网络空间安全数学基础

第2部分

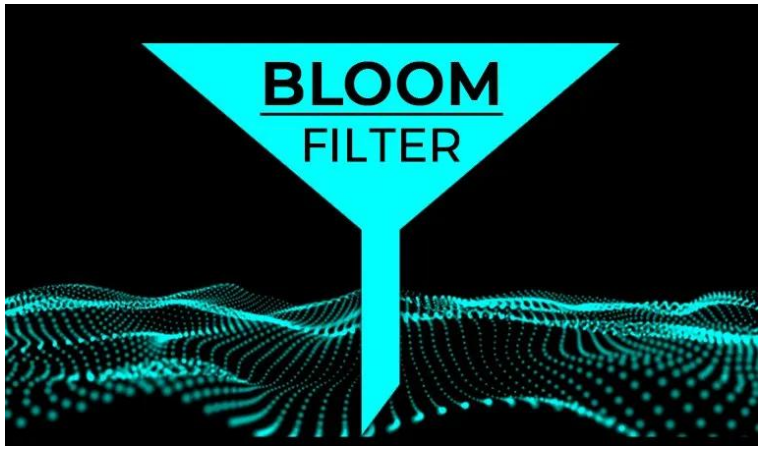
第1章 散列法及Bloom过滤器

教师信息

- 曾勇　刘志宏
- 办公地点：网安大楼A1143
- 电话：13892841398
- 邮箱：liuzhihong@mail.xidian.edu.cn

本章要点

- 散列法
- Bloom过滤器
- 信息指纹
- 相似性比较
- 距离度量



生日悖论

- Q1：教室里有30个同学，某两个人生日相同的概率大，还是没有两人生日相同的概率大？
- 30人全部生日不同的概率是：

$$\frac{365}{365}\cdot\frac{364}{365}\cdots\frac{336}{365}=(1-\frac{1}{365})(1-\frac{2}{365})...(1-\frac{29}{365})\doteq 0.2937$$

- 有两个人生日相同的概率为：

$$1-(1-\frac{1}{365})(1-\frac{2}{365})...(1-\frac{29}{365})\doteq 0.7$$

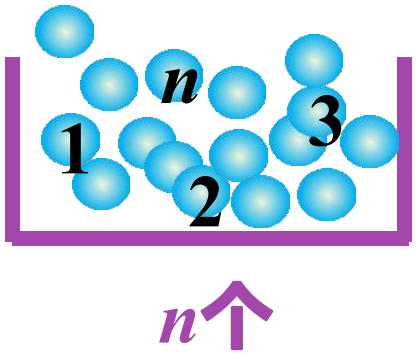
- 类似可以得到，教室里只需有23名同学，有两人生日相同的概率超过0.5

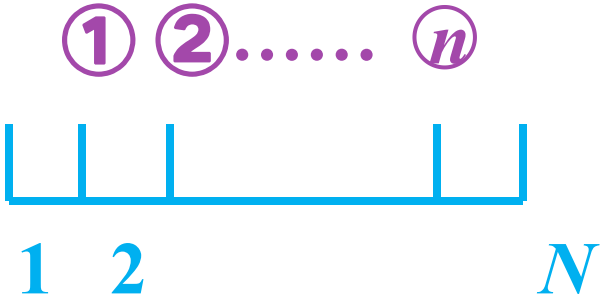
生日悖论

- Q2：在有n个同学的教室里，至少存在一个同学与Alice生日相同的概率是多少？

$$Pro = 1 - (1 - \frac{1}{365})(1 - \frac{1}{365})...(1 - \frac{1}{365}) = 1 - (\frac{364}{365})^n$$

- 当n = 84时，Pro = 0.5
- 生日悖论就是一个球和箱子模型





生日悖论

- 利用生日悖论求指数

$$y = g^x \bmod p \quad (p\text{为素数})$$

- Q3：这是个单向函数，已知*y*求*x*

利用穷举比较方法，需要最多*p*-1次运算能够找到，如果*p*很大，则不行。如$p = 2^{100}$

- 利用生日悖论，则穷举空间可降为

$$\sqrt{p} \doteq 2^{50}$$

生日悖论

- $y = g^x \bmod p$ （p为素数）
- Q3：这是个单向函数，已知*y*求*x*

Step 1：输入参数 y, g, p；

Step 2：创建两个表 TabA, TabB；

Step 3：随机生成一个 $1 \sim p-1$ 之间的整数 a，并计算 $A = g^a \pmod p$，将 a 和 A 填入 TabA 中；

Step 4：随机生成一个 $1 \sim p-1$ 之间的整数 b，并计算 $B = y \cdot g^b \pmod p$，将 b 和 B 填入 TabB 中；

Step 5：重复 Step 3 和 Step 4，直到 $n = \lfloor \sqrt{p} \rfloor$ 次；

Step 6：在 TabA 和 TabB 中找重，如果找到 TabA 中的某个 A 与 TabB 中的某个 B 相等，则记下相对应的 a 和 b；如果没找到，则回到 Step 2；

Step 7：计算 $x = a - b \pmod{p-1}$，即为解。

生日悖论

- $y = g^x \bmod p$ （p为素数）
- Q3：这是个单向函数，已知*y*求*x*

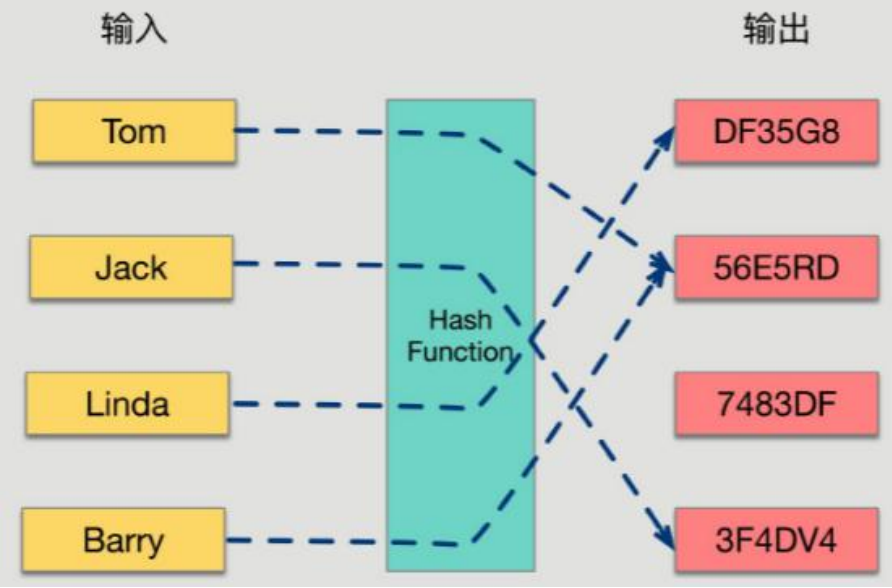
根据生日悖论，随机$\lfloor\sqrt{p}\rfloor$个 a 和随机$\lfloor\sqrt{p}\rfloor$个 b，出现 $A=B$ 的概率约为 50%。

也就是说，上面 Step 6 中有一半的可能性找到重项。再由

$$g^a = y \cdot g^b (\bmod p) = g^x \cdot g^b (\bmod p) = g^{x+b} (\bmod p) \Longrightarrow a = x + b (\bmod p - 1)$$

散列法

- Q：设计一个口令验证程序，保存一个不能接受的口令字典，阻止人们使用字典中容易被破译的口令。
- 方法1：顺序存储，按字母顺序存储不能接受的口令，并对字典进行二元搜索，检查口令是否不能接受。
  - 对m个单词，二元搜索的时间复杂度为$O(\log_2 m)$
- 方法2：散列存储 Hash

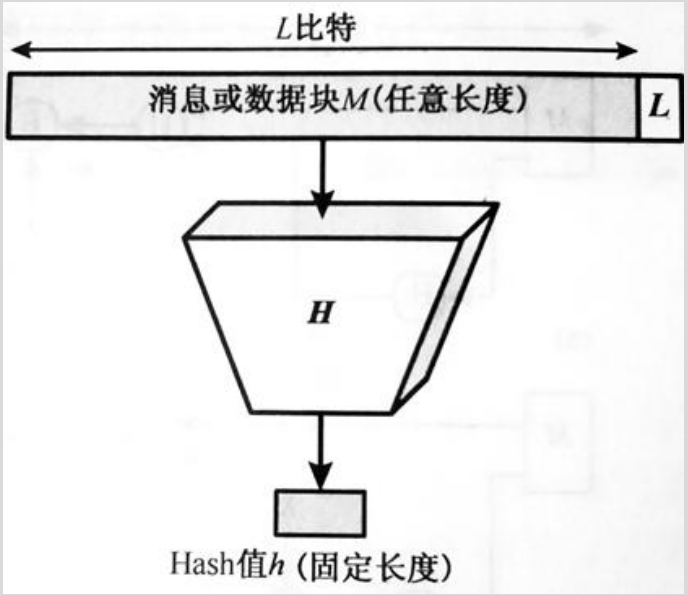


安全散列函数

- 为文件、消息或其他数据块产生“指纹”，浓缩任意长的消息M到一个固定长度的取值

  h = H(M)

