RSA的应用：数字签名

最普遍的应用，网站身份认证。如何证明我们连上的网站就是支付宝alipay呢？如果因为各种原因，如域名污染，我们的浏览器访问了攻击者网站，这时一定要进行验证。

验证，就是要检查一个证书。当我们以HTTPS的方式连上一个网站时，网站会首先给我们发送一个证书。这个证书里包含有它的域名、公钥等信息。同时这个证书是由专门的第三方公信机构CA使用自己的私钥签了名的。浏览器在拿到这个证书之后，首先用第三方公信机构CA的公钥对这个证书解密，然后查看和比对证书里的域名和浏览器地址栏的域名，完全匹配才认为是正确的网站。

如果域名被污染，虽然攻击者网站可以拷贝一份正常网站的证书，但是因为证书中包括了正常网站的公钥，如果它不能获得正常网站的私钥，那么它就没有办法对加密信息进行解密。从而不能正常建立连接。

那攻击者有没有可能伪造一份证书呢？只要攻击者拿不到第三方CA的私钥，就没有办法完成签名。那攻击者有没有可能伪造CA呢?

不OK，因为浏览器会内置CA，默认有CA的信息

ECC椭圆曲线加密

- 产生原因

  RSA算法是当前使用最广的非对称加密算法。但是RSA的缺点在于为了抵抗攻击，不得不增加公钥的长度。而随着长度的增加，计算量和复杂度也不断增加。正是因为非对称加密复杂度太高，所以一般仅用于在网络连接建立时的密钥协商过程。**而且随着数字大小的增加，分解的效率会提高，乘法和分解的难度差距会减小**。

  所以RSA并不是将来密码学中最理想的系统。在理想的trapdoor函数中，正向计算（简单的计算）和反向计算（复杂计算）的难度应该随着数字的增加同步地增加。 所以需要更好的trapdoor函数。

- 相关概念

  1985年提出了基于椭圆曲线加密ECC（Elliptic Curve Cryptography）。

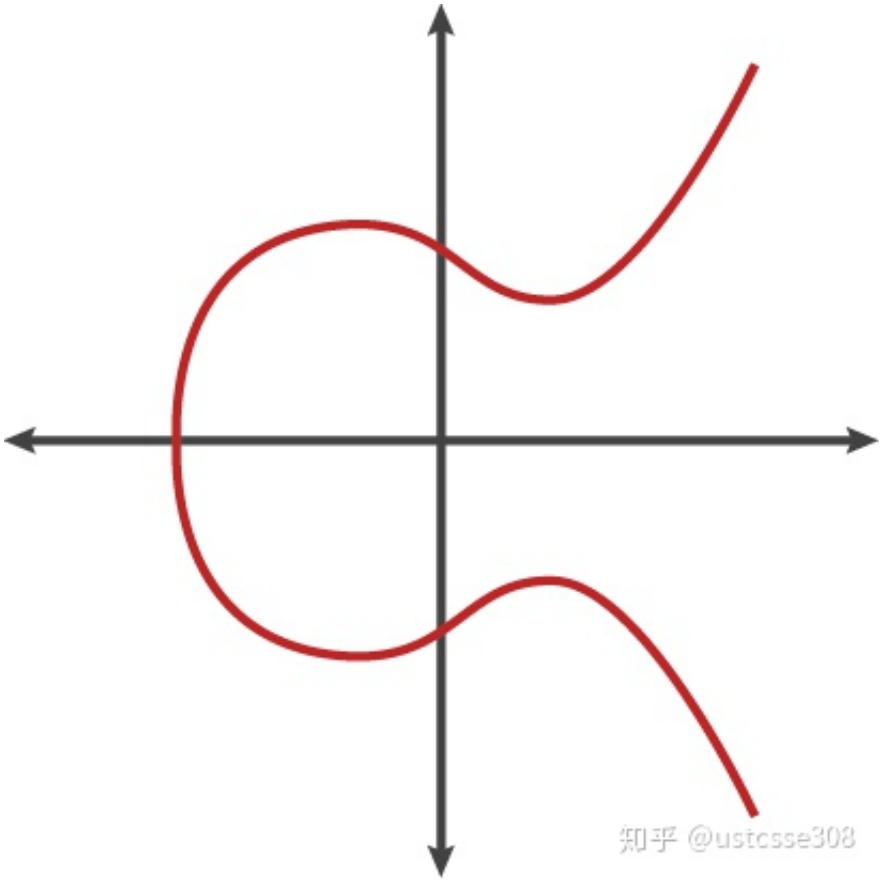
  ECC和RSA不同，RSA相对而言更容易理解。因为大家都知道指数、幂乘、模数运算等概念。但是椭圆曲线的概念更抽象，不太好理解。

