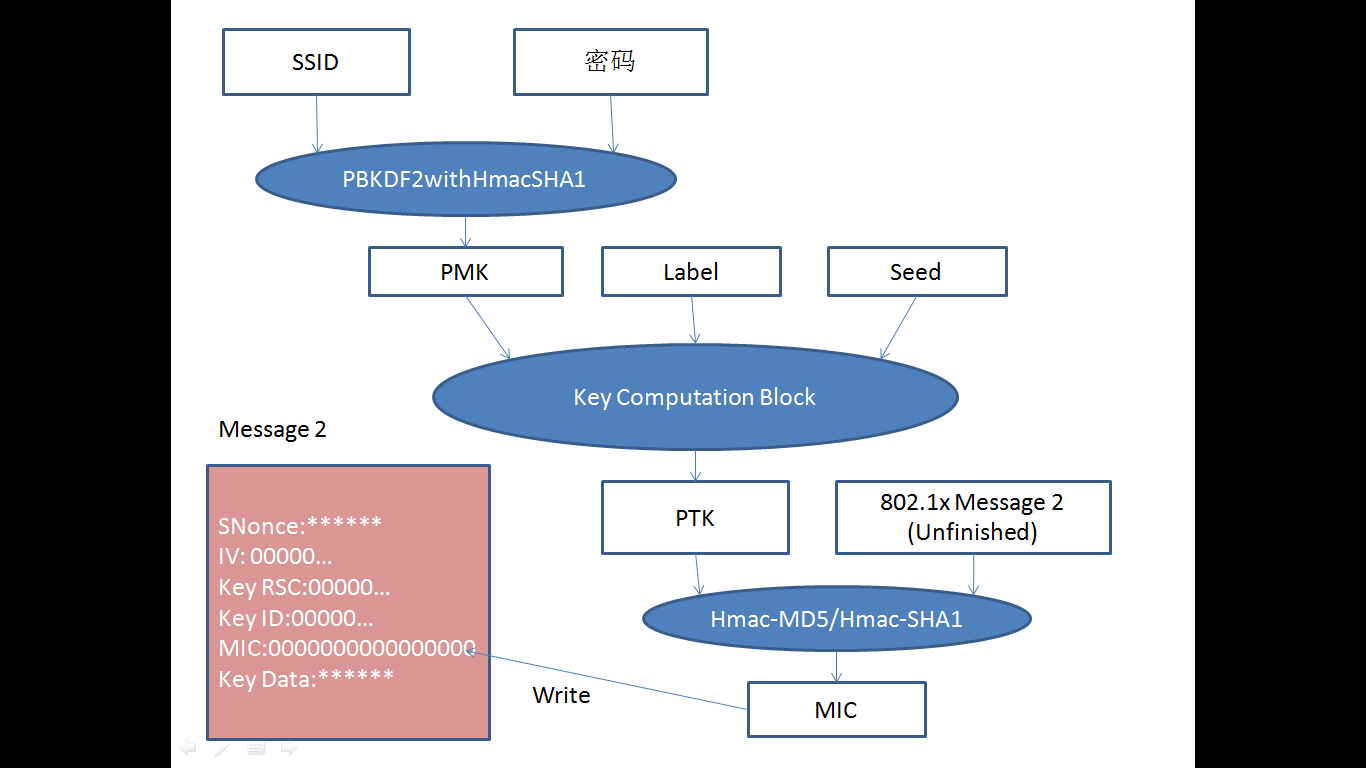
（1）MapReduce系统的主要任务

WPA/WPA2加密方式过程为



MapReduce的任务就是第一部分（PBKDF2withHmacSHA1）。用SSID和密码（用户输入的密码），用PBKDF2迭代HmacSHA1到4096次，取前256位获得PMK。由于此步骤要迭代4096次，因此消耗较多计算资源，耗时最长。后面的两个步骤都只调用1次Hmac算法，因此PBKDF2部分消耗时间占了破解全流程的2048/2049=99.95%。现在哈希表破解的原理就是预先生成字典里的密码与特定WiFi名结合生成PMK的字典，再从这个PMK字典里直接提取PMK进行破解。

详细请参照文档《WPA/WPA2加密全流程》

（2）Map任务

内容：读取指定的HDFS中的文件，解析成多个键值，map方法之后转换为多组键值对<key, value>输出到Reduce任务，Reduce根据这些键值对逐一计算并输出计算结果。

在本课题中，从（1）的分析中应有：Map任务的工作就是从字典读取密码，与SSID生成键值对输出到Reduce。

但注意一下key和value的取值。这里SSID是固定字符串，而密码有很多，最后生成的PMK字典中每一个密码对应一个PMK。在MapReduce中，相同的Key的值通过一系列操作后最后只会剩下1个值。因此输出的字典中应该是一个密码对应一个PMK，输出到Reduce的Map任务中，key应该是密码，而value是SSID。这里和一般的键值对取值思维不一样。

因此设计应该是，字典中每个密码读取为key，而value直接取SSID（Map类和Reduce类都有个比较尴尬的地方，就是无法通过传参数给类里的变量赋值。Map类想要从外部取值，必须只能读取hdfs里文件的值。）

因此map函数代码如下：

StringTokenizer stk = new StringTokenizer(value.toString());

getSSID(); //SSID赋值

while(stk.hasMoreTokens()){ //对字典里所有的Key

Password.set(stk.nextToken()); //把密码赋值给key

context.write(Password, SSID); //输出键值对<key, value>

}

（3）Reduce任务

内容：从map任务中输出的<key, value>对会作为Reduce的输入，相同的key的所有value会根据Reduce方法的内容去处理，并得到最后的值，作为这个key对应的结果。

那么本课题中Reduce的任务就是根据所有的key，去执行PMK计算方法生成这个key对应的PMK值。因此for循环内只有1行：

PMK.set(EncryptUtils.encryptPBKDF2(val.toString(), key.toString()));

其中EncryptUtils类中静态方法encryptPBKDF2(String Text, String key)根据迭代需要的文本和密钥就可以输出PMK。具体见（1）。

（4）主类

常规配置即可。注意如果需要分别保留不同SSID的字典，则在设置输出路径的名字时加一个特殊标记（比如”Dict\_”+SSID），这样会保留所有的字典，而不会出现以前生成过而被不同SSID的字典覆盖后需要重新生成的情况。