第 5 章：消息认证和安全哈希函数

赵俊舟

junzhou.zhao@xjtu.edu.cn

2025 年 4 月 1 日

目录

目录

网络通信环境中的攻击

- **泄密**：将消息透露给没有密钥的第三方
- **传输分析**：分析双方通信模式
- **消息伪装**：欺诈源向网络中插入一条消息
- **身份伪装**：发送方伪装为其他身份
- **内容篡改**：对消息内容的修改
- **顺序篡改**：对消息顺序的修改
- **计时篡改**：对消息的延时和重放
- **信源抵赖**：发送方否认发送过某消息
- **信宿抵赖**：接收方否认接收过某消息

网络通信环境中的攻击

- **泄密**：将消息透露给没有密钥的第三方
- **传输分析**：分析双方通信模式

消息保密

- **消息伪装**：欺诈源向网络中插入一条消息
- **身份伪装**：发送方伪装为其他身份
- **内容篡改**：对消息内容的修改
- **顺序篡改**：对消息顺序的修改
- **计时篡改**：对消息的延时和重放
- **信源抵赖**：发送方否认发送过某消息
- **信宿抵赖**：接收方否认接收过某消息

网络通信环境中的攻击

- **泄密**：将消息透露给没有密钥的第三方
- **传输分析**：分析双方通信模式

消息保密

- **消息伪装**：欺诈源向网络中插入一条消息
- **身份伪装**：发送方伪装为其他身份
- **内容篡改**：对消息内容的修改
- **顺序篡改**：对消息顺序的修改
- **计时篡改**：对消息的延时和重放

消息认证

- **信源抵赖**：发送方否认发送过某消息
- **信宿抵赖**：接收方否认接收过某消息

网络通信环境中的攻击

- **泄密**：将消息透露给没有密钥的第三方
- **传输分析**：分析双方通信模式

消息保密

- **消息伪装**：欺诈源向网络中插入一条消息
- **身份伪装**：发送方伪装为其他身份
- **内容篡改**：对消息内容的修改
- **顺序篡改**：对消息顺序的修改
- **计时篡改**：对消息的延时和重放

消息认证

- **信源抵赖**：发送方否认发送过某消息
- **信宿抵赖**：接收方否认接收过某消息

数字签名 + 其他手段

消息认证

消息认证（Message Authentication）

验证所收到的消息确实来自真正的发送方且未被修改，也可以验证消息的顺序和时效。

消息认证关心的问题：

- 验证消息来源的真实性
- 验证消息的完整性

消息认证的方法：

- 消息认证码
- 安全哈希函数

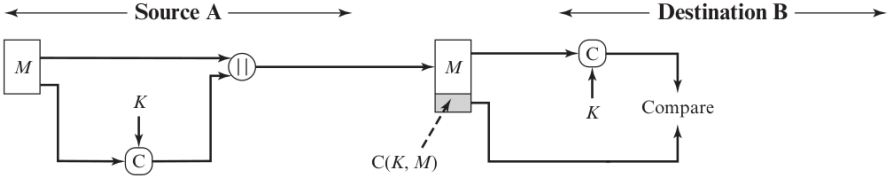
消息认证码

消息认证码（Message Authentication Code, MAC）

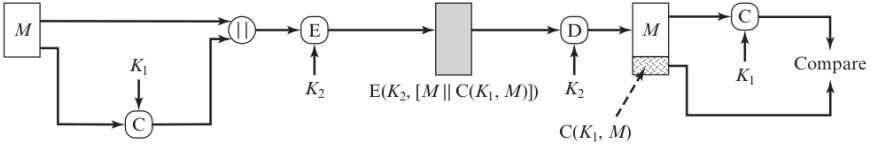
又称消息认证函数或密码校验和，利用密钥 k 生成消息 m 的一个定长短数据块 $\mathrm{MAC} = C(k, m)$，并将 MAC 附加在消息后。

- 接收方对收到的消息重新计算 MAC，并与接收到的 MAC 比较。因为只有双方知道密钥，如果两个 MAC 匹配，则：
  - 接收方可以确信报文未被更改
  - 接收方可以确信报文来自声称的发送者
- MAC 函数类似于加密，但非数字签名，也无需可逆。
- 将 MAC 直接与明文并置，然后加密传输比较常用。

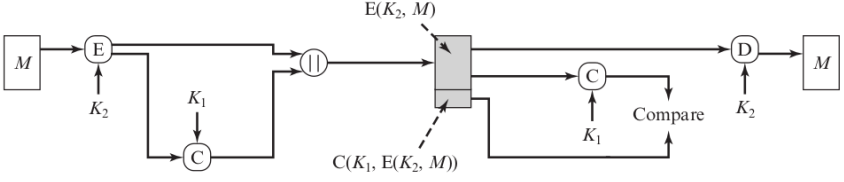
消息认证码的几种工作方式



(a) Message authentication



(b) Message authentication and confidentiality; authentication tied to plaintext



(c) Message authentication and confidentiality; authentication tied to ciphertext

