Date: Wed, 14 Apr 1999 17:23:41 +0300 (EET DST)
From: Markku-Juhani Saarinen <mjos@cc.jyu.fi>
To: AESFirstRound@nist.gov
Subject: AES First Round Comment.

Please find attached the file "sshnote.pdf" which is intended as an AES first round comment. Other file formats are available by request (PostScript, LaTeX).

- mj

Submitter:

Markku-Juhani Saarinen <mjos@ssh.fi>
SSH Communications Security Ltd.
Tekniikantie 12, FIN-02150 Espoo, Finland

Title:

A Note Regarding the Hash Function Use of MARS and RC6

Abstract:

When a block cipher is used in Davies-Meyer mode, the speed of the resulting hash function is directly related to the key size used. Therefore, it is often desirable to use keys that are as long as possible.

In this paper, we investigate the security of MARS and RC6 in the Davies-Meyer hash mode when long keys are used. We give an algorithm that finds equivalent keys for MARS with 2^12 effort. We also give an algorithm that finds "almost" equivalent keys for RC6 with 2^17 effort. As a result of these algorithms, the Davies-Meyer hash functions built from MARS and RC6 with certain key sizes can be considered insecure.

The security of MARS and RC6 with 128, 192 and 256 bit key sizes is not affected by the results presented in this paper. We propose that the key size range of these ciphers should be limited to match the actual security offered by them.

A Note Regarding the Hash Function Use of MARS and RC6

Markku-Juhani O. Saarinen

SSH Communications Security Ltd.
Tekniikantie 12, FIN-02150 Espoo, Finland
mjos@ssh.fi

Abstract. When a block cipher is used in Davies-Meyer mode, the speed of the resulting hash function is directly related to the key size used. Therefore, it is often desirable to use keys that are as long as possible. In this paper, we investigate the security of MARS and RC6 in the Davies-Meyer hash mode when long keys are used. We give an algorithm that finds equivalent keys for MARS with 2^{12} effort. We also give an algorithm that finds "almost" equivalent keys for RC6 with 2^{17} effort. As a result of these algorithms, the Davies-Meyer hash functions built from MARS and RC6 with certain key sizes can be considered insecure. The security of MARS and RC6 with 128, 192 and 256 bit key sizes is not affected by the results presented in this paper. We propose that the key size range of these ciphers should be limited to match the actual security offered by them.

1 Introduction

The most typical method for turning a block cipher E_k into a hash function is the Davies-Meyer hash [10]. In the Davies-Meyer hash, the message M is padded and split into pieces M_i of the same length as the key k.

$$H_i = E_{M_i}(H_{i-1}) \oplus H_{i-1}$$

The hash of M is the final value of H_i. The XOR operation is sometimes replaced with another arithmetic operation, such as addition modulo 2^n.

Most of the dedicated hash functions, such as MD4 [7], MD5 [8], and SHA-1 [9] also resemble this construction. The compression function of these hash functions is bijective and can be efficiently computed in both directions. They can be viewed as 128 / 160 - bit block ciphers with a 512-bit key.

It is easy to see that the speed of a Davies-Meyer hash is almost directly related to the key size of the block cipher. Therefore it is interesting to note that of the fifteen AES candidates, four allow key sizes that are larger than 256 bits:

- MARS [1] allows key sizes up to 1248 bits. The key length does not significantly affect the speed of key setup.
- RC6 [3] allows key sizes up to 2040 bits. If the key is longer than 1408 bits, additional mixing steps are required.

– HPC [4] allows arbitrary key sizes. An efficient algorithm for finding equivalent keys for HPC (any key size) has been found [5] [6].
– FROG [11] allows key sizes up to 1000 bits. FROG has been cryptanalyzed and will not be considered in this paper [12].

In this paper, we will give initial findings on the security of MARS and RC6 when they are used in Davies-Mayer hashing mode. For both of these ciphers, key length does not affect the speed of encryption.

2 Equivalent keys in MARS

MARS allows keys sizes up to 1248 bits. The specification states that long key sizes are allowed for convenience. As an example, the authors of MARS mention that the entire result from a Diffie-Hellman key exchange could be used as a key.

