HTTPS 协议到底比 HTTP 协议多些什么

HTTP 简介

HTTP 协议是 Hyper Text Transfer Protocol（超文本传输协议）的缩写,是用于从万维网（WWW:World Wide Web ）服务器传输超文本到本地浏览器的传送协议。

HTTP 是一个基于 TCP/IP 通信协议来传递数据（HTML 文件, 图片文件, 查询结果等）。

HTTP 是一个属于应用层的面向对象的协议，由于其简捷、快速的方式，适用于分布式超媒体信息系统。它于 1990 年提出，经过几年的使用与发展，得到不断地完善和扩展。目前在 WWW 中使用的是 HTTP/1.0 的第六版，HTTP/1.1 的规范化工作正在进行之中，而且 HTTP-NG(Next Generation of HTTP)的建议已经提出。

HTTP 协议工作于客户端-服务端架构为上。浏览器作为 HTTP 客户端通过 URL 向 HTTP 服务端即 WEB 服务器发送所有请求。Web 服务器根据接收到的请求后，向客户端发送响应信息。

HTTP 特点：

1、简单快速：客户向服务器请求服务时，只需传送请求方法和路径。请求方法常用的有 GET、HEAD、POST。每种方法规定了客户与服务器联系的类型不同。由于 HTTP 协议简单，使得 HTTP 服务器的程序规模小，因而通信速度很快。

2、灵活：HTTP 允许传输任意类型的数据对象。正在传输的类型由 Content-Type 加以标记。

3、无连接：无连接的含义是限制每次连接只处理一个请求。服务器处理完客户的请求，并收到客户的应答后，即断开连接。采用这种方式可以节省传输时间。

4、无状态：HTTP 协议是无状态协议。无状态是指协议对于事务处理没有记忆能力。缺少状态意味着如果后续处理需要前面的信息，则它必须重传，这样可能导致每次连接传送的数据量增大。另一方面，在服务器不需要先前信息时它的应答就较快。
5、支持 B/S 及 C/S 模式。