消息认证码的特点

- MAC 是一种多对一函数，类似于加密函数，但不要求可逆。
- 定义域由任意长的消息组成，值域由所有可能的 MAC 组成。
- 若使用 n 位长的 MAC，则有 2^n 个可能的 MAC。有 N 条可能的消息，$N \gg 2^n$。
- 若密钥长度为 λ，则有 2^λ 种可能的密钥，即 2^λ 种可能的映射关系。
- 例如，$N = 2^{100}$，$n = 10$，共有 2^{10} 种不同的 MAC，平均每一个 MAC 可由 $2^{100}/2^{10} = 2^{90}$ 条不同的消息产生。若 $\lambda = 5$，则从消息集合到 MAC 值的集合有 $2^5 = 32$ 种不同映射。

对消息认证码的要求

- 若攻击者已知 m 和 $C(k, m)$，则构造满足 $C(k, m') = C(k, m)$ 的消息 m' 在计算上是不可行的。
- $C(k, m)$ 是均匀分布的，即对任何随机选择的消息 m 和 m'，$C(k, m') = C(k, m)$ 的概率是 2^{-n}，其中 n 是 MAC 的位数。
- 对于 $m' \neq m$，都有

$$Pr[C(k, m) = C(k, m')] = 2^{-n}$$

即只要两消息不相同，其 MAC 值相同的概率都只是 2^{-n}。

对消息认证码的要求

- 若攻击者已知 m 和 $C(k, m)$，则构造满足 $C(k, m') = C(k, m)$ 的消息 m' 在计算上是不可行的。
- $C(k, m)$ 是均匀分布的，即对任何随机选择的消息 m 和 m'，$C(k, m') = C(k, m)$ 的概率是 2^{-n}，其中 n 是 MAC 的位数。
- 对于 $m' \neq m$，都有

$$Pr[C(k, m) = C(k, m')] = 2^{-n}$$

即只要两消息不相同，其 MAC 值相同的概率都只是 2^{-n}。

问题：攻击者如何用穷举法找到消息认证函数的密钥？

消息认证码的安全性

- 假设攻击者可以访问明文及其消息认证码。
- λ 为密钥长度，n 为消息认证码长度，$\lambda > n$。
- 对于 $T_1 = C(k, m_1)$，攻击者需要穷举所有可能的密钥 k_i，计算 $T_i = C(k_i, m_1)$，那么至少会找到一个密钥使得 $T_i = T_1$。
- 一共产生了 2^λ 个消息认证码，但只有 2^n 个不同消息认证码。
- 许多密钥会产生相同的消息认证码，而攻击者不知道哪个密钥正确。
- 平均来说，有 $2^\lambda / 2^n = 2^{\lambda - n}$ 个密钥会产生正确的消息认证码，因此攻击者对这些密钥必须做重复攻击。
- 攻击者需要用另一对消息-消息认证码：$T_2 = C(k, m_2)$，对找到的 $2^{\lambda-n}$ 个密钥计算 $T_i = C(k_i, m_2)$，得到 $2^{\lambda-2n}$ 个密钥。

消息认证码的安全性

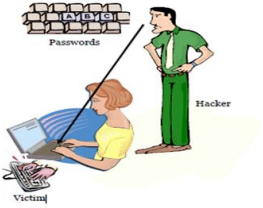
- 若 $\lambda = \alpha n$，则攻击者需要进行 α 次验证，才能找到正确密钥。
- 所以攻击者的搜索空间大小为：

$$2^{\lambda} + 2^{\lambda - n} + 2^{\lambda - 2n} + \ldots + 1 = 2^{\lambda} \frac{1 - 2^{-(\alpha+1)n}}{1 - 2^{-n}} > 2^{\lambda}$$

