Hash 函数作业答案

1. 为什么存储口令 (Password) 的 Hash 值的时候往往需要进行加盐 (Salt) 处理?

   说明: 所谓 Salt 就是随机的字符串。加盐存储是指每当用户注册账户时, 服务器会随机生成一个 Salt 字符串, 然后计算口令 (Password) 和 Salt 字符串连接的 Hash 函数值, 将盐 (Salt) 和 Hash 值存储在服务器的口令表中。

   答: 身份认证是网络应用中重要的基础安全问题。有多种手段能实现网络上的身份认证, 其中口令认证是实现代价最低, 最容易部署, 应用范围最广的方法, 但也面临着口令存储的问题。

   为了保证口令安全, 一般要求人们在不同的应用中使用不同的口令, 并且定期修改口令。在设置口令时, 一般建议融合字母大小写、数字、非字母字符等不同类型的字符设计口令。同时, 口令越长越安全。这样做的目的是为了扩大口令空间, 提高敌手暴力破解的代价。但现实应用中人们往往难以记住多种不同的口令, 一般人们常用的口令个数都比较少, 而且会跟个人或家人的一些特征相关。所以在理想与现实之间存在一定的鸿沟, 口令的存储变得至关重要。

   将口令明文进行存储是最不安全的方式, 在这种情况下如果存储在服务器的口令表泄露, 则敌手可以完全获得整个系统的身份标识和明文口令对。敌手的成本较低。

   常用的方式是服务器存储 Hash 过后的口令表, 用户登录时输入口令, 服务器对用户输入的口令做 Hash 运算, 将运算值和数据库中的 Hash 值进行比较, 如果相同, 则允许登录。存储 Hash 的口令表要比存储明文口令表安全, 但仅仅存储 Hash 的口令表, 也是很容易被破解。假设服务器存储的 Hash 口令表泄露了, 敌手可以使用字典攻击和暴力攻击找到口令明文。字典攻击将常用的字符 (暴力攻击是将各种可能的字符) 组合起来进行 Hash 运算, 然后与泄露的 Hash 口令表进行比较, 如果相同, 表示用户的口令被破解。由于 Hash 函数运行速度非常快, 而人们常用的口令空间较小, 口令长度较短, 所以字典攻击和暴力攻击对破解 Hash 的口令表是可行的。

   更进一步, 如果在做 Hash 之前对口令进行加盐的操作, 则能够获得更高的安全性。在加盐的情况下, 服务器在存储口令时, 首先为每个用户的口令生成一个盐值(随机字符串), 然后计算口令与盐值连接的 Hash 值。在这种情况下, 即使相同的口令也因为盐值不同而产生不同的 Hash 值。同时, 加盐操作相当于增长了用户的口令。服务器对 Hash 口令表和盐值表分别存储, 即使是 Hash 的口令表泄露, 由于不知道盐值, 字典攻击和暴力破解的难度也大大提升。从而, 加盐操作能大幅提高存储口令的安全性。

2. 考虑用 RSA 加密算法构造哈希函数, 将消息分组后用 RSA 公钥加密第一个分组, 加密结果与第二个分组异或后再对其进行加密, 一直进行下去直到最后一个分组。设一个消息被分成两个分组 M_1 和 M_2, 其哈希值为 $H(M_1, M_2) = \mathrm{RSA}(\mathrm{RSA}(M_1) \oplus M_2)$。对于

该哈希函数, 给定一个分组 C_1, 请给出另外一个分组 C_2 使得 $H(C_1, C_2) = H(M_1, M_2)$。(即对该 Hash 函数, 很容易找到碰撞。)

解: 假设找到 M_1, M_2 的碰撞 C_1, C_2 使得 $H(C_1, C_2) = H(M_1, M_2)$, 则有 $\mathrm{RSA}(\mathrm{RSA}(C_1) \oplus C_2) = \mathrm{RSA}(\mathrm{RSA}(M_1) \oplus M_2)$, 由于 RSA 是确定性加密, 从而有 $\mathrm{RSA}(C_1) \oplus C_2 = \mathrm{RSA}(M_1) \oplus M_2$, 可得 $C_2 = \mathrm{RSA}(C_1) \oplus \mathrm{RSA}(M_1) \oplus M_2$。

下面验证 C_1, C_2 是否是 M_1, M_2 的碰撞。

根据该 Hash 函数的定义有 $H(C_1, C_2) = \mathrm{RSA}(\mathrm{RSA}(C_1) \oplus C_2) = \mathrm{RSA}(\mathrm{RSA}(\mathrm{C}_1) \oplus \mathrm{RSA}(C_1) \oplus \mathrm{RSA}(M_1) \oplus M_2) = \mathrm{RSA}(\mathrm{RSA}(M_1) \oplus M_2) = H(M_1, M_2)$。

所以分组$(C_1, C_2) = (C_1, \mathrm{RSA}(C_1) \oplus \mathrm{RSA}(M_1) \oplus M_2)$是$(M_1, M_2)$的一个 Hash 碰撞。

3. 画出下面由分组密码 $e(\cdot)$ 构造的 Hash 函数的块图, 比如图(a)为 $e(H_{i\text{-}1}, x_i) \oplus x_i$ 的块图。

