第 5 章：密码学哈希函数
Cryptographic Hash Functions

赵俊舟

西安交通大学网安学院
junzhou.zhao@xjtu.edu.cn

2025 年 12 月 20 日

目录

目录

密码学哈希函数

定义 (密码学哈希函数)

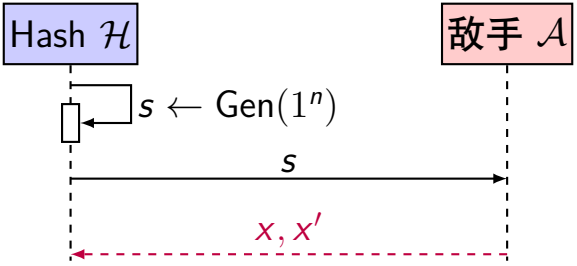
一个密码学哈希函数包含两个 PPT 算法 $\mathcal{H} = (\mathsf{Gen}, H)$，其中

- 密钥生成算法 Gen：输入安全参数 1^n，输出公开密钥 s。
- 哈希函数 H：输入密钥 s 和任意长串 $x \in \{0,1\}^*$，输出哈希值 $H_s(x) \in \{0,1\}^{\ell(n)}$，也称为消息摘要。

如果 $x \in \{0,1\}^{\ell'(n)}$ 且 $\ell'(n) > \ell(n)$，则称 $\mathcal{H}$ 为定长哈希函数。

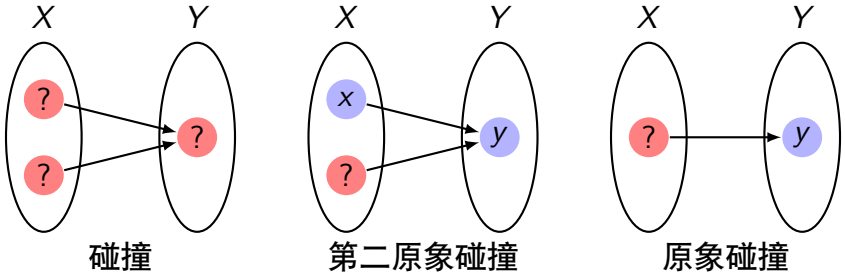
- 哈希值是消息所有比特的函数，消息任意一比特的改变都将引起哈希值的改变。
- 密码学哈希函数必然存在哈希碰撞，即对两个输入 $x \neq x'$，有 $H_s(x) = H_s(x')$。要求 PPT 敌手难以找到哈希碰撞。
- 实际使用的哈希函数不含密钥 s，可将 H_s 简写为 H。

抗碰撞哈希函数



- 当 $x \neq x'$ 且 $H_s(x) = H_s(x')$ 时，称敌手**碰撞成功**，记为
$$\text{Hash-coll}_{\mathcal{A},\mathcal{H}}(n) = 1$$
- 如果对于任何 PPT 敌手 $\mathcal{A}$，都存在可忽略函数 negl，使
$$\Pr[\text{Hash-coll}_{\mathcal{A},\mathcal{H}}(n) = 1] \leq \text{negl}(n)$$
称哈希函数 $\mathcal{H} = (\mathrm{Gen}, H)$ 具有**抗碰撞性**。

几种碰撞的区别



- **抗碰撞**（Collision Resistance）：很难找到两个输入 $x \neq x'$，使 $H(x) = H(x')$。 强
- **抗第二原象碰撞**（Second-Preimage Resistance）：给定 x，很难找到 $x' \neq x$，使 $H(x) = H(x')$。 较强
- **抗原象碰撞**（Preimage Resistance）：给定 $y = H(x)$，很难找到 x'，使 $H(x') = y$，即单向性。 弱

目录

目录

对哈希函数的原像攻击

- **原像攻击**：敌手对给定的哈希值 y，试图找到满足 $H(x) = y$ 的消息 x。
- **穷举攻击法**：敌手随机选择 x，计算其哈希值，直到碰撞出现。

原像攻击的数学描述

若一个函数可能有 N 个值，且已知一个值 $y = H(x)$，任选 k 个任意数作为输入，则 k 至少为多大才能保证至少找到一个输入值 x' 且 $H(x') = y$ 的概率大于 $1/2$？

