

讲到用户密码加密的时候 ---- 数据的加密方式有三种

1. 对称加密
2. 非对称加密 也叫 公钥加密
3. 单向加密 也叫Hash加密

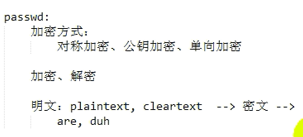
现在就是为了即便截获数据 + 知道如何转换的过程 也不能还原看懂

加密：明文 --- plain text 或者clear text ---- 加密前的内容 转换成 密文 ---- code

正常接到之后 要还原成clear text

早期的加密 通过移位得到了 比如 a 向后移三位 a->e d->h

接收方就是简单向前移三位 --- 非常容易破解 ---- 古罗马



军事上 加密解密是非常关键的

美国来讲 核心加密算法对外都是禁用的

早起最常见的加密 ---- 计算机诞生之后 就是对称加密

对称加密如何工作？

简单来讲 有一中加密算法 本身实现之后 有这样一种功效：

加密算法+ 临时给的密码串（也叫口令）去加工本来要输入的信息数据（明文数据）---- 通过各种各样的转换（严重依赖口令）------ 输出密文

这个算法很有可能所有人都知道 -----

加密机制的安全性不应该依赖于安全算法 ---- 否则加密算法就会失效

加密算法固然可以设计很复杂 但是 不能仅仅依赖于算法 还要依赖于灵活的口令

没有一种方式算法可以绕过口令来解密

所以 对称加密就遵循上面的原理

为什么还会被破解？现在的攻击 有一种枚举攻击 --- 暴力破解

如果口令过于简单 就可以被暴力破解 ---- 这就是字典攻击

所以设计主机口令的时候 不能用过于简单的字符串 比如生日之类的

因此一定不能用生日作为密码 --- 因为字典攻击都作为攻击的条目 这就是加密可以被人破解的原因

这种情况下 对方并不是攻克了加密算法 如果枚举速度足够快 所以 就能被攻破

一秒钟一亿次

密码20位 键盘上加入100个可以用于输入密码的按键

100^20 ---- 10^40 意味着 最坏情况：需要10^32s的时间可以破解出来 （速度是1s可以计算一亿次）---- 量子计算机 是可以做到的

可以不明白你的算法是什么 暴力破解就好了

目前来讲 ---- 加密技术来讲 民用 商用的加密算法 攻克还是有点难的

但是 两个组织 组织对抗 这个有可能

只要密码足够复杂 破解很难

这个加密算法的安全与否 就严重依赖于口令

对称加密：加密方 解密方使用同一个口令

目前来讲 能实现对称加密的算法

DES ----- Data Encryption Standard ---- 数据加密标准

2000左右 美国边境委员会 使用了一台价值100万美元的电脑 并 在3小时破解了所有的DES ------ 所以 就出现了AES

AES ----- Advanced Encrypt Standard ------ 高级加密标准

DES ---- 使用了56位秘钥 ------- AES使用了128位秘钥

AES出现了很多变种 ----- AES192 AES256 AES384 后面的数字是秘钥的长度

AES是国际上的对称加密机制

DES被破解之后 人们还想出了 3DES ------ 加密3轮



3DES虽然被广泛使用 ------ 理论上不安全

所以 使用的还是AES

还有著名的对称加密算法 Blowfish, Twofish, iDEA(商业加密算法 使用付费), RC6, CAST5, Serpent

【



所以 单向加密 实际上是一种hash算法 不是加密算法

Base64 crc32是编码方式 不属于加密的范畴

Base64实际上就是byte[] -> String --- encode

Base64的decode就是 String->byte[]