An equivalent key pair (k_1, k_2) has a property that any plaintext block encrypted under these distinct keys result in equal ciphertexts: $E_{k_1}(P) = E_{k_2}(P)$. If an equal key pair is found for the encryption (or compression) function E, collisions for the Davies-Meyer hash can be produced.

Since MARS has 40 subkeys (1280 bits of internal keying material), and a non-surjective key schedule, an easy probabilistic argument can be used to show that equivalent keys do exist. In the following, we will give a simple and computationally efficient algorithm for finding such a key pair.

2.1 Attacking the MARS key schedule

The key expansion of a n-word key works roughly as follows:

1. Fill the array $t[0..38]$ with key words.
2. Perform a linear transformation on $t[0..38]$.
3. Set $t[39] = n$.
4. Stir the array $t[0..39]$ using a "type-1" Feistel network.
5. Shuffle the words in $t[0..39]$ and store the result in $k[0..39]$.
6. Fix the keys in k that are used for multiplication, if necessary.

$k[0..39]$ is the expanded key.
(The numbering of steps differs from that in [1])

Steps 2, 4, and 5 are easily reversible, but steps 1, 3, and 6 are not.

In step 6, sixteen multiplication keys are "fixed" to guarantee that the keys are not weak; both of the lowest 2 bits should be 1 and the words should not contain ten consecutive 0's or 1's.

Since approximately 4.09 words get "fixed" into the same word, $4.09^{16} = 2^{32.5}$ $k[\,]$ arrays produce the same expanded keys in step 6. For these $k[\,]$ arrays, K_2 may be found by reversing steps 5 and 4 and checking if $t[39] = 39$. If this is the case, one can reverse step 2 and obtain the 39-word key from $t[0 \ldots 38]$.

If $t[39]$ is modeled as random, the probability for $t[39] = 39$ is 2^{-32}. Since there are $2^{32.5}$ pre-images that one can try, this gives a 0.75 probability for finding

at least one such equivalent key. The effort needed to find K_2 is approximately 2^{32}.

Therefore, a collision can be produced for arbitrary messages with 2^{32} effort. Note that this method can do more than just find collisions in hash functions; it can be used to find an equivalent 39-word key for any given key with a nonneglible probability.

Although the design document states that MARS probably does not have equivalent keys, it also anticipates the possibility. We quote [1], page 55:

> Also, in all likelihood MARS does not have any equivalent keys: it is highly unlikely that any two different 40-word keys have the same behavior, and the key expansion process is "random enough" so that it is highly unlikely that any two different keys yield the same expanded key array. (...) The only operation which may result in collisions is the "key fixing", where we ignore the lowest two bits in some of the key words. (...) Therefore, as long as the original key is less than about 600 bits, it is highly unlikely that any pair of keys result in the same expanded array.

2.2 Optimizing the equivalent key search

Perhaps surprisingly, equivalent keys can be found with only 2^{12} effort when key size is 5 words (160 bits), or if the key repeats itself in a 5-word cycle.

Feedback in the the linear transformation partially cancels out when key size is 5 words, since it is defined as ($K[\,]$ is the input key)

$$\text{for } i = -7 \ldots -1,\ T[i] = S[i+7]$$
$$\text{for } i = 0 \ldots 38,\ T[i] = ((T[i-7] \oplus T[i-2]) \lll 3) \oplus K[i \bmod n] \oplus i$$

When reversing expanded and fixed random 5-word keys, the value of $t[39]$ in step 3 was found to be strongly biased towards certain small numbers. After 10^8 iterations, the following experimental probabilities were found.

n	$P(t[39] = n)$
5	0.00353
7	0.00315
9	0.00277
11	0.00246
39	0.00021
41	0.00038
43	0.00054

$P(t[39] = n)$ for almost all other n was found to be close to the expected value $2.33 * 10^{-10}$. The value $P(t[39] = 39) = 0.00021$ gives a complexity of 2^{12} for this attack.

