



第一部分 加解密

04. 哈希函数

厚德健行

- ❑ 假设Alice对M进行签名
 - ❑ Alice发送M和 $S = [M]_{\text{Alice}}$ 给Bob
 - ❑ Bob通过 $M = \{S\}_{\text{Alice}}$ 来验证签名
 - ❑ Alice仅发送S也可以吗？
- ❑ 如果M的长度非常大， $[M]_{\text{Alice}}$ 计算量就非常大
- ❑ 相反地，如果存在长度远远小于M的长度，并可以代表M的 $h(M)$ ，那么Alice可以对 $h(M)$ 进行签名
 - ❑ Alice发送M和 $S = [h(M)]_{\text{Alice}}$ 给Bob
 - ❑ Bob通过 $h(M) = \{S\}_{\text{Alice}}$ 来验证签名

- 所以, Alice签名 $h(M)$
 - 就是说, Alice计算出 $S = [h(M)]_{\text{Alice}}$
 - 然后Alice发送 (M, S) 给Bob
 - Bob检验 $h(M) = \{S\}_{\text{Alice}}$
- $h(M)$ 必须满足哪些要求呢?
 - 假设Trudy发现了 M' , 所以 $h(M) = h(M')$
 - 然后Trudy 能替换 (M, S) 和 (M', S)
- Bob能发现这样的篡改吗?
 - 不能, 因为 $h(M') = h(M) = \{S\}_{\text{Alice}}$

- 密码学hash函数 $h(x)$ 必须满足下列特性
 - 压缩 — 输出长度很小
 - 效率 — 对于任意给定的 x ，计算 $h(x)$ 很容易
 - 单向 — 给定任意值 y ，找到一个值 x 使得 $h(x) = y$ 在计算上是不可行的
 - 弱抗碰撞 — 给定 x 和 $h(x)$ ，找到 y ，并且 $y \neq x$ ，使得 $h(y) = h(x)$ ，在计算上不可行
 - 强抗碰撞 — 找到任意的 x 和 y ，并且 $x \neq y$ ，使得 $h(x) = h(y)$ ，在计算上不可行
- 存在很多的碰撞，但是碰撞是难以寻找的

- 假设你和N个人在一个屋子里
- 存在有人和你生日相同的概率超过 $\geq 1/2$ ，需要N为多大？
 - 解答N： $1/2 = 1 - (364/365)^N$
 - 求解N = 253

- 任何两个或两个以上人的生日相同的概率超过 $\geq \frac{1}{2}$ ，需要N为多大？
 - $1 - 365/365 \cdot 364/365 \cdots (365-N+1)/365$
 - 令上式等于 $1/2$ ，求解: **$N = 23$**
- 惊讶吗？感觉荒谬吗？
- 计算相同生日对x和y的临界点为 $N = \sqrt{365}$ ，这样看来生日悖论并非不合理
 - 有365种可能的生日

- 如果 $h(x)$ 产生 N 位输出，于是有 2^N 种不同的hash值
- 因此，对 $2^{N/2}$ 个随机数数去hash值，将会找到一个碰撞
 - $\text{sqrt}(2^N) = 2^{N/2}$
- **意味着：**产生 N 位输出的安全hash函数需要大约 $2^{N/2}$ 次计算就能攻破，而对于密钥长度为 N 位的对称密码需要大约 2^{N-1} 次计算才能攻破
 - 这是穷举搜索攻击

- ❑ 数据 $X = (X_0, X_1, X_2, \dots, X_{n-1})$ ，这里每个 X_i 是一个字节
- ❑ 定义 $\text{hash}(X) = X_0 + X_1 + X_2 + \dots + X_{n-1}$
- ❑ 这样安全吗？
- ❑ 例如: $X = (10101010, 00001111)$
- ❑ Hash值为10111001
- ❑ 与 $Y = (00001111, 10101010)$ 的hash值相同
- ❑ 碰撞很容易找到，因此是不安全的...