  简单来说，椭圆曲线就是满足一个函数的一些点的集合。

  椭圆曲线有各种形式，但一般而言，是包括两个变量的函数，其中一个次数是2，一个是3。一个椭圆曲线函数大概是这样：

  $$y^2 = x^3 + ax + b$$

  对应的曲线长这样：



虽然名字是椭圆曲线，但是本身并不像椭圆。

不同的椭圆曲线对应

不同的形状（b=1，a从2到-3）

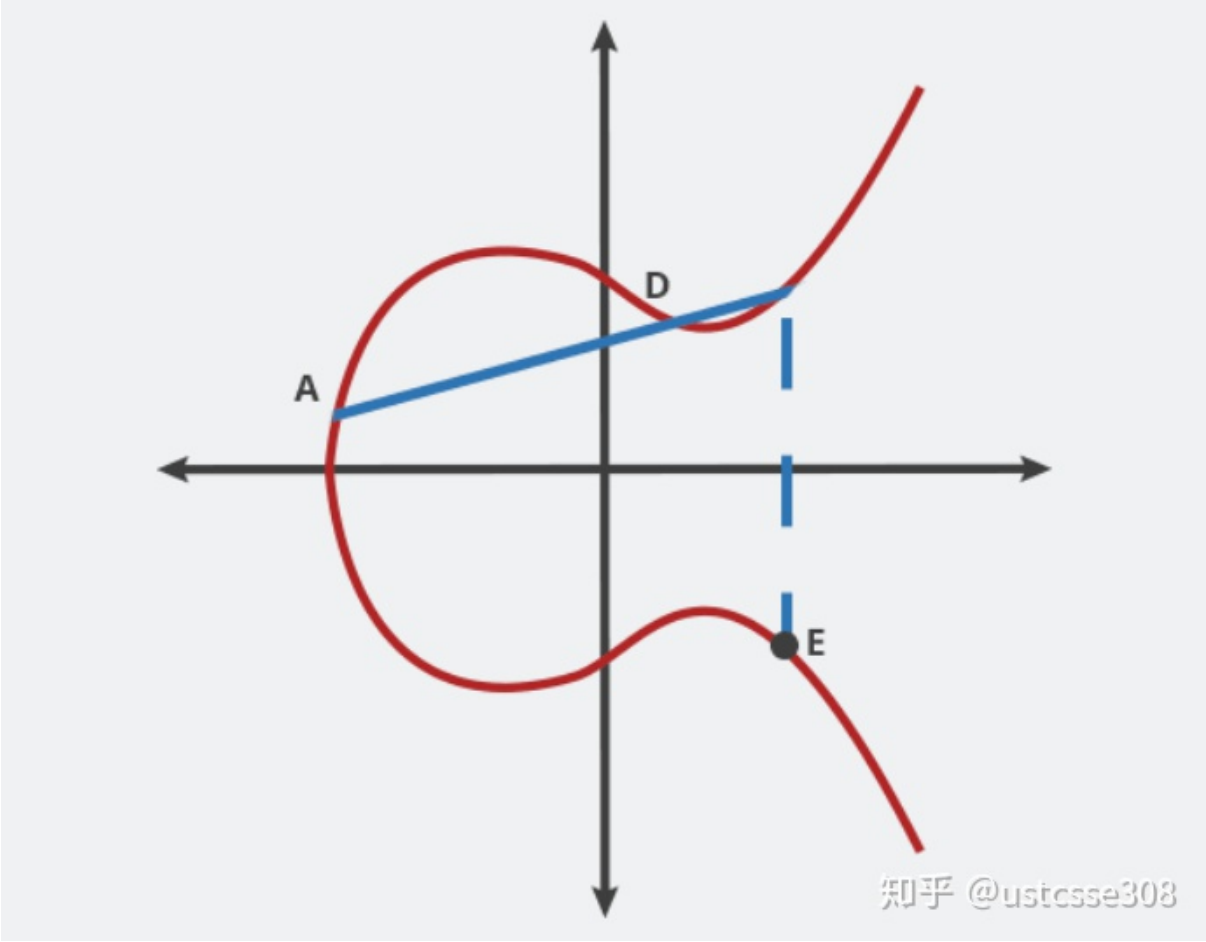
