2. SHA1

SHA1 是以 512 位分组来处理输入信息的，产生 160 位散列值。

SHA1 算法的计算过程与 MD5 类似，主要包括以下步骤。

（1）预填充

填充方法与 MD5 完全一致。

（2）主循环

主循环的次数为信息中 512 位分组的数目。SHA1 中需要 5 个 32 位的链接变量：A=0x67452301，B=0xefcdab89，C=0x98badcfe，D=0x10325476，E=0xc3d2e1f0。

每个主循环有 4 轮，每轮 20 次操作（MD5 有 4 轮，每轮 16 次操作）。在主循环开始前，先将 5 个链接变量复制到另外 5 个变量中：A 到 a，B 到 b，C 到 c，D 到 d，E 到 e。每次操作对 a、b、c、d、e 中的 3 个进行一次非线性运算，然后进行与 MD5 中类似的移位运算和加运算。

每个主循环完成以后，将 A、B、C、D、E 分别加上 a、b、c、d、e，然后用下一个 512 位分组数据继续进行主循环运算，直至所有分组都完成主循环运算。

（3）输出处理

将最后一个主循环生成的 A、B、C、D、E 这 5 个 32 位值进行级联，生成一个 160 位的摘要值。

SHA1 算法的详细步骤请参见 RFC 3174（US Secure Hash Algorithm 1 (SHA1)）。

验证 SHA1 算法正确性的测试用例如下：

SHA1 ("abc") = A9993E364706816ABA3E25717850C26C9CD0D89D。

SHA1 ("abcdbcdecdefdefgefghfghighijhi jkijkljklmklmnlmnomnopnopq") =
84983E441C3BD26EBAAE4AA1F95129E5E54670F1。

SHA1("a")=34AA973CD4C4DAA4F61EEB2BDBAD27316534016F。

SHA1("01234567012345670123456701234567012345670123456701234567012345670123456701234567")=
DEA356A2CDDD90C7A7ECEDC5EBB563934F460452。

3. SM3

SM3 算法是由中国国家密码管理局于 2012 年 12 月 17 日发布的摘要算法，主要满足电子认证服务系统等应用需求。

针对长度为 l（$l < 2^{64}$）比特的消息 m，SM3 算法经过填充和迭代压缩，生成杂凑值，杂凑值长度为 256 位。其中填充过程为：假设消息 m 的长度为 l 位。首先将位“1”添加到消息的末尾，再添加 k 个“0”，k 是满足 $l + 1 + k = 448 \bmod 512$ 的最小的非负整数；然后再添加一个 64 位的比特串，该比特串是长度 l 的二进制表示；填充后的消息 m' 的比特长度为 512 的倍数。

验证 SM3 算法正确性的测试用例如下：

SM3("abc") = 66c7f0f4 62eeedd9 d1f2d46b dc10e4e2 4167c487 5cf2f7a2 297da02b 8f4ba8e0。

SM3(“61626364 61626364 61626364 61626364 61626364 61626364 61626364 61626364”)= debe9ff9
2275b8a1 38604889 c18e5a4d 6fdb70e5 387e5765 293dcba3 9c0c5732。

5.2 运算模式（工作模式）

运算模式（Mode of Operation）又称工作模式，是指分组密码算法在处理任意长度消息时，对多个分组进行特殊处理的技术方案。常用的运算模式主要有 ECB、CBC、CFB、OFB 等。

5.2.1 ECB

ECB 是 Electronic Code Book（电子密码本）的缩写。

该模式下，每个分组独立进行加密/解密运算，不同分组之间没有任何关联。于是，在相同密钥的情况下，相同的明文总是产生相同的密文。

该模式在安全性方面的特点是：明文模式不能隐藏、分组加密的输入不是随机的（与明文一样）、一个密钥可以加密多个消息、明文很容易篡改（分组可被删除、再现或互换）等。

该模式在效率方面的特点是：速度与分组算法相同、密文比明文长一个分组（由于填充）、不可能进行预处理、处理过程可并行进行等。

该模式在容错性方面的特点是：一个密文错误会影响整个明文分组、同步错误不可恢复等。

5.2.2 CBC

CBC 是 Cipher Block Chaining（密码分组链接）的缩写。

该模式下，在加密当前分组之前，将上一个分组的加密结果与当前明文分组进行异或后，再进行加密，如此形成一个密文链；解密时类似。

该模式在安全性方面的特点是：明文模式能隐藏（通过与前一个密文分组相异或）、分组加密的输入是随机的（与前一个密文分组相异或）、一个密钥可以加密多个消息、篡改明文稍难（分组可以从消息头或尾处删除，第一个分组位可被更换，并且复制允许控制的改变）等。