- 所以说，与消息加密相比，消息认证码更不容易被攻破。

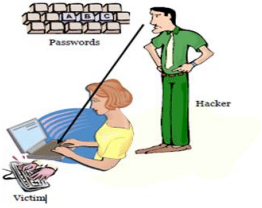
实际应用：肩窥攻击与双因素认证

- 肩窥攻击（Shoulder Surfing Attack）



实际应用：肩窥攻击与双因素认证

- 肩窥攻击（Shoulder Surfing Attack）



- 双因素认证（Two-Factor Authentication）

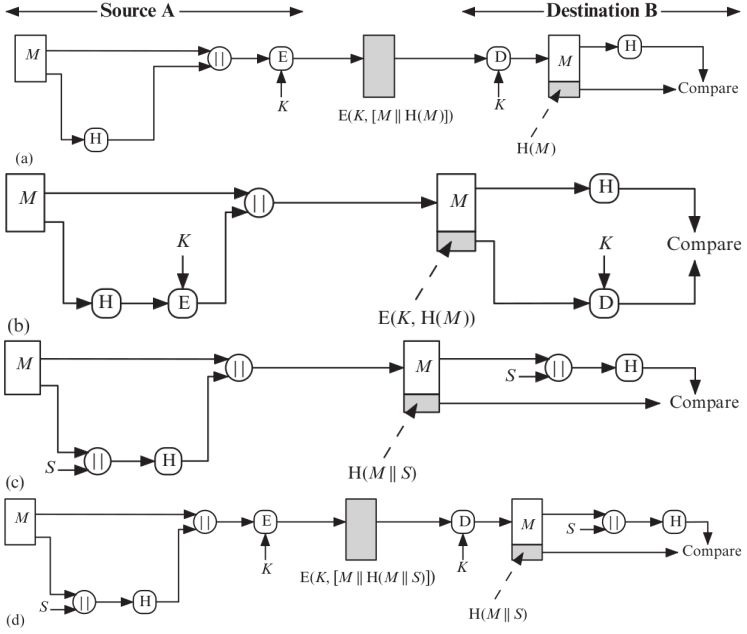
- A user logs into a service using something they know (e.g., a password) and something they have (e.g., a phone).
- Timed One-Time Password (TOTP): $T = C(k, t)$ where t is current time and rounded into 30 seconds.

目录

密码学哈希函数

- 一个哈希函数以变长的消息 m 作为输入，产生定长的哈希码 $H(m)$ 作为输出，哈希码 $H(m)$ 亦称作报文摘要。
- 哈希码是消息所有比特的函数值，具有差错检测能力，消息任意一比特的改变都将引起哈希码的改变。
- 哈希码使用方式：
  - 对附加了哈希码的消息进行加密
  - 使用常规加密方法仅对哈希码加密
  - 使用公开密钥方法仅对哈希码加密，提供数字签名
  - 同时提供保密和签名，可以分别使用常规方法加密消息及使用公开密钥方法加密哈希码

哈希函数用于消息认证


Source A
Destination B
M
H
E
K
D
Compare
E(K, [M || H(M)])
H(M)
(a)
E(K, H(M))
(b)
S
H(M || S)
(c)
E(K, [M || H(M || S)])
(d)

密码学哈希函数的安全性要求

1. H 可以应用于任意大小的数据块；
2. H 产生固定长度的输出；
3. **单向性**：对任意给定的明文 x，计算 $H(x)$ 容易；对任意给定的哈希码 h，找到满足 $H(x) = h$ 的 x，在计算上不可行；
4. **抗弱碰撞性**：对任何给定的消息 x，找到满足 $y \neq x$ 且 $H(x) = H(y)$ 的消息 y，在计算上不可行；
5. **抗强碰撞性**：找到任何满足 $H(x) = H(y)$ 的消息对 (x, y)，在计算上不可行。

弱哈希函数 vs. 强哈希函数

- 条件 1, 2, 3 是所谓单向性问题；
- 条件 4、5 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 5 主要用于防范所谓的生日攻击法。
- 能同时满足条件 1-4 的，称为**弱哈希函数**；
- 能同时满足条件 1-5 的，称为**强哈希函数**。
- 应用于密码学上的哈希函数必须是强哈希函数。

对哈希函数的原像攻击

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

对哈希函数的原像攻击

- **原像攻击**：攻击者对给定的哈希值 h，试图找到满足 $H(x) = h$ 的消息 x。
- **穷举攻击法**：攻击者随机选择 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 $H(x)$，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 y 且 $H(y) = H(x)$ 的概率大于 $1/2$？

对哈希函数的原像攻击

- **原像攻击**：攻击者对给定的哈希值 h，试图找到满足 $H(x) = h$ 的消息 x。
- **穷举攻击法**：攻击者随机选择 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 $H(x)$，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 y 且 $H(y) = H(x)$ 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 y，满足 $H(y) = H(x)$ 的概率是 $1/n$。

对哈希函数的原像攻击

- 对于任意 y，满足 $H(y) = H(x)$ 的概率是 $1/n$。
- 则 k 个任意输入，至少有一个满足 $H(y) = H(x)$ 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，$(1-a)^k \approx 1 - ka$。

对哈希函数的原像攻击

- 对于任意 y，满足 $H(y) = H(x)$ 的概率是 $1/n$。
- 则 k 个任意输入，至少有一个满足 $H(y) = H(x)$ 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 y 满足 $H(y) = H(x)$ 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 $1/2$。

