Slide Attack on Spectr-H64

Selçuk Kavut and Melek D. Yücel

Electrical & Electronics Eng. Dept., Middle East Technical University,
TÜBİTAK-BİLTEN, Information Technologies and Electronics Research Institute,
06531, Ankara, Turkey
{kavut, melek-yucel}@metu.edu.tr

Abstract. We compare one round diffusion characteristics of the block cipher Spectr-H64 to those of AES-Rijndael and Safer K-64, in terms of the Avalanche Weight Distribution (AWD) criterion and observe a weakness in the round transformation of Spectr-H64. We exploit this weakness to break one round of Spectr-H64 extracting half of the key bits, and develop a chosen plaintext slide attack against the overall encryption algorithm, which works for 2^{32} elements of the key space (out of 2^{256}). We also observe 2^{128} weak keys, for which encryption becomes the same function as decryption, and 2^{32} fixed points for each weak key.

Keywords: Slide attack, Spectr-H64, Avalanche Weight Distribution (AWD).

1 Introduction

Spectr-H64 is a 12-round Feistel-like cipher, which is designed by N. D. Goots, A. A. Moldovyan and N. A. Moldovyan [1]. It is based on data-dependent permutations and data-dependent transformation of round keys, with 64-bit input block length and 256-bit key length as explained in Appendix A. The output $P^r = (P_L^{\,r}, P_R^{\,r})$ of the rth round is found using the 32-bit left and right halves $P_L^{\,r\text{-}1}$ and $P_R^{\,r\text{-}1}$ of the previous round output as

$$P_L^{\,r} = f\,(P_R^{\,r\text{-}1}, P_L^{\,r\text{-}1}, Q_r), \qquad \textbf{(1)}$$

$$P_R^{\,r} = P_L^{\,r\text{-}1}, \qquad \textbf{(2)}$$

where the round keys Q_1, …, and Q_{12} are derived from the original key $K \in \{0,1\}^{256}$ (see Table A-1 in Appendix A).

The transformation f in (1) has a weak diffusion property such that, if k bits in the right half $P_R^{\,r\text{-}1}$of the rth round input $P^{r\text{-}1} = (P_L^{\,r\text{-}1}, P_R^{\,r\text{-}1})$ are complemented to obtain $\overline{P}^{\,r\text{-}1} = (P_L^{\,r\text{-}1}, \overline{P}_R^{\,r\text{-}1})$, then corresponding round outputs differ in the left half exactly by k bits, hence the Hamming weight of the difference vector remains the same, i.e.,

$$k = wt\,(P^{\,r\text{-}1} \oplus \overline{P}^{\,r\text{-}1}) = wt\,(P_R^{\,r\text{-}1} \oplus \overline{P}_R^{\,r\text{-}1}) = wt\,(P_L^{\,r} \oplus \overline{P}_L^{\,r}) = wt\,(P^r \oplus \overline{P}^{\,r}). \qquad \textbf{(3)}$$

In Section 2, we describe the experimental work that leads us to notice the weakness given by (3), and use it to break one round of Spectr-H64. We then propose a slide attack [2, 3] against the overall algorithm in Section 3, which works for 2^{32} weak keys of the form $K = (K_1, K_1, K_1, K_1, K_1, K_1, K_1, K_1)$ and $K_1 \in \{0,1\}^{32}$. This attack requires 2^{17} chosen plaintexts and 2^{32} comparisons. We describe the slide attack on a modified variant of Spectr-H64, and implement it by using 2^{17} chosen plaintexts and 2^{16} comparisons in a sorted list.

Finally, we discuss in Section 4 that for the 2^{128} keys of the form K = (K_1, K_1, K_2, K_2, K_3, K_3, K_4, K_4), which also include the above mentioned 2^{32} weak keys, encryption is the same function as decryption; thus, double encryption reveals the plaintext. For each key in this set of size 2^{128}, we observe that there are 2^{32} fixed points.

2 Breaking One Round of Spectr-H64

Our idea to break one round of Spectr-H64 is based upon the influential work on differential cryptanalysis [4, 5]. We utilise the weak diffusion property described by (3) to extract 128 key bits after one round of the algorithm, using a single known plaintext and 32 adaptively chosen plaintexts. To demonstrate the weakness given by (3), we compare one round diffusion characteristics of Spectr-H64 [1] to those of AES-Rijndael [6] and Safer K-64 [7]. The latter two algorithms are compared in [8] with respect to their Avalanche Weight Distribution (AWD) curves, which are defined as the Hamming weight histograms of ciphertext difference vectors [9]. Calling an encryption function *F*, the avalanche vector for an input P is the output difference vector $A = F(P) \oplus F(P \oplus e_i)$, where e_i is a unit vector with a 1 in position *i*. The AWD curves are defined simply as "the number of occurrences (in the sample space of all inputs) of the Hamming weight *wt*(A) sketched versus *wt*(A)"; and they are expected to be binomially distributed in the ideal case [9]. In Fig.1, we sketch

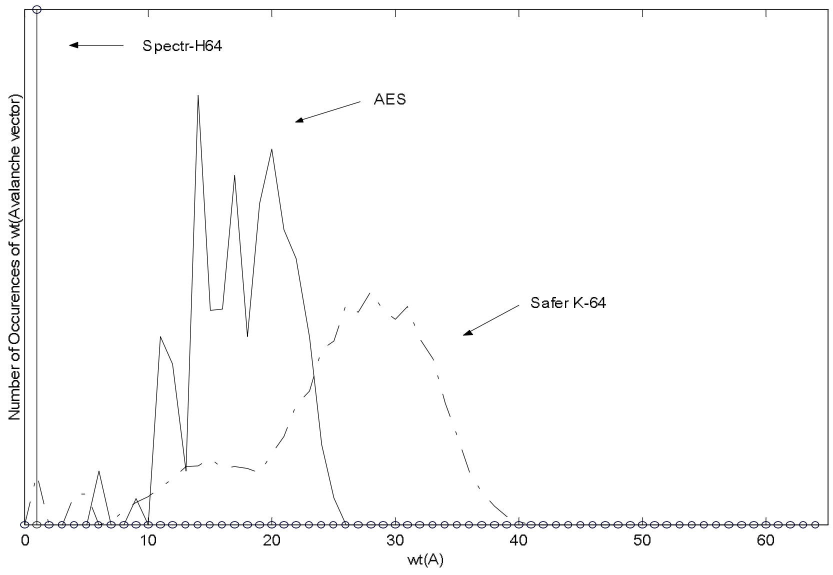


Fig. 1. One round AWD curves of Spectr-H64, Safer K-64 and AES corresponding to the worst case single bit input differences.