Example. We define K_1 and K_2 as

```
k1[5] =
  00000b51 00000000 00000000 00000000 00000000

k2[39] =
  d27d2295 ecdcfd37 108a2302 7296a0ed 54a262b4 592f4b68 ba74630a
  7a71a855 c66043a7 1e5f38ab 2ad4ff38 5b70ca03 7fc725e4 6db4aff1
  bebabe1f 65136b58 084dc0df 2a17c855 cf10e275 567f3823 4fd87ab4
  21d5c132 945b1198 86131b5f 9e066049 cb4b21eb 131238ca 90e7c908
  1cb74b19 3cf36938 22c0d4f7 49ade389 4b2cb166 acbeeba7 ec8dacec
  5476eb60 fb48d976 33b9dbc4 64c6c5c2
```

These keys produce the same key schedule and are equivalent. Only 2897 trials (a fraction of a second) were needed before this pair was found. Note that these 32-bit words are stored into the computer memory in little endian manner.

3 Almost equivalent keys for RC6

What makes the hash function use of RC6 particularly interesting is the fact that RC6 is a parameterized cipher; the block size can be grown in a straightforward manner to 256 bits and beyond. If a balanced security level of 2^{128} is desired in a cryptosystem, a 256-bit hash is required.

Also, RC6 has a maximum key size of 2040 bits, making the corresponding Davies-Meyer hash function very fast. It is clear that equivalent keys must exist, especially when the key size is larger than the total size of the subkeys (1408 bits).

The 20-round AES version of RC6 uses 44 words of keying material: 2 for each round, and 2 for pre- and post-whitening. In particular, the subkeys $S[42]$ and $S[43]$ are added modulo 2^{32} to the first and third words of the 128-bit block before outputting it as ciphertext.

If the same plaintext block is encrypted using keys that only differ in $S[42]$ and $S[43]$, the ciphertexts will have a constant difference. We call such keys almost equivalent, because they have the same underlying enciphering permutation. Altering the subkeys $S[40]$ and $S[41]$ also has a similar effect on the ciphertexts.

3.1 Attacking the RC6-32/20/1408 key schedule

The RC6 key schedule is practically identical to that of RC5 [2]. In the following, we will give a functionally equivalent, but simplified description of the RC6-32/20/1408 key schedule.

Let $L[44]$ be an array of words, containing the input key. S[44] is the subkey array. In this special case, the indexing of S[ ] and L[ ] is always synchronized:

for $i = 0$ to 43 do
  $S[i] = \texttt{9e3779b9} * i + \texttt{b7e15163}$
$A = B = 0$
for $round = 1$ to 3 do
  for $i = 0$ to 43 do
    $A = S[i] = (S[i] + A + B) \lll 3$
    $B = L[i] = (L[i] + A + B) \lll (A + B)$

The key schedule can be viewed as 3-round widened Feistel-like structure that encrypts the array $S[\,]$ with the key $L[\,]$, at the same time mixing the contents of $L[\,]$. The key schedule is not directly reversible, because the rotation amount $(A + B)$ is lost. The information flow between the rounds is in the word pair (A, B).

Approximately one pair in a set of 2^{64} keys will have the same values of A and B after passes 1 and 2 by the birthday paradox. This means that the propagation of differences stops at the round boundary and changing the key at position $L[i]$ only affects subkey words $S[(i + 1) \ldots 43]$. In fact, $L[42]$ and $L[43]$ can be chosen (using straightforward arithmetic) so that the difference in A and B is zero after the first round.

Thus, almost equivalent 1408-bit keys can be found in 2^{32} steps by running through different values of $L[40]$ and $L[41]$, making the appropriate changes to $L[42]$ and $L[43]$, and hoping for a birthday effect in round 2. The correct pair can be easily recognized, because the values of $S[0 \ldots 41]$ will match.

3.2 An optimized method for almost equivalent key search

One way to approach the problem of finding equivalent or almost equivalent keys is to look for a differential characteristic for $L[\,]$ that has the following properties:

1. Changes in A and B will cancel out before the round boundary, so that the start of the next round is unaffected.
2. The characteristic causes minimal change to $S[\,]$. Ideally, the characteristic would not change the final value of $S[\,]$ at all.
3. The probability of the characteristic is sufficiently high.

