雷杰夫斯基的工作

- 德军重复发送密钥所带来的漏洞：
  - 首先有一个记录了一个月中每天所使用的密钥的密码本，然后在当天的每次加密前，选取三个字母表示转子的初始方向，例如ZRW，作为本次加密的临时密钥，用密码本上当天的密钥加密两遍临时密钥，例如ZRWZRW（为了防止偶然的发送或接受错误），之后在用临时密钥加密正文。所以每天德军所发送的密文的前六个字母都是用当天的密钥所加密的重复两遍的临时密钥。
- 假设临时密钥为ZRW，ZRWZRW用当日密钥加密后为ASDFGH。考虑前后两个Z，它们分别被加密为了A和F，这是由于转子在两次加密Z之间转动了三次。这样A和F之间作为同一个字母第一次和第四次被加密的结果，就存在对应关系。通过当天收集大量的密文，可以找出密文第一个和第四个字母间存在的所有的对应关系。

雷杰夫斯基的工作

- 例如：
  - 第一个字母: ABCDEFGHIJKLMNOPQRSTUVWXYZ
  - 第四个字母: FQHPLWOGBMVRXUYCZITNJEASDK
- 在这个表中，存在若干字母循环圈，例如A->F->W->A。写出所有的字母循环圈：
  - A→F→W→A　　3个字母的循环圈
  - B→Q→Z→K→V→E→L→R→I→B　　9个字母的循环圈
  - C→H→G→O→Y→D→P→C　　7个字母的循环圈
  - J→M→X→S→T→N→U→J　　7个字母的循环圈
- 这些循环圈是由当日密钥决定的，但是循环圈的个数、每个循环圈的长度只由转子的情况决定，与接线板的情况无关。
- 循环圈的个数、每个循环圈的长度可以作为转子状态的一个特征。

雷杰夫斯基的工作

- 如果能够构造一个完备的对照表，可以通过这些特征，确定转子的状态。由于这里的分析与连接板状态无关，而转子的状态一共有3!*26^3=105456种，可以采用穷举的关系建立上述的对照表。
- 恢复接线板的设置：模糊文本辨认
- 通过检索表，波兰人在1933年就可以对德国的电报进行破译。
- 缺陷：如果德军不再重复发送密钥，或者增加转子的个数，此方法便不再可行。

图灵的工作

- 二战爆发前后，德军对enigma进行加强改进：
  - 转子数量从3个增加到5个
  - 插线板的最多导线数目从6根变成10根
  - 信息密钥只发送1次
- 利用致命的弱点：一个字母加密后不可能是它本身
- 已知明文攻击：
  - 德国人发报都会有固定的格式，比如在开头的某个位置会有“天气预报”（德语wetterbericht）的字样。可以用这一段已知的明文来做出攻击。
- 如果能确定明文出现的大概位置，完全可以用明文在电文的某个区间逐个比较明文和密文的字母，找到一个位置，满足“明文的所有字母不被加密为自身”，如果明文足够长，这样的位置是容易确定的，因此可以认为找到了这段明文对应的密文。

