The breaking of the AR Hash Function

Ivan B. Damgård and Lars R. Knudsen

Aarhus University, Denmark

Abstract. The AR hash function has been proposed by *Algorithmic Research Ltd* and is currently being used in practice in the German banking world. AR hash is based on DES and a variant of the CBC mode. It produces a 128 bit hash value.
In this paper, we present two attacks on AR hash. The first one constructs in one DES encryption two messages with the same hash value. The second one finds, given an arbitrary message M, an $M' \neq M$ with the same hash value as M. The attack is split into two parts, the first part needs about 2^{33} DES encryptions and succeeds with probability 63%, the second part needs at most about 2^{66} DES encryptions and succeeds with probability about 99% of the possible choices of keys in AR. Moreover, the 2^{33} respectively 2^{66} encryptions are necessary only in a one-time preprocessing phase, i.e. having done one of the attacks once with success, a new message can be attacked at the cost of no encryptions at all. Since the hash value is 128 bits long, the times for the attacks should be compared to 2^{64}, resp. 2^{128} DES encryptions for brute force attacks.
For the particular keys chosen in AR hash we implemented the first part of the second attack. In 2^{33} encryptions we found two messages that breaks AR hash.

1 The AR Hash Function

The AR hash function has been proposed by Algorithmic Research Ltd., it has been distributed in the ISO community [1] for informational purposes, but has not been considered a standard. It is currently in use in the German banking world.

In the following, $DES_k(y)$ will denote the DES-encryption of block y using key k.

The basic structure in AR-hash can be described as a variant of DES in CBC-mode, where the last 2 ciphertext blocks are added to the current input, and where the state consists of the last two "ciphertext" blocks computed. To do the entire function, the message is processed with two keys, yielding a result of 2 times 128 bits. This is then further compressed to get a result of 128 bits.

To define AR more precisely, we first divide the message m to be hashed into 8-byte blocks, denoted by $m_1, m_2, ..., m_n$ (0-padding is used on the last block if it is incomplete).

We then define a series of 64-bit blocks $o_{-1}, o_0, o_1, ..$by

$$o_{-1} = o_0 = 0$$

and

$$o_i = m_i \oplus DES_k(m_i \oplus o_{i-1} \oplus o_{i-2} \oplus \eta),$$

where k is an arbitrary DES key, and the constant η is defined by

$$\eta = 01\ 23\ 45\ 67\ 89\ AB\ CD\ EF$$

in hexadecimal notation. We now let $f_1(m,k), f_2(m,k)$ denote o_{n-1}, o_n, respectively.

In the actual hash function AR/DFP, two different keys k_1 and k_2 are used, specified as

$$k_1 = 00\ 00\ 00\ 00\ 00\ 00\ 00\ 00, \quad k_2 = 2A\ 41\ 52\ 2F\ 44\ 46\ 50\ 2A$$

One then first computes

$$c_1 = f_1(m,k_1),\ c_2 = f_2(m,k_1),\ c_3 = f_1(m,k_2),\ c_4 = f_2(m,k_2)$$

and the hash value is now the concatenation of the two 8 byte blocks

$$G(G(c_1,c_2,k_1),G(c_3,c_4,k_1),k_1) \text{ and } G(G(c_1,c_2,k_2),G(c_3,c_4,k_2),k_2),$$

where G is the function defined by

$$G(x,y,k) = DES_k(x \oplus y) \oplus DES_k(x) \oplus DES_k(y) \oplus y.$$

For convenience in the following, we will let $DFP(c_1,c_2,c_3,c_4,k)$ denote the final hash result.

2 Properties of f_1, f_2 and G

In the following, let A and B be messages of length a multiple of 8 bytes, and let $A|B$ be the concatenation of A and B. Choose a fixed, but arbitrary DES key k, and let $y = f_1(A,k)$, $z = f_2(A,k)$. Let m be an arbitrary 8-byte block. Let $C(A,m)$ be the three-block message

$$m \oplus \eta \oplus y \oplus z|\ DES_k(m) \oplus y|\ DES_k(m) \oplus z$$

Let $D(A,m)$ be the three-block message

$$m \oplus \eta \oplus y \oplus z|\ m \oplus y|\ m \oplus z$$

Let $E(A,m)$ be the three-block message

$$m \oplus \eta \oplus y \oplus z|\ m \oplus y|\ DES_k^2(m) \oplus z$$

Then we have the following result, showing that it is very easy to find collisions for the functions f_1, f_2:

