Slide Attacks

Alex Biryukov[⋆] David Wagner[⋆⋆]

Abstract. It is a general belief among the designers of block-ciphers that even a relatively weak cipher may become very strong if its number of rounds is made very large. In this paper we describe a new generic known- (or sometimes chosen-) plaintext attack on product ciphers, which we call the *slide attack* and which in many cases is independent of the number of rounds of a cipher. We illustrate the power of this new tool by giving practical attacks on several recently designed ciphers: TREYFER, M6, WAKE-ROFB, as well as on variants of DES and Blowfish.

1 Introduction

As the speed of computers grows, fast block ciphers tend to use more and more rounds, rendering all currently known cryptanalytic techniques useless. This is mainly due to the fact that such popular tools as differential [1] and linear analysis [11] are statistic attacks that excel in pushing statistical irregularities and biases through surprisingly many rounds of a cipher. However any such approach finally reaches its limits, since each additional round requires an exponential effort from the attacker.

This tendency towards a higher number of rounds can be illustrated if one looks at the candidates submitted to the AES contest. Even though one of the main criteria of the AES was speed, several prospective candidates (and not the slowest ones) have really large numbers of rounds: RC6(20), MARS(32), SERPENT(32), CAST(48). This tendency is a reflection of a belief/empirical evidence that after some high number of rounds even a relatively weak cipher becomes very strong. It is supported by the example of DES, where breaking even 16 rounds is already a very hard task, to say nothing about 32–48 rounds (e.g. double- or triple-DES). Thus for the cryptanalyst it becomes natural to search for new tools which are essentially independent of the number of rounds of a cipher. The first step in this direction can be dated back to a 1978 paper by Grossman and Tuckerman [5], which has shown how to break a weakened Feistel cipher [3] by a chosen plaintext attack, independent of the number of rounds. We were also inspired by Biham's work on related-key cryptanalysis [2], and Knudsen's early work [10].

⋆ Applied Mathematics Department, Technion - Israel Institute of Technology, Haifa, Israel 32000. Email: `albi@cs.technion.ac.il`

⋆⋆ University of California, Berkeley. Email: `daw@cs.berkeley.edu`

3 An 8-round Feistel cipher with eight bits of key material per round used to swap between two S-boxes S_0 and S_1 in a Lucifer-like manner. A really weak cipher by modern criteria.

In this paper we introduce a new class of generic attacks which we call *slide attacks* together with a new set of cryptanalytic tools applicable to all product (mainly iterative) ciphers and even to any iterative (or recursive) process over the finite domain (stream ciphers, etc.). Such attacks apply as soon as the iterative process exhibits some degree of self-similarity and are in many cases independent of the exact properties of the iterated round function and of the number of rounds.

While the two other generic cryptanalytic attacks—differential and linear analysis—concentrate mainly on the propagation properties of the encryption engine (assuming a strong key-scheduling which produces independent subkeys), the degree of self-similarity of a cipher as studied by slide attacks is a totally different aspect. Depending on the cipher's design, slide attacks range from exploiting key-scheduling weaknesses to exploiting more general structural properties of a cipher. The most obvious version of this attack is usually easy to prevent by destroying the self-similarity of an iterative process, for example by adding iteration counters or fixed random constants. However more sophisticated variants of this technique are harder to analyze and to defend against.

We start by analyzing several block ciphers that decompose into r iterations of a single key-dependent permutation F_i. We call such ciphers *homogeneous*. This usually arises when the key-schedule produces a periodic subkey sequence, when $F_i = F_j$ for all $i \equiv j \bmod p$ where p represents the period. In the simplest case, $p = 1$ and all round subkeys are the same. We call these attacks *self-related key attacks*, since they are essentially a special case of related-key attacks [2, 10]. Note, however, that these attacks require only a known- (or sometimes chosen-) plaintext assumption and thus are much more practical than the related key attacks. For the case of block ciphers operating on a n-bit block, the complexity of slide attacks (if they work) is usually close to $O(2^{n/2})$ known plaintexts. For Feistel ciphers where the round function F_j modifies only half of the block, there is also a chosen-plaintext variant which can often cut the complexity down to $O(2^{n/4})$ chosen texts.

A somewhat less expected observation is that schemes relying on key-dependent S-boxes are also vulnerable to sliding. In general, autokey ciphers and data-dependent transformations are potentially vulnerable to such attacks. We summarize our results in Table 1.

