- **算数基础定理**： 任何一个数，只能由它的素数因子以唯一种方式组成。如30的可分成15个2，10个3，6个5.所以2，3，5是30的素数因子，那么2^x+3^y+5^z=30,其中x,y,z只有唯一解。亦即所有的自然数都有一个唯一的钥匙->它的素数因子唯一的一种组合方式。

- **概率稳定性**： 对于一组长序列数，每个样本出现的概率相等，同时，每个样本组成的小序列在长序列中出现的概率也相等。如01010100010101101011100010010101001001中，判断它是否真正随机，若即使序列中0，1出现概率一样，但000，001，010，011，100，101，110，111这些小序列出现的概率有差别也可证明这一长序列数是非随机的。

- **破解凯撒密码的概率分析**： 利用正常英文中每个字母出现的概率特征，算出密文中每个字母的出现概率，两相对比，推算偏移量。
- **解决方法->多字母密码**： 平滑密文中的字母概率分布。

- **一次性码本**： 首先偏移量序列长度大于等于明文长度，不存在循环时。其次，偏移量序列有骰子掷出，完全随机，没有频率差异。

- **完美秘密**： 明文空间与秘文空间相同。

- **伪随机数生成器**：
  中间平方->随机产生种子，之后将种子x种子，取中间位数，依次循环。
  但存在的问题是当中间位数出现了和初代种子一样的数时，随机序列出现周期。周期的长度依赖于初代种子的位数。当种子长度足够长时，周期长达万亿长度->伪随机。
  但其概率空间与真实随机数小很多。

- **离散对数问题**:
  生成器g和巨大的素数模p。由于本地产生的随机数x是不通过网络的，窃听者无法盗取。**g^x mod p = y，而已知结果y，无法在合理时间内通过生成器及素数模推算出随机数x。**所以y（包括生成器a和巨大的素数模b）可以放心的网络中传播。Alice与Bob交换彼此的结果Ya与Yb。结论是Yb^Xa mod p = Ya^Xb mod p = s。因为Yb=g^Xb mod p,所以Yb^Xa mod p = (g^Xb)^Xa mod p（同余运算）。同理， Ya^Xb mod p = (g^Xa)^Xb mod p。所以Alice与Bob通过生成器g和素数模p的计算公式，交换Ya，Yb即交换了隐形的Xa，Xb。已知了自己随机数与对方的隐形随机数，即可将共同产生的s定为双方约定好的密钥。

  密钥s作为种子产生长度大于等于明文信息的伪随机数，伪随机数作为一次性码本在明文内容上是依次作为偏移量来加密内容

  Diffie_Hellman密钥交换算法即找到了一个公式可以将信息A hash成B，将信息C hash成D再传输。而func(A,D)=func(B,C)

- **RSA**：
  Alice生成pubKey与priKey，前提是已知priKey可以快速算出pubKey，但是已知pubKey无法在合理时间内得到priKey。且pubKey与priKey互为逆。Alice将pubKey公开，则所有的Bob都可以用pubKey加密内容发

给Alice，Alice用priKey解密即可。
对于公私钥的寻找，在算数基础定理中提到：任何一个数，只能由它的素数因子以唯一种方式组成。而寻找这种组合方式的办法有能通过从小到大的素数不断试错。恰巧计算机也不能快速计算出这个答案，随着这个自然数位数的增加，计算机分解素数因子的时间指数式上升，当自然数位数达到一定程度上，这个问题即在合理时间内无解。
所以，Alice随机生成两个150位的素数p1与p2。将p1xp2得到一个300多位自然数N。
https://www.jianshu.com/p/ff2b538a77e2

- **hash函数**：
  1. collision resistance：已知H(m)，难以找到m使得`H(m)=H(m`)。
  2. hiding：x->H(x)这一过程单向，隐藏原始信息。
  3. puzzle friendly:x->H(x)这一过程不存在特性或规律，即要生成一个符合某个特殊规律的hash并不容易，只能暴力试出。（BTC要求，其中BTC使用的hash函数叫做SHA—256，SHA:secure hash aglorithm）

- **nonce**：
  nonce=number once hash函数要求具有hiding特点，hiding的前提是输入空间足够大且概率分布均匀，以避免轻易地暴力穷举到。但实际中，如果一个应用场景下，输入的样本空间通常不够大，这时可以在样本值后面加上一个随机数再hash，hash(x||nonce)
  在BTC中，每个块头有多个信息，矿工在当前高度的块头中存储的这些信息基础上，不断尝试加上一个nonce值，然后对整体信息作hash，以产生一个符合要求格式的hash值，即挖矿成功。而正是hash函数中puzzle friendly的特性才令矿工只能蛮力的不断尝试寻找那个合适的nonce。
  一个矿工挖矿成功后，其他人只需做一次hash即可验证->diffcult to solve，but easy to verify。

- **BTC账户**： 首先是（public key，private key），非对称加密算法（asymmetric encryption algorithm）。同时要有一个好的随机数生成器，这样就不会发生私钥碰撞的情况。
  vscode

  1. 生成256bit二进制随机数（转化成16进制就是64字节的私钥） 采用ECDSA椭圆曲线数字签名算法（属于ECC算法，较RSA算法更安全快速），取曲线y^2=x^3+ax+b上的一点A(x,y)，做椭圆运算kx+ky=公钥，其中椭圆加法A+B是点A与点B的连线与曲线相交，交点作x轴对称得到的点就是A+B的结果，kA就是最终得到的公钥，只是一个点，有一个点无法推出原始的位置和私钥k（Trapdoor function）。
  2. 另外，这个原始的点A不可以是随便的简单的点，这样的话同私钥k相乘得到的公钥就会很简单 ，所以要对这个点提要求，把这个椭圆曲线对应到某个质数阶p的有限域内，质数p越大越安全。
  3. btc使用secp256k1标准，确定是哪条椭圆曲线（a=0和b=7），大质数p和基准点G的坐标点(即原始的A点)，G(x,y)其中x和y各32位字节，私钥K和G相乘后得到的公钥P是65字节（64+0x04）。
     vscode

  我们将信息hash之后用公钥加密，私钥解密。签名时用私钥签名，公钥验证。