Analysis of a hash-function of Yi and Lam*

Keith Martin
Katholieke Universiteit Leuven,
ESAT-COSIC,
Kardinaal Mercierlaan 94, B-3001 Heverlee, Belgium
keith.martin@esat.kuleuven.ac.be

Chris J. Mitchell
Information Security Group,
Royal Holloway, University of London,
Egham, Surrey TW20 0EX, UK
C.Mitchell@rhbnc.ac.uk

16th September 1998

Abstract

A block cipher based hash-function of Yi and Lam [5] is analysed and shown to be significantly weaker than originally intended.

1 Introduction

Yi and Lam [5] give a method for deriving a $2m$-bit hash-function from a block cipher with an m-bit block length and a $2m$-bit key length. We show that the hash-function is somewhat less secure than claimed in [5]; indeed, it appears to offer no significant gains over the 'single length' block cipher based hash-function in ISO/IEC 10118-2 [1].

2 The Yi-Lam hash-function

The hash-function is based on the iterated use of a round-function, which is, in turn, block cipher based. Data to be hashed is split into m-bit blocks, with padding added, as necessary, to the final block. An extra final block is added, containing an encoding of the data's bit-length prior to padding. We denote the resulting string of blocks by: $M_1, M_2, \ldots, M_n$, where M_n contains the encoded length value.

Denote block cipher encryption by $E_K(M)$, where M is an m-bit block and K is a $2m$-bit key (we also use D to denote decryption). The hash-function is computed by recursively computing the following values, for i successively equal to $1, 2, \ldots, n$.

$$H_i = E_{K_i}(M_i) \oplus M_i,$$

*This work was supported by the European Commission under ACTS project AC095 (ASPeCT).

$$G_i = (E_{K_i}(M_i) \oplus G_{i-1})[+]H_{i-1}, \quad (1)$$

where:

- G_0 and H_0 are 'specified initial values'[1],
- $\oplus$ denotes bit-wise exclusive-or of blocks,
- $[+]$ and $[-]$ denote addition and subtraction modulo 2^m, where m-bit blocks are treated as binary representations of numbers in the range $[0, 2^m - 1]$,
- K_i is the $2m$-bit key obtained by concatenating G_{i-1} and H_{i-1} ($1 \leq i \leq n$), and
- the $2m$-bit hash-code is the concatenation of G_n and H_n.

Unfortunately the fact that the triple G_{i-1}, G_i and H_i can be used to compute M_i in (1) means that this hash-function is susceptible to three *solving one-half attacks* [2]. For completeness we describe in detail how to implement the general attacks described in [2]. We assume throughout that the block cipher behaves as a random function; if it does not, then other attacks are likely to be possible.

3 Finding a collision

Suppose an attacker wishes to find two different n-block data strings yielding the same hash-code. The attacker chooses an arbitrary m-bit value G_{n-1} and arbitrary data blocks $M_1, M_2, \ldots, M_{n-3}$. The attacker then computes the pair of values (G_{n-3}, H_{n-3}). The attacker now performs the following steps $2^{m/2}$ times.

1. Choose a data block M_{n-2}.
2. Compute (G_{n-2}, H_{n-2}) . Let K_{n-1} be the $2m$-bit cipher key obtained by concatenating (G_{n-2}, H_{n-2}).
3. Compute $M_{n-1} = D_{K_{n-1}}((G_{n-1}[-]H_{n-2}) \oplus G_{n-2})$.
4. Compute $H_{n-1} = E_{K_{n-1}}(M_{n-1}) \oplus M_{n-1}$.

Each pair of data blocks (M_{n-2}, M_{n-1}) and the corresponding H_{n-1} are stored. At the end of this process the attacker checks all the m-bit values H_{n-1} (there will be $2^{m/2}$ of them) to see if any pair are equal. By the "birthday problem" there is a high probability that such a pair will exist. If the matching values of H_{n-1} correspond to the message pairs (M_{n-2}, M_{n-1}) and (M'_{n-2}, M'_{n-1}) then it is simple to verify that the sequences $(M_1, \ldots, M_{n-3}, M_{n-2}, M_{n-1})$ and $(M_1, \ldots, M_{n-3}, M'_{n-2}, M'_{n-1})$ both hash to (G_{n-1}, H_{n-1}). To complete the attack we append the additional block M_n to each sequence, where M_n is a valid encoding for a message containing $(n-1)m$ bits. We then have two data strings with the same hash-code.

Each iteration of the above steps involves 3 encryptions and decryptions. Hence the attack complexity is $3.2^{m/2}$, substantially less than the brute force value of around 2^m.

[1] Note that it is not clear whether Yi and Lam intend these values to be fixed for all applications of the hash-function, although, since this is generally the most secure option, we assume that they are fixed, at least within a particular domain of use.

Note that, to get two messages of (different) pre-determined meanings with the same hash, then we perform two sets of $2^{m/2}$ iterations of the above steps, the first (second) set being performed with $2^{m/2}$ variants of the first (second) message. A match between the first and second sets will give the desired 'collision'.

4 Finding a second pre-image

Note that this attack was referred to as a "target attack" in [5]. Suppose an attacker has a data string $M_1, M_2, \ldots, M_n$ and the corresponding hash-code (G_n, H_n). We show how the attacker can find another data string (of the same length) with the same hash-code.