This paper is organized as follows. In Section 2, we describe the details of a typical slide attack. We proceed with an introductory example: a 96-bit DES variant with 64-rounds, which we call 2K-DES, Section 3. The next three sections are devoted to cryptanalysis of several concrete cipher proposals. Section 4 breaks TREYFER, a cipher published in *FSE'97*; Section 5 breaks M6, a cipher proposed for the FireWire standard; and Section 6 analyzes stream cipher proposals based on WAKE presented at *FSE'98*. Section 7 shows slide attacks on ciphers with key-dependent S-boxes, focusing on a variant of Blowfish with zero round subkeys.

Cipher	(Rounds)	Key Bits	Our Attack	
			Data Complexity	Time Complexity
Blowfish[1]	(16)	448	2^{27}CP	2^{27}
M6	(10)	40–64	2^{16}CP	2^{16}
M6	(10)	40–64	2^{32}KP	2^{27}
Treyfer	(32)	64	2^{32}KP	2^{44}
2K-DES	(64)	96	2^{32}KP	2^{50}
WAKE	(k)	$32n$	2^{17}KP	2^{17}

[1] – Modified variant, without round subkeys. KP — known-plaintext, CP — chosen-plaintext.

Table 1. Summary of our attacks on various ciphers.

2 A typical slide attack

In Figure 1, we show the process of encrypting the n-bit plaintext X_0 under a typical product cipher to obtain the ciphertext X_r. Here X_j denotes the intermediate value of the block after j rounds of encryption, so that $X_j = F_j(X_{j-1}, k_j)$. For the sake of clarity, we often omit k by writing $F(x)$ or $F_i(x)$ instead of $F(x,k)$ or $F_i(x,k)$.

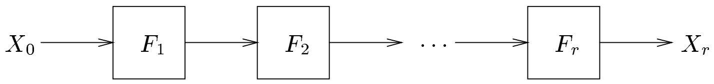

Fig. 1. A typical block cipher

As we mentioned before, the attack presented in this note is independent of the number of rounds of the cipher, since it views a cipher as a product of identical permutations $F(x,k)$, where k is a fixed secret key (here F might include more than one round of the cipher). Moreover its dependence on the particular structure of F is marginal. The only requirement on F is that it is very weak against known-plaintext attack with two plaintext-ciphertext pairs. More specifically, we call F a **weak** permutation if given the two equations $F(x_1,k) = y_1$ and $F(x_2,k) = y_2$ it is "easy" to extract the key k. This is informal definition since the amount of *easiness* may vary from cipher to cipher. We can show that 3 and 4 rounds of DES form a weak permutation [4]. One and a half round IDEA is also weak.

[4] For $F =$ three rounds of DES, using the meet in the middle attack with two pairs we can find 16 "expanded" bits of the key and then the full DES 56-bit key can be found in time faster than that of one DES encryption.

We next show in Figure 2 how a slide attack against such a cipher might proceed. The idea is to “slide” one copy of the encryption process against another copy of the encryption process, so that the two processes are one round out of phase. We let X_0 and X'_0 denote the two plaintexts, with $X_j = F_j(X_{j-1})$ and $X'_j = F_j(X'_{j-1})$. With this notation, we line up X_1 next to X'_0, and X_{j+1} next to X'_j.

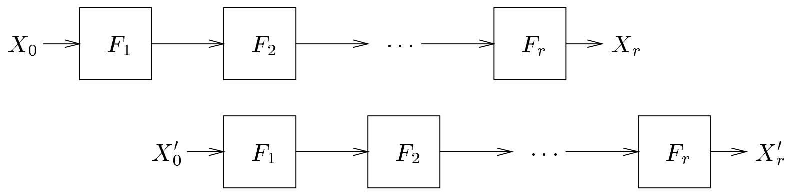

Fig. 2. A typical slide attack

Next, we suppose that $F_j = F_{j+1}$ for all $j \geq 1$; this is the assumption required to make the slide attack work. In this case, all the round functions are the same, so for the remainder of this section we will drop the subscripts and simply write F for the generic round transform.

The crucial observation is that if we have a match $X_1 = X'_0$, then we will also have $X_r = X'_{r-1}$. The proof is by induction. Suppose that $X_j = X'_{j-1}$. Then we may compute $X_{j+1} = F(X_j) = F(X'_{j-1}) = F(X'_{j-1}) = X'_j$, which completes the proof. Therefore, we call a pair (P, C), (P', C') of known plaintexts (with corresponding ciphertexts) a **slid** pair if $F(P) = P'$ and $F(C) = C'$.

