Differential Cryptanalysis of Nimbus*

Vladimir Furman

Computer Science Department, Technion - Israel Institute of Technology, Haifa 32000, Israel. vfurman@cs.technion.ac.il.

Abstract. Nimbus is a block cipher submitted as a candidate to the NESSIE project by Alexis Machado. Like many other ciphers Nimbus combines multiplication operations with XOR operations, a common technique to protect against various kinds of cryptanalysis. In this paper we present two new differential properties of multiplication operations with probability about 1/2 which we use to design a one-round iterative characteristic of Nimbus. We iterate it to a characteristic of the full cipher with probability 1/32, which in turn we use to attack the full cipher and find all the key material using 256 chosen plaintexts and 2^{10} complexity. Thus, we show that the inclusion of multiplication operations in a cipher does not necessarily protect against attacks.

1 Introduction

Nimbus [2] is a block cipher submitted as a candidate to the NESSIE project by Alexis Machado. The cipher has 64-bit blocks and 128-bit keys. It consists of five rounds of the form

$$Y_i = K_i^{odd} \cdot g(Y_{i-1} \oplus K_i),$$

where i is the round number, K_i and K_i^{odd} are subkeys (K_i^{odd} is always odd), '$\oplus$' is exclusive-OR (XOR), g is the bit-reversal function, '$\cdot$' is multiplication modulo 2^{64}, Y_0 is the plaintext, and Y_5 is the ciphertext.

The multiplication operation used in Nimbus was considered to be a "good" operation against differential cryptanalysis. The main "bad" property that was known about multiplication operation was that it preserves the least significant bits of form $1\underbrace{0\ldots0}_{i}$ for any i. Therefore, the cipher with the multiplication operation together with the bit-reversal function were considered to have a "good" defense against cryptanalysis.

In this paper we present two new differential properties of the multiplication operation with probability slightly larger than 1/2. In addition, the input/output difference in one of them has a palindromic form. Hence this differential property is not affected by the bit-reversal function. We also describe a set of new

* The work described in this paper has been supported by the European Commission through the IST Programme under Contract IST-1999-12324 and by the fund for the promotion of research at the Technion.

M. Matsui (Ed.): FSE 2001, LNCS 2355, pp. 187–195, 2002.

differential properties of the multiplication operation with probabilities that are bigger than random probability.

We use these differential properties to describe a 1-round iterative differential characteristic [1] with probability 1/2, whose iteration to the full cipher has a probability of 1/32. We use this characteristic to break Nimbus using 256 chosen plaintexts and 2^{10} complexity.

The paper is organized as follows: In Section 2 we describe the new differential properties of multiplication. In Section 3 we use them to devise a 1-round iterative characteristic of Nimbus. In Section 4 we present a chosen plaintext attack on the full Nimbus, and in Section 5 we show a chosen ciphertext attack, which is similar to the chosen plaintext attack. Finally, in Appendix A we describe a subkey recovering procedure that is used in the attacks.

2 New Multiplication Characteristics

Let A, B and K^{odd} be n-bit random numbers, where K^{odd} is odd.

Lemma 1. *[given without a proof] Let A and B be n-bit integers.*

- *If the least significant bits of A and B are 0 then*

$$A \oplus B = 0\underbrace{1\ldots1}_{n-2}0 \Leftrightarrow A + B = 0\underbrace{1\ldots1}_{n-2}0,$$

  where the numbers on the right hand sides of the equalities are given in a binary notation.
- *If the least significant bits of A and B are 1 then*

$$A \oplus B = 0\underbrace{1\ldots1}_{n-2}0 \Leftrightarrow A + B = 1\underbrace{0\ldots0}_{n-1}.$$

Claim 1: Let A, B and K^{odd} be n-bit random integers such that K^{odd} is odd. Then

$$A \oplus B = 0\underbrace{1\ldots1}_{n-2}0 \Rightarrow (A \cdot K^{odd}) \oplus (B \cdot K^{odd}) = 0\underbrace{1\ldots1}_{n-2}0 \tag{1}$$

holds with probability $1/2 + 2/2^n$, and also

$$A \oplus B = \underbrace{1\ldots1}_{n-1}0 \Rightarrow (A \cdot K^{odd}) \oplus (B \cdot K^{odd}) = \underbrace{1\ldots1}_{n-1}0 \tag{2}$$

holds with the same probability.

