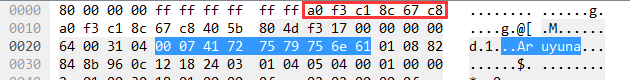
# WPA/WPA2 加密全流程

李大燊

中山大学

2018.1.4

WiFi无线接入点（以下简称AP）在启动后会向周围发送Beacon无线信号，表示本大爷在这一带，你们知道密码的可以连过来。此时发送的Beacon帧中包含有自己的WiFi名字（以下简称SSID），Beacon帧的前几位如下。



问题1：怎么知道这个是Beacon帧

答案：前面4字节是8后面7个0的就是

问题2：SSID在哪里

答案：上图中深蓝底部分，前2字节表示SSID的长度，这里是7，表示接下来的7字节就是SSID，从41……到61这部分就是SSID的二进制编码（全英文的话就是ascii码），可以看右边的深蓝底部分得到明文形式。SSID长度值的位置固定在0x0024-25这个位置，在这里提取长度（假设为x）后，从0x0026开始往后推到x的字节的这部分就是SSID的值

问题3：那个红框框着的是啥

答案：该AP的Mac地址，之后解包时要根据这个值确定这个握手包对应的SSID，因为握手包里没有SSID

之后，AP会根据自身设定的SSID和密码，计算出认证时使用的PMK（PMK的计算会在下面介绍）。

手机（或者平板电脑等其他可以无线上网的设备，以下简称STA）开启WiFi后，会搜索周围的WiFi信号，接收到AP发送的Beacon帧，就会将该AP的信息打在手机界面上，包括SSID，加密方式（开放/WEP/WPA/WPA2/802.11）。

一般的AP（包括无线路由器，手机热点）加密方式都是WPA/WPA2方式（当然也有可能是开放，这种如果不是需要浏览器登录方式的话简直是小天使ヾ（´▽｀））

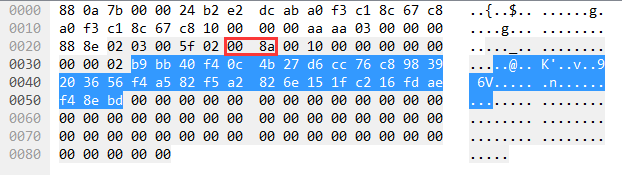
WEP作为10年前就公认的SB加密方式就不去讨论了。

如果是WPA/WPA2方式，那么需要验证WiFi密码才能登陆。

验证过程包括4次握手：

**Round 1. AP→STA**

AP（路由器）产生了一个256位（32字节）的随机数（下面简称ANonce）用于这次验证，同时EAPOL数据包里还包含了本WiFi的加密方式（体现在Key Information的最后3位，如果WiFi是WPA加密，这3位为001，使用HMAC\_MD5加密算法计算MIC；如果是WPA2加密，这3位为010，使用HMAC\_SHA1加密算法计算MIC。当然WPA和WPA2就只有计算MIC算法不同这一个差别，其他部分是一致的。哦还有，WPA数据传输时使用TKIP加密，这是有可能确解的，而WPA2数据传输使用AES-CCMP加密，这个无法反推得到）。其他资料完全后，AP把这个包发给STA。



问题1：我怎么知道这是EAPOL数据包

答案：开头第1字节是88的，中间有一段Logical-Link Control字段（001a~0021那里为”aa aa 03 00 00 00 88 8e”）。

问题2：我怎么知道是WPA还是WPA2

答案：看红框（固定在0x0027-28位置），为Key Information字段，最后字符是a，写成二进制就是1010，后3位是010，写成十六进制是2，从前面的说法，知道这是WPA2加密方式

