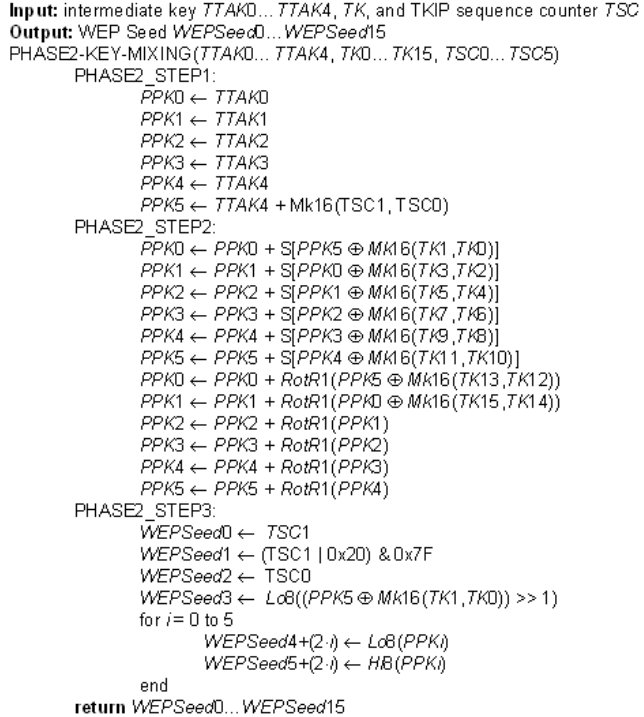


图43m-Phase2键混合

**8.3.2.6TKIP重播保护程序**

TKIP实现应使用TSC字段通过实施以下规则来防御重放攻击：

a）每个MPDU应具有唯一的TKIPTSC值。

b）每个发送器应为每个PTKSA，GTKSA和STAKeySA维护一个TSC（48位计数器）。

c）TSC应实现为48位单调递增计数器，当初始化或刷新相应的TKIP临时密钥时初始化为1。

d）WEPIV格式携带48位TSC的16个LSB，由TKIP混合功能（第2阶段，第3阶段）定义。TSC的其余部分在扩展IV字段中携带。

e）接收方应为每个PTKSA，GTKSA和STAKeySA维护一套单独的TKIPTSC重播计数器。

f）TKIP重放检测在MIC验证和ACK处理所需的任何重新排序之后进行。因此，接收方应延迟推进TKIPTSC重放计数器，直到MSDU通过MIC检查，以防止攻击者向MPDU注入有效的ICV和TSC，但无效的MIC。

注意-这是有效的，因为如果攻击者修改了TSC，则加密密钥被修改，因此ICV和MIC通常都会被错误地解密，从而导致接收到的MPDU被丢弃。

g）对于每个PTKSA，GTKSA和STAKeySA，接收方应为每个帧优先级维护一个单独的重播计数器，并应使用从接收帧恢复的TSC来检测重放帧，但受限于所支持的重播计数器数量的限制在RSNCapabilities字段中，如7.3.2.25中所述。当从接收帧中提取的TSC小于或等于帧优先级的当前重放计数器值时，发生重放帧。发送器不得重新排序具有不同优先级的帧，而不确保接收器支持所需数量的重放计数器。发送器不应对重放计数器内的帧重新排序，但可以跨重放计数器重新排序帧。重新排序帧的一个可能原因是IEEE802.11MSDU优先级。IEEE802.11没有定义信号帧优先级的方法。

h）接收方应丢弃无序接收的任何MPDU，并应为此密钥增加dot11RSNAStatsTKIPReplays的值。

**8.3.3具有CBC-MAC协议（CCMP）的CTR**

本子条款规定了CCMP，它提供机密性，身份验证，完整性和重播保护。CCMP对于RSN合规性是强制性的。

**8.3.3.1CCMP概述**

CCMP基于AES加密算法的CCM。CCM结合了CTR的机密性和CBC-MAC的身份验证和完整性。CCM保护MPDU数据字段和IEEE802.11MPDU报头的选定部分的完整性。

AES算法在FIPSPUB197中定义.CCMP中使用的所有AES处理都使用具有128位密钥和128位块大小的AES。

CCM在IETFRFC3610中定义.CCM是一种通用模式，可以与任何面向块的加密算法一起使用。CCM有两个参数（M和L），CCMP使用以下CCM参数值：

-M=8;表明MIC是8个八位字节。

-L=2;指示长度字段是2个八位字节，这足以保持以八位字节表示的最大可能IEEE802.11MPDU的长度。

CCM需要每个会话的新临时密钥。对于受给定临时密钥保护的每个帧，CCM还需要唯一的随机数值，CCMP为此使用48位分组编号（PN）。使用相同的临时密钥重用PN会使所有安全保证失效。

附件H提供了CCM的测试向量。

**8.3.3.2CCMPMPDU格式**

图43n描绘了使用CCMP时的MPDU。

CCMP处理将原始MPDU大小扩展16个八位字节，CCMP报头字段为8个八位字节，MIC字段为8个八位字节。CCMP标头字段由PN，ExtIV和KeyID子字段构成。PN是48位PN，表示为6个八位字节的数组。PN5是PN的最重要的八位字节，PN0是最不重要的。请注意，CCMP不使用WEPICV。

密钥ID八位字节的ExtIV子字段（比特5）用信号通知CCMP报头字段将MPDU报头扩展总共8个八位字节，与使用WEP时添加到MPDU报头的4个八位字节相比。对于CCMP，ExtIV位（第5位）始终设置为1。

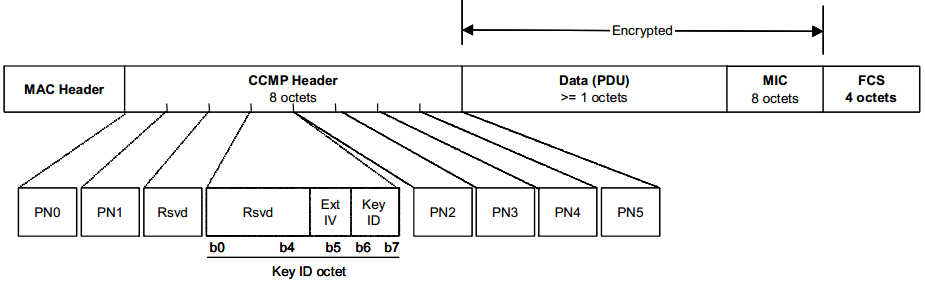


图43n-ExpandedCCMPMPDU

密钥ID八位字节的比特6-7用于密钥ID子字段。

保留位应设置为0，接收时应忽略。

**8.3.3.3CCMP封装**

CCMP封装过程如图43o所示。

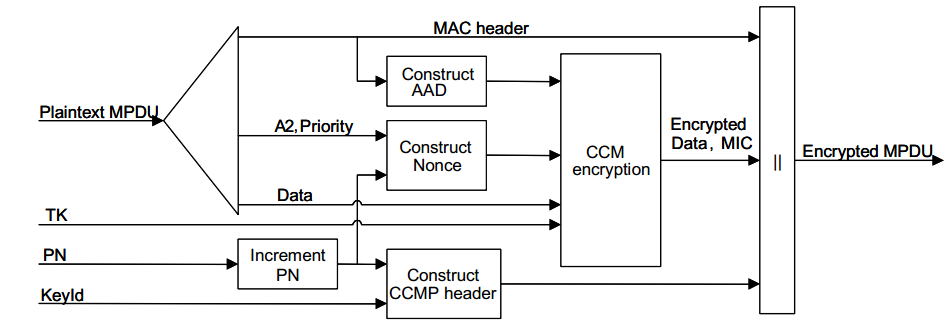


图43o-CCMP封装框图

CCMP使用以下步骤加密明文MPDU的有效负载并封装生成的密文：

a）递增PN，以获得每个MPDU的新PN，以便PN永远不会重复相同的临时密钥。请注意，重传时不会修改重新传输的MPDU。

b）使用MPDU报头中的字段为CCM构建附加认证数据（AAD）。CCM算法为AAD中包含的字段提供完整性保护。通过在计算AAD时将其屏蔽为0，可以在重传时改变的MPDU报头字段被静音。

c）根据MPDU的PN，A2和优先级字段构造CCMNonce块，其中A2是MPDU地址2.优先级字段的保留值设置为0。

d）将新的PN和密钥标识符放入8个八位字节的CCMP头中。

e）使用临时密钥，AAD，随机数和MPDU数据来形成密文和MIC。此步骤称为CCM创建者处理。

f）通过组合原始MPDU报头，CCMP报头，加密数据和MIC形成加密的MPDU，如8.3.3.2中所述。

CCM参考描述了密钥，随机数，AAD和数据的处理，以产生加密输出。有关从MPDU创建AAD和nonce以及相关的MPDU特定处理的详细信息，请参见8.3.3.3.1到8.3.3.3.5。

**8.3.3.3.1PN处理**

对于每个MPDU，PN递增正数。对于使用相同临时密钥的一系列加密MPDU，PN永远不会重复。

**8.3.3.3.2构建AAD**

AAD的格式如图43p所示。

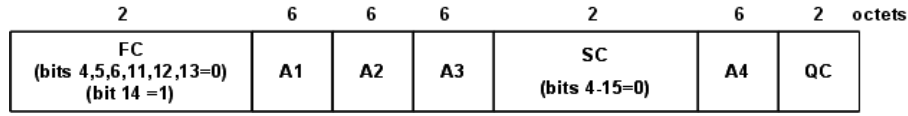


图43p-AAD结构

AAD由MPDU报头构成。AAD不包括标题持续时间字段，因为持续时间字段值可能由于正常的IEEE802.11操作而改变（例如，重传期间的速率改变）。出于类似的原因，帧控制字段中的几个子字段被屏蔽为0.AAD构造按如下方式执行：

a）FC-MPDU帧控制字段，带

1）子类型位（位456）被屏蔽为0

2）重试位（位11）被屏蔽为0

3）PwrMgt位（位12）被屏蔽为0

4）MoreData位（位13）被屏蔽为0

5）受保护的帧位（位14）始终设置为1

b）A1-MPDU地址1字段。

c）A2-MPDU地址2字段。

d）A3-MPDU地址3字段。

e）SC-MPDU序列控制字段，其序列号子字段（序列控制字段的第4-15位）被屏蔽为0.片段号子字段未被修改。

f）A4-MPDU地址字段，如果存在于MPDU中。

g）QC-服务质量控制字段，如果存在，则包括MSDU优先级的2字节字段;该字段留作将来使用。

当没有A4字段和没有QC字段时，AAD的长度是22个八位字节，当MPDU包括A4字段时，AAD的长度是28个八位字节。

**8.3.3.3.3构造CCMnonce**

Nonce字段占用13个八位字节，其结构如图43q所示。



图43q-Nonce构造

Nonce字段具有PriorityOctet||的内部结构A2||PN（“||”是串联），其中

-此优先级八位字节字段应为0，并保留供将来使用IEEE802.11帧优先级。

-MPDU地址A2字段占用八位字节1-6。这应该用在八位字节索引1处的A2八位字节0和在八位字节索引6处的A2八位字节5排序的八位字节进行编码。

-PN字段占用八比特组7-12。PN的八位字节应按顺序排列，使PN0位于八位位组索引12，PN5位于八位位组索引7处。

**8.3.3.3.4构造CCMP头**

8字节CCMP报头的格式在8.3.3.2中给出。标头对用于加密MPDU的PN，密钥ID和ExtIV字段值进行编码。

**8.3.3.3.5CCM发起者处理**

CCM是一种通用的验证和加密分组密码模式，在此修订中，CCM与AES分组密码一起使用。

CCM创建者处理有四个输入：

a）密钥：临时密钥（16个八位字节）。

b）Nonce：如8.3.3.3.3所述构造的nonce（13个八位字节）。

c）帧体：MPDU的帧体（1-2296个八位字节;2296=2312-8个MIC八位字节--8个CCMP头八位字节）。

d）AAD：根据8.3.3.3.2中描述的MPDU报头构造的AAD（22-30个八位字节）。

CCM发起者处理提供帧体和AAD的认证和完整性以及帧体的机密性。CCM发起者处理的输出包括加密数据和8个加密MIC的八位字节（见图43n）。

**8.3.3.4CCMP解封装**

图43r描绘了CCMP解封装过程。

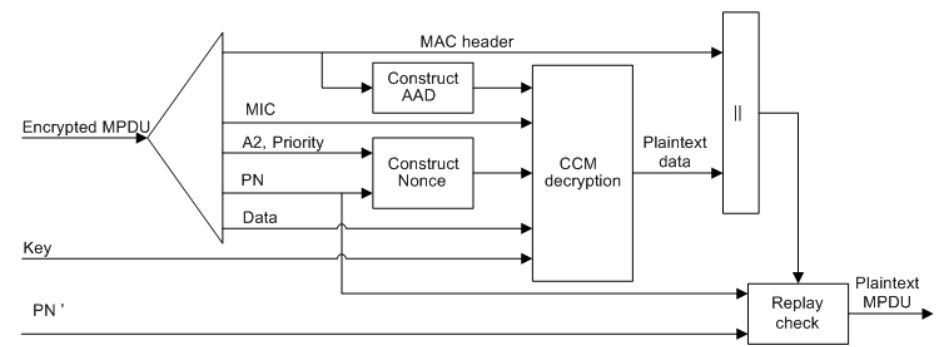


图43r-CCMP解封装框图

CCMP解密密文MPDU的有效载荷，并使用以下步骤解封装明文MPDU：

a）解析加密的MPDU以构造AAD和nonce值。

b）AAD由加密的MPDU的MPDU报头构成。

c）nonce值由A2，PN和PriorityOctet字段构成（保留并设置为0）。

d）提取MIC以用于CCM完整性检查。

e）CCM接收者处理使用临时密钥，AAD，随机数，MIC和MPDU密文数据来恢复MPDU明文数据以及检查AAD和MPDU明文数据的完整性。

f）可以连接从CCM接收者处理接收的MPDU报头和MPDU明文数据，以形成明文MPDU。

g）解密处理通过验证MPDU中的PN大于为会话维护的重放计数器来防止重放MPDU。

有关此处理的详细信息，请参见8.3.3.4.1至8.3.3.4.3。

**8.3.3.4.1CCM接收者处理**

CCM收件人处理必须使用与CCM创建者处理相同的参数。

CCM收件人处理有四个输入：

-密钥：临时密钥（16个八位字节）。

-Nonce：如8.3.3.3.3所述构造的nonce（13个八位字节）。

-加密帧体：来自接收到的MPDU的加密帧体。加密的帧体包括8个八位字节的MIC（9-2304个八位字节）。

-AAD：AAD（22-30个八位字节），它是8.3.3.3.2中描述的规范MPDU报头。

CCM接收方处理检查帧体和AAD的认证和完整性以及解密帧体。仅在MIC检查成功时才返回明文。

从无错误的CCM收件人处理中有一个输出：

-帧体：明文帧体，比加密的帧体小8个八位字节。

**8.3.3.4.2解密的CCMPMPDU**

当计算出的MIC与从解密接收的加密MPDU获得的MIC值匹配时，解封装过程成功。原始MPDU报头与由成功的CCM接收者处理产生的明文数据连接以创建明文MPDU。

**8.3.3.4.3PN和重放检测**

为了实现重放检测，接收器从CCMP报头中提取PN。有关如何在CCMP报头中编码PN的说明，请参见8.3.3.2。以下处理规则用于检测重放：

a）PN值按顺序对每个MPDU进行编号。

b）每个发射机应为每个PTKSA，GTKSA和STAKeySA保持一个PN（48位计数器）。

c）PN应实现为48位单调递增非负整数，当初始化或刷新相应的临时密钥时初始化为1。

d）接收方应为每个PTKSA，GTKSA和STAKeySA维护一组单独的PN重放计数器。当接收器重置对等体的临时密钥时，接收器将这些重放计数器初始化为0。重播计数器设置为接受的CCMPMPDU的PN值。

e）对于每个PTKSA，GTKSA和STAKeySA，接收方应为每个IEEE802.11MSDU优先级维护一个单独的重播计数器，并应使用从接收帧恢复的PN来检测重放帧，但须遵守

限制RSN功能字段中指示的受支持重播计数器的数量（见7.3.2.25）。当从接收帧提取的PN小于或等于帧的MSDU优先级的当前重放计数器值时，发生重放帧。如果没有确保接收器支持所需数量的重放计数器，则发送器不得使用IEEE802.11MSDU优先级。发送器不应对重放计数器内的帧重新排序，但可以跨重放计数器重新排序帧。重新排序帧的一个可能原因是IEEE802.11MSDU优先级。

f）接收方应丢弃其组成MPDUPN值不是顺序的MSDU。接收方应丢弃其PN小于或等于重放计数器的任何MPDU，并为该密钥增加dot11RSNAStatsCCMPReplays的值。

**8.4RSNA安全关联管理**

**8.4.1安全协会**

**8.4.1.1安全关联定义**

IEEE802.11使用安全关联的概念来描述安全操作。安全通信仅在安全关联的上下文中是可能的，因为这是提供IEEE802.11密码套件的正确操作所需的状态加密密钥，计数器，序列空间等的上下文。

安全关联是一组用于保护信息的策略和密钥。安全关联中的信息由安全关联的每一方存储，必须在所有各方之间保持一致，并且必须具有标识。标识是密钥的紧凑名称和安全关联信息的其他位，以适合表索引或MPDU。RSNSTA支持四种类型的安全关联：

-PMKSA：成功的IEEE802.1x交换，预共享PMK信息或通过其他机制缓存的PMK的结果。

-PTKSA：成功的四方握手的结果。

-GTKSA：成功的群组握手或成功的4次握手的结果。

-STAKeySA：STAKey握手成功的结果。

**8.4.1.1.1PMKSA**

当PMKSA是成功的IEEE802.1X身份验证的结果时，它来自AS提供的EAP身份验证和授权参数。此安全关联是双向的。换句话说，双方都使用安全关联中的信息进行发送和接收。当EAP身份验证成功完成或配置PSK时，PMKSA由Supplicant的SME创建。当根据从AS传输的密钥信息创建PMK或配置PSK时，PMKSA由Authenticator的SME创建。PMKSA用于创建PTKSA。PMKSA缓存到它们的生命周期。PMKSA由以下元素组成：

-PMKID，如8.5.1.2中所定义。PMKID标识安全关联。

-身份验证器MAC地址。

-PMK。

-寿命，如8.5.1.2中所定义。

-AKMP。

-AS或本地配置指定的所有授权参数。这可以包括诸如STA的授权SSID之类的参数。

**8.4.1.1.2PTKSA**

PTKSA是4次握手的结果。此安全关联也是双向的。PTKSA用于创建密钥层次结构。PTKSA在PMKSA的生命周期中被缓存。由于PTKSA与PMKSA绑定，因此它仅具有来自4次握手的附加信息。只有一个PTKSA具有相同的请求方和身份验证方MAC地址。在4次握手的消息1和消息3之间创建了状态。在请求者验证消息3并且验证者验证消息4之前，这不会创建PTKSA。PTKSA包含以下要素：

-PTK

-成对密码套件选择器

-请求者MAC地址

-身份验证器MAC地址

**8.4.1.1.3GTKSA**

GTKSA来自成功的4次握手或组密钥握手，是单向的。在ESS中，存在一个GTKSA，专门用于加密由AP发送的广播/多播MPDU以及用于解密由STA接收的广播/多播传输。在IBSS中，每个STA定义其自己的GTKSA，其用于加密其广播/多播传输，并为每个对等STA存储单独的GTKSA，以便可以解密从其他STA接收的加密的广播/多播流量。当接收到4次握手的消息3或者接收到组密钥握手的消息1时，由请求方的SME创建GTKSA。当SME更改GTK并将GTK发送给具有PTKSA的所有STA时，GTKSA由Authenticator的SME创建。GTKSA由以下元素组成：

-方向向量（GTK是用于发送还是接收）。

-组密码套件选择器。

-GTK。

-身份验证器MAC地址。

-本地配置指定的所有授权参数。这可以包括诸如STA的授权SSID之类的参数。

当GTK用于加密单播流量时（可选择的密码套件是“使用组密钥”），GTKSA是双向的。

**8.4.1.1.4STAKeySA**

STAKeySA是STAKey握手的结果。此安全关联是从发起方到对等方的单向链接。只有一个STAKeySA具有相同的启动器和对等MAC地址。使用相同的启动器和对等MAC地址创建新的STAKeySA将导致删除现有的STAKeySA。当STAKey握手的消息1被验证时，创建STAKeySA。STAKeySA由以下元素组成：

-STAKey

-成对密码套件选择器

-发起者MAC地址

-对等MAC地址

**8.4.1.2安全关联生命周期**

STA可以在ESS或IBSS中操作，并且安全关联对于每个都具有不同的生命周期。

**8.4.1.2.1ESS中的安全关联**

在ESS中有两种情况：

-STA和ESS之间的初始联系

-STA在ESS内漫游

STA和AP通过以下步骤建立初始安全关联：

a）STA通过在通告适当SSID的AP中进行选择来选择授权的ESS。

b）然后，STA使用IEEE802.11开放系统认证，然后与所选AP相关联。安全参数的协商在关联期间发生。

**注：**

1-可能存在多个PMKSA。例如，第二个PMKSA可能通过PMKSA缓存而存在。STA可能会离开ESS并刷新其缓存。在其PMKSA在AP缓存中到期之前，STA返回ESS并从AP的角度建立第二个PMKSA。

2-密钥派生程序将检测到改变安全参数的攻击。

3-IEEE802.11开放系统认证不提供安全性，但包含在内以保持与IEEE802.11状态机的向后兼容性（见5.5）。

1. AP的Authenticator或STA的Supplicant启动IEEE802.1X身份验证。IEEE802.1X使用的EAP方法将支持相互认证，因为STA需要保证AP是合法AP。

**注：**

1-在完成IEEE802.1X身份验证和密钥安装之前，AP中的IEEE802.1X受控端口将阻止所有数据帧。IEEE802.1X受控端口返回到未授权状态，并在调用MLME-DELETEKEYS.request原语之前阻止所有数据帧。IEEE802.1XUncontrolledPort允许IEEE802.1X帧在Supplicant和Authenticator之间传递。尽管IEEE802.1X不需要请求方受控端口，但此修订假定请求方具有受控端口以提供所需的安全级别。没有受控端口的请求者会破坏RSN安全性，不应使用。

2-任何安全网络都不能支持混杂关联，例如，IEEE802.11的不安全操作。在关联和安全操作之前，STA和目标SSID的AS之间必须存在信任关系，以使关联可信。原因是攻击者可以像合法网络提供商部署合法AP一样轻松部署欺诈AP，因此在ESS和STA之间建立凭证需要某种先验关系。

d）最后一步是密钥管理。验证过程创建IEEE802.1XAS和STA之间共享的加密密钥。AS将这些密钥传送到AP，AP和STA使用一个密钥确认握手（称为4次握手）来完成安全关联建立。密钥确认握手指示链接何时被密钥保护并准备好允许正常数据流量。

在ESS内漫游的STA通过以下三种方案之一建立新的PMKSA：

-在（重新）关联后跟随IEEE802.1X或PSK认证的情况下，STA重复与初始联系人关联相同的动作，但其请求者在从旧AP漫游时也删除PTKSA。当STA的请求者与ESS中的所有基本服务集标识符（BSSID）解除关联/解除认证时，它也删除PTKSA。

-STA（AP）可以保留其先前已执行完整IEEE802.1X认证的ESS中的AP（STA）的PMK。如果STA希望漫游到已经缓存了一个或多个PMKSA的AP，则它可以在其（重新）关联请求帧的RSN信息元素中包括一个或多个PMKID。其Authenticator为一个或多个PMKID保留PMK的AP可以跳过802.1X身份验证并继续进行4次握手。AP应在4次握手的消息1中包括所选PMK的PMKID。如果缓存的PMKSA的PMKID都不匹配任何提供的PMKID，则Authenticator应执行另一个IEEE802.1X身份验证。类似地，如果STA未能发送PMKID，则STA和AP必须执行完整的IEEE802.1X认证。