With this observation in hand, the attack proceeds as follows. We obtain $2^{n/2}$ known texts (P_i, C_i), and we look for slid pairs. By the birthday paradox, we expect to find about one pair of indices i, i' where $F(P_i) = P_{i'}$, which gives us a slid pair.

Furthermore, slid pairs can often be recognized relatively easily. In the case of Feistel ciphers, $F((l, r)) = (r \oplus f(l), l)$; therefore, the condition $F(x) = x'$ can be recognized by simply comparing the left half of x against the right half of x', and this filtering condition eliminates all but $2^{-n/2}$ of the wrong pairs. In general, we recognize slid pairs by checking whether it is possible that $F(P_i) = P_{i'}$ and $F(C_i) = C_{i'}$ both hold for some key. When the round function is *weak*, we are assured that this condition will be easy to recognize. Moreover, when X, X' do not form a slid pair, they have only a 2^{-n} probability of surviving the filtering condition, so the signal-to-noise ratio is very good unless the key is longer than the block length.

Once we have found a slid pair, we expect to be able to recover some key bits of the cipher. If the round function is *weak*, we can in fact recover the entire key with not too much work. In general, we expect a single slid pair to disclose about n bits of key material; when the cipher’s key length is longer than n bits, we may use exhaustive search to recover the remainder of the key, or we may alternatively obtain a few more slid pairs and use them to learn the rest of the key material.

Let us summarize the attack. For a cipher with n-bit blocks, we need about $2^{n/2}$ known plaintexts. Checking all 2^n pairs, we expect to see a slid pair that discloses the key (or a large portion of the key material, at least).

The complexity of the attack as we have stated it is $2^{n/2}$ known texts and 2^n off-line work. However, for many ciphers the amount of off-line work required can be reduced to about $O(2^{n/2})$ operations by use of specific properties of the round function. In the case of a Feistel cipher, for a slid pair n bits of P_i, C_i must match another n bits of $P_{i'}, C_{i'}$; such matches can be identified by using a lookup table (or sorted list) with $2^{n/4}$ entries. This usually leads to a chosen-plaintext slide attack with $O(2^{n/4})$ data and time complexity.

3 Modified DES Example: 2K-DES

The following constitutes in our opinion a nice problem for a student crypto-course or an introductory crypto-textbook. Suppose one proposes to strengthen DES in the following way. One increases the number of rounds from 16 to 64, and extends the number of key-bits from 56 to 96 in the following simple way: given two independent 48-bit keys K_1, K_2 one uses K_1 in the odd rounds and K_2 in the even rounds instead of DES subkeys. This version is obviously immune to exhaustive search. The conventional differential and linear attacks probably will also fail due to the increased number of rounds. The question is: "Is this cipher more secure than DES?" Below we show two attacks on this cipher which use the symmetry of the key-scheduling algorithm and are independent of the number of rounds.

One very simple way to attack such cipher is as follows. For any known plaintext-ciphertext pair (P, C), decrypt ciphertext C one round under all possible 2^{48} guesses of K_2. For each of the 2^{48} resulting texts C', request the encryption $P' = E_K(C')$. This is equivalent to decryption all way back to the plaintext P and further by one more round to $F^{-1}(P, K_2) = P'$. Since the subkey K_2 is known, one can check if the equation $F(P', K_2) = P$ holds (here F includes the Feistel swap of the halves). This procedure leaves only the correct guess for K_2 with high probability. Now K_1 can be found by exhaustive search. This simple attack uses one known-plaintext (P, C) pair, 2^{48} adaptive chosen plaintexts and 2^{49} time. A similar attack will actually work for any "almost"-symmetric key-scheduling. Notice that if the number of rounds r is odd and key-scheduling is symmetric then double encryption with such Feistel-cipher becomes an identity permutation.

The data complexity of attack can be improved using the ideas of the present paper. By applying slide techniques, we can show that this cipher is much weaker than one would expect even when its number of rounds r is arbitrarily large. For any fixed value of K_1, K_2 this cipher can be viewed as a cascade of $\frac{r}{2}$ identical fixed permutations. Thus given a pool of 2^{32} known plaintexts, one can recover all 96 bits of the secret key just by checking all the possible pairs in about $2^{63}/64 = 2^{57}$ naive steps (each step is equivalent to one 2K-DES encryption operation). Each pair of plaintexts (P, P^*) suggests 2^{16} candidates for K_1 and 2^{16} candidates for K_2 which are immediately checked against a pair of corresponding ciphertexts (C, C^*). Thus on the average after this process we are left with a few candidate 96-bit keys which can be further checked with trial encryption. Using a more sophisticated approach (ruling out many pairs simultaneously) it is possible to reduce the work factor considerably. For each plaintext we guess the left 24 bits of K_1, which allows us to calculate 16-bits of the S-box output and thus 16-bits of the possible related plaintext and 16-bits of related ciphertext. This gives a 32-bit condition on the possible related plaintext/ciphertext pair; then analyzing the pool of texts will take a total of $2^{24} \times 2^{32}/64 = 2^{50}$ steps.