- 通常假设hash函数是公开的且不使用密钥（MAC使用密钥）



安全散列函数的要求

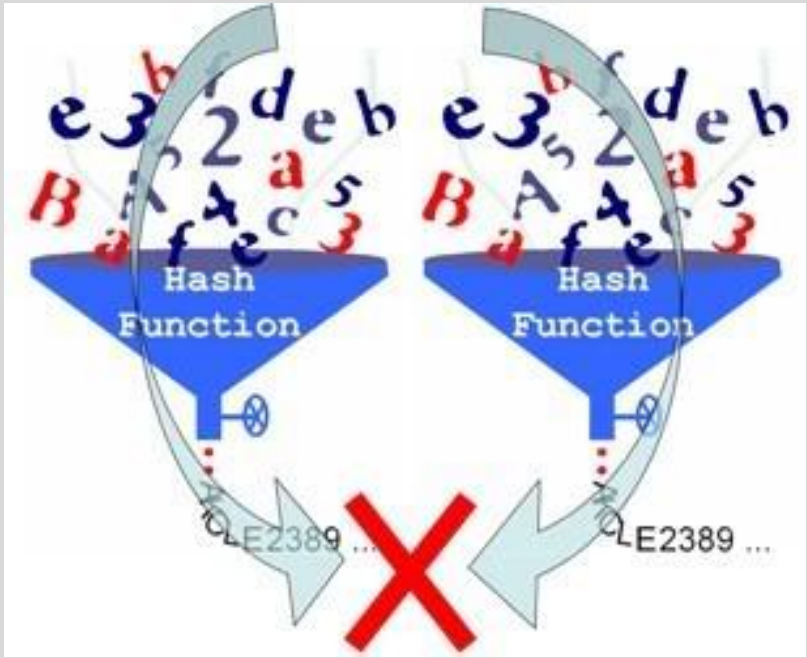
- 对于Hash函数h=H(x)，称x是h的原像，即把数据块x作为输入，使用Hash函数H得到h。

- 因为H是多对一映射，所以对于任意给定的Hash值h，对应有多个原像。如果满足x不等于y且H(x)=H(y)，则称出现**碰撞**(collision)。使用Hash函数的目的是保证数据完整性，碰撞是不希望轻易能找到的。

假设Hash码的长度是n位，函数H的输入消息或数据块长度是b位，且b>n。那么消息的全部可能取值的个数为2^b, Hash值的全部可能取值的个数为2^n，平均每个Hash值对应$2^{b/n}$个原像。

安全散列函数的要求

- 单向性：对于预先给定的Hash值找不到对应的数据块
- 抗碰撞性：找不到不同的数据块对应相同的Hash值



安全散列函数的要求

- 1.**输入长度可变**：H可以应用于任意大小的数据块
- 2.**输出长度固定**：H产生固定长度的输出
- 3.**效率高**：对任意给定的明文x，计算H(x)容易，可由硬件或软件实现
- 4.**抗原像攻击（单向性）**：对任意给定的散列码h，找到满足H(x)=h的x，在计算上不可行，单向性
- 5.**抗第二原像攻击（抗弱碰撞攻击）**：对任何给定分组x，找到满足y≠x且H(x)=H(y)的y，在计算上不可行，抗弱碰撞性
- 6.**抗碰撞攻击（抗强碰撞攻击）**：找到任何满足H(x)=H(y)的偶对(x, y)，在计算上不可行，抗强碰撞性
- 7.**伪随机性**：H的输出满足伪随机性测试标准

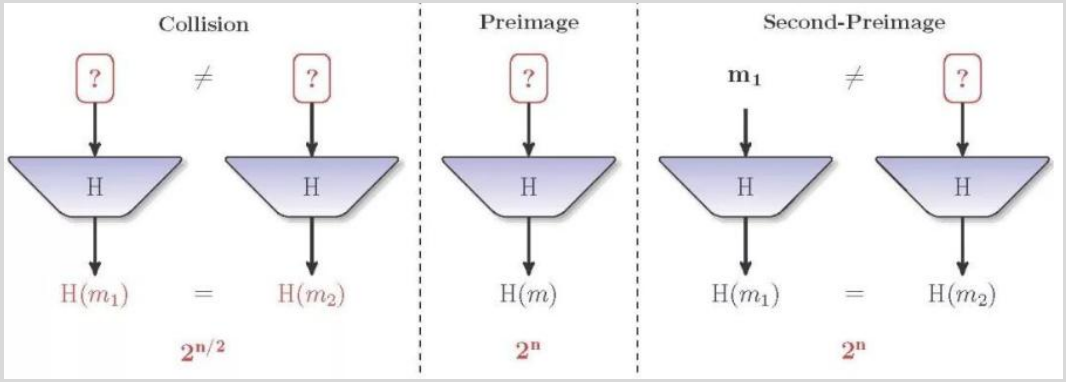
安全散列函数的要求

- 条件1, 2, 3，4是所谓单向性问题(One-way)
- 条件5、6是对使用的哈希值的数字签名方法所做的安全保障，否则攻击者可由已知的明文及相关的数字签名任意伪造对其他明文的签名
- 条件6主要用于防范所谓的生日攻击法
- 能满足条件1-5的，称为**弱哈希函数**(Weak Hash Function)
- 能同时满足条件6的，称为**强哈希函数**(Strong Hash Function)
- 应用在数字签名上的哈希函数必须是强哈希函数

散列函数的安全性

- 蛮力攻击法
  - 完全依赖于算法生成的散列码长度

抗原像	2^n
抗第二原像	2^n
抗碰撞	$2^{n/2}$



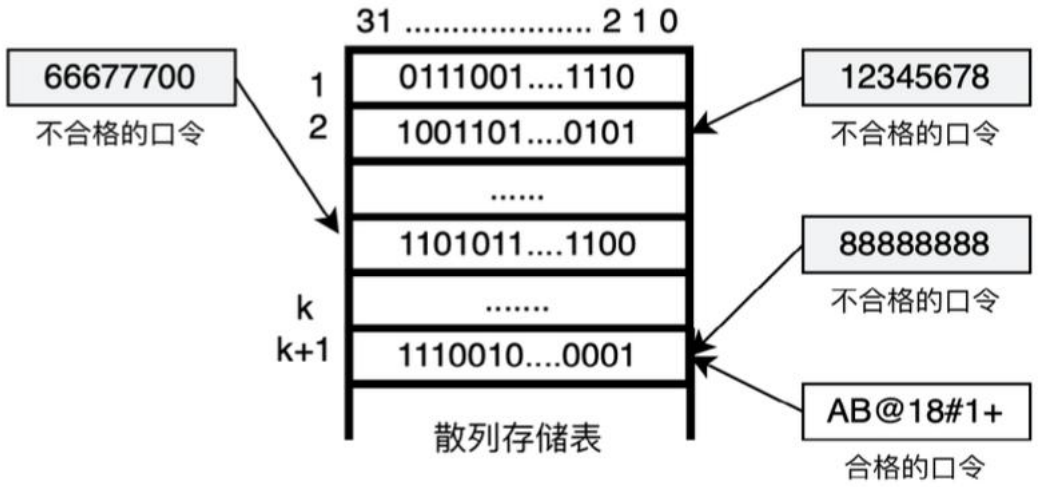
- 密码分析法
  - 寻找利用了该算法在逻辑上的缺陷

2004年8月中国密码学家王小云公布了寻找MD5碰撞的新方法。利用该方法用普通微机数分钟内即可找到MD5的碰撞，128比特的散列长度不够。对于160比特的散列长度，同样的搜索机器要花费超过4000年的时间才能找到一个冲突。在当今的技术下，搜索时间可能会缩短许多，所以160 比特的散列长度现在看来也有些不可靠。

网络与信息安全学院
School of Cyber Engineering

散列法

- Q：设计一个口令验证程序，保存一个不能接受的口令字典，阻止人们使用字典中容易被破译的口令。
- 方法2：散列存储 Hash
  - 假设口令为8个ASCII字符，64bit，用散列函数将每个口令映射到一个32bit的二进制串，称为信息指纹。然后将指纹保存在一个排序表中。
  - 检测口令时，首先计算口令的指纹，并在表中查找。如果指纹在表中，则判定该口令不能接受。

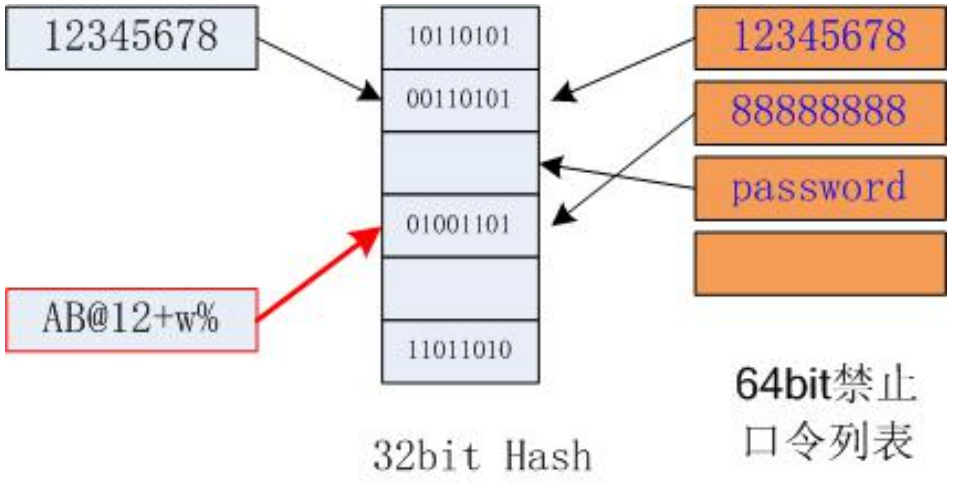


散列法

- 假阳性：一个合格的口令被判定为不合格。
- 设S={s_1, s_2,…, s_m}为不允许的口令，口令指纹为b比特，对于一个可以接受的口令，假阳性的概率为：

$$1-(1-\frac{1}{2^b})^m \geq 1-\exp\{-\frac{m}{2^b}\}$$

- 如果有2^{16} = 65536个词，用32比特指纹，则假阳性的概率小于1/65525.