-已经与ESS关联的STA可以在关联到该新AP之前请求其IEEE802.1X请求方对新AP进行认证。通过旧AP的DS的正常操作提供STA与新AP之间的通信。STA的IEEE802.11管理实体延迟与新AP的重新关联，直到通过DS完成IEEE802.1X身份验证。如果IEEE802.1X身份验证成功完成，则将缓存新AP和STA之间共享的PMKSA，从而实现可能的重新关联使用，而无需后续的完整IEEE802.1X身份验证过程。

MLME-DELETEKEYS.request原语破坏为安全关联建立的临时密钥，因此它们不能用于保护后续的IEEE802.11流量。STA的SME在删除PTKSA或GTKSA时使用该原语。

**8.4.1.2.2IBSS中的安全关联**

在IBSS中，当STA的SME与对等STA建立安全关联时，它为对等体创建IEEE802.1X请求者和认证者。

STA可以从先前未知的MAC地址接收IEEE802.1X消息。

IBSS内的任何STA可拒绝与加入IBSS的STA形成安全关联。形成安全关联的尝试也可能失败，因为例如，对等体使用与STA期望的不同的PSK。

在IBSS中，每个STA定义其自己的组密钥，即GTK，以保护其广播/多播传输。每个STA应使用4次握手或组密钥握手将其发送GTK分配给其新的对等STA。当STA生成新的GTK时，它还使用组密钥握手将新的GTK分发给每个已建立的对等体。

**8.4.2RSNA选择**

准备建立RSNA的STA应通过在信标和探测响应消息中包括RSN信息元素来通告其能力。所包含的RSN信息元素应指定由STA的策略启用的所有认证和密码套件。STA不得通告任何未启用的身份验证或密码套件。

STA的IEEE802.11管理实体将利用MLME-SCAN.request原语来识别断言鲁棒安全性的相邻STA并通告标识授权的ESS或IBSS的SSID。STA可能拒绝与未在其信标和探测响应帧中通告RSN信息元素或未通告授权SSID的STA通信。STA还可以拒绝与不在其RSN信息元素内通告授权认证和密码套件的其他STA进行通信。

STA应在其信标和探测响应帧中通告相同的RSN信息元素。

注：

1-具有强健安全性的STA是否可以尝试与不包括RSN信息元素的STA通信是策略问题。

2-实际上，如果最大互操作性是目标，AP将支持TKIP以及CCMP。

处理RSN信息元素时，STA应遵守以下规则：

-STA应通告其支持的最高版本。

-STA应请求其支持的最高版本字段值小于或等于对等STA通告的版本。

-没有重叠支持的版本字段值的两个对等STA不应使用RSNA方法来保护它们的通信。

-STA应忽略它无法识别的套件选择器。

**8.4.3ESS中的RSNA策略选择**

ESS中的RSNA策略选择使用正常的IEEE802.11关联过程。RSNA策略选择由关联STA执行。STA通过在其（重新）关联请求中包括RSN信息元素来做到这一点。

在RSN中，AP不应与前RSNASTA相关联，即，与不能在关联或重新关联请求帧中包括RSN信息元素的STA相关联。

发起关联的STA的SME应将RSN信息元素插入其（Re）关联请求中;当目标AP指示RSNA支持时，通过MLME-ASSOCIATE.request原语。发起STA的RSN信息单元应包括目标AP在其信标和探测响应帧中通告的那些中的一个认证和成对密码套件。它还应指定目标AP指定的组密码套件。如果来自AP的RSN信息元素的至少一个RSN信息元素字段未能与STA支持的任何值重叠，则STA将拒绝与该AP关联。

如果支持RSNA的AP接收到包含RSN信息元素的（重新）关联请求，并且如果它选择接受该关联作为安全关联，则它应使用（重新）关联请求中的认证和成对密码套件，除非AP包括4次握手的消息3中的可选的第二RSN信息元素。如果在消息3中提供第二RSN信息元素，则安全关联使用的成对密码套件（如果建立的话）应该是来自第二RSN信息元素的成对密码。

为了适应本地安全策略，STA可以选择不与不支持任何成对密码套件的AP关联。AP通过将“使用组密钥”作为成对密码套件选择器进行通告来指示它不支持任何成对密钥。

**注-**当ESS使用PSK时，STA协商成对密码。但是，ESS中的任何STA都可以通过捕获4次握手的前两个消息来导出使用相同PSK的任何其他成对密钥。这为恶意内部人员提供了窃听能力以及建立中间人攻击的能力。

**8.4.3.1ESS中的TSN策略选择**

在TSN中，RSNSTA应在其（Re）关联请求中包括RSN信息元素。

被配置为在TSN中操作的具有RSNA能力的AP应包括RSN信息元素，并且可以与RSNA和前RSNASTA相关联。换句话说，具有RSNA能力的AP将响应于包括RSN信息元素的关联STA，就像在RSN中一样。

如果在TSN内操作的AP在没有RSN信息元素的情况下接收（重新）关联请求，则其IEEE802.1X受控端口最初将被阻止。启用WEP后，SME将取消阻止IEEE802.1X受控端口。

**8.4.4IBSS中的RSNA策略选择**

在IBSS中，所有STA必须使用单个组密码套件，并且所有STA必须支持成对密码套件的公共子集。但是，任何一对STA的SME都可以协商使用它们都支持的任何常见的成对密码套件。每个STA应在其信标和探测响应消息中包括组密码套件及其成对密码套件列表。如果两个STA已经通告了相同的组密码套件，则它们可能只建立PMKSA。类似地，如果STA已经通告了不成对的成对密码套件，则两个STA不应建立PMKSA。

当IBSSSTA的SME想要与对等STA建立安全关联但不知道对等方的策略时，它必须首先使用探测请求帧获取对等方的安全策略。两个STA的SME实体使用4次握手之一选择成对密码套件。IBSS内的每对STA的SME可以使用EAPOL-Key4次握手来选择成对密码套件。如8.5.2中所规定的，4次握手的消息2和消息3传达RSN信息元素。消息2RSN信息元素包括所选择的成对密码套件，消息3包括STA将在探测响应帧中发送的RSN信息元素。

这对STA应使用由AuthenticatorSTA发送的具有较高MAC地址的4次握手的消息3中指定的成对密码套件（见8.5.1）。

中小企业应检查组密码套件和AKMP是否与IBSS的信标和探测响应帧中的密码套件匹配。

**注：**

1-消息2和消息3中的RSN信息元素与信标帧中的RSN信息元素不同。组密码和AKMP是相同的，但是成对密码可以不同，因为来自不同STA的信标帧可以通告不同的成对密码。因此，IBSS中的STA使用相同的AKM套件和组密码，而在STA对之间可以使用不同的成对密码。

2-当IBSS网络使用PSK时，STA可以协商成对密码。但是，IBSS中的任何STA都可以通过捕获4次握手的前两个消息来导出使用相同PSK的任何其他PTK。这为恶意内部人员提供了窃听能力以及建立中间人攻击的能力。

**8.4.4.1IBSS中的TSN策略选择**

前RSNASTA生成没有RSN信息元素的信标和探测响应帧，并且将忽略RSN信息元素，因为它们是未知的。这允许RSNASTA识别从其接收到信标和探测响应帧的前RSNASTA。

如果RSNASTA的SME通过MLMEPROTECTEDFRAMEDROPPED.indication原语基于接收的广播/多播消息识别可能的IBSS成员，则它不能直接识别对等体的安全策略。SME可以尝试通过探测请求帧获得对等STA的安全策略。

**8.4.5IEEE802.1X受控端口的RSN管理**

当策略选择过程选择IEEE802.1X身份验证时，此修订假定IEEE802.1XSupplicants和Authenticators通过IEEE802.1XUncontrolled端口交换协议信息。阻止IEEE802.1X受控端口在STA之间传递通用数据流量，直到IEEE802.1X身份验证过程通过IEEE802.1X非受控端口成功完成。RSNA的安全性取决于这种假设是正确的。

在ESS中，STA通过调用MLME-ASSOCIATE.confirm或MLME-REASSOCIATE.confirm原语来指示IEEE802.11链路可用。这向请求者发信号通知MAC已从禁用状态转换为启用状态。此时，请求方的受控端口被阻止，并且未授权通过该端口发送或接收的所有非IEEE802.1XMSDU的通信。

在ESS中，AP通过调用MLME-ASSOCIATE.indication或MLME-REASSOCIATE.indication原语来指示IEEE802.11链路可用。此时，对应于STA关联的Authenticator的受控端口被阻止，并且未授权通过受控端口发送或接收的所有非IEEE802.1XMSDU的通信。

在IBSS中，STA应在初始化时阻止所有IEEE802.1X端口。通过受控端口发送或接收的所有非IEEE802.1XMSDU的通信未经授权。

此修订假定每个受控端口保持阻塞状态，直到IEEE802.1X状态变量portValid和keyDone都变为真。该假设意味着IEEE802.1X受控端口在将加密密钥安装到MAC之前丢弃通过IEEE802.11信道发送的MSDU。这可以保护STA的主机免受伪造的MSDU在其初始化时写入该通道。

MAC不区分受控端口的MSDU和非受控端口的MSDU。换句话说，只有在调用MLMESETPROTECTION.request原语后才会加密IEEE802.1XEAPOL帧。

此修订假定IEEE802.1X在通过重新认证触发认证时不会阻止受控端口。在IEEE802.1X重新认证期间，现有RSNA可以保护STA之间交换的所有MSDU。在通过RSNA重新认证期间不需要阻止MSDU。

**8.4.6ESS中的RSNA认证**

在建立RSNA时，STA应在（重新）关联之前使用IEEE802.11开放系统认证。

IEEE802.1X身份验证由以下任一机制启动：

-如果STA在（重新）关联期间协商使用IEEE802.1X认证，则STA的管理实体可以通过请求STA的请求者（或AP的认证者）发起IEEE802.1X来响应MLME-ASSOCIATE.confirm（或指示）原语。认证。因此，在这种情况下，认证由STA的关联决定和AP接受关联的决定驱动。

-如果STA的MLME-SCAN.confirm原语在当前ESS内找到另一个AP，则STA可以发信号通知其请求方使用IEEE802.1X来预先验证该AP。

注-漫游STA的IEEE802.1X请求者可以通过其旧AP通过DS向新AP发送EAPOL-Start消息来启动预身份验证。

-如果STA收到IEEE802.1X消息，则将其传递给其Supplicant或Authenticator，后者可以启动新的IEEE802.1X身份验证。

**8.4.6.1预身份验证和RSNA密钥管理**

除非采用成对密钥，否则STA不应使用预身份验证。除非新AP在RSN信息元素中通告预身份验证功能，否则不应使用预身份验证。

使用预身份验证时，则

a）身份验证独立于漫游。

b）STA的请求者可以一次向多个AP进行认证。

注-预身份验证可用作性能增强，因为重新关联不包括使用时完全重新身份验证的协议开销。

预身份验证使用IEEE802.1X协议和状态机与EtherType88-C7，而不是EtherType88-8E。只有IEEE802.1X帧类型EAP-Packet和EAPOL-Start对预身份验证有效。

注-某些IEEE802.1X认证器可能无法桥接IEEE802.1X帧，如IEEEP802.1X-REV的C.1.1中所述。预身份验证使用不同的EtherType来使这些设备能够桥接预身份验证帧。

STA的请求者可以在完成4次握手并配置所需的临时密钥时启动预身份验证。为了实现预身份验证，STA的请求者发送IEEE802.1XEAPOL-Start消息，其中DA是目标AP的BSSID，RA是与其相关联的AP的BSSID。目标AP应使用等于其认证者的MAC地址的BSSID。由于预身份验证帧不使用IEEE802.1XEAPOLEtherType字段，因此STA当前关联的AP无需应用任何特殊处理。STA中的AP和MAC应以与其他具有任意EtherType字段值的帧相同的方式处理这些帧，这些帧需要通过DS进行分配。

通过DS接收EAPOL-Start消息的AP的认证器可以通过DS向STA发起IEEE802.1X认证。DS将该消息转发给与STA关联的AP。

如果IEEE802.1X身份验证成功完成，则预身份验证的结果可能是PMKSA。如果预身份验证产生PMKSA，则当请求者的STA与预认证的AP关联时，请求者可以使用具有4次握手的PMKSA。

通过IEEE802.1X成功完成EAP身份验证可在Supplicant上建立PMKSA。当AS完成认证时，认证者具有PMKSA，将密钥信息（认证，授权和计费[AAA]密钥，其中一部分是PMK）传递给认证者，并且认证者使用PMK创建PMKSA。。PMKSA被插入到PMKSA缓存中。因此，如果Supplicant和Authenticator失去与PMKSA的同步，则4次握手将失败。在这种情况下，MIB变量dot11RSNAStats-4Way-HandshakeFailures应递增。

无论目标AP是否在无线电范围内，STA的请求者都可以启用其当前ESS内的任何AP的预身份验证，并启用预身份验证。

即使STA已经预先认证，它仍然可能必须经历完整的IEEE802.1X认证，因为AP的认证器可能已经清除其PMKSA，例如，由于资源不可用，STA关联的延迟等。

**8.4.6.2缓存的PMKSA和RSNA密钥管理**

STA可以保留由于先前的认证而建立的PMKSA。缓存时无法更改PMKSA。PMKSA中的PMK与4次握手一起用于建立新的PTK。

如果ESS中的非APSTA已经确定其具有其将要（重新）关联的AP的有效PMKSA，则其包括（重新）关联请求中的RSN信息元素中的PMKSA的PMKID。在接收到具有一个或多个PMKID的（重新）关联请求时，AP检查其认证者是否保留了PMKID的PMK以及PMK是否仍然有效。如果是这样，它在关联完成后通过开始4次握手声称拥有该PMK;否则它会在关联完成后开始完整的IEEE802.1X身份验证。

如果双方声称拥有缓存的PMKSA，但4次握手失败，则双方可能会删除所选PMKID的缓存PMKSA。

如果STA漫游到预先验证的AP并且STA没有该AP的PMKSA，则STA必须发起完整的IEEE802.1XEAP身份验证。

**8.4.7IBSS中的RSNA认证**

当在IBSS中使用认证时，它由希望建立通信的每个STA驱动。该STA的管理实体选择其可能想要认证的一组STA，然后可以使MAC向每个目标STA发送IEEE802.11开放系统认证消息。候选STA可以从信标帧，探测响应帧和来自相同BSSID的数据帧中识别。在与从数据帧识别的STA通信之前，可以例如通过向STA发送探测请求帧并获得探测响应帧来获得STA的安全策略。希望响应的目标STA可以向发起STA返回IEEE802.11开放系统认证消息。

当使用IEEE802.1X认证时，STA管理实体将请求其本地IEEE802.1X实体为对等STA创建请求者PAE。请求方PAE将通过向对等方发送EAPOL-Start消息来向对等STA发起认证。STA管理实体还将在接收到EAPOL-Start消息时请求其本地IEEE802.1X实体为对等STA创建认证器PAE。Authenticator将通过发送EAPRequest消息向对等STA发起认证，或者，如果PSK模式有效，则向4次握手的消息1发起认证。

在任何一对STA之间进行初始认证时，不允许在IEEE802.1X消息之外的数据帧在STA对之间流动，直到每对STA中的两个STA已成功完成AKM并且已提供所提供的加密密钥。

在由一对STA的任何STA启动IEEE802.1X重新认证后，数据帧将在认证完成时继续在STA之间流动。在认证过程中超时或失败时，发起重新认证的STA的认证者将导致将Deauthentication消息发送给作为重新认证目标的STA的请求者。Deauthentication消息将导致STA对中的两个STA遵循11.3.3和11.3.4中定义的解除认证过程。

每个STA中的IEEE802.1X重新认证定时器是独立的。如果具有较高MAC地址的STA的重新认证定时器（参见8.5.1进行MAC比较）通过其Authenticator触发重新认证，则其Supplicant必须向具有较低MAC地址的STA的认证者发送EAPOL-Start消息以触发在其他STA上重新认证。该过程使得这对STA在IEEE802.1X重新认证时相对于新时间密钥的推导保持一致状态。

当它因Open而收到MLME-AUTHENTICATE.indicate原语时

系统认证请求，目标STA上的IEEE802.11管理实体，如果要与对等STA建立安全关联，则请求其认证者开始IEEE802.1X认证，即发送EAPRequest/Identity消息或消息请求者的四向握手中的一个。

EAPOL-Key帧用于在Supplicant和Authenticator之间交换信息以协商新的PTK。4次握手从PMK产生单个PTK。4次握手和组密钥握手使用PTK保护GTK，因为它被传送到接收STA。

PSK认证也可以用在IBSS中。当在IBSSSTA之间共享单个PSK时，希望建立通信的STA向目标STA发送4次握手消息1。目标STA利用4次握手的消息2响应消息1，并通过向发起STA发送消息1来开始其4次握手。两个4次握手建立PTKSA和GTKSA以在发起STA和目标STA之间使用。也可以使用PSKPMKID，支持成对PSK。

IBSS中的安全模型不是通用的。特别是，它假设如下：

a）本子条款中描述的认证程序有效的用例集如下：

1）基于PSK的认证，通常由H.4中描述的密码短语哈希方法管理

2）基于EAP的身份验证，使用已在公共管理域（例如单个组织）内的STA上发布和预安装的凭据

b）所有STA都在直接无线电通信中。特别是，第三STA没有路由，桥接或转发流量以实现通信。做出这个假设，因为该模型没有规定保护IBSS拓扑信息免受其中一个成员的篡改。

**8.4.8ESS中的RSNA密钥管理**

当IEEE802.1X身份验证成功完成时，此修订假定STA的IEEE802.1X请求者和IEEE802.1XAS将共享一个秘密，称为PMK。AS使用超出本修订范围的技术将AAA密钥内的PMK传输到AP;从MSK推导出PMK是EAP方法特有的。在PMK到位的情况下，AP发起与STA的密钥确认握手。密钥确认握手将IEEE802.1X状态变量portValid（如IEEEP802.1X-REV中所述）设置为TRUE。

密钥确认握手由4次握手实现。四方握手的目的如下：

a）确认对等体上是否存在PMK。

b）确保安全关联密钥是新鲜的。

c）将临时密钥的安装同步到MAC中。

d）将GTK从Authenticator转移到Supplicant。

e）确认密码套件的选择。

注：

1-可以伪造4次握手的消息1。但是，伪造尝试将在四方握手失败时被发现。

2-AP和STA都不能将PMK用于任何目的，而是使用此处指定的PMK而不会损害密钥。如果AP将其用于其他目的，则STA可伪装成AP;类似地，如果STA在另一个上下文中重用PMK，则AP可以伪装成STA。

Supplicant和Authenticator通过利用MLME-SETKEYS.request原语将商定的时间成对密钥配置到IEEE802.11MAC中并通过调用MLME-SETPROTECTION.request原语来启用其使用来发信号通知密钥管理的完成。

还定义了第二个密钥交换组密钥握手。它分发后续的GTK。AP的Authenticator可以使用组密钥握手在STA的Supplicant上更新GTK。组密钥握手使用EAPOL-Key帧进行此交换。完成后，STA的Supplicant可以使用MLME-SETKEYS.request原语将GTK配置为IEEE802.11MAC。

**8.4.9IBSS中的RSNA密钥管理**

要在IBSS中的两个STA之间建立安全关联，每个STA的SME必须具有附带的IEEE802.1X认证者和请求者。每个STA的SME发起从认证者到对等STA的请求者的4次握手（见8.4.7）。进行两次单独的四向握手。

4次握手用于协商成对密码套件，如8.4.4中所述。IEEE802.11SME将PTK的临时密钥部分配置为IEEE802.11MAC。每个Authenticator使用由其发起的交换协商的PTK的KCK和KEK部分来分发其自己的GTK。每个Authenticator都会生成自己的GTK，并使用4次握手或组密钥握手将GTK传输给其已完成4次握手的其他STA。在任何两个STA之间使用的成对密钥应该是由具有最高MAC地址的STA发起的4次握手的成对密钥。

加入IBSS的STA需要采用IBSS的安全配置，包括组密码套件，成对密码套件和AKMP（见8.4.4）。STA不应与具有不同安全配置的任何STA建立安全关联。IBSS内的各种STA的信标和探测响应帧必须反映一致的安全策略，因为信标启动在STA之间旋转。

加入IBSS的STA应支持并在Beacon帧中通告IBSS的安全配置，其中包括组密码套件，通告的成对密码套件和AKMP（见8.4.4）。STA可以使用探测请求帧来发现STA的安全策略，包括STA支持的附加单播密码套件。STA应忽略宣传不同安全策略的Beacon帧。

**8.4.10RSNA安全关联终止**

当SME接收或调用任何MLME关联，重新关联，解除关联，身份验证或解除身份验证请求或指示原语时，或者如果它认为它已经偏离另一个STA的无线电范围，它将删除一些安全关联。在ESS的情况下，非APSTA的SME将删除PTKSA和GTKSA，并且AP的SME将删除PTKSA。在IBSS的情况下，STA的SME将删除PTKSA和接收GTKSA。一旦安全关联被删除，SME就会调用MLME-DELETEKEYS.request原语来删除与删除的安全关联相关联的所有临时密钥。IEEE802.1X受控端口返回被阻止。因此，在调用MLMEDELETEKEYS.request原语之前，所有数据帧都是未经授权的。

如果STA失去密钥状态同步，则可以应用以下规则来恢复：

a）应丢弃所接收的任何受保护帧，并调用MLMEPROTECTEDFRAMEDROPPED.indication原语。

b）如果STA启用了RSNA并且已加入IBSS，则SME应执行11.3.1中描述的认证过程。

c）如果STA启用了RSNA并且已加入ESS，则SME应执行11.3.3中描述的解除认证过程。但是，如果STA已启动RSN安全关联，但尚未调用MLMESETPROTECTION.request原语，则不需要执行其他操作。

注：

1-在Supplicant上调用MLME-SETPROTECTION.request原语和在Authenticator上调用它之间存在竞争条件。在此期间，可以接收不能解密的加密MPDU;并且在没有发生重新认证的情况下丢弃MPDU。

2-因为IEEE802.11空数据MPDU不是从MAUNITDATA.request派生的，所以它不受保护。

如果所选择的AKMP在STA和相关联的AP之间失败，则STA和AP都应调用11.3.3中描述的MAC解除认证过程。

**8.5密钥和密钥分发**

**8.5.1密钥层次结构**

RSNA定义了两个关键层次结构：

a）成对密钥层次结构，以保护单播流量

b）GTK，一种由单个密钥组成的层次结构，用于保护多播和广播流量

注-使用TKIP或CCMP的成对密钥支持允许接收STA检测MAC地址欺骗和数据伪造。RSNA架构将发送和接收地址绑定到成对密钥。如果攻击者使用欺骗性TA创建MPDU，则接收器处的解封装过程将产生错误。GTK没有这个属性。

密钥层次结构的描述使用以下两个函数：

-L（Str，F，L）从Str开始，从左侧开始，使用7.1.1中的IEEE802.11位约定提取位F到F+L-1。

-PRF-n伪随机函数，产生n位输出，在8.5.1.1中定义。

在ESS中，IEEE802.1X认证者MAC地址（AA）和AP的BSSID是相同的，并且请求者的MAC地址（SPA）和STA的MAC地址相等。出于比较的目的，MAC地址被编码为6个八位字节，用于表示无符号二进制数。MAC地址的第一个八位字节应用作最重要的八位字节。7.1.1中的位编号约定应在每个八位字节内使用。

