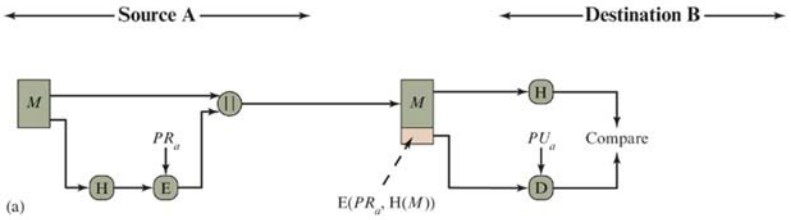
数字签名

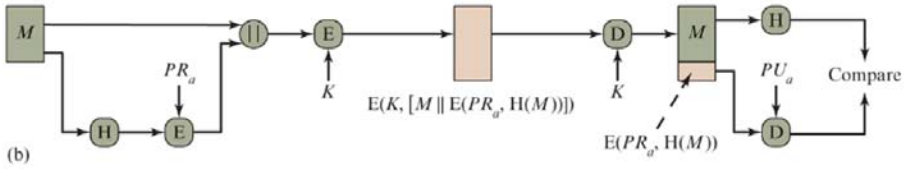
- 数字签名的操作与MAC类似。
- 数字签名使用用户的私钥对消息的哈希值进行加密。
- 任何知道用户公钥的人都可以验证与数字签名相关的消息的完整性。
- 在这种情况下，想要更改消息的攻击者需要知道用户的私钥。

数字签名-示例

- (a) 哈希码使用发送方的私钥进行加密。只有发送方才能生成加密的哈希代码。这就是数字签名的本质。



- (b) 如果需要增加数字签名的保密性，可以使用对称密钥对消息加上私钥加密的哈希码进行对称加密。



常用Hash函数

- 目前常用的 Hash 函数主要有 MD5，SHA1，SHA256，SHA512。目前的大多数 hash 函数都是迭代性的，即使用同一个 hash 函数，不同的参数进行多次迭代运算。

算法类型	输出 Hash 值长度
MD5	128 bit / 256 bit
SHA1	160 bit
SHA256	256 bit
SHA512	512 bit

其他Hash函数的使用

- 伪随机函数（pseudorandom function）
  - 生成会话密钥、随机数
  - 从密码生成密钥
  - 协同从主密钥派生密钥
- 伪随机数生成器（pseudorandom number generator）
  - Vernam 密码/OTP
  - S/Key

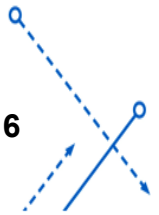
其他Hash函数的使用

- 创建单向密码文件
  - 大多数操作系统（例如，Unix、Windows NT ……）存储密码的哈希而不是实际密码来保护密码。因此，获得密码文件访问权限的黑客无法检索实际密码。当用户输入密码时，会将该密码的哈希值与存储的哈希值进行比较来验证。（etc/passwd, etc/shadow）
  - 彩虹表（rainbow table）

- 用于入侵检测和病毒检测
  - 保留并检查系统上文件的哈希值，可以通过重新计算哈希值检查用于下载的文件是否已经被篡改。

MD5-来源

- MD5 （MD5 Message-Digest Algorithm）由美国密码学家罗纳德·李维斯特（Ronald Linn Rivest）设计，于1992年公开，用以取代MD4算法。这套算法的程序在 RFC 1321 中被加以规范。
- 1992年8月，罗纳德·李维斯特向互联网工程任务组（IETF）提交了一份重要文件，描述了这种算法的原理。由于这种算法的公开性和安全性，在90年代被广泛使用在各种程序语言中，用以确保资料传递无误等。
- MD5由MD4、MD3、MD2改进而来，主要增强算法复杂度和不可逆性。