- ❑ 数据 $X = (X_0, X_1, X_2, \dots, X_{n-1})$
- ❑ 假设hash函数为
 - ❑ $h(X) = nX_0 + (n-1)X_1 + (n-2)X_2 + \dots + 1 \cdot X_{n-1}$
- ❑ hash函数是否安全呢？
- ❑ 至少
 - ❑ $h(10101010, 00001111) \neq h(00001111, 10101010)$
- ❑ 但是 $(000000001, 00001111)$ 的hash值与 $(00000000, 00010001)$ 的hash值是相同的
- ❑ 并不是单向的，但是它在非密码学应用中被成功使用

- ❑ 循环冗余校验码(CRC)
- ❑ 本质上讲这是计算除法，余数作为CRC值
- ❑ 除数一旦选定，很容易找出碰撞
 - ❑ 随机误差不可能产生碰撞
- ❑ 但也很容易构造碰撞
- ❑ CRC 有时候被错误地用于有密码学方面要求的应用中(WEP)

- **MD5** — 由Rivest设计的
 - 128位的输出
 - 注意: 近来发现MD5碰撞
- **SHA-1** — 是美国政府标准 (类似 MD5)
 - 160位的输出
- 还有很多的hash函数, 但是MD5和SHA-1是当今使用最广的
- Hash函数是通过对块消息进行hash完成工作的

- ❑ 需要的特性：雪崩效应
 - ❑ 任何一个输入位的改变都将引起一半的输出位值的改变
- ❑ 密码学hash函数由若干轮组成
- ❑ 需要的安全性和速度
 - ❑ 数轮迭代后产生雪崩效应
 - ❑ 每轮的操作越简单越好
- ❑ 同设计迭代分组密码一样

- "快速和健壮"
- 由Ross Anderson和Eli Biham提出的
- 设计标准
 - 安全的
 - 对于64位处理器做出优化
 - 适合取代MD5或SHA-1

- 通过使用"hashed MAC" 或HMAC根据密钥K能够计算消息M的消息认证码MAC
- HMAC是根据密钥进行hash计算的一个例子
 - 为什么需要密钥K呢？
- 怎样来计算HMAC呢？
- 两个直观的方法： $h(K,M)$ 和 $h(M,K)$
- 哪个更好呢？

- ❑ 我们能用 $h(K,M)$ 来计算HMAC吗？
- ❑ 对分组进行hash处理
 - ❑ 对于某个F和常量A 有： $h(B_1, B_2) = F(F(A, B_1), B_2)$
 - ❑ 那么 $h(B_1, B_2) = F(h(B_1), B_2)$
- ❑ 使得 $M' = (M, X)$
 - ❑ 那么 $h(K, M') = F(h(K, M), X)$
 - ❑ 攻击者可以不用K就能计算 M' 的HMAC
- ❑ $h(M, K)$ 是否会更好呢？
 - ❑ 答案是肯定的，但是如果 $h(M') = h(M)$ ，那么我们会得到 $h(M, K) = F(h(M), K) = F(h(M'), K) = h(M', K)$

- ❑ 如RFC 2104所述
- ❑ 令B为hash的以字节为单位的分组长度
 - ❑ 对于MD5,SHA-1和Tiger, $B = 64$
- ❑ $\text{ipad} = 0x36$ 重复B次
- ❑ $\text{opad} = 0x5C$ 重复B次
- ❑ 那么

$$\text{HMAC}(M,K) = H(K \oplus \text{opad}, H(K \oplus \text{ipad}, M))$$

- ❑ 认证 (HMAC)
- ❑ 消息完整性 (HMAC)
- ❑ 消息指纹
- ❑ 数据损毁检测
- ❑ 辅助数字签名
- ❑ Hash函数能够完成对称密钥密码所能完成的任何功能
- ❑ 同时，还有许多灵巧的，令人惊奇的用法

- ❑ 假设Alice, Bob 和 Charlie是投标者
- ❑ Alice 标价A, Bob标价B, Charlie标价C
- ❑ 他们担心标价会被泄露
- ❑ 解决途径?
 - ❑ Alice, Bob, Charlie分别提交hash值 $h(A)$, $h(B)$, $h(C)$
 - ❑ 所有hash值都收到并在线公开
 - ❑ 然后标价A, B 和 C才提交
- ❑ Hash函数是安全的 (具有单向性)
- ❑ 提交标价的hash值后便不能改变标价 (碰撞)
- ❑ 但这里有一个缺陷

1.请回顾前述在线竞标方案。

a.在这种方案中,一个赖以预防诈骗的安全哈希函数 h 应该具备什么样的一些特性呢?