当 $k > N/2$ 时，碰撞概率将超过 $1/2$。

对哈希函数的原像攻击

- 对于任意 x'，满足 $H(x') = y$ 的概率是 $1/N$。
- 则 k 个任意输入，至少有一个满足 $H(x') = y$ 的概率是 $1-(1-1/N)^k$，根据二项式定理

$$(1-a)^k = 1 - ka + \frac{k(k-1)}{2!}a^2 - \frac{k(k-1)(k-2)}{3!}a^3 \ldots$$

当 $a \to 0$ 时，$(1-a)^k \approx 1 - ka$。

- 所以，至少找到一个 x' 满足 $H(x') = y$ 的概率几乎等于 $1-(1-k/N) = k/N$。当 $k > N/2$ 时，这个概率将超过 $1/2$。

Robin 攻击方法

一个输出 ℓ 比特的哈希函数，敌手需要尝试 $2^{\ell-1}$ 个消息，就有可能获得超过 $1/2$ 的成功机会产生某个特定哈希码上的碰撞。

目录

对哈希函数的碰撞攻击

- 碰撞攻击：敌手试图找到两个不同消息 x 和 x'，满足 $H(x) = H(x')$，也称为生日攻击（Birthday Attack）。
- 可以证明，碰撞攻击的穷举规模要比原像攻击的穷举规模小很多。
- 生日悖论（Birthday Paradox）：
  - 23 个人中，存在两个人生日相同的概率大于 50%；
  - 50 个人中，存在两个人生日相同的概率大于 96%。

生日攻击

一个输出 ℓ 比特的哈希函数，敌手只需要尝试 $2^{\ell/2}$ 个消息，就有可能获得超过 $1/2$ 的成功机会产生哈希碰撞。

生日攻击的数学背景

生日攻击的数学描述

从编号为 1 到 N 的 N 个球中有放回的随机取出 k 个球，当 k 为多大时可以保证取出的 k 个球有重复的概率大于 $1/2$？

- 有放回随机取 $k < N$ 个球，球的编号互不相同的概率为

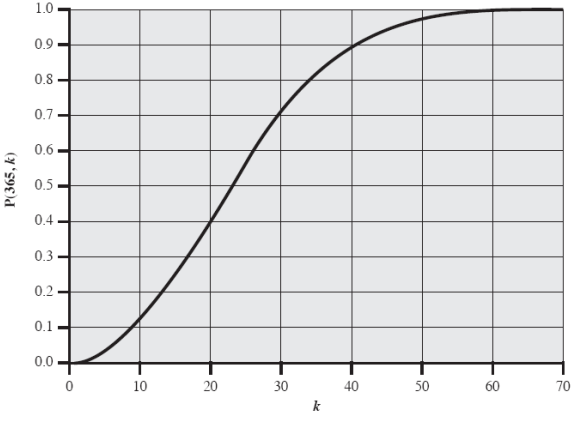
$$\frac{N(N-1)\cdots(N-k+1)}{N^k} = \prod_{i=0}^{k-1}(1-\frac{i}{N}) \le \prod_{i=0}^{k-1} e^{-i/N} = e^{-\frac{k(k-1)}{2N}}$$

  利用了 $1-x \le e^{-x}, x \in [0,1]$。

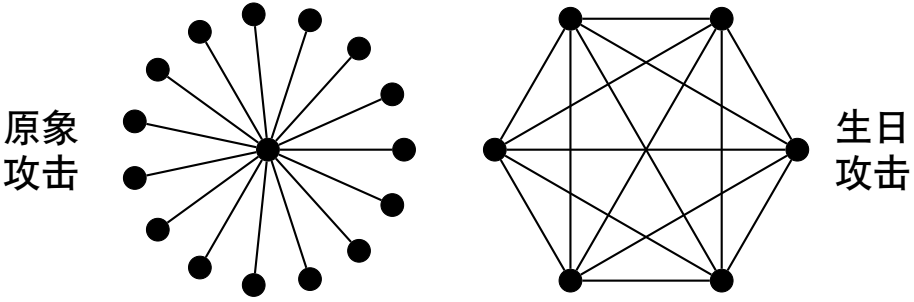
- 则至少两个球编号相同的概率为 $1 - e^{-\frac{k(k-1)}{2N}}$
- 希望上述概率不小于 0.5，得出 $k^2 > k(k-1) \ge 2N\ln 2$，所以 $k > 1.17\sqrt{N}$。

