# Diffie-Hellman

[编辑](javascript:;) [锁定](http://baike.baidu.com/view/10812319.htm)

Diffie-Hellman:一种确保共享KEY安全[穿越](http://baike.baidu.com/subview/287172/14317184.htm" \t "_blank)不安全网络的方法，它是OAKLEY的一个组成部分。Whitefield与Martin Hellman在1976年提出了一个奇妙的密钥交换协议，称为Diffie-Hellman密钥交换协议/算法(Diffie-Hellman Key Exchange/Agreement Algorithm).这个机制的巧妙在于需要安全通信的双方可以用这个方法确定对称密钥。然后可以用这个密钥进行加密和解密。但是注意，这个密钥交换协议/算法只能用于密钥的交换，而不能进行消息的加密和解密。双方确定要用的密钥后，要使用其他对称密钥操作加密算法实际加密和解密消息。

Diffie-Hellman密钥交换算法的安全性依赖于这样一个事实：虽然计算以一个素数为模的指数相对容易，但计算离散对数却很困难.对于大的素数，计算出离散对数几乎是不可能的. 下面给出例子.密钥交换基于素数q = 97和97的一个原根a = 5.A和B分别选择私有密钥XA = 36和XB = 58.每人计算其公开密钥 YA = 5^36 mod 97= 50 ,YB = 5^58 mod 97= 44 在他们相互获取了公开[密钥](http://baike.baidu.com/view/934.htm" \t "_blank)之后，各自通过计算得到双方共享的秘密密钥如下： K = (YB)^XA mod 97 = 44^36 mod 97= 75 ,K = (YA)^XB mod 97 = 50^58 mod 97= 75 从|50,44|出发，攻击者要计算出75很不容易. 下图给出了一个利用Diffie-Hellman计算的简单协议.[1]

**Diffie-Hellman协议**

[编辑](javascript:;)

假设用户A希望与用户B建立一个连接，并用一个共享的秘密密钥加密在该连接上传输的报文.用户A产生一个一次性的私有密钥XA，并计算出公开密钥YA并将其发送给用户B.用户B产生一个私有密钥XB，计算出公开密钥YB并将它发送给用户A作为响应.必要的公开数值q和a都需要提前知道.另一种方法是用户A选择q和a的值，并将这些数值包含在第一个报文中. 下面再举一个使用Diffie-Hellman算法的例子.假设有一组用户（例如一个局域网上的所有用户），每个人都产生一个长期的私有密钥XA，并计算一个公开密钥YA.这些公开密钥数值，连同全局公开数值q和a都存储在某个中央目录中.在任何时刻，用户B都可以访问用户A 的公开数值，计算一个秘密密钥，并使用这个密钥发送一个加密报文给A.如果中央目录是可信任的，那么这种形式的通信就提供了保密性和一定程度的鉴别功能.因为只有A和B可以确定这个密钥，其它用户都无法解读报文（保密性）.接收方A知道只有用户B才能使用此密钥生成这个报文（鉴别）. Diffie-Hellman算法具有两个吸引力的特征： 仅当需要时才生成密钥，减小了将密钥存储很长一段时间而致使遭受攻击的机会. 除对全局参数的约定外，密钥交换不需要事先存在的基础结构.

**原根的定义**

[编辑](javascript:;)

设m是正整数，a是整数，若a模m的阶等于φ(m)，则称a为模m的一个原根。（其中φ(m)表示m的[欧拉函数](http://baike.baidu.com/view/107769.htm" \t "_blank)）

假设一个数g对于P来说是原根，那么g^i mod P的结果两两不同,且有 1<g<P, 0<i<P,那么g可以称为是P的一个原根,归根到底就是g^(P-1) = 1 (mod P)当且仅当指数为P-1的时候成立.(这里P是素数).