使用CCMP的RSNASTA应支持任何<TA，RA>对的至少一个成对密钥。<TA，RA>标识成对密钥，其不对应于任何WEP密钥标识符。

在混合环境中，AP可以同时使用具有共享WEP密钥的WEP与使用CCMP或使用成对密钥的TKIP的STA进行通信。运行WEP的STA使用默认密钥0-3表示共享WEP密钥;这里重要的一点是WEP仍然可以使用WEP默认密钥0.可以将AP配置为使用WEP默认密钥0中的WEP密钥进行WEP;如果以这种方式配置AP，则不能与TKIP成对密钥同时支持WEP缺省密钥0的STA应在RSNCapabilities字段中指定NoPairwise子字段。如果AP配置为使用WEP默认密钥0作为WEP密钥并且“无成对”STA关联，则AP不应在4向握手中设置安装位。换句话说，STA将不会安装成对临时密钥，而是将WEP默认密钥0用于所有流量。

注-“不成对”STA的行为仅用于支持WEP向RSNA的迁移。

混合环境中的TKIPSTA预期通过使用密钥映射密钥或映射到默认密钥0来支持单个成对密钥.AP将使用成对密钥用于AP和STA之间的单播流量。如果密钥映射密钥可用，则<RA，TA>对标识密钥;如果没有键映射键，则使用默认键0，因为消息中的键索引将为0。

不能支持TKIP密钥和WEP默认密钥0的STA同时通过在其（重新）关联请求中向AP发送的RSN信息单元中设置NoPairwise子字段来通告该缺陷。作为响应，AP将在4次握手的消息3中清除安装位以通知STA不安装成对密钥。相反，AP将WEP共享密钥作为WEP默认密钥0发送到要检测的站;然后，该密钥将与WEP一起用于在AP和STA之间发送和接收单播流量。

具有此限制的TKIPSTA可能不知道它将被强制使用WEP用于所有传输，直到它与AP关联并且被给予密钥使用。（STA无法知道AP已配置为使用WEP默认密钥0进行WEP通信。）如果这不满足STA配置的安全策略，则STA唯一的办法是解除关联并尝试使用其他AP。

TSN中的CCMPSTA应同时支持成对密钥和WEP默认密钥0。当CCMP是配置的密码之一时，STA无协商无成对子字段是无效的。

**8.5.1.1PRF**

PRF在本修正案的许多地方使用。根据其用途，可能需要输出128位，192位，256位，384位或512位。本小节定义了五个功能：

-PRF-128，输出128位

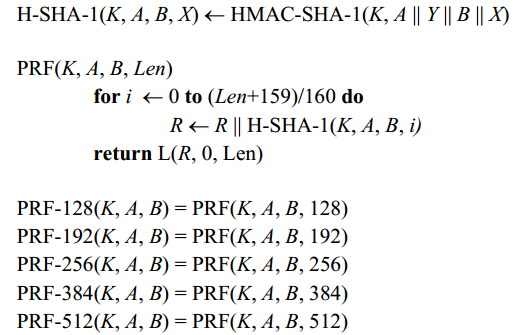
-PRF-192，输出192位

-PRF-256，输出256位

-PRF-384，输出384位

-PRF-512，输出512位

在下文中，A是PRF的每个不同目的的唯一标签;Y是一个包含0的单个八位字节;X是包含参数的单个八位字节;和||表示连接：



**8.5.1.2成对密钥层次结构**

成对密钥层次结构利用PRF-384或PRF-512从PMK导出会话特定密钥，如图43s所示。PMK应为256位。成对密钥层次结构采用PMK并生成PTK。PTK被划分为KCK和KEK，以及MAC使用的临时密钥，以保护认证者和请求者各自的STA之间的单播通信。PTK用于单个请求者和单个身份验证者之间。

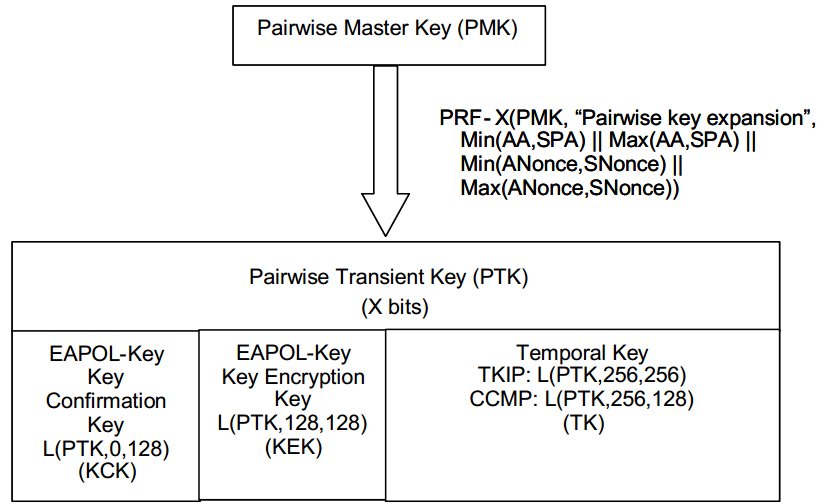


图43s-成对密钥层次结构

当不使用PSK时，PMK从AAA密钥导出。PMK应被计算为AAA密钥的前256位（位0-255）：PMK←L（PTK，0,256）。使用此派生时，AAA密钥必须至少包含256位。

PTK的使用时间不得超过PMK寿命，由AS指示的PMK寿命的最小值确定，例如Session-Timeout+dot1xAuthTxPeriod或dot11RSNAConfigPMKLifetimeMIB变量。使用RADIUS并且Session-Timeout属性不在RADIUSAccept消息中时，如果未指定密钥生存期，则PMK生存期是无限的。

**注：**

1-如果Authenticator（或AP）和AS之间的协议是RADIUS，则MS-MPPE-Recv-Key属性（vendor-id=17;参见IETFRFC2548中的第2.4.3节）可用于传输PMK到AP。

2-重新验证和更改成对键时，可能会出现竞争条件。如果在处理MLME-SETKEYS.request原语时接收到帧，则可以用一个密钥解密接收的帧，并用不同的密钥检查MIC。避免这种竞争条件的两个可能选项如下：可以根据旧的MIC键检查帧，并且可以在改变键时对接收的帧进行排队。

3-如果AKMP是RSNA-PSK，则可以在STA和AP中配置256位PSK，或者可以将通行短语配置到请求者或认证者中。用于配置PSK的方法不在此修订中，但一种方法是通过用户交互。如果配置了密码短语，则派生256位密钥并将其用作PSK。在任何RSNA-PSK方法中，PSK直接用作PMK。实现可以为每对通信STA支持不同的PSK。

在此，以下假设适用：

-SNonce是由请求者提供的随机或伪随机值;当PTK被实例化并被发送到PTKAuthenticator时，将获取其值。

-ANonce是Authenticator提供的随机或伪随机值。

-PTK应由PMK通过PTK←PRF-X（PMK，“成对键扩展”，Min（AA，SPA）||Max（AA，SPA）||Min（ANonce，SNonce）||Max（ANonce，SNonce））TKIP使用X=512并且CCMP使用X=384.IEEE802地址的最小和最大操作是将地址转换为正整数，将第一个发送的八位字节视为整数的最高有效八位字节。nonce的Min和Max操作是将nonce视为7.1.1中指定的正整数。

**注**-Authenticator和Supplicant通常每个关联只派生一次PTK。请求者或认证者可以使用4次握手来获得新的PTK。Authenticator和Supplicant都为每个4次握手实例创建一个新的nonce值。

-KCK应计算为PTK的前128位（位0-127）：KCK←L（PTK，0,128）IEEE802.1X使用KCK在4次握手中提供数据源真实性和组密钥握手消息。

-KEK应计算为PTK的位128-255：KEK←L（PTK，128,128）EAPOL-Key帧使用KEK在4次握手和组密钥握手消息中提供机密性。

-临时密钥（TK）应计算为PTK的位256-383（对于CCMP）或256-511位（对于TKIP）：TK←L（PTK，256,128）或TK←L（PTK，256），256）

EAPOL-Key状态机（见8.5.6和8.5.7）使用MLMESETKEYS.request原语将临时密钥配置到STA中。STA使用临时密钥和成对密码套件;对此值的解释是密码套件特定的。

PMK标识符定义为

PMKID=HMAC-SHA1-128（PMK，“PMK名称”||AA||SPA）

这里，HMAC-SHA1-128是其参数列表的HMAC-SHA1的前128位。

**8.5.1.3组密钥层次结构**

GTK应为随机数。

可以在配置到AP中的时间间隔重新初始化任何组主密钥（GMK），以在GMK受到损害时减少数据的暴露。

注：

1-Authenticator可能会因为多种原因更新GTK：

a）认证者可以在解除关联或取消认证STA时改变GTK。

b）STA的SME内的事件可以触发组密钥握手。

2-组密钥层次结构可以使用PRF-128（用于CCMP）或PRF-256（用于TKIP）来导出GTK。图43t描绘了组密钥层次的密钥之间的一种可能的关系。在此模型中，组密钥层次结构采用GMK并生成GTK。GTK被划分为MAC使用的临时密钥以保护广播/多播通信。GTK用于单个Authenticator和对该Authenticator进行身份验证的所有Supplicants之间。当Authenticator想要更新GTK时，它可以派生出新的GTK。

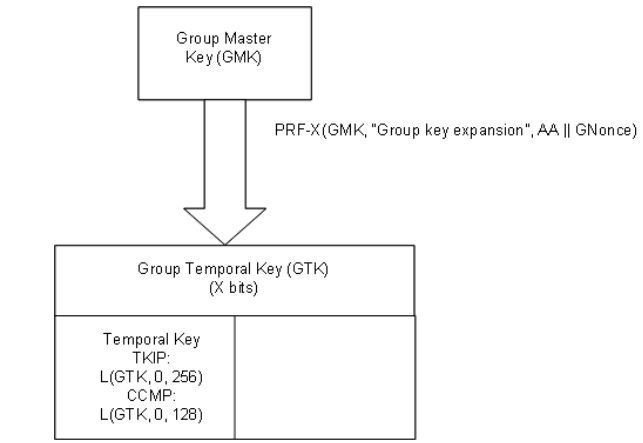


图43t-组密钥层次结构（提供信息）

在此，以下假设适用：

-组随机数（GNonce）应为IEEE802.1XAuthenticator提供的随机或伪随机值。

-GTK应来自GMK

GTK←PRF-X（GMK，“组密钥扩展”||AA||GNonce）

TKIP使用X=256，CCMP使用X=128，WEP使用X=40或X=104.AA表示为IEEE802地址，GNonce表示为7.1.1中定义的位串。

-临时密钥（TK）应为GTK的0-39位，0-103位，0-127位或0-255位：

TK←L（GTK，0,40）或

TK←L（GTK，0,104）或

TK←L（GTK，0,128）或

TK←L（GTK，0,256）

EAPOL-Key状态机（见8.5.6和8.5.7）通过MLME-SETKEYS.request原语将临时密钥配置为IEEE802.11，IEEE802.11使用该密钥。它的解释是密码套件特定的。

**8.5.2EAPOL-Key帧**

IEEE802.11使用EAPOL-Key帧在STA的请求者和认证者之间交换信息。这些交换导致加密密钥和安全关联状态的同步。EAPOL-Key框架用于实现三种不同的交换：

-4次握手，以确认相关STA之间的PMK是相同的并且存在并将GTK传送到STA。

-组密钥握手，以更新STA的GTK。

-STAKey握手，将STAKey传送到发起和对等STA。

IEEE802.11使用的RSNA密钥描述符不使用IEEE802.1X密钥描述符。相反，它使用本子条款中描述的密钥描述符。

EAPOL-Key帧中字段的位和八位字节约定在IEEEP802.1X-REV的7.1中定义。包含无效字段值的EAPOL-Key帧应以静默方式丢弃。图43u描绘了EAPOL-Key帧的格式。

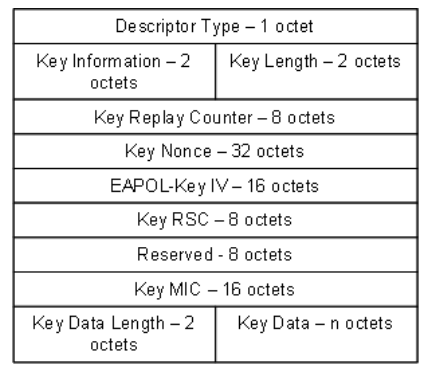


图43u-EAPOL-Key框架

EAPOL-Key框架的字段如下：

a）描述符类型。该字段是一个八位字节，具有由IEEEP802.1X-REV定义的值，用于标识IEEE802.11密钥描述符。

b）关键信息。该字段为2个八位字节，指定密钥的特征。见图43v。

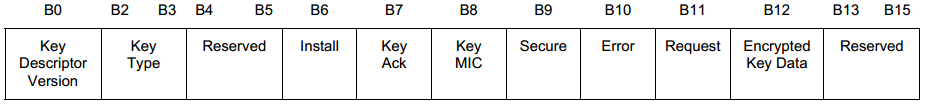


图43v-Key信息位布局

使用的位约定与IEEEP802.1X-REV的7.1中一样。KeyInformation字段的子字段如下：

1. 密钥描述符版本（位0-2）指定密钥描述符版本类型。

i）当组和成对密码都不是密钥描述符1的CCMP时，值1应用于所有进出STA的EAPOL-Key帧。该值表示以下内容：

-HMAC-MD5是EAPOL-KeyMIC。

-RC4是用于保护密钥数据字段的EAPOL-Key加密算法。

ii）当成对或组密码为密钥描述符2的AES-CCMP时，值2应用于进出STA的所有EAPOL-Key帧。该值表示以下内容：

-HMAC-SHA1-128是EAPOL-KeyMIC。HMAC在IETFRFC2104中定义;和SHA1，由FIPSPUB180-1。HMAC-SHA1的输出应截断为其128个MSB（HMAC-SHA1的摘要输出的字节0-15），即，生成的最后四个八位字节应被丢弃。

-NISTAES密钥包装是用于保护密钥数据字段的EAPOL-Key加密算法。IETFRFC3394定义了NISTAES密钥包装算法。

2）密钥类型（第3位）指定该EAPOL-Key帧是否是导出PTK的4次握手的一部分。

i）值0（组/STAKey）表示该消息不是PTK推导的一部分。

ii）值1（成对）表示该消息是PTK推导的一部分。

3）保留（位4-5）。发送方应将它们设置为0，接收方应忽略这些位的值。

4）安装（第6位）。

i）如果密钥类型（第3位）的值为1，则对于安装位，

-值1表示IEEE802.1X组件应将从该消息导出的临时密钥配置到其IEEE802.11STA中。

-值0表示IEEE802.1X组件不应将临时密钥配置到IEEE802.11STA中。

ii）如果密钥类型（第3位）的值为0，则该位在发送时应为0，在接收时应忽略。

5）如果响应此消息需要EAPOL-Key帧，则在来自Authenticator的消息中设置KeyAck（第7位），否则清除。请求者对此消息的响应应使用与此消息相同的重播计数器。

6）如果MIC在此EAPOL-Key帧中，则设置键MIC（第8位），如果此消息不包含MIC，则清除。

7）初始密钥交换完成后，设置安全（第9位）。

在请求者具有PTK和GTK之前发送的所有EAPOL-Key帧中，Authenticator应将Secure位设置为0。Authenticator应在其发送给Supplicant的所有EAPOL-Key帧中将Secure位设置为1，该Supplicant包含完成Supplicant初始化所需的最后一个密钥。

在具有PTK和GTK之前以及在从Authenticator接收到EAPOL-Key帧且安全位设置为1之前，请求方应将其在所有EAPOL-Key帧中的安全位设置为0（这应该在之前）接收4次握手的消息3）。在此之后发送的所有EAPOL密钥帧中，请求方应将Secure位设置为1，直到它丢失与Authenticator共享的安全关联为止。

8）请求方设置错误（位10）以报告在TKIPMSDU中发生MIC故障。只有在设置了请求（位11）时，请求方才应设置此位。

9）请求者（位11）由请求者设置，以请求认证者发起4次握手或组密钥握手，并由迈克尔MIC失败报告中的请求者设置。请求方不应将此位设置为正在进行的4次握手，即，在任何设置了请求位的消息中不应设置密钥确认位（位7）。身份验证者永远不会设置此位。

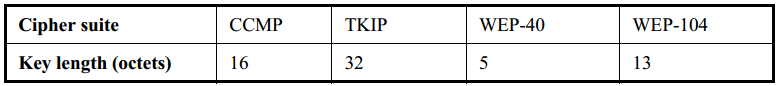
在MichaelMIC故障报告中，设置该位不是发起新握手的请求。然而，接收者可以在接收到这样的消息时发起新的握手。如果设置了请求位的EAPOL-Key帧的密钥类型为Pairwise，则Authenticator应发起4次握手。如果设置了请求位的EAPOL-Key帧的密钥类型为Group/STAKey，则Authenticator应更改GTK，与Supplicant发起4次握手，然后对所有Supplicants执行组密钥握手。

10）如果密钥数据字段已加密，则设置加密密钥数据（第12位），如果密钥数据字段未加密，则设置为加密密钥数据。如果帧中包括任何密钥材料（例如，GTK或STAKey），则应设置该子字段，并且密钥数据字段应被加密。

11）保留（位13-15）。发送方应将它们设置为0，接收方应忽略这些位的值。

c）密钥长度。该字段长度为2个八位字节，表示为无符号二进制数。该值定义配置成IEEE802.11的成对临时密钥的八位字节长度。见表20f。

表20f-Cipher套件密钥长度



d）密钥重放计数器。该字段为8个八位字节，表示为无符号二进制数，并在PMK建立时初始化为0。当响应EAPOL-Key帧时，请求者应在接收的EAPOL-Key帧中使用密钥重放计数器。它携带协议用于检测重放的EAPOL-Key帧的序列号。

请求者和身份验证者应跟踪每个安全关联的密钥重放计数器。密钥重放计数器应在（重新）关联时初始化为0。Authenticator应在每个连续的EAPOL-Key帧上递增密钥重放计数器。

当回复来自Authenticator的消息时，Supplicant将使用从Authenticator收到的最后一个有效EAPOL-Key帧的KeyReplayCounter字段值。Authenticator应使用密钥重播计数器来识别无效丢弃的无效消息。请求者还应使用密钥重放计数器，并忽略EAPOL-Key帧，其密钥重放计数器字段值小于或等于在有效消息中接收的任何值。在检查EAPOL-KeyMIC并且有效之前，不应更新本地密钥重放计数器字段。换句话说，Supplicant从不更新4次握手中消息1的KeyReplayCounter字段，因为它不包含MIC。这意味着在检查消息3的密钥重放计数器时，请求者必须允许重传消息1。

请求者应保持一个单独的密钥重播计数器，用于向认证者发送EAPOL密钥请求帧;Authenticator还应强制单独重播计数器的单调性，以过滤接收到的EAPOL-Key请求帧。

注-密钥重放计数器在4次握手中的性能优化之外不起任何作用。特别地，通过选择要包含在PTK中的从未使用过的nonce值来提供重放保护。但是，它确实在组密钥握手中发挥了有用的作用。

e）KeyNonce。该字段为32个八位字节。它传递来自Authenticator的ANonce和来自Supplicant的SNonce。如果不需要发送nonce，它可以包含0。

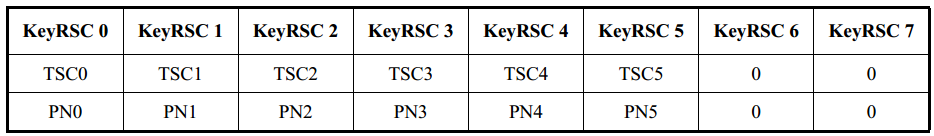
f）EAPOL-KeyIV。该字段为16个八位字节。它包含与KEK一起使用的IV。当不需要IV时，它应包含0。它应该通过获取全局密钥计数器的当前值（见8.5.7）然后递增计数器来初始化。请注意，仅使用计数器值的低16个八位字节。

g）关键RSC。该字段长度为8个八位字节。它包含安装在IEEE802.11中的GTK的接收序列计数器（RSC）。它用于4次握手的消息3和组密钥握手的消息1，其中它用于同步IEEE802.11重放状态。它也可以用在迈克尔MIC故障报告帧中，以报告经历MIC故障的帧的TSC字段值。其他消息中应包含0。KeyRSC字段给出GTK的当前消息号，以允许STA识别重放的MPDU。如果密钥RSC字段值的长度小于8个八位字节，则剩余的八位字节应设置为0.TSC或PN的最低有效八位字节应位于密钥RSC字段的第一个八位字节中。

注-TKIP的密钥RSC字段值是前6个八位字节中的TSC，而CCMP是前6个字节中的PN。见表20g。

对于WEP，密钥RSC值在发送时应设置为0，不得在接收器上使用。

表20g-KeyRSC字段



h）关键MIC。当密钥描述符版本子字段为1或2时，该字段长度为16个八位字节.EAPOL-KeyMIC是EAPOL-Key帧的MIC，包括密钥描述符版本字段（密钥信息字段），包括并且包括密钥数据字段，在密钥MIC字段设置为0的情况下计算。如果设置了加密密钥数据子字段（密钥信息字段），则在计算MIC之前加密密钥数据字段。

1）密钥描述符版本1：HMAC-MD5;IETFRFC2104和IETFRFC1321一起定义了这个功能。

2）密钥描述符版本2：HMAC-SHA1-128。

i）关键数据长度。该字段长度为2个八位字节，用于表示无符号二进制数。这表示以八位字节为单位的密钥数据字段的长度。如果设置了加密密钥数据子字段（密钥信息字段），则长度是加密后密钥数据字段的长度，包括任何填充。

j）关键数据。该字段是可变长度字段，用于包括密钥交换所需的任何其他数据，这些数据未包含在EAPOL-Key帧的固定字段中。附加数据可以是零个或多个信息元素（例如RSN信息元素）和零个或多个密钥数据封装（KDE）（例如GTK（s），STAKey（s）或PMKID（S））。在密钥数据字段中发送的信息元素包括元素ID和长度子字段。应使用图43w中的格式封装KDE。

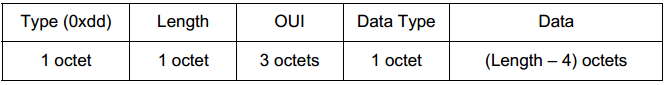
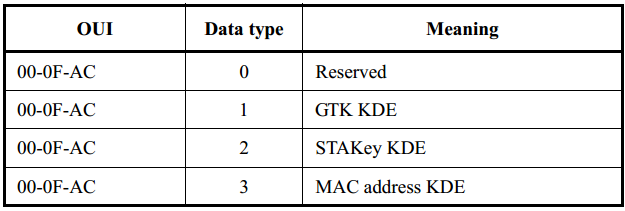


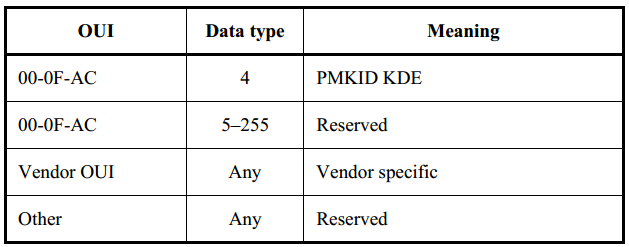
图43w-KDE格式

Type字段应设置为0xdd。Length字段指定OUI，DataType和Data字段中的八位字节数。OUI字段的顺序应遵循7.1.1中MAC地址的排序约定。

表20h列出了本修订定义的KDE选择器。

表20h-KDE





STA应忽略他们不理解的任何信息元素和KDE。

如果设置了加密密钥数据子字段（密钥信息字段），则应加密整个密钥数据字段。如果密钥数据字段使用NISTAES密钥包装，那么密钥数据字段应在加密之前填充，如果密钥数据长度小于16个八位字节或如果它不是8的倍数。填充包括附加单个八位字节0xdd后跟零或更多0x00个八位字节。处理收到的EAPOL-Key消息时，接收方应忽略此尾随填充。应接受加密但不包含GroupKey或STAKeyKDE的密钥数据字段。