Bloom过滤器

- Q：过滤垃圾邮件
- 全世界有几十亿个发垃圾邮件的地址，将这些地址都存储下来，需要大量的存储空间。
- 如果采用散列法，一个Email地址对应一个8字节的信息指纹，则存储一亿个Email地址需要1.6GB的内存，十亿个地址，需要16GB的存储空间。
- 下面看一种Bloom过滤器，只需散列表的1/8到1/4的大小，就可以解决同样的问题。

Bloom过滤器

- Bloom Filter是1970年由布隆提出的。
- 是一个很长的二进制向量和一系列随机映射函数。
- 布隆过滤器可以用于检索一个元素是否在一个集合中。

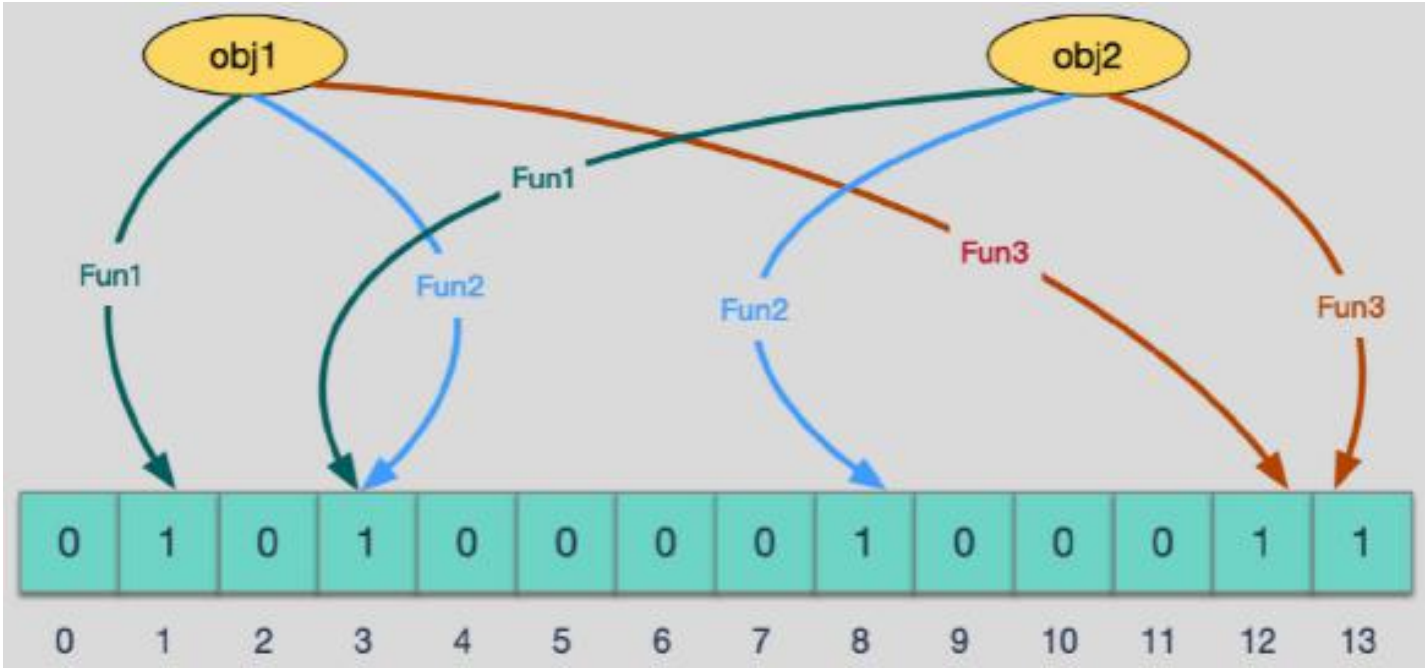
- 优点：空间效率和查询时间都远远超过一般的算法。
- 缺点：有一定的误识别率和删除困难。

Bloom过滤器

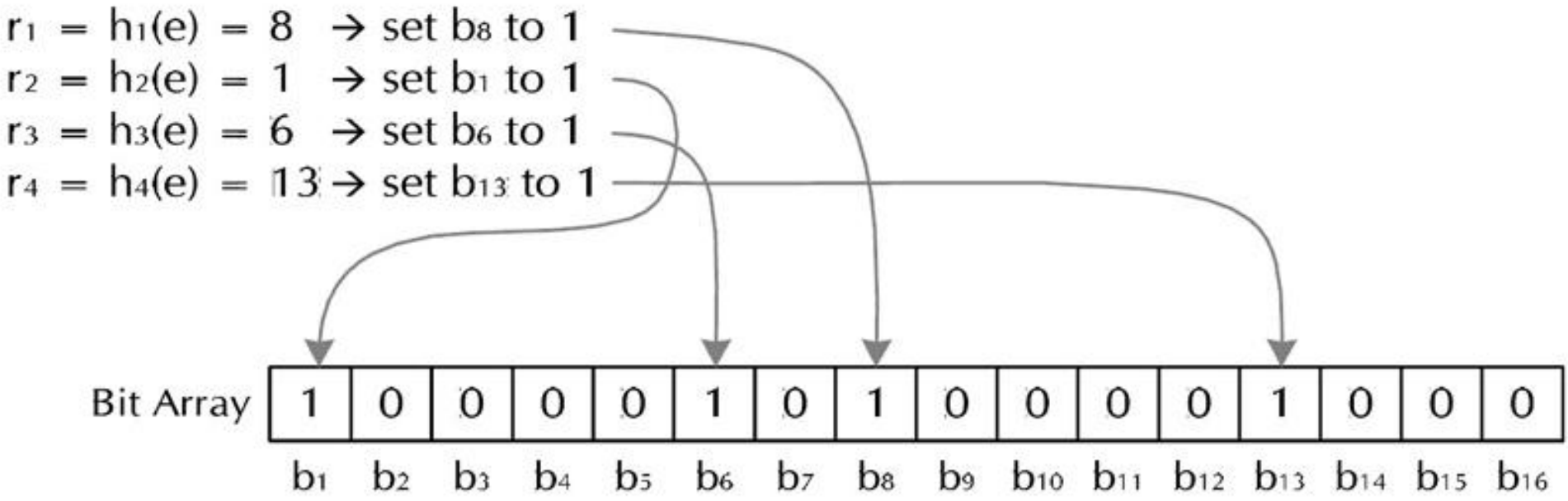
- Bloom Filter 是由一个固定大小的二进制向量或者位图（bitmap）和一系列映射函数组成的。
- 在初始状态时，对于长度为 m 的位数组，它的所有位都被置为0。

Bloom过滤器

- 当有变量被加入集合时，通过 K 个映射函数将这个变量映射成位图中的 K 个点，把它们置为 1（假定有两个变量都通过 3 个映射函数）。

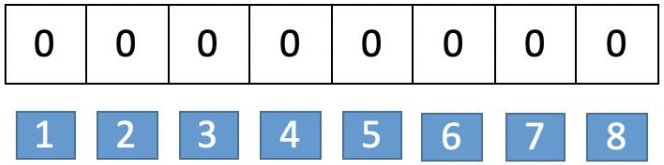


Bloom过滤器

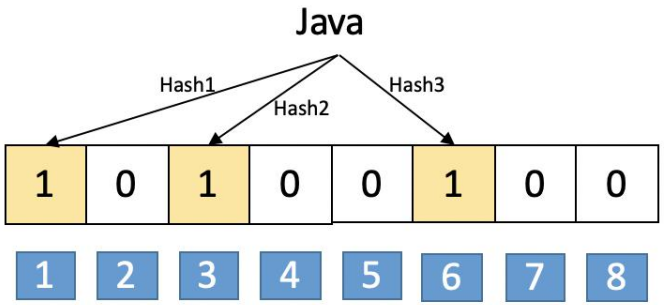

r1 = h1(e) = 8 → set b8 to 1
r2 = h2(e) = 1 → set b1 to 1
r3 = h3(e) = 6 → set b6 to 1
r4 = h4(e) = 13 → set b13 to 1
Bit Array
1 0 0 0 0 1 0 1 0 0 0 0 1 0 0 0
b1 b2 b3 b4 b5 b6 b7 b8 b9 b10 b11 b12 b13 b14 b15 b16