Proof: Without loss of generality, it is sufficient to prove the first equation; the proof for the second equation is similar.

We observe that multiplication by an odd number preserves the least significant bit. Assume $A \oplus B = 0\underbrace{1\ldots1}_{n-2}0$. There are two cases:

1. The least significant bits of A and B are 0.
   By Lemma 1, the sum $A + B$ is $0\underbrace{1\ldots1}_{n-2}0$ (i.e., $2^{n-1} - 2$). Hence, the sum $A \cdot K^{odd} + B \cdot K^{odd} = (A + B) \cdot K^{odd}$ is $(2^{n-1} - 2) \cdot K^{odd}$ for any odd K^{odd}. On the other hand, when the right hand side of Equation (1) holds, the same lemma ensures that $(A+B) \cdot K^{odd} = 2^{n-1} - 2$. Therefore, $(2^{n-1} - 2) \cdot K^{odd} = 2^{n-1} - 2$. This is satisfied when K^{odd} is either 1 or $2^{n-1} + 1$. As K^{odd} can obtain 2^{n-1} odd values, it happens with a probability of $2/2^{n-1}$.
2. The least significant bits of A and B are 1.
   By Lemma 1, the sum $A + B$ is $1\underbrace{0\ldots0}_{n-1}$ (i.e., 2^{n-1}). Hence, the sum $A \cdot K^{odd} + B \cdot K^{odd} = (A + B) \cdot K^{odd}$ is 2^{n-1} for any odd K^{odd}. By the same lemma, $(A \cdot K^{odd}) \oplus (B \cdot K^{odd})$ is always $0\underbrace{1\ldots1}_{n-2}0$. Therefore, it happens with probability 1.

Each case happens with a probability of 1/2, so the total probability is $1/2 + 2/2^n$. □

We conclude that Equation (1) and Equation (2) are always (with a probability of 1) satisfied for $K^{odd} = 1$ and $K^{odd} = 2^{n-1} + 1$. For all the other values of K^{odd} they are satisfied with probability 1/2.

Now we give a more general claim, that may be proved in a similar way.

Claim 2: Let A, B and K^{odd} be n-bit random variables such that K^{odd} is odd, and let $i, j \geq 0$, $i + j \leq n - 2$. Then

$$\begin{aligned} A \oplus B = \underbrace{0\ldots0}_{i}\ \underbrace{1\ldots1}_{n-2-i-j}\ 10\ \ \underbrace{0\ldots0}_{j} \Rightarrow \\ (A \cdot K^{odd}) \oplus (B \cdot K^{odd}) = \underbrace{0\ldots0}_{i}\ \underbrace{1\ldots1}_{n-2-i-j}\ 10\ \ \underbrace{0\ldots0}_{j} \end{aligned} \tag{3}$$

holds with probability

$$\begin{cases} 1/2^{j+1} \cdot 1/2^{i-1} + 1/2^{n-2-j} \cdot (1 - 2^{-(j+1)}), & for\ i \geq 1, \\ 1/2^{j+1} + 1/2^{n-2-j} \cdot (1 - 2^{-(j+1)}), & for\ i = 0. \end{cases}$$

In the case where the XOR difference has a palindromic form we obtain the following claim:

Claim 3: Let A, B and K^{odd} be n-bit random variables, and $0 \leq 2 \cdot i \leq n - 4$. Then

$$\begin{aligned} A \oplus B = \underbrace{0\ldots0}_{i}\ 01\ \underbrace{1\ldots1}_{n-4-2\cdot i}\ 10\ \ \underbrace{0\ldots0}_{i} \Rightarrow \\ (A \cdot K^{odd}) \oplus (B \cdot K^{odd}) = \underbrace{0\ldots0}_{i}\ 01\ \underbrace{1\ldots1}_{n-4-2\cdot i}\ 10\ \ \underbrace{0\ldots0}_{i} \end{aligned} \tag{4}$$

holds with probability $1/2^{2\cdot i+1} + 1/2^{n-2-i} \cdot (1 - 2^{-(i+1)})$.

