2n-BIT HASH-FUNCTIONS USING n-BIT SYMMETRIC BLOCK CIPHER ALGORITHMS

Jean-Jacques Quisquater[1] *Marc Girault*[2]

[1] *Philips Research Laboratory*
Avenue Van Becelaere, 2
B-1170 Brussels, Belgium

[2] *Service d'Etudes communes des Postes et Télécommunications*
42 rue des Coutures
BP6243, 14066 Caen, France

ABSTRACT

We present a new hash-function, which provides 2n-bit hash-results, using any n-bit symmetric block cipher algorithm. This hash-function can be considered as a extension of an already known one, which only provided n-bit hash-results. The difference is crucial, because a lot of symmetric block cipher algorithms use 64-bit blocks and recent works have shown that a 64-bit hash-result is greatly insufficient.

J.J. Quisquater and J. Vandewalle (Eds.): Advances in Cryptology - EUROCRYPT '89, LNCS 434, pp. 102-109, 1990.

1. INTRODUCTION

Public-key systems provide methods of producing digital signatures of messages. Nonetheless, if these systems are used according to their primitive description, the signature produced is at least as long as the message itself and production time may be very high. This is the reason why one generally prefers to efficiently reduce the message to a short imprint, prior to applying the secret function of the public-key system. Such a reduction must be carefully designed, so that it does not introduce any weakness in the resulting digital signature scheme.

The functions which achieve this sort of reduction are often called hash-functions and may be defined as cryptographically secure methods of computing a fixed-length imprint of a message. A signature of this message can thereafter be generated by applying the secret function of a digital signature scheme to the imprint instead of applying it to the whole message.

More precisely, a hash-function is said secure if it is collision-free, i.e. if it is computationally infeasible to construct distinct messages which hash to the same imprint. Generally speaking, the collision-free property requires that the size h of the imprint be at least about 100 bits (say 128 bits, to preserve a safety margin). Indeed if it is much smaller (for example 64 bits), an attack exists which allows to efficiently construct distinct messages with the same imprint.

This attack, due to Yuval [Yu], consists in preparing two sets of $2^{h/2}$ messages. Each of these sets can be easily built by creating a few (h/2) variations of a unique message and by combining them together. It can be shown that the probability of finding a message M_1 from the first set and a message M_2 from the second set which have the same imprint is greater than 1/2. Now, such twin messages can be found by sorting the imprints of the first set and matching them with each imprint of the second set.

As best known (sequential) techniques of sorting are of time complexity O(NlogN), where N is the size of the list, it appears that h/2 should be greater than 32 in order to make this type of attack computationnally infeasible. This leads to a convenient length of more than 64 bits for the imprint. Moreover, new techniques due to Quisquater and Delescaille [QD] allow both to avoid sorting step and to use very few memory space, so that the so-called twin messages can be found in a much more efficient manner. Therefore, it appears reasonable to require the imprint to be (say) 128-bit long.

This size can however be reduced in some cases, due to the following reasons:

- the above mentioned attack is no longer effective if a random number is systematically inserted in the messages to be hashed, or if the initializing vector I is randomly chosen; nonetheless, it must be pointed out that the attack remains effective for the signer himself, since I is chosen by him.

- the collision-free property is not required in some applications, when it appears that the opponent has no practical way to profit from the collisions he found. Even in this case, however, the hash-function has still to be one-way in the following strong sense: given a message M and its imprint H, it must be computationally infeasible to find another message M' with the same imprint H; for that reason, a mimimum size of 64 bits is required.

Nevertheless, a size of 128 bits or more appears (nowadays) to be secure for all types of applications. Various authors have, in the past, also recommended such a size (e.g. [Ju]).

2. THE "SINGLE-LENGTH" HASH-FUNCTIONS

Much attention has been paid to hash-functions based on a symmetric block-cipher algorithm, generally DES. But until now, only schemes providing imprints of length equal to n (the block-length of the cipher algorithm) have been proposed. As n is often equal to 64, it results from the discussion of section 1 that such schemes are not secure enough from a general point of view.

