**加密技术简介**

1 加解密简介

加密的目的就是，不能让发送方和接收方以外的其他第三方明白两者之间传送的数据的意义。举个例子：上课，小明给小红传纸条，中间会途径小王， 那么这个传送的过程中，不能保证小王是否会偷看这个纸条的内容。但是小明想这个纸条只能是我和小红看不想给别人看。于是他跟小红在课下定了一个文字规则， 纸条上的“哈哈”，代表“我”， “呵呵”代表“你”，“老狐狸”代表“老师”， “\*\*”代表“答案”等等。。于是乎他俩传纸条，传出去的字都是经过这种规则去写出来的。小王即使打开了纸条，对于一个不懂文字规则的人简直就是天书。更别提获取啥有价值的八卦信息了。但是小红收到了纸条之后，按照跟小明定好的规则进行还原。就能看明白纸条到底写了啥。

其实这样一个场景，模拟的正是加密和解密的过程。互联网对数据的加解密，大致也就这样的套路。只不过，双方之间加密的规则肯定是要比例子中要复杂的多。

任何一个加密系统至未少包括以下四个部分组成：

1 未加密的报文，称为铭文

2 加密后的报文， 称为密文

3 加密解密的设备或者算法

4 加密解密的秘钥

发送方用加密秘钥，通过加密的设备或者算法，将信息加密后发送出去。接收方在接收到秘闻之后，用解密的秘钥进行密文解密，将密文恢复为铭文。如果传输途中有人窃取，他只能得到无法理解的密文，从而对信息起到保护作用。

2 加密技术分类

加密技术的分类按照标准不同往往有不同的分类。这里只探讨以秘钥为标准划分的分类

对称式加密 收发双方使用相同秘钥的密码，叫做对称式加密。传统的密码都是此类。

非对称式加密 收发双方使用不同秘钥的密码，叫做非对称式加密。如现代密码中的公共秘钥密码就是此类。

3 常见的加密技术介绍

参考文章： 浅谈加密技术

加密技术分为两个元素， 算法和秘钥。算法是将普通的文本或者是可以理解的信息，与一串数字(秘钥)相结合，产生不可理解的密文的步骤。秘钥是用来对数据进行编码和解码的一种参数。在安全保密中，可以通过适当的秘钥加密技术和管理机制来保证网络的信息通信安全。秘钥加密的技术密码分为对称密钥和费对称密钥两种。相应的，对数据加密的技术也分为两类。即对称加密（私人秘钥加密）和非对称加密（公开密钥加密）。

对称加密以数据加密标准DES（Data Encryption Standard）算法为代表。

非对称加密以RSA（Rivest Shamir Adleman）算法为代表。

对称加密的加密秘钥和解密密钥相同，而非对称加密的加密秘钥和解密密钥不同并且其中一把钥匙（公钥）可以公开，而另一把钥匙（私钥）需要自己保密。

3.1对称密码(置换和代换)：

如图：

X：为明文，算法的输入

Y: 通过密钥K，加密后的密文

K：为密钥

对称密码（单密密码或传统密码）:发送方和接收方使用相同的密钥

步骤:

1,加密算法根据输入信息X和密钥K,生成密文Y (具体的函数由密钥K的值决定)

其中要通信的双方持有相同的密钥,通过安全通道传递.

2,到达目的地后,通过密钥K解密即可得到明文.

使用最广泛的加密体制是数据加密标准(DES-Data Encryption Standard)使用64位分组长度和56为密钥长度.其他常见的对称加密算法:三重DES,高级加密标准(AES-AdvancedEncryption Standard),Blowfish算法,RC5算法.

缺点:

通信双方事先对密钥达成一致协议在.现实中,未见过面的双方使用应用对称密钥密钥进行通信.解决方案-使用双方都信任的密钥分发中心KDC(keyDistribution Center).

3.2公钥密码(基于数学函数的算法)：

如图：

步骤:

1 接收方乙生成一对秘钥，即公钥和私钥。然后将公钥发布出来。例如给了甲丙丁这几个人公钥

2 甲，丙，丁，给接收方乙发送数据，都用公钥对这段数据进行加密

3 乙收到消息后,用自己的私钥对消息解密.