椭圆曲线之所以用在加密系统中，是因为它有一些很好的特性可以适合用来加密。

椭圆曲线的一个特性是关于X轴对称。另一个特性是任何不垂直的线最多与曲线有三个交点。

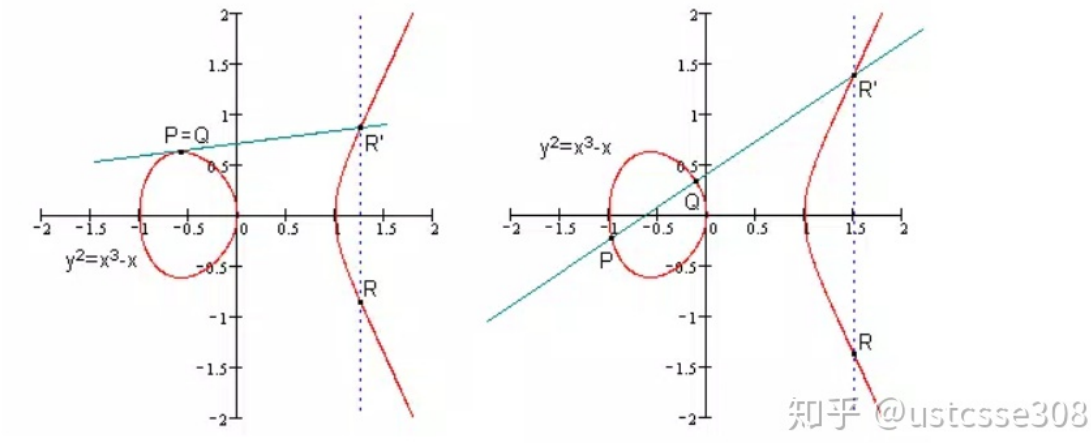
基于这些特性，在椭圆曲线上定义一些运算。（后面的“加法”和“乘法”只是为方便而取的名字，和我们所熟知的加法和乘法完全不同）