该模式在效率方面的特点是：速度与分组算法相同、不考虑 IV 时密文比明文长一个分组、不可能进行预处理、加密不是并行的、解密是并行的且具有随机存取特性等。

该模式在容错性方面的特点是：一个密文错误会影响整个明文分组以及下一个分组的相应位、同步错误不可恢复等。

5.2.3 CFB

CFB 是 Cipher Feed-Back（密码反馈）的缩写。

设对称算法的分组长度为 n 位，初始向量为 IV（n 位），密钥为 K，则 x 位 CFB 模式下（x 在 1 与 n 之间），加密过程如下：

① 设定一个 n 位长的队列，队列初始值为 IV，并将明文消息分成若干个 x 位长的比特块。

② 依次对每个 x 位比特块明文进行以下操作：用密钥 K 对队列进行加密，将该密文中最左端的 x 位与 x 位比特块明文异或，即可获得其 x 位比特块密文；然后将该 x 位比特

块密文放入队列的最右端，并丢弃队列最左端的 x 位。

③ 将所有 x 位比特块密文依次级联，即得到消息密文。

解密过程是加密过程的逆过程。

该模式在安全性方面的特点是：明文模式可以隐藏，分组算法的输入是随机的，用不同的 IV，一个密钥可加密多个消息，篡改明文稍难（分组可以被从消息头或尾处删除，第一个分组位可被更换，并且复制允许控制的改变）等。

该模式在效率方面的特点是：速度与分组算法相同，不考虑 IV 时密文与明文大小相同，在分组出现之前做些预处理是可能的，前面的密文分组可以被加密，加密不是并行的，解密是并行的且有随机性存取特性等。

该模式在容错性方面的特点是：一个密文错误会影响明文的相应位及下一个分组，同步错误是可恢复的，1 位 CFB 能恢复 1 位的添加或丢失等。

5.2.4 OFB

OFB 是 Output Feed-Back（输出反馈）的缩写。

该模式类似于 CFB，主要区别是 OFB 将队列加密后密文中最左端的 x 位放入队列最右端，而 CFB 是将 x 位比特块密文放入队列最右端。

设对称算法的分组长度为 n 位，初始向量为 IV（n 位），密钥为 K，则 x 位 OFB 模式下（x 在 1 与 n 之间）的加密过程如下：

① 设定一个 n 位长的队列，队列初始值为 IV，并将明文消息分成若干个 x 位长的比特块。

② 依次对每个 x 位比特块明文进行以下操作：用密钥 K 对队列进行加密，将该密文中最左端的 x 位与 x 位比特块明文异或，即可获得其 x 位比特块密文；同时将队列加密后密文中最左端的 x 位放入队列的最右端，并丢弃队列最左端的 x 位。

③ 将所有 x 位比特块密文依次级联，即得到消息密文。

解密过程是加密过程的逆过程。

该模式在安全性方面的特点是：明文模式可以隐藏，分组算法的输入是随机的，用不同的 IV，一个密钥可加密多个消息，明文很容易被控制篡改，任何对密文的改变都会直接影响明文等。

该模式在效率方面的特点是：速度与分组算法相同，不考虑 IV 时密文与明文大小相同，在分组出现之前做些预处理是可能的，OFB 处理过程不是并行的，计数器处理是并行的，等等。

该模式在容错性方面的特点是：一个密文错误仅影响明文的相应位，同步错误不可恢复等。

5.3 扩展机制

5.3.1 MAC 与 HMAC

MAC 是 Message Authentication Code（消息认证码）的缩写。

MAC 机制的工作原理是：消息发送者使用密码算法和密钥对消息进行处理，生成一个

固定大小的小数据块，该数据块称为 MAC 值。消息接收者通过该 MAC 值来验证消息的完整性和来源。

MAC 机制的具体流程如下：

① 消息发送者和消息接收者预先协商好密码算法和密钥。

② 消息发送者使用已协商的密钥和算法对消息进行处理，生成一个固定大小的 MAC 值。

③ 消息发送者将消息和 MAC 值一起发送给消息接收者。

④ 消息接收者使用已协商的密钥和算法对消息进行处理，生成新的 MAC 值。

⑤ 消息接收者比较新 MAC 值和接收到的 MAC 值，若相同，则确信该消息未被改变且来自消息发送者；若不同，则认为该消息已被改变。

根据使用的密码算法不同，MAC 机制可以分为三类：基于对称算法的 MAC、基于非对称算法的 MAC 和基于 Hash 算法的 MAC。由于已有数字签名机制，一般不使用基于非对称算法的 MAC 机制。目前常用的 MAC 机制有：CBC-DEC-MAC 和 HMAC。前者基于对称算法 DES/3DES，后者基于摘要算法。