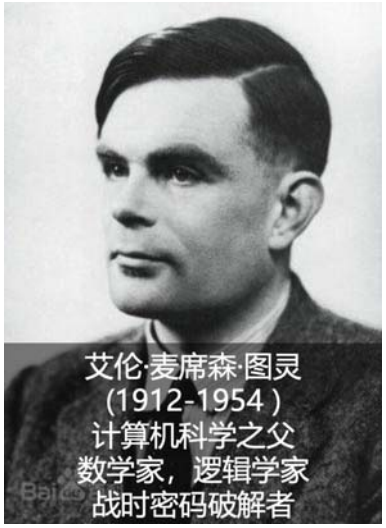
艾伦·图灵

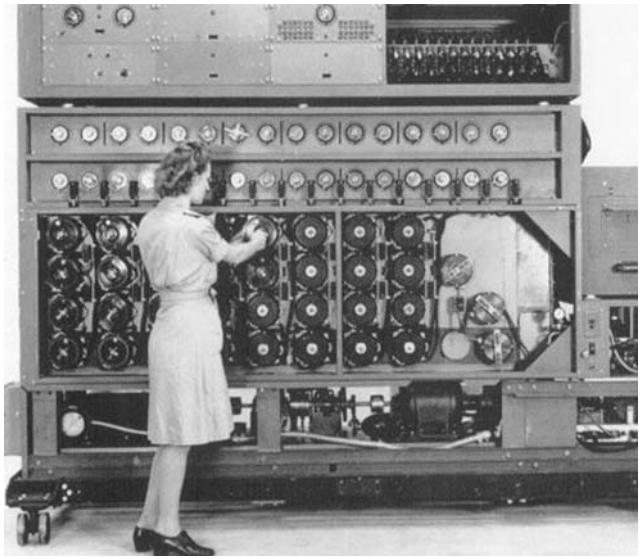
- 英国计算机科学家、数学家、逻辑学家、密码分析学家和理论生物学家，被誉为计算机科学与人工智能之父。
- 二次世界大战期间，设计了一些加速破译德国密码的技术，包括改进波兰战前研制的机器Bombe。图灵在破译截获的编码信息方面发挥了关键作用，使盟军能够在包括大西洋战役在内的许多重要交战中击败轴心国海军，并因此帮助赢得了战争。
- 图灵对于人工智能的发展有诸多贡献，例如图灵曾写过一篇名为《计算机器和智能》的论文，提问“机器会思考吗？”（Can Machines Think?），作为一种用于判定机器是否具有智能的测试方法，即图灵测试。
- 图灵提出的著名的图灵机模型为现代计算机的逻辑工作方式奠定了基础。
- 图灵还是一位世界级的长跑运动员。他的马拉松最好成绩是2小时46分03秒（手动计时），比1948年奥林匹克运动会金牌成绩慢11分钟。1948年的一次越野赛跑中，他跑赢了同年奥运会银牌得主汤姆·理查兹。

