RC21471(96898)7MAY1999

ComputerScience/Mathematics

Research Report

Rationale for Bit Fixing in the MDC-2 Algorithm

Don Coppersmith, Stephen M. Matyas, Mohammad Peyravian
IBM T. J. Watosn Research Center
P. O. Box 218
Yorktown Heights, NY 10598

Research Division
Almaden - Austin - Beijing - Haifa - T. J. Watson - Tokyo - Zurich

ABSTRACT

The MDC-2 algorithm is a hash function for computing a hash value on input data whose length L is a multiple of 64 bits and where $L \geq 128$. With the MDC-2 algorithm, outputs computed in each step of the algorithm are used, after being modified slightly, as key values in the next step of the algorithm. More particularly, the modification consists of fixing certain bits in these values before they are used as keys. This technical report provides the rationale for the bit-fixing operation used within the MDC-2 algorithm.

KEYWORDS

- cryptography
- hash function
- analysis
- MDC-2

1 Background

The MDC-2 algorithm is a hash function that computes a 128-bit hash value on an input X consisting of n 64-bit data blocks, X = X1, X2, ..., Xn, where n > 1. An input X that does not adhere to this length restriction must first be padded via a suitable padding algorithm in order that its length will conform to the length restriction. A suitable padding algorithm is described below.

The MDC-2 algorithm is so-named because two DEA encryptions are performed for each 64-bit block of input plaintext processed by the algorithm (see Fig. 1). At step "i", an input block Xi is DES-encrypted with two key values Ki and Li, and the input Xi is then Exclusive-ORed with each respective ciphertext value to produce 64-bit output values (Ai, Bi) and (Ci, Di), as follows:

$$e(Ki, Xi) \oplus Xi = Ai \,||\, Bi$$
$$e(Li, Xi) \oplus Xi = Ci \,||\, Di$$

The notation e(K, X) denotes a 64-bit ciphertext produced by DES-encrypting a 64-bit plaintext X with a 64-bit key K, where K contains 56 independent key bits and 8 bits that may be used for parity. The symbol "⊕" denotes the Exclusive-OR operation and the symbol "||" denotes concatenation.

A 32-bit swap function is then used to generate Ki+1 and Li+1 from (Ai, Bi) and (Ci, Di), as follows:

Ki+1 = Ai || Di where bits 1 and 2 are set to B'10'.
Li+1 = Ci || Bi where bits 1 and 2 are set to B'01'.

Note that bits are numbered 0, 1, 2, etc. from most significant to least significant. The initial or starting values of K1 and L1 are defined as follows:

K1 = X'5252525252525252'

L1 = X'2525252525252525'

The 128-bit hash value (denoted MDC) produced with the MDC-2 algorithm is the output at step "n", as follows:

MDC = An || Dn || Cn || Bn

As one can see, the initial or starting values K1 and L1 are used only to process the first 64-bit block of plaintext, X1. Thereafter, input values K2, K3, ..., Kn are derived from output values (A1, D1), (A2, D2), ..., (An-1, Dn-1), respectively, and input values L2, L3, ..., Ln are derived from output values (C1, B1), (C2, B2), ..., (Cn-1, Bn-1), respectively. In other words, the outputs of each iteration are fed back, modified slightly, and used as the keys at the next iteration. The 32-bit swapping function merely interchanges Bi and Di.

1.1 Padding Considerations

The MDC-2 algorithm processes data in multiples of 64 bits, with a 128-bit minimum. No padding is performed by the algorithm, although such padding could be performed as a service by either hardware or software. When padding is required, a padding algorithm f should be used