生日悖论

- 在 23 个人中，考虑某个人的特定生日，在剩下的 22 个人中找到相同生日的概率只有 $22/365$；
- 但是若只考虑同一天出生，那么 23 个人会产生 $\binom{23}{2} = 253$ 种不同的组合，所以找到同生日的概率 $253/365 > 1/2$。



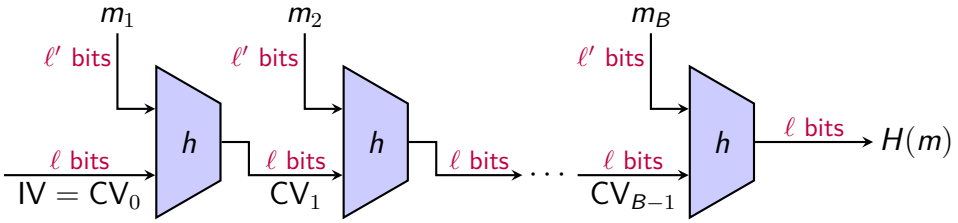
从图论角度理解生日悖论



- 一条边表示一次哈希碰撞，边数表示碰撞的机会数。
- 原象攻击中，图为星形，要获得 N 次碰撞机会，得有 N 个节点（即需要进行 N 次哈希以获得 N 次碰撞机会）。
- 生日攻击中，图为完全图，要获得 N 次碰撞机会，只需 $\sqrt{N}$ 个节点（即只需进行 $\sqrt{N}$ 次哈希就能获得 N 次碰撞机会）。

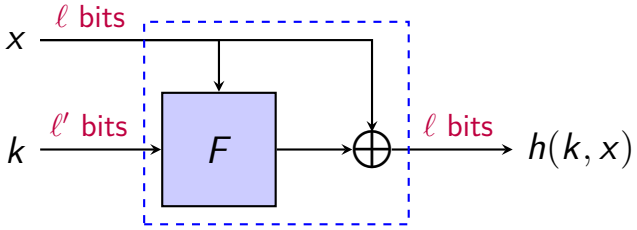
目录

Merkle-Damgård 变换：定长哈希 → 任意长哈希



- 实际中更容易实现定长哈希函数（即压缩函数），然后再通过 Merkle-Damgård 变换实现能压缩任意长消息的哈希函数。
- 将输入消息分为 B 个定长分组，分组长度 $\ell' \triangleq \ell'(n)$ 位。如果最后一个分组不足 ℓ' 位，需要对最后一个分组进行填充。
- 重复使用压缩函数 h 对前一步得到的 ℓ 位输出分组（称为链接变量 CV）和当前 ℓ' 位输入分组进行运算，得到 ℓ 位输出。
- 最后一个压缩函数的输出分组即为消息 m 的哈希值 $H(m)$。

Davies-Meyer 构造：分组密码 → 定长哈希函数



- 通过 Davies-Meyer 构造可以把一个分组密码变换为一个压缩函数，即定长哈希函数。
- F 为一个分组密码，密钥长度 ℓ' 位，分组长度 ℓ 位。
- 定义压缩函数 $h\colon \{0,1\}^{\ell+\ell'} \mapsto \{0,1\}^{\ell}$ 为

$$h(k,x) \triangleq F_k(x) \oplus x$$

安全哈希算法（Secure Hash Algorithm, SHA）

- MD5 算法于 1991 年提出，哈希值为 160 位。2004 年，王小云提出一种高效的产生 MD5 碰撞的方法。目前使用一台 PC 可以在一分钟内产生一个碰撞。
- SHA-1 是由 NIST 设计，并于 1995 年成为标准，哈希值为 160 位。2005 年，王小云提出一种攻击方法，可以用 2^{69} 次计算找到一个碰撞。2022 年 12 月，NIST 建议 2030 年底前淘汰 SHA-1。
- SHA-2 于 2001 年成为标准，包含 224、256、384、512 等版本，没有 SHA-1 的缺点。
- SHA-3 于 2015 年成为标准，包含 224、256、384、512 等版本，采用的算法与 SHA-1 和 SHA-2 差别较大。
- SHA-2 以及 SHA-3 是目前推荐使用的抗碰撞哈希函数。