以上简单回顾 HTTP 协议的相关内容。HTTP 协议有什么弊端么，还需要一个叫做 HTTPS 协议的安全协议。这里先说一下 HTTP 协议有什么样弊端？

HTTP 弊端：

当我们往服务器发送比较隐私的数据（比如说你的银行卡，身份证）时，如果使用 http 进行通信。那么安全性将得不到保障。

首先数据在传输的过程中，数据可能被中间人抓包拿到，那么数据就会被中间人窃取。

其次数据被中间人拿到后，中间人可能对数据进行修改或者替换，然后发往服务器。

最后服务器收到数据后，也无法确定数据有没有被修改或替换，当然，如果服务器也无法判断数据就真的是来源于客户端。

总结下来，**http 存在三个弊端：**

- 无法保证消息的保密性
- 无法保证消息的完整性和准确性
- 无法保证消息来源的可靠性

HTTPS 了解一下

如何解决 HTTP 弊端呢？HTTPS 就是为了解决上述问题应运而生的。

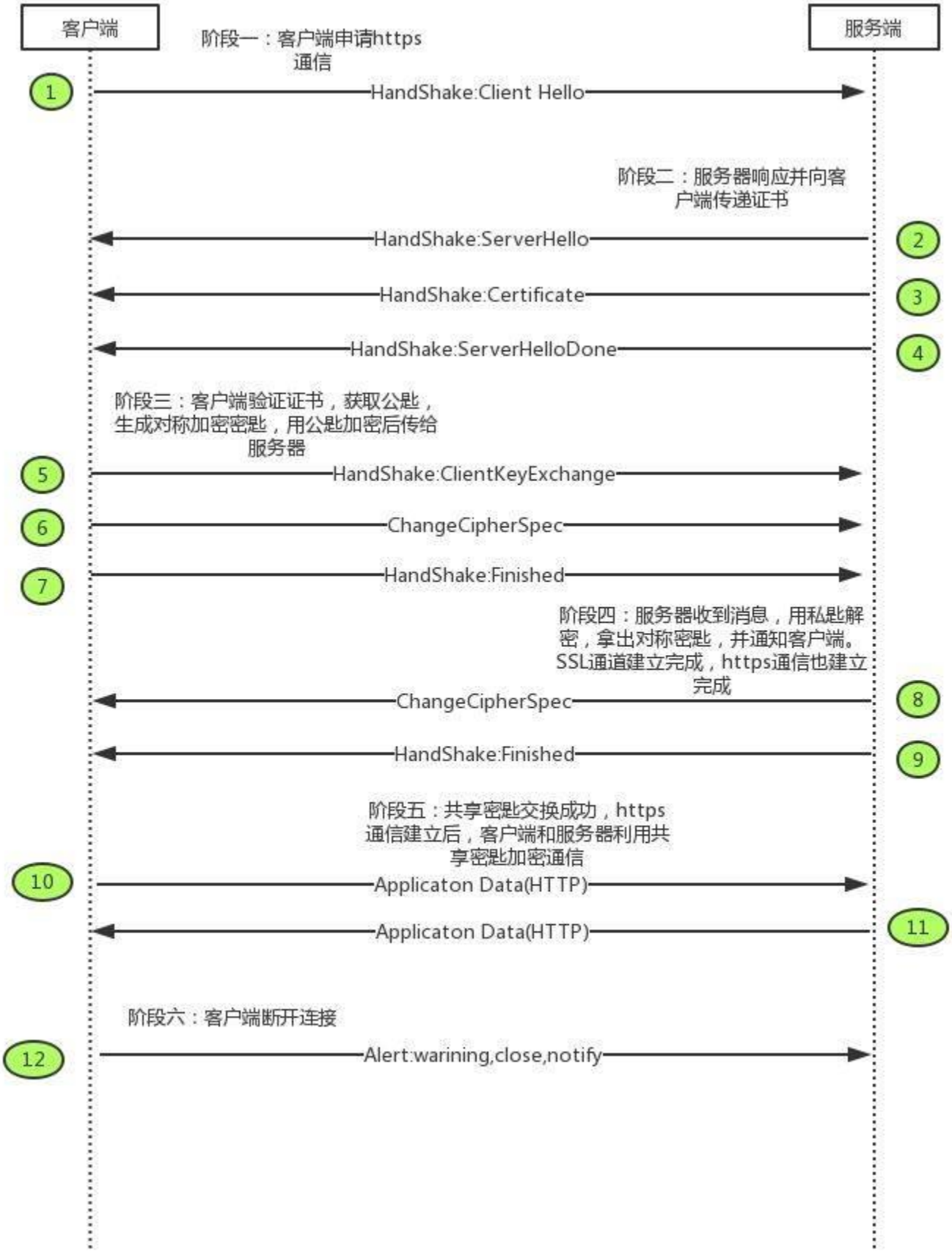
HTTPS（全称：Hyper Text Transfer Protocol over Secure Socket Layer），是以安全为目标的 HTTP 通道，简单讲是 HTTP 的安全版。

即 HTTP 下加入 SSL 层，HTTPS 的安全基础是 SSL，因此加密的详细内容就需要 SSL。 现在它被广泛用于万维网上安全敏感的通讯，例如交易支付方面。

HTTPS 通过非对称加密算法可以使得我们传的明文信息，无法通过逆推得出明文。接下来我们来看看在具体的工作流程是怎么样的？

工程原理：

https 的建立


客户端
服务端
阶段一：客户端申请https通信
1
HandShake:Client Hello
阶段二：服务器响应并向客户端传递证书
HandShake:ServerHello
2
HandShake:Certificate
3
HandShake:ServerHelloDone
4
阶段三：客户端验证证书，获取公匙，生成对称加密密匙，用公匙加密后传给服务器
5
HandShake:ClientKeyExchange
6
ChangeCipherSpec
7
HandShake:Finished
阶段四：服务器收到消息，用私匙解密，拿出对称密匙，并通知客户端。SSL通道建立完成，https通信也建立完成
ChangeCipherSpec
8
HandShake:Finished
9
阶段五：共享密匙交换成功，https通信建立后，客户端和服务器利用共享密匙加密通信
10
Applicaton Data(HTTP)
Applicaton Data(HTTP)
11
阶段六：客户端断开连接
12
Alert:warining,close,notify