4 TREYFER

In this section we apply slide attacks to cryptanalyze TREYFER, a block-cipher/MAC presented at FSE'97 by Gideon Yuval [6] and aimed at smart-card applications. It is characterized by a simple, extremely compact design (only 29 bytes of code) and a very large number of rounds (32). We show an attack on TREYFER that is independent of the number of rounds and exploits the simplicity of key-schedule of this cipher. It uses 2^{32} known-plaintexts and requires 2^{44} time for analysis.

Description of TREYFER

TREYFER is a 64-bit block cipher/MAC, with a 64-bit key, designed for a very constrained architectures (like a 8051 CPU with 1KB flash EPROM, 64 bytes RAM, 128 bytes EPROM and peak 1MHz instruction rate). The algorithm is as follows:

```
for(r=0; r < NumRounds; r++){
    text[8] = text[0];
    for(i=0; i<8; i++)
       text[i+1] = (text[i+1] + Sbox[(key[i]+text[i])%256])<<< 1;
      //rotate 1 left
    text[0] = text[8];
}
```

Here `text` is an eight-byte plaintext, `key` is an eight-byte key, `S-box` denotes an 8x8-bit S-box chosen at random, and `NumRounds` stands for 32 rounds. After 32 rounds of encryption `text` contains eight-byte ciphertexts. One of the motivations behind the design of this cipher was that in spite of the simplicity of the round function a huge number of rounds (32) will make any possible attack impractical.

As an aside (without any connection to our attack), we observe that TREYFER exhibits much weaker diffusion in the decryption direction: it takes two rounds for a one-byte difference to influence all eight bytes in the encryption direction, but it takes seven rounds in the decryption direction.

Our Attack on TREYFER

The idea of our attack is very similar to the related-key attacks [2, 10], however our attack is known-plaintext and not chosen-key like the attacks in [2].

In our attack we use the fact that due to hardware constraints the designers of TREYFER sacrificed a proper key-scheduling to make a more compact and faster cipher. Thus key-scheduling of TREYFER simply uses its 64-bit key K byte by byte. This is done exactly in the same fashion at each round.

However the simplicity of key-schedule causes TREYFER to be a cascade of 32 identical permutations! Thus suppose that two plaintexts P and P^* are encrypted by TREYFER to C and C^*. Denote the intermediate encrypted values after each round by $P_1, \ldots, P_{32}$, where $P_{32} = C$. Denote the round encryption function of TREYFER by F. Now, if two plaintexts are related by a one-round encryption as $F(P, K) = P^*$ then it must be that the same relation holds for the ciphertexts $F(C, K) = C^*$. Due to simplicity of the round function F, given a properly related pair the full 64-bit key K of TREYFER can be derived either from equation $F(P, K) = P^*$ or from equation $F(C, K) = C^*$. If P, P^* is a properly related pair both equations suggest the same value of the key. However if the pair is not properly related there is no reason for the two keys to be equal.

Thus on TREYFER with arbitrary number of rounds and with arbitrarily chosen S-box it is possible to mount an attack with about 2^{32} known plaintexts and in the time of 2^{44} offline TREYFER encryptions (performed on the attacker's computer and not on the slow smart-card processor). Due to the birthday paradox a pool of 2^{32} known plaintexts will contain a properly related pair with high probability. Thus a naive approach is to try all the possible 2^{63} pairs, and each time the two equations $F(P, K) = P^*$ and $F(C, K) = C^*$ suggest the same 64-bit key, check this candidate key with trial encryption. Since per each pair we perform 1/16 of the TREYFER encryption, the overall complexity of this naive attack is 2^{59} TREYFER encryptions, which is still faster than exhaustive search. However we can do better than that if for each plaintext we do $2^{16} = 2^8 \cdot 2^8$ guesses of the two subkeys `k[7],k[0]`. For each guess we arrive at a 32-bit condition on the possible co-related plaintext. Thus on the average only one out of 2^{32} plaintexts passes the 32-bit condition and it can be easily found in a sorted array of plaintexts. Then the newly formed pair is checked for the version of the full 64-bit key as it was done in a naive approach. The time required by the analysis phase of this attack is equivalent to $2^{16} \cdot 2^{32} \cdot \frac{1}{16} = 2^{44}$ TREYFER encryptions.