对于公钥加密,通信实体必须先交换公共密钥.一个用于可以通过很多方式公布它的公共密钥,例 如在个人网 页上,但是这种简单的方式可能会遭到第三方的主动攻击,解决办法-认证中心CA(CertificationAuthority）

相关算法–RSA：

第一个既能用于数据加密也能用于数据签名的算法.以发明者的名字命名,采用的分组密码。其他的公钥加密 算法:ElGamal算法(数据加密,数据签名),背包加密算法

3.3对称性加密与非对称加密的优缺点比较

3.3.1对称性加密优缺点：

优点：加密速度快保密度高。

缺点： 1.密钥是保密通信的关键，发信方必须安全、妥善的把密钥送到收信方，不能泄露其内容，密钥的传输必须安全，如何才能把密钥安全送到收信方是对称加密体制的突出问题。

2. n个合作者，就需要n个不同的密钥，如果n个人两两通信需要密钥数量n(n-1)，使得密钥的分发复杂。即，每对用户每次使用对称加密算法时，都需要使用其他人不知道的惟一钥匙，这会使得发收信双方所拥有的钥匙数量成几何级数增长，密钥管理成为用户的负担。

3.通信双方必须统一密钥，才能发送保密信息，如果双方不相识，这就无法向对方发送秘密信息了。

4.难以解决电子商务系统中的数字签名认证问题。对开放的计算机网络，存在着安全隐患，不适合网络邮件加密需要。

3.3.2非对称加密体制的优缺点:

优点： 1.公钥加密技术与对称加密技术相比，其优势在于不需要共享通用的密钥。

2.公钥在传递和发布过程中即使被截获，由于没有与公钥相匹配的私钥，截获的公钥对入侵者没有太大意义。

3.密钥少便于管理，N个用户通信只需要N对密钥，网络中每个用户只需要保存自己的解密密钥。

4.密钥分配简单，加密密钥分发给用户，而解密密钥由用户自己保留。

缺点：加密算法复杂，加密和解密的速度比较慢

3.3.3 总结：

通过我们的了解,对称加密的实现比较简单,但是安全性不是很高,一般用在对安全要求级别不是很高的情况下,但是非对称加密比较复杂,但是安全性很高,在实际应用中需要我们综合利弊考虑.

4 加密相关代码API简介

参考文章：Cipher

android提供了一个专门搞加密的API。 类名叫做Cipher，可以进行对称性加密解密和非对称性加密解密。所以以下重点看看Clipher代码：

先看他是怎么使的，以解密为例：

 /\*\*

 \* AES 解密（128位密钥）

 \*

 \* @param cipherText

 \* @param key

 \* @return

 \* @throws Exception

 \*/

 public static byte[] decodeAES(byte[] cipherText,

 String key) throws Exception {

 SecureRandom secureRandom = SecureRandom.getInstance("SHA1PRNG", "Crypto");

 secureRandom.setSeed(key.getBytes());

 KeyGenerator kgen = KeyGenerator.getInstance(ALGORITHM\_AES);

 kgen.init(128, secureRandom);

 SecretKey secretKey = kgen.generateKey();

 byte[] enCodeFormat = secretKey.getEncoded();

// 构建秘钥规范，实则是一个被用来做 指定算法加密的 秘钥。

 SecretKeySpec skeySpec = new SecretKeySpec(enCodeFormat, ALGORITHM\_AES);

// Cipher 我认为实则是一个加密器的包装类 ，就是用来做加密解密的。但是事先我们需要

// 定义好 究竟要以那种模式进行加密或者解密

// 获取实例，参数是设置加密参数，目前为AES加密

 Cipher cipher = Cipher.getInstance(ALGORITHM\_AES);

// 设置为解密模式， 并把秘钥传进去

 cipher.init(Cipher.DECRYPT\_MODE, skeySpec);

 byte[] result = cipher.doFinal(cipherText);

 return result;

 }

1

2

3

4

5

6

7

8

9

10

11

12

13

14

15

16

17

18

19

20

21

22

23

24

25

26

27

这里面我们重点看两个项目：1 SecretKeySpec， 2 Cipher。以及他们的用法。

4.1 整体理解

目前我个人的理解是，SecretKeySpec描述的就是一把秘钥。 而Cipher则更像是一个万能的加密解密器。 我们用的时候只需要给出正确的秘钥，然后给这个加密器说明此次需要的秘钥，和要进行哪种方式的加密。然后再调用相应的代码即可。

4.2 SecreKeySpec

主要是用来构建秘钥规范。其代码是比较简单的。但是我认为它实则就是描述了一把秘钥！并且加密的时候或者解密的时候会用到这把秘钥。

其代码比较简单：

总共就没几个方法，就三个属性。

看看构造方法：

 /\*\*

 \* key： 秘钥数据

 \* algorithm：采用的算法模式

 \* \*/

 public SecretKeySpec(byte[] key, String algorithm) {

 if (key == null) {

 throw new IllegalArgumentException("key == null");

 }

 if (key.length == 0) {

 throw new IllegalArgumentException("key.length == 0");

 }

 if (algorithm == null) {

 throw new IllegalArgumentException("algorithm == null");

 }

 this.algorithm = algorithm;

 this.key = new byte[key.length];

 System.arraycopy(key, 0, this.key, 0, key.length);

 }

 public SecretKeySpec(byte[] key, int offset, int len, String algorithm) {

 if (key == null) {

 throw new IllegalArgumentException("key == null");

 }

 if (key.length == 0) {

 throw new IllegalArgumentException("key.length == 0");