Enigma破解

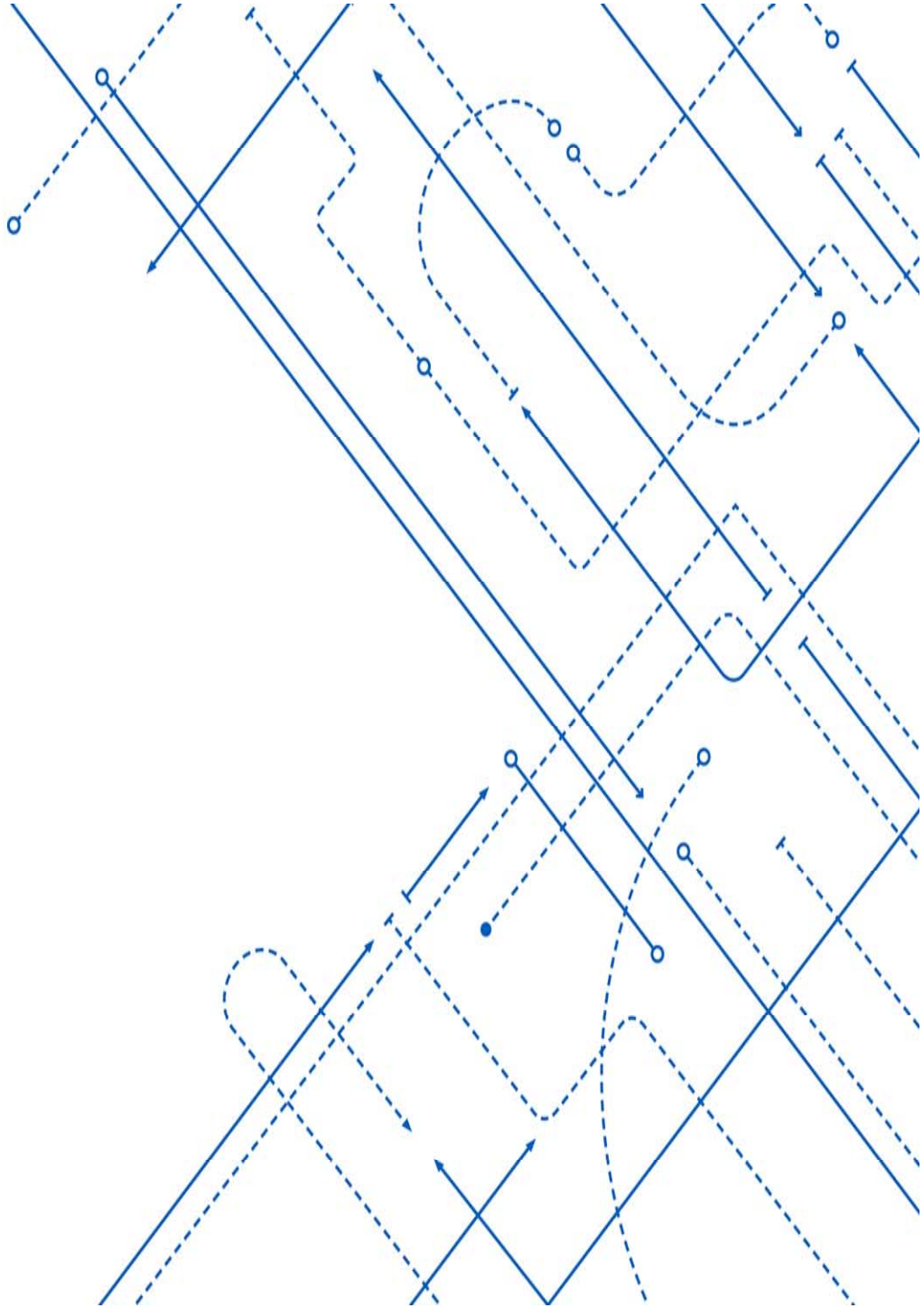
图灵和bombe密码破译机

- bombe是二战期间英国密码学家用来帮助破译德国Enigma密码机的机电设备，复制了几台连接在一起的Enigma机器的行为。
- 最初设计来源于艾伦·图灵。





第3章 Hash函数



浙江大学
ZHEJIANG UNIVERSITY

目录

CONTENTS

Hash函数

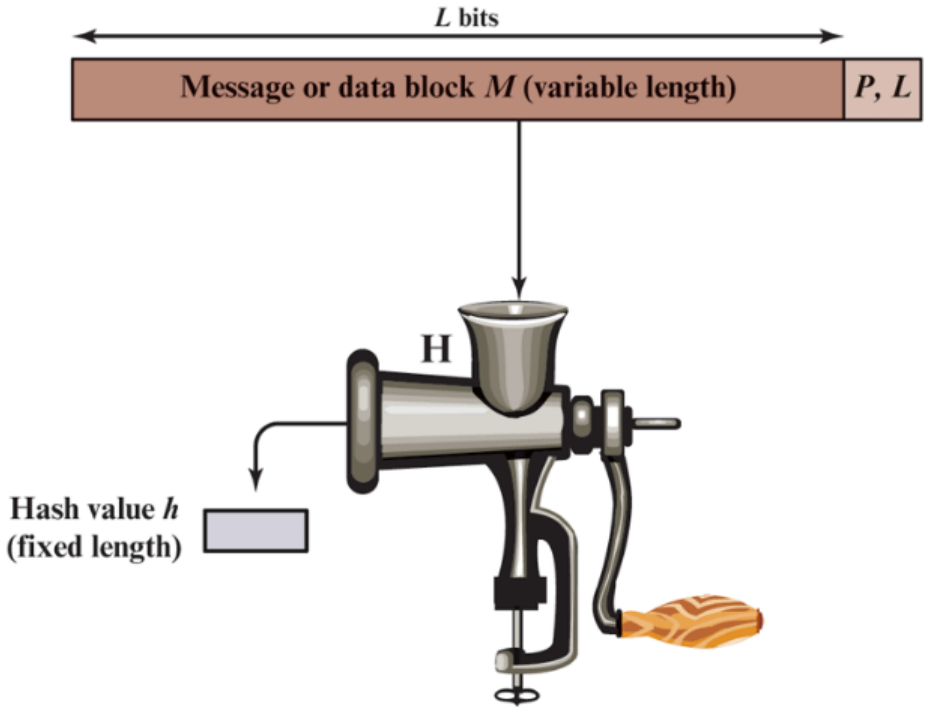
- Hash，一般翻译做散列、杂凑，或音译为哈希。散列函数（英语：Hash function）又称散列算法、哈希函数，是一种从任何一种数据中创建小的数字“指纹”的方法。散列函数把消息或数据压缩成摘要，使得数据量变小，将数据的格式固定下来。该函数将数据打乱混合，重新创建一个叫做散列值（hash values，hash codes，hash sums，或hashes）的指纹。
- hash函数H接受可变长度的数据M作为输入块，并生成固定大小的hash值
  - h = H（M）
  - 主要目标是保证数据完整性
- 密码散列函数（Cryptographic hash function），在计算上无法找到：
  - （a）映射到预先指定的hash结果的数据对象（单向属性）
  - （b）映射到相同哈希结果的两个数据对象（抗碰撞属性）