单向不可逆、抗碰撞

b.假如Charlie能确定Alice和Bob将提交的竞投价格必定会在\$10000和\$20000之间。请描述一种前向检索攻击,使得Charlie可以基于Alice和Bob两人各自的哈希值,使用这种攻击手段确定Alice和Bob的竞投价格。

遍历1000与2000之间的数

c.在b中所描述的攻击,是否会构成一个实际的安全问题?

会

d.对这个网上竞价过程进行什么样的改造,才能够防止类似b中所述的那种前向检索攻击呢?

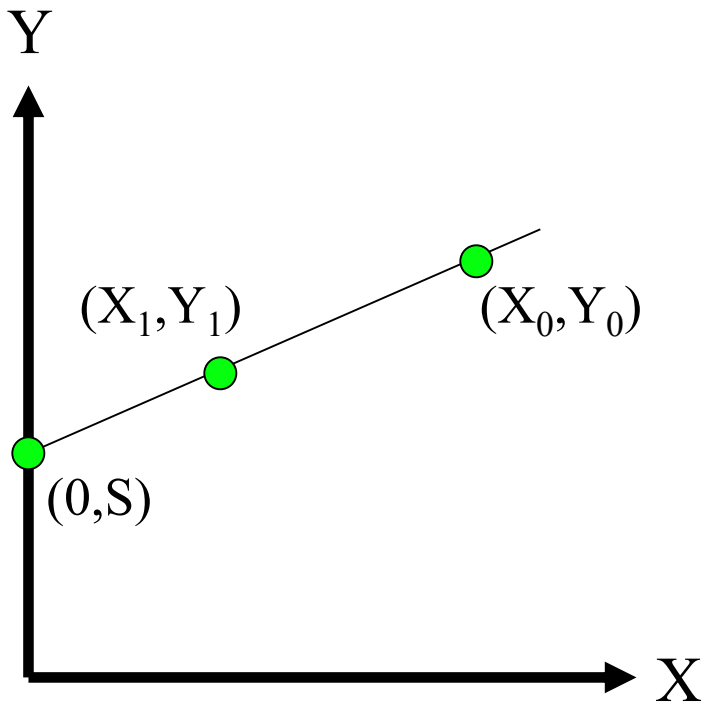
选择随机数 R_a ,提交 $h(A, R_a)$

- 清理垃圾邮件
- 我拒绝接收邮件，除非证实你花费了一定"精力" (如CPU时钟)来发送这个e-mail
 - 这里，精力==时钟周期
- 目标是控制e-mail的发送数量
 - 这种方法不能消除垃圾邮件
 - 相反，使垃圾邮件的发送代价更加昂贵

- 令M为email
- 令R为必须找到的值
- 令T为当前时间
- 发送者必须找到值R
 - $\text{hash}(M, R, T) = (00\dots 0, X)$
 - 使得hash值的前N位都是0
- 然后发送者发送(M, R, T)
- 接收方接收e-mail, 并确定
 - $\text{hash}(M, R, T)$ 的前面是N个0