For example, the following scheme (attributed sometimes to Davies, sometimes to Meyer), which we will call DM, is a good example of a "single-length" hash-function using DES [DP]: first, the message M is split into 56-bit blocks M_1, M_2, ..., M_r. Then, the imprint H is calculated in the following iterative way (where (+) stands for bitwise Exclusive-OR and $DES_K(X)$ denotes the encipherment of X with key K):

$$H_0 = I \qquad \text{(initializing value)}$$

then:
$$H_i = DES_{Mi}(H_{i-1}) \,(+)\, H_{i-1} \qquad \text{for i from 1 to r}$$

The imprint is $H = H_r$.

This hash-function is as good as possible (apart from the fact that weak and semi-weak keys should be avoided; see [MS]). In particular, it seems to be resistant to a "meet-in-the-middle" attack ([DP] or [Co]). This sort of attack must be considered with care, especially after very recent results [QD2] which show its efficiency when implemented with a right time-memory trade-off.

In fact, the only default of DM-scheme is to provide too short imprints. So the question is: can we specify an efficient and secure scheme which provides twice as long imprints, still using a 64-bit cipher algorithm?

3. OUR PROPOSAL: A "DOUBLE-LENGTH" HASH-FUNCTION

3.1 General

We propose here a secure scheme which provides 2n-bit imprints using n-bit block cipher algorithms. Moreover, computation time of this scheme is almost the same as for DM (or similar) scheme, contrary to some other ones [MS]. In that way, we can answer "yes" to the question raised at the end of the previous section.

This scheme is the "good" generalization of DM scheme, in that it uses also feedforward techniques to avoid meet-in-the-middle attacks, and is specified in such a way that all the possible attacks (exhaustive or birthday ones) require a number of steps which is the square of the number required in the DM scheme.

As in DM scheme, the number of encipherments is equal to the number of blocks of the message to be hashed. The other operations are only bitwise Exclusive-Or and addition modulo 2^k-1, so that the scheme is almost as efficient as DM scheme, while it offers a much greater security.

This scheme is general, in that it can a priori use any symmetric block cipher algorithm. But it must be kept in memory that any weakness in this algorithm will probably induce a weakness in the scheme itself (e.g. weak or semi-weak keys of DES).

3.2 Informal description

The major problem to solve, which is inherent to the goal we wish to achieve, is the following one: since the basic operation is an n-bit encipherment, it is a priori still possible to perform a "local" attack at this n-bit level. This is particularly true after

Quisquater and Delescaille results already mentioned [QD], which show that one or two hours may suffice to find a DES collision, i.e., given a value I, two distinct keys K and K' such that $DES_K(I) = DES_{K'}(I)$. This attack only requires one very fast DES chip (or ten moderately fast DES chips!) and a personal computer to pilot this chip.

This statement has the following incidence: each block of the message should appear at least twice along the hashing operation, in a form or another. Meyer and Schilling [MS] propose that each block be involved in two encipherments, but this leads to a computation time which is twice as long as computation time of DM scheme.

We rather suggest to introduce two supplementary blocks at the end of the message; each of these blocks is dependent on all the preceding significant blocks, calculated by very simple (but as "independent" as possible) functions.

The basic step of our hash-function is composed of two encipherments (with blocks M_{2i+1} and M_{2i+2}), followed by Exclusive-Or operation with the hash-result which was obtained at the end of the previous step ($H_{2i-1}||H_{2i}$) to provide the new current hash-result ($H_{2i+1}||H_{2i+2}$), where || is the symbol for concatenation. This feedforward connection is the analogue of the feedforward connection of DM scheme. Its role is to prevent from going backwards in the hash-function, in order to defeat meet-in-the middle attacks.

3.3 Formal description

Let e be a symmetric block-cipher algorithm, whose block-length is n and key-length is k (for example, n=64 and k=56 if e is DES). We denote the encipherment of input X under key K by eK(X). Let I and J be two n-bit initializing values, preferably chosen at random. Then, the imprint H of a binary message M is calculated in four steps.

Step 1 (splitting):
M is split into k-bit blocks $M_1, M_2, \ldots$

Step 2 (first completion):
If the number of blocks is even, a supplementary block filled with '0's is added. Let n=2m be the number of blocks at the end of this step.