Lemma 1 *For arbitrary A, B, k, m as above, we have that*

$$f_i(A|\ B, k) = f_i(A|\ C(A, m)|\ B, k), i = 1, 2$$

$$f_2(A, k) = f_2(A|\ E(A, m), k)$$

If k is a weak DES key, then we also have

$$f_i(A|\ B, k) = f_i(A|\ D(A, m)|\ B, k), i = 1, 2$$

Proof: By combining the definition of $C(A, m)$ and f_1, f_2 and by letting f_0 be the hash value produced just before f_1 we obtain

$$\begin{aligned}
f_0(A|\ C(A, m), k) &= m \oplus \eta \oplus y \oplus z \oplus DES_k(m \oplus \eta \oplus y \oplus z \oplus y \oplus z \oplus \eta) \\
f_1(A|\ C(A, m), k) &= DES_k(m) \oplus y \oplus DES_k(DES_k(m) \oplus y \oplus m \oplus \\
&\qquad \eta \oplus y \oplus z \oplus DES_k(m) \oplus z \oplus \eta) \\
&= y \\
f_2(A|\ C(A, m), k) &= DES_k(m) \oplus z \oplus DES_k(DES_k(m) \oplus z \oplus y \oplus \\
&\qquad m \oplus \eta \oplus y \oplus z \oplus DES_k(m) \oplus \eta) \\
&= z
\end{aligned}$$

This proves the first statement. The second and third are proved similarly, using for the third that if k is a weak key, then by definition we have that $DES_k(DES_k(m)) = m$ for all m. □

By inspection of the definition of G, it is trivial to show the following lemma:

Lemma 2 *The functions G, DFP have the following properties for arbitrary c_1, c_2, k:*

$$G(c_1, c_2, k) = G(c_1 \oplus c_2, c_2, k)$$

$$G(c_1, 0, k) = DES_k(0)$$

$$DFP(c_1, c_1, c_1, c_1, k) = (c_1, c_1), \quad DFP(c_1, 0, c_2, 0) = 0$$

Thus, it is also very easy to find collisions for G and DFP.

Although none of these properties imply directly a collision for the hash function itself, they will be useful in the following.

3 Attacks on AR Hash

3.1 Collision attack

A collision attack finds two messages m and m' that hash to the same value. This first attack on AR hash exploits the fact that for a weak key k it is easy to find fixpoints for DES, i.e. to find m s.t. $DES_k(m) = m$. There are exactly 2^{32} such fixpoints for a weak key [2] and each fixpoint can be found in half a DES encryption. Since all round keys for a weak key are equal, a necessary and

sufficient condition for a fixpoint is that the halves of the encrypted value after 8 rounds of encryption are equal.

If A is the empty message in Lemma 1, then $y = z = 0$. Let $X(m)$ be the 3-block message $m \oplus \eta | \ DES_{k_2}(m) | \ DES_{k_2}(m)$. This means that by Lemma 1

$$f_1(X(m), k_2) = 0$$
$$f_2(X(m), k_2) = 0$$

for any m. Let m be a fixpoint for k_1, then

$$f_1(X(m), k_1) = DES_{k_2}(m) \oplus DES_{k_1}(DES_{k_2}(m))$$
$$f_2(X(m), k_1) = DES_{k_2}(m) \oplus DES_{k_1}(DES_{k_1}(DES_{k_2}(m))) = 0$$

since k_1 is a weak key. The above four values are also the c_i values produced by hashing $X(m)$. But by Lemma 2, a G-value is invariant in the first argument if the second is 0, so it is clear that for fixpoints (for k_1) $m \neq m'$, $X(m)$ and $X(m')$ will be hashed to the same value. Finding two fixpoints for k_1 takes in time one DES encryption, which leads to:

Theorem 1 *There exists an algorithm, which finds in time one DES encryption, two different messages with the same AR hash value.*

The above attack can be extended to attacks that in time $n/2$ encryptions find n messages that hash to the same value, where $n \leq 2^{32}$. By contrast, a brute force attack that finds two messages that hash to the same value would require computation of about 2^{64} hash values.

3.2 Preimage attack

A preimage attack takes a given message M as input and tries to find a new message with the same hash value.

AR hash uses two fixed keys. In the following we consider arbitrary keys, where one key, k_1, is a weak key[1].

The basic idea in this second attack on AR hash is to try to find a message which takes the initial state back to itself, i.e. leads to a set of all-zero c-values. If Z is such a message, then clearly $AR(M) = AR(Z|M) = AR(Z|Z|M) = \cdots$. It is also clear that once we have found such a Z, any message M can be attacked at no further cost.