one round AWD curves corresponding to a (worst case) single bit plaintext difference for the block ciphers AES-Rijndael, Safer K-64, and Spectr-H64, using 1.000.000 randomly chosen plaintext pairs for AES-Rijndael, and 10.000 such pairs for Safer K-64 and Spectr-H64. (The graphs in the figure are normalized in order to make them comparable. Notice that since there are 2^{128} elements in the input vector space of Rijndael, as compared to 2^{64} elements of the other two ciphers, AWD curve of Rijndael is more fluctuating than others. This is so, because the experimental set of 10.000 random plaintexts divided by 2^{64} is a much higher fraction than 1.000.000 / 2^{128}. Hence, presented AWD curves of Safer K-64 and Spectr-H64 are relatively more reliable than that of Rijndael.) We observe that although perfect diffusion cannot be obtained in a single round of these ciphers; yet Safer K-64, and Rijndael are quite successful in diffusing single bit changes, whereas Spectr-H64 is not if the complemented bit is in the right half.

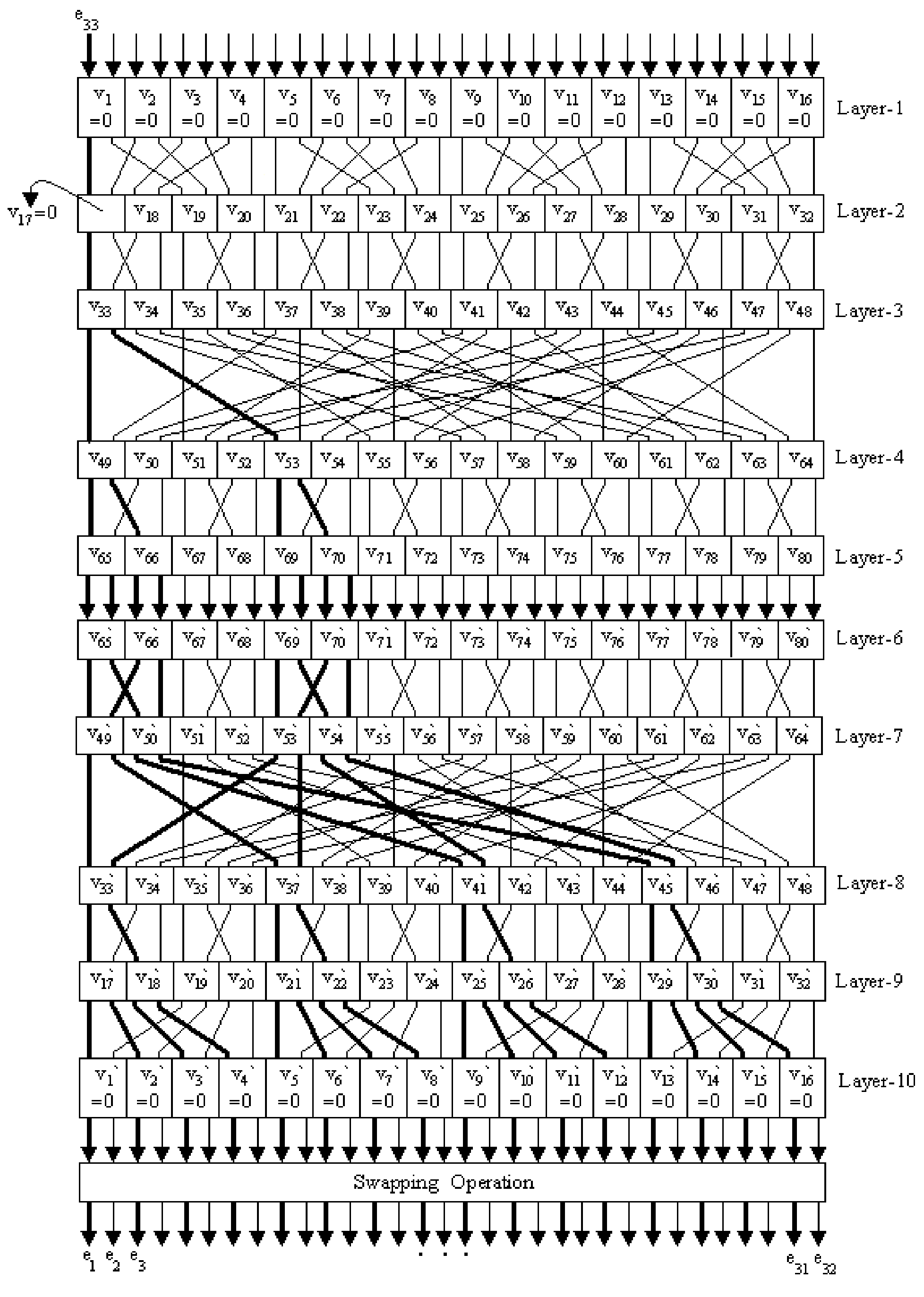
The reason why a right half difference vector propagates without any weight change in a given round is that, for the input vectors P and $\overline{P}$ with identical left half parts, all control bits of the data-dependent permutations $P_{32/80}$ and $P^{-1}_{32/80}$ and the output of the nonlinear function G remain the same under the same key (See Fig. A-1). Hence, the difference between the right half parts of P and $\overline{P}$ is only permuted and reflected to the first round output. It is possible to prove the weak diffusion property given by (3) starting from the round description equation (A-5) given in Appendix A.