对于 加密encryptor 就有解密 decryptor

对于 编码 就有解码 encoder decoder

】

【看一下我们的360s和getSolution 请求 和 返回 ---- 具体怎么加密 解密 都是业务那边要求的 和吕品对一对】

对于对称加密 ----- 就是先将原文按照固定大小切分成块 block ---- 对切分的块进行加密 ------ 因为逐字符加密速度太慢

加密时候 第一个块加密 ---- 然后第二个块加密 --- 第一个块的加密和第二个块的加密做异或运算 作为第二个块的输出

所以 每拿到一个块 要依赖于前一个块 做异或运算 ----- 块链 block chain

对称加密算法的特性：

(1). 加密 解密使用同一个口令

(2). 加密算法 将原文分隔成固定大小的数据块 然后对这些块进行加密 ----- 加密之后形成块链 ---- 内部用到某些算法有 ECD, CBC等

对称加密的缺陷：数据加密了 传递给了对方 对方需要依赖算法和口令进行解密 ---- 算法可以公开出去 但是口令呢？口令也很可能被截获了 就完蛋了

数据送过去了 对方如何得到口令？发邮件 打电话 怎么才能可靠？

必须事先约定好密码 ----- 如果见面 ---- 这样严重依赖于人 这样依赖于人的非常不靠谱（比如 假冒对方公司的社工 ----- 这就不靠谱了）----- 加密环节中 最薄弱的环节就是人

🡺密码或者口令传输成为了麻烦/问题

如果 要对接的对象很多 很多人 用户 密码都非常多

管理起来非常麻烦

---- 对称加密解决不了的就是 太多的通信对象导致过多的密码维护 ---- 同一个密码也会导致出现 一个泄露 整个就完蛋了

对称加密的缺点：



对称加密解决不了的问题：（上面是解决了 但是 不如意 ）

密钥传输：一个此前从来没有见面的人 ---- 怎么给你 ----就要面临秘钥交换的问题

用户认证：如何验证你就是我要通信的人？

数据完整性：数据被截获了 --- 但是 看不懂 解密不了 就随便加了其他的字符信息 -----数据被篡改了 ---- 这样真正获取到数据之后 被解密 数据不对了

此前双方没见过面 如何把数据给你？---- 见到面了 如何保证就是要给你？

--------- 非对称加密可以完成密钥交换（但是不是最好） -------

通过一种就是密钥不用传输 对方就能获取到 ---- 就是著名的DH算法



双方对照密钥的时候 没有发送任何密钥的信息 就是这样

Bob和Alice分别本地生成一段各自的字符串 ----- 然后传输到对方 ------ 对方把接收到的各自的字符串分别再次运算之后 双方的计算结果的串是一模一样的 ----- 但是 任何人截获了这个串之后 都无法获取和密钥有关的信息

密钥交换中 有一个简称 就是 IKE ---- internet key exchange

DH是IKE中的一种

通信双方要完成密钥交换 要有DH算法的软件 ---也麻烦

-------- 非对称加密 ------

非对称加密要解决对称加密的问题

对称加密面临一个问题 就是一个人要维护多个密码

公钥可以公开 私钥只有本地才可用

想和别人通信 把接受者的公钥获取过来 ------ 用这个公钥进行加密

每个人记忆自己的一对密钥就行 用到谁 就跟谁要

公钥加密的密码强度 早起是512位 现在是1024位 1024非常常用

公钥加密的速度比对称加密的速度慢了差不多3个数量级 1000倍

如果对称加密20M的数据 要1s 那么使用非对称加密公钥 就差不多1000s ---- 20多分钟

\*\*很少拿公钥加密来加密数据

现在Bob和Alice现在要在网络上面传输一段信息 但是 想保证信息的私密性 注意 Alice和Bob这两个人之前从未见过面

Bob本地生成一个对称密码 + 选一个对称加密算法（这是因为对称加密的速度比较快 ---- 数据的量相对密钥要大 所以 这个时候 为了提高加密的效率 明智的做法就是使用对称加密 ----- 所以 正好适合了前面说的“很少拿公钥来加密数据”）=====> 这样 **要传输的数据被加密了**