In more detail, we try, inspired by Lemma 1, with Z of the form $Z = m_1 \oplus \eta | m_2 | m_2$. It is now easy to write down the equations that m_1, m_2 must satisfy in order for $f_1(Z, k_i) = f_2(Z, k_i) = 0, i = 1, 2$. We get the following:

$$DES_{k_1}(m_1) \oplus m_1 = DES^{-1}_{k_1}(m_2) \oplus m_2 \tag{1}$$

$$DES_{k_2}(m_1) \oplus m_1 = DES^{-1}_{k_2}(m_2) \oplus m_2 \tag{2}$$

[1] The DES has 4 weak keys.

It is difficult in general to say anything about the number of solutions to these equations, or how hard it is to find them. There is a special case, however, that is easier:

Let m_1 be a fixpoint for k_1. Put $m_2 = DES_{k_2}(m_1)$. Then (2) is always satisfied and (1) is true if

$$DES_{k_1}(DES_{k_2}(m_1)) = DES_{k_2}(m_1) \tag{3}$$

which is true if also $DES_{k_2}(m_1)$ is a fixpoint for k_1. It is reasonable to assume that the mapping $DES_{k_2}(\cdot)$ distributes fixpoints for k_1 uniformly. Therefore the probability that $DES_{k_2}(m_1)$ is a fixpoint for k_1 is 2^{-32}. By running through all fixpoints for k_1 the probability that (3) is satisfied is

$$1 - (1 - 2^{-32})^{2^{32}} \simeq 1 - e^{-1} \simeq 0.63$$

Since checking whether a message is a fixpoint for a weak key takes half a DES encryption, the attack needs a total of $2 \times 2^{32} = 2^{33}$ DES encryptions. A similar attack appeared in [3].

To confirm the validity of the 0.63 probability, we did a computer simulation on a "scaled-down" version of DES, working with 32-bit blocks, thus making it easy to run through all fixpoints. The experiments confirmed the theory. The test ran through all 2^{16} fixpoints for 100 pairs of keys, where one key was a weak key in a 32 bit block version of DES. Out of 100 key pairs, the equation (3) had a solution for 62 pairs.

The above attack is quite feasible, and can be executed in at most a few days, even hours, using up to date hardware. Later in this section we give the results of an implementation of the attack on AR hash with the two keys given in [1].

The above probability can be improved to almost 1 on the cost of a squared complexity. In this case we proceed as follows (where m_1 is not necessarily a fixpoint for k_1):

If we put $m_2 = DES_{k_1}(m_1)$, then equation (1) is trivially satisfied, and (2) is satisfied as well, if

$$DES_{k_1}(m_1) = DES_{k_2}(m_1) \tag{4}$$

or

$$DES_{k_2}(m_1) \oplus m_1 = DES_{k_1}(m_1) \oplus DES_{k_2}^{-1}(DES_{k_1}(m_1)) \tag{5}$$

Symmetrically, we can put $m_2 = DES_{k_2}(m_1)$. This means that (2) is now always satisfied, and that (1) is true if either $DES_{k_1}(m_1) = DES_{k_2}(m_1)$ (same condition as (4)) or if

$$DES_{k_1}(m_1) \oplus m_1 = DES_{k_2}(m_1) \oplus DES_{k_1}^{-1}(DES_{k_2}(m_1)) \tag{6}$$

Finally, since k_1 is a weak key, there is another possibility, namely to put $m_1 = m_2$. Once again, this trivially satisfies (1), and (2) is in this case satisfied, if

$$DES_{k_2}^2(m_1) = m_1 \tag{7}$$

To summarize, if we can find a 64-bit block m_1 that satisfies (4), (5), (6) or (7) then we have a 3-block sequence Z that makes the attack successful. Checking

if a block satisfies any of the equations requires at most 5 encryptions, so going through all possibilities for m_1 will require about $5 \cdot 2^{64} \simeq 2^{66}$ encryptions.

The remaining question is of course if there are any solutions to the equations at all. Simply doing the 2^{66} encryptions is not feasible today (although it probably will become feasible in the not too distant future). Therefore the best we can do is to see if we can estimate the probability that solutions exist, assuming that the two keys k_1, k_2 are randomly chosen, but where k_1 is a weak key.