The initial transformation does not improve this weakness, since its bit permutations between adjacent pairs cannot mix separate halves of the plaintext with each other. Hence, as far as the propagation of the right half input difference is concerned, the first round of the algorithm can be modelled as a cascade connection of $P_{32/80}$ and $P^{-1}_{32/80}$ boxes, as shown in Fig.2. The first 5 layers in this figure belong to the permutation $P_{32/80}$ and the last 5 layers belong to the inverse permutation $P^{-1}_{32/80}$. The XOR boxes between $P_{32/80}$ and $P^{-1}_{32/80}$ boxes are not included in Fig.2 since they do not have any effect on difference vectors.

Another weakness of the round transformation, exploited to extract the key bits after the first round, is the fact that the control vector V_1 used in Layer 1 of $P_{32/80}$ and Layer 10 of $P^{-1}_{32/80}$ boxes is completely known. Initially, V_1 is equal to the 16-bit right half of the rotated 32-bit left half of the input vector (see equation (A-2a) and Fig.A-1).

In the following, we use these two weaknesses of Spectr-H64 to describe how the control bits of one permutation layer are found using all control bits of the previous permutation layer and propagation of one bit input differences. Calling one round encryption of Spectr-H64, $F(P)$, for a single bit input difference vector e_i, $F(P \oplus e_i) = F(P) \oplus e_j$, for all $i \in \{33, 34, \ldots, 64\}$, and $j \in \{1, 2, \ldots, 32\}$. Knowing the output difference bits (e_j) caused by the input difference bits (e_i), the control vector bits are found completely, which are then used to obtain 32×4 key bits K_1, K_2, K_3 and K_4 by applying the inverse of the extension transformation.

In Fig.2, we sketch the first round propagation of one bit input difference vector e_{33} (corresponding to the state of all zero control bits at the first and last layers) without considering the initial transformation. Notice that only the right halves of propagating differences are shown in the figure since the left half differences are all zero vectors. All control bits of Layer 1 and Layer 10 are known since the plaintext is known (we



e33
v1 =0
v2 =0
v3 =0
v4 =0
v5 =0
v6 =0
v7 =0
v8 =0
v9 =0
v10 =0
v11 =0
v12 =0
v13 =0
v14 =0
v15 =0
v16 =0
Layer-1
v17=0
v18 v19 v20 v21 v22 v23 v24 v25 v26 v27 v28 v29 v30 v31 v32
Layer-2
v33 v34 v35 v36 v37 v38 v39 v40 v41 v42 v43 v44 v45 v46 v47 v48
Layer-3
v49 v50 v51 v52 v53 v54 v55 v56 v57 v58 v59 v60 v61 v62 v63 v64
Layer-4
v65 v66 v67 v68 v69 v70 v71 v72 v73 v74 v75 v76 v77 v78 v79 v80
Layer-5
Layer-6
Layer-7
Layer-8
Layer-9
Layer-10
Swapping Operation
e1 e2 e3
. . .
e31 e32

Fig. 2. The combination of the transformations $P_{32/80}$ and $P^{-1}_{32/80}$, illustrating the propagation of one bit input difference (e_{33}) after one round encryption of the algorithm without the initial transformation.

assume $v_i = v_i` = 0$ for $i = 1, 2, ..., 16$). The first control bit v_{17} of Layer 2 is found using the propagation of input difference e_{33}. If one of the output bits indicated by bold lines in Fig.2 is complemented in response to the input difference e_{33}, then $v_{17} = 0$; otherwise $v_{17} = 1$. As a second step, the value of v_{18} is found similarly, considering the propagation of either e_{37} or e_{39}, which are controlled by v_{18}. Knowing the positions of the output bits in response to each single input bit difference, one can then obtain all control bits of Layer 2.

Now since we know all the control bits of Layer 2, we can also obtain the input vectors of Layer 3 and corresponding output vectors of Layer 10; therefore the control bits of Layer 3 are found similarly. In this way all control bits and the key bits K_1, K_2, K_3 and K_4 are obtained using a known plaintext-ciphertext pair (P, C) and 32 adaptively chosen plaintext-ciphertext pairs (P_i, C_j) of one round encryption for each value of i ($P_i = P \oplus e_i$, $C_j = C \oplus e_j$, $i \in \{33, 34, \ldots, 64\}$, $j \in \{1, 2, \ldots, 32\}$).

If the initial transformation is included, we observe a similar distribution of output difference bits, as for one round of Spectr-H64 without the initial transformation, in response to a single input bit difference. Therefore, the propagation of one bit input difference can also be exploited similarly to extract the key bits K_1, K_2, K_3 and K_4 after one round of Spectr-H64 with the initial transformation.

3 Applying Slide Attack on Spectr-H64

In this section, firstly the slide attack is applied to a modified variant of Spectr-H64, without the initial and final transformations for the 2^{32} weak keys, and then it is shown how the initial and final transformations affect the number of operations necessary to break the algorithm for the same keys.

The first step in "sliding" direction can be dated back to a 1978 paper by Grossman and Tuckerman [10]. Afterwards, Biham's work on related-key cryptanalysis [11], and Knudsen's early work [12] are the milestones in this direction.

Slide attacks are, in general, independent of the exact properties of the iterated round function and the number of rounds, which is not the case for the conventional cryptanalytic tools such as differential and linear cryptanalysis for which each additional round requires an exponential effort from the attacker [2, 3]. A typical slide attack exploits the *self similarity* of a block cipher and views the cipher as a product of identical transformations F_Q (P), where Q is the round key (here F might include more than one round of the cipher). The only requirement on F is that it can be broken easily once an input-output pair is obtained. Calling the identical transformation $F_Q = F$, and the overall encryption function $E = F \circ F \circ \ldots \circ F = F^r$, the crucial observation leading to the slide attack [2, 3] is

$$P' = F(P) \text{ and } C = F^r(P) \text{ implies } C' = F^r(P') = F^r(F(P)) = F(F^r(P)) = F(C).$$

Hence, a standard slide attack tries to find plaintext-ciphertext pairs (P, C) and (P′, C′) with $C' = F(C)$. Such pairs are called *slid pairs*, and once a slid pair is found, an extra relation $P' = F(P)$ is obtained.