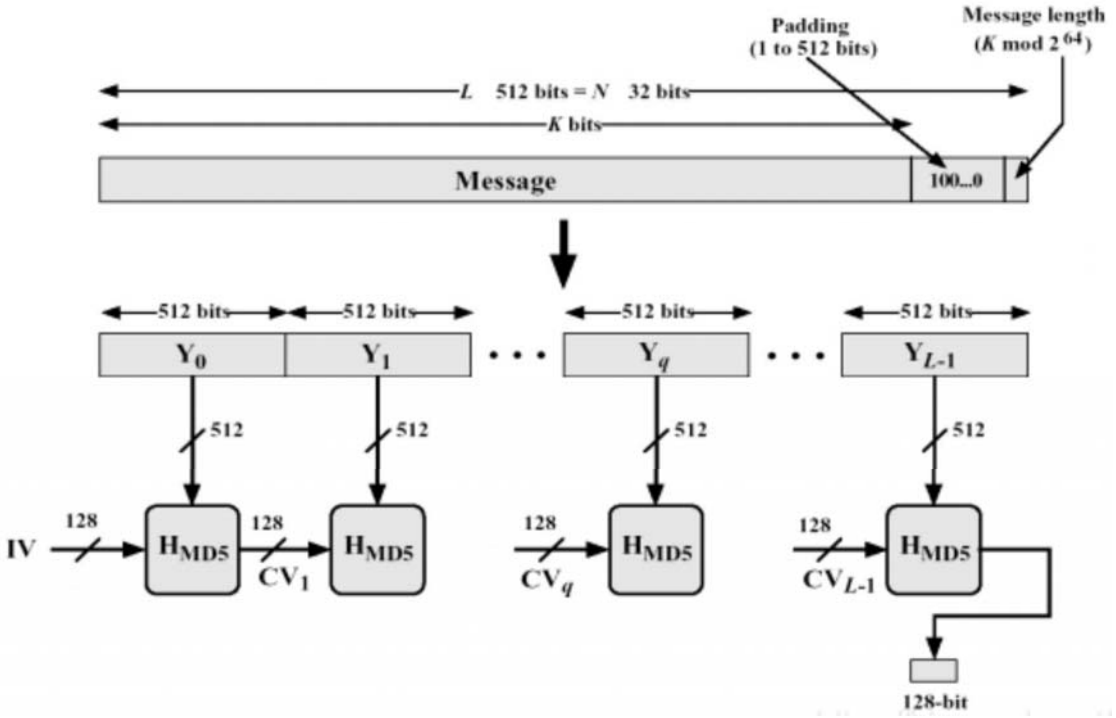


MD5

- MD5消息摘要算法（MD5 Message-Digest Algorithm），一种被广泛使用的密码散列函数，可以产生出一个128位的散列值（hash value），用于确保信息传输完整一致。
- 1996年后被证实存在弱点，可以被加以破解。对于需要高度安全性的数据，专家一般建议改用其他算法，如SHA-2
- 2004年，证实MD5算法无法防止碰撞攻击，因此不适用于安全性认证，如SSL公开密钥认证或是数字签名等用途。

MD5

- 将数据（如一段文字）运算变为另一固定长度值，是散列算法的基础原理。
- 单向函数，从报文到摘要，相当于有损压缩，中间不需要密钥。



MD5计算-分块及填充

- 分块计算,每块大小为64字节。
- 当最后一块大小刚好为64字节时,该块只能算倒数第2块,即它的后面那块(大小为0字节)才算最后一块。故最后块的大小为[0,63]字节,该块需要按以下步骤补充数据凑成64字节的一个块或者128字节的两个块:
- (1)假定该块大小n<56字节,则在末尾补上以下数据:
- 0x80 0x00 0x00 0x00 ... 0x00; 共56-n个
- 例如n==55时，只要补0x80一个字节;
- 当n==54时，要补上0x80及0x00两个字节。

MD5计算-分块及填充

- (2)假定该块大小n在[56,63]范围内时，则应在末尾补上64-n+56个字节。例如当n==56时，应该补上64即8+56个字节；当n==57时，应该补上63即7+56个字节。

- (3)再在后面补上8个字节，这8个字节相当于一个64位的整数，它的值=message总共的位数(不含填充内容)。

MD5计算-分块及填充示例

- 比如有一个文件的内容为'A'，仅一个字节，则:

- A,0x80,0x00,…,0x00, 0x08,0x00,0x00,…,0x00

共补上55个字节

共8字节，其类型为64位整数
=0x00 00 00 00 00 00 00 08
这8个字节以小端格式保存。

MD5计算-分块及填充示例

- 比如，长度为55字节时，就在原文后面填充：一个字节的0x80，再加上8字节的 0xB8,0x01,0x00,…,0x00

  共8字节，其类型为64位整数

  =0x00 00 00 00 00 00 01 B8

  =440

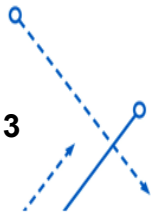
  =55*8

调用openssl库函数进行MD5计算-代码演示

```
#include <openssl/md5.h>
#pragma comment(lib, "libeay32.lib")
#pragma comment(lib, "ssleay32.lib")
main()
{
  int i;
  unsigned char s[100]="Hello", t[100];
  MD5(s, strlen(s), t);
     for(i=0; i<16; i++)
                printf("%02X ", t[i]);
}
```