为了接收端解密对称加密的数据 ----- 要求 接受端必须拥有对称加密的密钥 ---- 这样如果直接传递密钥的明文 风险就会非常大！ ----- 所以 关键就是 用非对称加密对对称加密的密钥进行加密 ------- 那么Bob就要事先获取Alice那边的公钥（因为最后一定是Alice来解密）---- Bob用了Alice那边的公钥加密对称加密的密码之后 =====> **要传输的对称加密的密码也被公钥进行了非对称的加密了**

---- 互联网上传输的**数据(仅仅是被对称加密 ---- 数据量大 对称加密效率高)** + **对称加密的密钥(进行了非对称的加密)**都是**加密**的

----- 如果被截获 ------ 截获方没有私钥 所以 无法解密获取到公钥加密的对称加密的密钥的明文 ---- 这样传输的数据就无法被破解【思路就是 我要提高加密的效率 那就对称加密 但是对称加密的密钥不安全 那我就让他安全 所以 对称加密的公钥使用非对称加密】

Alice对接收到这个数据之后 先用自己的私钥解密加密的对称加密的密钥 ------ **对称加密的密钥获取到**

------ 再使用这个对称加密的密钥去解密使用对称加密的数据 ------- **对称加密的数据获取到**

【个人认为 编码和加密的区别 就是 编码是算法 不一定可逆 没有密钥 --- 也就是没有个性化的东西 但是 加密需要个性化的东西 对于对称加密 就是密钥 但是 对于非对称加密 就是公钥和私钥

至于单向加密 为什么不能叫单向编码呢？因为编码 就一定有解码 但是 MD5不可逆 所以不能解码 所以 只能是单向编码 ----- 变成了单向加密

】

---- 刚才的公钥加密的密码在互联网上传输了 有潜在的风险 实现了密码的交换

但是DH在互联网上没有传输密码 都是在本地计算出来

密钥交换算法中：公钥加密的密钥交换机制 需要在互联网上传输密码 但是 对称加密的DH算法不需要 所以 就密钥交换机制来讲 对称加密DH的安全性高

--------

非对称加密是可以实现密钥交换 第二个重要的功能就是实现**用户认证**：

不是**用户密码认证** 而是**用户身份认证**

场景：Eve想要冒充Bob给Alice发送信件 ---- 所以 首先Bob从Alice那里获取到了Alice的公钥 ----- 然后按照上面的做法 以Bob的身份给Alice发送了一份伪造的邮件 ----- 结果 Alice使用同样的方式 解密得到的信息 ---- 误认为这是Bob发送的 ===== 没有身份认证的结果

所以Alice必须验证发信人必须是Bob 而不是Eve

比如 某人冒充tom登录淘宝 使用了tom的用户和密码 这样身份就错了

怎么发送订单的时候 就知道对方就是这个人？怎么进行身份认证？

对于公钥加密：

**实现身份认证** ---- 每个人都有一对密钥 ---- 如果我发送一段数据 使用了我的私钥进行加密 ---- 那么 其他人都可以获得我的公钥 那就都能解密 ---- 但是注意 这里面发送数据不是目的 而是 能够解密才是目的 能正确解密 说明别人获取了我的公钥 ----- 解密的一定是我的私钥加密的内容 ---- 我的私钥别人都没有 ---- 所以 验证的是 这段信息一定是我发的 ==== 还是面临着 我的非对称加密解密的速度不够快 -----（**身份认证 是 使用私钥加密 公钥解密 和前面的加密信息的思路相反**）

现在的身份认证结合了第三种算法就是单向加密 ----- 单向加密的另一个特点就是能够保证数据的完整性

思路就是 单向加密算法可以获取发送数据的指纹信息 --- 指纹信息是数据完整性的

----然后 就是 如果我的数据被篡改了 对方接收到篡改的信息 计算md5 就和发送方给的md5不一样 就知道这个不可信了

===== 同样存在的问题是：如何把发送方给出的**信息的指纹**发送给接收方？

===== ${message}^${md5} 以这种格式发送出去 ----- 但是 被截获之后 ${md5}被丢弃 ---- ${message}被篡改 重新计算出md5 ---- ${messageNew}^${md5New} 这样接收方收到之后 还是没有办法验证出来 ------