Hash函数的基本需求

需求	描述
输入长度可变	hash 函数可以应用于任意长度的数据
输出长度固定	hash 函数的输出长度固定
效率	对于任意消息 x，计算 H(x) 很容易
单向性	对于任意哈希值 h，想要找到满足H(x)=h 的 x 在计算上不可行。
抗弱碰撞性	对于任意消息 x，找到满足另一消息 y，满足H(x)=H(y)，在计算上不可行。
抗强碰撞性	找到任意一对满足 H(x)=H(y) 的消息 x 和 y 在计算上不可行。
伪随机性	哈希函数的输出满足伪随机性测试标准。

Hash函数的一般计算过程

- 图中描述了Hash函数的一般操作。
- 通常，输入被填充为某个固定长度的整数倍（例如，1024 位），并且填充位一般包括原始消息的长度值（以位为单位）。



P, *L* = padding plus length field

Hash函数用途

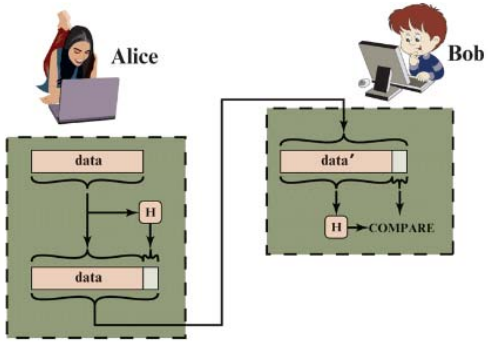
- 消息完整性检查（Message Integrity Check）
  - 发送消息的hash值（摘要）
- 消息身份认证码（Message Authentication Code）
  - 发送消息的keyed hash值
- 数字签名
  - 使用私钥加密哈希（签名）
  - 使用公钥验证哈希（验证）

消息认证（Message authentication）

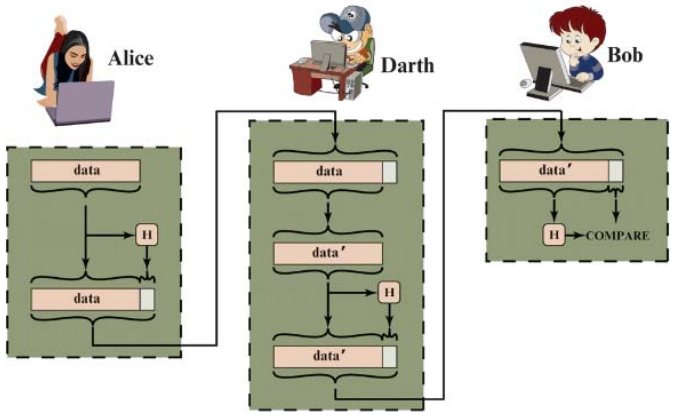
- 消息认证是一种用于验证消息完整性的机制或服务。消息身份验证确保接收到的数据与发送的数据完全相同（即没有修改、插入、删除或重放）。
- 当使用hash函数提供消息认证时，hash函数值通常被称为消息摘要。
- 发送方根据消息计算hash值，并同时传输hash值和消息。接收方对消息执行相同的hash计算，并将该值与传入的hash值进行比较。如果不匹配，接收方就知道消息（或可能是哈希值）已被更改。