Bloom过滤器

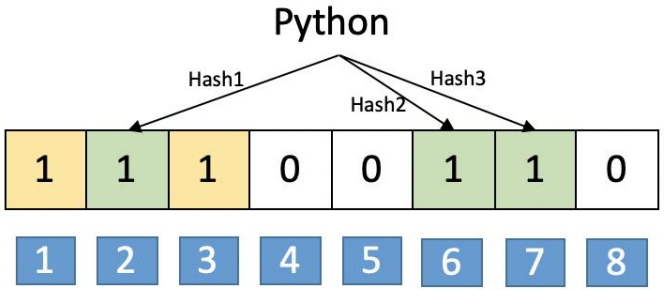
①



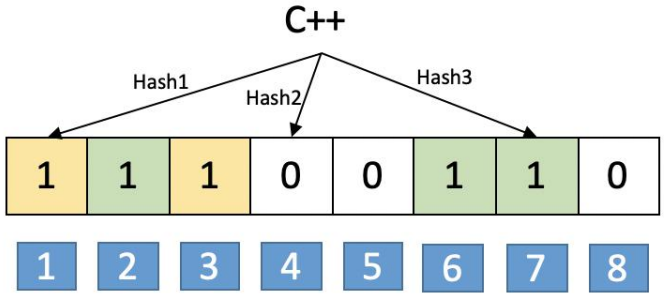
②



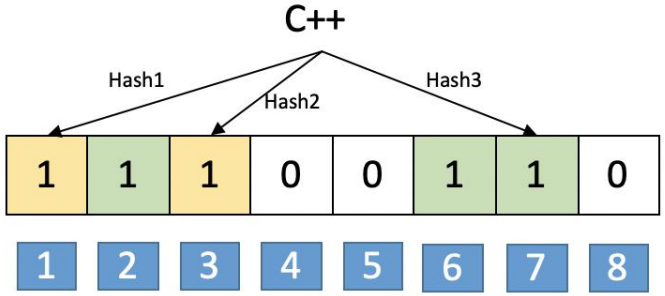
③



④

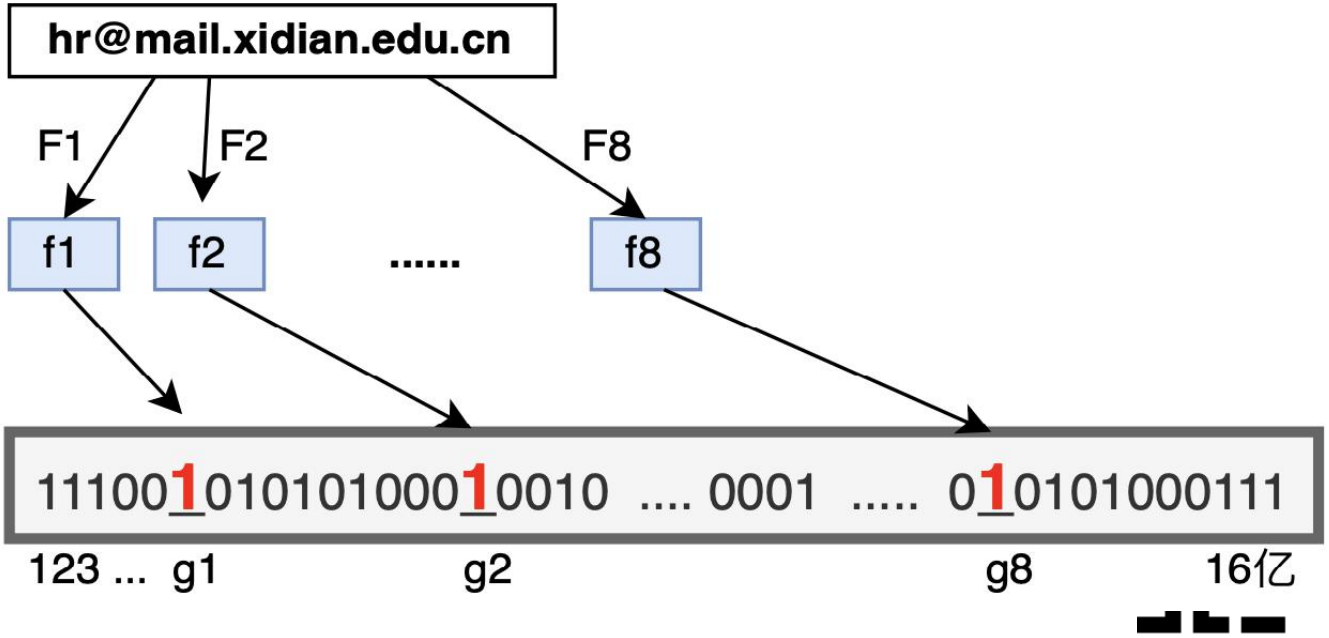


⑤



Bloom过滤器

- Q：过滤垃圾邮件
- 假设存储1亿个Email地址，先建一个16亿比特的全零向量，对每个Email地址，用8个不同的随机数产生器（F1,...F8）产生8个信息指纹（f1,...,f8）。再用一个随机数产生器G把8个信息指纹映射到1-16亿中的8个自然数g1,...,g8，把这8个位置的比特全部置为1。



Bloom过滤器

- 查询某个变量的时候，只要看看这些点是不是都是 1 就可以大概率知道集合中有没有它。
  - 如果这些点有任何一个 0，则被查询变量一定不在；
  - 如果都是 1，则被查询变量很可能存在
- 为什么说是可能存在，而不是一定存在呢？那是因为映射函数本身就是散列函数，散列函数是会有碰撞的。
- **误判率**
  - 布隆过滤器的误判是指多个输入经过哈希之后在相同的bit位置1了，这样就无法判断究竟是哪个输入产生的，因此误判的根源在于相同的 bit 位被多次映射且置 1。

错误率估计

- 当集合S={x_1, x_2,…,x_n}所有元素都被k个哈希函数映射到m位的位数组中时，这个位数组中某一位还是0的概率是：

$$p = (1 - \frac{1}{m})^{kn} \approx e^{-kn/m}$$

- $(1-p)^k$表示k次哈希都刚好选中1的区域，即false positive rate:

$$f = (1 - e^{-kn/m})^k = (1-p)^k \approx (1 - e^{\frac{-nk}{m}})^k$$

m/n	K=5	K=6	K=7	K=8
10	0.00943	0.00844	0.00819	0.00846
20	0.00053	0.000303	0.000196	0.00014
30	8.53e-05	3.55e-05	1.69e-05	9.01e-06

Bloom过滤器设计

- **Hash Function的数目 k**
  - Hash Function数目k的增加可以减小误判率P(true)，但是随着Hash Function数目k的继续增加，反而会使误判率P(true)上升，即误判率是一个关于Hash Function数目k的凸函数。
  - 通过m和n来确定最优的Hash Function数目k为

$$k = \frac{m}{n} ln2 \approx 0.7 \frac{m}{n}$$

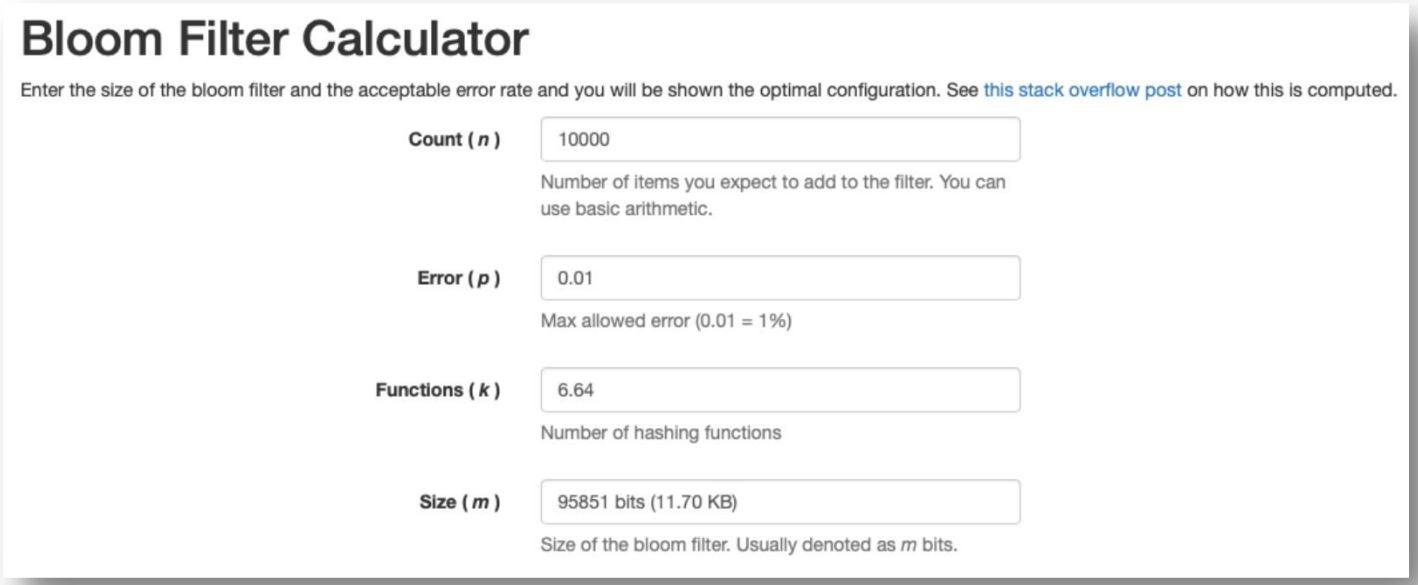
- **BitArray数组的大小 m**

$$m = -\frac{nlnP(true)}{(ln2)^2}$$

过滤器设计

- **布隆过滤器在线计算的网址：**

- https://krisives.github.io/bloom-calculator/



Bloom过滤器

- 特性
  - 一个元素如果判断结果为存在的时候元素不一定存在，但是判断结果为不存在的时候则一定不存在。
  - 布隆过滤器可以添加元素，但是不能删除元素。因为删掉元素会导致误判率增加。
- 优点
  - 相比于其它的数据结构，布隆过滤器在空间和时间方面都有巨大的优势。布隆过滤器存储空间和插入/查询时间都是常数 O(K)，
  - 散列函数相互之间没有关系，方便由硬件并行实现。
  - 布隆过滤器不需要存储元素本身，在某些对保密要求非常严格的场合有优势。