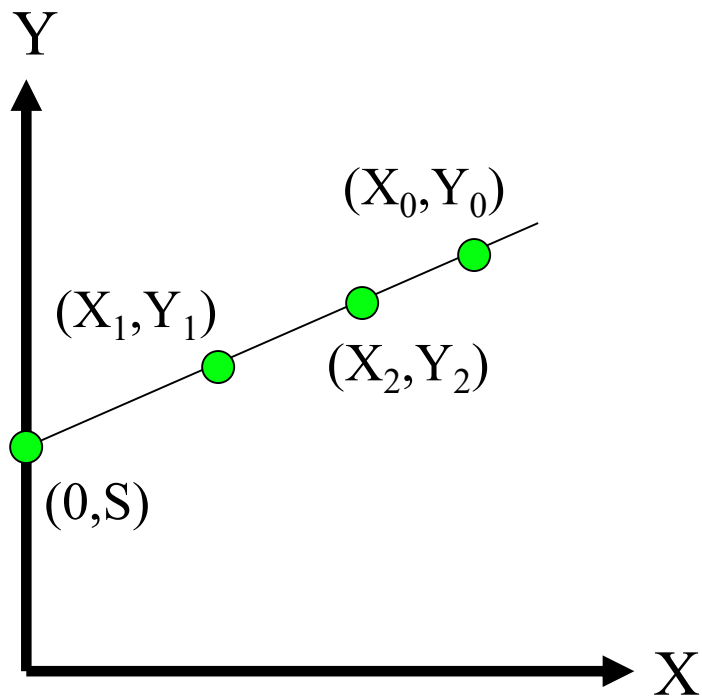
- 发送方: $\text{hash}(M, R, T)$ 的前面是 N 个 0
- 接收方: 确认 $\text{hash}(M, R, T)$ 的前面是 N 个 0
- 发送方工作量: 2^N 次 **hash** 计算
- 接收方工作量: 1 次 **hash** 计算
- 发送者的工作量根据 N 呈指数增长
- 不管 N 的取值如何, 接收方的工作量都是一样的
- 合理选择 N
 - 对于普通 **email** 用户而言工作量是可以接受的
 - 对于大量发送垃圾邮件的用户工作量是无法承受的

秘密共享



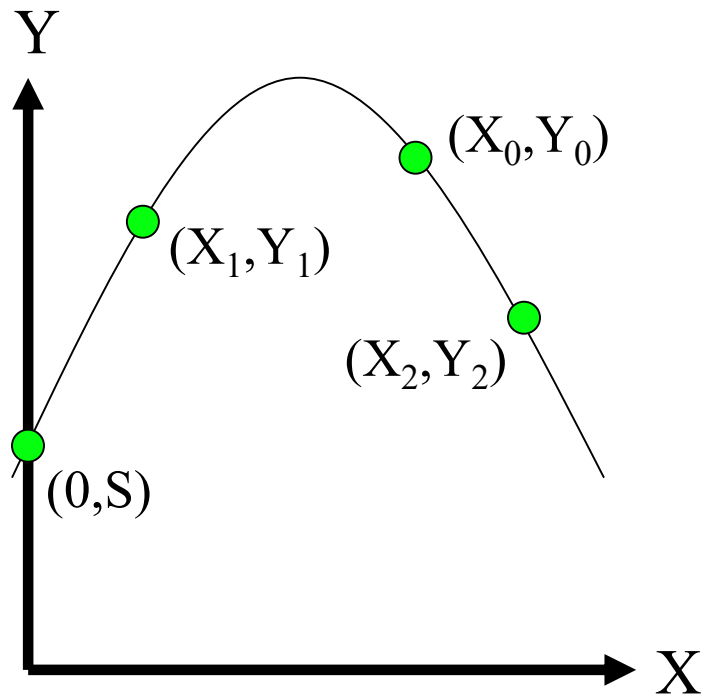
2 选 2

- 两点确定一线
- 分配 (X_0, Y_0) 给 Alice
- 分配 (X_1, Y_1) 给 Bob
- 然后 Alice 和 Bob 必须合作来找到秘密 S
- 同样适用于分离的模型
- 容易延伸到任意 $m \leq n$ 的 "n 选 m" 秘密共享方案