1. CBC-DES-MAC

ANSI X9.9 定义 MAC 机制中使用的对称算法为 DES 算法；ANSI X9.19 定义使用双倍长密钥的 3DES 算法计算 MAC 的方法，且它完全与 ANSI X9.9 的单 DES MAC 计算方法兼容。ANSI X9.19 标准使用 MAC 双倍长密钥前半部分对 MAC 数据按 X9.9 计算，然后使用 MAC 密钥后半部分对最后一个分组结果解密计算，再用前半部分加密得到 MAC 值。

ANSI X9.9 和 ANSI X9.19 算法如图 5-5 所示。

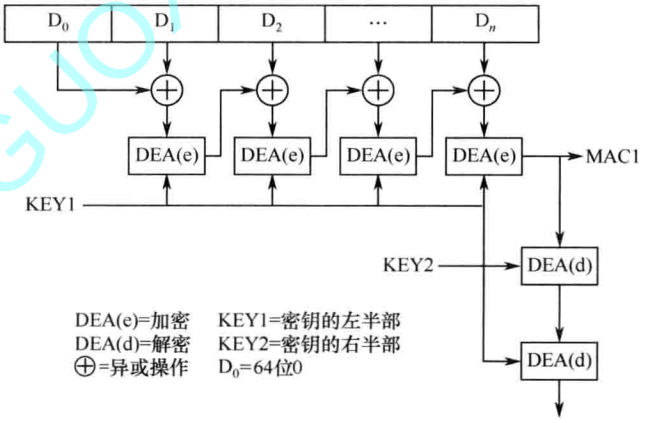

图 5-5　ANSI X9.9 和 ANSI X9.19 算法

其中，MAC1 是 ANSI X9.9 计算的 MAC，MAC2 是 ANSI X9.19 计算的 MAC。对于 MAC 值的长度，ANSI X9.9 等规范要求取最后一个分组密文最左端的 4～8 字节。

不同应用环境下，具体的 MAC 计算方式可能会略有差异。

2. HMAC

HMAC 是 Keyed-Hashing for Message Authentication 的缩写，目前常用的摘要算法是 MD5 或 SHA1。

设 text 表示输入消息，H 表示摘要算法，K 表示密钥，B 表示 H 分组的长度，L 表示 H 输出的长度，ipad=B 个 0x36，opad=B 个 0x5C，则 HMAC 算法可以表示为：

HMAC 码=H(K XOR opad, H(K XOR ipad, text))。

HMAC 的计算流程描述如下：

① 在密钥 K 后面添加 0 来创建一个字长为 B 的字符串。（例如，如果 K 的字长是 20 字节，B=64 字节，则 K 后会加入 44 个“0”字节 0x00）。

② 将上一步生成的 B 字长字符串与 ipad 做异或运算。

③ 将 text 填充至第②步的结果字符串中。

④ 用 H 作用于第③步生成的数据流。

⑤ 将第①步生成的 B 字长字符串与 opad 做异或运算。

⑥ 再将第④步的结果填充进第⑤步的结果中。

⑦ 用 H 作用于第⑥步生成的数据流，输出最终结果，即为 HMAC 值。

5.3.2 OTP

OTP 是 One-Time Password（一次性口令）的缩写，又称动态口令。

OTP 机制的工作原理是：基于密码算法和密钥，对同步因子进行处理后，得到口令；随时变化的同步因子，保证了每次产生的口令均不相同，从而能有效防止口令被攻击或被窃取等安全风险。

OTP 机制的具体流程描述如下：

① 用户和服务器预先协商好密码算法和密钥。

② 用户获取当前同步因子（在本地或从服务器获取），使用已协商的密钥和算法对其进行处理，生成 OTP 码。

③ 用户将 OTP 码发送给服务器。

④ 服务器获取当前用户的当前同步因子，使用已协商的密钥和算法对其进行处理，生成新的 OTP 码。

⑤ 服务器比较新 OTP 码和接收到的 OTP 码，若相同，则确信该用户身份合法；若不同，则认为该用户身份非法。

OTP 机制中的密码算法可以采用对称算法、非对称算法或摘要算法，目前比较常用的是对称算法。用户端产生 OTP 码的系统包括：客户端软件、硬件令牌、手机等。

OTP 机制采用的同步因子主要有以下几类。

1. 时间同步

先保证用户端和服务器端的时间同步，然后基于用户端或服务器端的本地时间来计算动态口令。该方法对服务器端和用户端的时间精度要求很高；用户端常用硬件令牌实现，内嵌高精度的时钟芯片。一般每 30 秒或 60 秒产生一个新的动态口令。

2. 事件同步

先保证用户端和服务器端的事件计数器同步，然后基于用户端或服务器端的本地事件来计算动态口令。每次计算动态口令后，用户端和服务器端的事件计数器均要更新。