如果KeyKey或STAKeyKDE包含在KeyData字段中，但KeyData字段未加密，则应忽略EAPOL-Key帧。

GTKKDE的格式如图43x所示。

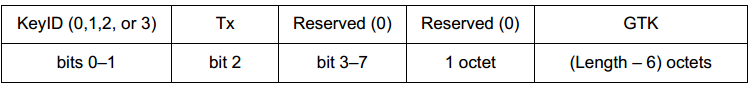


图43x-GTKKDE格式

如果Tx字段的值为1，则IEEE802.1X组件应将从该KDE导出的临时密钥配置到其IEEE802.11STA中以用于发送和接收。

如果Tx字段的值为0，则IEEE802.1X组件应将从该KDE导出的临时密钥配置到其IEEE802.11STA中以仅用于接收。

STAKey和对等MAC地址KDE的格式如图43y所示。

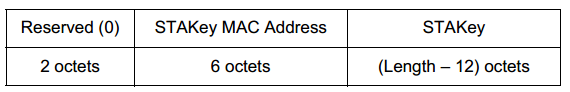


图43y-STAKeyKDE格式

MAC地址KDE的格式如图43z所示。



图43z-MAC地址KDE格式



图43z-MAC地址KDE格式

PMKIDKDE的格式如图43aa所示。

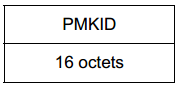


图43aa-PMKIDKDE格式

以下EAPOL-Key框架用于实现三种不同的交换：

-4次握手消息1是EAPOL-Key帧，其密钥类型子字段设置为1.密钥数据字段应包含用于此密钥派生的PMK的封装PMKID，无需加密。

-4次握手消息2是EAPOL-Key帧，KeyType子字段设置为1.KeyData字段应包含RSN信息元素，不需要加密。ESS请求方的SME应在其（重新）关联请求帧中插入它发送的RSN信息单元。RSN信息单元被包括在管理帧中发送。在接收到消息2时，认证者的SME将针对在IEEE802.11（重新）关联请求中接收的RSN信息元素来验证所选择的安全配置。

IBSS请求者的SME应插入一个RSN信息元素，其中包含它想要协商的成对密码套件选择。Authenticator的SME应验证所选的成对密码套件是其配置的密码套件之一，并且组密码套件和AKM是一致的。

-4次握手Message3是EAPOL-Key帧，KeyType子字段设置为1.KeyData字段应包含一个或两个RSN信息元素。如果已经协商了组密码，则该字段还应包括封装的GTK。如果包含GTK，则该字段应加密。

认证者的SME应插入其在信标或探测响应帧中发送的RSN信息元素。请求方的SME将针对在消息3中接收的RSN信息元素来验证所选择的安全性配置。如果存在第二个可选的RSN信息元素，则STA应该使用该密码套件及其成对密钥或取消认证。在任何一种情况下，如果值不匹配，则接收方应考虑修改的RSN信息元素，并应使用MLMEDEAUTHENTICATE.request原语来破坏关联。此时应记录安全错误。

例如，可能发生STA的请求者选择由AP通告的成对密码套件，但该策略不允许该特定STA。因此，认证者可以插入第二RSN信息元素以否决STA的选择。认证者的SME必须在第一个RSN信息元素之后插入第二个RSN信息元素，仅用于此目的。包括的第二个RSN信息单元中的成对密码套件应该是Authenticator通告的密码之一。第二RSN信息单元中的所有其他字段应与第一RSN信息单元相同。

应包含封装的GTK，并且GTK的未加密长度比八位字节中GTKKDE的长度小6。整个密钥数据字段应按密钥描述符版本的指定进行加密。

-4次握手Message4是EAPOL-Key帧，KeyType子字段设置为1.KeyData字段可以为空。

-组密钥握手消息1是EAPOL-Key帧，密钥类型子字段设置为0.密钥数据字段应包含GTKKDE并且应加密。

-组密钥握手消息2是EAPOL-Key帧，密钥类型子字段设置为0.密钥数据字段可以为空。

-STAKey握手消息1是EAPOL-Key帧，KeyType子字段设置为0.KeyData字段应包含STAKeyKDE并且应加密。STAKey用于保护直接在与同一AP关联的两个STA之间发送的单播流量。STAKey应与GTK以加密方式分开。

-STAKey握手消息2是EAPOL-Key帧，其KeyType子字段设置为0.KeyData字段应包含MAC地址KDE。

选择的密钥包装算法取决于密钥描述符版本：

-密钥描述符版本1：RC4用于使用派生PTK中的KEK字段加密密钥数据字段。不得使用衬垫。加密密钥是通过连接EAPOL-KeyIV字段和KEK生成的。在使用KEK进行RC4流密码初始化之后，应丢弃RC4密钥流的前256个八位字节，并使用第257个密钥流八位字节开始加密。

-密钥描述符版本2：在IETFRFC3394中定义的AES密钥包装应使用来自派生PTK的KEK字段来加密密钥数据字段。应使用密钥包装默认初始值。

注-AES密钥包装算法的密文输出比明文输入长8个八位字节。

**8.5.2.1STA-to-STA链路安全的STAKey握手**

STA-to-STA密钥用于将数据帧直接保护到另一STA，同时与AP相关联。AP必须与每个STA建立RSNA。在STA建立STA到STA连接之后，AP将STAKey握手消息1发送到每个STA，提供用于保护连接的密钥。该STAKey用于在两个STA之间创建STAKeySA。

始发STA通过向AP发送EAPOL-Key帧来请求STAKey，其中KeyType设置为0，请求比特设置为1，并且密钥数据字段中具有MAC地址KDE。与STAKey一起使用的密码应为EAPOL-Key帧中密钥描述符版本子字段中指示的密码：版本1表示TKIP，版本2表示CCMP。

STAKeyEAPOL-Key交换提供了用于获得用于基础设施BSS中的STA-STA直接通信的密钥的机制。STA应在传输任何直接STA到STA数据帧之前使用此交换。当STA到STA连接终止时，应删除STAKeys。图43ab描绘了配置STAKey所需的事件序列。

**8.5.2.2EAPOL-Key帧表示法**

在8.5的剩余部分中使用以下表示法来表示EAPOL-Key帧：

EAPOL-Key(S,M,A,I,K,KeyRSC,ANonce/SNonce,MIC,RSNIE,GTK[N])

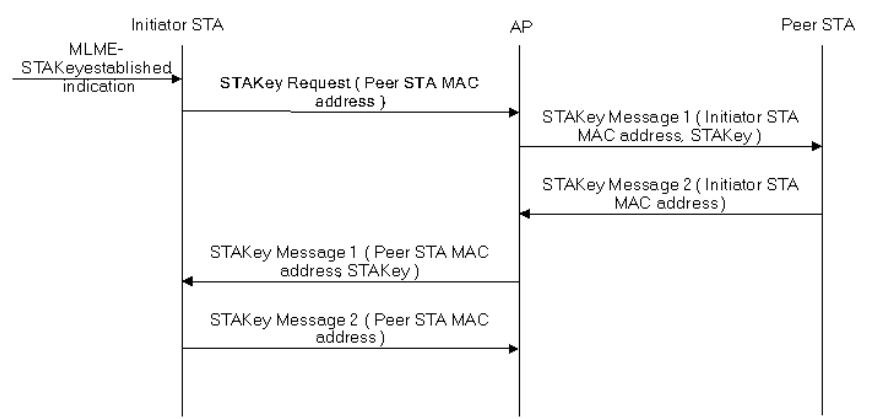


图43ab-STAKey消息交换

S 表示初始密钥交换完成。这是密钥信息字段的安全位。

M 表示MIC在消息中可用。这应该在除4次握手的消息1之外的所有消息中设置。这是密钥信息字段的关键MIC位。

A 表示此消息需要响应。当接收方应响应此消息时使用此方法。这是KeyInformation字段的KeyAck位。

I 是安装位：成对密钥的安装/不安装。这是“密钥信息”字段的“安装”位。

K 是键类型：P（成对），G（组/STAKey）。这是密钥信息字段的密钥类型位。

KeyRSC 是关键的RSC。这是KeyRSC字段。

ANonce/SNonce是Authenticator/Supplicantnonce。这是KeyNonce字段。

MIC 是完整性检查，使用KCK生成。这是关键MIC领域。

RSNIE 是RSN信息元素。这是在关键数据字段中。

GTK 是封装的GTK。这是在关键数据字段中。

N 是密钥标识符，它指定应该为此GTK使用哪个索引。除了混合环境之外，索引0不应用于GTK，如8.5.1中所述。

**8.5.3四次握手**

RSNA定义了一种使用IEEE802.1XEAPOL-Key帧的协议，称为4次握手。握手完成了IEEE802.1X身份验证过程。4次握手的信息流如下：

消息1.认证者→请求者：EAPOL-Key（0,0,1,0，P，0，ANonce，0,0,0）

消息2.请求者→认证者：EAPOL-Key（0,1,0,0，P，0，SNonce，MIC，RSNIE，0）

消息3.认证者→请求者：EAPOL密钥（1,1,1,1，P，KeyRSC，一次性数，MIC，RSNIE，GTK[N]）

消息4.请求者→身份验证者：EAPOL-Key（1,1,0,0，P，0,0，MIC，0,0）

在此，以下假设适用：

-EAPOL-Key（⋅）表示使用8.5.2.2中引入的符号传达指定参数列表的EAPOL-Key帧。

-ANonce是Authenticator贡献的nonce。ANonce在消息1和消息3中具有相同的值。

-SNonce是来自请求者的随机数。

-P表示成对位置位。

-使用8.5.1.2中定义的KCK，在EAPOL-Key帧的主体上计算MIC（在计算之前，KeyMIC字段首先归零）。

-RSNIE表示适当的RSN信息元素。

-GTK[N]表示封装的GTK及其密钥标识符。

注-虽然MIC计算在每个方向上相同，但KeyAckbi在每个方向上都不同。它在Authenticator的EAPOL-Key帧中设置，并在Supplicant的EAPOL-Key帧中清除。来自请求方的4次握手请求设置了请求位。Authenticator和Supplicant必须检查这些位以阻止反射攻击。消息1内容不得更新状态，特别是正在使用的密钥，直到使用消息3验证数据。

**8.5.3.14次握手消息1**

消息1对每个EAPOL-Key帧字段使用以下值：

描述符类型=N-见8.5.2

关键信息：

密钥描述符版本=1（使用HMAC-MD5进行RC4加密）或2（NISTAES密钥包装

与HMAC-SHA1-128）

键类型=1（成对）

安装=0

KeyAck=1

键MIC=0

安全=0

错误=0

请求=0

加密密钥数据=0

保留=0-此协议版本未使用

密钥长度=密码套件特定;见表20f

KeyReplayCounter=n-允许Authenticator匹配来自Supplicant的正确消息2

KeyNonce=ANonce

EAPOL-KeyIV=0

密钥RSC=0

键MIC=0

关键数据长度=22

密钥数据=在此交换期间使用的PMK的PMKID

在协商PSK认证，使用高速缓存的PMKSA或STA请求新密钥之后，认证者在成功的IEEE802.1X认证结束时将消息1发送给请求者。在接收消息1时，请求者确定之前是否已使用当前PMKSA使用密钥重放计数器字段值。如果KeyReplayCounter字段值小于或等于当前本地值，则Supplicant将丢弃该消息。否则，请求者

a）生成一个新的nonceSNonce。

b）派生PTK。

c）构造消息2。

**8.5.3.24次握手消息2**

消息2对每个EAPOL-Key帧字段使用以下值：

描述符类型=N-见8.5.2

关键信息：

密钥描述符版本=1（使用HMAC-MD5进行RC4加密）或2（NISTAES密钥包装

使用HMAC-SHA1-128）-与消息1相同

密钥类型=1（成对）-与消息1相同

安装=0

KeyAck=0

键MIC=1

安全=0-与消息1相同

错误=0-与消息1相同

请求=0-与消息1相同

加密密钥数据=0

保留=0-此协议版本未使用

密钥长度=0

KeyReplayCounter=n-让Authenticator知道这对应于哪个消息1

KeyNonce=SNonce

EAPOL-KeyIV=0

密钥RSC=0

键MIC=MIC（KCK，EAPOL）-在此EAPOL-Key帧的主体上计算的MIC

键MIC字段首先初始化为0

密钥数据长度=包含的RSN信息单元的八位字节的长度

密钥数据=包括RSN信息元素-发送STA的RSN信息元素

请求方将消息2发送给Authenticator。

在接收消息2时，Authenticator检查密钥重放计数器是否对应于未完成的消息1.如果不是，它将以静默方式丢弃该消息。否则，验证者

a）派生PTK。

b）验证消息2MIC。

1）如果计算出的MIC与EAPOL-Key帧中包含的请求者的MIC不匹配，则Authenticator将静默丢弃消息2。

2）如果MIC有效，则Authenticator检查RSN信息元素是否与（Re）AssociationRequest消息中的RSN信息元素匹配。

i）如果这些不完全相同，则Authenticator使用MLMEDEAUTHENTICATE.request原语来终止关联。

ii）如果它们按位匹配，则Authenticator构造消息3。

**8.5.3.34次握手消息3**

消息3对每个EAPOL-Key帧字段使用以下值：

描述符类型=N-见8.5.2

关键信息：

密钥描述符版本=1（使用HMAC-MD5进行RC4加密）或2（NISTAES密钥包装

使用HMAC-SHA1-128）-与消息1相同

密钥类型=1（成对）-与消息1相同

仅当AP不支持密钥映射密钥或者STA具有否时，才安装=0/1-0

成对位（在RSN功能字段中）设置，仅使用组密钥

KeyAck=1

键MIC=1

安全=1（已安装密钥）

错误=0-与消息1相同

请求=0-与消息1相同

加密密钥数据=1

保留=0-此协议版本未使用

密钥长度=密码套件特定;见表20f

密钥重播计数器=n+1

KeyNonce=ANonce-与消息1相同

EAPOL-KeyIV=0（版本2）或随机（版本1）

密钥RSC=认证者的STA将在受保护的MPDU中使用的起始序列号

GTK

键MIC=MIC（KCK，EAPOL）-在此EAPOL-Key帧的主体上计算的MIC

键MIC字段首先初始化为0

密钥数据长度=包含的RSN信息单元和GTK的八位字节的长度

密钥数据=AP的信标/探测响应帧的RSN信息元素，并且可选地，第二RSN信息元素是认证者的成对密码套件分配，并且如果已经协商了组密码，则封装的GTK和GTK的密钥标识符（见8.5.2）

Authenticator将Message3发送给Supplicant。

在接收消息3时，如果已经使用了密钥重放计数器字段值，或者消息3中的ANonce值与消息1中的ANonce值不同，则请求方默默地丢弃该消息。请求方也

a）验证RSN信息元素。

1）如果它与STA在信标或探测响应帧中接收的不同，则STA应解除关联。如果消息中提供了第二个RSN信息单元，则请求方应使用第二个RSN信息单元中指定的成对密码套件或取消认证。

2）如果RSN信息单元是正确的，则请求者前进到步骤b。

b）验证消息3MIC。

1）如果计算出的MIC与认证者包含在EAPOLKey帧中的MIC不匹配，则请求者默默地丢弃消息3。

2）否则请求者

i）更新KeyReplayCounter字段的最后看到的值。

ii）构造消息4。

iii）向验证者发送消息4。

iv）使用MLME-SETKEYS.request原语配置IEEE802.11MAC以发送和接收由PTK保护的3类单播MPDU。GTK也由MLME-SETKEYS原语配置。

**8.5.3.44次握手消息4**

消息4对每个EAPOL-Key帧字段使用以下值：

描述符类型=N-见8.5.2

关键信息：

密钥描述符版本=1（使用HMAC-MD5进行RC4加密）或2（使用HMAC-SHA1-128进行NISTAES密钥包装）-与消息1相同

密钥类型=1（成对）-与消息1相同

安装=0

KeyAck=0-这是最后一条消息

键MIC=1

安全=1

错误=0

请求=0

加密密钥数据=0

保留=0-此协议版本未使用

密钥长度=0

密钥重播计数器=n+1

KeyNonce=0

EAPOL-KeyIV=0

密钥RSC=0

键MIC=MIC（KCK，EAPOL）-在此EAPOL-Key帧的主体上计算的MIC

键MIC字段首先初始化为0

关键数据长度=0

关键数据=无需要

请求者将消息4发送给身份验证器。请注意，首次使用4次握手时，将以明文形式发送消息4。

在接收消息4时，Authenticator验证密钥重放计数器字段值是否是它在此4次握手上使用的值。如果不是，它会默默地丢弃该消息。否则，验证者

a）检查MIC。

1）如果计算出的MIC与EAPOL-Key帧中包含的请求者的MIC不匹配，则Authenticator将静默丢弃消息4。

2）如果MIC有效，则Authenticator使用MLME-SETKEYS.request原语将PTK配置为IEEE802.11MAC。

b）更新KeyReplayCounter字段，以便在需要重定密钥时使用新值。

**8.5.3.54次握手实现注意事项**

如果Authenticator没有收到对其消息的回复，它将尝试dottRSNAConfigPairwiseUpdateCount发送消息，加上最终超时。第一次超时的重传超时值应为100ms，第二次超时的重传超时值应为监听间隔的一半，以及后续超时的监听间隔。如果没有监听间隔，则所有超时值都应使用100ms。如果在重试后仍未收到响应，则Authenticator应取消认证STA。

如果STA在预期的时间间隔内（在IEEE802.1X超时之前）没有接收到消息1，则它应该解除关联，取消认证并尝试另一个AP/STA。

Authenticator应该忽略EAPOL-Key帧，它不会期望回复它已发送的消息或者设置了Ack位的EAPOL-Key帧。这可以阻止攻击者将第一条消息发送给响应身份验证者的请求者。

实施应该将KCK和KEK保存在4次握手之外，因为组密钥握手需要它们以及从TKIPMIC故障中恢复。

在将消息4发送到Authenticator之后，Supplicant使用MLME-SETKEYS.request原语将时间密钥从8.5.1配置到其STA中。

注：

1-如果RSN信息元素检查消息2或消息3失败，IEEE802.1X应记录错误并取消验证对等方。

2-Supplicant应检查如果RSN信息单元指定了成对密码套件，则4次握手确实指定将PTK的临时密钥部分配置到IEEE802.11STA中。

**8.5.3.6样本四次握手（资料性）**

在AP发送EAP-Success完成IEEE802.1X认证后，AP发起4次握手。见图43ac。

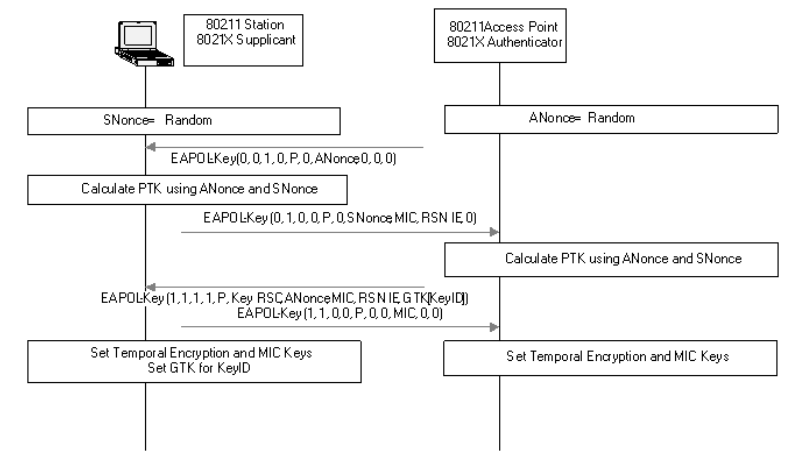


图43ac-Sample4次握手

4次握手包括以下步骤：

a）Authenticator发送包含ANonce的EAPOL-Key帧。

b）请求者从ANonce和SNonce导出PTK。

c）请求者发送包含SNonce的EAPOL-Key帧，来自（Re）关联请求帧的RSN信息元素和MIC。

d）Authenticator从ANonce和SNonce派生PTK并验证EAPOLKey帧中的MIC。

e）Authenticator发送包含ANonce的EAPOL-Key帧，来自其Beacon或ProbeResponse消息的RSN信息元素，MIC，是否安装临时密钥，以及封装的GTK。

f）请求方发送EAPOL-Key帧以确认已安装临时密钥。

**8.5.3.74次握手分析（资料性）**

本子条款明确规定了本协议中使用的信任假设。该协议假定以下内容：

-PMK仅由请求方的STA和认证方的STA知道。

-请求者的STA使用IEEE802地址SPA。

-Authenticator的STA使用IEEE802地址AA。

在许多实例中，RSNA架构立即打破了第一个假设，因为IEEE802.1XAS也知道PMK。因此，需要额外的假设：

-AS不会将PMK暴露给其他方。

-AS不会伪装成认证者的请求者。

-AS不会伪装成请求者的身份验证者。

-AS不伪装成请求者的STA。

-AS不伪装成Authenticator的STA。

该协议还假设这个特定的Supplicant-Authenticator对被授权知道这个PMK并在4次握手中使用它。如果这些假设中的任何一个被破坏，则协议无法提供任何安全保证。

该协议还假设AS向具有IEEE802地址AA的AP递送正确的PMK，并且具有IEEE802地址SPA的非APSTA承载与AS协商PMK的请求者。IEEEStd802.11,1999Edition和IEEEP802.1X-REV定义的协议都不允许AS，认证者，请求者或STA验证这些假设。

PTK推导步骤：

PTK←PRF-X(PMK,“Pairwisekeyexpansion”||Min(AA,SPA)||Max(AA,SPA)||  
Min(ANonce,SNonce)||Max(ANonce,SNonce))

执行许多功能：

-在计算中包括AA和SPA

-将PTK绑定到通信的STA和

-防止在具有这两个IEEE802地址的STA之间针对4次握手消息的未检测到的中间人攻击。

-如果随机选择ANonce，包括ANonce

-保证STA在IEEE802地址AA处PTK是新鲜的，

-保证消息2和消息4是活动的，并且

-将PTK唯一标识为<AA，ANonce>。

-如果随机选择SNonce，包括SNonce

-保证STA在IEEE802地址SPA中PTK是新鲜的，

-保证Message3是实时的，并且

-将PTK唯一标识为<SPA，SNonce>。

随机选择随机数有助于防止预计算攻击。对于不可预测的nonce，使用Supplicant预先计算消息以攻击Authenticator的man-inhe-middle攻击无法超越Message2，对Supplicant的类似攻击无法超越Message3.协议可以在错误发生之前执行如果使用可预测的随机数。

消息1将ANonce传递给请求者并启动新PTK的协商。它将AA识别为请求者STA的对等STA。如果攻击者修改了地址或ANonce中的任何一个，则Authenticator将在验证消息2中的MIC时检测结果。消息1不携带MIC，因为请求者无法在不保持状态的情况下将此消息与重放区分开来所有安全关联的所有时间（PMK可能是一个静态密钥）。

消息2将SNonce传递给Authenticator，以便它可以导出PTK。如果Authenticator随机选择了ANonce，则Message2还向Authenticator证明Supplicant是活的，PTK是新鲜的，并且没有中间人攻击，因为PTK包含IEEE802MAC地址都。在PTK推导中包含ANonce也可以防止重放。MIC可防止未检测到的消息2内容的修改。