3选2

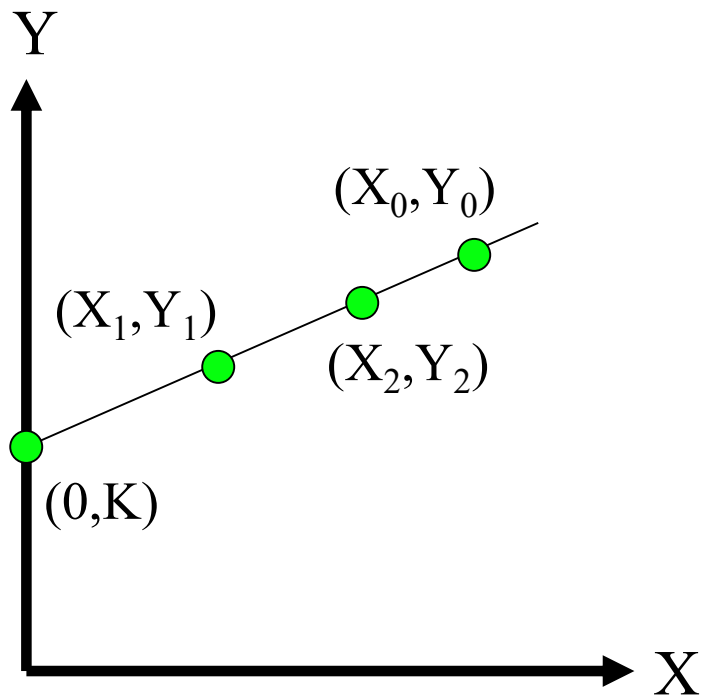
- 分配 (X_0, Y_0) 给 Alice
- 分配 (X_1, Y_1) 给 Bob
- 分配 (X_2, Y_2) 给 Charlie
- 然后 Alice, Bob 和 Charlie 中的任意两人合作就可以发现秘密 S
- 但是单人不能发现秘密 S
- "3选2" 方案



3选3

- ❑ 分配 (X_0, Y_0) 给 Alice
- ❑ 分配 (X_1, Y_1) 给 Bob
- ❑ 分配 (X_2, Y_2) 给 Charlie
- ❑ 3个点决定一条抛物线
- ❑ Alice, Bob和Charlie必须一起合作才能找到秘密S
- ❑ "3选3"方案
- ❑ 你能设计一个"4选3"方案吗?

















- **密钥托管** — 需要将密钥存储在某个地方
- 如果有法院授权，便可以使用密钥
- 密钥托管代理可能是不可信的，即便是**FBI**
- 我们可以使用密钥共享机制
 - 例如，三个不同的政府代理
 - 两个必须联合在一起才能恢复出**S**



















- 对称密钥是 K
- 分配点 (X_0, Y_0) 给 **FBI**
- 分配点 (X_1, Y_1) 给 **DoJ**
- 分配点 (X_2, Y_2) 给 **DoC**
- 两个代理必须联合在一起才能恢复出 K
- 没有任何一个代理可以单独获得 K

- 另一种形式的密码共享...
- Alice和Bob “共享” 一张图片
- 双方继续合作才能展示这张图片
- 没有人能单从Alice的部分或Bob的部分获取这个图片的任何信息
 - 这就是说两个部分都是必须的
- 这可能吗?

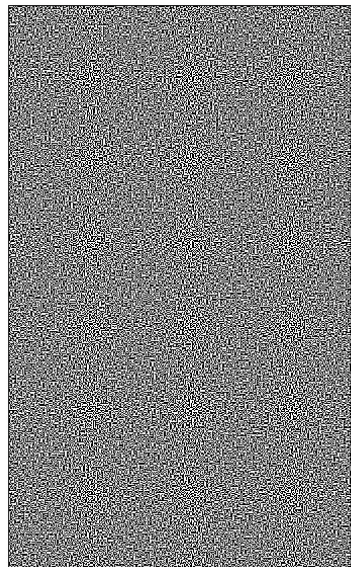
- 如何共享一个像素?
- 假设图像是黑白的
- 那么每个像素不是黑色就是白色的
- 我们像右图一样分割像素

	Pixel	Share 1	Share 2	Overlay
a.				
b.				
c.				
d.				