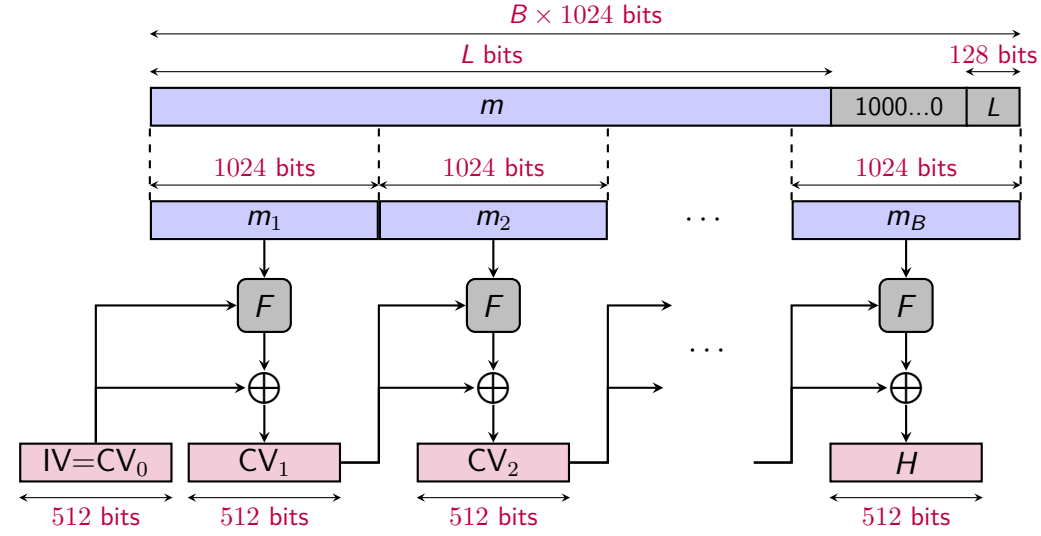
SHA-1 和 SHA-2 参数比较

Algorithm	Message Size	Block Size	Word Size	Message Digest Size
SHA-1	$< 2^{64}$	512	32	160
SHA-224	$< 2^{64}$	512	32	224
SHA-256	$< 2^{64}$	512	32	256
SHA-384	$< 2^{128}$	1024	64	384
SHA-512	$< 2^{128}$	1024	64	512
SHA-512/224	$< 2^{128}$	1024	64	224
SHA-512/256	$< 2^{128}$	1024	64	256

Note: All sizes are measured in bits.

SHA-512 总体结构

- 输入最大长度为 2^{128} 位的消息，输出 512 位的消息摘要。



SHA-512 算法步骤

1. **附加填充位**：使消息长度 $\equiv 896 \pmod{1024}$，填充位数在 1 到 1024 之间，由 1 和后续的 0 组成。
2. **附加长度**：在消息后附加 128 位的块，将其看作 128 位无符号整数，代表填充前消息的长度。
3. **初始化哈希缓冲区**：8 个 64 位寄存器（8 个字）

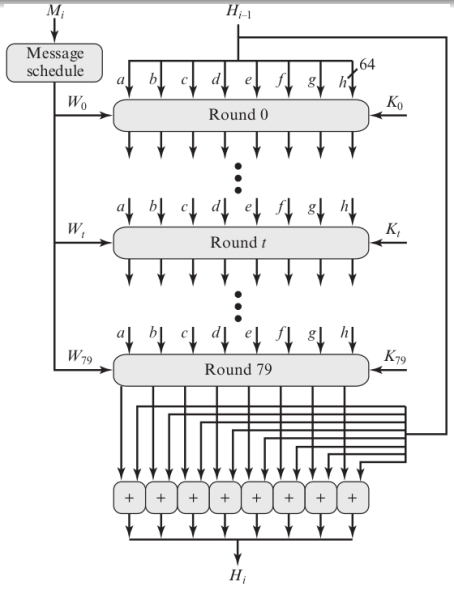
$$a = \texttt{6A09E667F3BCC908} \qquad e = \texttt{510E527FADE682D1}$$
$$b = \texttt{BB67AE8584CAA73B} \qquad f = \texttt{9B05688C2B3E6C1F}$$
$$c = \texttt{3C6EF372FE94F82B} \qquad g = \texttt{1F83D9ABFB41BD6B}$$
$$d = \texttt{A54FF53AF1D336F1} \qquad h = \texttt{5BE0CD19137E2179}$$