In order to find the degree of self similarity in Spectr-H64, we first examine the key schedule given in Table A-1. It is observed that Spectr-H64 has 2^{128} weak keys of the form K = (K_1, K_1, K_2, K_2, K_3, K_3, K_4, K_4) for which encryption is the same function as decryption, and thus double encryption reveals the plaintext (see Section 4). However, in this case, the key scheduling does not yield periodic round keys. More specifically, all round keys are different with the exception that $Q_1 = Q_9$, $Q_3 = Q_{10}$ and $Q_4 = Q_{12}$, therefore slide attack does not apply.

On the other hand, it is observed that if all K_i's ($i \in \{1, 2, ..., 8\}$) are equal to each other, the same round keys are produced for each round, and $Q_1 = Q_2 = ... = Q_{12} = Q$ (see Table A-1). Spectr-H64 can then be viewed as a product of identical permutations for this key subspace of 2^{32} weak keys in which all K_i's are equal to each other. Our attack is applicable for only these keys.

The overall algorithm consists of the initial transformations, 12 identical round transformations and the final transformation for the mentioned 2^{32} weak keys. Because of the weakness described in Section 3, once we obtain a pair (P, P′) of one round encryption where P′ = *F* (P), 128 key bits of K_1, K_2, K_3 and K_4 can be extracted in a negligible time. Therefore the *easy cryptanalysis* requirement on *F* is satisfied.

We implement the slide attack on a modified variant of Spectr-H64 without the initial and final transformations, as illustrated in Fig.3 in simplified form, where *f* denotes the transformation applied to the right half of the round input.

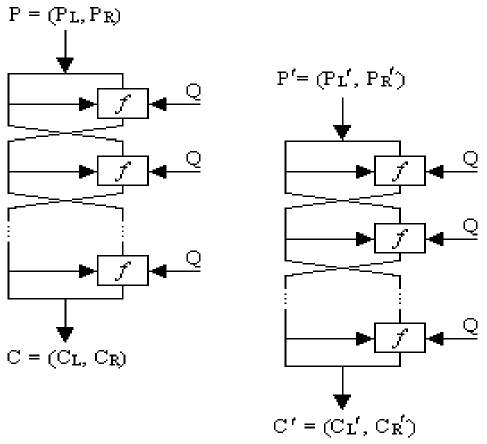


Fig. 3. Illustration of the slide attack on Spectr-H64, applicable for the 2^{32} keys, without initial and final transformations; if $C'_L = C_R$, then P′ is the one round encrypted form of P.

A chosen plaintext slide attack encrypts two pools of chosen plaintexts P = (P_L, P_R) and P′ = (P'_L, P'_R), where $P_L = P'_R$ is fixed, and P_R and P'_L both take 2^{16} random values. Then, checking whether $C'_L = C_R$, we expect to find a vector P′ in the second pool which is one round encrypted form of the element P from the first pool with high probability, by the birthday paradox. We identify such a slid pair by using a lookup table (or sorted list) with 2^{16} comparisons [2] in a negligible time. After finding a slid pair, other adaptively chosen 32 plaintext-ciphertext pairs of the first round are easily found with the help of the procedure explained below:

1. Encrypt 32 plaintexts $P_i = P \oplus e_i$, and 32 plaintexts $P'_j = P' \oplus e_j$ corresponding to the slid pair (P, P′); for all $i \in \{33, 34, \ldots, 64\}$, $j \in \{1, 2, \ldots, 32\}$, and obtain 32×32 ciphertext pairs C_i and C'_j. (Notice that the subscript of the ciphertext simply indicates the corresponding plaintext, $F^r(P_i) = C_i$, and it does not imply that $C_i = C \oplus e_i$.)
2. Check whether $C'_{iL} = C_{jR}$ for each ciphertext pair; if they are equal, corresponding plaintext pair P_i and P'_j satisfy the equation $F(P_i) = P'_j$.

Notice that since one bit input differences cause one bit output differences after one round encryption of Spectr-H64, the above procedure works. Now, one can extract the 128 key bits used for the first round, as explained in Section 3. Since our attack works for the 2^{32} weak keys in which all K_i's ($i \in \{1, 2, ..., 8\}$) are equal to each other, knowing the extracted key bits is equivalent to knowing all the key bits. The attack is independent of the number of rounds of the cipher, and requires 2^{17} chosen plaintexts and 2^{16} comparisons.

Next, we consider the effect of the initial and final transformations on the cryptanalysis of Spectr-H64. One can observe from Fig.A-1 that, if successive bits of the plaintext entering the same $P_{2/1}$ boxes are chosen the same, the initial transformation output is independent of the control bits. Hence, in the attempt of finding a slid pair, where one assigns the plaintexts $P = (P_L, P_R)$ and $P' = (P'_L, P'_R)$ such that $P_L = P'_R$, we can choose P_L and P'_R in the above mentioned form that the initial transformation cannot affect. For example, if P_L and P'_R are chosen as all zero vectors (see Fig.4), the effect of the initial transformation is removed; however, the final transformation still remains effective unless $Q_{FT_L} = Q_{FT_R}$.