这里把 https 建立到断开分为 6 个阶段，12 过程。下面将对 12 个过程一 一做解释

1. **客户端—Hello**：客户端通过发送 Client Hello 报文开始 SSL 通信。报文中包含客户端支持的 SSL 的指定版本、加密组件（Cipher Suite）列表（所使用的加密算法及密匙长度等）。

2. **服务器—Hello**：服务器可进行 SSL 通信时，会以 Server Hello 报文作为应答。和客户端一样，在报文中包含 SSL 版本以及加密组件。服务器的加密组件内容时从接收到的客户端加密组件内筛选出来的。

3. **服务器—发证书**：服务器发送证书报文。报文中包含公开密匙证书。

4. **服务器—我说完了**：最后服务器发送 Server Hello Done 报文通知客户端，最初阶段的 SSL 握手协商部分结束。

5. **客户端—发送秘钥**：SSL 第一次握手结束之后，客户端以 Client Key Exchange 报文作为回应。报文包含通信加密中使用的一种被称为 Pre-master secret 的随机密码串。该报文已用步骤 3 中的公开密匙进行加密。

6. **客户端—就用这个秘钥了**：该报文会提示服务器，在此报文之后的通信会采用 Pre-master secret 密匙加密。

7. **客户端—我说完了**：该报文包含连接至今全部报文的整体校验值。这次握手协商是否能够成功，要以服务器是否能够正确解密该报文作为判定标准。

8. **服务器—发送 Change Cipher Spec 报文（我正在接收秘钥）**

9. **服务器—发送 Finished 报文（我收完秘钥了）**

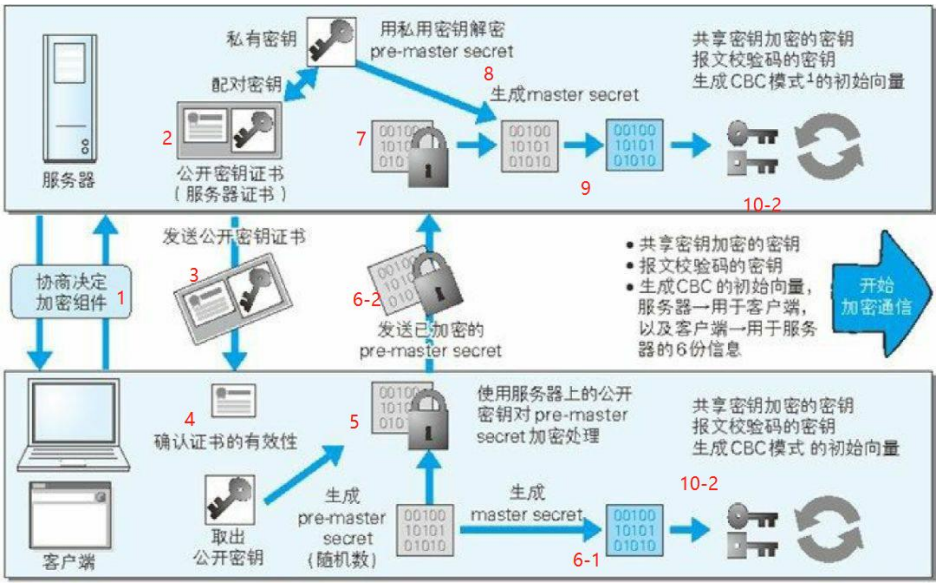
10. **客户端—开始发送正文**：服务器端发送 HTTP 请求，发送相关内容。