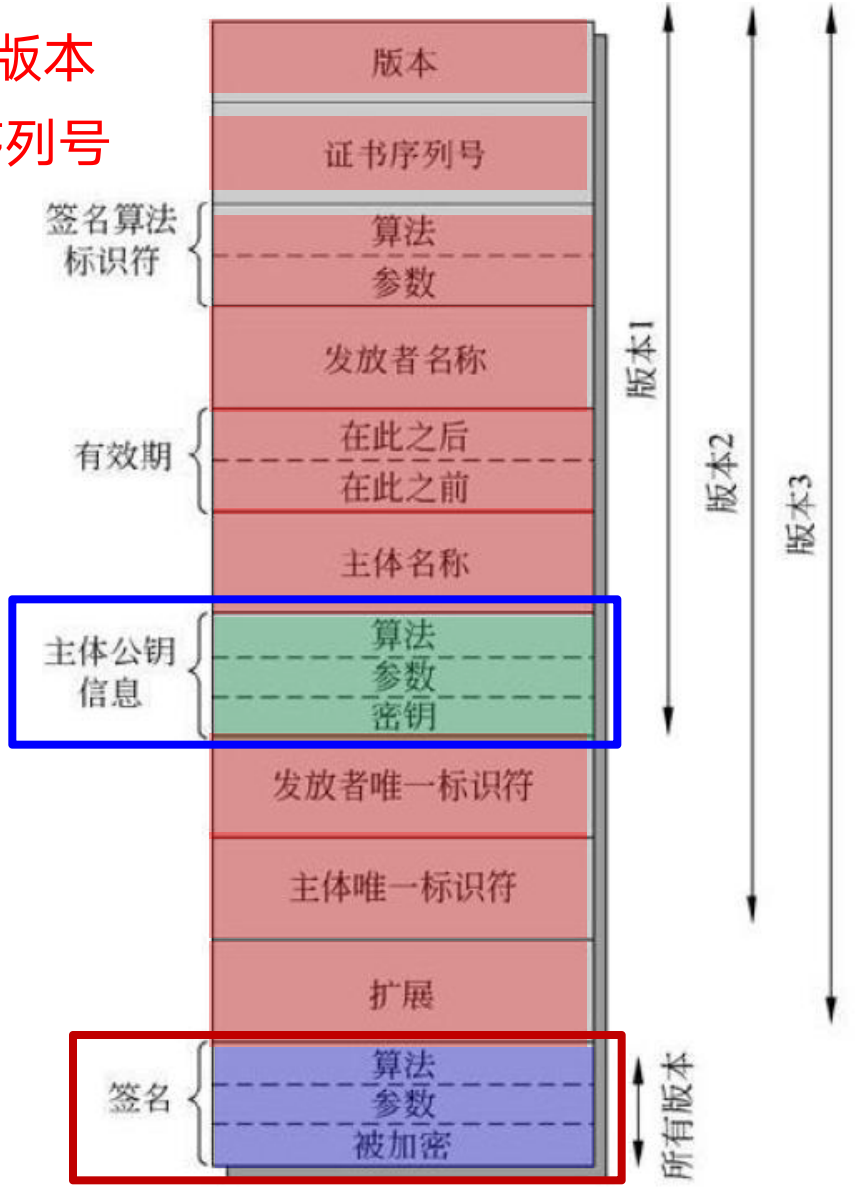
Bloom过滤器使用场景

- 利用它可以快速地解决项目中一些比较棘手的问题。如网页 URL 去重、垃圾邮件识别、大集合中重复元素的判断等问题。
  - 数据库。 Google Bigtable，HBase 和 Cassandra 以及 Postgresql 使用BloomFilter来减少不存在的行或列的磁盘查找。避免代价高昂的磁盘查找会大大提高数据库查询操作的性能。
  - 判断用户是否阅读过某视频或文章，比如抖音或头条，当然会导致一定的误判，但不会让用户看到重复的内容。
  - WEB拦截器，如果相同请求则拦截。用户第一次请求，将请求参数放入布隆过滤器中，当第二次请求时，先判断请求参数是否被布隆过滤器命中，可以提高缓存命中率。Squid 网页代理缓存服务器在 cache digests 中就使用了布隆过滤器。Google Chrome浏览器使用了布隆过滤器加速安全浏览服务
  - Venti 文档存储系统也采用布隆过滤器来检测先前存储的数据。
  - SPIN 模型检测器也使用布隆过滤器在大规模验证问题时跟踪可达状态空间。

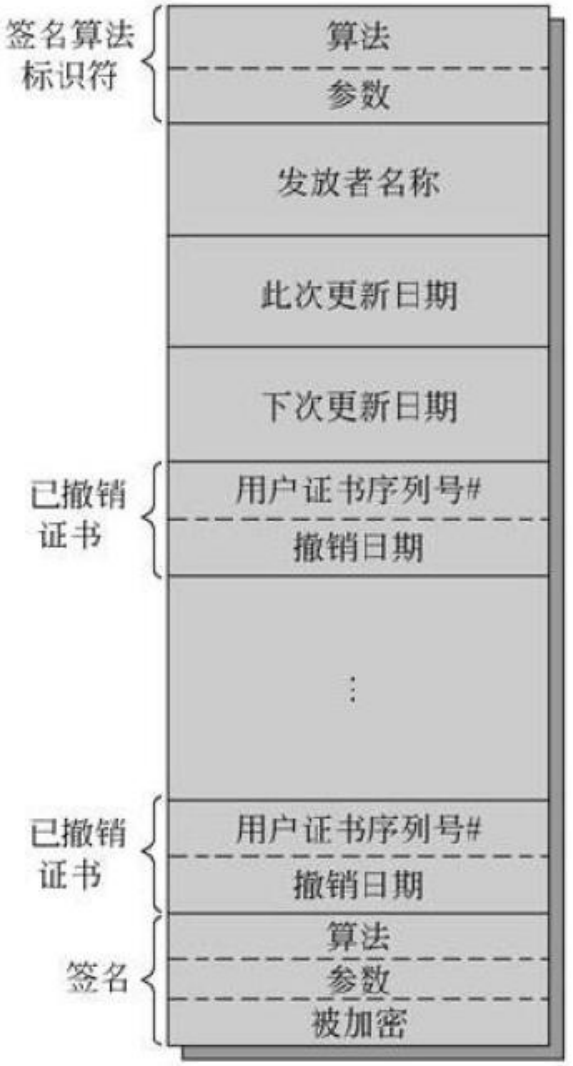
X.509 Certificates

版本

序列号



(a) X.509证书



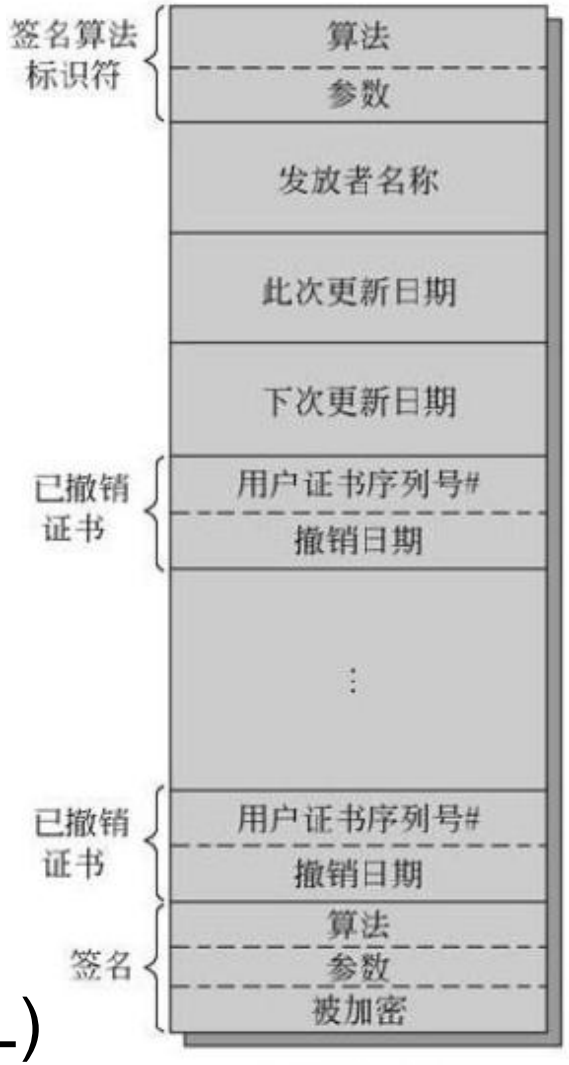
(b) 撤销证书列表

网络与信息安全学院
School of Cyber Engineering

证书的撤销

证书一旦发出，是不可能收回，只能申请注销。所谓注销，就是要求当初颁发这张证书的单位（CA）出一张告示，宣布某张证书已经被注销作废，警告有关的交易者注意，不要再使用与这张证书绑定的公钥。

- 每个证书都有一个有效期，到期失效
- 可能因为如下原因提前撤销证书
  - 用户的私钥被认为不安全了
  - 用户不再信任该CA
  - CA的证书被认为不安全了
- 每个CA维护一张证书撤销列表
  - the Certificate Revocation List (CRL)
- 用户应去CRL看看某个证书是否有效



