Digest And RSA Explain

[1:总纲: 1](#_Toc487818762)

[2:参考文档: 2](#_Toc487818763)

[3 :理解数字签名，数字证书的使用场景 2](#_Toc487818764)

[4:对称加密的种类: 4](#_Toc487818765)

[4.1: DES 4](#_Toc487818766)

[4.2 :TripleDES 4](#_Toc487818767)

[4.3: RC2 5](#_Toc487818768)

[4.4: Rijndae 5](#_Toc487818769)

[5.0: Hash 算法种类 5](#_Toc487818770)

[5.1:BaseCode64 5](#_Toc487818771)

[5.2:MD5 5](#_Toc487818772)

[5.3: SHA256 5](#_Toc487818773)

[5.4:HMAC 6](#_Toc487818774)

[6.0:RSA 6](#_Toc487818775)

# 1:总纲:

阐述数字签名，数字证书，RSA加密，对称加密，Hash算法的使用场景。

算法用到了，位移置换，替换，移位，异或等算法。

# 2:参考文档:

<http://justjavac.iteye.com/blog/1144151>

# 3 :理解数字签名，数字证书的使用场景

举例：bob 与susan 通信，如何保证内容的不被第三个人看见读取呢？

如果是正常明文，每个人都可以读取，怎么办？

可以为其分配一个账号，通过账号的方式，登录的方式来登录进去正常通信。

但是，在数据交换的过程中，会造成数据的拦截？

用什么方式可以拦截到互相通信的信息呢？现在正如web传输的过程中用Http协议，都是明文传输的，可以用钓鱼网站。黑客种植在服务器上，用监听工具来监听发送的请求，来传输给伪造的第三方数据中心，然后来破译。内部人员部署监听工具，来监听交易的数据信息。

所以造成了传输过程中的不安全？

那么怎么办？

需要把传输来加密？

正常的都是用到对称加密技术，把数据进行加密传输，对称加密有个限制条件就是，加密与解密的密钥key都是一致的，必然会引起如果，密钥被暴露，将导致信息泄露，造成不安全。

这个时候就需要用非对称加密技术，产生私钥与公钥对，私钥有自己存储，公钥付给第三方。

这样就可以维护信息的安全。Rsa技术。

这样就可以用Bob用公钥书写，susan 用私钥来读取。

如果此过程中，万一被劫持了，因为是加密的，数据也是不可以被读取的？

那我如何来保证数据在传输的过程中有没有被篡改呢？

万一被私钥泄露了，中间被篡改了，或者没有被篡改，随便添加了点内容，susuan就打不开了，她根本就不知道已经被在传输的途中被篡改了，那怎么？

这个时候就需要数字摘要技术了？

为什么要用到数字摘要呢？因为数字摘要中有一个很著名的效应 雪崩效应，牵一发而动全身的技术，如果更改就会引起签名的不一致，susan在接受的时候可以对 数字摘要进行比对，看看是否在传输的过程中是否被篡改哈。

还有一个原因，因为数字摘要技术，用到的是hash算法，hash算法本身产生的是一个定长的位数。简单，效率高，本身也可以起到加密的性质，hash算法有Md5，sha1，sha256，Hmac（提供密码的方式进行hash的）。

因为数字摘要是可以破解的，产生了数字签名，因为不可能用rsa来加密大段的文字，效率低，所以针对这个数字摘要做了，一个签名，即:数字摘要+rsa算法。

首先对文件进行摘要，然后用rsa算法私钥签名，对方获取用公钥进行验证，是否是一致的。

这样就可以确定是Bob给我在通信了，可以对他进行正常的交易了。在交易的过程中在对交易的信息进行对称加密技术，进行加密。在每次传输的过程中都用上数字签名技术，带上数字摘要，来保证每一次传输的内容没有被篡改。