We have found the following high-probability characteristic:

Difference	Initial	Round 1	Round 2	Final
L[41] - L'[41]	1000000	1000000	8000000	
S[42] - S'[42]	0	8000000	0	Low weight
L[42] - L'[42]	f000000	8000000	0	
S[43] - S'[43]	0	0	0	Low weight
L[43] - L'[43]	8000000	0	0	
A - A' (end of round)		0	0	
B - B' (end of round)		0	0	

The experimental probability that the resulting subkeys have a Hamming distance of less than 32 is $2^{17.3}$. Only the post-whitening subkeys S[42] and S[43] are affected.

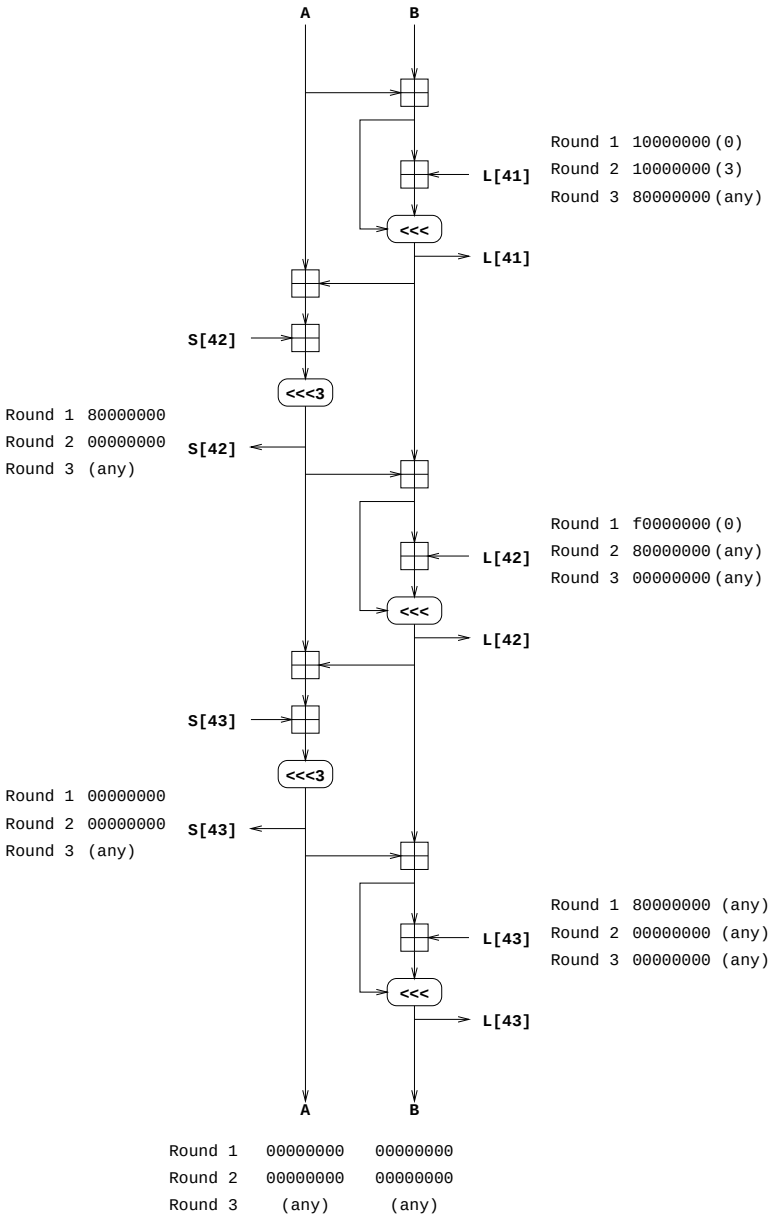

Fig. 1. Finding almost equivalent keys for RC6. Left side contains the difference in S[i] after the round. Right side contains the difference in L[i] before the round. The assumed cyclic rotation amount is given inside parenthesis. The resulting subkeys S[i] only differ in S[42] and S[43].

Example. We present two keys that, given the same plaintexts, produce ciphertexts that have an average Hamming distance of 4.2 bits. This key pair was found in less than a minute with a personal computer.