Step 3 (second completion):
Two supplementary blocks are added to the message. The first one, M_{n+1}, is equal to the Exclusive-Or of all the preceding blocks:

$M_{n+1} = M_1 (+) M_2 (+) ... (+) M_n$

The second one, M_{n+2}, is equal to the addition modulo 2^k-1 of the same blocks, seen as integers expressed in base 2:

$M_{n+2} = M_1 + M_2 + ... + M_n \quad \text{modulo } 2^k\text{-}1$

Step 4 (iteration): The output values $H_1, H_2, ..., H_{n+1}, H_{n+2}$ are calculated in the following iterative way (see figure 1) :

$H_{-1} = I$ $\qquad$ $H_0 = J$

$H'_{2i+1} = eM_{2i+1}(H_{2i-1}) (+) H_{2i}$ $\qquad$ $H'_{2i+2} = eM_{2i+2}(H'_{2i+1}) (+) H_{2i-1}$

$H_{2i+1} = H'_{2i+2} (+) H_{2i}$ $\qquad$ $H_{2i+2} = H'_{2i+1} (+) H_{2i-1}$

for i from 0 to m.

The imprint is $H = H_{n+1} || H_{n+2}$.

3.4 Remark

Another iteration step had been suggested at the time of Eurocrypt'89 conference, but it was later shown to be weak by D. Coppersmith [Co2].

4. CONCLUSION

We have described a hash-function providing 2n-bit imprints, using a n-bit symmetric block cipher algorithm. In that way, Yuval's attack requires 2^n calculations (instead of only $2^{n/2}$), a prohitive number if, e.g., n=64 (there are more than 500,000 years in 2^{64} microseconds). Meet-in-the-middle attacks are made impossible, because of a feedforward connection similar to the one of Davies-Meyer scheme. Other attacks are also rendered unpractical, because of two supplementary blocks which are introduced at the end of the message

5. BIBLIOGRAPHY

[Co] D. Coppersmith, "Another birthday attack", Advances in Cryptology, Proc. of Crypto '85, LNCS, Vol. 218, Springer-Verlag, 1986, pp. 14-17.

[Co2] D. Coppersmith, private communication.

[DP] D.W. Davies and W.L. Price, "Security for computer networks", ed. J. Wiley & Sons, 1984.

[Ju] R.R. Jueneman, "A high speed Manipulation Detection Code", Advances in Cryptology, Proc. of Crypto '86, LNCS, Vol. 263, Springer-Verlag, 1987, pp. 327-346.

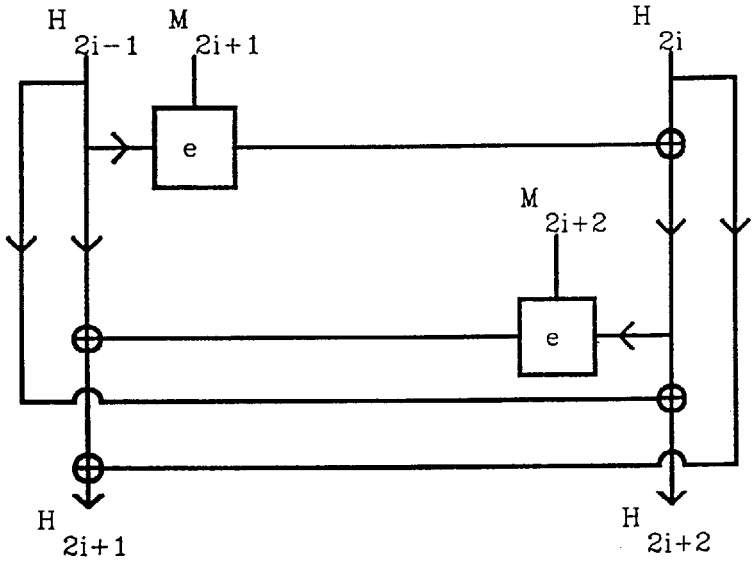
[MS] C.H. Meyer and M. Schilling, "Secure program load with Manipulation Detection Code", Securicom 88, pp. 111-130.

[QD] J.J. Quisquater and J.P.Delescaille, "How easy is collision search? Application to DES", these proceedings.

[QD] J.J. Quisquater and J.P.Delescaille, "How easy is collision search? New results and applications to DES", Proc. of Crypto '89, to appear.

[Yu] G. Yuval, "How to swindle Rabin", Cryptologia, Vol. 3, Jul. 1979, pp. 187-189.

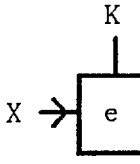
Figure 1



H_{-1}, H_0 : initializing values

M_1, M_2, ... : message blocks

e : block-cipher algorithm



Y means : Y = eK (X)