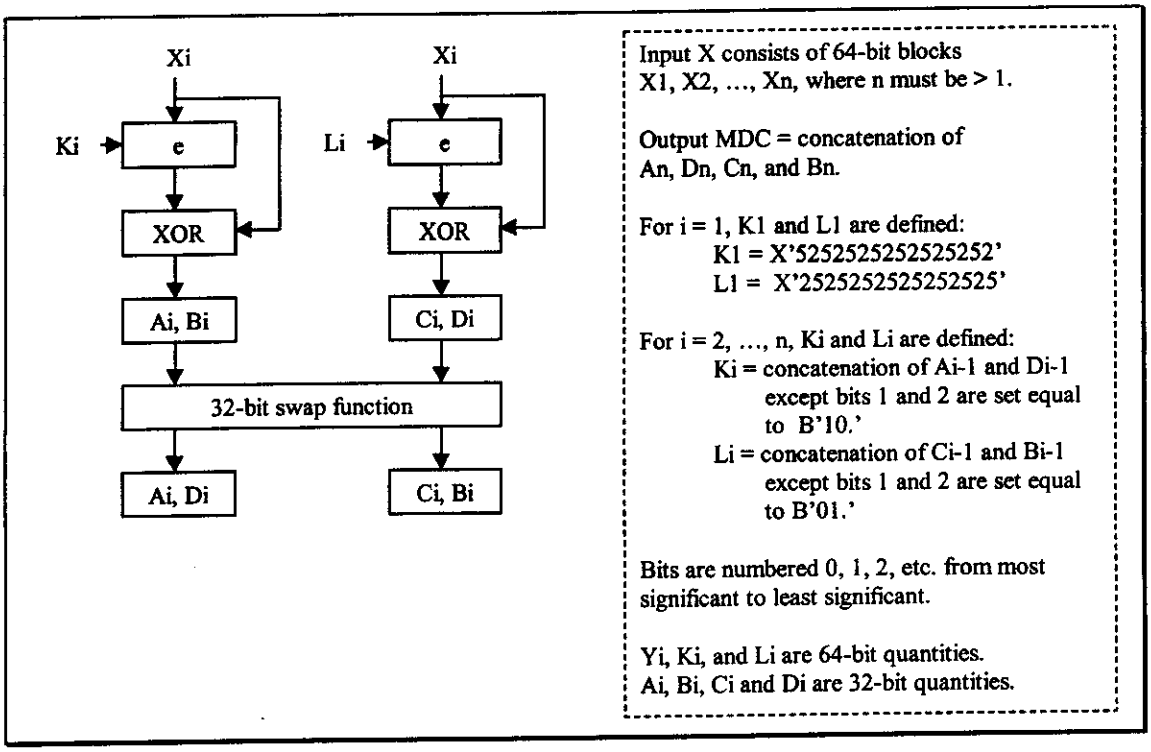

Figure 1: MDC-2 algorithm

that is guaranteed not to produce synonyms. That is, if Y and Y' are two different data inputs, the padded value of Y must not equal the padded value of Y', or mathematically speaking, $Y \neq Y'$ guarantees that $f(Y) \neq f(Y')$. A padding algorithm satisfying this requirement is given below. The method that requires the input to consist of a whole number of bytes is based on a padding rule described in ANSI X9.23 [3]. (For convenience, the rule is described in terms of bytes, not bits.) If the data length is less than 8 bytes, pad bytes are added to make the data length 16 bytes. If the data length is 8 or more bytes, pad bytes are added to make the data length a multiple of 8 bytes. Padding is done even if the current data length is a multiple of 8 bytes. All pad bytes except the last pad byte contain a value of X'FF.' The last pad byte is a pad count (in hexadecimal) of the total number of pad bytes, including the pad byte containing the pad count.

To illustrate the problem of synonyms, suppose that the above padding rule is followed, except that padding is not performed when the data length is already a multiple 8 bytes. Thus, an input Y equal to X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF01' (32 hex characters) is not padded, since its length is already a multiple of 8 bytes. But an input Y' equal to X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF' (30 hex characters) is padded with X'01' to produce a value X'FFFFFFFFFFFFFFFFFFFFFFFFFFFFFF01' (32 hex characters) equal to Y. Thus, inputs Y and Y' yield the same padded values and hence produce the same MDC values.

2 Rationale for Bit Fixing

In this section, we provide the rationale for the bit fixing operation used within the MDC-2 algorithm. The bit fixing operation consists of forcing bits 1 and 2 in Ki (for i = 0, 1, 2, etc.) to the value B'10' and forcing bits 1 and 2 in Li (for i = 0, 1, 2, etc.) to the value B'01'.

In very broad terms, the bit fixing operation serves the following two purposes:

1. It guarantees, at each step of the algorithm, that Ki is not equal to Li (i.e., $Ki \neq Li$). If it were the case that the MDC-2 algorithm did not prevent K from equaling L, and it happened by chance that K = L at some step, say step "i", then it would also be the case that K = L for every step i+1, i+2, etc. thereafter. That is, if Ki=Li then (Ai||Bi)=(Ci||Di), which implies (Ai||Di)=(Ci||Bi) and (Ki+1)=(Li+1). The situation would persist through the remainder of

the computation. Instead of carrying 112=2×56 bits of information from one round to the next, we are now carrying only 56 bits of information. The security becomes that of a 56-bit hash: 2^{28} operations to find a collision, 2^{56} operations to match a given hash value (in the same restricted range).

2. The main building block of the MDC-2 algorithm is the mapping f(K,X) = e(K, X) ⊕ X. Except for the 4 conditions noted below, which we call the four exceptions, it is apparently very difficult, for a given value Y, to find values K and X such that f(K,X) = Y. The only known way to find such a (K, X), except for the four exceptions noted below, is by trial and error, which requires about 2^{64} trials. Even if one pair of values (K,X) is given such that f(K,X) = Y, the task of finding a second pair (K',X') such that f(K',X') = f(K,X) = Y, apparently requires 2^{64} trials. The four exceptions are noted below:

   a) If Kc denotes the complement of K, one obtains f(K,X) = f(Kc,Xc) due to the complementary property of the DES [4]. To show this, we note that by the complementary property, e(K,X) = [e(Kc,Xc)]c = e(Kc,Xc) ⊕ 1...1. If we Exclusive-OR X on each side of the equation, we obtain e(K,X) ⊕ X = e(Kc,Xc) ⊕ X ⊕ 1...1. But, X ⊕ 1...1 = Xc, thus giving e(K,X) ⊕ X = e(Kc,Xc) ⊕ Xc, which is the expanded form of f(K,X) = f(Kc,Xc).