Thus we have shown an attack on TREYFER, with 2^{32} known plaintexts, 2^{44} time of analysis and 2^{32} memory. The interesting property of this attack is that it is independent of the number of rounds and of the exact choice of the S-box. This attack seems to be on the verge of practicality, due to very slow smart-card encryption (6.4 msec per block) and very slow communication wire (10KBPS) speed. However this task is easily parallelizable if an attacker obtains many smart-cards containing the same secret key. Once the attacker receives the data, the analysis can be done in a few days on an average computer.

It should be possible to make TREYFER immune to this attack by adding a more complex key-schedule[5].

5 M6

M6 is a cipher proposed in the IEEE1394 FireWire standard. FireWire is a technology for digital interconnection between consumer electronics and personal computers [13, 16]. It is already shipping in some computers, and it includes provisions for content protection (i.e. copyright). Those provisions use the M6 cipher for encrypting content.

We show how to break M6 with 2^{32} known texts and off-line work comparable to 2^{27} trial encryptions. We also give a chosen-plaintext slide attack that needs just $2^{16.3}$ chosen texts and a similar amount of work. This shows that M6 is highly susceptible to slide attacks, and offers only a relatively low level of security.

The standard also suggests that it might be possible to create other variations on the basic M6 construction by changing the order of the g functions, by swapping additions for XORs (or vice versa), and/or by changing the rotation amounts. We note that the slide attack on M6 is very robust, in the sense that it applies to all of these variations on M6; the only property needed is that the same subkeys are used in every round.

The cipher

We briefly describe the cipher M6 here, for convenience. See also Figure 3.

M6 uses a 40–64 bit key, with a simple key schedule. Let K_1 be the high 32 bits of the key, and W be the lower 32 bits of the key (so that K_1 and W share 24 bits in common for a 40-bit key case). Set $K_2 = K_1 + W \bmod 2^{32}$. Then K_1, K_2 are the output of the key schedule.

M6 is a 10-round Feistel cipher. Define the Feistel function f by

$$\begin{array}{ll} g_1(x) = x \oplus K_1 & g_2(y) = ROL_2(y) + y + 1 \bmod 2^{32} \\ g_3(z) = ROL_8(z) + z \bmod 2^{32} & g_4(a) = a + K_2 \bmod 2^{32} \\ g_5(b) = ROL_{14}(b) + b \bmod 2^{32} & f(x) = (g_5 \circ g_4 \circ g_3 \circ g_2 \circ g_1)(x) \end{array}$$

where $ROL_s(x)$ denotes the result of rotating the 32-bit quantity x left by s bit positions. The round function F updates a 64-bit block (x, y) according to

$$F((x, y)) = (y + f(x) \bmod 2^{32}, x).$$

[5] Following the results of this paper round counters were introduced into the round function of TREYFER, as a counter-measure against such attacks [7].

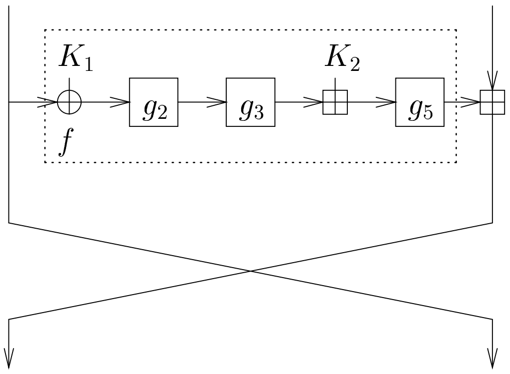

Fig. 3. One round of the M6 block cipher

Analysis of f

We first show that the f function is *weak* in the sense of Section 2, i.e. that we can recover the key "easily" given two known input-output pairs $c_j = f(x_j)$ for the f function.

First, note that the function g_5 can be inverted relatively easily. Suppose that we wish to calculate a list of all pre-images $b = g_5^{-1}(c)$ of a given output c for g_5. Guess the high 14 bits of the input b. Since $c \equiv g_5(b) \equiv \text{highbits}_{14}(b) + b \bmod 2^{14}$, the low 14 bits of b are easily obtained. Then considering $c \bmod 2^{28}$ gives the next 14 low bits of b, and that is sufficient to recover a suggested value b in its entirety. Finally, we may verify that $g_5(b) = c$ as required.