问题3：ANonce在哪？

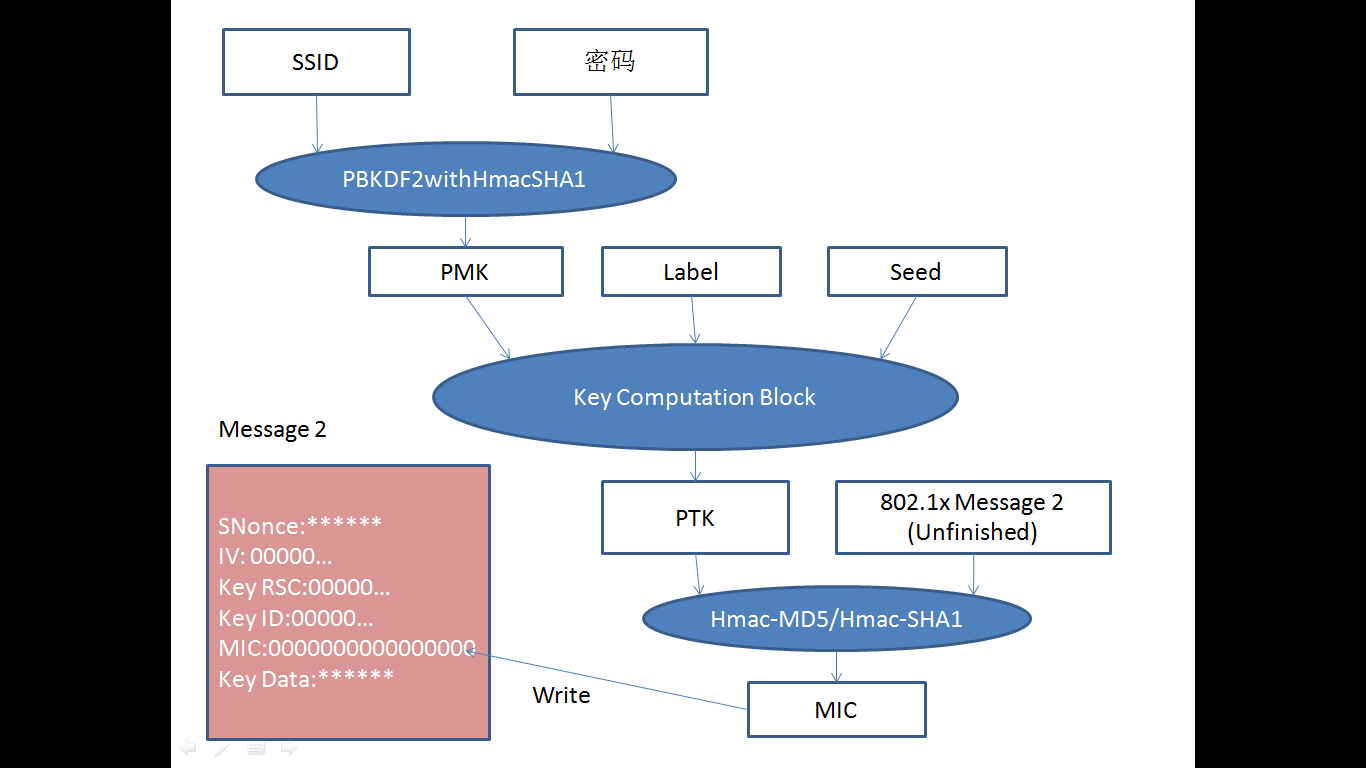
答案：上图中深蓝底部分（固定0x0033-52位置）

**Round 2. AP→STA**

关键部分来了， 破解的关键全在这个包（除了没有ANonce）。

STA（你的手机）接到AP发来的第一个包后开始了一系列的计算：

一图以蔽之：



下面将详细介绍这个图。

你输入了这个AP的密码（可能是对的也可能是错的，但STA并不知道这是不是对的，它只管拿到你输入的密码，然后开始计算）

STA使用这个AP的SSID和你刚刚输入的密码（简称psw）使用PBKDF2\_SHA1算法加密得到临时主密钥（PMK）：

PMK=PBKDF2（HMAC\_SHA1(), psw, SSID, 4096, 256）;

参数列表：HMAC\_SHA1()是加密用的迭代算法，WiFi加密全部用HMAC\_SHA1。psw为密码，SSID字面意思，4096表示迭代4096次，256是最后输出的位数（256位，32字节）

对PBKDF2算法理解不够深，细节方面现在说不了。以后补。

由于4096这个参数，计算PMK时将会执行4096次HMAC\_SHA1加密函数，还有一些其他的乱七八糟的字符串拼接等等的操作，因此计算一个PMK是及其耗时的，也是破解的难点所在。这也是为什么手机连一个新的WiFi时连上去要好几秒，而连一个之前连过的旧WiFi都是秒连，因为你连一个旧WiFi不用再输密码，也就不用修改PMK（PMK的结果只与WiFi名字和密码有关）。当然如果旧WiFi路由器偷偷改了密码，你连上去会给你说密码错误，因为PMK已经失效了。

