学习Linux运维时，难免遇到晦涩难懂的安全相关的话题，如在接触SSL、建立https安全通信等话题时，往往会遇到各种诸如hash、数字签名、证书、密钥等术语；本文并不打算深入研究，只是对一些基本术语以生动的图例稍作阐述让更多人跨过这道坎；我个人也仅仅是很浅地了解过这个话题，所以如果有不妥当之处，还望告知我及时更正

从最基本开始，让我们先认识3位密码学朋友

1.Alice 女主人公

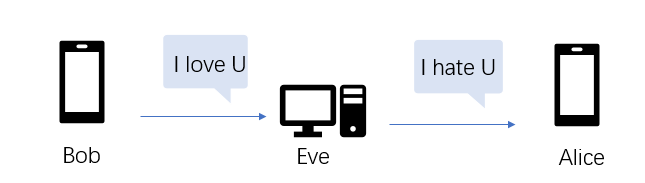
2.Bob 男主人公

3.Eve ——Bob的情敌（黑客）

情景1：数据明文传输



作为情敌的Eve很想了解这对小情侣互相发送着什么小秘密，于是黑客想尽一切办法用尽各种软件终于能够捕获到Bob发送给Alice的情话，此时，Bob心中咬牙切齿，脑中一阵邪念过后，Eve使用软件将Bob发送的消息更改为：I hate U；修改过后，在Eve心中一句邪恶的“嘿嘿”后，将报文转交给Alice，Alice看到消息，对Bob一阵破口大骂，与Bob断了关系；



此时Bob恍然回过神来，原来只是一次可怕的幻想，在Bob按下发送键前，他终于意识到数据在经过Internet传输过程有可能被诸如Eve的人对他羡慕嫉妒恨，于是Bob咨询了安全专家，决定对数据进行加密

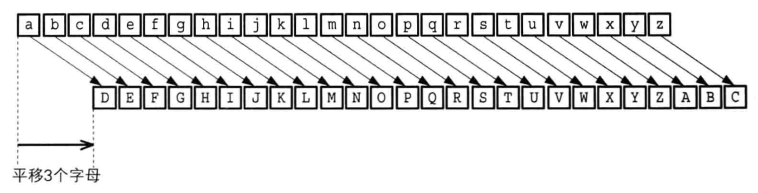
为了能让各位理解情景2中涉及的术语，这里不得不介绍几个名词

先举个具体例子好了：

有一种古老的加密算法叫做：凯撒密码

凯撒密码这样加密：把原本的字符串的每个字母按照26个字母顺序统统向右移x位

举例：将hello的每个字母向右移3位就将变成：khoor；这样一串字母即使被捕获，黑客可能也只是一头雾水



这里可以解释这些名词了

明文：hello

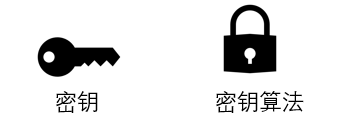
密文：khoor

密钥算法：凯撒密码，即按字母表顺序平移这个操作

密钥：3（平移3位）所以密钥一般就是一长串数字

密钥空间：一共有多少种密钥可用于加解密，这里是26（一共26种平移方式）

所以，拿生活中的例子来说的话（可能不太恰当），密钥就是钥匙，密钥算法就是锁，锁的种类有很多，而每种锁对应的密钥总数可能成千上万；





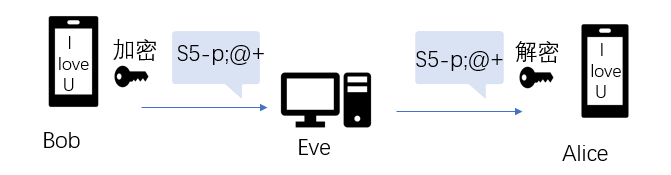
下面我总结了2点基本的密码学相关的论点，各位选看，不必着急理解

1.由于现代密钥算法的复杂性，想要试图“猜出”密钥，而后从密文转为明文是很难的（通常是通过暴力破解的方法）因为密钥空间往往非常庞大，如2^128、2^256这样的规模，但是务必记住，很难破解并不代表不能破解，利用能执行大量计算的机器能够在几天内甚至几小时完成暴力破解，而从用户的角度，你无须更换密钥算法，只需定期更换密钥就行了

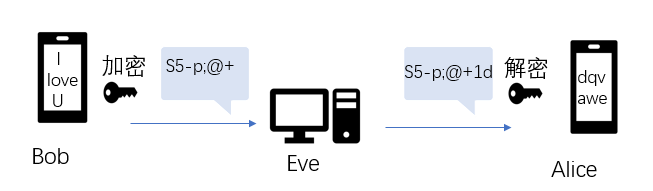
2.密钥算法是公开的，这样有利于安全（试想，你开发了一款自认为足够安全的密钥算法并藏起来偷偷使用，结果某天密钥泄露后，他人抓住该算法的漏洞并轻易计算出你的密钥，这将是得不偿失的）