在椭圆曲线上两点的加，指的是经过椭圆曲线上两点的线，和椭圆曲线的交点（第三个点）关于X轴的对称点。

譬如说，在下面这张图中，A+D = E



还有这种，P+Q=R （右图）

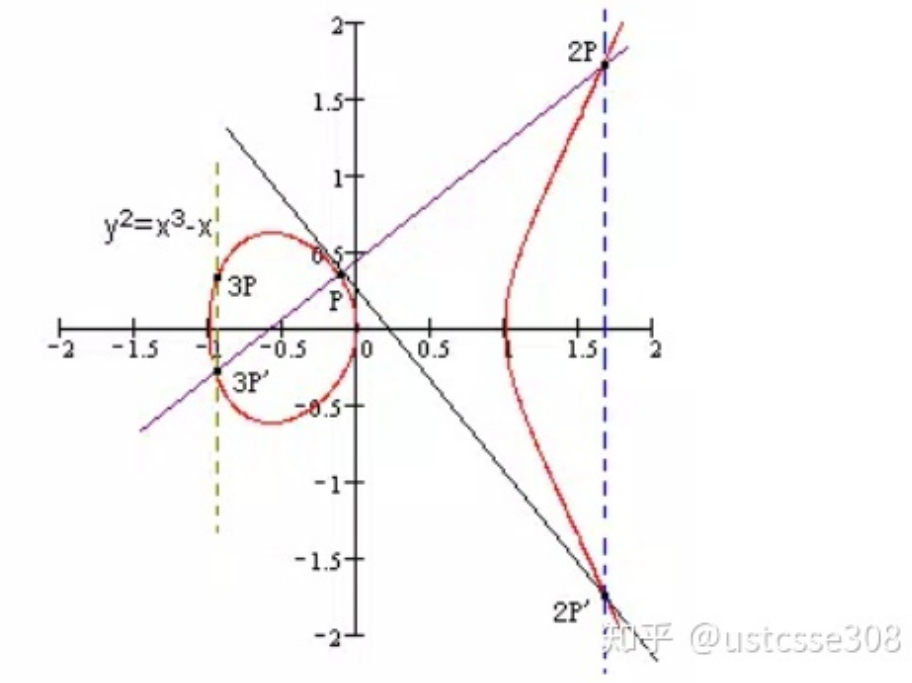


特别地，如果要计算P+P（也即P=Q），那就作椭圆曲线在P点处的切线，与曲线相交于第二点，然后关于X轴对称得到第三点，则第三点为P点与自身的和，R=P+P。（左图）

根据这种定义，可以看出加法具有交换律。

还能**定义乘法**。

P+P首先得到2P。连接2P和P的直线与椭圆曲线相交，通过取与X轴对称的点，得到3P。如此，可以一直计算下去，得到kP(1 < k < n)。



- 椭圆曲线难以破解的原因

  椭圆曲线难以破解的地方在于，给定点G和K，K = kG (1 < k < n)，想要推导出k是一件很困难的事情，目前没有比枚举k的值好很多的算法，而通常n会很大。