消息认证（Message authentication）

- hash函数必须以安全的方式传输，必须保护hash函数，以便如果攻击者更改或替换消息，对手也无法更改hash值以欺骗接收者。
- 在这个例子中，Alice 传输data并附加一个hash值。 Darth 截获消息，更改或替换数据块，并计算并附加一个新的hash值。 Bob 接收到带有新hash值的更改数据，并且没有检测到更改。为了防止这种攻击，必须保护 Alice 生成的哈希值。



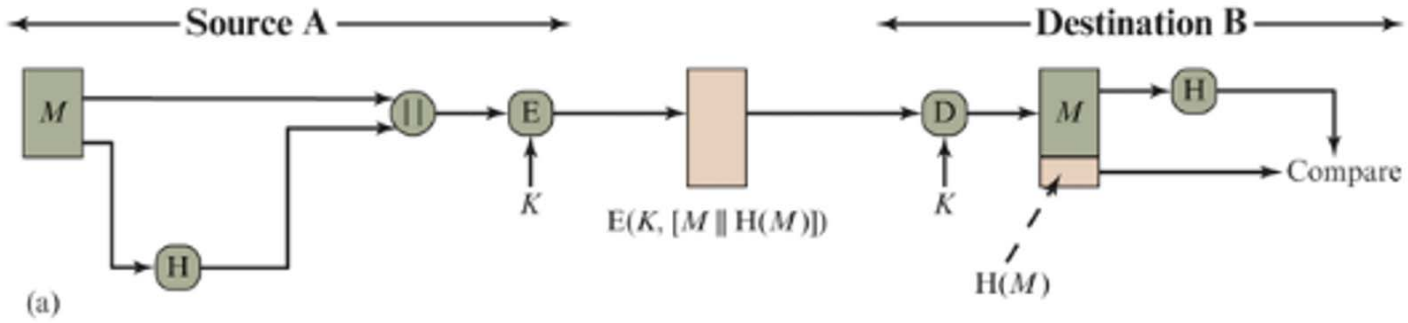
(a) Use of hash function to check data integrity



(b) Man-in-the-middle attack

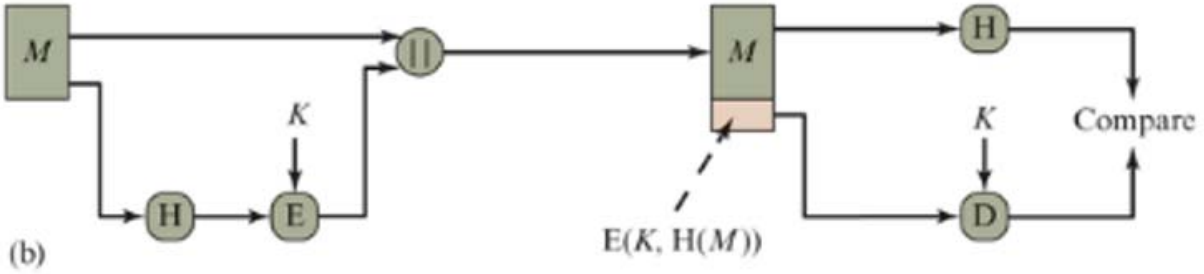
消息认证（Message authentication）

- 使用哈希码提供消息身份验证的多种方式:
- (a) 消息加上哈希值之后使用对称加密进行加密。因为只有 A 和 B 共享密钥，所以消息一定来自 A 并且没有被更改。哈希值提供了实现认证所需的结构或冗余。由于对整个消息加上哈希值进行了加密，因此还提供了机密性。