以下先假设我们使用的是对称密钥，所谓对称密钥就是加解密使用相同密钥（钥匙）

情景2：数据密文传输



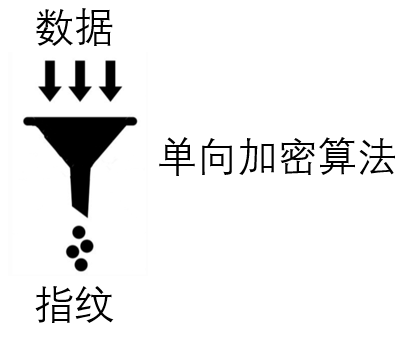
现在Eve拦截到Bob的消息后，一脸懵逼，完全看不懂Bob说了些什么，就在Eve准备放弃篡改数据时，一阵灵感涌来：既然我看不懂，我也不许Alice看懂！！于是Eve立马在原本的密文基础上添加了一堆“乱码”



Alice向Bob反馈这一情况后，Bob又去咨询安全专家，决定加入单向加密算法

单向加密算法：代表的有md5、sha1，以md5为例（这里使用32位的密文长度）

作用：md5经常与footprint一同提起，footprint表示唯一标识数据的指纹，它通常用于让接收方确保数据的完整性与正确性

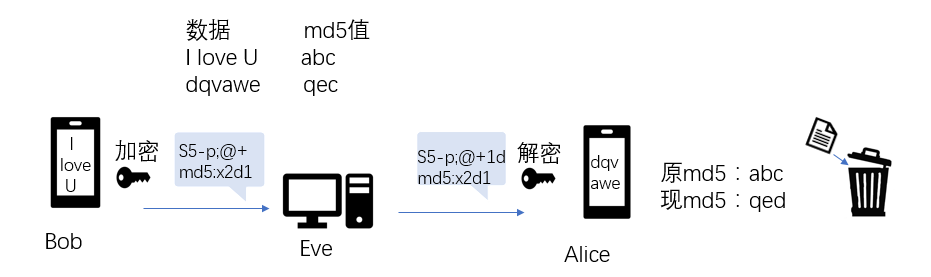


也就是说：任意的长度的数据最后将生成32位的定长密文字符串，该字符串是能唯一标识该数据的指纹

单向加密算法的特性：

1. 不等长输入，等长输出
2. 雪崩效应，即对原文做出了一丁点改动，对应的密文将发生翻天覆地的变化
3. 不可逆，即无法从密文反推出明文

情景3：密文传输+单向加密

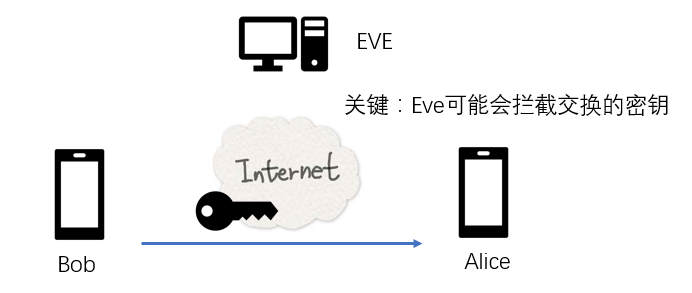


现在Bob在发送密文数据的同时一并将数据的md5值一并附上，假设Eve中途篡改了报文，现在Alice收到报文后，她将数据解密后并不会心急去查看消息内容，而是同样使用md5对解密后的明文数据计算md5值，并将结果和报文附上的md5值做比对，这样就知道了中途有人作怪，篡改了数据，Alice将数据丢弃并通知Bob重新发送数据

很好，我们继续下个话题

对称密钥从何而来？显然，上方讨论的都是双方事先约定好的预共享密钥，那么问题来了，在我们与对方素未谋面时又谈何“预共享”呢？所以，在使用对称加密时，我们往往面临一大重要难题：在Internet大环境下，双方如何安全地获取对方的密钥（使用对称密钥时，发送方需要用密钥加密，接收方需要用相同的密钥解密）