对哈希函数的原像攻击

- 对于任意 y，满足 $H(y) = H(x)$ 的概率是 $1/n$。
- 则 k 个任意输入，至少有一个满足 $H(y) = H(x)$ 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 y 满足 $H(y) = H(x)$ 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

64 位的哈希函数，有 2^{64} 种组合，攻击者需要尝试 2^{63} 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。

对哈希函数的碰撞攻击

- **碰撞攻击**：攻击者试图找到两个消息 x 和 y，满足 $H(x) = H(y)$，也称为**生日攻击**（Birthday Attack）。

对哈希函数的碰撞攻击

- 碰撞攻击：攻击者试图找到两个消息 x 和 y，满足 $H(x) = H(y)$，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。

对哈希函数的碰撞攻击

- 碰撞攻击：攻击者试图找到两个消息 x 和 y，满足 $H(x) = H(y)$，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）：

对哈希函数的碰撞攻击

- 碰撞攻击：攻击者试图找到两个消息 x 和 y，满足 $H(x) = H(y)$，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%；

对哈希函数的碰撞攻击

- 碰撞攻击：攻击者试图找到两个消息 x 和 y，满足 $H(x) = H(y)$，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%。

对哈希函数的碰撞攻击

- 碰撞攻击：攻击者试图找到两个消息 x 和 y，满足 $H(x) = H(y)$，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%。
- Yuval 提出用生日悖论对哈希函数进行碰撞攻击，证明 64 位哈希函数只需要 2^{32} 次运算就有可能获得超过 $1/2$ 的成功机会。

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 $1/2$？

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

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

利用了 $1-x \leq e^{-x}, x \in [0,1]$。

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

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

  利用了 $1-x \leq e^{-x}, x \in [0,1]$。

- 则至少两个球编号相同的概率为 $1 - e^{-\frac{k(k-1)}{2n}}$

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

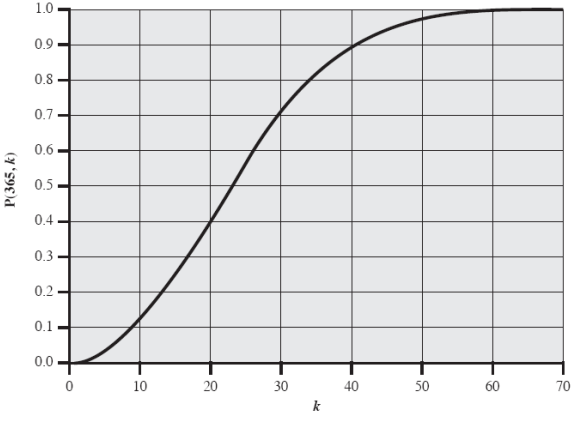
$$\frac{n(n-1)\cdots(n-k+1)}{n^k} = \prod_{i=0}^{k-1}(1-\frac{i}{n}) \leq \prod_{i=0}^{k-1} e^{-i/n} = e^{-\frac{k(k-1)}{2n}}$$

  利用了 $1-x \leq e^{-x}, x \in [0,1]$。

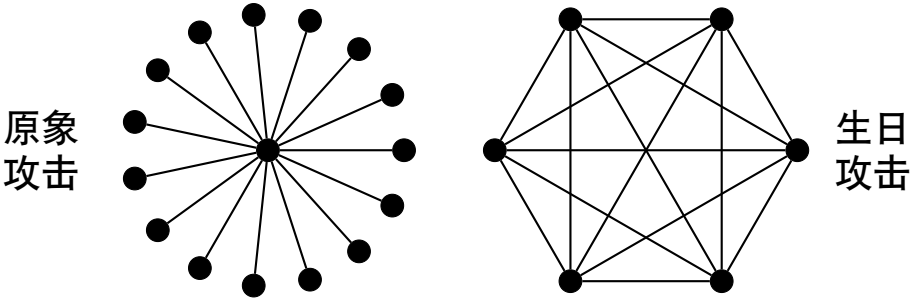
- 则至少两个球编号相同的概率为 $1 - e^{-\frac{k(k-1)}{2n}}$
- 希望上述概率不小于 0.5，得出 $k^2 > k(k-1) \geq 2n\ln 2$，所以 $k > 1.17\sqrt{n}$。

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

安全哈希算法（SHA）

- SHA 是由美国标准与技术协会 NIST 设计，并于 1993 作为联邦标准 FIPS 180 发布，1995 年修订为 FIPS 180-1，即 SHA-1。
- SHA 算法建立在 MD4 算法之上，基本框架与 MD4 类似。
- SHA-1 产生 160 位哈希值，以后的修订版分别为 SHA-256，SHA-384，SHA-512，与 SHA-1 有相同的基础结构。
- 2005 年 NIST 宣布了逐步废弃 SHA-1 的意图。一周后，王小云在 Crypto 2005 会议上发表 Finding Collisions in the Full SHA-1，宣布了一种攻击方法，用 2^{69} 次操作找到两个独立的消息具有相同哈希值（此前认为需要 2^{80} 次操作）。这一结果加速了向 SHA 其他版本的过渡。
- 2022 年 12 月，NIST 建议 2030 年底前淘汰 SHA-1。