======= 所以 上面的问题就是 我这个md5可以被重新计算 ----- 那能不能让我这个md5计算出来的结果是对我来讲是唯一的呢？ ---- 可以 那就是发送方使用自己的私钥对md5进行加密 ----- 这样 一旦数据被截获了 篡改了 要重新计算这个md5 ---- 计算出来之后 篡改方没有办法获取发送方的私钥进行加密 ----- 只能使用篡改方自己的私钥加密 ---- 这样就再发给接收方 但是 注意 发送方会把自己的公钥发送给接收方 ---- 这样一旦接收方获取到篡改的内容之后 首先使用发送方的公钥去解密被篡改的md5的密文 ----- 解密出来之后 篡改的md5

再把篡改的内容解密出来 重新计算md5 ---- 这两个md5进行比较 一定是不一样的 这样 这个信息不可信【md5很短 所以非对称加密很快】

===== 这样发送的整个信息的格式：

**${明文的数据}^${用于进行身份验证的明文数据的md5非对称加密的密文(发送方的私钥加密)}**

----- 实现了**身份认证**（bob的一定是bob）[这个是发送方私钥加密起的作用]

----- 也实现了**数据的完整性**（数据如果被篡改 会被发现）[这个是md5具有信息完整性起的作用] ---- 可以校验数据完整性

==== 但是 **私密性能保证么？ ----- 不能 因为单向加密把特征码发过去了 但是 特征码不可逆 所以 无论怎么样 原文一定要发送 也就是明文 ---- 私密性不能保证！！！**

现在 想 私密性+完整性+身份认证 都要保证 怎么办？------ 因为传递的是明文 所以数据的私密性没有了【需要发送数据私密性的原因 是 我如果发送的内容被截获之后 对方虽然无法在信息上面搞破坏 但是 他可以知道我们之间的聊天内容 这样 我们的行动的方案就会暴露 第三方可以在非信息方面进行其他的破坏 所以 要保证发送数据的私密性】

首先有一个原文 ---- Bob实现使用单向加密算法 抽取数据的特征码 然后Bob为了做到身份验证 会使用自己的私钥单向加密这段特征码 并附加在这段数据原文的后面

 ----- 绿色的是数据原文 红色的是使用bob的私钥进行非对称加密过的原文数据的md5 ---- md5特征码本身可以保证数据的完整性 加密的md5不能被篡改【如果新计算出来的md5只能使用篡改方自己的私钥进行加密 这样 接收方使用发送方的公钥进行解密 这样 不是同一对公钥私钥 解密就会出现异常错误】 保证了可以进行身份认证 ---- 之后bob找一个密码 把加密之后的md5+数据原文进行对称加密 ----

 ---- 对称加密算法快 ----- 这样明文数据（绿色）+ 发送放进行私钥加密的明文数据的MD5（小块红色）---- 合并到一起 外面再使用对称加密（大块红色）

 ------ 再使用**对方的公钥**把对称加密的密钥进行对称加密（黑颜色的）----🡪 这样 就可以进行传输了

图中Eve截获到这条数据

----- 数据和密码都加密 但是 Eve没有接收方的私钥 所以 对称加密的密码不能解析 这样 使用对称加密的密钥也不能被解密 ----- 这样对称加密的原文和加密的md5都不能被破解 --- 私密性满足 ---- Alice收到数据数据之后 ---- 先用自己的私钥解密这段对称加密的密码 --- 得到密码和数据原文之后 使用发送方的公钥解密这段加密的md5 ----- 然后重新计算原文的md5 是否一样 一样 认证通过 数据正确 否则 数据不接受