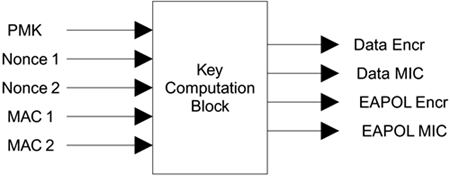
STA在计算完PMK后，它会从Beacon帧里抓出AP的Mac地址（这个在第1次握手里也能抓到；下称AMac），从第1次握手包里抓出ANonce，然后STA自己也生成一个随机数（也是256位，下称SNonce），再加上它自己的Mac地址（下称SMac），这5个东西作为参数生成成对扩展密钥（PTK）……

——————————————起始分割线——————————————

在这里先打住，讲点非常好玩的东西！

之前翻阅了很多讲述WPA加密的书，对于PTK的计算，他们的说法是：利用PMK，AMac，SMac，ANonce，SNonce通过一系列加密生成512位（64字节）的PTK，PTK分为4部分：KCK（0-127位），KEK（128-255位），TK1（256-383位），TK2（384-511位）。【注：WPA2貌似没有TK2，不过这都不影响了，因为破解密码只需要KCK，剩下的是连上WiFi之后的事了。】

然后贴上这个图

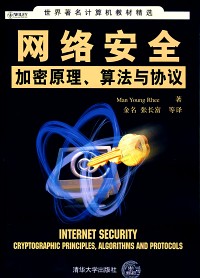


这个图我相信已经烂大街了，部分介绍WPA四次握手的技术博客也用了。但问题是这个Key Computation Block里面到底是什么？

书上没找到，去Internet上翻了一圈，所有讲述了WPA加密的文章这里的说法全是“通过一系列计算生成PTK”，然后摆出上面那张图。兄弟们啊，这个Block里面有啥？？

当然好一点的是：PTK=SHA1\_PRF(…)。好歹还把算法讲出来了，但是关于这个算法，几乎找不到任何资料支撑（天上掉下来的？），而提到这个计算公式的文章，就把这个公式摆出来，然后提一下“前16字节用于生成MIC”然后就到了下一步……

后来在图书馆里找到一本关于网络安全算法介绍的书，如下图（《网络安全：加密原理、算法与协议》，ISBN 978-7-302-15259-0）



里面提到了SHA1，HMAC和PRF算法（希望的曙光？），但没有提到前面的PBKDF2。

算了，见一步走一步就行，于是开始琢磨这个传说中的PRF，发现还要人为去定义这个PRF的规则（emmmmm）。

去撸代码实现了书上的例子，发现把这5个玩意扔进去最后的结果一点都不一样（怎么会一样，书上的例子规则和WPA规则完全不一样）

因此在一脸懵逼之际，突然想起来Aircrack-ng这个软件是开源的！于是去瞄了一眼源代码，找到了计算PTK的方法，这才终于解决了这个PRF算法是什么的问题（哈哈哈哈哈）……

故事完结，回到正文。

——————————————结束分割线——————————————

PTK=PRF(HMAC\_SHA1(), secret, label, seed);

参数说明：

HMAC\_SHA1是加密算法（又是你）。

secret是PMK。

label是标签（一段扩展密钥用的字符串，在WPA加密的PTK扩展中是固定字符串[Pairwise key expansion]，区分大小写，有空格，22个字节）。

seed是一段东西的拼接，这些东西共76字节，前12字节是AMac与SMac的值（较小在前，较大在后），后64字节是ANonce与SNonce（较小在前，较大在后），然后这4个东西直接拼接形成76字节作为seed。

那么生成PTK的PRF算法的规则是啥？下面告诉你：

第①步：先把label和seed通过特殊规则拼起来得到Labelseed（共100字节）……

——————————————起始分割线——————————————

好玩的东西又来了。

之前提到的那本书提到了PRF算法，他对PRF是这样描述的：

A(0)=HMAC\_SHA1(psw, Label || seed);（这里psw是一般加密密钥）

B(0)=HMAC\_SHA1(psw, A(0));

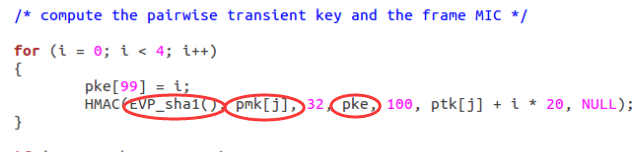
A(n)=HMAC\_SHA1(psw, A(n-1) || Label || seed);

B(n)=HMAC\_SHA1(psw, A(n)); (n=1,2,……)