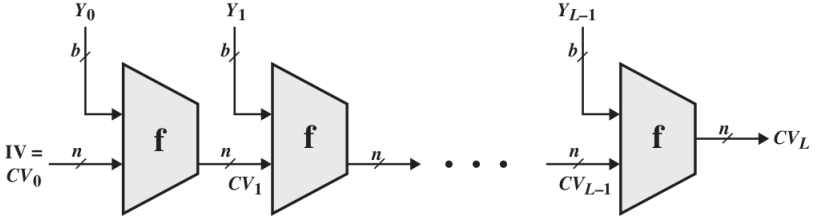
SHA 参数比较

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.

密码学安全哈希函数的结构

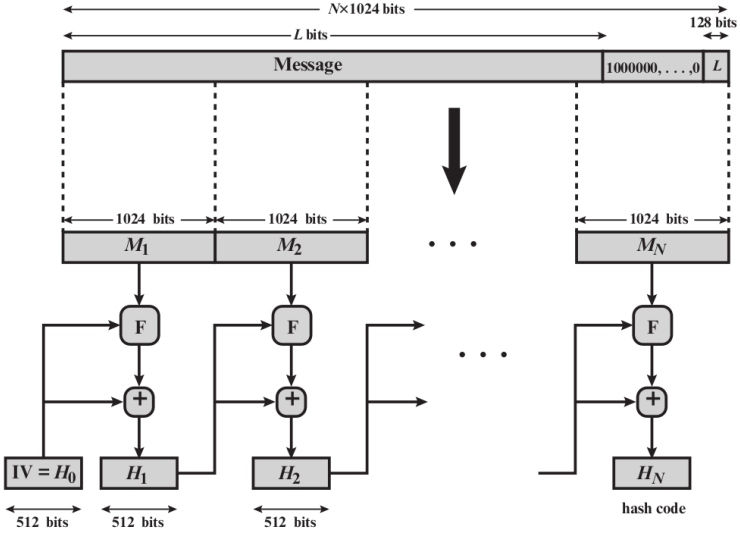
- 包括 SHA 在内的绝大多数密码学安全哈希函数都是基于迭代哈希函数结构



- 将输入消息分为 L 个固定长度的分组，每组 b 位。
- 最后一个分组不足 b 位时要填充，并且最后一个分组包含消息总长度。
- 重复使用压缩函数 f 对前一步得到的 n 位分组（称为链接变量 CV）和当前 b 位输入分组进行运算，得到一个 n 位分组。

SHA-512 总体结构

算法的输入是最大长度小于 2^{128} 位的消息，输出是 512 位的消息摘要，输入消息以 1024 位的分组为单位处理。



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）

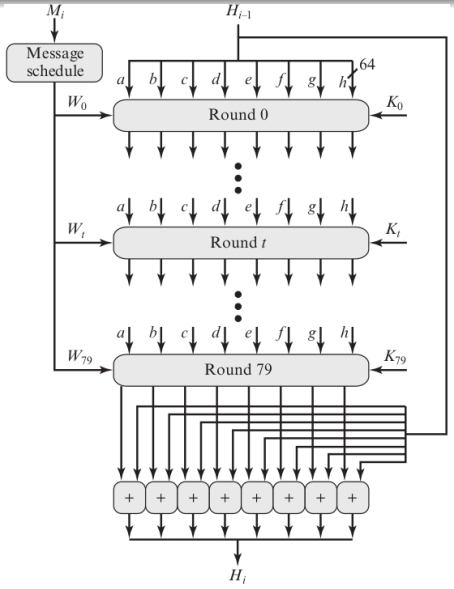
$$a = \texttt{6A09E667F3BCC908} \qquad e = \texttt{510E527FADE682D1}$$
$$b = \texttt{BB67AE8584CAA73B} \qquad f = \texttt{9B05688C2B3E6C1F}$$
$$c = \texttt{3C6EF372FE94F82B} \qquad g = \texttt{1F83D9ABFB41BD6B}$$
$$d = \texttt{A54FF53AF1D336F1} \qquad h = \texttt{5BE0CD19137E2179}$$

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 N 个 1024 比特分组都处理完后从第 N 阶段输出的是 512 比特的消息摘要。