3 A One-Round Iterative Characteristic

Nimbus has 64-bit words, so the characteristics of the multiplication operation described in Claim 2 have a probability of $1/2 + 2/2^{64}$. Moreover, the characteristics' probability are key dependent: when the subkeys 1 and $2^{63}+1$ are used in the multiplication operation the characteristics are always satisfied (probability 1). When other subkeys are used in the multiplication operation the probability of the characteristics is exactly 1/2.

The characteristic from Equation (1) has a palindromic form, so the g function always preserves the difference $0\underbrace{1\ldots1}_{n-2}0$. Hence, we get a one-round iterative characteristic $0\underbrace{1\ldots1}_{n-2}0 \rightarrow 0\underbrace{1\ldots1}_{n-2}0$ with a probability of $1/2 + 2/2^{64}$.

Nimbus has five rounds, so the characteristic $0\underbrace{1\ldots1}_{n-2}0 \rightarrow 0\underbrace{1\ldots1}_{n-2}0$ of the full cipher has a probability of $(1/2 + 2/2^{64})^5 \cong 2^{-5}$. For most keys the probability of the characteristic is 2^{-5}. There are, however, a few keys for which the characteristic has probabilities 2^{-4}, 2^{-3}, 2^{-2}, 2^{-1}, and 1.

4 A Chosen Plaintext Attack

Denote $\Delta = 0\underbrace{1\ldots1}_{n-2}0$. We generate structures of 64 plaintexts as follows: We randomly choose 32 plaintexts whose least and most significant bits are 0. We XOR each of these plaintexts with Δ, and get 32 pairs of plaintexts. Then, we build three additional structures by:

1. Complementing the most significant bits of all the plaintexts.
2. Complementing the least significant bits of all the plaintexts.
3. Complementing the most and the least significant bits of all the plaintexts.

We get four structures of 32 pairs of plaintexts each, and request to encrypt these 256 plaintexts to their ciphertexts under the unknown key.

Exactly two structures lead to difference Δ after one round. These two structures differ only by complementing the least significant bits of their plaintexts. Exactly one structure from these two structures leads to difference Δ after two rounds. We call such structure *Δ-structure*.

The iterative characteristic described in the previous section predicts that the pairs of plaintexts that belongs to the Δ-structure, have difference Δ after five rounds with a probability of 1/8. Thus, we have about $32 \cdot 1/8 = 4$ pairs from this structure that cause a difference of Δ after five rounds according to the above characteristic. Randomly, a pair (from any structure) can lead to a difference of Δ after five rounds with a probability of 2^{-64}, so we do not expect any wrong pair to lead to difference Δ after five rounds. Based on the ciphertext differences we can easily determine the which structure is the Δ-structure. We can, therefore, conclude which structure has difference Δ after one round, but

not after two rounds, and which structures do not have a difference of Δ after one round.

We continue the analysis with the 32 pairs of the Δ-structure. In this step of the attack we use the two-round characteristic $\Delta \to \Delta$. We use this characteristic starting from the third round. Given the ciphertexts of the above structures we find the subkey K_5^{odd} as follows:

1. The two-round characteristic $\Delta \to \Delta$ has probability 1/4, so a pair with difference Δ before the third round has difference Δ after round 4 with probability 1/4, i.e., there are 8 pairs on average with difference Δ after round 4. Each such pair does not lead to difference Δ after the fifth round with probability 1/2. Thus, we have 4 pairs on average with difference Δ after four rounds but without difference Δ after the fifth round. We call such a pair a *matching pair*.
2. We can recognize the matching pairs by testing that the following condition is satisfied, and discarding all the pairs that fail this test.