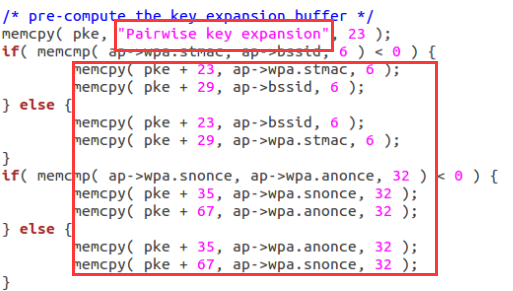
当时发现了WPA的PRF算法后我直接把Label和seed拼起来作为Text输入，但结果永远与Aircrack-ng的不一样。

WTF？？

Aircrack-ng源码上的确也是这个参数格式啊



我对了一遍label也没错，seed也没错……



咦不对，从ptk内存的第23个位置开始填，那label是多长来着？

8+3+9+2=22

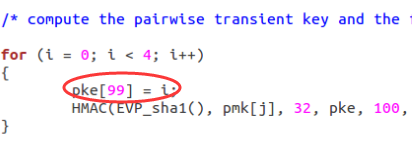
嗯？再一个一个数……还是22

原来第23个位置空出来的？

回去把计算模块更新到下标22的位置空出来，从23到98为那4段文字的拼接，输进去，发现还是错的

……………………

又回去Aircrack-ng源码那里，这次发现了一行短到被忽视的代码



原来PRF是这么个意思！！！

故事完结，回到正文

——————————————结束分割线——————————————

Labelseed的拼接规则是：0-21字节为Label字段（22字节），第22字节为0（全0二进制位，不是ascii为48的那个0），第23-98字节为seed字段（76字节），第99字节为扩展次数（初始为0）。

第②步：把上面生成的Labelseed作为Text与PMK作为key通过HMAC\_SHA1算法得到第1个结果（共20字节）。扩展次数加1。

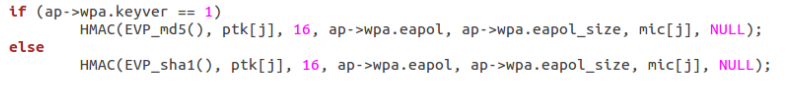
第③步：重复①②，后续每次的HMAC\_SHA1得到的结果直接拼到前一次结果的后面，一共4次最终得到80字节的扩展码。最后取前64（WPA）/48（WPA2）字节作为PTK。

PTK的前16字节作为KCK与一个802.1x数据帧作为参数根据WPA或者WPA2的加密方式选择对应加密算法计算MIC……

——————————————起始分割线——————————————

好了又要讲故事了，正文真短。

从Aircrack-ng源码里，刚刚PTK的代码后面就是MIC计算代码



参数是PTK前16字节，以及ap结构里的wpa成员的eapol值，这个值我翻了一下结构描述的头文件，一个buffer为256的char型，记录EAPOL帧内容。

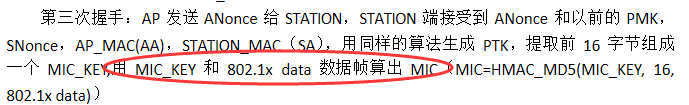
但从代码里没看出来这指的是哪个帧，从数据完整性判断很大可能性是第2次握手的数据。