简单来说，g^i mod p ≠ g^j mod p （p为素数）

其中i≠j且i, j介於1至(p-1)之间

则g为p的原根。

求原根目前的做法只能是从2开始枚举，然后暴力判断g^(P-1) = 1 (mod P)是否当且当指数为P-1的时候成立

而由于原根一般都不大，所以可以暴力得到.

**原根的性质**

[编辑](javascript:;)

1）可以证明，如果正整数(*a*,*m*) = 1和正整数 d 满足a^d≡1(mod m)，则 d 整除 φ(*m*)。因此*Ordm*(*a*)整除φ(*m*)。在例子中，当*a*= 3时，我们仅需要验证 3 的 1 、2、3 和 6 次方模 7 的[余数](http://baike.baidu.com/view/1068391.htm" \t "_blank)即可。

2）记δ = *Ordm*(*a*)，则a^1，……a^(δ-1)模 m 两两不[同余](http://baike.baidu.com/view/79282.htm" \t "_blank)。因此当*a*是模*m*的原根时，a^0,a^1，……a^(δ-1)构成模 m 的[简化剩余系](http://baike.baidu.com/view/1685888.htm" \t "_blank)。

3）模*m*有原根的充要条件是*m*= 1,2,4,*p*,2*p*,*p^n*，其中*p*是奇质数，*n*是任意[正整数](http://baike.baidu.com/view/464125.htm" \t "_blank)。

4）对正整数(*a*,*m*) = 1，如果 a 是模 m 的原根，那么 a 是**整数模n乘法群**（即加法群 **Z**/m**Z**的可逆元，也就是所有与 m [互素](http://baike.baidu.com/view/47657.htm)的正整数构成的[等价类](http://baike.baidu.com/view/732593.htm)构成的乘法群）**Z**n的一个[生成元](http://baike.baidu.com/view/3599892.htm" \t "_blank)。由于**Z**n有 φ(*m*)个元素，而它的生成元的个数就是它的可逆元个数，即 φ(φ(*m*))个，因此当模*m*有原根时，它有φ(φ(*m*))个原根。

**原根的例子**

[编辑](javascript:;)

设*m*= 7，则φ（7）等于6。

设*a*= 2，由于2^3=8≡1(mod 7)，2^6=64≡1(mod7)，而2!=3，2^3≡2^6(mod7)，所以 2 不是模 7 的一个原根。设*a*= 3，由于3^1≡3(mod 7)，3^2≡2(mod 7)，3^3≡6(mod 7)，3^4≡4(mod 7)，3^5≡5(mod 7)，3^6≡1(mod 7)，所以 3 是模 7 的一个原根。

补充一点，根据原根的性质1，只需要验证3^1，3^2，3^3，3^6即可，这样可以简化运算。

# Diffie-Hellman密钥交换算法

之前做过的一个项目中用过DH算法（Diffie-Hellman），这种密钥交换技术的目的在于使得两个用户安全地交换一个共享密钥（shared secret）以便用于以后的报文加密。

DH算法的有效性依赖于计算离散对数的难度。简言之，可以如下定义离散对数：首先定义一个素数*p*的原根，为其各次幂产生从1 到*p*-1的所有整数根，也就是说，如果*g*是素数*p*的一个原根，那么数值

*g* mod *p*, g2 mod *p*, ..., gp-1 mod *p*

是各不相同的整数，并且以某种排列方式组成了从1到*p*-1的所有整数。

对于一个整数*b*和素数*p*的一个原根*g*，可以找到惟一的指数*i*，使得

*b* = *gi* mod *p*     其中0 ≤ *i*≤ (*p*-1)