11. **服务器—开始接收正文**：客户端接收 HTTP 请求，并处理相关内容。

12. **客户端—断开链接：**最后由客户端断开连接。断开连接时，发送 close_notify 报文。上图做了一些省略，这步之后再发送 TCP FIN 报文来关闭与 TCP 的通信。

另外，在以上流程图中，应用层发送数据时会附加一种叫做 MAC（Message Authentication Code）的报文摘要。MAC 能够查知报文是否遭到篡改，从而保证报文的完整性。

下面再用图解来形象的说明一下，此图比上面数字证书的图更加的详细一些（图片来源于《图解 HTTP》）



在上面说明了 HTTPS 的建立以及通信中的过程。既然实际工作流程是这个样子的，是怎样的算法能实现这样的功能，是怎样的方式能做到非对称加密？在数学角度是如何计算的？那么对应的理论基础是什么？是什么支撑的 HTTPS 使得他能进行加密传输？

HTTPS 的理论原理：

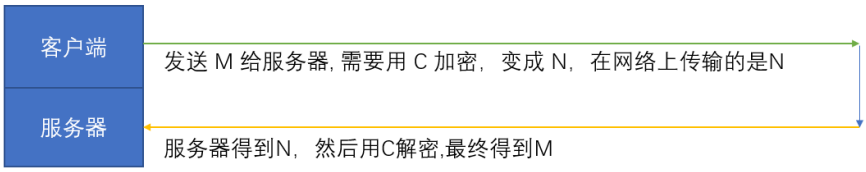
https 采用了一些加解密，数字证书，数字签名的技术来实现。下面先介绍一下这些技术的基本概念

对称加密与非对称加密

为了保证消息的保密性，就需要用到加密和解密。加解密算法目前主流的分为对称加密和非对称加密。

1.对称加密（共享密匙加密）：客户端和服务器公用一个密匙用来对消息加解密，这种方式称为对称加密。客户端和服务器约定好一个加密的密匙。客户端在发消息前用该密匙对消息加密，发送给服务器后，服务器再用该密匙进行解密拿到消息。

图示加密过程：



这里采用的对称加密算法，

这里 M 指的是**明文**，我们本意要传输的内容。

这里 C 指的是**秘钥**，在对称加密算法中需要用秘钥加密，用秘钥解密（加密算法可以很简单，加减乘除，也可以很复杂）。

这里 N 指的是**密文**，明文用秘钥加密得到的内容，被称为密文，在网络上传输的也是密文。

比如客户端向服务器传输 1（明文），1 + 3 （3 是秘钥） = 4 得到密文，进行传输，服务器得到 密文 4， 4-3（3 是秘钥）=1 得到明文，使得客户端和服务器端进行通信，反之亦然

对称加密的优点：

- 对称加密解决了 http 中消息保密性的问题

对称加密的缺点：

- 对称加密虽然保证了消息保密性，但是因为客户端和服务器共享一个密匙，这样就使得密匙特别容易泄露。
- 因为密匙泄露风险较高，所以很难保证消息来源的可靠性、消息的完整性和准确性。

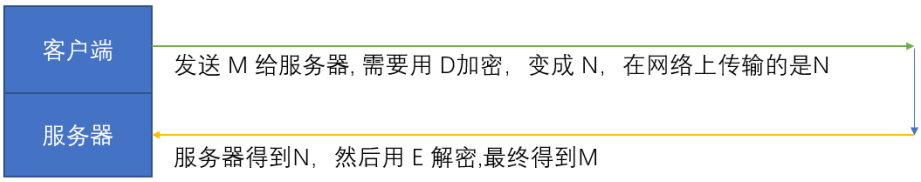
对称加密秘钥泄露风险很高，秘钥固定，导致很容易被破解，那么有没有更好的方式去进行加密传输，比如说每次用的秘钥都不相同，每次解密的秘钥也都不相同，或者其他的情况来增加安全性呢？