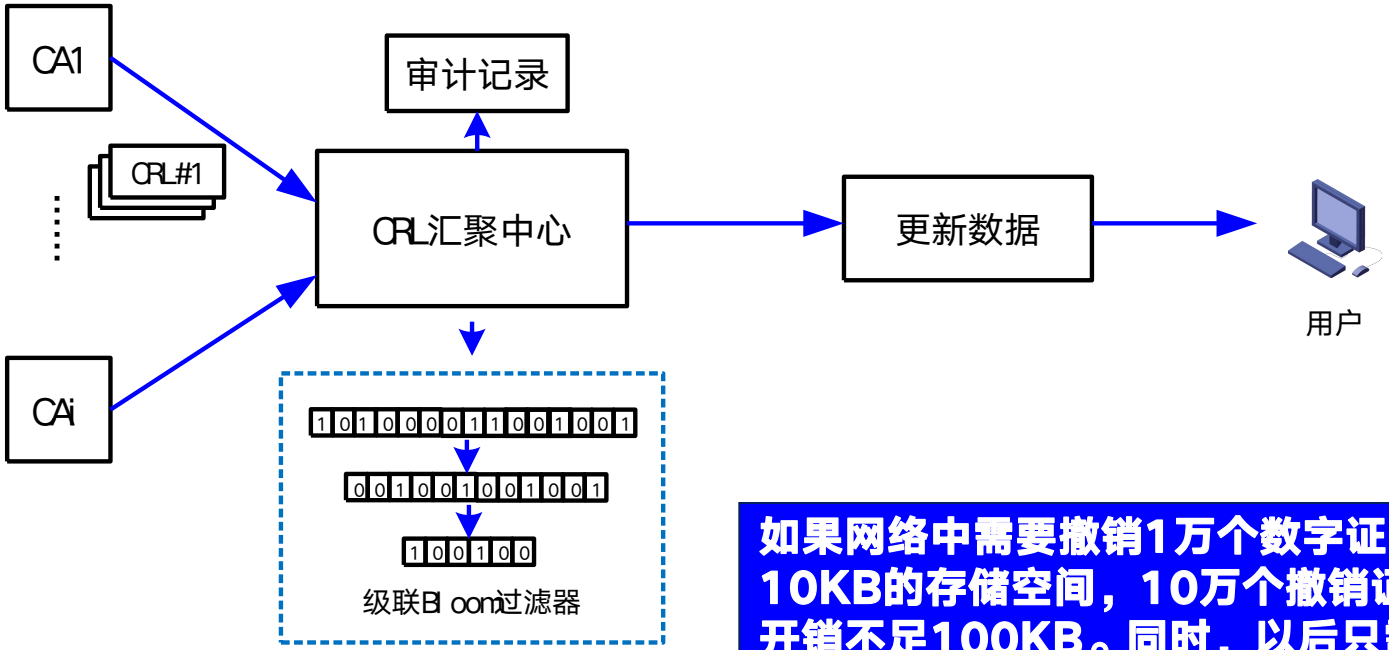
(b) 撤销证书列表

证书的撤销

- 在Internet中，由于用户和CA机构数量众多，证书撤销管理比较复杂。以往用户采用按需下载证书撤销信息模式（Pull Model），但由于网络时延、带宽限制等因素，许多网络应用转而采用推送证书撤销列表的模式（Push Model），如Google的CRLSets和Mozilla的OneCRL。
- CRLSets和OneCRL针对每一条撤销证书，需要存储111比特和1928比特信息，如果需要撤销12.7M个已经签发的数字证书，则分别需要166MB和2.9GB的存储空间。据统计，2017年1月17日一天，CRLSet和OneCRL就分别新增了14436条和357条撤销记录。

证书的撤销　CRLite的证书撤销管理机制

- 服务器端的CRL汇聚中心负责收集原始吊销的数字证书，为客户端生成撤销证书的更新文件，并定期推送到用户端。用户不用在线查询CA服务器。



数字证书撤销管理

如果网络中需要撤销1万个数字证书，平均只需10KB的存储空间，10万个撤销证书所需的存储开销不足100KB。同时，以后只需定期推送一次新增撤销证书的更新文件，如果新增的撤销证书不足1千，则更新文件不到1KB。

Bloom过滤器的改进

- 关于元素删除的问题，一个改良方案是对bloom filter引入计数，原来每个bit空间就要扩张成一个计数值，空间效率上降低了。
- Cuckoo Filter 布谷鸟过滤器
  - 由于布谷过滤器本身的思想就源自于布谷散列。产生布谷散列表的一个重要背景是人们对于球盒问题的分析：给定N个球，随机的放在N个盒子里，在装球最多的盒子里，球的个数的期望是多少？答案是Θ(logN/loglogN) 。后来人们发现：每次放球的时候，如果随机选择两个盒子，将球放到当时较空的那个盒子里，那么这个期望就变成了Θ(loglogN)，这个界小于之前的界，这给了布谷散列表作者启发。

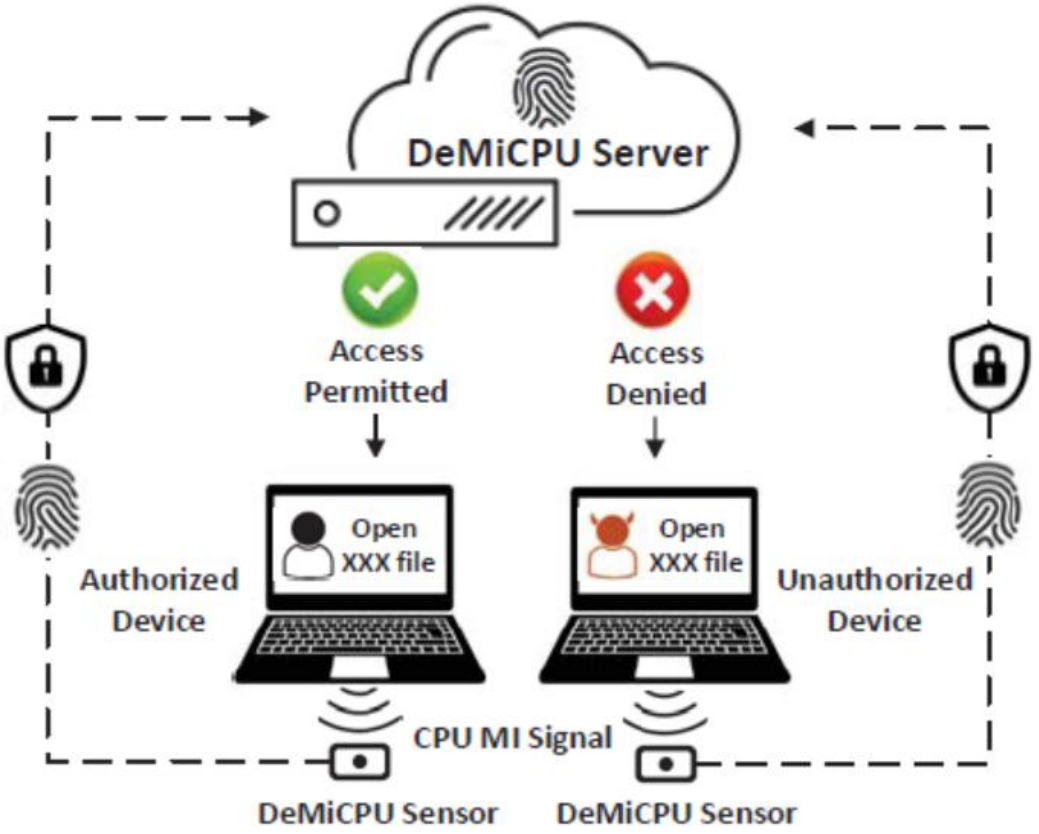
信息指纹

- 通过提取一个信息的特征，通常是一组词或者一组词+权重，然后根据这组词调用特别的算法例如MD5算法，将之转化为一组代码，这组代码就是标识这段信息的指纹。

设备指纹

- 计算机的CPU模块（包括CPU芯片和供电模块）存在微小差异，这种差异导致磁感应（ magnetic induction ）信号不同，可以作为设备指纹，难以修改或模仿。

设备指纹
Device Fingerprinting

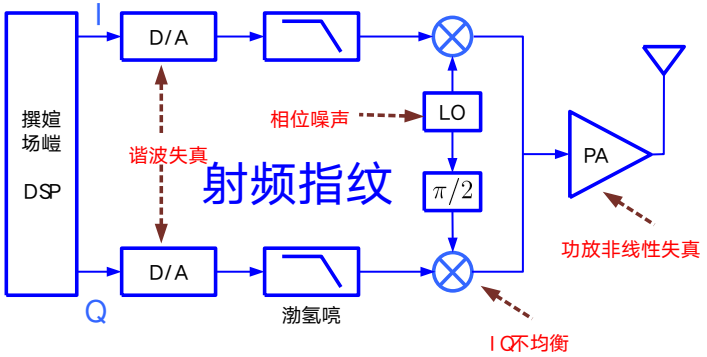


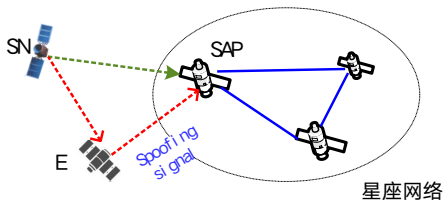
DeMiCPU: Device Fingerprinting with Magnetic Signals Radiated by CPU, ACM CCS'19

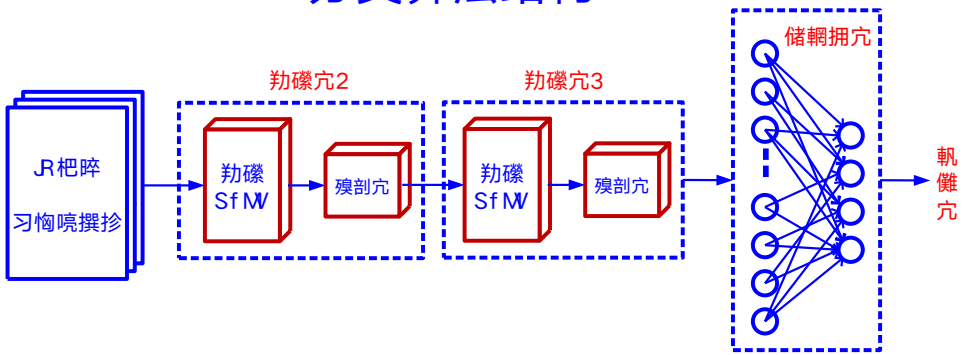
网络与信息安全学院
School of Cyber Engineering

设备认证

- 在网络安全等级要求较高的场合，不仅需要保证用户的可信，还需要保证接入设备的可信。（设备证书）
- 设备指纹提取分为两大类：射频指纹和物理指纹。射频指纹主要是指与设备的射频部件和信号相关的特征，如载波频率偏移、IQ增益不平衡等。物理指纹是指与设备的其他部件相关的物理特性，如卫星节点的多普勒频移、轨道数据、独特的传感器特性等。