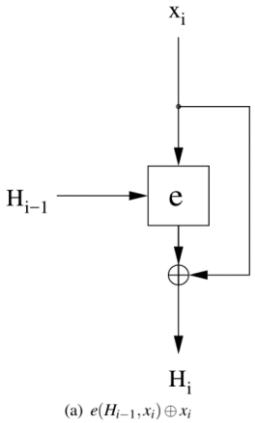
(b) $e(H_{i\text{-}1}, x_i \oplus H_{i\text{-}1}) \oplus x_i \oplus H_{i\text{-}1}$

(c) $e(H_{i\text{-}1}, x_i) \oplus x_i \oplus H_{i\text{-}1}$

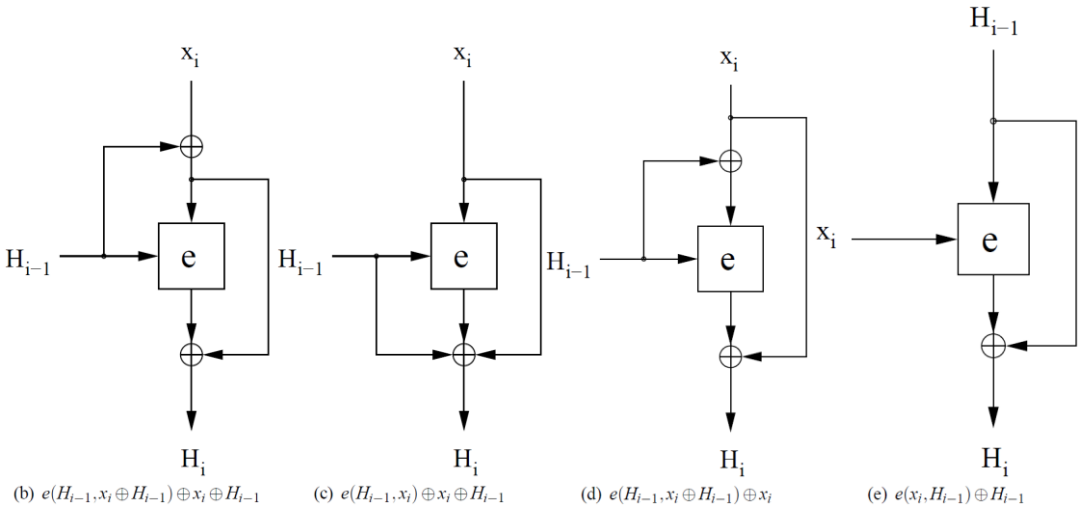
(d) $e(H_{i\text{-}1}, x_i \oplus H_{i\text{-}1}) \oplus x_i$

(e) $e(x_i \oplus H_{i\text{-}1}) \oplus H_{i\text{-}1}$

解: 这些分组密码构造的 Hash 函数的块图分别如下:



(a) $e(H_{i-1}, x_i) \oplus x_i$



(b) $e(H_{i-1}, x_i \oplus H_{i-1}) \oplus x_i \oplus H_{i-1}$ (c) $e(H_{i-1}, x_i) \oplus x_i \oplus H_{i-1}$ (d) $e(H_{i-1}, x_i \oplus H_{i-1}) \oplus x_i$ (e) $e(x_i, H_{i-1}) \oplus H_{i-1}$

4. 假设一种 Hash 函数的计算公式如下

$$C_i = b_{i1} \oplus b_{i2} \oplus b_{i3} \oplus b_{i4} \oplus b_{i5} \oplus b_{i6} \oplus b_{i7} \oplus b_{i8}$$

比如, 对于二进制编码 $(0000\ 0001)_2$, 其 Hash 值可用以上公式计算为 $C = 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 0 \oplus 1 = 1$。

每个 8 位的块构成一个 ASCII 编码的字符。

(a) 将字符串 CRYPTO 编码为二进制。

(b) 根据以上公式计算 CRYPTO 的 6 bit 的 Hash 值。

(c) 如何找到此 Hash 函数的碰撞, 试举出有实际意义的字符串说明。

解: (a) CRYPTO 的二进制编码为 01000011 01010010 01011001 01010000 01010100 01001111。

(b) 根据上述 Hash 函数计算公式 CRYPTO 对应的 Hash 值为 110010。

(c) 从 Hash 函数的计算公式可知, 该 Hash 函数将每 8bit 的二进制串压缩为 1 位的二进制串, 即对 26 个大写字母作为输入, 只有两种不同的输出 0 和 1, 所以该 Hash 函数存在大量的碰撞。

实际上字母集合 1 = {A, B, D, G, H, K, M, N, P, S, U, V, Y, Z}中每个字母的 Hash 输出都为 0, 而字母集合 2 = {C, E, F, I, J, L, O, Q, R, T, W, X}中每个字母的输出都为 1。

所以, 当分别从字母集合 1, 字母集合 1, 字母集合 2, 字母集合 2, 字母集合 1, 字母集合 2 任选一个字母组成的六个字母, 这六个字母组成的字符串一定是 CRYPTO 的 Hash 碰撞。比如, COMMON 的 Hash 值为 110010, 所以 COMMON 是 CRYPTO 的一个有实际意义的碰撞。