但第2次握手中MIC字段已经是结果，是要抽出来对比的，这一整个数据段带MIC写进去肯定最后计算出的MIC不一样。

我试过把最后22字节（标示为Key Data）的部分作为参数，但结果不对。

然后想翻翻网上有没有什么介绍，一翻把我笑尿了。

在那些介绍WPA加密原理的文章里，对于MIC的计算描述大概全是这个样子



后面MD5是WPA模式使用的算法，WPA2用的SHA1这个从前面的代码里看出来了，所以不管那个问题。问题是，这个“802.1x data数据帧”又是啥？

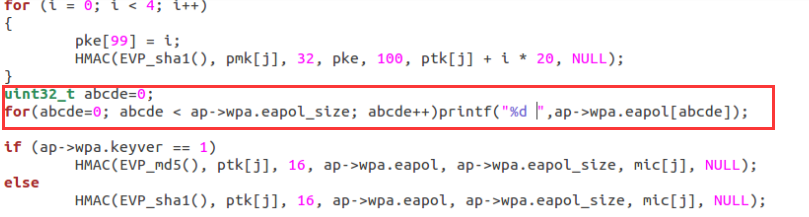
所有文章的该段文字前后都没有提及这个MIC是什么（葛优躺

兄弟们啊，你们写篇技术类博客不要整篇照抄的好伐？搞事呢？

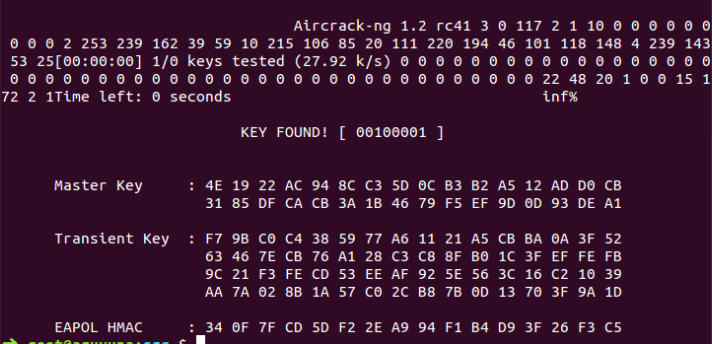
代码都看不出来是什么，正当束手无策之际，突然想起来了一个绝招——

Debug！

于是我加了这么两行（前面是计算PTK的，后面是计算MIC的，无视%d后面那个光标）



重新make了一次，运行跑包程序，界面被我搞成了这个样子



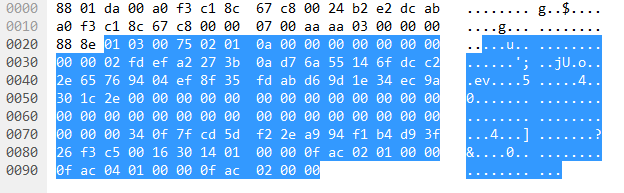
（原来的第一行到”rc4”就结束了）

好了，这个EAPOL data帧到底是谁很明了了，对比前面几位是1（0x01），3（0x03），0（0x00），117（0x75），2（0x02），1（0x01）……

那么在十六进制模式下看到的应该是

01 03 00 75 02 01……

回去Wireshark一比对，真的似李，第2次！



中间和最后有部分文字被Aircrack的说明文覆盖了，很明显这个eapol帧指的是这一整段深蓝色部分。但问题来了，MIC字段那里原来应该是什么？

MIC位置在倒数第2,3行那个位置，后面接的是00 16代表后面0x16个字节是Key Data，所以倒回到00 16前面的部分（在命令行里的输出应该是0 22 48 20……）

很容易就找到了，发现这里原来全是零（震惊！困扰了这么久的问题居然是……）

故事结束，回到正文。

——————————————结束分割线——————————————

计算MIC的这两个参数分别是PTK的前16位作为Key，自身的802.1x Auth数据帧部分（固定0x0022-最后，0x0024-25存了后续数据总长，这个数加上4就是数据帧总长）把其他该填的都填好（MIC暂时不填，全0），保存下来作为Text，使用HMAC\_MD5（对WPA）或者HMAC\_SHA1（对WPA2）算法计算结果，前16位填充到这次握手包的MIC字段中。最后，STA把这个EAPOL帧发给AP。

**Round 3：AP→STA**

终于来到了Round 3了，前面Round 2真是充满了怨念hhhhhhhh

AP之前第1次握手时已经指定了STA的MAC，也保存了自己的Nonce和MAC，以及早已计算好的PMK（见第1页）。收到了第2次握手包后，从里面抓出来SNonce和将要对比的MIC。这些参数用上面同样的方法生成PTK，MIC，并把这个MIC结果和第2次握手包里抓出来的MIC进行匹配，完全一致则原来STA输入了正确密码，验证通过，发一个数据包给AP表示密钥已装好，可以开始通信了【通信数据加密使用TK（256/128位对于WPA/WPA2）作为加密Key】。

如果MIC不一致，（滑稽）则AP丢掉这个包，认证失败。你想连接就重来，从第1次握手重新开始。之后ANonce，SNonce都会不一样。

**Round 4: STA→AP**

STA收到AP的通过帧后同样安装密钥进行通信，并给AP发送一个ACK帧表示已经准备好。同时STA的屏幕界面显示已连接。

如果想破解这个密码的话，按照Round 2的流程把程序弄出来就行了。当然，重点在优化PMK的生成上面。还有需要注意的是，这必须要是一个完整的4次握手包才能确解，因为出现第3，4次才表明验证成功，如果只有2次的话结果是不正确的，因为密码错误了算出来的MIC也是错误的（虽然Aircrack-ng查到2次握手照样跑而且如果字典里有的话会跑出一个你输入的错误密码2333333）。