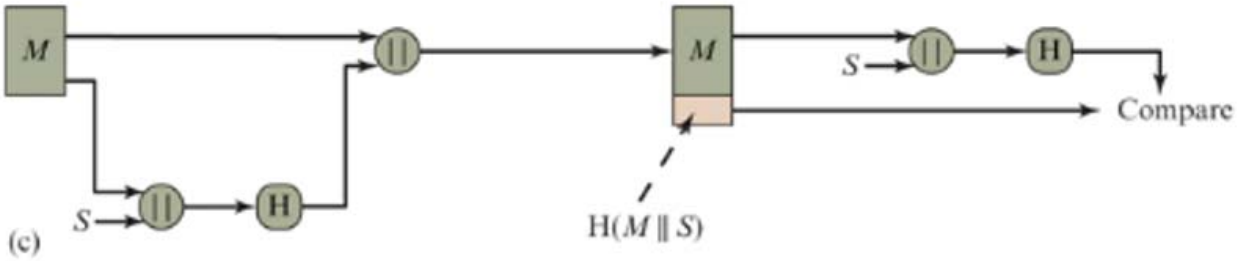
消息认证（Message authentication）

- (b) 只有哈希值使用对称加密。这减轻了那些不需要保密的应用程序的处理负担。



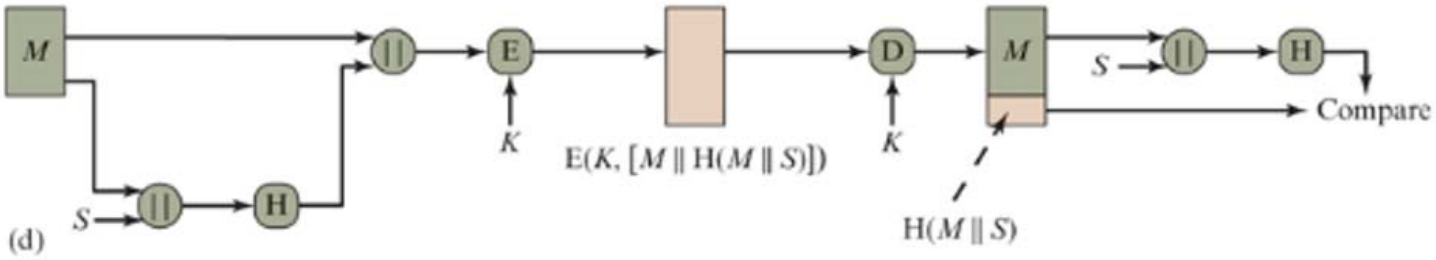
消息认证（Message authentication）

- (c) 使用哈希函数但不加密消息认证。该技术假设通信双方共享一个共同的秘密值S。A 计算M附加S的哈希值，并将得到的哈希值附加到 M上。因为 B 拥有 S，它可以重新计算哈希值来验证。因为秘密值S本身没有被发送，所以攻击者无法修改截获的消息，也无法生成虚假消息。



消息认证（Message authentication）

- (d) 通过对整个消息加上哈希值进行对称加密，可以增强方法（c）的安全性。



Message Authentication Code (MAC)

- 通常MAC在共享密钥的双方之间使用，以验证双方之间交换的信息。
- MAC函数将密钥和数据块作为输入，并生成一个hash值，称为MAC，该hash值与受保护的消息相关联。
- 如果需要检查消息的完整性，可以对消息应用MAC功能，并将结果与相关MAC值进行比较。在不知道密钥的情况下，更改消息的攻击者将无法更改相关的MAC值。

认证加密（Authenticated Encryption）

- 同时保护通信的机密性和真实性（完整性）
- 方法：
- Hash-then-encrypt: E(K, (M || H(M))
- MAC-then-encrypt: E(K2, (M || MAC(K1, M))
- Encrypt-then-MAC: (C=E(K2, M), T=MAC(K1, C)
- Encrypt-and-MAC: (C=E(K2, M), T=MAC(K1, M)
- 解密和验证都很简单。
- 都存在安全漏洞。

HMAC

- 密钥散列消息认证码（英语：Keyed-hash message authentication code），又称散列消息认证码（Hash-based message authentication code，缩写为HMAC），是一种通过特别计算方式之后产生的消息认证码（MAC），使用密码散列函数，同时结合一个加密密钥。它可以用来保证资料的完整性，同时可以用来作某个消息的身份验证。
- 无需修改即可使用可用的哈希函数。尤其是在软件中表现良好的哈希函数，其代码可以自由且广泛地使用。
- 在发现或需要更快更安全的哈希函数时，允许轻松替换嵌入式哈希函数。
- 保持哈希函数的原始性能，而不会导致显著降低性能。
- 以简单的方式使用和处理密钥。