SHA-512 的运算

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

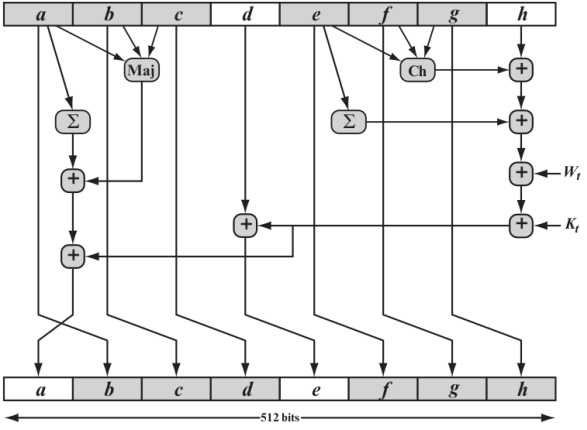
- IV：缓冲区初值。
- $abcdefgh_i$：第 i 个分组处理的输出
- N：消息中的分组数。
- SUM64：模 2^{64} 加。
- MD：最后的消息摘要。
- $K_t, t = 0, \ldots, 79$ 为轮常数，用来使每一轮的运算不同。
- $W_t, t = 0, \ldots, 79$ 由消息导出。



对单个 1024 位分组的处理

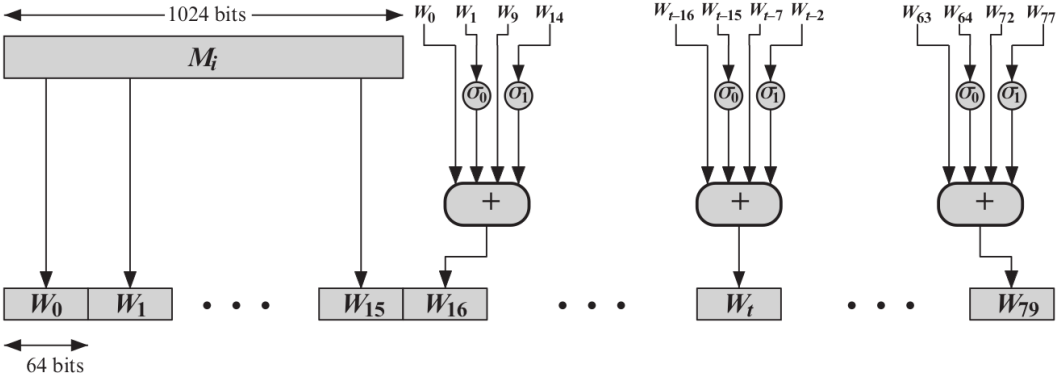
SHA-512 轮函数

- 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。

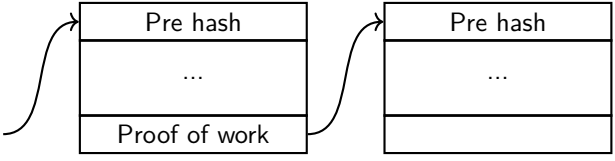


SHA-512 的特点

- 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^{512} 次操作

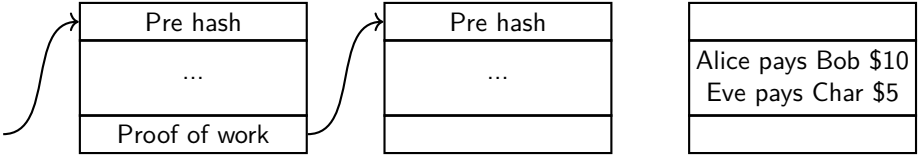
应用：区块链中的工作量证明机制

- 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，通过安全 hash 函数实现。



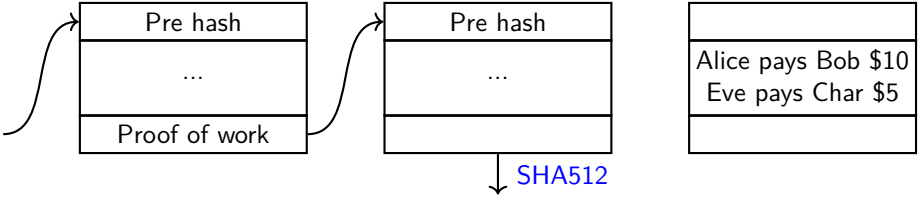
应用：区块链中的工作量证明机制

- 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，通过安全 hash 函数实现。



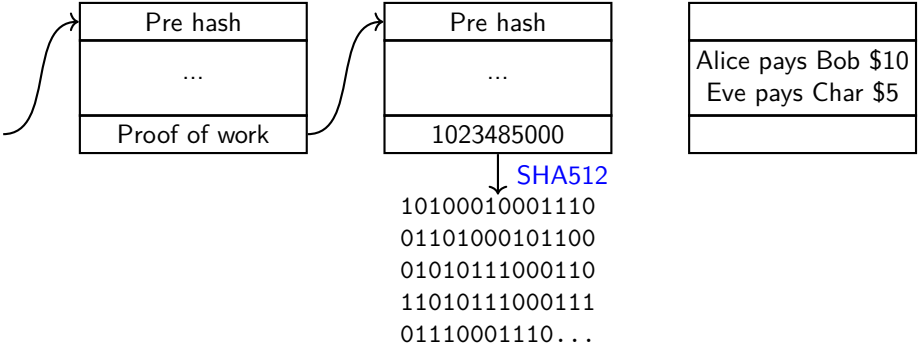
应用：区块链中的工作量证明机制