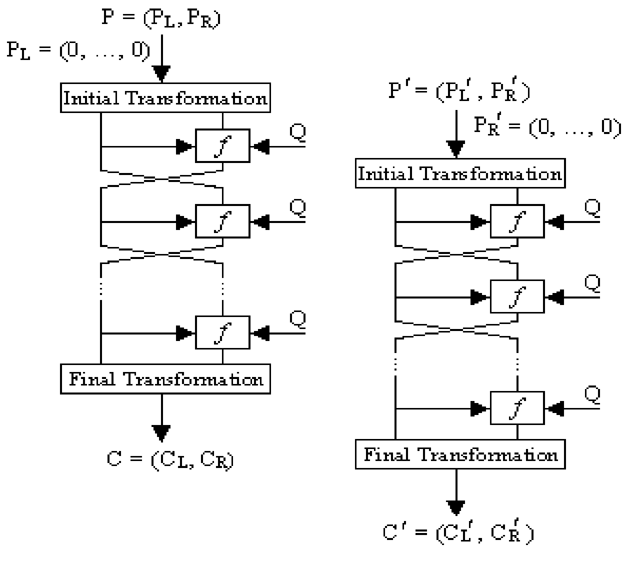


Fig. 4. Illustration of the effect of the initial and final transformations on the slide attack shown in Fig.3.

If $Q_{FT_L} = Q_{FT_R}$, the left and right halves of the final transformation inputs are subject to the same permutation; and one can find a match for the vectors C'_L and C_R, with 2^{16} comparisons. On the other hand, if $Q_{FT_L} \neq Q_{FT_R}$, we have to guess all possible (2×2^{16}) input vectors of the final transformation in order to find a match between C'_L and C_R.

The slide attack on Spectr-H64 also requires 2^{17} chosen plaintexts, as for the slide attack on the modified variant of Spectr-H64. However, in this case, since we have 2×2^{32} possible vectors C'_L and C_R due to the different left and right halves of the vector Q_{FT}, time complexity increases remarkably from 2^{16} to 2^{32}. In addition, there may be some false matches between the vectors C'_L and C_R while guessing the vectors Q_{FT_L} and Q_{FT_R}, which can be checked immediately during the process of extracting the key bits.[1] After finding a slid pair, 32 adaptively chosen plaintexts can be found using a similar procedure, which is explained for the attack on the modified variant. The key bits are extracted after breaking one round of Spectr-H64 with the initial transformation, as explained in Section 3.

4 Weak Keys and Fixed Points

The analysis of fixed points in DES weak keys and cycle structure of DES using these keys are explained by Coppersmith [13]. Moore and Simmons published more extensive work later on DES weak keys [14].

We observe that Spectr-H64 also has a set of 2^{128} weak keys of the form $K = (K_1, K_1, K_2, K_2, K_3, K_3, K_4, K_4)$, for which encryption is the same function as decryption. Since the round keys for decryption become the same as the round keys for encryption (see Table 1), double encryption reveals the plaintext.

Table 1. The round keys of Spectr-H64 used for both encryption and decryption, for the keys of the form $K = (K_1, K_1, K_2, K_2, K_3, K_3, K_4, K_4)$.

Q_{IT}	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}	Q_{FT}
	K_1	K_4	K_3	K_2	K_1	K_3	K_4	K_2	K_1	K_3	K_3	K_2	
	K_1	K_3	K_4	K_2	K_1	K_4	K_3	K_3	K_1	K_4	K_4	K_2	
K_1	K_3	K_1	K_1	K_3	K_4	K_2	K_2	K_4	K_3	K_1	K_1	K_3	K_1
	K_4	K_2	K_2	K_4	K_3	K_1	K_1	K_3	K_4	K_2	K_2	K_4	
	K_2	K_3	K_3	K_1	K_2	K_4	K_3	K_1	K_2	K_3	K_4	K_1	
	K_2	K_4	K_4	K_1	K_2	K_3	K_4	K_1	K_2	K_4	K_3	K_1	

Notice that, the final transformation is the inverse of the initial transformation, and the ith $(i = 1, 2, \ldots, 12)$ round key (A, B, C, D, E, F) of the decryption process is used to decrypt the $(13\text{-}i)$th round key (E, F, C, D, A, B) of the encryption process. If all round keys of the encryption and decryption are equal as in Table 1, the last six rounds of encryption can decrypt the intermediate cipher result obtained after the first

[1] The correctness for the guesses of the left and right halves of the vector Q_{FT} can also be checked by a trial encryption if we exploit the restriction on the key bits. In this case, the secret-key can be found directly without the cryptanalysis process explained in Section 3.

six rounds of encryption, whenever the left and right halves of the intermediate cipher result are equal to each other. Hence, there are 2^{32} fixed points for each weak key, which may cause some problems in using Spectr-H64 to build secure hash functions.

5 Conclusion

In this paper, we first describe an attack on single round of Spectr-H64, which exploits the diffusion weakness of the algorithm described by (3). This one round attack extracts 128 bits of the 256-bit key by using 1 known and 32 chosen plaintexts. Easy cryptanalysis of the first round then leads us to propose a slide attack against the overall algorithm, which uses 2^{17} chosen plaintexts. The described slide attack requires 2^{32} comparisons of 32-bit blocks, which takes much less work than 2^{32} encryption operations needed for a brute force attack in a subset of 2^{32} keys. The same attack is implemented on the modified variant of Spectr-H64 (only excluding initial and final transformations), and unknown key is identified using 2^{17} chosen plaintexts and 2^{16} comparisons. Since the attack is applicable for a small key subspace (one out of 2^{192}), it does not indicate that Spectr-H64 is weak, but it should be interpreted as an independent confirmation of the observation in [2], which states that auto-key ciphers and data-dependent transformations are potentially vulnerable to conventional slide attacks. On the other hand, 2^{128} weak keys and 2^{32} fixed points for each weak key that we observe, indicate that the key scheduling of the algorithm should be improved.

As a final remark, we should mention that the key scheduling of Spectr-H64 yields four round periodicity for 2^{64} weak keys of the form $K = (K_1, K_1, K_1, K_1, K_2, K_2, K_2, K_2)$, which makes the cipher vulnerable to a similar slide attack [2], where the identical transformation F includes four rounds of the cipher. Although one may suspect that this periodicity makes Spectr-H64 also a candidate for advanced slide attacks [3], we think that this is not possible. Because, a suitable advanced slide attack with four round periodicity would be a combination of "*complementation slide*" and "*sliding with a twist*"; however, the complementation slide is not applicable because of data-dependent permutations of the round keys.