 }

 if (len < 0 || offset < 0) {

 throw new ArrayIndexOutOfBoundsException("len < 0 || offset < 0");

 }

 if (key.length - offset < len) {

 throw new IllegalArgumentException("key too short");

 }

 if (algorithm == null) {

 throw new IllegalArgumentException("algorithm == null");

 }

 this.algorithm = algorithm;

 this.key = new byte[len];

 System.arraycopy(key, offset, this.key, 0, len);

 }

其实两个构造参数的目的是一样的，一个就是设置原始的秘钥参数，另外一个就是设置这把秘钥所针对的编码方式。

这个构造参数是关键，其余的方法感觉没啥好讲的。另外有一个关键点是，这个类实现了三个接口：SecretKey, KeySpec, Serializable， 其中最后一个，很有意思，说明这把钥匙是可以序列化的。存储，传输应该没毛病。

4.3 编码解码工具 cipher：

cipher类相比于上一个类就要复杂的多。但是他的目的是用来根据秘钥和加密模式来对指定的数据进行加密的。但是他又是可以支持多种类型的加密方式的，并且同时也支持相应的解密。所以大致来说，它必须需要两个关键信息。1 加密模式 2 秘钥！所以最上面的解密demo为啥子要这样用，以及参数的意义到现在心里应该有个大概的认识。我们暂且剖析几个比较关键的方法。

4.3.1 Cipher.getInstatnce()

看了上方的使用范例，我们首先会Cipher.getInstance(ALGORITHM\_AES); 这个方法来获取一个cipher实例。其实Cipher还有一个重载的getInstance()方法。但是两者的目的是相同的，都是用来产生一个cipher对象。期间涉及了一些参数检测相关的操作。

方法如下：

public static final Cipher getInstance(String transformation)

public static final Cipher getInstance(String transformation, String provider)

其中这个方法有个关键参数，是String类型的。但这并不代表着你随便塞一个字符串就行的。这个参数是有特殊意义的。这个字符串实则是描述了要对给定的输入究竟做哪种操作。比如单单是DES加密算法，里面的内部操作有些规则还是要单独定义的。比如分组的时候，是以8位为单元划分还是16位单元划分，如果最后不够一个单元的话，需要用什么来填补其他的位来凑成一个单元。这些细节都是需要我们去定义的。transformation这个参数恰恰是要说明这件事情的。

1 transformation参数: 长得大致格式如下

“algorithm/mode/padding” 或者

“algorithm”

例如 ：Cipher c = Cipher.getInstance("DES/CBC/PKCS5Padding");

algorithm:表示要用到的算法。

mode：表示要用到的算法模式。通常是用来指定算法分割数据时的单元要有多少位。因为毕竟加密算法时把数据进行一系列有顺序的位操作的。当然这个mode具体要代表是何种操作根据具体的算法而定，这里只是说明一下个人见解。

padding：代表的是填充模式。

算法名 mode padding 支持API

AES CBC

CFB

CTR

CTS

ECB

OFB

GCM ISO10126Padding

NoPadding

PKCS5Padding

NoPadding 1+

10+

三个参数的具体表现，请参考：node.js AES/ECB/PKCS5Padding 与其他语言的加密解密通用

三个参数的具体原理，请参考：对称加密和分组加密中的四种模式(ECB、CBC、CFB、OFB)

4.3.2 cipher.init（）方法。

作用是设定一些加密工具，里面涉及了设置底层代码参数的，得到真正的加解密工具。拿最简单的一个init（）重载方法 init(int opmode, Key key)来说， 分析下参数.所有的重载方法都有一个参数叫 opmode。这个参数是很重要的。

其中第一个参数opmode：用来表示此次操作类型。通常采取下面几个参数：

 /\*\*

 \* 表示加密

 \*/

 public static final int ENCRYPT\_MODE = 1;

 /\*\*

 \*表示解密

 \*/

 public static final int DECRYPT\_MODE = 2;

 /\*\*

 \* 包装密钥

 \*/

 public static final int WRAP\_MODE = 3;

 /\*\*

 \* 解包装密钥

 \*/

 public static final int UNWRAP\_MODE = 4;

第二个参数 key：这个很简单，他就是来描述一个密钥。通常是KEY这个接口的实现类。

cipher.doFinal(data);

经过前面的设置，程序里已经得知，要进行的操作（加密还是解密？）， 具体的加密算法， 加密算法用到的密钥。 接下里就是进行这些动作综合描述出来的操作了。产出的结果，就是被按照原先设置的规则来加密或者解密之后的byte数组。

————————————————

版权声明：本文为CSDN博主「娅娅梨」的原创文章，遵循 CC 4.0 BY-SA 版权协议，转载请附上原文出处链接及本声明。

原文链接：https://blog.csdn.net/weixin\_28774815/article/details/81359015