- 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，通过安全 hash 函数实现。



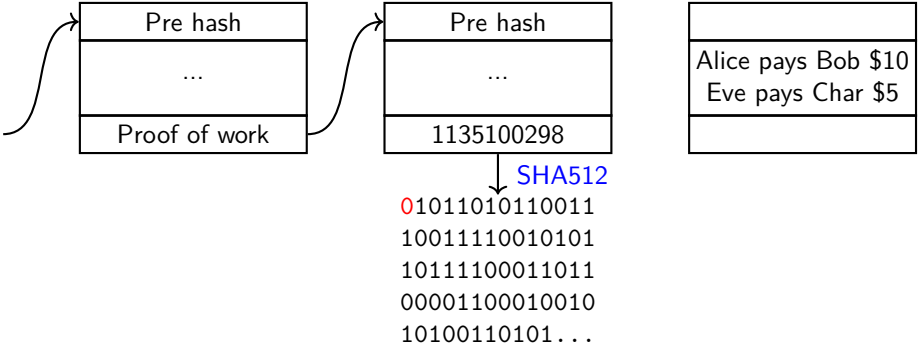
应用：区块链中的工作量证明机制

- 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，通过安全 hash 函数实现。



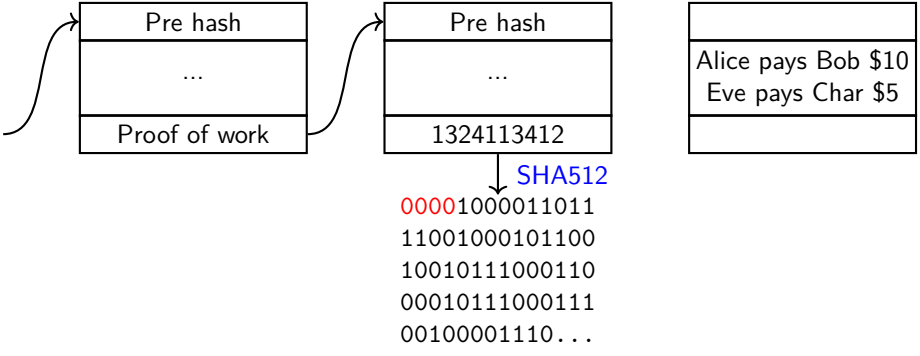
应用：区块链中的工作量证明机制

- 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，通过安全 hash 函数实现。



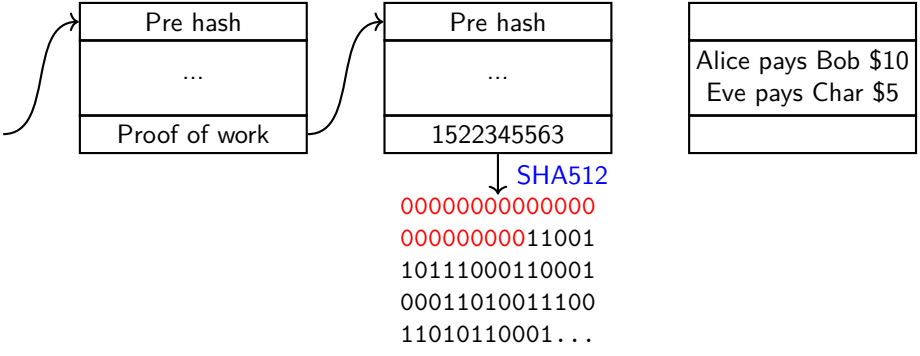
应用：区块链中的工作量证明机制

- 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，通过安全 hash 函数实现。



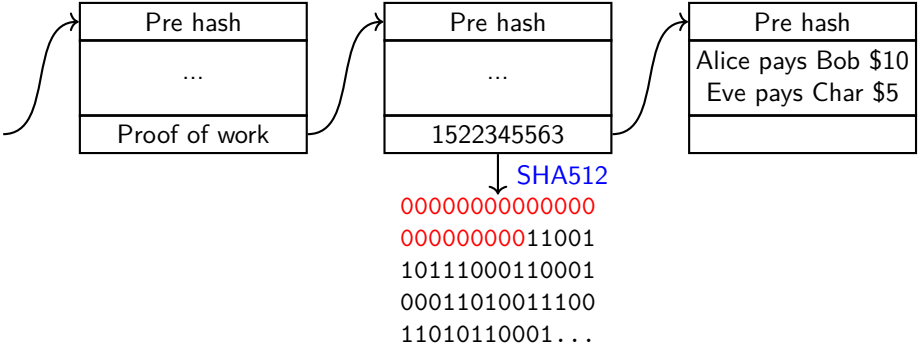
应用：区块链中的工作量证明机制

- 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，通过安全 hash 函数实现。



应用：区块链中的工作量证明机制

- 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，通过安全 hash 函数实现。



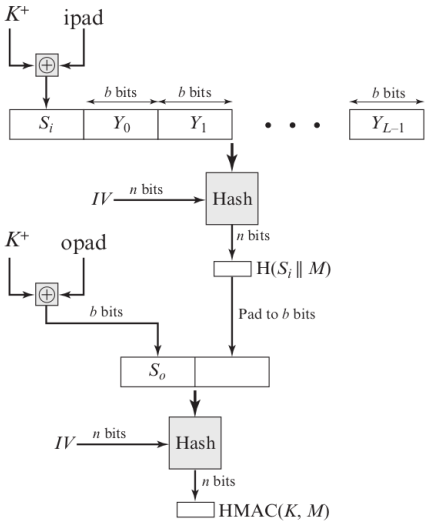
目录

基于哈希函数的 MAC：HMAC

- 人们越来越感兴趣利用密码学哈希函数设计 MAC，因为：
  - 哈希函数软件执行速度比对称分组密码快。
  - 有许多密码学哈希函数代码库。
- HMAC（RFC2104）给出 HMAC 的设计目标是：
  - 不必修改而直接使用现有的哈希函数。