4. **以 1024 比特分组为单位处理消息**：核心是 80 轮运算，每一轮都把 512 位缓冲区的值 $abcdefg$ 作为输入并更新。
5. **输出**：所有 B 个 1024 比特分组都处理完后从第 B 阶段输出的是 512 比特的消息摘要。

SHA-512 的运算

$$H_0 = \mathrm{IV}$$
$$H_i = \mathrm{SUM64}(H_{i-1}, abcdefgh_i)$$
$$\mathrm{MD} = H_N$$

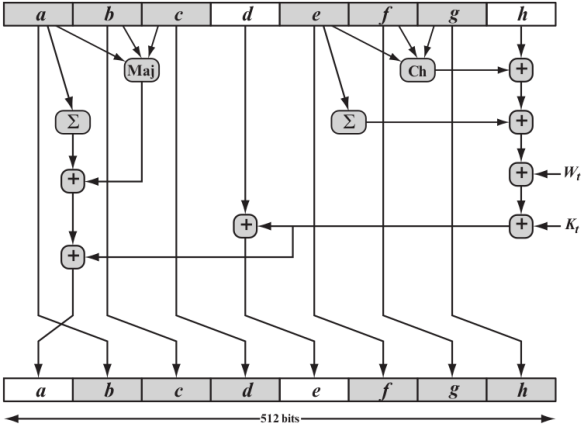
- IV：缓冲区初值。
- $abcdefgh_i$：第 i 个分组处理的输出
- B：消息中的分组数。
- SUM64：模 2^{64} 加。
- MD：最后的消息摘要。
- $K_t, t = 0, \ldots, 79$ 为轮常数，用来使每一轮的运算不同。
- $W_t, t = 0, \ldots, 79$ 由消息导出。



对单个 1024 位分组的处理

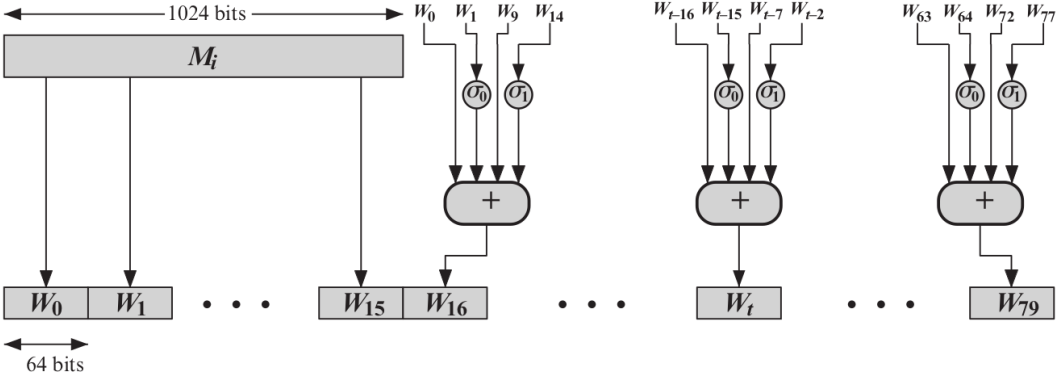
SHA-512 轮函数

- 输出 8 个字中的 6 个是简单的轮转置换（阴影部分）。
- 输出中只有 2 个字是通过替代置换产生的。
- Σ, Maj 和 Ch 为复杂二进制运算。



W_t 的导出

- 前 16 个 W_t 直接取自当前分组的 16 个字。
- 余下的 64 个 W_t 由前面的 4 个 W_t 推导得出。
- σ_0, σ_1 为复杂二进制运算。



SHA-512 的特点

- Hash 码的每一位都是全部输入位的函数。
- 基本函数 F 多次复杂重复运算使得结果充分混淆。
- 找到两个具有相同 Hash 码的消息的复杂度为 2^{256} 次操作。
- 给定 Hash 码，寻找消息的复杂度为 2^{511} 次操作。

目录

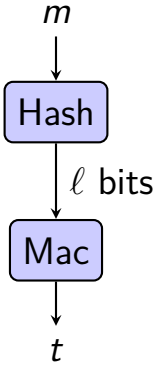
基于哈希函数的 MAC：Hash-and-MAC

- 可以将输出长度为 $\ell(n)$ 的哈希函数和消息长度为 $\ell(n)$ 的定长消息认证码结合，实现对任意长消息的认证。

设计 (Hash-and-MAC)