消息3向请求者确认没有中间人攻击。如果Supplicant随机选择了SNonce，它还表明PTK是新鲜的并且Authenticator是实时的。MIC再次阻止对消息3的未检测到的修改。

虽然消息4不用于加密目的，但它用作对消息3的确认。它需要确保可靠性并通知Authenticator，请求方已经安装了PTK和GTK，因此可以接收加密帧。

发送消息4后，使用MLME.SETKEYS.request原语安装PTK和GTK。PTK安装在GTK之前。

然后，4次握手使用正确但不寻常的机制来防止重放。如本子节前面所述，ANonce为Authenticator提供重播保护，为请求者提供SNonce。在大多数会话发起协议中，通过随机选择随机数并且要求对等体在响应消息中反映所接收的随机数来明确地完成重放保护。而4次握手则将ANonce和SNonce混合到PTK中，并且通过MIC故障隐式地检测重放。特别是，密钥重放计数器字段在4次握手中不起加密作用。然而，它的存在并不是有害的，并且它作为处理消息2的陈旧实例的次要性能优化起到了有用的作用。这种重放机制是正确的，但其隐含性使得协议比明确的方法更难理解。

对于4次握手的正确性至关重要的是每条消息中至少有一位不同。在4次握手中，消息1可以被识别为唯一一个键MIC位清除的消息，这意味着消息1不包括MIC，而消息2通过消息4。消息3与消息2的不同之处在于不通过断言Ack位来断言Ack位和消息4。消息2通过包括RSN信息元素而与消息4不同。

请求消息不能与4次握手消息混淆，因为前者断言请求位，而4次握手消息则不然。组密钥握手消息不能被误认为4次握手消息，因为它们断言不同的密钥类型。

**8.5.4组密钥握手**TheAuthenticatorusesthe组密钥握手tosendanewGTKtotheSupplicant.

当请求者被解除关联或取消身份验证时，身份验证者可以启动交换。

消息1：认证者→请求者：EAPOL-Key（1,1,1,0，G，KeyRSC，0，MIC，0，GTK[N]）

消息2：请求者→认证者：EAPOL-Key（1,1,0,0，G，0,0，MIC，0,0）

在此，以下假设适用：

-密钥RSC表示使用GTK发送的最后帧序列号。

-GTK[N]表示使用8.5.1.2中定义的KEK和相关的IV在8.5.2中定义的密钥标识符封装的GTK。

-使用8.5.1.2中定义的KCK，在EAPOL-Key帧的主体上计算MIC（MIC字段用于计算）。

请求方可以通过发送请求位设置为1的EAPOL-Key帧和组密钥位的类型来触发组密钥握手。

如果两者都需要完成，则认证者应在组密钥握手之前进行4次握手。

注-在4次握手成功完成之前，Authenticator无法发起组密钥握手。

**8.5.4.1组密钥握手消息1**

消息1对每个EAPOL-Key帧字段使用以下值：

描述符类型=N-见8.5.2

关键信息：

密钥描述符版本=1（使用HMAC-MD5进行RC4加密）或2（NISTAES密钥包装

与HMAC-SHA1-128）

键类型=0（组/STAKey）

安装=0

KeyAck=1

键MIC=1

安全=1

错误=0

请求=0

加密密钥数据=1

保留=0

密钥长度=0

密钥重播计数器=n+2

KeyNonce=0

EAPOL-KeyIV=0（版本2）或随机（版本1）

密钥RSC=GTK的最后发送序列号

关键MIC=MIC（KCK，EAPOL）

关键数据长度=密码套件特定;见表20f

密钥数据=加密，封装的GTK和GTK的密钥标识符（见8.5.2）

Authenticator将消息1发送给请求者。

收到消息1，请求者

a）验证之前还没有看到密钥重放计数器字段值，即，其值严格地大于在该会话期间到目前为止接收的任何其他EAPOL-Key帧中的值。

b）验证MIC是否有效，即它使用作为PTK一部分的KCK来验证没有数据完整性错误。

c）使用MLME-SETKEYS.request原语将时间GTK配置为其IEEE802.11MAC。

d）通过创建组密钥握手的消息2并将其发送到认证器并递增重放计数器来响应。

注-Authenticator必须在每个Message1实例上递增并使用新的KeyReplayCounter字段值，甚至重试，因为响应早期Message1的Message2可能已丢失。如果Authenticator没有递增重播计数器，则Supplicant将丢弃重试，并且没有响应的消息2将到达。

**8.5.4.2组密钥握手消息2**

消息2对每个EAPOL-Key帧字段使用以下值：

描述符类型=N-见8.5.2

关键信息：

密钥描述符版本=1（使用HMAC-MD5进行RC4加密）或2（NISTAES密钥包装

使用HMAC-SHA1-128）-与消息1相同

密钥类型=0（组/STAKey）-与消息1相同

安装=0

KeyAck=0

键MIC=1

安全=1

错误=0

请求=0

加密密钥数据=0

保留=0

密钥长度=0

密钥重播计数器=n+2-与消息1相同

KeyNonce=0

EAPOL-KeyIV=0

密钥RSC=0

关键MIC=MIC（KCK，EAPOL）

关键数据长度=0

关键数据=无需要

在接收消息2时，验证者

a）验证KeyReplayCounter字段值是否与它在组密钥握手中使用的值匹配。

b）验证MIC是否有效，即它使用作为PTK一部分的KCK来验证没有数据完整性错误。

**8.5.4.3组密钥握手实现注意事项**

如果Authenticator没有收到对其消息的回复，它将尝试dot11RSNAConfigGroupUpdateCount发送消息，加上最终超时。第一次超时的重传超时值应为100ms，第二次超时的重传超时值应为监听间隔的一半，以及后续超时的监听间隔。如果没有监听间隔，则所有超时值都应使用100ms。如果在此之后仍未收到响应，则Authenticator的STA应使用MLME-DEAUTHENTICATE.request原语来取消认证STA。

**8.5.4.4样本组密钥握手（提供信息）**

8.5.6和8.5.7中的状态机改变了网络使用的GTK。见图43ad。

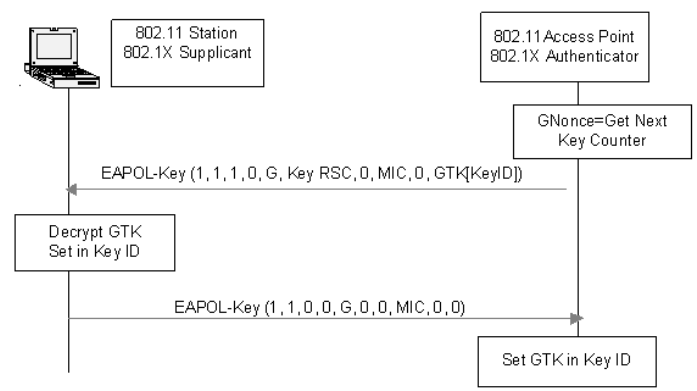


图43ad-Sample组密钥握手

发生以下步骤：

a）Authenticator生成一个新的GTK。它封装GTK并发送包含GTK（消息1）的EAPOL-Key帧，以及与GTK（RSC）一起使用的最后一个序列号。

b）在接收EAPOL-Key帧时，Supplicant验证MIC，解封GTK，并使用MLME-SETKEYS.request原语在其STA中配置GTK和RSC。

c）然后，请求者构造并向认证者发送EAPOL-Key帧以确认。

d）在接收EAPOL-Key帧后，Authenticator验证MIC。如果GTK尚未配置为IEEE802.11MAC，则在Authenticator将GTK传送到所有关联的STA之后，它使用MLMESETKEYS.request原语将GTK配置为IEEE802.11STA。

**8.5.5STAKey握手**

STA可以请求AP在其自身与与同一AP相关联的另一STA之间建立STAKeySA。与4次握手和组密钥握手不同，STAKey握手由STA发起。因此，STAKey请求消息由发起STA的EAPOL-Key请求重放计数器保护。这是一个单调递增的数字。AP为每个STA维护一个单独的请求重播计数器以强制执行重播保护。

STAKeyRequest:InitiatingSTA→Authenticator:  
EAPOL-Key(1,1,0,0,G/0,0,0,MIC,0,PeerMACKDE)  
Message1:Authenticator→PeerSTA:  
EAPOL-Key(1,1,1,0,G/0,0,0,MIC,0,InitiatorMACKDE,STAKey)  
Message2:PeerSTA→Authenticator:  
EAPOL-Key(1,1,0,0,G/0,0,0,MIC,0,InitiatorMACKDE,STAKey)  
Message1:Authenticator→InitiatingSTA:  
EAPOL-Key(1,1,1,0,G/0,0,0,MIC,0,PeerMACKDE,STAKey)  
Message2:InitiatingSTA→Authenticator:  
EAPOL-Key(1,1,0,0,G/0,0,0,MIC,0,PeerMACKDE,STAKey)

**8.5.5.1STAKey请求消息**

STAKey请求消息对每个EAPOLKey帧字段使用以下值：

描述符类型=N-见8.5.2

关键信息：

密钥描述符版本=1（使用HMAC-MD5进行RC4加密）或2（NISTAES密钥包装

与HMAC-SHA1-128）

键类型=0（组/STAKey）

安装=0

KeyAck=0

键MIC=1

安全=1

错误=0

请求=1

加密密钥数据=0

保留=0

密钥长度=0

密钥重放计数器=发起STA的请求重播计数器

KeyNonce=0

EAPOL-KeyIV=0

密钥RSC=0

关键MIC=MIC（启动STA的KCK，EAPOL）

密钥数据长度=八位字节中密钥数据字段的长度

关键数据=对等MAC地址KDE（见图43z）

STA利用对等STA的MAC地址向AP发送受保护的STAKey请求消息。在接收到STAKey请求消息时，AP验证所接收的密钥重放计数器字段值是否等于或大于其计数器的本地副本。然后，它验证MIC是否有效，并将发起STA的本地密钥重放计数器字段值设置为接收的值。

**8.5.5.2STAKey消息1**

STAKey消息1对每个EAPOL-Key帧字段使用以下值：

描述符类型=N-见8.5.2

关键信息：

密钥描述符版本=1（使用HMAC-MD5进行RC4加密）或2（NISTAES密钥包装

与HMAC-SHA1-128）

键类型=0（组/STAKey）

安装=1

KeyAck=1

键MIC=1

安全=1

错误=0

请求=0

加密密钥数据=1

保留=0

密钥长度=密码套件特定;见表20f

密钥重放计数器=n+3（假设这遵循对等STA之间的组密钥握手

和AP）

KeyNonce=0

EAPOL-KeyIV=0（版本2）或随机（版本1）

密钥RSC=0

关键MIC=MIC（对等STA的KCK，EAPOL）

密钥数据长度=八位字节中密钥数据字段的长度

密钥数据=加密的发起者MAC地址KDE和STAKey（见图43y）

响应于STAKey请求消息，AP将STAKey消息1发送到由请求消息标识的对等STA。在接收到STAKey消息1时，STA验证在当前PMKSA的上下文中之前从未见过所接收的密钥重放计数器字段值。然后，它验证MIC并从STAKey消息1中的接收值设置本地密钥重放计数器字段值。接下来，对等STA配置STAKey以与发起STA直接通信。最后，它将STAKey消息2发送给AP。

**8.5.5.3STAKey消息2**

STAKey消息2对每个EAPOL-Key帧字段使用以下值：

描述符类型=N-见8.5.2

关键信息：

密钥描述符版本=1（使用HMAC-MD5进行RC4加密）或2（NISTAES密钥包装

与HMAC-SHA1-128）

键类型=0（组/STAKey）

安装=0

KeyAck=0

键MIC=1

安全=1

错误=0

请求=0

加密密钥数据=0

保留=0

密钥长度=0

密钥重播计数器=n+3（如STAKey消息1中所示）

KeyNonce=0

EAPOL-KeyIV=0

密钥RSC=0

关键MIC=MIC（对等STA的KCK，EAPOL）

密钥数据长度=八位字节中密钥数据字段的长度

关键数据=启动器MAC地址KDE（见图43z）

STAKey消息2是对STAKey消息1的完整性和重放保护确认。在接收到STAKey消息2时，AP验证所接收的密钥重放计数器字段值是否与它在STAKey消息1中发送的值相同。然后它验证MIC。并递增对等STA的KeyReplayCounter字段值。AP最终将STAKey消息1发送给发起STA。

**8.5.5.4STAKey消息1和消息2到发起STA**

在与对等STA交换STAKey消息之后，AP发起与发起STA相同的STAKey消息1和消息2交换。这将相同的STAKey分配给发起STA，并使用与对等STA的STAKey交换中相同的消息，除了以下差异：

1）密钥重放计数器字段值，KCK和KEK对应于AP和发起STA之间的PMKSA，而不是AP和对等STA之间的PMKSA的PMKSA。

2）发起STA的MAC和对等STA的MACKDE被反转。

3）如果交换失败，则AP通知对等STA删除STAKeySA。

**8.5.6RSNA请求者密钥管理状态机**

只要系统初始化，请求方就必须重新初始化Supplicant状态机。请求者从MAC请求另一个STA进行身份验证的事件进入AUTHENTICATION状态。当从Authenticator接收到EAPOL-Key帧时，请求者进入STAKEYSTART状态。如果MIC或任何EAPOL-Key帧出现故障，请求者将静默丢弃该帧。图43ae描绘了请求者状态机。

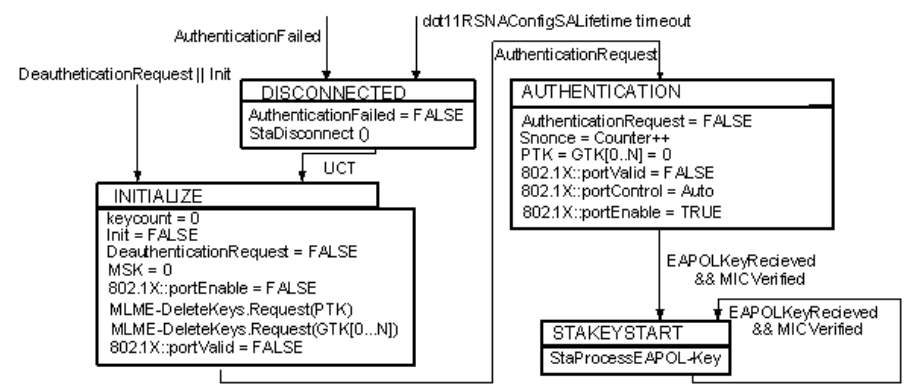


图43ae-RSNA请求者密钥管理状态机

无条件转移（UCT）意味着事件触发立即转换。

此状态机不使用超时或重试。IEEE802.1X状态机具有从身份验证失败等恢复的超时。

管理实体在需要经过身份验证的身份验证时将发送身份验证请求事件。这可以在STA关联到AP之前或之后。在IBSS环境中，将在收到探测响应帧时生成事件。

**8.5.6.1请求状态机状态**

以下列表总结了Supplicant状态机的状态：

-认证：STA的请求者在发送IEEE802.1XAuthenticationRequest以对SSID进行身份验证时进入此状态。

-DISCONNECTED：当IEEE802.1X身份验证失败时，STA的请求者进入此状态。Supplicant执行StaDisconnect并进入INITIALIZE状态。

-INITIALIZE：STA的请求者从DISCONNECTED状态进入此状态，当它收到Disassociation或Deauthentication消息或STA初始化时，导致STA的Supplicant初始化密钥状态变量。

-STAKEYSTART：STA的请求者在收到EAPOL-Key帧时进入此状态。处理EAPOL-Key帧的所有信息都在消息中，并在StaProcessEAPOL-Key过程中进行了描述。

**8.5.6.2请求状态机变量**

以下列表总结了Supplicant状态机使用的变量：

* DeauthenticationRequest-如果请求者的STA，请求者将此变量设置为TRUE

报告已收到Disassociation或Deauthentication消息。

- AuthenticationRequest-如果其STA的IEEE802.11管理实体报告它想要对SSID进行身份验证，则Supplicant将此变量设置为TRUE。这可以在关联时或在其他时间。

- AuthenticationFailed-如果IEEE802.1X身份验证，请求方将此变量设置为TRUE

失败。请求者使用MLME-DISASSOCIATE.request原语使其STA与认证者的STA解除关联。

- EAPOLKeyReceived-Supplicant在收到EAPOL-Key时将此变量设置为TRUE

帧。

- IntegrityFailed-当STA报告致命数据时，Supplicant将此变量设置为TRUE

已发生完整性错误（例如，迈克尔失败）。

注意-迈克尔故障与MICVerified不同，因为如果迈克尔生成了IntegrityFailed

完整性检查失败;MICVerified是通过验证EAPOL-Key完整性检查生成的。还要注意

STA不为CCMP生成此事件，因为不需要对策。

- MICVerified-如果收到的EAPOL-Key上的MIC，则Supplicant将此变量设置为TRUE

框验证为正确。Supplicant默默地丢弃任何收到的EAPOL-Key帧

MIC无效。

- Counter-Supplicant使用此变量作为用于生成随机数的全局计数器。

- SNonce-此变量表示请求者的随机数。

- PTK-此变量表示当前PTK。

- TPTK-该变量表示当前的PTK，直到4次握手的消息3到达

并经过验证。

- GTK[]-此变量表示每个组密钥索引的当前GTK。

- PMK-该变量代表当前的PMK。

- keycount-此变量在IBSS模式下用于决定何时传递所有密钥

IBSS链接是安全的。

- 802.1X::XXX-此变量表示此处未指定的另一个IEEE802.1X状态变量XXX

修正。

**8.5.6.3请求状态机程序**

以下列表总结了Supplicant状态机使用的过程：

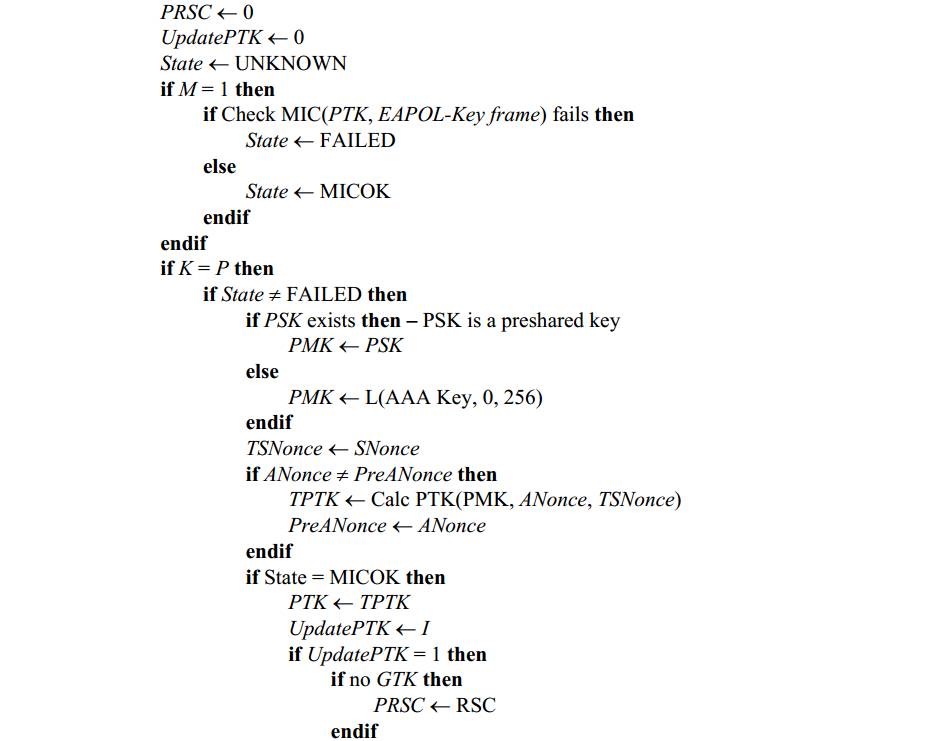
- STADisconnect-Supplicant调用此过程以取消其STA与AP的解除关联和取消验证。

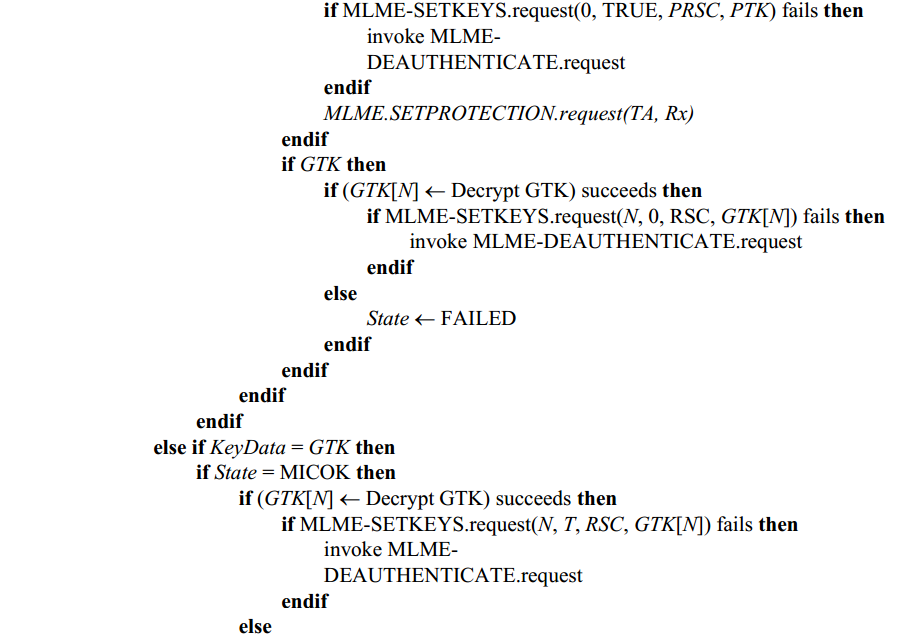
- MIC（x）-请求者调用此过程来计算数据x的MIC。

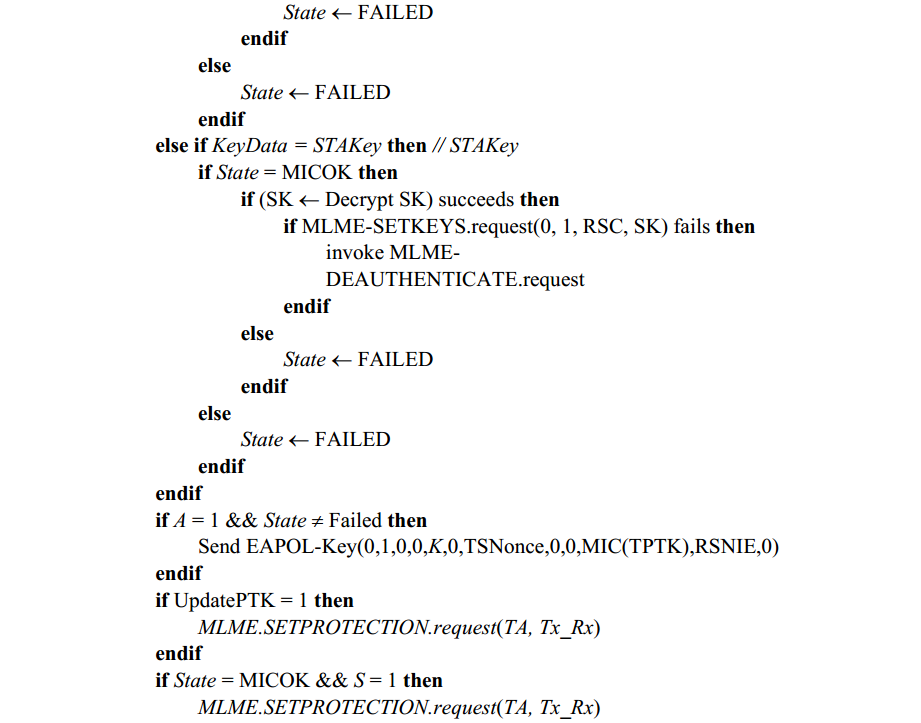
- CheckMIC（）-请求者调用此过程来验证MIC（）函数计算的MIC。

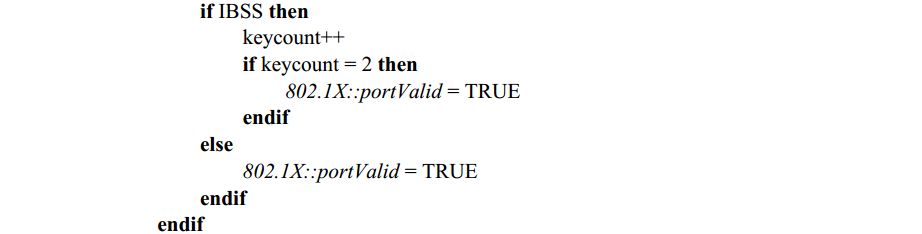
- StaProcessEAPOL-Key-Supplicant调用此过程来处理收到的EAPOLKey帧。此过程的伪代码如下：StaProcessEAPOL-Key（S，M，A，I，K，RSC，ANonce，RSC，MIC，RSNIE，GTK[N]）