   Condition 1: All the three criteria are satisfied for matching pairs:
   - Multiplication by an odd number preserves the least significant bits of the form $1\underbrace{0\ldots0}_{i}$, where $i \geq 0$. Thus, matching pairs must have the bits 10 as the two least significant bits of the ciphertext XOR difference.
   - The matching pairs must have 0 as the least significant bit of the ciphertexts (otherwise, it would lead to difference Δ after five rounds as described in Section 2).
   - All matching pairs must have the same third least significant bit of their ciphertext XOR difference, because (for matching pairs) this bit depends only on the second least significant bit of the subkey used in multiplication.
     Note that we use this criterion when, there is a majority of matching pairs. Hence, we can recognize the right value of this bit, and in wrong pairs the corresponding bit equals to the right value with a probability of 1/2.

   This condition is satisfied randomly with a probability of $1/4 \cdot 1/2 \cdot 1/2 = 1/16$, hence we have about $(32 - 8) \cdot 1/16 \cong 2$ wrong pairs that satisfy Condition 1.
3. We select the pairs that satisfy Condition 1 (matching and wrong pairs). For each selected pair we solve the equation

$$0\underbrace{1\ldots1}_{n-2}0 \cdot K' = C_1 + C_2,$$

   where C_1, C_2 are the ciphertexts of this pair. We have 4 matching pairs that must suggest the same K' and only two wrong pairs that may suggest other values. Hence, we expect that K_5^{odd} is the most frequently suggested K'. Note that we do not know the most significant bit of K_5^{odd} from the above equation. So we actually have two possible subkeys K_5^{odd} which differ by the

most significant bit. It suffices to work only with one of them, and choose the right one in the later stage of the attack.

Given K_5^{odd} we find the subkeys K_5 and K_4^{odd} using the one-round characteristic $\Delta \to \Delta$. We use the same structure as in the previous stage, and decrypt the received ciphertexts by a half round using the recovered K_5^{odd}. We call these values *partially decrypted values*. Each pair leads to difference Δ after the third round, but does not lead to difference Δ after the 4th round with a probability of $1/2 - 1/4 = 1/4$. In this stage of the attack such a pair is called a *matching pair*. There are about $32 \cdot 1/4 = 8$ matching pairs. According to the partially decrypted values of the pairs that lead to difference Δ after 4 rounds, we can find the least significant bit of K_5. Using this knowledge we select the pairs according to Condition 1, except that here we are not looking at ciphertexts but at partially decrypted values. Wrong pairs may satisfy Condition 1 with probability 1/16. So only about $(32 - 16) \cdot 1/16 = 1$ wrong pair is selected. For the selected pairs (matching and wrong), we have the following equation:

$$\begin{cases} (A \cdot K') \oplus K'' = C \\ (B \cdot K') \oplus K'' = D \end{cases} \tag{5}$$

where A, B, K' and K'' are unknown values with $A \oplus B = \Delta$, and C, D are the partially decrypted values. We use the procedure described in Appendix A to find K' and K''. This procedure takes 4 pairs, solves the system of the above equations for these 4 pairs, and returns a set of values (K',K''). There are about 9 selected pairs (8 matching and 1 wrong). We randomly take 4 pairs (from the 9), run the procedure described in Appendix A and obtain a set of values. Then we take another subgroup of 4 pairs (from the 9), and run the procedure described in Appendix A on these pairs. If the returned set has an intersection with a previous set, we take the intersection as a new set instead of these two sets. We continue the process with other subgroups until the intersection of some sets gives the set of combinations which differ only by bits that we cannot uniquely identify by additional running of the procedure (see detailed description in Appendix A). The received set of candidates is expected to contain the correct value of (K_4^{odd},K_5), and it consists of 512 combinations on average. The probability that the received set of candidates does not contain the correct value is negligible. On average, about 6 subsets should be analyzed to identify the set that contains the correct value of (K_4^{odd},K_5). We cannot identify the correct value uniquely at this stage. We discard the wrong values from this set later in the attack.

All the other subkeys of rounds $1, \ldots, 4$ can be found in a similar way. We now show briefly how we do it efficiently.