令 $\Pi' = (\mathsf{Mac}', \mathsf{Vrfy}')$ 为消息长度为 $\ell(n)$ 的定长消息认证码，令 $\mathcal{H} = (\mathsf{Gen}_H, H)$ 为输出长度为 $\ell(n)$ 的哈希函数。按如下方式构造消息认证码 $\Pi = (\mathsf{Gen}, \mathsf{Mac}, \mathsf{Vrfy})$：

- Gen: 输入 1^n，选择任意 $k \in \{0,1\}^n$ 为消息认证码 Π' 的密钥，计算 $s \leftarrow \mathsf{Gen}_H(1^n)$，输出密钥 (k, s)；
- Mac: 输入消息 $m \in \{0,1\}^*$，输出 $t \leftarrow \mathsf{Mac}'_k(H_s(m))$；
- Vrfy: 输入消息 $m \in \{0,1\}^*$ 和消息认证码 t，输出 $\mathsf{Vrfy}'_k(H_s(m), t)$。



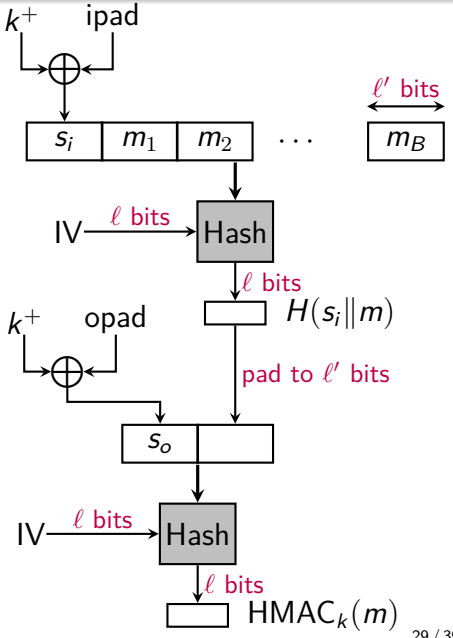
基于哈希函数的 MAC：HMAC

- Hash-and-MAC 需要使用两个密码原语：哈希函数和伪随机函数；
- 而且对输入输出长度有严格要求。

设计 (HMAC)

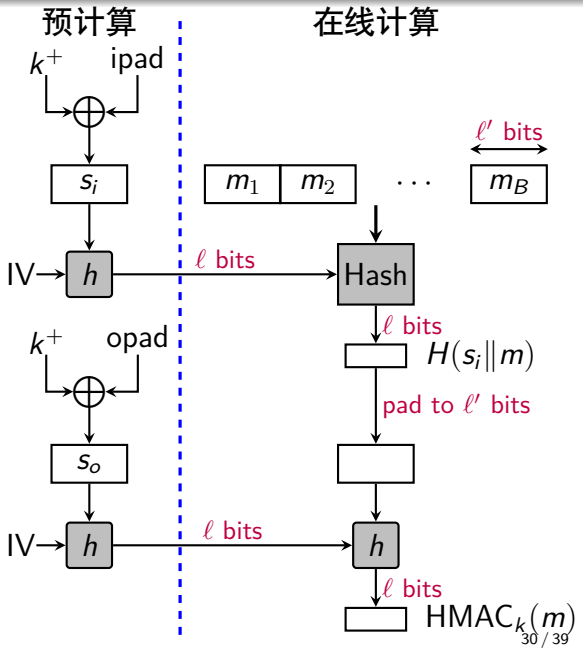
1. 在 k 左边填充 0，得到 ℓ' 位的 k^+。
2. k^+ 与 ipad 异或，产生 ℓ' 位分组 s_i。
3. 将消息 m 附于 s_i 后，进行哈希。
4. k^+ 与 opad 异或，得 ℓ' 位分组 s_o。
5. 将步骤 3 中的哈希码附于 s_o 后。
6. 哈希后得到最终消息认证码：

$$\mathrm{HMAC}_k(m) = H\big(s_o \| H(s_i \| m)\big)$$



通过预计算提高 HMAC 的计算效率

- 相比于消息哈希，HMAC 多执行了三次压缩函数 h。
- 对于长消息，HMAC 和哈希函数的执行时间大致相同。
- 对于短消息，可以通过预计算其中的两个压缩函数来提高效率。
- 这样在线计算只多执行了一次压缩函数。