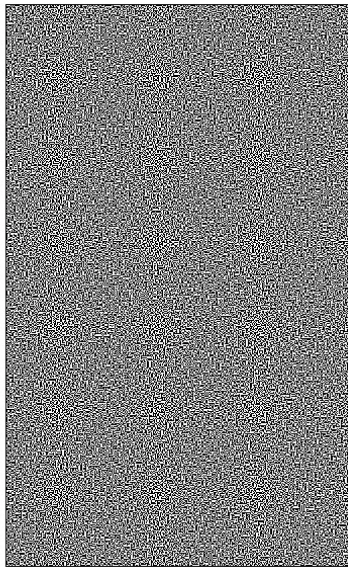
- 如果像素是白色的,为 Alice或Bob的部分随机选择a 或 b
- 如果像素是黑色的,随机选择c或者d
- 在一个“部分”中没有任何信息

	Pixel	Share 1	Share 2	Overlay
a.				
b.				
c.				
d.				

□ Alice的部分



Bob的部分



共享的图像



- ❑ 图形“密码”相比于普通密码如何?
- ❑ 在图像密码中, 没有密钥...
 - ❑ 或者, 两部分的图片就是密钥?
- ❑ 加密, 穷举搜索
 - ❑ 除了一次一密乱数本
- ❑ 图形密码中的穷举搜索?
 - ❑ 没有一种穷举搜索的可能!

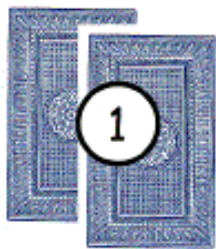
- 图形密码 — 不存在穷举搜索...
- 图形“密码”相比于普通密码如何?
 - 图形密码是“信息理论”安全的—其它秘密共享方案的实现
 - 相比于常规的加密,目标是使分析计算不可信
- 图形密码的秘密共享的一个例子
 - 通常意义上不是一个真正形式的密码

密码学随机数

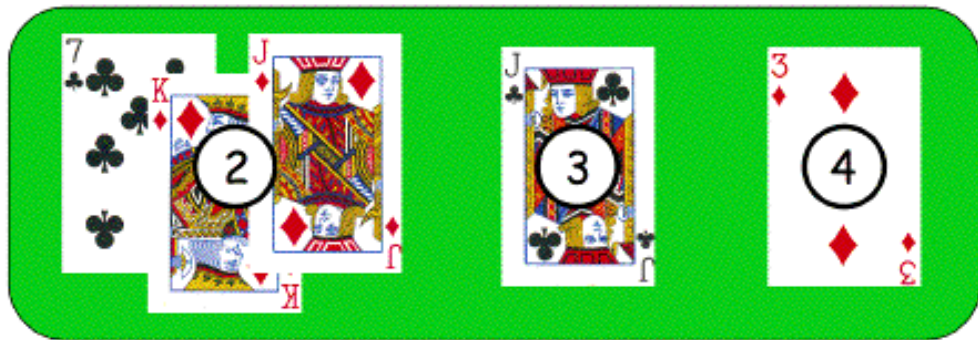
- 利用随机数产生密钥
 - 对称密钥
 - RSA: 素数
 - Diffie Hellman: 秘密值
- 随机数应用于特定场合
 - 有时一串序列就可以
 - 但有时要求必须随机
- 随机数可以应用于仿真和统计
 - 此时的随机数必须统计上随机

- ❑ 密码学随机数必须要求统计上随机
- ❑ 假设服务器产生如下对称密钥
 - ❑ Alice: K_A
 - ❑ Bob: K_B
 - ❑ Charlie: K_C
 - ❑ Dave: K_D
- ❑ Alice, Bob和Charlie都不喜欢Dave
- ❑ Alice, Bob和Charlie 联合起来不能推测密钥 K_D

- ❑ 德州扑克的在线版本
 - ASF软件公司



Player's hand



Community cards in center of the table

- ❑ 随机数用于洗牌
- ❑ 程序并未进行真正的随机洗牌
- ❑ 玩家能及时地通过判断整副牌的顺序来作弊

- ❑ 扑克共有 $52! > 2^{225}$ 种洗牌结果
- ❑ 扑克程序使用一个32位的“随机”二进制数来确定洗牌的结果
 - ❑ 仅产生 2^{32} 种洗牌结果
- ❑ 程序使用Pascal内置的伪随机数产生器(PRNG): Randomize()
- ❑ PRNG种子值由一个时钟毫秒的函数在每天午夜计算出来
- ❑ 一天中包含的毫秒数少于 2^{27}
 - ❑ 因此共有少于 2^{27} 种可能的洗牌结果