For finding K_3^{odd} and K_4, we use the same structure as before. In this stage of the attack, the pairs causing difference Δ after two rounds and not after three rounds are called *matching pairs*. About half of the pairs are matching pairs, so we have $32 \cdot 1/2 = 16$ matching pairs. All the pairs from this structure have difference Δ after two rounds, and those of them that have difference Δ after three rounds were used in the previous stages of the attack. Hence, all the

remaining pairs (which do not have difference Δ after round 3) are the matching pairs. At this point we have $2 \cdot 512 = 1024$ possible combinations of the subkeys (K_4^{odd},K_5,K_5^{odd}). The right subkey must decrypt each matching pair to values with an XOR difference that has two least significant bits 10 (the first criterion of Condition 1). A wrong subkey combination passes this criterion with probability $(2^{-2})^{16} = 2^{-32}$ (for all 16 matching pairs). Thus, we can discard all wrong subkey combinations (K_4^{odd},K_5,K_5^{odd}), except for one wrong subkey combination that differs from the right subkey combination by the most significant bit of K_4^{odd}. This bit does not influence the analysis in this stage of the attack, so it is sufficient to continue the analysis only with one of the combinations, and identify the right combination in a later stage of the attack. Given the combination we choose to analyze, we decrypt the partially decrypted values by one more round. From now on, these values are called partially decrypted values. We run the procedure described in Appendix A on 4 randomly chosen pairs among the 16 matching pairs and intersect the resulting sets as in the previous case. About three runs of this procedure are needed to obtain a set of values that are expected to contain the correct value of (K_3^{odd},K_4). The wrong values from the remaining set are discarded later in the attack.

For finding K_2^{odd} and K_3, we use the structure that leads to difference Δ after one round but not to difference Δ after two rounds. As in the previous stage, we discard almost all wrong subkey combinations and have only two possibilities for (K_3^{odd},K_4,K_4^{odd},K_5,K_5^{odd}) which differ by the most significant bit of K_3^{odd}. This bit does not influence the analysis in this stage of the attack, so it is sufficient to continue the analysis only with one of the combinations, and identify the right combination in a later stage of the attack. Given the combination we choose to analyze, we decrypt the partially decrypted values by one more round, and run Appendix A on four pairs randomly chosen among the 32 pairs from this structure. About three runs of this procedure are needed to obtain the set of values that are expected to contain the correct value of (K_2^{odd},K_3). The wrong values from this set are discarded later in the attack.

Finally, we have the equation:

$$\begin{cases} ((A \oplus K) \cdot K') \oplus K'' = C \\ ((B \oplus K) \cdot K') \oplus K'' = D. \end{cases} \tag{6}$$

For finding the remaining subkeys (K_1,K_1^{odd},K_2), we use the pairs from the two structures that do not lead to difference Δ after one round. As in the previous stage, we discard almost all wrong subkey combinations and have only two possibilities of (K_2^{odd},K_3,K_3^{odd},K_4,K_4^{odd},K_5,K_5^{odd}) which differ by the most significant bit of K_2^{odd}. This bit does not influence the analysis in this stage of the attack, so it is sufficient to continue the analysis only with one of the combinations, and identify the right combination in a later stage of the attack. Given the combination we choose to analyze, we decrypt the partially decrypted values by one more round. Equation (6) is similar to Equation (5), so we find K' and K'' by running Appendix A on four pairs randomly chosen among the 64 pairs from these structures. About three runs of this procedure are needed to obtain the set

of values that are expected to contain the correct value of (K_1^{odd},K_2). For each received combination we can easily find an expected value of K_1. Thus we obtain a set of values that are expected to contain the correct value of (K_1,K_1^{odd},K_2).

We finally have $2 \cdot 512 = 1024$ possible combinations of all subkeys. The right subkeys may be found by encrypting one or two plaintexts.

In this attack we completely recover all the subkeys of Nimbus, which suffice to encrypt and decrypt without knowing the original key. The attack requires 256 chosen plaintexts with a complexity equivalent to 2^{10} full cipher encryptions.

5 A Chosen Ciphertext Attack

The chosen ciphertext attack works in a similar way, except that we know exactly which structures lead to difference Δ after one round, because in decryption the multiplication is the first operation. We need only two structures (in which the least significant bit is 1), i.e., only 128 ciphertexts, to find all the subkeys, except for K_5, K_5^{odd}. According to Appendix A, we need about 4 additional pairs (8 ciphertexts) in order to find these subkeys.