目录

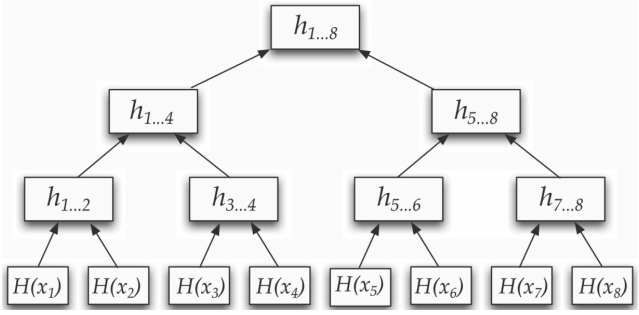
指纹和去重

- 如果 H 为抗碰撞哈希函数，则 $H(x)$ 可以作为输入 x 的唯一标识，或称为指纹。
- 病毒指纹：杀毒软件的病毒库记录了已知病毒的指纹，即对于一份病毒代码 x，其指纹为 $H(x)$。
- 去重：在云存储场景中，用户上传文件到云中。为了避免重复存储同样的文件，可以通过指纹比对确定是否已经存储过该文件。
- P2P 文件共享：P2P 网络中，peer 节点可以广播其存储的文件哈希，使其他 peer 节点知道文件的存储位置。

Merkle 树

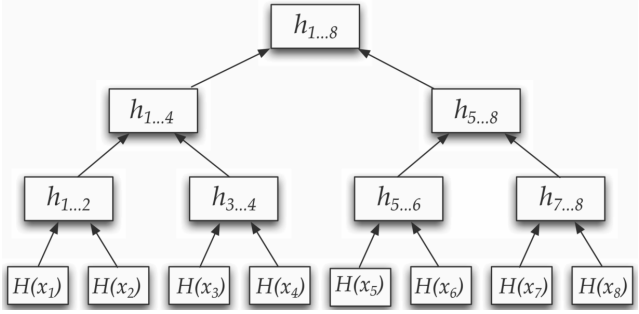
- 客户端上传一个文件 x 到服务器，之后从服务器下载相同的文件，客户端如何确保文件没有被篡改？
- 客户端上传文件 x 的同时，在本地保存文件的哈希值 $h = H(x)$。下载文件 x' 后，重新计算哈希值 $H(x')$，如果 $H(x') = h$，则认为文件未被篡改，即 $x' = x$。
- 推广到有 t 个文件 $x_1, \ldots, x_t$ 的情况。客户端上传这 t 个文件的同时，在本地保存哈希值 $h = H(x_1, \ldots, x_t)$。
- 客户端下载其中一个文件 x_i，如何保证该文件未被篡改？此时，客户端需下载其余 $t - 1$ 个文件，然后重新计算其哈希，与 h 比较。
- 有没有更好的方法，能够减少客户端需要下载的数据量？

Merkle 树



- Ralph Merkle 最早提出用 Merkle 树解决该问题。
- Merkle 树的每个叶子节点 i 存储哈希值 $h_i = H(x_i)$，其他节点存储其子节点哈希值拼接后的哈希，例如 $h_{3...4} = H(h_3, h_4)$。
- 客户端在本地保存根节点存储的哈希值，例如 $h_{1...8}$。

Merkle 树



- 当服务器返回文件 x_i 时，同时返回 Merkle 树从根节点到叶子节点 x_i 的路径上的所有节点的兄弟节点存储的哈希值。
- 例如，返回 x_3 的同时，须返回 $h_4, h_{1...2}, h_{5...8}$。
- 客户端计算 $h'_{3...4} = H(H(x_3), h_4), h'_{1...4} = H(h_{1...2}, h'_{3...4})$ 和 $h'_{1...8} = H(h'_{1...4}, h_{5...8})$，并与本地 $h_{1...8}$ 比对实现验证。