References

1. N.D. Goots, A.A. Moldovyan, and N.A. Moldovyan, *Fast Encryption Algorithm Spectr-H64*. In: V.I. Gorodetski, V.A. Skormin, L.J. Popyack (Eds.), Information Assurance in Computer Networks: Methods, Models, and Architectures for Network Security. Lecture Notes in Computer Science, Vol. 2052, pp. 275-286, Springer-Verlag, 2001.
2. A. Biryukov and D. Wagner, *Slide Attacks*. In: L.R. Knudsen (Ed.), Fast Software Encryption – FSE'99. Lecture Notes in Computer Science, Vol. 1636, pp. 245-259, Springer-Verlag, 1999.
3. A. Biryukov and D. Wagner, *Advanced Slide Attacks*. In: B. Preneel (Ed.), Advances in Cryptology – EUROCRYPT'2000. Lecture Notes in Computer Science, Vol. 1807, pp. 589-606, Springer-Verlag, 2000.
4. S. Murphy, *The Cryptanalysis of FEAL-4 with 20 Chosen Plaintexts*. Journal of Cryptography, Vol.2, No.3, pp.145-154, 1990.

5. A. Shamir and E. Biham, *Differential Cryptanalysis of DES-like Cryptosystems*. Journal of Cryptology, Vol.4, No.1, pp.3-72, 1991.
6. J. Daemen and V. Rijmen, The Design of Rijndael, AES-The Advanced Encryption Standard. Springer-Verlag, 2002.
7. J.L. Massey, *Safer K-64: A Byte Oriented Block-Ciphering Algorithm*. In: R.J. Anderson, Fast Software Encryption – FSE'93. Lecture Notes in Computer Science, Vol. 809, pp.1-17, Springer-Verlag, 1994.
8. S. Kavut, and M.D. Yücel, *On Some Cryptographic Properties of Rijndael*. In: V.I. Gorodetski, V.A. Skormin, L.J. Popyack (Eds.): Information Assurance in Computer Networks: Methods, Models, and Architectures for Network Security. Lecture Notes in Computer Science, Vol. 2052, pp.300-311, Springer-Verlag, 2001.
9. E. Aras and M.D. Yücel, *Performance Evaluation of Safer K-64 and S-Boxes of Safer Family*. Turkish Journal of Electrical Engineering & Computer Sciences, Vol.9, No.2, pp. 161-175, 2001.
10. E.K. Grossman and B. Tuckerman, *Analysis of a Weakened Feistel-like Cipher*. Proc. International Conference on Communications, pp.46.3.1-46.3.5, Alger Press, 1978.
11. E. Biham, *New Types of Cryptanalytic Attacks Using Related Keys*. Journal of Cryptology, Vol.7, pp.229-246, 1994.
12. L.R. Knudsen, *Cryptanalysis of LOKI*91. In: J. Seberry and Y. Zheng (Eds.): Advances in Cryptology – ASIACRYPT'92. Lecture Notes in Computer Science, Vol. 718, pp.196-208, Springer-Verlag, 1993.
13. D. Coppersmith, *The Real Reason for Rivest's Phenomenon*, Proc. CRYPTO'85, pp.535-536, Springer-Verlag, 1986.
14. J.H. Moore and G.J. Simmons, *Cycle Structure of the DES with Weak and Semi-Weak Keys*, Proc. CRYPTO'86, pp.9-32, Springer-Verlag, 1987.

Appendix A: Description of Spectr-H64

The algorithm [1] is designed as a sequence of the initial transformation IT, 12 iterative rounds, and the final transformation FT. The overall encryption structure is shown in Fig.A-1. Round keys Q_1, …, and Q_{12} are derived from the original key $K \in \{0,1\}^{256}$, as shown in each column of Table A-1. Notice that in Table A-1, L_1, L_2, …, $L_8 \in \{0,1\}^{32}$ indicate segments of the original key $K=(L_1, \ldots, L_8)$ and A, B, C, D, E, F (which are also used in Fig.A-1) correspond to 32-bit segments of each round key $Q_i \in \{0,1\}^{192}$. The initial and final transformations use the 32-bit keys, $Q_{IT} = L_1$ and $Q_{FT} = L_2$. In Fig.A-1, q_i and $q_i^{`}$, indicate the elements of $Q_{IT} = (q_1, q_2, \ldots, q_{32})$ and $Q_{FT} = (q_1^{`}, q_2^{`}, \ldots, q_{32}^{`})$, respectively.

Table A-1. Key scheduling for Spectr-H64 encryption.

Round key segment	Q_1	Q_2	Q_3	Q_4	Q_5	Q_6	Q_7	Q_8	Q_9	Q_{10}	Q_{11}	Q_{12}
A	L_1	L_8	L_5	L_4	L_1	L_6	L_7	L_4	L_2	L_6	L_5	L_3
B	L_2	L_6	L_7	L_3	L_2	L_8	L_5	L_3	L_1	L_8	L_7	L_4
C	L_6	L_1	L_2	L_5	L_7	L_3	L_4	L_8	L_6	L_1	L_2	L_5
D	L_7	L_4	L_3	L_8	L_6	L_1	L_2	L_5	L_7	L_4	L_3	L_8
E	L_3	L_5	L_6	L_2	L_4	L_7	L_6	L_1	L_4	L_5	L_8	L_1
F	L_4	L_7	L_8	L_1	L_3	L_5	L_8	L_2	L_3	L_7	L_6	L_2