指数 *i*称为*b*的以*g*为基数的模*p*的离散对数或者指数。该值被记为ind*g ,p*(b)。

下面是DH算法的原理：

有两个全局公开的参数，一个素数p和一个整数g，g是p的一个原根。

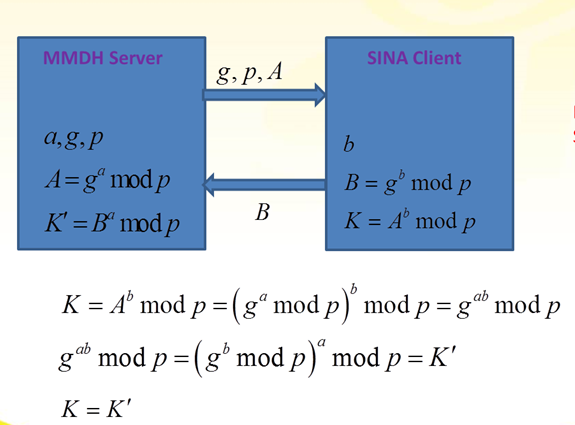
服务端的私钥和公钥分别是a和A，客户端的私钥和公钥分别是b和B；

服务端根据a、p、g，可以计算出公钥A；

服务端将g, p, A明文传送给客户端，客户端可以计算自己的公钥B，以及共享密钥K；

客户端将B明文发送给服务端，服务端也可以计算出共享密钥K。

具体如下图：



根据上述流程，客户端于服务端之间明文传输的是g、p、A、B，然后双方就可以得到一个同样的共享密钥K，并以K对后续要传输的数据进行对称加密、解密即可。

由于计算K需要一方的私钥和另一方的公钥，所以理论上，只要双方的私钥a、b不被第三方知道，这个共享密钥K就是安全可用的。

**DH算法具有两个吸引力的特征：**

1. 仅当需要时才生成密钥，减小了将密钥存储很长一段时间而致使遭受攻击的机会；
2. 除对全局参数的约定外，密钥交换不需要事先存在的基础结构；

然而，该技术也存在许多不足：

1. 没有提供双方身份的任何信息；
2. 计算密集性，因此容易遭受阻塞性攻击，即对手请求大量的密钥。受攻击者花费了相对多的计算资源来求解无用的幂系数而不是在做真正的工作；
3. 没办法防止重演攻击；
4. 容易遭受中间人的攻击。第三方C在和A通信时扮演B；和B通信时扮演A。A和B都与C协商了一个密钥，然后C就可以监听和传递通信量。中间人的攻击按如下进行：

* B在给A的报文中发送他的公开密钥YB。
* C截获并解析该报文。C将B的公钥保存下来并给A发送报文，该报文具有B的用户ID但使用C的公钥YC，但仍按照好像是来自B的样子被发送出去。A收到C的报文后，将YC和B的用户ID存储在一块。类似地，C使用YC向B发送好像来自A的报文。
* B基于私钥XB和YC计算共享密钥K1，A基于私钥XA和YC计算共享密钥K2，C使用私钥XC和YB计算K1，并使用XC和YA计算K2。
* 从现在开始，C就可以转发A发给B的报文或转发B发给A的报文，在途中根据需要修改它们的密文。使得A和B都不知道他们在和C共享通信。

Oakley算法是对Diffie-Hellman密钥交换算法的优化，它保留了后者的优点，同时克服了其弱点。

Oakley算法具有五个重要特征：

1、它采用称为cookie程序的机制来对抗阻塞攻击。

2、它使得双方能够协商一个全局参数集合。

3、它使用了现时来保证抵抗重演攻击。

4、它能够交换Diffie-Hellman公开密钥。

5、它对Diffie-Hellman交换进行鉴别以对抗中间人的攻击。

Oakley可以使用三个不同的鉴别方法：

1、数字签名：通过签署一个相互可以获得的散列代码来对交换进行鉴别；每一方都使用自己的私钥对散列代码加密。散列代码是在一些重要参数上生成的，如用户ID和现时。

2、公开密钥加密：通过使用发送者的私钥对诸如ID和现时等参数进行加密来鉴别交换。

3、对称密钥加密：通过使用某种共享密钥对交换参数进行对称加密，实现交换的鉴别。