- 椭圆曲线的加密通信过程

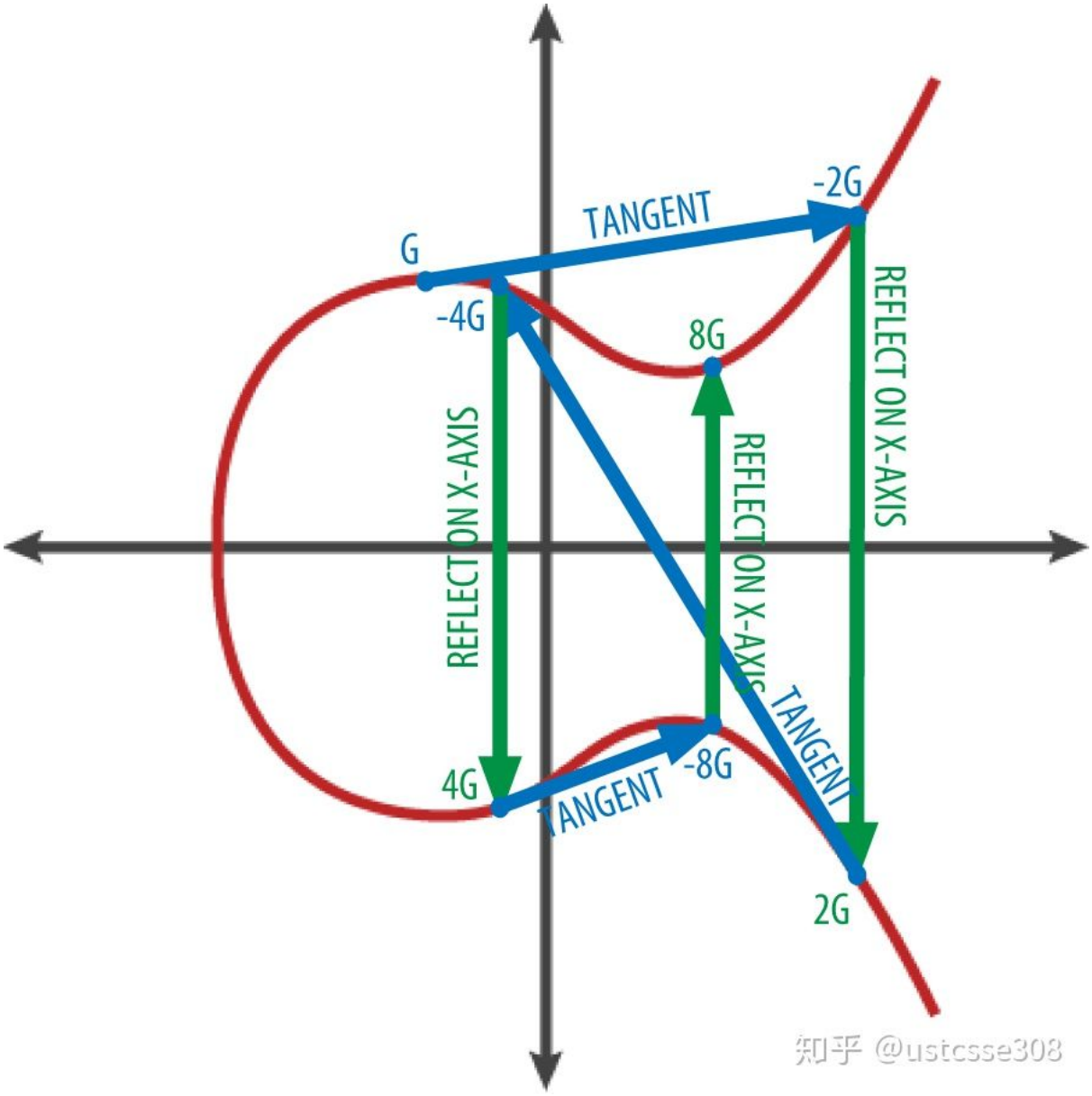
  1、Alice选定一条椭圆曲线EC(x,y)，并取椭圆曲线上一点，作为基点G。
  2、Alice选择一个**私有密钥k**，并生成**公开密钥K=kG**。
  3、Alice将EC(x,y)和点K，G传给Bob。
  4、Bob接到信息后，将待传输的明文通过一定的方法编码到EC(x,y)上的一点M，并生成随机整数r（r<n）。
  5、Bob计算点C1=M+rK；C2=rG。
  6、Bob将C1、C2传给Alice。
  7、Alice接到信息后，计算C1-kC2：C1-kC2=M+rK-k(rG)=M+rK-r(kG)=M；然后对点M进行解码就可以得到明文。