这里UNKNOWN，MICOK和FAILED是Supplicant伪代码中使用的变量State的值。State用于决定如何进行密钥处理。当检查EAPOL-Key的MIC并且有效时，设置MICOK。在处理EAPOL-Key帧时发生故障时使用FAILED。UNKNOWN是变量State的初始值。

当处理4次握手Message3时，GTK从EAPOL-Key帧解密并安装。PTK应安装在GTK之前。

请求者为响应接收的EAPOL-Key帧而发送的EAPOL-Key帧使用的密钥重放计数器字段应为接收的密钥重放计数器字段。无效的EAPOL-Key帧，例如无效的MIC，没有MIC的GTK等，应被忽略。

注：

1-TPTK用于阻止攻击者通过发送4次握手的第一条消息来更改Supplicant上的PTK。在进行4次握手时，攻击者仍然可以影响4次握手。

2-如果未使用预共享模式，PMK将由IEEE802.1X使用的验证方法提供。

3-PTK根据Tx/Rx位配置到加密/完整性引擎中，但如果已配置，则始终为传输密钥。GTK配置到加密/完整性引擎中，与Tx/Rx位的状态无关，但是GTK是否用作发送密钥取决于Tx/Rx位的状态。

-CalcGTK（x）-生成GTK。

-DecryptGTK（x）-从EAPOL-Key帧解密GTK。

**8.5.7RSNAAuthenticator密钥管理状态机**

Authenticator有一个状态图。在ESS中，Authenticator将始终位于AP上;在IBSS环境中，Authenticator将在每个STA上。

图43af至图43ai中的部分所示的状态图包含以下状态：

a）AUTHENTICATION，AUTHENTICATION2，INITPMK，INITPSK，PTKSTART，PTKCALALEGOTIATING，PTKCALCNEGOTIATING2，PTKINITONEGOTIATING，PTKINITDONE，DISCONNECT，DISCONNECTED和INITIALIZE状态。这些状态处理初始化，4次握手，拆卸和一般清理。这些状态是每个关联的STA。

b）IDLE，REKEYNEGOTIATING，KEYERROR和REKEYESTABLISHED状态。这些状态处理GTK到关联客户端的传输。这些状态是每个关联的STA。

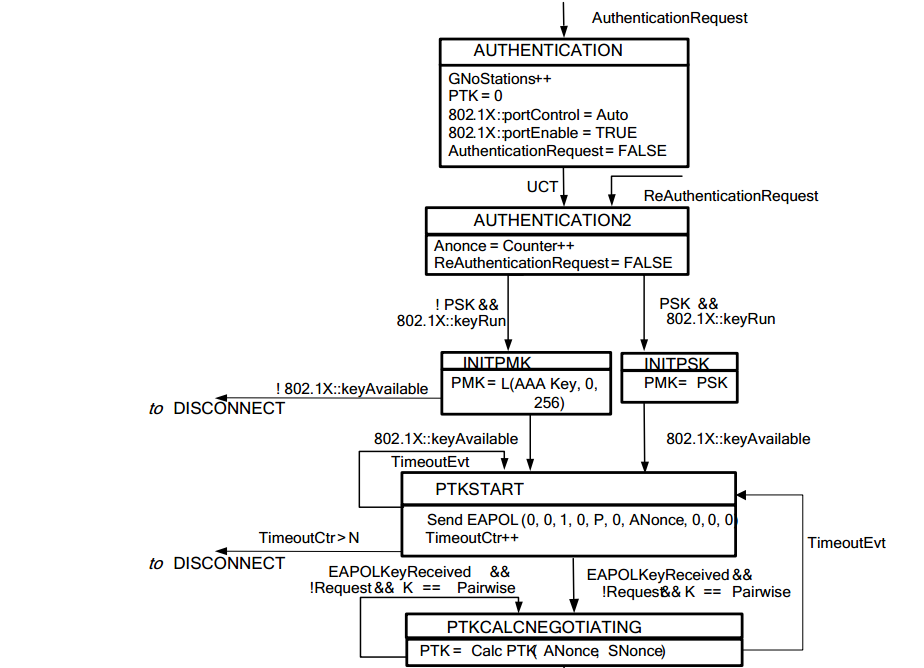
c）GTK\_INIT，SETKEYS和SETKEYSDONE状态。这些状态在需要时更改GTK，触发所有PTK组密钥状态机，并在所有STA具有更新的GTK时更新Authenticator的AP中的IEEE802.11MAC。这些状态对Authenticator来说是全球性的。

因为有两个GTK，所以更新这些密钥的责任是给组密钥状态机（见图43ah）。换句话说，该状态机随时确定正在使用哪个GTK。

当第二STA关联时，组密钥状态机已经初始化，并且GTK已经可用并且正在使用中。

当要更新GTK时，设置变量GTKReKey。SETKEYS状态更新GTK并触发当前存在的所有PTK组密钥状态机-每个关联的STA一个。每个PTK组密钥状态机将GTK发送到其STA。当所有STA都收到GTK（或未能接收到密钥）时，执行SETKEYSDONE状态，用新密钥更新AP加密/完整性引擎。

PTK状态机和PTK组密钥状态机都使用接收的EAPOL-Key帧作为改变状态的事件。PTK状态机仅使用EAPOL-Key帧，KeyType字段设置为Pairwise，PTK组密钥状态机仅使用EAPOL-Key帧，KeyType字段设置为Group/STAKey。



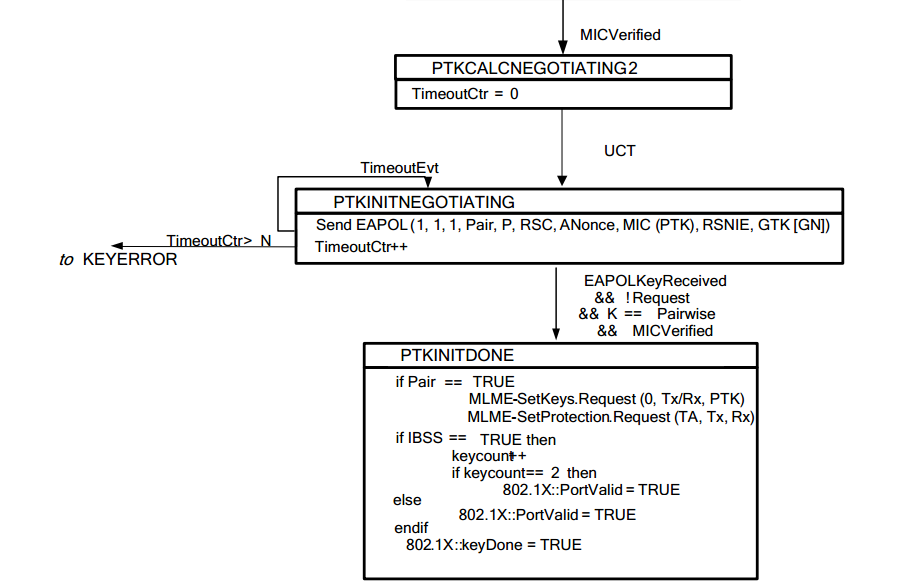


图43af-Authenticator状态机，第1部分

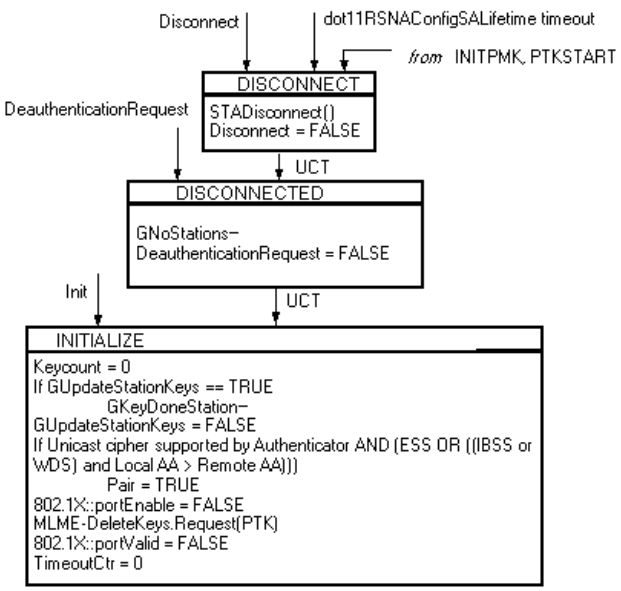


图43ag-Authenticator状态机，第2部分

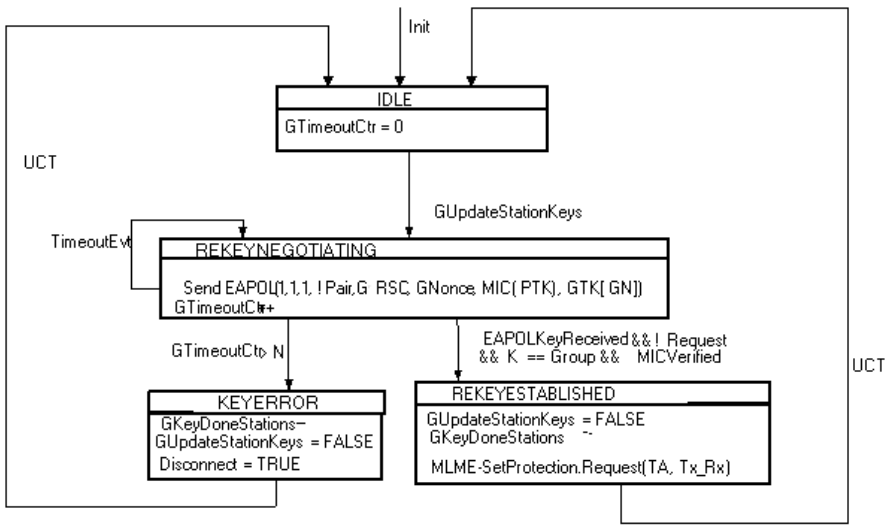


图43ah-Authenticator状态机，第3部分

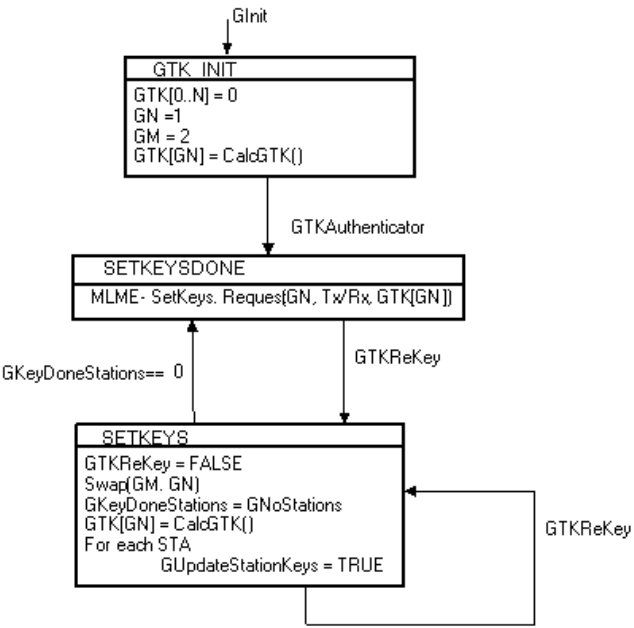


图43ai-Authenticator状态机，第4部分

**8.5.7.1认证者状态机状态**

**8.5.7.1.1认证者状态机：4次握手（每个STA）**

以下列表总结了Authenticator状态机用于支持4次握手的状态：

- AUTHENTICATION：当从管理实体发送AuthenticationRequest以验证BSSID时，进入该状态。

- AUTHENTICATION2：此状态从AUTHENTICATION状态或PTKINITDONE状态进入。

- DISCONNECT：如果收到EAPOL-Key帧并且未通过MIC检查，则进入该状态。它向STA发送Deauthentication消息并进入INITIALIZE状态。

- DISCONNECTED：收到Disassociation或Deauthentication消息时进入此状态。

- INITIALIZE：当发生解除身份验证请求事件或站初始化时，从DISCONNECTED状态进入此状态。状态初始化关键状态变量。

- INITPMK：当IEEE802.1X后端AS成功完成时，进入此状态。如果提供PMK，则进入PTKSTART状态;否则，它进入DISCONNECTED状态。

- INITPSK：配置PSK时进入此状态。

- PTKCALCNEGOTIATING：当按键类型为Pairwise接收到4次握手的第二个EAPOLKey帧时，进入此状态。

- PTKCALNEGOTIATING2：当验证4次握手的第二个EAPOL-Key帧的MIC时，进入该状态。

- PTKINITNEGOTIATING：当验证了4次握手的第二个EAPOL-Key帧的MIC时，进入此状态。当在状态PTKINITNEGOTIATING中发送4次握手的消息3时，应在数据字段的末尾发送加密的GTK，并将GTK长度放入GTK长度字段中。

- PTKINITDONE：当按键类型为Pairwise接收到4次握手的最后一个EAPOL-Key帧时，进入此状态。该状态可以调用SetPTK;如果此调用失败，则AP应检测并从该情况中恢复，例如，通过为此关联执行断开连接事件。

- PTKSTART：从INITPMK或INITPSK输入此状态以启动4次握手，或者如果没有响应4次握手。

**8.5.7.1.2认证者状态机：组密钥握手（每个STA）**

以下列表总结了Authenticator状态机用于支持组密钥握手的状态：

- IDLE：当没有发生组密钥握手时进入该状态。

- KEYERROR：如果未收到组密钥握手的EAPOL-Key确认，则进入该状态。

- REKEYESTABLISHED：当从“请求者”接收到EAPOL-Key帧并且“密钥类型”子字段设置为“组/STAKey”时，进入此状态。

- REKEYNEGOTIATING：当要将GTK发送给请求者时输入此状态。

注-发送GTK的TxRx标志始终与成对密钥是否用于数据加密/完整性相反。如果成对密钥用于加密/完整性，则STA从不与GTK一起发送;否则，STA使用GTK进行传输。

**8.5.7.1.3认证者状态机：组密钥握手（全局）**

以下列表总结了Authenticator状态机用于协调所有STA的组密钥更新的状态：

- GTK\_INIT：在系统初始化时进入该状态。

- SETKEYS：如果要在所有请求者上更新GTK，则进入此状态。

- SETKEYSDONE：如果GTK已在所有请求者上更新，则进入此状态。

NOTE-SETKEYSDONE调用SetGTK为所有关联的STA设置GTK。如果此操作失败，则通过此密钥进行的所有通信都将失败，并且AP需要检测并从此情况中恢复。

**8.5.7.2认证者状态机变量**

以下列表总结了Authenticator状态机使用的变量：

- AuthenticationRequest-如果STA的IEEE802.11管理实体希望对关联进行身份验证，则此变量设置为TRUE。这可以在STA关联时或在其他时间设置。

- ReAuthenticationRequest-如果IEEE802.1X身份验证器收到eapStart或设置了802.1X::reAuthenticate，则此变量设置为TRUE。

- DeauthenticationRequest-如果收到Disassociation或Deauthentication消息，则此变量设置为TRUE。

- Disconnect-当STA应启动解除身份验证时，此变量设置为TRUE。

- EAPOLKeyReceived-当收到EAPOL-Key帧时，此变量设置为TRUE。响应于由Authenticator发送的EAPOL-Key帧而接收的EAPOL-Key帧必须包含与发送的消息中的KeyReplayCounter字段相同的KeyReplayCounter字段值。应丢弃包含不同KeyReplayCounter字段值的EAPOL-Key帧。请求方响应来自Authenticator的EAPOLKey帧发送的EAPOL-Key帧不得设置Ack位。请求方发送的EAPOL-Key帧不响应来自Authenticator的EAPOL-Key帧，必须设置Request位。

注：

1. EAPOL-密钥类型为Pairwise和非零密钥索引的关键帧应该被忽略。

2-应忽略具有组/STAKey密钥类型和无效密钥索引的EAPOL-Key帧。

3-当接收到Ack位清零的EAPOL-Key帧时，它被期望作为对Authenticator发送的消息的回复，并且根据发送的EAPOL-Key帧中使用的重放计数器检查重放计数器。当接收到设置了请求位的EAPOL-Key帧时，则使用这些消息的重放计数器，该计数器是与用于向请求者发送消息的重放计数器不同的重放计数器。

- GTimeoutCtr-此变量维护组密钥握手的EAPOL-Key接收超时计数。每次EAPIOL-Key接收事件发生超时时，它都会递增，并初始化为0.附件D详细说明了超时值。EAPOL-Key帧的密钥重放计数器字段值应在EAPOL-Key帧的每次传输时递增。

- GInit-此变量用于初始化组密钥状态机。这是一个组变量。

- Init-此变量用于初始化每STA状态机

- TimeoutEvt-如果发出的EAPOL-Key帧无法从Supplicant获取响应，则此变量设置为TRUE。变量可以通过管理操作设置，也可以在PTKSTART和REKEYNEGOTIATING状态下通过超时操作设置。

- TimeoutCtr-此变量保持EAPOL-Key接收超时的计数。每次EAPOL-Key接收事件发生超时时，它都会递增，并初始化为0.附件D包含超时值的详细信息。EAPOL-Key帧的密钥重放计数器字段值应在EAPOL-Key帧的每次传输时递增。

- MICVerified-如果接收到的EAPOLKey帧上的MIC已经过验证且正确，则此变量设置为TRUE。任何具有无效MIC的EAPOL-Key帧都将被丢弃并被忽略。

- GTKAuthenticator-如果Authenticator在AP上，或者它是IBSS的指定认证者，则此变量设置为TRUE。

- GKeyDoneStations-更新GTK所需的STA数量。这是一个全局变量。

- GTKRekey-当需要组密钥握手时，此变量设置为TRUE。这是一个全局变量。

- GUpdateStationKeys-当新的GTK可用于发送给Supplicants时，此变量设置为TRUE。

- GNoStations-此变量计算Authenticators的数量，因此需要知道需要向GTK发送多少请求者。这是一个全局变量。

- Counter-此变量是全局STA密钥计数器。

- ANonce-如果STA是Authenticator，则此变量保存当前的nonce。

- GN，GM-这些是GTK当前的关键指标。交换（GM，GN）意味着GN中的全局关键指数与GM中的全局关键指数交换，因此现在GM和GN被逆转。

- PTK-此变量是当前PTK。

- GTK[]-此变量是每个GTK索引的当前GTK。

- PMK-该变量是保存当前PMK的缓冲区。

- 802.1X::XXX-此变量是IEEE802.1X状态变量XXX。

- keycount-此变量在IBSS模式下用于决定何时传递所有密钥并且IBSS链路是安全的。

**8.5.7.3认证者状态机程序**

以下列表总结了Authenticator状态机使用的过程：

-**STADisconnect**（）-执行此过程取消验证STA。

-**CalcGTK**（x）-生成GTK。

-**MIC**（x）-计算明文数据的MIC。

**8.5.8Nonce生成（提供信息）**

所有STA都包含一个全局密钥计数器，其大小为256位。它应该在系统启动时初始化为新的加密质量随机数。关于随机数生成，请参阅H.6。建议将计数器值初始化为以下值：

PRF-256(Randomnumber,“InitCounter”,LocalMACAddress||Time)

本地MAC地址应该是Authenticator上的AA和Supplicant上的SPA。

随机数的大小为256位。只要有可能，时间应该是[来自网络时间协议（NTP）或NTP格式的另一个时间]。此初始化是为了确保在系统重新启动时出现不同的初始密钥计数器值，而不管实时时钟是否可用。每次将值用作IV时，密钥计数器必须递增（全部256位）。不允许密钥计数器包装到初始化值。