HMAC

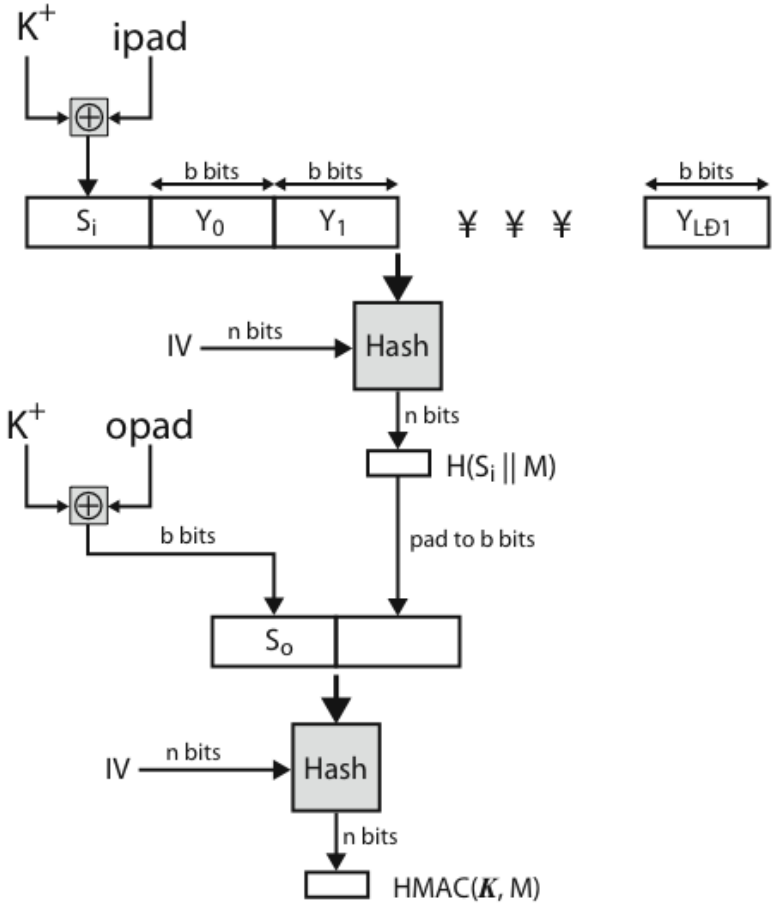
- 根据RFC 2104，HMAC的数学公式为：

$$HMAC(K,m) = H\Big((K' \oplus opad) \,||\, H\big((K' \oplus ipad) \,||\, m\big)\Big)$$

- H为密码散列函数（如SHA家族），K为密钥，m是要认证的消息
- K'是从原始密钥K导出的另一个秘密密钥（如果K短于散列函数的输入块大小，则向右填充（Padding）零；如果比该块大小更长，则对K进行散列）
- || 代表串接，⊕ 代表异或（XOR）
- opad 是外部填充（0x5c5c5c…5c5c，一段十六进制常量）
- ipad 是内部填充（0x363636…3636，一段十六进制常量）

HMAC

- 开销仅比消息本身所需的哈希计算多3次
- 可以使用任何哈希函数
  - 例如MD5, SHA-1, RIPEMD-160, Whirlpool



HMAC-安全性

- HMAC的安全性与底层哈希算法的安全性有关
- 攻击HMAC:
  - 对使用的密钥进行暴力攻击
  - 生日攻击（但需要攻击者使用同一密钥观察大量消息）
- 根据速度与安全性选择使用的哈希函数

生日攻击

- 生日悖论（Birthday Paradox） 不求同年同月同日生?
- 生日攻击是一种密码学攻击手段，所利用的是概率论中生日问题的数学原理。
- 此攻击依赖于在随机攻击中的高碰撞概率和固定置换次数（鸽巢原理）。
- 生日攻击原理:
  - 给定用户准备签署的有效消息 x
  - 攻击者生成 x 的 $2^{m/2}$ 个变体 x'，所有变体的含义基本相同，并将其保存
  - 攻击者生成所需欺诈信息 y 的$2^{m/2}$ 个变体y '
  - 比较两组消息以找到具有相同哈希的对（由于生日悖论，概率>0.5 ）
  - 让用户在有效消息上签名，然后替换具有有效签名的伪造文件
- 结论是需要使用更大的 MAC/hash