- 为什么逆推k很难，而计算K=kG却比较简单?

  譬如说计算8G，这时只需要计算2G=（G+G），4G = (2G+2G)，以及8G=（4G+4G）即可。

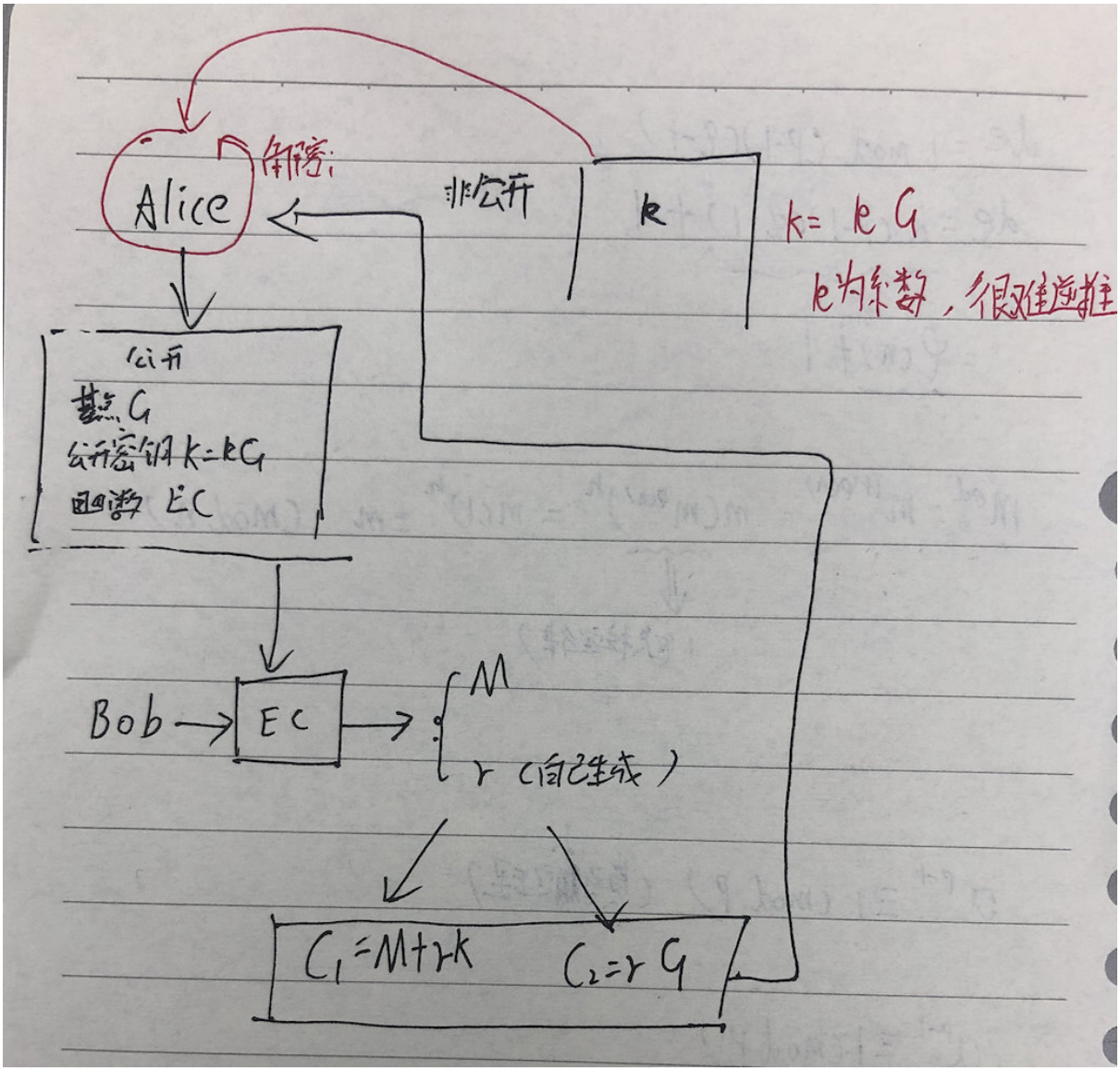
  10G类似，在计算出8G的情况下，再加上2G就行，而2G之前也已计算得出。

  （可以对照RSA中的快速幂乘法）



但是在不知道k的情况下只能枚举来尝试

- ECC椭圆曲线算法图示



非对称加密在区块链中的应用

在区块链中，公钥==身份。

在这之前，需要介绍一下数字签名的概念。数字签名类似于人们在纸上的签名。数字签名对应于手写签名需要有两个特点：1. 签名不能伪造，也即只有自己能做出这个签名；2. 签名是针对一份文档的，也即，一个签名只能和一个文档相关联，不能拿到一个签名用于多个文档，类似于不能把一份文件上的签名撕下来贴到另一个文件上。

- 具体实现
    1. 首先生成一对公私钥 (pk, sk) 。 pk 是公钥， sk 是私钥。
    2. 签名sign。 $sig = sign(sk, message)$ ，使用私钥对一份消息message进行处理，譬如先对message进行哈希得到摘要，然后使用私钥对摘要进行加密。
    3. 验证verify。 $verify(pk, message, sig)$ 。验证方法，获得输入message，签名的结果sig，以及公钥。譬如，可以使用公钥对签名结果进行解密，对message进行同样散列得到的摘要，比较解密的结果和摘要结果，如果两个相同，这验证通过，否则，验证失败。

    也即，如果能够进行`verify(pk,msg,sig)==true`，那么，就可以认为pk确实说了msg。当然，为了以pk的身份说话，就必须有相应的私钥sk。

所以，可以认为 pk 就是身份identity，在比特币区块链中，使用 pk 的哈希作为身份。在生成一对公私钥之后，拥有 sk 的人可以控制 pk 。

- 使用公钥(的哈希)作为身份带来的好处

  分布式的身份管理。没有必要使用身份管理中心来管理这些身份。人们可以生成任意多个公钥进行交易。在区块链中，称为地址address（实际上是公钥的哈希）。