This attack requires 136 chosen ciphertexts, and its complexity is at most 2^{10} full cipher encryptions.

References

[1] Eli Biham, Adi Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer Verlag, 1993.
[2] Alexis Warner Machado, *The Nimbus Cipher: A Proposal for NESSIE*, NESSIE Proposal, September 2000.

A Recovering K' and K''

Given four pairs, we recover a set of options for K' and K'' by solving the system of eight equations received from combination of the Equation (5) of the four pairs. For pair $i \in 0 \ldots 3$, the equations are:

$$\begin{cases} (A_i \cdot K') \oplus K'' = C_i \\ (B_i \cdot K') \oplus K'' = D_i \end{cases}$$

where A_i, B_i, K' and K'' are unknown, and C_i, D_i are known values. In addition, we know that for all the pairs $A_i \oplus B_i = \Delta$, K' is an odd number, and the least significant bits of A_i and B_i are 0.

K' is an odd number, so its least significant bit is 1, and thus the multiplication of some number by K' preserves the least significant bit of this number. The least significant bit of A_0 and B_0 is 0, so we can easily find the least significant bit of K'' from C_0 and D_0. In contrast, we may get both values for the second least significant bit of K'', and we cannot identify this bit by this procedure. We can find the second least significant bit of K' in a deterministic

way: $1 \oplus$ the third least significant bit of $(C_0 \oplus D_0)$. If this bit is zero, then we can find the next (third) least significant bit of K' in a similar way using the 4th least significant bits of C_0 and D_0. If the third least significant bit is zero as well, we can find the next (fourth) least significant bit of K' in a similar way, and so on till the first 1 appears. We denote the number of such found zero bits of K' by Z. Note that for Z least significant bits of K'', starting from the third least significant bit, we cannot identify the correct value using this procedure.

When we meet the first 1 we run the following procedure: We need at least four pairs for the following operations. We start by finding the first two bits of K' that cannot be found in a previous step. We find these bits in both numbers (K' and K'') together, as follows:

- We guess the next 3 least significant bits of A_0 for one pair and the next 2 least significant bits of K'. Note that the most significant bit, among the guessed bits of A_0, is used only for better identifying the remaining bits, and is not remembered in the next step.
- From A_0 we get the 3 corresponding bits of B_0 (as we know $A_0 \oplus B_0 = \Delta$), multiply both numbers by the guessed K' (i.e., all the bits of K' guessed so far) and check if the XOR difference of results are equal to the XOR difference of C_0 and D_0. If it is not equal, then we continue with the next guess. Otherwise, we find the bits of K'' ($K'' = C_0 \oplus (A_0 \cdot K')$), decrypt the other pairs using the received K' and K'', and check that the received A_i and B_i have the required XOR difference.

There are at most 31 such steps. In this procedure, the following bits may obtain any possible values:

- complementing any of the $Z + 1$ least significant bits of K'' (starting from the second least significant bit),
- complementing the most significant bit of K',
- replacing K' by $K' \oplus (\underbrace{1 \ldots 1}_{61-Z} \ll (Z + 2))$,
- complementing the most significant bit of K''.

Hence we get 2^{Z+4} possible combinations of (K',K''), or about 512 possible combinations on average. If the four given pairs were chosen badly, then we can obtain some "noise" - additional unidentified bits of (K',K''). We can discard the wrong values of these bits by running this procedure with other pairs. We need to run the procedure about 3 times on average to discard the "noise".

All combinations of (K',K'') returned by this procedure have strong relations with themselves. If we have one of the possible combinations of (K',K'') we can obtain the others by complementing the corresponding bits. So it is sufficient to find one of them and to calculate all the others.

The whole procedure takes up to $31(steps) \cdot (2^3 \cdot 2^2)(guesses) \cdot 8$ (4 pairs) $\cong 2^{13}$ operations which are equivalent to about 2^8 one-round computations.