难题：



面对这个问题，我们稍后来谈

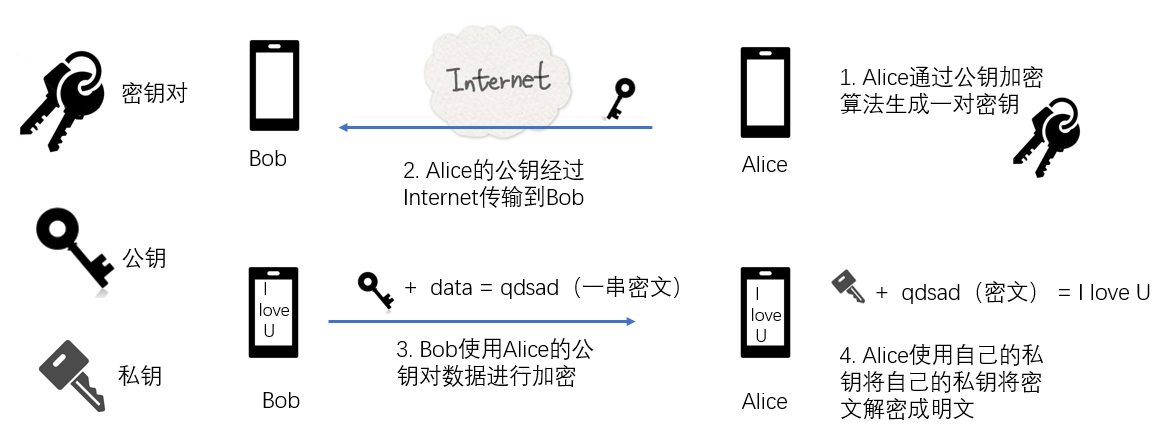
现在引入另外一类密钥算法——公钥加密算法（也叫非对称加密算法）

公钥算法的思想是：产生2把不同的密钥，其中使用公钥加密，而后使用私钥解密（当然反之亦可，具体使用看应用场合，下文会有阐述）

对于这两把密钥，通常私钥自己保留，同时向外分发公钥，如果使用公钥加密数据，那么情形是：

Alice将自己的公钥分发给Bob，Bob拿着Alice的公钥对数据进行加密，密文到达Alice后，Alice再用私钥对数据进行解密

情形4：使用公钥加密算法加密数据



现在，即使在密钥交换过程中Eve窃取了Alice的公钥，也无济于事，因为他没有Alice的私钥，也就无法对从Bob截取到的密文进行解密；这样一来，似乎很完善了嘛，密钥传递问题得到了解决，而且看似也很安全

不不不，问题还多着呢，一个个来，先来谈谈公钥加密算法的缺陷

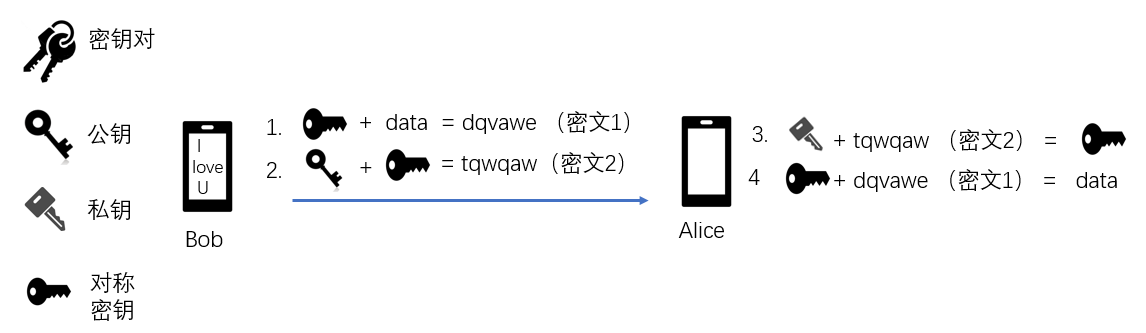
看完对称加密和非对称加密，你是不是觉得公钥加密相当地高大上，一对密钥中的2把密钥扮演了不同的角色（一把用于加密，一把用于解密）但是！！逼格往往是需要代价的，也就是说公钥加密有下面几个致命缺陷：（当然这些缺点就是对称加密的优点了）

1. 加密过程相当地慢
2. 加密后的数据相比原先数据而言非常大（如加密前数据为100MB，加密后数据可能剧增为200MB）

所以，一个词来概括公钥加密算法的缺陷：低效，而对称加密算法加密时是高效的——快且加密后数据量相比原来不会增大太多

额，现在处于两难境地了，对称加密算法没法做到安全交换密钥，而公钥加密大数据效率很低，那么是时候亮出大宝剑了，经典解决方案如下：

情形5：结合对称加密算法与公钥加密算法



Bob顺利拿到Alice的公钥后，发生的事情：

1. Bob用本地对称密钥算法生成的对称密钥对数据进行加密生成密文1
2. Bob用Alice的公钥对第一步中的对称密钥进行加密生成密文2
3. Alice收到2个密文后，先用自己的私钥对密文2进行解密以获取可以用于解密实际数据的对称密钥
4. Alice使用第三步中的对称密钥对密文1进行解密以获得真正的数据