2. 非对称加密（公有密匙加密）：既然对称加密中，密匙那么容易泄露，那么我们可以采用一种非对称加密的方式来解决。

采用非对称加密时，客户端和服务端均拥有一个公有密匙和一个私有密匙。公有密匙可以对外暴露，而私有密匙只有自己可见。

使用公有密匙加密的消息，只有对应的私有密匙才能解开。反过来，使用私有密匙加密的消息，只有公有密匙才能解开。这样客户端在发送消息前，先用服务器的公匙对消息进行加密，服务器收到后再用自己的私匙进行解密。

图示加密过程：



这里 M 指的是**明文**，我们本意要传输的内容。

这里 D 指的是**公钥**，在非对称加密算法中需要用公钥加密。

这里 E 指的是**私钥**，在非对称加密算法中需要用私钥解密。

这里 N 指的是**密文**，明文用秘钥加密得到的内容，被称为密文，在网络上传输的也是密文。

在服务器这一次生成公钥 D 以及私钥 E，私钥自己留存。然后将公钥 D 进行对外公开，想与服务器端通信的客户端用公钥 D 进行加密发送给具有私钥 E 的服务器，服务器用私钥 E 就可以进行密文解密，最终拿到明文。

简单介绍一个非对称加密算法 RSA：

RSA 是目前最有影响力的公钥加密算法，它能够抵抗到目前为止已知的绝大多数密码攻击，已被 ISO 推荐为公钥数据加密标准。 今天只有短的 RSA 钥匙才可能被强力方式解破。到 2008 年为止，世界上还没有任何可靠的攻击 RSA 算法的方式。只要其钥匙的长度足够长，用 RSA 加密的信息实际上是不能被解破的。但在分布式计算和量子计算机理论日趋成熟的今天，RSA 加密安全性受到了挑战。

RSA 算法基于一个十分简单的数论事实：**将两个大质数相乘十分容易，但是想要对其乘积进行因式分解却极其困难，因此可以将乘积公开作为加密密钥。**

RSA 算法的过程

RSA 算法用到的数学知识特别多，所以在中间介绍这个算法生成私钥和公钥的过程中会穿插一些数学知识。生成步骤如下：

1. 寻找两个不相同的质数

随意选择两个大的质数 p 和 q，**p 不等于 q，计算 N=p*q;**

什么是质数?我想可能会有一部分人已经忘记了，定义如下：

> 除了 1 和该数自身外，无法被其他自然数整除的数（也可定义为只有 1 该数本身两个正因数]的数）。

比如 2，3，5，7 这些都是质数，9 就不是了，因为 3*3=9 了

2. 根据欧拉函数获取 r

$r = \phi(N) = \phi(p)\phi(q) = (p-1)(q-1)$。

这里的数学概念就是什么是欧拉函数了，什么是欧拉函数呢？

欧拉函数的定义：

> **欧拉函数** $\phi(n)$*是小于或等于* n 的正整数中与 n 互质的数的数目。

互质的定义：

> 如果两个或两个以上的整数的最大公约数是 1，则称它们为**互质**

例如：$\phi(8) = 4$，因为 $1,3,5,7$ 均和 8 互质。

推导欧拉函数:

（1）如果 $n = 1$，$\phi(1) = 1$；(小于等于 1 的正整数中唯一和 1 互质的数就是 1 本身)；

（2）如果 n 为质数，$\phi(n) = n - 1$；因为质数和每一个比它小的数字都互质。比如 5，比它小的正整数 1,2,3,4 都和他互质；

(3) 如果 n 是 a 的 k 次幂，则 $\phi(n) = \phi(a\hat{}k) = a\hat{}k - a\hat{}(k-1) = (a-1)a\hat{}(k-1)$；