The attacker first computes the pair (G_{n-1}, H_{n-1}), by hashing all but the last block of the data string. The attacker then chooses data blocks $M_1^*, M_2^*, \ldots, M_{n-3}^*$ and computes the pair of values (G_{n-3}^*, H_{n-3}^*). The attacker now performs the following steps as many times as necessary.

1. Choose a data block M_{n-2}^*.
2. Compute the pair (G_{n-2}^*, H_{n-2}^*). Let K_{n-1}^* be the $2m$-bit cipher key obtained by concatenating (G_{n-2}^*, H_{n-2}^*).
3. Compute $M_{n-1}^* = D_{K_{n-1}^*}((G_{n-1}[-]H_{n-2}^*) \oplus G_{n-2}^*)$.
4. Compute $H_{n-1}^* = E_{K_{n-1}^*}(M_{n-1}^*) \oplus M_{n-1}^*$.
5. If $H_{n-1} = H_{n-1}^*$ then it is simple to verify that $(M_1^*, M_2^*, \ldots, M_{n-1}^*, M_n)$ has hash-code (G_n, H_n), i.e. we have a second pre-image for the specified hash-code. It is important to note that M_n is the same as the value for the original message, since this encodes the message length.

The probability of success in each iteration of the above steps is 2^{-m}, and hence the expected number of times they must be performed to find a (second) pre-image is 2^{m-1}. Each iteration involves 3 encryptions or decryptions, and hence the expected attack complexity is 3.2^{m-1}, significantly less than the 2^{2m} required for a brute force attack.

5 Finding a pre-image

Conducting a pre-image attack is only marginally more difficult than a second pre-image attack. We note that the full details of such an attack were not provided in [2]. In this case the attacker has a hash-code (G_n, H_n), but does not know the corresponding data string. We show how to find a data string giving this hash-code.

The attacker starts by choosing a value M_n, which encodes a valid length for an $(n-1)$-block data string (e.g. the value $m(n-1)$). The attacker now performs the following steps as many times as necessary.

1. Choose an m-bit block H_{n-1}^{**}.
2. Find the unique value G_{n-1}^{**} which satisfies
$$H_n \oplus M_n \oplus G_{n-1}^{**} = G_n[-]H_{n-1}^{**}.$$
Let K_n^* be the $2m$-bit cipher key obtained by concatenating $(G_{n-1}^{**}, H_{n-1}^{**})$.

3. Check whether or not $E_{K_n^*}(M_n) = H_n \oplus M_n$. If so, then exit this iterative process and save $(G_{n-1}^{**}, H_{n-1}^{**})$.

Note that it is not guaranteed that the above steps will succeed in finding a pair $(G_{n-1}^{**}, H_{n-1}^{**})$, since such a pair will not always exist; however, the probability of success is greater than 0.5. Moreover, if the attacker happens, by accident or design, to choose the same value of M_n as was used to originally generate the hash-code, then the existance of at least one pair is guaranteed. The attacker now proceeds as for the second pre-image attack, except with (G_{n-1}, H_{n-1}) replaced by $(G_{n-1}^{**}, H_{n-1}^{**})$.

The success probability for both the search for $(G_{n-1}^{**}, H_{n-1}^{**})$ and the pre-image search is 2^{-m}, and so the expected number of times they must be performed is 2^{m-1}. Each iteration of the first and second sets of steps respectively involves 1 and 3 encryptions or decryptions. The expected attack complexity is thus 2^{m+1}, again significantly less than the complexity of a brute force attack.

6 Conclusions

It has been shown that contrary to claims in [5] the hash-function of Yi and Lam is not significantly more secure than an m-bit hash function of the type described in ISO/IEC 10118-2 [1]. This is due to fatal design flaw that leaves the hash-function susceptible to the "solving one-half attacks" described in [2]. For recent work on how best to design a hash-function using a block cipher see [3, 4].

The authors would like to thank Bart Preneel and Vincent Rijmen for useful discussions.

References

[1] International Organization for Standardization, Genève, Switzerland. *ISO/IEC 10118-2, Information technology—Security techniques—Hash-functions; Part 2; Hash-functions using an n-bit block cipher algorithm*, 1994.

[2] L.R. Knudsen, X. Lai and B. Preneel. Attacks on fast double block length hash functions. *Journal of Cryptology*, **11**:59–72, 1998.

[3] L.R. Knudsen and B. Preneel. Hash functions based on block ciphers and quaternary codes. In K. Kim and T. Matsumoto, eds., *Advances in Cryptology, Proc. Asiacrypt '96*, Lecture Notes in Computer Science **1163**, pp. 77–90. Springer-Verlag, Berlin, 1996.

[4] L.R. Knudsen and B. Preneel. Fast and secure hashing based on codes. In B. Kaliski, ed., *Advances in Cryptology, Proc. Crypto '97*, Lecture Notes in Computer Science **1294**, pp. 485–498. Springer-Verlag, Berlin, 1997.

[5] X. Yi and K.Y. Lam. Hash function based on block cipher. *Electronics Letters*, **33**:1938–1940, 1997.