   b) If K is one of the four "weak keys" [4], then f(K,X) = f(K,e(K,X)). Note that e(K,e(K,X)) = X if K is a weak key.

   c) If K is one of the four "weak keys," there are 2^{32} values of X (easily found) such that f(K,X) = 0...0 (64 bits of 0s).

   d) If K is one of the four "semi-weak keys" alternating in each 28-bit internal half-key, C and D, i.e., where one of the four cases: (i) C = (01...01) and D = (01...01) or (ii) C = (01...01) and D = (10...10) or (iii) C = (10...10) and D = (01...01) or (iv) C = (10...10) and D = (10...10), then there are 2^{32} values of X (easily found) such that f(K,X) = 1...1 (64 bits of 1s).

The properties 2c and 2d follow from reference [5].

Except for the four exceptions (2a, 2b, 2c, and 2d), it is apparently also difficult (2^{64} trials) to find a value of X, given K and Y, such that f(K,X) = Y, or to find a value of K, given X and Y, such that f(K,X) = Y.

The special properties of the "weak keys" (2b and 2c) and "semi-weak" alternating keys (2d) make it desirable to avoid these keys in the key positions. Undesirable keys are eliminated in the MDC-2 algorithm by fixing bits 1 and 2 in Ki to B'10' and fixing bits 1 and 2 in Li to B'01'. This reduces the effective key length from 56 bits to 54 bits.

We now examine the four exception cases (2a, 2b, 2c and 2d).

2.1 Exception Case 2a

The keys Ki have bits 1 and 2 fixed to B'10'. This guarantees that a key K and its complement Kc will not both be used as Ki or Kj during the processing of a message M, or of two different messages M and M'. We will not have Ki=K and Kj=Kc, because both Ki and Kj have bits 1 and 2 fixed to B'10'. A similar condition holds for keys Li and Lj.

The complement of a key Ki can occur as a key Lj, since (B'10')c=B'01'. This does not appear to lead to any weakness. As an example, an adversary could arrange that, during the processing of two messages X and X', we have (Ki)c=Lj' and (Li)c=Kj'. To take advantage of Exception case 2a, the adversary would arrange that Xi=Xj', so that f(Ki,Xi)=f(Lj',Xj'), and also f(Li,Xi)=f(Kj',Xj'). Then (Ai||Bi)=(Cj'||Dj'), and (Ci||Di)=(Aj'||Bj'). But the symmetry would be destroyed on the next round: Ki+1=(Ai||Di) with bits 1 and 2 set to B'10', while Lj+1'=(Cj'||Bj') with bits 1 and 2 set to B'01'. Thus Ki+1 and Lj+1' would agree everywhere except the two bits 1 and 2, where they disagree. Further encryptions with these two keys would be unrelated and the attack would fail.

2.2 Exception Cases 2b and 2c

There are four possible "weak keys." A weak key is one for which the two 28-bit internal half-

keys are defined as follows: (i) C = 0...0 and D = 0...0, or (ii) C = 0...0 and D = 1...1 or (iii) C = 1...1 and C = 0...0 or (iv) C = 1...1 and D = 1...1. However, bits 1 and 2 in the external key both end up in the half-key C. For both external keys K and L, the B'10' or B'01' in bits 1 and 2 guarantees that the half-key C cannot be all 0s or all 1s, which in turn guarantees that K and L cannot be "weak keys."

2.3 Exception Case 2d

Fixing bits 1 and 2 to B'10' in K and B'01' in L guarantees that the 28-bit internal half-key C cannot be alternating. This is so because bits 1 and 2 in the external key are always both in odd positions or both in even positions in the half-key C, at each round of DES encryption. Hence, this guarantees that K and L cannot be one of the four "semi-weak" alternating keys.

3 References

[1] D. Coppersmith, S. Pilpel, C.H. Meyer, S.M. Matyas, M.M. Hyden, J. Oseas, B. Brachtl, and M. Schilling, *Data Authentication Using Modification Detection Codes Based on a Public One Way Encryption Function*, U.S. Patent No. 4,908,861 (March 13, 1990).

[2] C.H. Meyer and M. Schilling, Secure program load with modification detection code, *Proceedings of the Fifth Worldwide Congress on Computer and Communications Security and Protection (SECURICOM 88 — SEDEP),* 8, Rue de la Michodiere, 75002 Paris, pp. 111-130.

[3] American National Standard X9.23-1995, *Encryption of Wholesale Financial Messages*, American Bankers Association, Washington, D.C., 1995.

[4] C.H. Meyer and S.M. Matyas, *Cryptography: A New Dimension in Computer Data Security*, John Wiley & Sons, Inc., New York, NY, (1982), pp. 517-577.

[5] D. Coppersmith, "The Real Reason for Rivest's Phenomenon," *Advances in Cryptology – CRYPTO '85*, Springer-Verlag, Lecture notes in Computer Science, Vol. 218, pp 535-536.