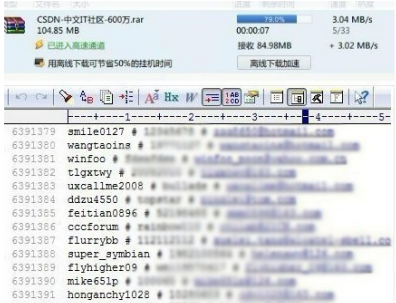
密码保护

- 考虑一个登录系统，用户输入用户名和密码，如果后台验证通过，则授权用户访问系统。
- 如果后台使用明文存储用户密码，则存在严重的安全风险。

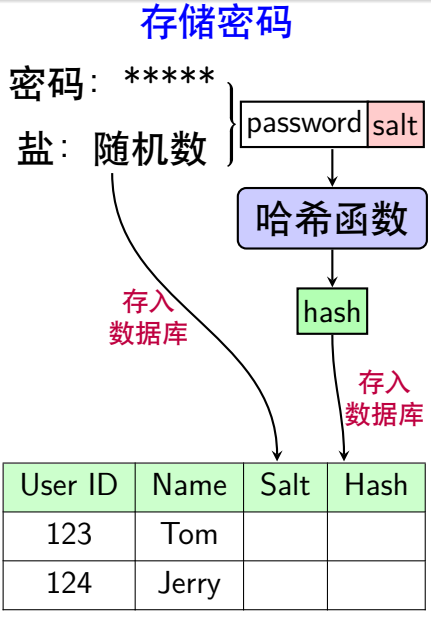
CSDN被黑：600余万个明文的注册邮箱帐号和密码被黑客公开

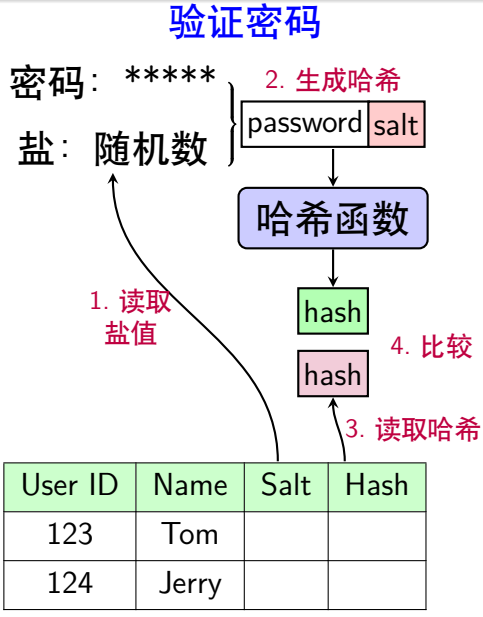
2011年12月21日 17:00

号称中国最大的开发者技术社区"中国软件开发联盟"（CSDN）数据库被黑了。360安全卫士发布消息说有600余万个明文的注册邮箱帐号和密码被黑客公开。



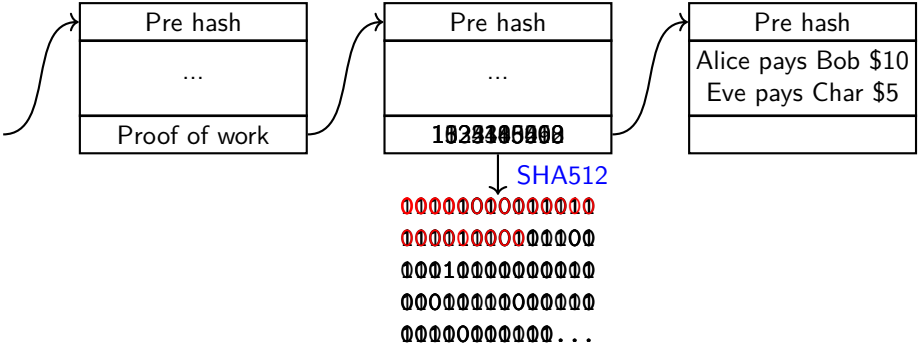
密码保护


存储密码
密码：*****
盐：随机数
password
salt
哈希函数
hash
存入
数据库
存入
数据库
User ID
Name
Salt
Hash
123
Tom
124
Jerry


验证密码
密码：*****
盐：随机数
2. 生成哈希
password
salt
哈希函数
hash
1. 读取
盐值
hash
4. 比较
3. 读取哈希
User ID
Name
Salt
Hash
123
Tom
124
Jerry

区块链中的工作量证明

- 区块链是一群分散的客户端节点，并由所有参与者组成的分布式数据库，是对所有比特币交易历史的记录。
- 基于工作量证明机制（Proof of Work）实现将新的交易记录链到区块链上，形成一个新的区块。
- PoW 的过程即为挖矿，通过哈希函数实现。



小结

1. 基本概念
2. 碰撞攻击
3. 安全哈希算法
4. 基于哈希的消息认证码
5. 哈希函数的其他应用