Each of the 4 equations can be written in the form $h(m_1) = 0$, where h is some function that depends on the keys, and is built from a number of DES en- and decryptions. It is a generally accepted assumption that DES in a context like this one behaves like a random function. This means that the 3 equations (4),(5) and (7) each have solutions with an independent probability of

$$1 - (1 - 2^{-64})^{2^{64}} \simeq 1 - e^{-1} \simeq 0.63$$

However since (6) contains (3) as a special case this probability splits into two depending on whether fixpoints are examined or not, the probability that (6) has a solution therefore is

$$1 - ((1 - 2^{-64})^{2^{64}-2^{32}} \times (1 - 2^{-32})^{2^{32}}) \simeq 1 - e^{-2}$$

Thus we expect that the probability over the choice of k_1, k_2 with k_1 weak that solutions do exist is about $1 - e^{-5} \simeq 0.99$.
In summary we have the following:

Theorem 2 *There exists two attacks on AR hash that constructs from a given message M a new one $M' \neq M$ such that $AR(M) = AR(M')$. The attacks takes time at most about 2^{33} and about 2^{66} DES encryptions, respectively. Under reasonable heuristic assumptions, the attacks can be shown to be successful for respectively about 63% and 99% of the possible choices of keys in AR hash. Both attacks can be done in a preprocessing phase, after which each message can be attacked at no further cost.*

These attacks are much faster than a brute-force attack, which would require computation of about 2^{128} hash values.

For the keys chosen in AR hash we did an exhaustive search through all fixpoints for the weak key, $k_1 = 0$. We obtained

Theorem 3 *For AR hash there exists two 3-block messages Z_1 and Z_2, s.t. any message M can be prefixed with either Z_1 or Z_2 (or both) any number of times, yielding unchanged AR hash value, where*

$$\begin{aligned} Z_1 &= \text{7a6199a238bb8643} \mid \text{8073d91a57ca1e2a} \mid \text{8073d91a57ca1e2a} \\ Z_2 &= \text{02bb2604aafcbecf} \mid \text{6421e999f02ddfd6} \mid \text{6421e999f02ddfd6} \end{aligned}$$

4 Conclusion

The weaknesses we have found in AR hash clearly make it very problematic to continue using the hash function as it is. The collision and preimage attacks can be thwarted by adding the message length to the message, however because of Theorem 3 collisions still can be obtained in constant time, because $Z_1|M$ and $Z_2|M$ would hash to the same value.

So the question arises whether one can repair the function so that our attacks are prevented.

We have of course exploited the fact, that there are 2^{32} fixpoints for a weak DES key and that they are easy to find. However, avoiding weak keys still would enable a preimage attack, since equations (4), (5) and (6) can be set up independently of the nature of the keys. The probability for success for this attack is expected to be $1 - e^{-3} \simeq 95\%$.

To confirm this we did another computer simulation on a "scaled-down" version of DES. The test used 16 bit blocks and ran through all 2^{16} possible messages for 100 pairs of random keys. Out of 100 key pairs, for only 3 key pairs none of the equations (4), (5) and (6) had solutions thus confirming the theory.

Furthermore we made essential use of the fact that the initial state is all-zero, in particular that it consists of 4 blocks that are equal. Trying to prevent attacks only by changing the initial values is extremely dangerous and it is shown in [3] how to find collisions even in this case.

Section 2 shows a number of problematic properties of f_1, f_2 and G that are independent on the initial state and on the chosen keys, we therefore believe that the basic design of f_1, f_2 and G should be reconsidered. One can perhaps guess that AR hash (or rather the f_1, f_2 functions) was designed starting from the standard MAC-mode for DES (which uses a secret key), obtaining a hash function by using a known, fixed key, and adding some extra elements (the feed-forward, etc.) to compensate for the weaknesses implied by the fact that the key is now known.

Our attacks can be seen as an illustration that constructing a hash function in this way from a MAC is not easy, and that it is perhaps a better strategy to build a hash function mode "from scratch".

5 Acknowledgements

We would like to thank Dr. Bart Preneel for helpful discussions and comments.

References

1. *AR fingerprint function.* ISO-IEC/JTC1/SC27/WG2 N179, working document, 1992.
2. D. Coppersmith. *The real reason for Rivest's phenomenon.* Proceedings of Crypto 85, Springer Verlag LNCS series.

3. B. Preneel. *Analysis and Design of Cryptgraphic Hash Functions.* Ph.D. Thesis, Katholieke Universiteit Leuven, January 1993.

This article was processed using the LaTeX macro package with LLNCS style