  - 用新哈希函数替代原来嵌入的哈希函数很容易。
  - 能保持哈希函数原有特性，不过份降低其性能。
  - 对密钥的使用和处理应较简单。
  - 如果已知嵌入的哈希函数的强度，则完全可知认证机制抗密码分析的强度。

HMAC 算法

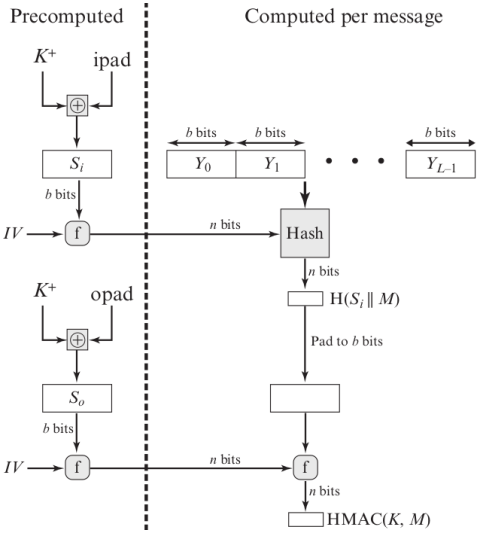
1. 在 K 左边填充 0，得到 b 位的 K^+。
2. K^+ 与 ipad 执行异或，产生 b 位分组 S_i。
3. 将消息 M 附于 S_i 后。
4. 将 H 作用于步骤 3 所得的结果。
5. K^+ 与 opad 执行异或，产生 b 位分组 S_o。
6. 将步骤 4 中的哈希码附于 S_o 后。
7. 将 H 作用于步骤 6 所得的结果并输出：



$$\mathrm{HMAC}(K, M) = H\left[(K^+ \oplus \mathrm{opad})||H[(K^+ \oplus \mathrm{ipad})||M]\right]$$

通过预计算提高 HMAC 的计算效率

- 相比于只对消息 Hash，HMAC 多执行了三次压缩函数 f。
- 对于长消息，HMAC 和嵌入的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。
- 这样只多执行了一次压缩函数。



HMAC 的安全性

- 任何建立在嵌入哈希函数基础上的 MAC，其安全性在某种程度上依赖于该哈希函数的强度。
- 攻击 HMAC 意味着：
  - 对密钥的穷举攻击。
  - 生日攻击。
- M. Bellar 等已经证明，如果攻击者已知若干（时间，消息-MAC）对，则成功攻破 HMAC 的概率等价于对嵌入哈希函数的下列攻击之一：
  - 即使 IV 是随机、秘密和未知的，攻击者也能够计算压缩函数的输出。
  - 即使 IV 是随机、秘密和未知的，攻击者也能够找到哈希函数中的碰撞。

小结

1. 消息认证
2. 密码学哈希函数
3. 安全哈希算法
4. HMAC 算法