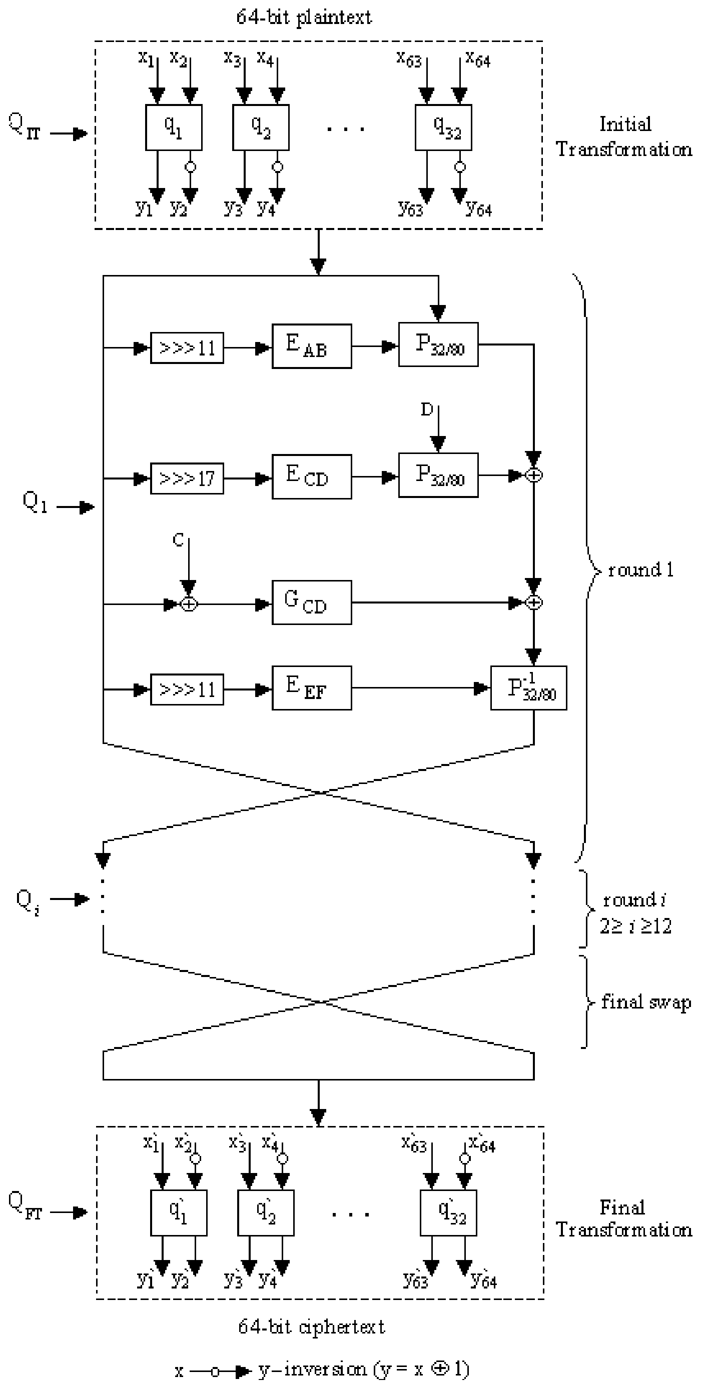


Fig. A-1. Overall encryption scheme of Spectr-H64.

The only difference between encryption and decryption of Spectr-H64 results from the key scheduling of the algorithm. If for $i \in \{1, 2,\ldots, 12\}$, the ith round key of the encryption process is Q_i = (A, B, C, D, E, F), then the (13-i)th round key of the decryption process is Q_{13-i} = (E, F, C, D, A, B). Similarly, the final vector Q_{FT} used for encryption is equal to the initial vector Q_{IT} of the decryption.

In each round of Spectr-H64, the following operations are used:

- cyclic rotation "$>>>k$" by fixed amount k,
- XOR operation "$\oplus$",
- nonlinear function G_{CD},
- data-dependent permutations $P_{32/80}$, $P^{-1}_{32/80}$, and
- extension operation E_{AB} (or E_{CD} and E_{EF}),

where the subscripts A, B,…, F denote 32-bit segments of the 192-bit round key Q_i = (A, B, C, D, E, F) used by the boxes G and E. The initial and final transformations also perform data-dependent permutations.

The nonlinear function G_{CD} uses the central segments C and D of the round key, and yields a 32-bit output vector Y, for a 32-bit input vector X, as

$$Y = G_{CD}(X) = M_0 \oplus M_1 \oplus (M_2 \otimes C) \oplus (M_2 \otimes M_5 \otimes D) \oplus (M_3 \otimes M_5) \oplus (M_4 \otimes D),$$

where the vectors M_0, …, M_5 are obtained recursively through X as follows:

$$M_0 = (m_1^{(0)}, m_2^{(0)}, \ldots, m_{32}^{(0)}) = X, \qquad \textbf{(A-1a)}$$

$$M_j = (m_1^{(j)}, m_2^{(j)}, \ldots, m_{32}^{(j)}) = (1, m_1^{(j-1)}, \ldots, m_{31}^{(j-1)}), \qquad \textbf{(A-1b)}$$

where $j = 1, \ldots, 5$.

The extension operation E_{AB} shown in Fig.A-1 is used to form an 80-bit control vector, which is represented as

$$E_{AB}(U) = (V_1, V_2, V_3, V_4, V_5) = V,$$

where $U \in \{0,1\}^{32}$, $V_1, V_2, \ldots, V_5 \in \{0,1\}^{16}$ and $V \in \{0,1\}^{80}$. The vectors V_1, $V_2,\ldots,V_5$ are determined according to

$$V_1 = U_R, \qquad \textbf{(A-2a)}$$

$$V_2 = \pi((U \oplus A)_R), \qquad \textbf{(A-2b)}$$

$$V_3 = \pi'((U \oplus B)_R), \qquad \textbf{(A-2c)}$$

$$V_4 = \pi'((U \oplus B)_L), \qquad \textbf{(A-2d)}$$

$$V_5 = \pi((U \oplus A)_L), \qquad \textbf{(A-2e)}$$

where the subscripts L and R denote left and right half of the vectors respectively and the fixed permutations π and π' are defined for $Z \in \{0, 1\}^{32}$ as

$$\pi(Z) = (Z_R^{>>>1}, Z_L^{>>>1}), \qquad \textbf{(A-3)}$$

$$\pi'(Z) = (Z_R^{>>>5}, Z_L^{>>>5}). \qquad \textbf{(A-4)}$$

The data-dependent permutation applied on the input $X \in \{0, 1\}^{32}$ by the $P_{32/80}$ box (Fig.A-2) produces the output $Y \in \{0, 1\}^{32}$:

$$Y = P_{32/80}(X,V),$$

where $V = (v_1, v_2, \ldots, v_{80})$ is the control vector formed by the extension operation.