(4) 若 m,n 互质，则 $\phi(mn) = \phi(m)\phi(n)$

证明：设 A，B，C 是跟 m，n，mn 互质的数的集，据中国剩余定理(经常看数学典故的童鞋应该了解，剩余定理又叫韩信点兵，也叫孙子定理)，$A*B$ 和 C 可建立双射一一对应)的关系。（来自维基百科）

3. 选择一个小于 r 并与 r 互质的整数 e

选择一个小于 r 并与 r 互质的整数 e，求得 e 关于 r 的模反元素，命名为 *d*（*ed = 1(mod r)*模反元素存在，当且仅当 e 与 r 互质），*e* 我们通常取 65537。

模反元素：

> 如果两个正整数 a 和 n 互质，那么一定可以找到整数 *b*，使得 ab-1 被 n 整除，或者说 ab 被 n 除的余数是 1。

比如 *3* 和 *5* 互质，*3* 关于 *5* 的模反元素就可能是 2，因为 *3*2-1=5* 可以被 5 整除。所以很明显模反元素不止一个，2 加减 5 的整数倍都是 3 关于 5 的模反元素*{...-3, 2,7,12…}* 放在公式里就是 *3*2 = 1 (mod 5)*

上面所提到的欧拉函数用处实际上在于欧拉定理：

欧拉定理：

> 如果两个正整数 *a* 和 *n* 互质，则 *n* 的欧拉函数 *φ(n)* 可以让下面的等式成立：
>
> *a^φ(n) = 1(mod n)*
>
> 由此可得：*a* 的 *φ(n - 1)次方肯定是 a* 关于 *n* 的模反元素。

欧拉定理就可以用来证明模反元素必然存在。

由模反元素的定义和欧拉定理我们知道，*a* 的 *φ(n)次方减去 1，可以被 n 整除。比如，3 和 5 互质，而 5* 的欧拉函数 *φ(5)等于 4，所以 3* 的 *4* 次方*(81)*减去 1，可以被* 5*整除（*80/5=16*）。

小费马定理：

> 假设正整数 a 与质数 p 互质，因为质数 p 的 *φ(p)等于 p-1*，则欧拉定理可以写成
>
> *a^(p-1) = 1 (mod p)*
>
> 这其实是欧拉定理的一个特例。

4. 销毁 p 和 q

此时我们的*(N , e)*是公钥，*(N, d)*为私钥，爱丽丝会把公钥(N, e)*传给鲍勃，然后将(N, d)*自己藏起来。一对公钥和私钥就产生了

RSA 算法的安全性

我们知道像 RSA 这种非对称加密算法很安全，那么到底为啥子安全呢？ 我们来看看上面这几个过程产生的几个数字：

- *p, q*：我们随机挑选的两个大质数；
- *N*：是由两个大质数 *p* 和 *q* 相乘得到的。*N = p * q*；
- *r*：由欧拉函数得到的 *N* 的值，*r = φ(N) = φ(p)φ(q) = (p-1)(q-1)*。
- *e*：随机选择和和 *r* 互质的数字，实际中通常选择 65537；
- *d*： d 是以欧拉定理为基础求得的 e 关于 r 的模反元素，*ed = 1 (mod r)*；

N 和 *e* 我们都会公开使用，最为重要的就是私钥中的 *d*，*d* 一旦泄露，加密也就失去了意义。那么得到 d 的过程是如何的呢？如下：

1. 比如知道 e 和 r，因为 d 是 e 关于 r 的模反元素；r 是 φ(N) 的值
2. 而 *φ(N)=(p-1)(q-1)*，所以知道 p 和 q 我们就能得到 d；
3. *N = pq*，从公开的数据中我们只知道 N 和 e，所以**问题的关键就是对 N 做大素数因式分解能不能得出 p 和 q？**