上述代码的输出结果为:

8B 1A 99 53 C4 61 12 96 A8 27 AB F8 C4 78 04 D7



调用openssl库函数进行MD5计算-代码演示

```
typedef struct _MD5_CTX {
    unsigned long  state[4]; /* 128位摘要 */
    unsigned long  count[2]; /* 已处理的报文的二进制位数,最大值=2^64-1 */
    unsigned char  data[64]; /* 64字节message块 */
}MD5_CTX;
int Init_MD5(MD5_CTX *MD5_ctx);  /*初始化*/
int Update_MD5(MD5_CTX *MD5_ctx, unsigned char *buffer, unsigned long count);
/*分块计算并更新*/
int Final_MD5(MD5_CTX *MD5_ctx); /*为最后一块进行填充操作*/
```

调用openssl库函数进行MD5计算-代码演示

```
#include <openssl/md5.h>
#pragma comment(lib, "libeay32.lib")
#pragma comment(lib, "ssleay32.lib")
main()
{
  int i;
  unsigned char s[100]="Hello", t[100];
  MD5_CTX m; //ctx:context
  MD5_Init(&m);
  MD5_Update(&m, s, 64);
  MD5_Update(&m, s+64, 64);
  MD5_Update(&m, s+64+64, 64);
  MD5_Update(&m, s+64+64+64, 1);

  MD5_Final(t, &m);
  for(i=0; i<16; i++)
       printf("%02X ", t[i]);
}
```

MD5-数字签名

- md5经常与其它算法如RSA结合在一起用于数字签名。
- m = md5(letter);
- m' = rsa(m, A的私钥); // m'就是数字签名
- 假定A把letter及m'发送给B, 则
- B收到信后验证签名:
- rsa(m', A的公钥) == md5(letter)

MD5碰撞

- md5碰撞可以达到骗取数字签名的目的。
- 假定可以找到报文m1及m2(m1≠m2)，使得md5(m1)==md5(m2)，则我们称发生了md5碰撞(collision)。

- 假定有碰撞md5(m1)==md5(m2),其中计算md5(m1)及md5(m2)时所用的4个state种子值均等于md5(letter)完成Final_MD5()前的4个state值，则一定有:
- md5(letter+m1) == md5(letter+m2)
- 因此letter+m1的签名可以用于letter+m2。

MD5破解-rainbow table

- 假定要建立一张包含4个大写英文字母的彩虹表，则可以按以下步骤进行：

for(i=1; i<=SomeBigNumber; i++)

{

①产生一个随机数n_0, 其中$n_0 \in [0, 26^4-1]$，找出与n_0对应的字母组合p_0, 计算m_0=md5(p_0)

for(j=1; j<=100; j++)

{

②计算$n_j = m_{j-1} \bmod 26^4$ /* 消减函数 */

③以n_j为序号,找出与它对应的4个英文字母组合p_j。

例如n=0时，p="AAAA"，n=1时，p="AAAB"，n=25时，p="AAAZ", n=26时，p="AABA"。

④计算m_j = md5(p_j)

}

⑤在数据库中记录步骤①所得的n_0及j==100时步骤④所得的m_{100}。

}

MD5破解-rainbow table

- 上述过程完成后，可以得到SomeBigNumber条链，每条链包含101个md5值。

- 假定M是由某4个大写字母组合生成的md5值，现要破解M究竟是由哪4个英文字母生成的，则可以按以下步骤进行:
- ①在数据库中查找M，若找到，则表示M是j==100时算出来的，于是执行for(j=1; j<=100; j++){②③④}，其中n的初始值n_0是从数据库中取出来的,不能用随机数,当j==100时,步骤③根据n_{100}得到的4个字母组合p_{100}即为所求。
- ②若数据库中找不到M，则设m_j=M，再执行②③④算出m_{j+1}，现再在数据库中查找此m_{j+1}，若找到，则表示M是j==99时算出来的，于是执行for(j=1; j<=99; j++){②③④}，其中j==99时，步骤③根据n_{99}得到的4个字母组合p_{99}极有可能满足md5(p_{99})==M。