The $P_{32/80}$ box consists of the 80 $P_{2/1}$ boxes arranged in 5 layers. Each $P_{2/1}$ box has one control bit v_i ($i \in \{1, \ldots, 80\}$), 2-bit input vector and 2-bit output vector. If the control bit $v_i = 0$, then the input vector is directly carried to the output, otherwise the input bits are interchanged. From Fig.A-2 it is seen that the $P_{32/80}$ box applies four permutations after the first 4 layers. It is important to observe that the initial values of the control bits ($v_1, v_2, \ldots, v_{16}$) of the first layer are equal to the right half part of 11-bit cyclically rotated form of the left half plaintext (see Fig.A-1 and equation (A-2a)).

The operation performed by $P^{-1}_{32/80}$ box is the inverse of the operation applied by the $P_{32/80}$ box. Therefore, the control bits of the last layer of $P^{-1}_{32/80}$ box are also equal to the right half part of 11-bit cyclically rotated form of the left half plaintext.

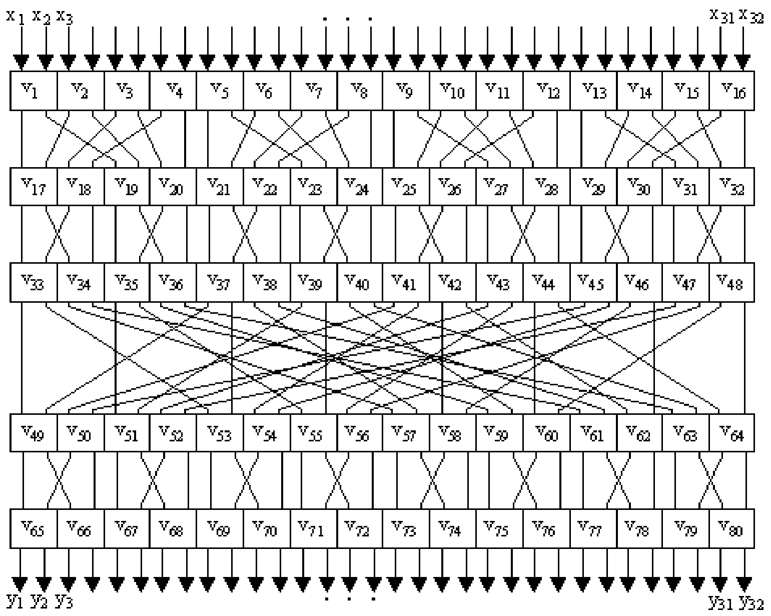

x1 x2 x3 . . . x31 x32
v1 v2 v3 v4 v5 v6 v7 v8 v9 v10 v11 v12 v13 v14 v15 v16
v17 v18 v19 v20 v21 v22 v23 v24 v25 v26 v27 v28 v29 v30 v31 v32
v33 v34 v35 v36 v37 v38 v39 v40 v41 v42 v43 v44 v45 v46 v47 v48
v49 v50 v51 v52 v53 v54 v55 v56 v57 v58 v59 v60 v61 v62 v63 v64
v65 v66 v67 v68 v69 v70 v71 v72 v73 v74 v75 v76 v77 v78 v79 v80
y1 y2 y3 . . . y31 y32

Fig. A-2. $P_{32/80}$ box.

We can now describe the round transformation of Spectr-H64, for $r = 0, 1, \ldots, 11$, as follows:

Let

$$X = E_{EF}(P_L^{\,r} >>> 11), \qquad \textbf{(A-5a)}$$

and

$$V = G_{CD}(P_L^{\,r} \oplus C) \oplus P_{32/80}(E_{CD}(P_L^{\,r} >>> 17), D) \oplus P_{32/80}(E_{AB}(P_L^{\,r} >>> 11), P_R^{\,r}) \qquad \textbf{(A-5b)}$$

then,

$$P_L^{\,r+1} = P^{-1}_{32/80}(X, V), \qquad \textbf{(A-5c)}$$

and

$$P_R^{\,r+1} = P_L^{\,r}, \qquad \textbf{(A-5d)}$$

where $(P_L^{\,r}, P_R^{\,r})$ is the intermediate result after the r^{th} round of encryption.

The initial (IT) and final (FT) transformations of the algorithm are represented as:

$$Y = IT(X, Q_{IT}); \quad Y` = FT(X`, Q_{FT}),$$

where X, X`, Y, Y` $\in \{0,1\}^{64}$, and Q_{IT}, $Q_{FT} \in \{0,1\}^{32}$. As shown in Fig.A-1, the initial transformation uses 32 bits of $Q_{IT} = (q_1, q_2, \ldots, q_{32})$ as the control bits of its 32 $P_{2/1}$ boxes, which interchange the two input bits with indices $2i$-1 and $2i$ whenever $q_i = 1$. Each even indexed bit at the $P_{2/1}$ box output is then inverted.

The final transformation becomes the inverse of the initial transformation, if $Q_{FT} = Q_{IT,}$ i.e., each even indexed bit of the input block is first inverted and then each pair of bits with indices $2i$-1 and $2i$ are interchanged whenever the control bit $q_i` = 1$, where $Q_{FT} = (q_1`, q_2`, \ldots, q_{32}`)$.