卫星节点多普勒频移、轨道数据、物理指纹

浏览器指纹

Fingerprint

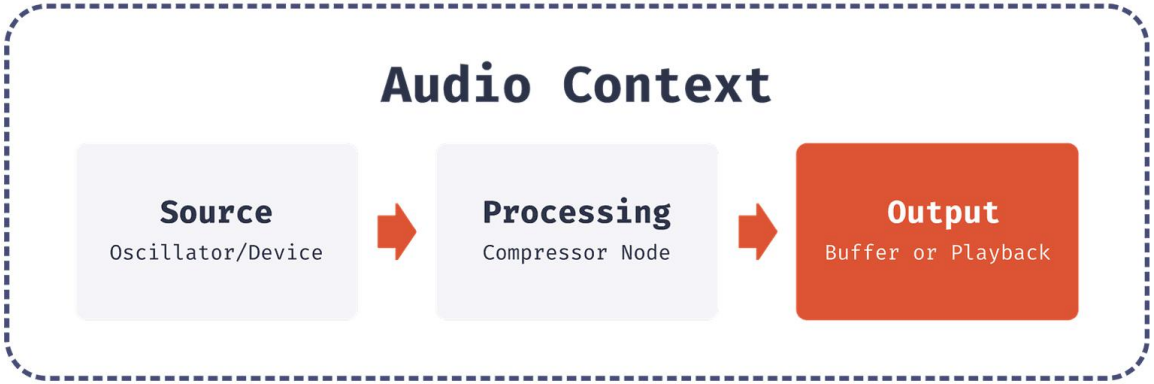
Text Demo | Visual Demo

```
1  Welcome to Fingerprint, Visitor rsLf7Q35BXikfjOHVwqy 👋!
2
3  It's great to have you here.
4  This is your second visit using Safari 🌐!
5
6  Your current IP suggests you're in Hong Kong, Hong Kong
7  Are you sure you're not using a VPN 🛡️?
8
9  You're not in Incognito mode right now.
10 We can consistently identify you across browsing sessions
11 incognito or not, even if you clear your cookies 🍪!
12
13 Curious to explore more? Sign up for a free trial right now!
14 Looking forward to your next visit 🧡!
```

网络与信息安全学院
School of Cyber Engineering

2024-03-22 Apple vs. Fingerprint

- audio fingerprinting：利用浏览器的Audio API来渲染一段音频（人听不到的频段，通过OfflineAudioContext接口，调用振荡器oscillator来产生一段特定的波形）。在不同浏览器、不同CPU等环境下，这段音频的特征是高度环境相关而且稳定不变的，就可以被用来标记特定的设备及用户。
- 为了对抗这种采样，作为最重视用户隐私的Apple自然要出手了，在Safari 17中引入了一套保护机制
- 具体针对音频信号，Safari 17会往每段音频里面加入一些扰动



集合相等的判断

- Q：如何判定两个集合相同？
- 方法1：对集合中的元素一一做比较。时间复杂度O(n^2)
- 方法2：将两个集合的元素分别排序，然后顺序比较。时间复杂度O(nlogn)
- 方法3：计算两个集合的信息指纹，直接比较。

$$S = \{e_1, e_2, \cdots, e_n\}$$

$$FP(S) = FP(e_1) + FP(e_2) + \cdots + FP(e_n)$$

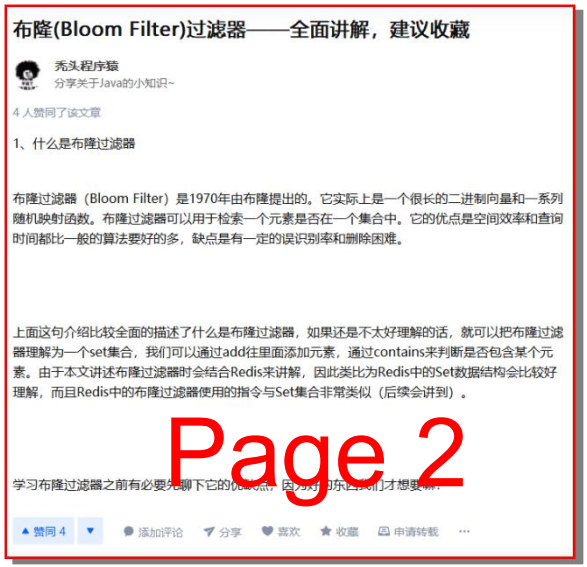
  - 如果两个集合相同，则指纹一定相同；如果不同，则指纹相同的概率很小。

集合近似相等的判断

- Q：如何判定两个集合基本相同？
- 给定两篇文章，如何判定相似程度？是否一篇文章抄袭另一篇文章？
- Simhash相似哈希算法
  - Simhash是在2002年的时候由Moses Chariker 提出。
  - Simhash用来计算网页的信息指纹。Simhash当初比较早些的时候用在了Google的网页爬虫中，用于重复网页的去重。

Simhash

- Q：比较两个网页的相似度？
- 比较两个网页的相似度，那么比较的关键就是里面的各种词，假设有t1，t2，t3等。
- 将文章切割成小片段，挑选特征词集合，计算每个特征词的权重。可以用分词系统来处理，统计词频，用词频作为权重值。
- Simhash的作用就是把这些值进行一个汇总，得到一个整个网页的哈希值。



Simhash

- 假设一个网页有若干个词 t1, t2, …, t8
- 每个词的权重（如TF-IDF）分布为 w1,w2,…,w8
- 每个词的哈希值指纹为8比特，如Hash(t1)=10100110
- Step 1.扩展：将8位二进制的指纹扩展为8个实数 r1, r2,…, r8
- 规则很简单，就是根据二进制位，根据1加0减的规则，做相应的权重操作。然后是遍历第二个词，操作的过程一样。
- 例如，t1的指纹

Hash(t1)=10100110

$$
\begin{array}{lrcl}
1, & r1 & = & 0 + w1 \\
0, & r2 & = & 0 - w1 \\
1, & r3 & = & 0 + w1 \\
0, & r4 & = & 0 - w1 \\
0, & r5 & = & 0 - w1 \\
1, & r6 & = & 0 + w1 \\
1, & r7 & = & 0 + w1 \\
0, & r8 & = & 0 - w1
\end{array}
$$

Simhash

- Hash(t1)=1010,0110
- Hash(t2)=0001,1001

$$
\begin{array}{llcl}
1, & 0 & r1 & = & 0 + w1 - w2 \\
0, & 0 & r2 & = & 0 - w1 - w2 \\
1, & 0 & r3 & = & 0 + w1 - w2 \\
0, & 1 & r4 & = & 0 - w1 + w2 \\
0, & 1 & r5 & = & 0 - w1 + w2 \\
1, & 0 & r6 & = & 0 + w1 - w2 \\
1, & 0 & r7 & = & 0 + w1 - w2 \\
0, & 1 & r8 & = & 0 - w1 + w2
\end{array}
$$

Simhash

- Hash(t1)=1010,0110
- Hash(t2)=0001,1001
- Hash(t3)=1101,0101

$$
\begin{array}{llcl}
1,0,1, & r1 & = & 0+w1-w2+w3 \\
0,0,1, & r2 & = & 0-w1-w2+w3 \\
1,0,0, & r3 & = & 0+w1-w2-w3 \\
0,1,1, & r4 & = & 0-w1+w2+w3 \\
0,1,0, & r5 & = & 0-w1+w2-w3 \\
1,0,1, & r6 & = & 0+w1-w2+w3 \\
1,0,0, & r7 & = & 0+w1-w2-w3 \\
0,1,1, & r8 & = & 0-w1+w2+w3
\end{array}
$$

网络与信息安全学院
School of Cyber Engineering

Simhash

- Hash(t1)=1010,0110
- Hash(t2)=0001,1001
- Hash(t3)=1101,0101
- ……
- Hash(t8)=1110,0110