**${对称加密的(${原文数据}^${非对称加密的对称加密的密钥(接收方的公钥加密)})}^${用于进行身份验证的明文数据的md5非对称加密的密文(发送方的私钥加密)}**

对称加密算法 非对称加密 单向加密都用到了 ----- 这里面的公钥加密算法举足轻重 ----- 因为公钥加密之后 所有的数据都无法解开

非对称加密算法：RSA EIGmail DSA(美国国家安全署 只能实现身份认证 不能实现数据加密)

以上是非对称加密的算法

单向加密 ----- 实际上不是加密 是抽取数据特征码 ----算法 md5 MD4 SHA1 SHA512

CRC32 循环冗余校验码(可以抽取特征码 本身速度很快 但是 不能保证完整性 ---- 所以在数据通信中 只能验证 数据包是否被改变了 不能作为加密功能)

--- 如果不需要数据私密 也不需要身份认证 就需要验证数据完整性 ---- 怎么办？有一种对单向加密的特殊 ---- 如何把数据的特征码传输呢？

有一种算法 叫**消息认证**算法 ----- 仅仅验证消息本身 CBC-MAC 和 HMAC 使用某种内生机制来加密这段特征码 中间的人获取到这段加密的特征码之后 也无从解密这段特征码【不进行详细展开】 ---- 仅仅认证消息本身是否靠谱

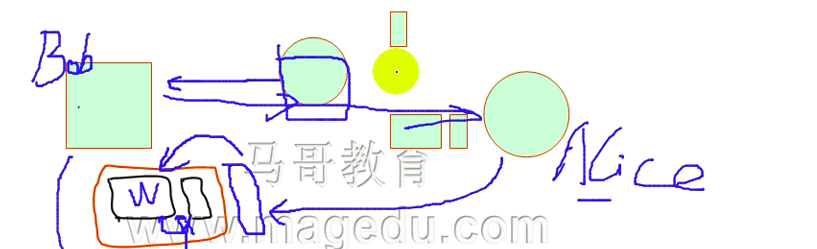
**MAC ---message authentication codes消息认证码**

所谓单向加密有一种特性 --- 雪崩效应 + 定长输出

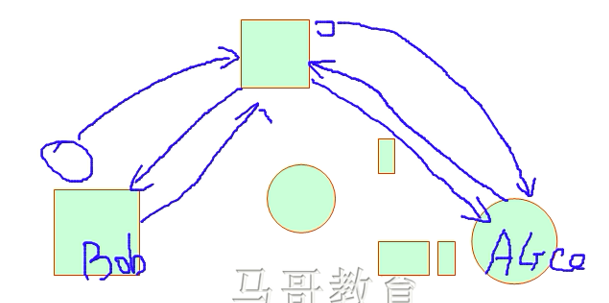
========= 上面环节的漏洞 =================

前面的过程中 重要的环节 就是 Bob需要Alice传输Alice的公钥 给Bob 这个过程 经过Eve的家门

现在Bob发给Alice的数据经过了Eve的家门 Eve伪装成Alice 把自己的公钥发给了Bob 然后 Eve伪装成Bob跟Alice获取了Alice的公钥



这样 bob写完信 发给了Alice --- Eve截获之后 使用自己的私钥解密了对称密钥 然后使用对称密钥解密了原始数据 看完了原文 ---- 然后 Eve修改了原文内容 并使用自己的公钥加密了对称密码 重新计算新的内容的md5 ----- 一并发给了Alice ---- Alice使用了Eve的公钥（她认为这个是Bob的公钥）进行了解密 重新判断 发现md5吻合 但是 这样数据已经泄露了



**这样漏洞非常大**

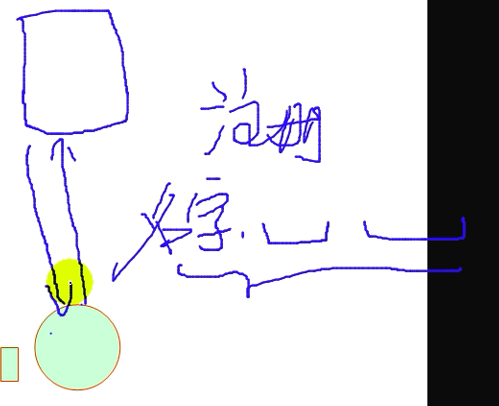
这个环节问题是 Bob跟Alice要了公钥 并且接收到之后 **Bob不得不相信这个公钥就是Alice发送的 ---这个是漏洞的关键【可能是这个公钥是别人假冒Alice给的 但是 就说 我是Alice 给你的公钥就是Alice的公钥】**