那为什么又产生了数字证书呢？

假如，读取信息的一方的公钥被替换了呢？是不是就可以第三方用自己的私钥来伪造数据信息了呢？

因为我保证不了我的公钥不被泄露和替换。

假如私钥里面存储的有信息要提交的地方，公钥读取了信息，是不是就把数据提交到了不应该提交的地方了。

这个时候，如果有一个人可以证明Bob是Bob就可以了，这个是就产生了证书，中数字记录了Bob的公钥，并可以证明Bob就是Bob。

特别是交易的过程中，我需要证明，第三方网站是否是安全的，不然我不放心与他进行交易哈。

所以服务器存储私钥，把其公钥证明自己的东西，用第三方证书的形式来，存储起来，然后用数字签名技术，来产生交易。

服务器，用私钥加密，附带本身的数字证书一起发送给客户端，客户端需要在本地的浏览器证书管理里面信任此证书，以后就可以正常交易了。

这个就是Https协议：证书+SSL。

后面在专门针对https做一个章节的研究。

至于数字证书里面到底存取何种内容呢？

数字证书在那些地方申领呢？如何制作呢？

# 4:对称加密的种类:

参考文档：<http://www.cnblogs.com/fujinliang/archive/2012/07/03/2574091.html>

.net 里面的对称加密技术用到了向量与模式。

对称加密的源码实例提供在项目中

## 4.1: DES

Des算法的原理：<http://blog.chinaunix.net/uid-29106641-id-4032988.html>

DES使用的密钥key为8字节，初始向量IV也是8字节。以8字节一块进行加密。

原始的des是没有初始化向量的，为什么要提供初始化向量呢

这个就涉及到，如果传输大量的数据，因为密码是不需要动态改变的，改变起来不方便。

这个时候用向量，既可以加深des的复杂度，这个其中涉及到四种模式。CipherMode的

四种模式，<http://cakin24.iteye.com/blog/2343834>

## 4.2 :TripleDES

TripleDES使用24字节的key 192位，初始向量IV也是8字节。以8字节一块进行加密。

其原理是与Des一样的，只是加密的Key的位数更多，也更安全。

## 4.3: RC2

Rc2 与des算法类似,密钥的16位，128字节,向量位8字节,密钥是从48-128 位变长的，但是以8的倍数..

这种密钥是变长的。

## 4.4: Rijndae

密钥长度有3种选择：128位、192位及256位。 大于128 32 的倍数,向量16字节 128位,AES（Advanced Encryption Standard）。

这种算法效率跟高，也更安全。

# 5.0: Hash 算法种类

## 5.1:BaseCode64

Base64 每24位 转换为32 位,不够补充为0,简单方便。

使用用途可以对资源定位符进行转换。网上说的对于资源定位符进行转换的时候关键字/与+会产生特殊字符%，我人为不对。进过测试。不知是我没有理解还是他们说的就是错的。

## 5.2:MD5

MD5 生成16字节的摘要，128bit,转换为X（16进制）2保留2位每次 ,所以生成为32 的字符串,如果想更加安全针对每个用户添加salt 加盐(md5(userName+salt)),解密的时候取出盐来比较哈，盐存放的位置，在数据库里面哈.

## 5.3: SHA256

Sha256 是sha1的升级版，生成的是256位hash，32字节，输入的报文不超过 2^64 bit。

## 5.4:HMAC

HMAC 是多添加了加密key在hash算法的基础上

生成的hash签名的位数跟具体的hash算法一致

1. 客户端发出登录请求（假设是浏览器的GET请求）

2. 服务器返回一个随机值，并在会话中记录这个随机值

3. 客户端将该随机值作为密钥，用户密码进行hmac运算，然后提交给服务器

4. 服务器读取用户数据库中的用户密码和步骤2中发送的随机值做与客户端一样的

hmac运算，然后与用户发送的结果比较，如果结果一致则验证用户合法

# 6.0:RSA

Rsa 参考文档 <http://bank.hexun.com/2009-06-24/118958531.html>

RSA的算法涉及三个参数，n、e1、e2。

其中，n是两个大质数p、q的积，n的二进制表示时所占用的位数，就是所谓的密钥长度。

e1和e2是一对相关的值，e1可以任意取，但要求e1与(p-1)\* (q-1)互质；再选择e2，要求(e2\* e1)mod((p-1)\* (q-1))=1。

（n，e1),(n，e2)就是密钥对。其中(n，e1)为公钥，(n，e2)为私钥。

KU =(e,n)=(3,33) KR =(d,n)=(7,33)。

明文：11，05，25 密文： 11 26 16

c=M的e次方%n

M=C的d次方%n

# 7.0::数字证书

## 7.1:数字证书类型

数字证书有个人数子证书：个人数字证书：符合X.509标准的数字安全证书，证书中包含个人身份信息和个人的公钥，用于标识证书持有人的个人身份。**数字证书**及对应的IC卡或私钥存储于ikey中，用于个人在网上进行合同签定、定单、录入审核、操作权限、支付信息等活动中标明身份。

机构数字证书：个人签名证书：符合X.509标准的数字安全证书，证书中包含个人的签名私钥与个人身份信息，用于标识证书持有人的个人身份。签名私钥存储于Keynet卡中，用于个人在网上进行合同签定、定单、录入审核、操作权限、支付信息等活动中标明身份。

## 7.2:X509证书

参考文档：<https://baike.baidu.com/item/x509/1240109?fr=aladdin>