**8.6将EAPOL密钥映射到IEEE802.11密钥**

**8.6.1将PTK映射到TKIP密钥**

有关从PTK派生的EAPOL临时密钥的定义，请参见8.5.1.2。

STA应使用临时密钥的0-127位作为TKIP阶段1和阶段2混合功能的输入。

STA应使用临时密钥的比特128-191作为从认证者STA到请求者STA或从发起STA到对等STA的用于STAKeys的MSDU的迈克尔密钥。

STA应使用临时密钥的比特192-255作为从请求者STA到认证者STA或从对等STA到STAKeys的发起STA的MSDU的迈克尔密钥。

**8.6.2将GTK映射到TKIP密钥**

有关从GTK派生的EAPOL临时密钥的定义，请参见8.5.1.3。

STA应使用临时密钥的0-127位作为TKIP阶段1和阶段2混合功能的输入。

STA应使用临时密钥的比特128-191作为从认证者STA到请求者STA或从发起STA到对等STA的用于STAKeys的MSDU的迈克尔密钥。

STA应使用临时密钥的比特192-255作为从请求者STA到认证者STA或从对等STA到STAKeys的发起STA的MSDU的迈克尔密钥。

**8.6.3将PTK映射到CCMP密钥**

有关从PTK派生的EAPOL临时密钥的定义，请参见8.5.1.2。

STA应使用临时密钥作为两个通信STA之间的MSDU的CCMP密钥。

**8.6.4将GTK映射到CCMP密钥**

有关从GTK派生的EAPOL临时密钥的定义，请参见8.5.1.3。

STA应使用临时密钥作为CCMP密钥。

**8.6.5将GTK映射到WEP-40密钥**

有关从GTK派生的EAPOL临时密钥的定义，请参见8.5.1.3。

STA应使用临时密钥的0-39位作为WEP-40密钥。

**8.6.6将GTK映射到WEP-104密钥**

有关从GTK派生的EAPOL临时密钥的定义，请参见8.5.1.3。

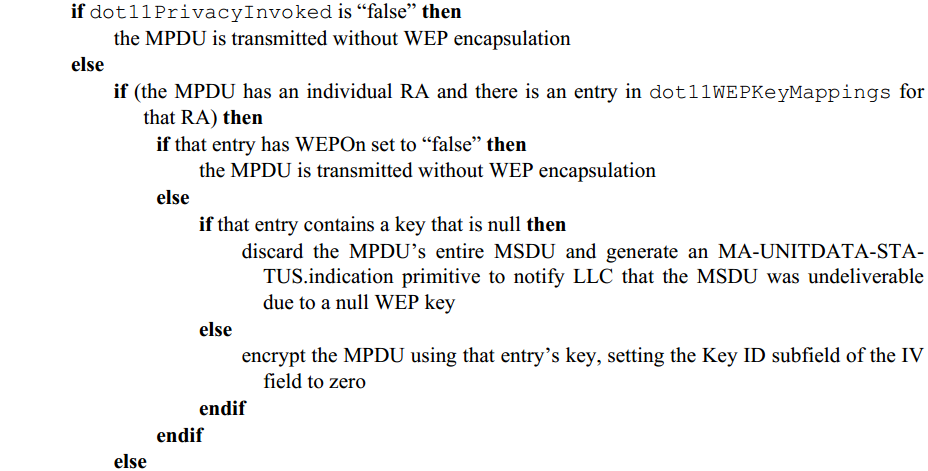
STA应使用临时密钥的0-103位作为WEP-104密钥。

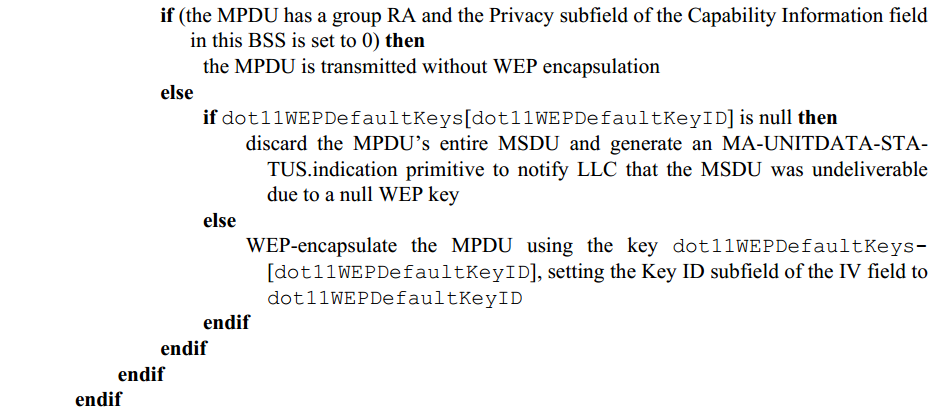
**8.7每帧伪代码**

**8.7.1WEP帧伪代码**

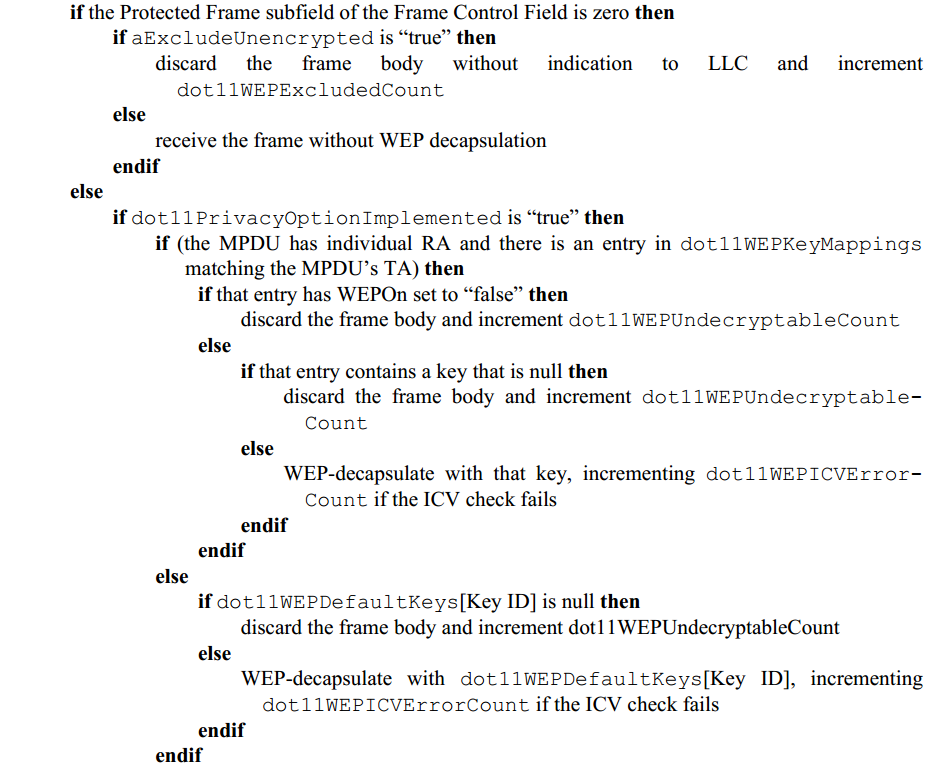
具有等于1的帧控制字段的受保护帧子字段的数据类型的MPDU被称为WEPMPDU。其他类型为Data的MPDU称为非WEPMPDU。

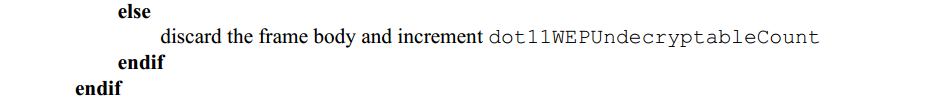
当MIB变量dot11PrivacyInvoked的值设置为FALSE时，STA不应传输WEP封装的MPDU。该MIB变量不影响MPDU或MAC管理协议数据单元（MMPDU）接收。





当布尔属性aExcludeUnencrypted设置为TRUE时，不应在MAC服务接口上指示非WEPMPDU，并且只应在MAC服务接口上指示从成功解密的MPDU成功重新组装的MSDU。当接收到Data类型的帧时，应根据以下决策树使用dot接收PHYRXSTART.indication原语时dot11PrivacyOptionImplemented，dot11WEPKeyMappings，dot11WEPDefaultKeys，dot11WEPDefaultKeyID和aExcludeUnencrypted的值。

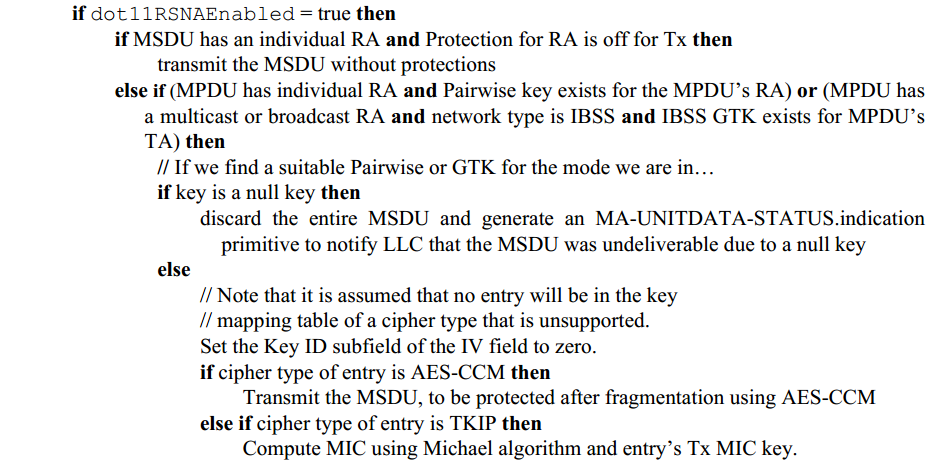


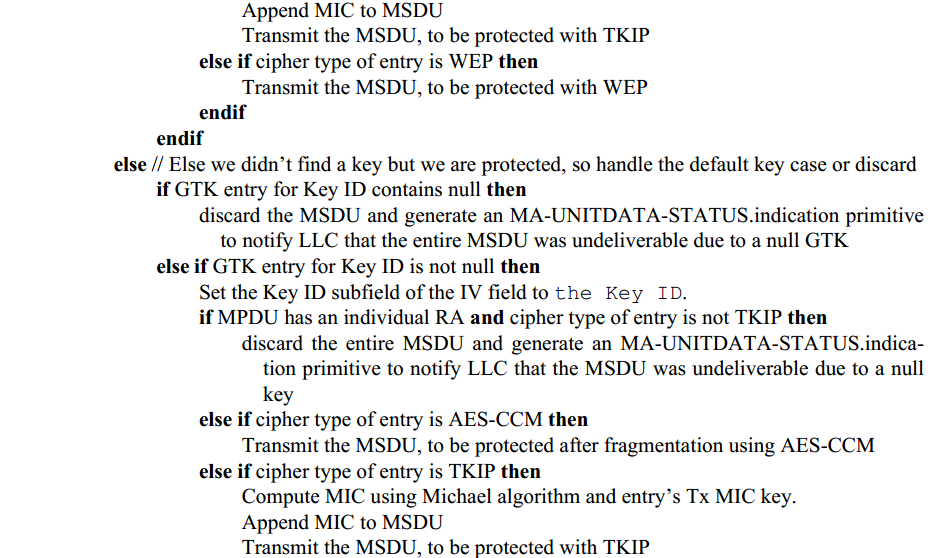


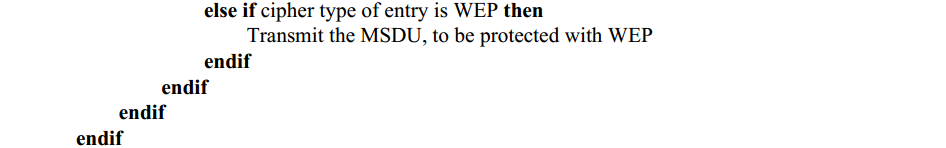
**8.7.2RSNA帧伪码**

当配置了临时密钥并且已经调用MLME.SETPROTECTION.request原语以便向该RA发送时，STA将受保护的MSDU发送到RA。当配置临时密钥并且已经调用MLME.SETPROTECTION.request原语以从该TA接收时，STA期望从TA接收受保护的MSDU。与这些条件不匹配的MSDU以明文形式发送，并以明文形式接收。

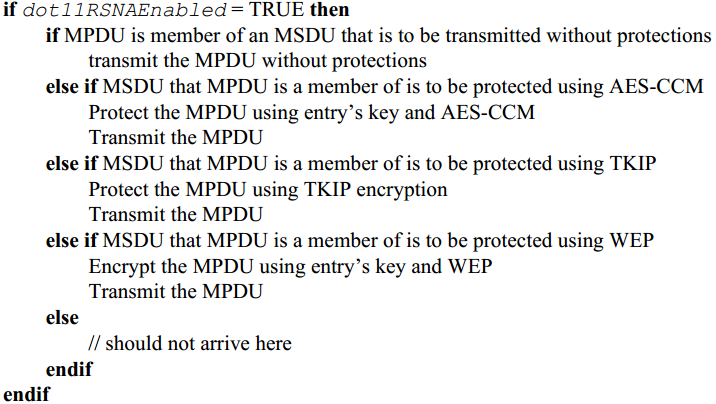
**8.7.2.1Per-MSDUTx伪代码**



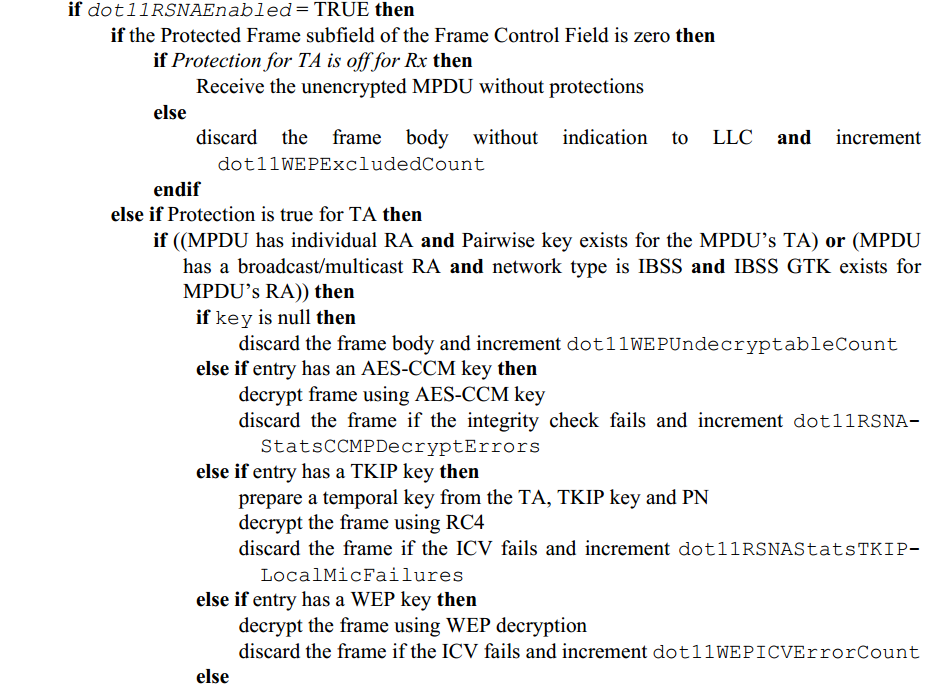


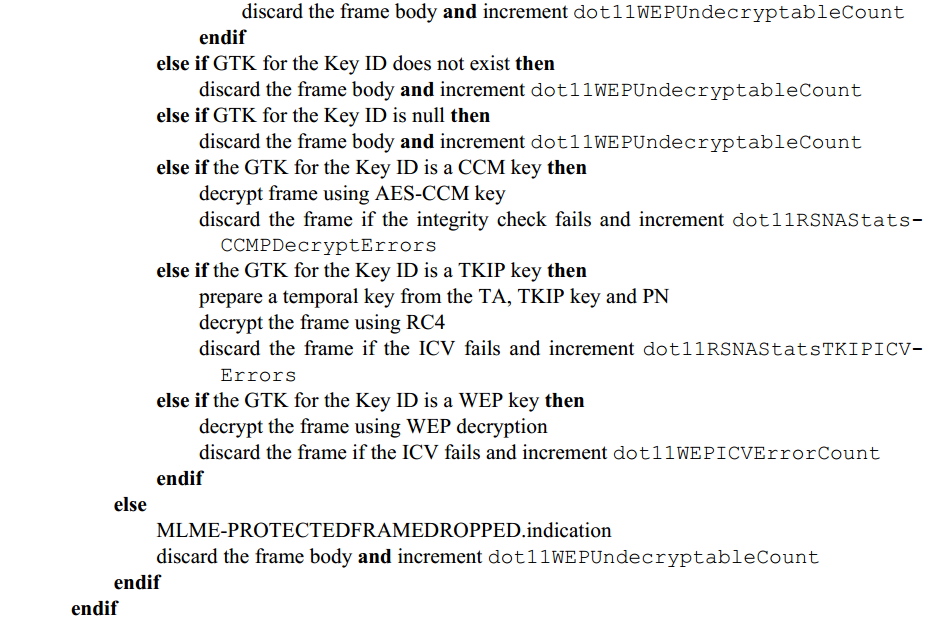


**8.7.2.2Per-MPDUTx伪代码**

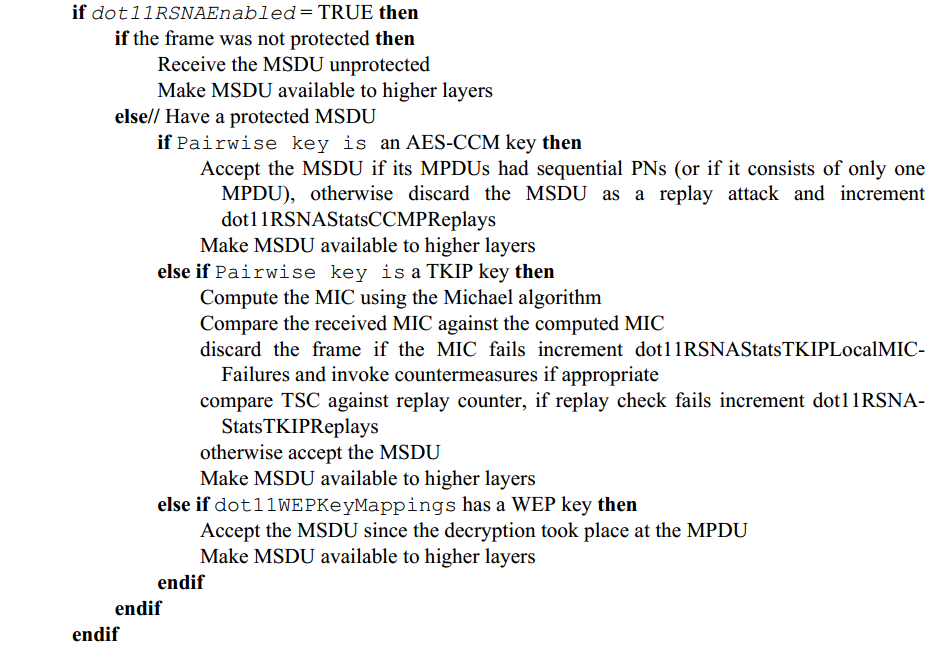


**8.7.2.3Per-MPDURx伪代码**





**8.7.2.4Per-MSDURx伪代码**



**10.层管理**

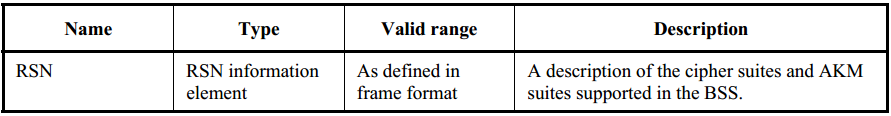
**10.3MLMESAP接口**

**10.3.2扫描**

**10.3.2.2MLME-SCAN.confirm**

**10.3.2.2.2服务原语的语义**

**在无标题表的末尾插入以下元素，列出BSSDescription的元素10.3.2.2.2：**

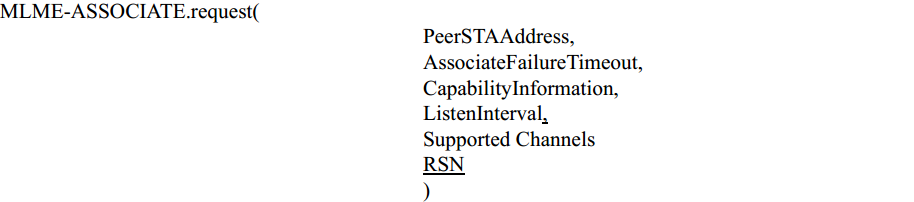


**10.3.6员工**

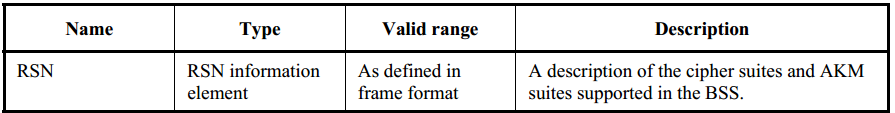
**10.3.6.1MLME-ASSOCIATE.request**

**10.3.6.1.2服务原语的语义**

**更改10.3.6.1.2中的以下原始参数列表：**



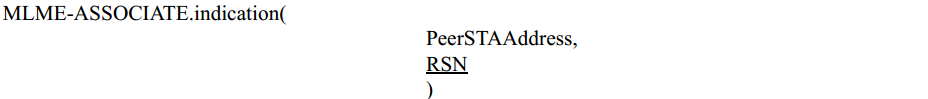
**在无标题表的末尾插入以下行，定义10.3.6.1.2中的基元参数：**



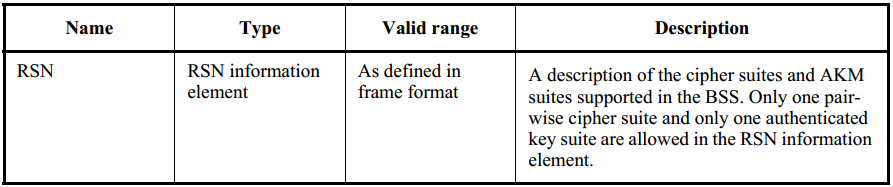
**10.3.6.3MLME-ASSOCIATE.indication**

**10.3.6.3.2服务原语的语义**

**更改10.3.6.3.2中的以下原始参数列表：**



**在无标题表的末尾插入以下行，定义10.3.6.3.2中的基元参数：**

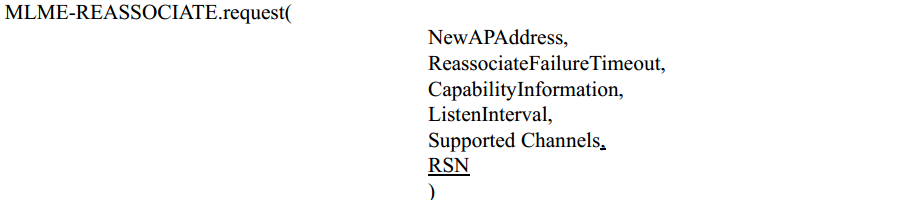


**10.3.7重新关联**

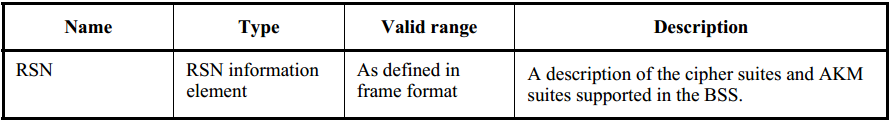
**10.3.7.1MLME-REASSOCIATE.request**

**10.3.7.1.2服务原语的语义**

**更改10.3.7.1.2中的以下原始参数列表：**



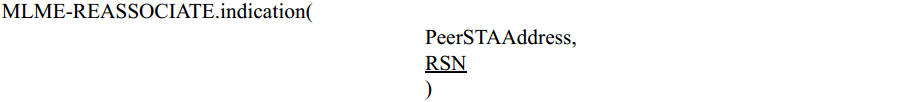
**在无标题表的末尾插入以下行，定义10.3.7.1.2中的基元参数：**



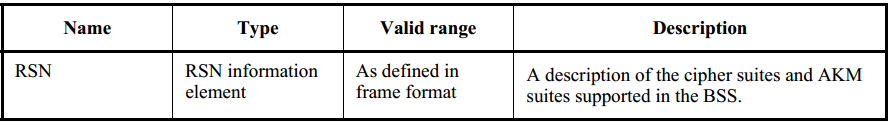
**10.3.7.3MLME-REASSOCIATE.indication**

**10.3.7.3.2服务原语的语义**

**更改10.3.7.3.2中的以下原始参数列表：**



**在无标题表的末尾插入以下行，定义10.3.7.3.2中的基元参数：**



**在10.3.16.2.4之后，插入10.3.17至10.3.23.1.4：**

**10.3.17SetKeys**

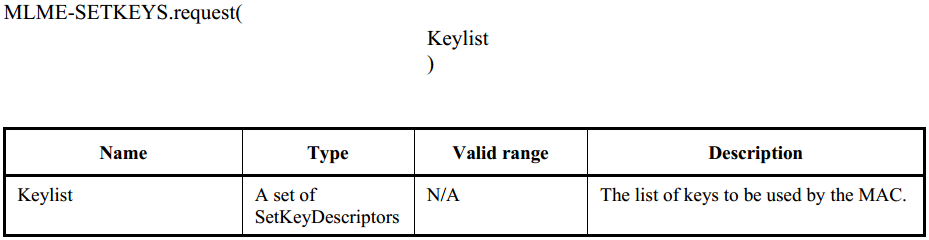
**10.3.17.1MLME-SETKEYS.request**

**10.3.17.1.1功能**

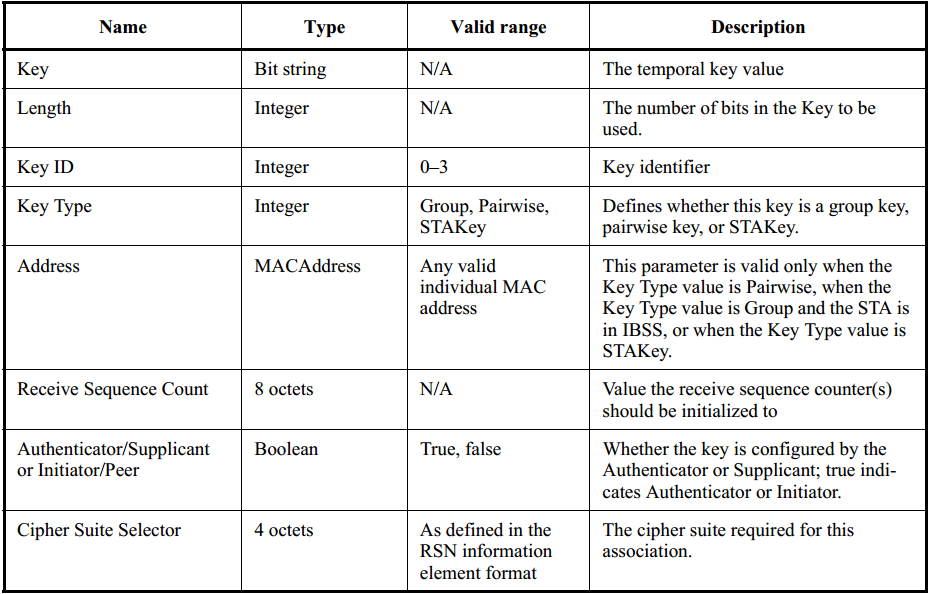
该原语使得在原语的参数中识别的密钥被设置在MAC中并且能够使用。

**10.3.17.1.2服务原语的语义**

原始参数如下：



每个SetKeyDescriptor包含以下元素：



**10.3.17.1.3生成时**

当要在MAC中设置一个或多个密钥时，该原语由SME生成。

**10.3.17.1.4收货的影响**

如果已经发出MLME-SETPROTECTION.request原语，则接收该原语使MAC设置适当的密钥并开始将它们用于将来的MAUNITDATA.request和MA-UNITDATA.indication原语。