- ❑ 种子值依赖于午夜产生的时钟毫秒
- ❑ PRNG 产生的种子决定了每次洗牌的结果
- ❑ 通过使自己时钟与服务器的时钟同步，Trudy可以把不同洗牌结果的尝试次数减少为 $< 2^{18}$
- ❑ 这 2^{18} 种洗牌结果可以实时产生
 - ❑ 与上一盘的实际洗牌结果相比较
- ❑ 在前面首五轮赌注之后，Trudy就可以知道洗牌结果

- ❑ 扑克程序是一个极端例子
 - ❑ 但是普通的PRNG系统是可预测的
 - ❑ 在确定序列之前，必须知道有多少位的输出
- ❑ 密码学随机数是不可预测的
 - ❑ 例如RC4产生的密钥流
- ❑ 怎样产生初始随机值呢？
 - ❑ 密钥（某些情况下的种子值仍是问题，RC4）

- 真随机序列是很难定义的
- 熵是定义随机序列的一种好的方法
- 真随机源
 - 放射性衰退 — 虽然放射性电脑并不是普遍存在的
 - 硬件设备 — 市场上有很多
 - Lava lamp — 依赖于混沌产生的

- ❑ 通过软件寻找随机源
 - ❑ 软件是确定性的
 - ❑ 真随机数必须在代码之外产生
 - ❑ 鼠标的移动, 键盘的敲击, 网络的流量等
- ❑ 通过软件能得到高质量的随机序列
- ❑ 但序列的质量是受限制的
- ❑ 牢记这句话: “使用伪随机过程产生秘密值会导致伪安全”

- 访问控制
 - 认证— 谁能去做?
 - 授权— 是否被允许做某件事?
- 信息隐藏

2. 假如Alice想要对消息M实施签名,并将其发送给Bob。

a. 按照我们的标准化表示方式,请问Alice需要执行哪些运算?

$$S = [h(M)]_{\text{Alice}}$$

b. 请问,Alice需要发送给Bob什么信息?Bob又是如何对签名进行验证的?

M与S

$$\text{Bob验证 } h(M) = \{S\}_{\text{Alice}}$$

3.请回顾一下，我们之前给过的定义是：如果已知的对一个加密方案最有效的攻击就是穷举式密钥检索攻击，那么该加密方案可以以为是安全的。如果一个加密方案是安全的，并且其密钥的空间很大，那么已知的对其最有效的攻击将是计算上不可行的——对于一个实际的加密方案来说，这是理想的情况。然而，总是会有这样的可能性：出现了一种新的聪明的攻击方法，使得一个之前是安全的加密方案变成了一个不安全的加密方案。相比之下，Shamir的基于多项式的秘密共享体制是信息论意义上安全的，也就是意味着不存在捷径攻击的可能性。换句话说，秘密共享体制可以确保永远是安全的。

a.假如我们有一个“2 out of 2”型的秘密共享方案，其中由Alice和Bob共享一个秘密S。请问，为什么Alice无法根据她自己的那份秘密来确定有关共享秘密的任何信息呢？

一个点不能唯一地确定一条线

b.假如我们有一个“m out of n”型的秘密共享方案。那么，任何m-1个参与者组成的集合都不能够确定有关共享秘密S的任何信息，请说明这是为什么？

同上

4.假如你收到一封电子邮件，是自称为Alice的某人发送过来的。该电子邮件包括一个数字证书，含有如下内容：

$$M = (\text{“Alice”}, \text{Alice's public key}) \quad S = [M]_{CA}$$

其中CA是一个证书权威机构。

a.请问，你该如何验证这个签名？

用CA的公钥验证（如果信任CA），验证 $\{S\}_{CA}$ 是否与 M相同。

b.为什么你还需要花费力气去验证签名呢？

Trudy能够自己创建公/私钥对，然后发送（“Alice”，Trudy's public key），使得你用Trudy的公钥加密信息，那么只有Trudy能解密。



关注我，下节内容更精彩：
05：信息隐藏