比如到某家公司查营业执照 出示之后 你会相信 **因为相信发证的机构**

----- **所以 找一个互相都相信的第三方机构 这个机构负责发证**

Alice为了和别人通信 **并且被新人** 就到这个组织去注册 注册需要提交的是 **第三方机构要进行审核 ------ 名字^其他^公钥 -----第三方机构把这个信息包装一下 就返回给了Alice**

**[名字和公钥绑定 说明 这个名字的公钥是被公正了]**



证书是否可能被伪造？有可能 要使用防伪标识 但是 对于数字的东西怎么防伪？

同样的方法

**第三方机构有自己的证 他的证谁来的**？--- 也就是要验证第三方用户的本身的可靠 ----- 自身私钥加密 ---- 把第三方自己的公钥传输给用户

发证的时候 会计算 这段数据的特征码 ----- **并且把这个特征码 使用第三方自己的私钥进行加密**  **一起发给用户** ---- 同时第三方机构也把自己的公钥发送给用户

======== 具体说一下 就是 **ca\_prv\_encrypt(抽取${user\_name}^${other\_info}^${user\_pub}的特征)** + ca\_pub 这两个信息返回给用户

-------- 这样用户一旦能解密 就说明这个**证书**是这个机构发的 ----- 证书来源靠谱

电子签名：这样 通过**自己的私钥**加密**普通用户的特征码**的这段信息 叫做**电子签名** ---因为是用来做身份认证的

所以 有了这段验证 发证的来源靠谱了

之后 Bob和Alice通信的时候 Alice直接把证书发给Bob

Bob获取数据之前 会到第三方结构获取他的公钥【Bob自己获取公证中心的公钥 不是Alice给的 这样靠谱】 然后解密这段签字签名 解密成功 说明数据来源可靠 ---- 这就意味着第三方机构认可的这个就是Alice ------ Bob认为这个就是Alice

这个时候Eve还是可以冒充这个第三方机构的 ----- Bob要用第三方的公钥解密 怎么获取？ 这个过程Eve还可以冒充 我就是第三方 这样就完蛋了

如何验证第三方？跑到当地 并且多方打听 但是 对于用户这种 代价太大

--- 所以选择相信你 出问题 自认倒霉

--- OS制造者 会内置全球知名的发证机构

Windows已经内置了全球很多知名的发证机构

**这个第三方机构 就叫证书颁发机构 CA ----- Certificate Association**

证书还有一个有效期 ---- 因为证书颁发机构 证书使用费 少则几万 多则上百万

----- 如果自己内部公司发证 怎么办？可以自己发证

在有限范围内 自建证书

找全球知名的CA ----- 会很贵 这些用户比如 淘宝 京东这样的网站

证书获取之后 --- 服务器被攻破了 私钥被盗用了 这个时候 证就作废了 ---- 证书需要吊销

所以Bob接收到Alice的证书之后 第一要验证证书是否可靠 并且 还要验证这个证书是否被吊销

**所以 这里面 有公钥加密 私钥加密 对称加密 CA吊销证书 形成一种技术规范 叫做PKI ---- 公钥基础设施**

**Public key Infrastructure --- PKI**

加密算法 PKI CA

全球的CA有很多个

比如 通信的双方信任的CA不一样

这样 怎么办？

如果这两个机构可以互信 这就没问题

虽然互信 但是有汇率

根CA 随便认可 要收费

**真正负责完成这些功能的软件有两个 一个是openssl 另一个是gpg**