Key 1 and the subkey array:

```
k1[44] =
  001950c0 00000000 .. (all zeros) .. 00000000

s1[44] =
  15481c74 560ce35c d958037e 76e7da25 2d85b706 09b2961e 82eaccd6
  86fa8fc9 2023a8b6 2107d73e 6f872835 c4921f3c e26f39ea 3f0e875f
  14d27648 74485465 55b875c3 b5ae127a 94abf44f 8ab43df9 b550c57b
  a931e9c4 08759ce3 c89d7386 ccc36232 2b6a6be0 23d1cb2b f76b5a1d
  7a26a4ce 872b66a6 d12839b6 26dc801a 6ec68932 3e82d528 cfff8a10
  8880df7f b49906de 0c490f92 d870dcca 8d695743 d281965e c8622906
  c8bc33cc 0519fea3
```

Key 2 and the subkey array:

```
k2[44] =
  001950c0 .. (zeros).. 10000000 f0000000 80000000
                        ^        ^        ^

s2[44] =
  15481c74 560ce35c d958037e 76e7da25 2d85b706 09b2961e ..
  cabc33cc a519fea3
   ^        ^
```

4 Conclusions

It is commonly thought that a well-designed encryption algorithm should have a security level that is consistent with its key size. RC6 and MARS allow keys that are more than 1200 bits in length. To give a crude upper bound for the security of these ciphers, we note that a straightforward meet-in-the-middle attack can be mounted against both, RC6 and MARS. This attack has a complexity of approximately 2^{600}.

One can argue that a 2^{600} attack is completely impractical, but we have shown that there are real dangers in using the RC6 and MARS encryption algorithms in Davies-Meyer hash mode when the key size is very long. This illustrates the point that an excessive degree of flexibility in key size can actually hurt security in certain applications.

Therefore, we propose that the degree of flexibility in the key size should be limited to match the actual security of the ciphers.

We emphasize that have studied these ciphers, and we have not found any significant cryptographic weaknesses when the key size matches the AES requirements (128, 192 or 256 bits).

References

1. C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. M. Matyas Jr., L. O'Connor, M. Peyravian, D. Safford and N. Zunic, "MARS - a candidate cipher for AES," AES-submission, July 1998.
2. R. L. Rivest, "The RC5 encryption algorithm," Fast Software Encryption, LNCS 1008, Springer-Verlag, 1995, pp. 86–96.
3. R. L. Rivest, M. Robshaw, R. Sidney and Y.L. Yin, "The RC6^{TM} Block Cipher," AES-submission, June 1998.
4. R. Schroeppel, "Hasty Pudding Cipher Specification," AES-submission, June 1998.
5. D. Wagner, "Equivalent keys for HPC," Rump session talk given at the Second AES Conference, `http://www.cs.berkeley.edu/~daw/papers`, March 1999.
6. C. D'Halluin, G. Bijnens, B. Preneel and V. Rijmen, "Equivalent keys of HPC," Version 1.0, manuscript, `http://www.esat.kuleuven.ac.be/~rijmen/pub99.html`, April 1999.
7. R. L. Rivest, "The MD4 Message Digest Algorithm," Proceedings of CRYPTO '90, LNCS 537, Springer-Verlag, 1991, pp. 303–311.
8. R. L. Rivest, "The MD5 Message Digest Algorithm," Internet RFC 1321, `ftp://ftp.isi.edu/in-notes/rfc1321.txt`, 1992.
9. National Institute of Standards and Technology, "Secure Hash Standard," FIPS PUB 180-1, 1995.
10. B. Preneel, R. Govaerts and J. Vandewalle, "Hash functions based on block ciphers: a synthetic approach," Proceedings of CRYPTO '93, LNCS 773, Springer-Verlag, 1994, pp. 368–378.
11. D. G. Georguidis, D. Leroux and B. S. Chaves, "the FROG Encryption algorithm," AES submission, 1998.
12. D. Wagner, N. Ferguson and B. Schneier, "Cryptanalysis of FROG," to appear in the proceedings of the Second AES Candidate Conference, April 1999.