**Practical Session 4**

***MD2***

The MD family. These hash functions are MD2, MD4 and MD5, all developed by Ron Rivest (the same Rivest who is the R in RSA). There are also MD1 and MD3, which have never been published. MD2 [1] is particularly easy to describe. It is based around a permutation S of the values 0 to 255, and consists of three steps:

1. **Padding**. The message is increased to be a multiple of 16 bytes: *i* copies of the byte with value *i* are appended to the message, with . Padding is performed as follows: "i" bytes of value "i" are appended (i.e. 1, 2, 3, etc are added to the message is necessary).
2. **Checksum**. The padded message is increased with another 16 bytes  called the checksum. With *N* the length of the padded message (in bytes), set each  and also . *S* is presented at the end of this file and taken from [1]. Then,

for i in range (N/16):

for j in range(16):

c=M[16i+j]

Cj=CjS[cL]

L=Cj

1. **The hash**. Start by initializing 48 bytes  to 0. Then with *N'* being the length of the message M with checksum:

for i in range (N’/16):

for j in range(16):

X[j+16]=M[16\*i+j]

X[j+32]=X[j+16]X[j]

t=0

for j in range(18):

for k in range(48):

t=X[k]S[t]

X[k]=t

t=(t+j)%256

The final hash is the first 16 bytes of X.

This hash is not secure it is too small, and collisions have been found. However the checksum step adds a measure of security that makes collisions harder to find. Its simplicity and ease of description is unique among bit-oriented hashes. Full details about its definition are given in [1] with errata available at [2].

**Test Vectors**

Use the following test vectors [1] to test your own MD2 implementation.

|  |  |
| --- | --- |
| **MD2 test suite** | |
| **Plaintext** | **Hash** |
| ("") | 8350e5a3e24c153df2275c9f80692773 |
| ("a") | 32ec01ec4a6dac72c0ab96fb34c0b5d1 |
| ("abc") | da853b0d3f88d99b30283a69e6ded6bb |
| ("message digest") | ab4f496bfb2a530b219ff33031fe06b0 |
| ("abcdefghijklmnopqrstuvwxyz") | 4e8ddff3650292ab5a4108c3aa47940b |
| ("ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz0123456789") | da33def2a42df13975352846c30338cd |
| ("12345678901234567890123456789012345678901234567890123456789012345678901234567890") | d5976f79d83d3a0dc9806c3c66f3efd8 |

**Activity**

1. Using the provided description and pseudo-code, discuss how this algorithm should be implemented, what would be the best programming language to do it and create a draft of such a code.
2. Upload a file with all that findings individually.

**Implementation**

1. Make your individual submission on Alphagrader your implementation in the programming language of your choice. The testing cases are the ones presented on the table above.

**References**

[1] Burton S. Kaliski. *The MD2 Message Digest Algorithm*. Available at <https://dl.acm.org/purchase.cfm?id=RFC1319>

[2] Jem Berkes and David Hopwood. RFC Errata : RFC-1319 *The MD2 Message Digest Algorithm*. April 2002. Available at <https://www.ietf.org/rfc/rfc1319.txt>

**Appendix**

Permutation of 0..255, called S, constructed from the digits of π. It gives a "random" nonlinear byte substitution operation.

static unsigned char PI\_SUBST[256] = {

41, 46, 67, 201, 162, 216, 124, 1, 61, 54, 84, 161, 236, 240, 6,

19, 98, 167, 5, 243, 192, 199, 115, 140, 152, 147, 43, 217, 188,

76, 130, 202, 30, 155, 87, 60, 253, 212, 224, 22, 103, 66, 111, 24,

138, 23, 229, 18, 190, 78, 196, 214, 218, 158, 222, 73, 160, 251,

245, 142, 187, 47, 238, 122, 169, 104, 121, 145, 21, 178, 7, 63,

148, 194, 16, 137, 11, 34, 95, 33, 128, 127, 93, 154, 90, 144, 50,

39, 53, 62, 204, 231, 191, 247, 151, 3, 255, 25, 48, 179, 72, 165,

181, 209, 215, 94, 146, 42, 172, 86, 170, 198, 79, 184, 56, 210,

150, 164, 125, 182, 118, 252, 107, 226, 156, 116, 4, 241, 69, 157,

112, 89, 100, 113, 135, 32, 134, 91, 207, 101, 230, 45, 168, 2, 27,

96, 37, 173, 174, 176, 185, 246, 28, 70, 97, 105, 52, 64, 126, 15,

85, 71, 163, 35, 221, 81, 175, 58, 195, 92, 249, 206, 186, 197,

234, 38, 44, 83, 13, 110, 133, 40, 132, 9, 211, 223, 205, 244, 65,

129, 77, 82, 106, 220, 55, 200, 108, 193, 171, 250, 36, 225, 123,

8, 12, 189, 177, 74, 120, 136, 149, 139, 227, 99, 232, 109, 233,

203, 213, 254, 59, 0, 29, 57, 242, 239, 183, 14, 102, 88, 208, 228,

166, 119, 114, 248, 235, 117, 75, 10, 49, 68, 80, 180, 143, 237,

31, 26, 219, 153, 141, 51, 159, 17, 131, 20

};