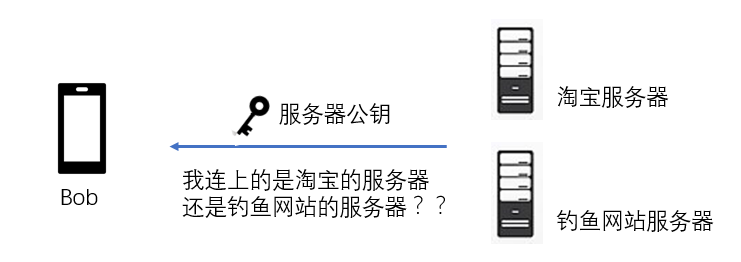
方案为何可行？

* 1. 对称密钥本身相对于实际数据而言很小，所以用公钥对大小很小的对称密钥进行加密并不会造成多大的损失
  2. 利用了公钥算法交换了实际用于加密数据的对称密钥，实际上还有其他的密钥交换方法，如DH算法，但本文不打算涉及，有兴趣的读者可以自行去了解

写得有点久了，就不卖关子了，直接看看其他不足的地方

难点：Alice如何确保数据就是Bob发来的呢（也就是说，和我正在通信的人真的就是Bob而不是他人冒充的吗）？

某天Bob兴高采烈地准备网上购物，却意识到一个安全隐患：



而数字签名正是针对此难点应运而生的技术，数字签名=数字证书=证书，以下可能混用这3个词

在探讨数字签名前，我们谈谈生活中的某个现象

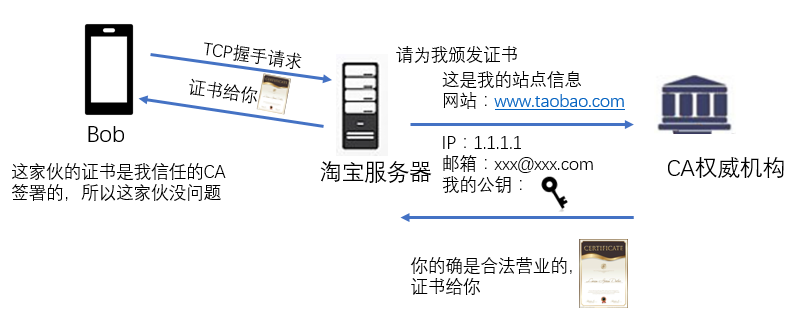
如今，大家可以通过自己的身份证来完成各种需要证明身份的活动，如考试、坐车等

那凭啥一张卡能让我们走天下让所有人一致地相信你就是真实的你呢？

原因很简单，身份证是公安机关签署的，而我们相信公安机关，也就相信公安机关发行的用于确认你身份的身份证；试想如果没有一个这样的权威机构的印记，你随意造出一张卡对外表面身份：我是某某某，相信没人会听你的鬼话

好了，下面再来谈数字签名的概念，应该就很好理解了

按照上面来类推，如果要让通信的双方能相信对方的身份，就得借助于一个权威机构，在数字证书的世界里，需要一个大家公认的第三方权威机构，专门用来为各个公司、机构颁发证书，我们称这些权威机构为CA；



具体过程为：

1. 淘宝服务器拿着自己的向CA提交个人信息与自己的公钥，请求证书
2. CA对淘宝服务器进行审核，审核通过后，用自己的私钥对淘宝服务器的公钥进行加密）
3. Bob与淘宝服务器完成TCP握手
4. 淘宝服务器向Bob回送证书自证身份
5. Bob发现证书是自己所信任的CA颁发的，所以也就信任了淘宝服务器，于是用CA的公钥对证书进行解密，而后提取其中包含的淘宝服务器的公钥
6. Bob用淘宝服务器的公钥完成后续的操作（参考情景5）

至此，基本完成，但仍遗留一个小问题：上述第5步中Bob需要使用CA的公钥对拿到的数字证书进行解密，但这就很尴尬了，因为在Internet环境下，CA想要安全将CA的公钥转交到Bob是很难的。所以，多数浏览器开发商发布版本时，会事先在内部植入常用认证机关的公开密钥，其实操作系统一般一会内置有值得信赖的CA的公钥

补充小知识：

上述我们看到的都是服务器给客户端发证书的例子

那有没有为客户端发证书的情形呢？

必须有，如下：

银行的网上银行就采用了客户端证书。在登录网银时不仅要求用户输入ID和密码，还会要求用户的客户端证书，也就是我们的网银

小结：由于连续撰写时间有点长了，可能后半部分内容稍稍不尽人意，还请各位谅解，当然基础密码学相关的内容还有很多，就不细说了

一句话：这篇文章是为后续的博文做铺垫（openssl、自建CA、ssl、https等）