将 a 和 b 相乘得出乘积 c 很容易，但要是想要通过乘积 c 推导出 a 和 b 极难。即对一个大素数进行因式分解极难

目前公开破译的位数是 768 位，实际使用一般是 1024 位或是 2048 位，所以理论上特别的安全。

非对称加密的优点：

- 非对称加密采用公有密匙和私有密匙的方式，解决了 http 中消息保密性问题，而且使得私有密匙泄露的风险降低。
- 因为公匙加密的消息只有对应的私匙才能解开，所以较大程度上保证了消息的来源性以及消息的准确性和完整性。

非对称加密的缺点：

- 非对称加密时需要使用到接收方的公匙对消息进行加密，但是公匙不是保密的，任何人都可以拿到，中间人也可以。那么中间人可以做两件事，第一件是中间人可以在客户端与服务器交换公匙的时候，将客户端的公匙替换成自己的。这样服务器拿到的公匙将不是客户端的，而是服务器的。服务器也无法判断公匙来源的正确性。第二件是中间人可以不替换公匙，但是他可以截获客户端发来的消息，然后篡改，然后用服务器的公匙加密再发往服务器，服务器将收到错误的消息。
- 非对称加密的性能相对对称加密来说会慢上几倍甚至几百倍，比较消耗系统资源。正是因为如此，https 将两种加密结合了起来。

对于非对称加密的性能慢的解决方案是：用一个密钥来加密通信，那个对称加密算法是非常快的，但是苦于密钥无法安全传输， 现在有了 RSA ,可以结合一下， 分两步走 (1) 生成一个对称加密算法的密钥， 用 RSA 的方式安全发给对方， (2) 随后就不用 RSA 了， 只用这个密钥，利用对称加密算法来通信。

对于中间人劫持的方式，我们需要考虑到使用数字证书与数字签名。

数字证书与数字签名

为了解决非对称加密中公匙来源的不安全性。我们可以使用数字证书和数字签名来解决。

1.**数字证书的申请**

在现实中，有一些专门的权威机构用来颁发数字证书，我们称这些机构为认证中心（CA Certificate Authority）。

我们（服务器）可以向这些 CA 来申请数字证书。

申请的过程大致是：

自己本地先生成一对密匙，然后拿着自己的公匙以及其他信息（比如说企业名称啊什么的）去 CA 申请数字证书。

CA 在拿到这些信息后，会选择一种单向 Hash 算法（比如说常见的 MD5）对这些信息进行加密，加密之后的东西我们称之为摘要。

单向 Hash 算法有一种特点就是单向不可逆的，只要原始内容有一点变化，加密后的数据都将会是千差万别（当然也有很小的可能性会重复，有兴趣的小伙伴鸽巢原理了解一下），这样就防止了信息被篡改。

生成摘要后还不算完，CA 还会用自己的私匙对摘要进行加密，摘要加密后的数据我们称之为数字签名。

最后，CA 将会把我们的申请信息（包含服务器的公匙）和数字签名整合在一起，由此而生成数字证书。然后 CA 将数字证书传递给我们。

2.**数字证书怎么起作用**

服务器在获取到数字证书后，服务器会将数字证书发送给客户端，客户端就需要用 CA 的公匙解密数字证书并验证数字证书的合法性。那我们如何能拿到 CA 的公匙呢？我们的电脑和浏览器中已经内置了一部分权威机构的根证书，这些根证书中包含了 CA 的公匙。

之所以是根证书，是因为现实生活中，认证中心是分层级的，也就是说有顶级认证中心，也有下面的各个子级的认证中心，是一个树状结构，计算机中内置的是最顶级机构的根证书，不过不用担心，根证书的公匙在子级也是适用的。

客户端用 CA 的公匙解密数字证书，如果解密成功则说明证书来源于合法的认证机构。解密成功后，客户端就拿到了摘要。

此时，客户端会按照和 CA 一样的 Hash 算法将申请信息生成一份摘要，并和解密出来的那份做对比，如果相同则说明内容完整，没有被篡改。最后，客户端安全的从证书中拿到服务器的公匙就可以和服务器进行安全的非对称加密通信了。服务器想获得客户端的公匙也可以通过相同方式。