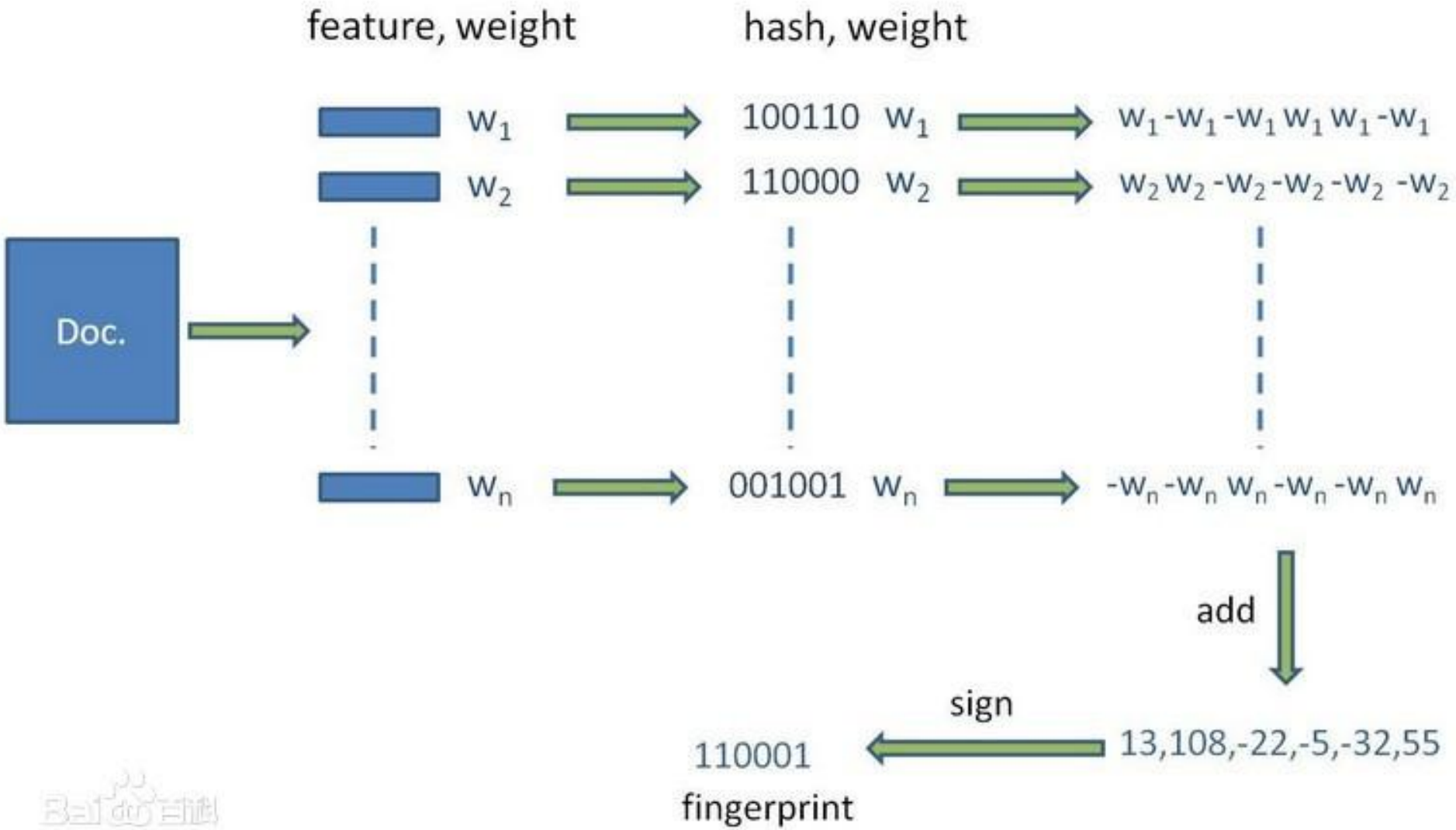
$$
\begin{aligned}
r1 &= -0.052 \\
r2 &= -1.3 \\
r3 &= 0.45 \\
r4 &= 0.32 \\
r5 &= -0.87 \\
r6 &= -1.1 \\
r7 &= -0.75 \\
r8 &= 0.65
\end{aligned}
$$

Simhash

- Hash(t1)=1010,0110, Hash(t2)=0001,1001, Hash(t3)=1101,0101, ……, Hash(t8)=1110,0110
- Step 2: 收缩。将最后的由权重值计算得到的网页总哈希值。如果某个位的值大于0，则此位置上的值设置为1，否则设置为0。
- Simhash = 0011,0001

$$
\begin{aligned}
r1 = -0.052 &\sim 0 \\
r2 = -1.3 &\sim 0 \\
r3 = 0.45 &\sim 1 \\
r4 = 0.32 &\sim 1 \\
r5 = -0.87 &\sim 0 \\
r6 = -1.1 &\sim 0 \\
r7 = -0.75 &\sim 0 \\
r8 = 0.65 &\sim 1
\end{aligned}
$$

Simhash



文本

在互联网的浪潮中，大数据技术如火如荼

1，清洗+分词

分词结果	给定权重
互联网	5
浪潮	1
大数据	4
技术	2
如火如荼	3

2，关键词hash

Hash值
10110
11000
00101
01010
11010

3，关键词向量加权建立

向量加权值
（5，-5，5，5，-5）
（1，1，-1，-1，-1）
（-4，-4，4，-4，4）
（-2，2，-2，2，-2）
（3，3，-3，3，-3）

4，合并向量累加

向量和累加
（3，-3，3，5，-7）

5，降维得指纹

10110

Simhash

- 如果2个网页相同，则相似哈希值必定相同，如果存在极个别少量的权重低的词不同，他们的相似哈希值也可能会相同。
- 两个网页的相似哈希相差越小，两个网页的相似性越高。
- Simhash对长文本500字+比较适用，短文本可能偏差较大，具体需要根据实际场景测试。使用海明距离求相似。在google的论文给出的数据中，64位的签名，在海明距离为3的情况下，可认为两篇文档是相似的或者是重复的，当然针对自己的应用可能又不同的测试取值。

向量空间距离表示

- Q：两个n维向量A(x11,x12,…x1n)与B(x21,x22,…,x2n)间的距离如何度量？
- 欧氏距离（Euclidean Distance）
  - 欧式距离（L2范数）是最易于理解的一种距离计算方法，源自欧式空间中两点间的距离

$$d_{AB} = \sqrt{\sum_{k=1}^{n}(x_{1k} - x_{2k})^2}$$

- 曼哈顿距离（Manhattan Distance）
  - 曼哈顿距离（L1范数）的计算方法从一个十字路口到另外一个十字路口的距离。

$$d_{AB} = \sum_{k=1}^{n}|x_{1k} - x_{2k}|$$

向量空间距离表示

- **夹角余弦（Cosine）**
  - 几何中的夹角余弦可用来衡量两个向量方向的差异，机器学习中借用这一概念来衡量样本向量之间的差异。

$$\cos\theta = \frac{AB}{|A||B|} = \frac{\sum_{k=1}^{n} x_{1k}x_{2k}}{\sqrt{\sum_{k=1}^{n} x_{1k}^2}\sqrt{\sum_{k=1}^{n} x_{1k}^2}}$$

- **汉明距离（Hamming Distance）**
  - 两个等长字符串s1与s2之间的汉明距离为将其中一个变为另外一个所需要的最小替换次数。信息编码

- 杰卡德距离（Jaccard Distance）
  - 用两个集合中不同元素占所有元素的比例来衡量两个集合的区分度。

$$J_\delta(A,B) = \frac{|A \cup B| - |A \cap B|}{|A \cup B|}$$

分布之间的距离度量

- Q：两个分布p和q之间的距离有多大？
- KL散度（Kullback-Leible divergence）
  - KL散度可以用于描述两个分布之间的距离，假设p(x)与q(x)是两个随机变量的分布，则它们的KL散度为

$$D(p||q) = \int_{-\infty}^{+\infty} p(x) log \frac{p(x)}{q(x)} dx$$

  - 离散情况下

$$D(p||q) = \sum_{i=1}^{n} p(x) log \frac{p(x)}{q(x)}$$

分布之间的距离度量

- KL散度（Kullback-Leible divergence）

X	1	2	3	4	5
p_X	0.1	0.2	0.3	0.3	0.1

Y	1	2	3	4	5
p_Y	0.2	0.1	0.2	0.4	0.1

$$D(P_1||P_2) = \sum_{q \in Q} P_1(q) \cdot \log_2 \frac{P_1(q)}{P_2(q)}$$

$$\begin{aligned} D_{KL}(X||Y) &= 0.1 \times \log_2 \frac{0.1}{0.2} + 0.2 \times \log_2 \frac{0.2}{0.1} + 0.3 \times \log_2 \frac{0.3}{0.2} \\ &\quad + 0.3 \times \log_2 \frac{0.3}{0.4} + 0.1 \times \log_2 \frac{0.1}{0.1} \\ &= 0.1510 \end{aligned}$$

分布之间的距离度量

- **Wasserstein距离**
  - Wasserstein距离又称为Earth-Mover距离(EM距离)，可以简单地理解为将分布p挪到分布q所需要的最短距离，计算公式如下

$$W(p,q) = \inf_{\gamma \sim \Pi(p,q)} \mathbb{E}_{(x,y)\sim\gamma}[||x-y||]$$

  - Π(p,q)表示p和q分布组合起来的分布的集合，对于每个联合分布γ，取得一个样本x, y，计算||x−y||的期望，这个期望能够取到的下界就是Wasserstein距离。
  - Wasserstein距离在CS界，用的更多的可能还是GAN（生成对抗网络）。

本章小结

- 散列法
- Bloom过滤器
- 信息指纹
- 相似性比较
- 距离度量

作业

Bloom过滤器设计

浏览器通常使用Bloom过滤器识别恶意链接，警告用户访问的网站可能是钓鱼网站。请设计一个存储钓鱼网站的Bloom过滤器，存储已知的钓鱼网站，用户可以快速查询某个网址是否是已知的钓鱼网站。

- 作业要求：

编写代码实现一个Bloom过滤器（Hash函数可以自己设计，也可以调用类似MD5、SHA1、SHA2的密码学Hash函数）；存储dataset.csv数据集中的钓鱼网站；要求查询错误概率小于0.01；撰写报告。

- 资源网址：

https://github.com/XD-ANG/Phishing_Detection_GNN

dataset.csv 数据集

网络与信息安全学院
School of Cyber Engineering