**10.3.17.2MLME-SETKEYS.confirm**

**10.3.17.2.1功能**

该原语确认已完成相关MLME-SETKEYS.request原语的动作。

**10.3.17.2.2服务原语的语义**

该原语没有参数。

**10.3.17.2.3生成时**

该原语由MAC响应于接收到MLMESETKEYS.request原语而生成。当请求的操作完成时，将发出此原语。

**10.3.17.2.4收货的影响**

通知SME已完成MLME-SETKEYS.request原语的请求动作。

**10.3.18DeleteKeys**

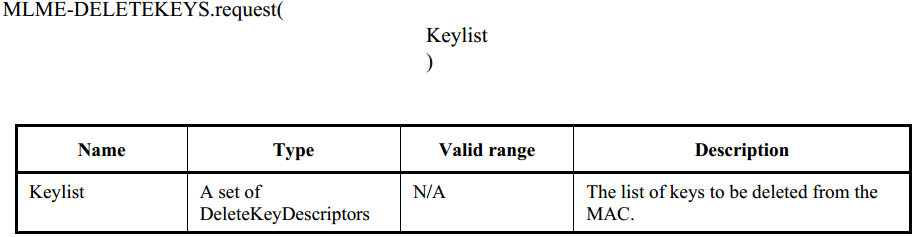
**10.3.18.1MLME-DELETEKEYS.request**

**10.3.18.1.1功能**

该原语使得从原语的参数中识别的密钥从MAC中删除并因此被禁用以供使用。

**10.3.18.1.2服务原语的语义**

原始参数如下：



每个DeleteKeyDescriptor包含以下元素：



**10.3.18.1.3生成时**

当要在MAC中删除用于安全关联的密钥时，该原语由SME生成。

**10.3.18.1.4收货的影响**

接收该原语使MAC删除由密钥组地址识别的临时密钥，包括Group，Pairwise和STAKey，并停止使用它们。

**10.3.18.2MLME-DELETEKEYS.confirm**

**10.3.18.2.1功能**

该原语确认已完成相关MLMEDELETEKEYS.request原语的动作。

**10.3.18.2.2服务原语的语义**

该原语没有参数。

**10.3.18.2.3生成时**

该原语由MAC响应于接收到MLMEDELETEKEYS.request原语而生成。当请求的操作完成时，将发出此原语。

**10.3.18.2.4收货的影响**

通知SME已完成MLMEDELETEKEYS.request原语的请求操作。

**10.3.19MIC（迈克尔）失败事件**

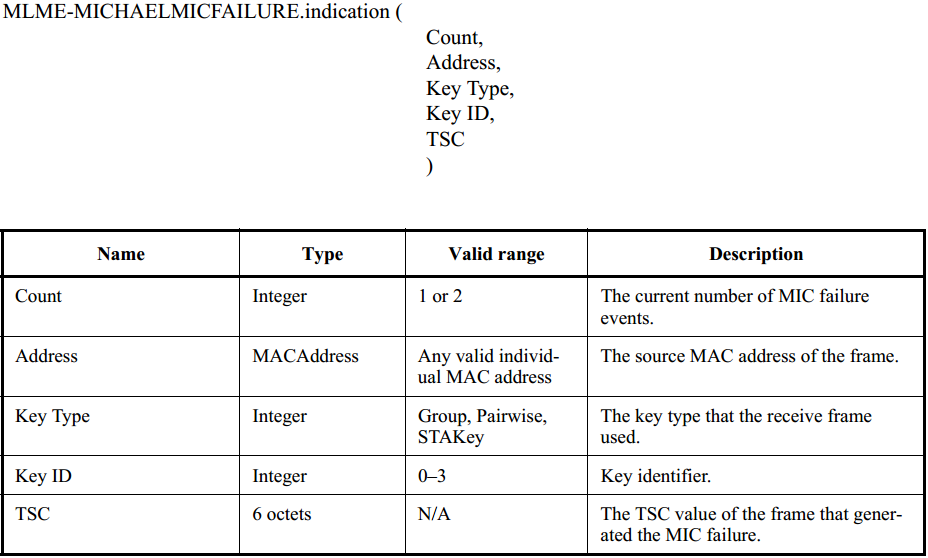
**10.3.19.1MLME-MICHAELMICFAILIL.indication**

**10.3.19.1.1功能**

该原语报告检测到MIC故障事件。

**10.3.19.1.2服务原语的语义**

原始参数如下：



**10.3.19.1.3生成时**

当MAC检测到MIC故障时，该原语由MAC生成。

**10.3.19.1.4收货的影响**

通知SMEMAC已检测到MIC故障。

**10.3.20EAPOL**

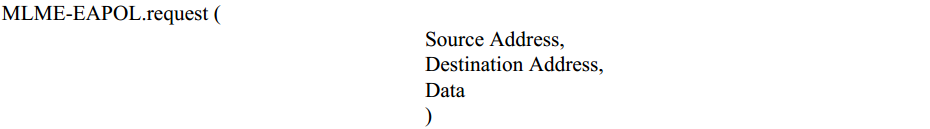
**10.3.20.1MLME-EAPOL.request**

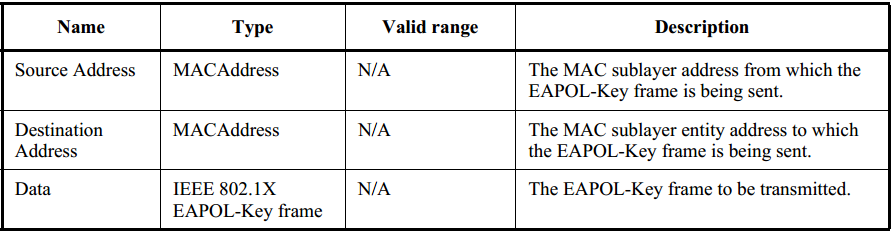
**10.3.20.1.1功能**

该原语用于传输迈克尔MIC失败报告帧。

**10.3.20.1.2服务原语的语义**

原始参数如下：





**10.3.20.1.3生成时**

当SME具有要发送的迈克尔MIC失败报告时，该原语由SME生成。

**10.3.20.1.4收据的效力**

MAC发送此EAPOL-Key帧。

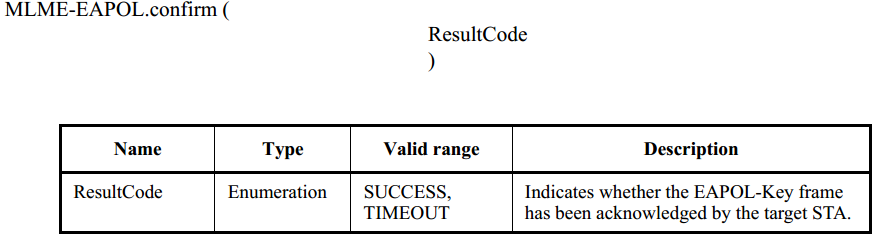
**10.3.20.2MLME-EAPOL.confirm**

**10.3.20.2.1功能**

该原语表示该IEEEEAPOL-Key帧已被IEEE802.11MAC确认。

**10.3.20.2.2服务原语的语义**

原始参数如下：



**10.3.20.2.3生成时**

由于生成MLME-EAPOL.request以发送EAPOL-Key帧，因此该原语由MAC生成。

**10.3.20.2.4收货的影响**

始终通知SME该EAPOL-Key帧是否已被IEEE802.11MAC确认。

**10.3.21MLME-STAKEYESTABLISHED**

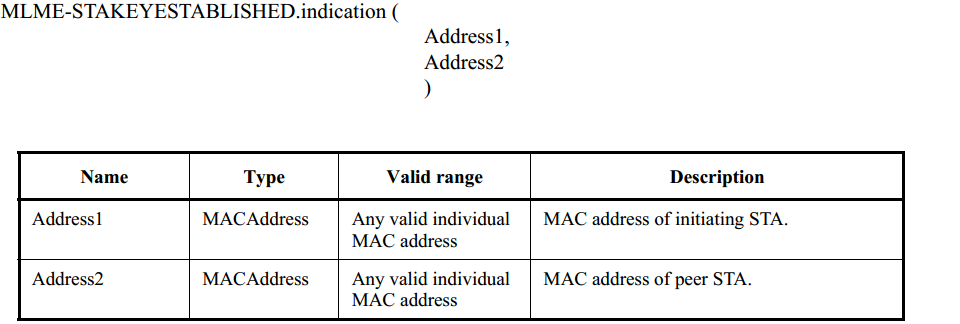
**10.3.21.1MLME-STAKEYESTABLISHED.indication**

**10.3.21.1.1功能**

该原语通知SME需要STAKey。

**10.3.21.1.2服务原语的语义**

该原语有两个参数，即两个STA的MAC地址。原始参数如下：



**10.3.21.1.3生成时**

当需要STAKey时，该原语由MAC生成。

**10.3.21.1.4收货的影响**

SME被通知需要STAKey，知道所涉及的两个STAMAC地址，然后可以向每个STA发送STAKey握手消息。

**10.3.22SetProtection**

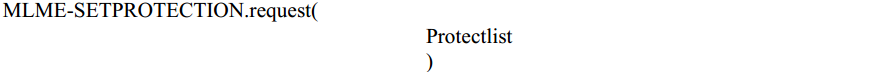
**10.3.22.1MLME-SETPROTECTION.request**

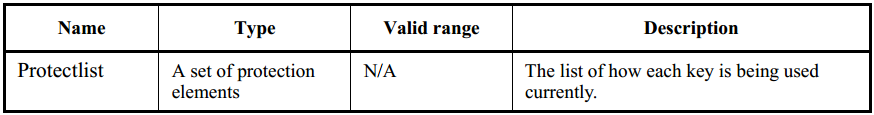
**10.3.22.1.1功能**

该原语指示对于发送到指示的MAC地址和从指示的MAC地址接收的帧是否需要保护。

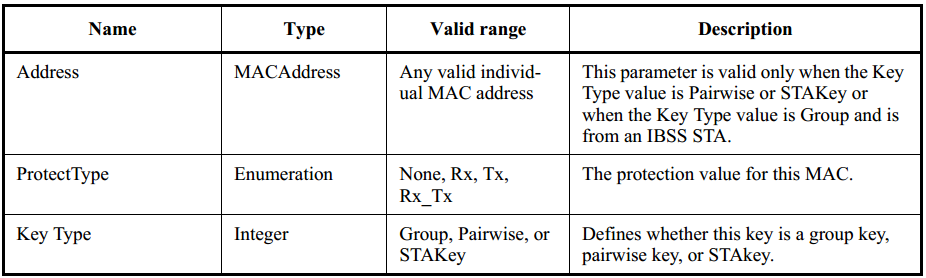
**10.3.22.1.2服务原语的语义**

原始参数如下：





每个Protectlist包含以下元素：



**10.3.22.1.3生成时**

当需要对发送到指示的MAC地址和从指示的MAC地址接收的帧进行保护时，该原语由SME生成。

**10.3.22.1.4收货的影响**

收到此原语会导致MAC设置保护并保护数据帧，如Protectlist参数的ProtectType元素所示：

- 无：指定不保护MAC地址和MAC地址的数据帧。

- Rx：指定应保护来自MAC地址的数据帧。

- Tx：指定应保护到MAC地址的数据帧。

- Rx\_Tx：指定应保护往返MAC地址的数据帧。

一旦指定数据帧受MAC地址保护或从MAC地址保护，这将由MLME-SETPROTECTION.request原语重置。MLMESETPROTECTION.request原语通过指定None来删除状态。

**10.3.22.2MLME-SETPROTECTION.confirm**

**10.3.22.2.1功能**

该原语表示帧保护请求已完成。

**10.3.22.2.2服务原语的语义**

此原语没有参数。

**10.3.22.2.3生成时**

当保护请求完成时，由MAC生成该原语。

**10.3.22.2.4收货的影响**

通知SME保护请求已完成。

**10.3.23MLME-PROTECTEDFRAMEDROPPED**

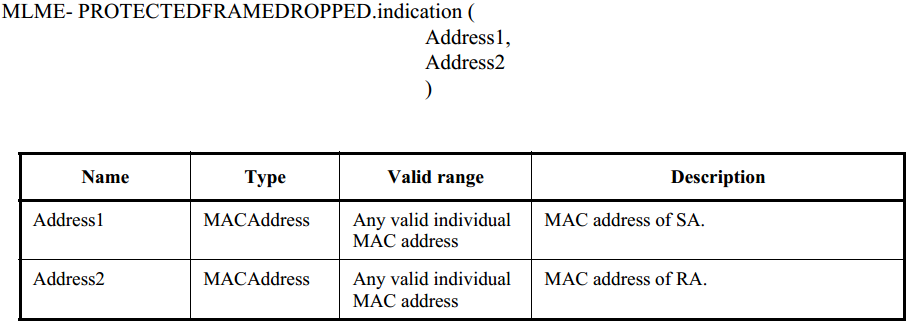
**10.3.23.1MLME-PROTECTEDFRAMEDROPPED.indication**

**10.3.23.1.1功能**

该原语通知SME由于临时密钥不可用而丢弃了帧。

**10.3.23.1.2服务原语的语义**

该原语有两个参数，即两个STA的MAC地址。原始参数如下：



**10.3.23.1.3生成时**

当帧被丢弃时由MAC生成该原语，因为没有临时密钥可用于该帧。

**10.3.23.1.4收货的影响**

通知SME丢弃了一个帧。SME可以在IBSS中使用该信息来发起与对等STA的安全关联。

**11.MAC子层管理实体**

用以下文字替换11.3和11.4的全部内容：

**11.3关联和重新关联**

本小节描述了用于IEEE802.11身份验证和解除身份验证的过程。本说明书中使用的状态是5.5中定义的状态。

**11.3.1发送认证的STA**

在收到MLME-AUTHENTICATE.request原语后，始发STA将使用以下过程与指示的STA进行认证：

a）在ESS中，或者在IBSS中，STA应执行8.2.2.2中描述的认证机制。

b）如果认证成功，则指示的STA的状态变量应设置为状态2。

c）STA应发出MLME-AUTHENTICATE.confirm原语，以通知SME认证结果。

在调用MLME-AUTHENTICATE.request原语之前，STA的SME应使用MLME-DELETEKEYS.request原语（见8.4.10）删除与指示的STA通信所保持的任何PTKSA和临时密钥。

**11.3.2认证目的STA**

在接收到认证事务序列号等于1的认证帧时，目标STA应使用以下过程与指示的STA进行认证：

a）STA应执行8.2.2.2中描述的认证机制。

b）STA应发出MLME-AUTHENTICATE.indication原语以通知SME认证。

STA的SME应在接收到MLMEAUTHENTICATE.indication原语时，通过使用MLMEDELETEKEYS.request原语（见8.4.10）删除与指示的STA进行通信而保持的任何PTKSA和临时密钥。

如果STA在IBSS中，如果SME决定发起RSNA，并且SME不知道对等体的安全策略，则可以通过调用MLMESCAN.request来发现单播探测请求帧到对等体。同行的安全政策。

**11.3.3发现身份验证的STA**

在收到MLME-DEAUTHENTICATE.request原语后，始发STA将使用以下过程与指示的STA解除验证：

a）如果指示的STA的状态变量处于状态2或状态3，则STA应向指示的STA发送解除认证帧。

b）指示STA的状态变量应设置为状态1。

c）STA应发出MLME-DEAUTHENTICATE.confirm原语，以通知SME完成取消认证。

STA的SME将在调用MLMEDEAUTHENTICATE.request原语之前，通过使用MLMEDELETEKEYS.request原语（见8.4.10）和调用MLMESETPROTECTION.request（None）来删除与指示的STA进行通信所保持的任何PTKSA和临时密钥。

**11.3.4解除身份验证目标STA**

在收到Deauthentication帧后，目标STA将使用以下过程与指示的STA解除验证：

a）指示STA的状态变量应设置为状态1。

b）STA应发出MLME-DEAUTHENTICATE.indication原语，以通知SME解除认证。

STA的SME将通过使用MLMEDELETEKEYS.request原语（见8.4.10）并在接收到MLMEDEAUTHENTICATE.indication原语时调用MLMESETPROTECTION.request（None）来删除与指示的STA进行通信而保持的任何PTKSA和临时密钥。

**11.4关联，重新关联和解除关联**

本子条款定义了STA如何与AP关联和重新关联以及它如何与AP解除关联。本说明书中使用的状态是5.5中定义的状态。

**11.4.1STA关联程序**

在收到MLME-ASSOCIATE.request原语后，STA应通过以下过程与AP关联：

a）STA应将关联请求帧发送到与该STA进行认证的AP。如果MLME-ASSOCIATE.request原语包含仅具有一个成对密码套件且仅有一个经认证的密钥套件的RSN信息元素，则该RSN信息元素应包括在关联请求帧中。

b）如果接收到状态值为“成功”的关联响应帧，则STA现在与AP相关联。状态变量应设置为状态3，MLME应发出指示操作成功完成的MLME-ASSOCIATE.confirm原语。

c）如果接收到具有除“成功”之外的状态值或者AssociateFailureTimeout到期的关联响应帧，则STA不与AP相关联。MLME应发出MLME-ASSOCIATE.confirm原语，指示操作失败。

d）SME应建立一个RSNA，或者它应通过调用保护类型设置为“Rx\_Tx”的MLME.SETPROTECTION.request原语来启用WEP，否则如果它不希望保护通信，它将不执行任何操作。

在调用MLME-ASSOCIATE.request原语之前，STA的SME应使用MLME-DELETEKEYS.request原语（见8.4.10）删除与指示的STA进行通信所保持的任何PTKSA和临时密钥。

**11.4.2AP关联程序**

当从STA接收到关联请求帧时，AP应使用以下过程与STA关联：

a）如果STA未被认证，则AP应向STA发送解除认证帧并终止关联过程。

b）在RSNA中，AP应检查RSN信息单元中接收的值，以查看接收的值是否与AP的安全策略相匹配。如果不是，则不接受该协会。

c）AP应发送一个具有7.3.1.9中定义的状态码的关联响应。如果状态值是“成功”，则分配给STA的关联标识符应包括在响应中。

d）当STA确认状态值为“成功”的关联响应时，STA被认为与该AP相关联。STA的状态变量应设置为状态3。

e）MLME应发布MLME-ASSOCIATE.indication原语，以通知SME该关联。

f）SME应建立一个RSNA，或者它应通过调用保护类型设置为“Rx\_Tx”的MLME.SETPROTECTION.request原语来启用WEP，否则如果它不希望保护通信，它将不执行任何操作。

g）中小企业将告知DS新协会。

STA的SME应在接收到MLME-ASSOCIATE.indication原语时，使用MLME-DELETEKEYS.request原语（见8.4.10）删除与指示的STA进行通信所保持的任何PTKSA和临时密钥。

**11.4.3STA重新关联程序**

收到MLME-REASSOCIATE.request原语后，STA应通过以下过程与AP重新关联：

a）如果状态变量处于状态1，则STA应通过发出MLME-REASSOCIATE.confirm原语通知SME重新关联失败。

b）STA应将重新关联请求帧发送到新AP。如果MLME-REASSOCIATE.request原语包含仅具有一个成对密码套件且仅有一个经认证的密钥套件的RSN信息元素，则该RSN信息元素应包括在重新关联请求帧中。

c）如果接收到状态值为“成功”的重新关联响应帧，则STA现在与新AP相关联。状态变量应设置为状态3，MLME应发出MLME-REASSOCIATE.confirm原语，指示操作成功完成。

d）如果接收到具有除“成功”以外的状态值或者AssociateFailureTimeout到期的重新关联响应帧，则STA不与AP相关联。MLME应发出MLME-REASSOCIATE.confirm原语，指示操作失败。

e）SME应建立一个RSNA，或者它应通过调用保护类型设置为“Rx\_Tx”的MLME.SETPROTECTION.request原语来启用WEP，否则如果它不希望保密通信则它将不执行任何操作。

在调用MLME-REASSOCIATE.request原语之前，STA的SME应使用MLME-DELETEKEYS.request原语（见8.4.10）删除与指示的STA进行通信的任何PTKSA和临时密钥。

**11.4.4AP重新关联程序**

每当从STA接收到重新关联请求帧时，AP使用以下过程来支持重新关联：

a）如果STA未被认证，则AP应向STA发送Deauthentication帧并终止重新关联过程。

b）在RSNA中，AP应检查RSN信息单元中接收的值，以查看接收的值是否与AP的安全策略相匹配。如果不是，则不接受该协会。

c）AP应发送一个重新关联响应帧，其状态码如7.3.1.9中所定义。如果状态值是“成功”，则分配给STA的关联标识符应包括在响应中。

d）当STA确认状态值为“成功”的重新关联响应帧时，STA被认为与该AP相关联。STA的状态变量应设置为状态3。

e）MLME应发布MLME-REASSOCIATE.indication原语，以通知SME该关联。

f）SME应建立一个RSNA，或者它应通过调用保护类型设置为“Rx\_Tx”的MLME.SETPROTECTION.request原语来启用WEP，否则如果它不希望保护通信，它将不执行任何操作。

g）中小企业将告知DS新协会。

STA的SME应在接收到MLME-REASSOCIATE.indication原语时，使用MLME-DELETEKEYS.request原语（见8.4.10）删除与指示的STA进行通信所保持的任何PTKSA和临时密钥。

**11.4.5STA解除关联程序**

在收到MLME-DISASSOCIATE.request原语后，关联的STA应使用以下过程与AP解除关联：

a）STA应将Disassociation帧发送到与该STA相关联的AP。

b）当且仅当它不是状态1时，AP的状态变量应设置为状态2。

c）MLME应发出MLME-DISASSOCIATE.confirm原语，指示操作的成功完成。

STA的SME应在调用MLMEDISASSOCIATE.request原语之前，通过使用MLMEDELETEKEYS.request原语（见8.4.10）和调用MLMESETPROTECTION.request（None）来删除与指示的STA进行通信所保持的任何PTKSA和临时密钥。

**11.4.6AP解除关联程序**

在从关联的STA收到解除关联帧后，AP应通过以下过程解除STA的关联：

a）STA的状态变量应设置为状态2。

b）MLME应发布MLME-DISASSOCIATE.indication原语，以通知SME解除关联。

c）中小企业将更新DS。

STA的SME将通过使用MLMEDELETEKEYS.request原语（见8.4.10）并在接收到MLMEDISASSOCIATE.indication原语时调用MLMESETPROTECTION.request（None）来删除为与指示的STA通信而保持的任何PTKSA和临时密钥。