This shows how to invert g_5 with about $3 \cdot 2^{14}$ operations. Alternatively, one could invert g_5 in one operation with the use of a pre-computed lookup table. In either case, the ability to invert g_5 allows us to recover a known input-output pair x, b for the function $g_4 \circ g_3 \circ g_2 \circ g_1$ from each known input/output pair $c = f(x)$ for the f function.

To recover the key from two known input-output pairs x_j, c_j, we simply note that

$$(g_3 \circ g_2 \circ g_1)(x_1) - (g_3 \circ g_2 \circ g_1)(x_2) \equiv b_1 - b_2 \bmod 2^{32}.$$

The function $g_3 \circ g_2 \circ g_1$ depends only on K_1, so we guess K_1, compute the left-hand side of the previous equation, and check whether it is equal to $b_1 - b_2 \bmod 2^{32}$. With about 2^{32} evaluations of $g_3 \circ g_2 \circ g_1$ (equivalent to about $2^{26.4}$ trial encryptions), we expect to find K_1; then K_2 and K can be recovered easily afterwards.

A known-plaintext attack

The known-plaintext attack on M6 is quite standard (see Section 2). Obtain 2^{32} known plaintexts, and look for a tell-tale match in the plaintext and ciphertext with the use of a lookup table. Since the filtering is excellent, we expect one right pair and no wrong pairs.

The right pair will suggest two known input/output pairs $c_j = f(x_j)$ for the f function $(j = 1, 2)$. As discussed above, we can recover the key from this information with off-line work equivalent to about 2^{27} trial encryptions. The key K can then be confirmed with a single trial encryption.

Summarizing, we have demonstrated an attack on M6 that requires just 2^{32} known plaintexts, 2^{32} memory, and off-line work comparable to 2^{27} trial encryptions.

A chosen-plaintext attack

It is possible to break M6 even more quickly under the chosen-plaintext model. The attack requires about $2^{16.3}$ chosen plaintexts and 2^{27} work. The key to the reduction in texts is the use of carefully-chosen structures.

One simplistic approach using 2^{17} chosen plaintexts works like this: fix x, choose 2^{16} plaintexts (x, y) by varying over 2^{16} random values for y, and then choose another 2^{16} plaintexts (y', x) by varying over another 2^{16} random choices for y'. We obtain 2^{32} pairs of plaintexts. A right pair occurs with probability 2^{-32} so we expect about one right pair. This right pair can be recognized and used to recover the key K as before. This usage of structures was first pioneered by Biham in his work on related-key cryptanalysis [2].

It is possible to improve on this simplistic approach by taking advantage of regularities in the f function. One can show that $f(x) \bmod 5 \in \{0, 4\}$ for all x, and this can be used to reduce the number of plaintexts needed. This improvement yields an attack needing $2^{16.3}$ chosen plaintexts and off-line work comparable to 2^{27} trial encryptions. Details are omitted due to a lack of space.

6 Stream ciphers, and WAKE-ROFB

It is also possible to mount slide attacks against stream ciphers. We show how to break two recent WAKE variants proposed in *FSE'98* under the name WAKE-ROFB [3]. Our attacks work only under restrictive assumptions on the IV selection and re-synchronization mechanism.

Note that this does not reflect poorly on the core of the WAKE-ROFB design; it merely shows that dealing with re-synchronization can be tricky, because it introduces the possibility of chosen-text attacks. (See also [4, 15].) In short, WAKE-ROFB is not broken. We point out these attacks merely to illustrate the intriguing theoretical possibility of applying slide attacks to stream ciphers.

WAKE-ROFB is a stream cipher with $32n$ bits of internal state, organized into n 32-bit words. The words are updated via a simple analogue of a non-linear feedback shift register, extended to operate on words instead of bits. Writing $R_1, \ldots, R_n$ for the state registers, WAKE-ROFB's state update function is defined as

$$R_1' \leftarrow R_{n-1} + F(R_n); \qquad R_j \leftarrow R_{j-1}; \qquad R_1 \leftarrow R_1'.$$

Here $F : \mathbf{Z}_2^{32} \to \mathbf{Z}_2^{32}$ is a key-dependent nonlinear function. Every k-th time we step the register, we output the value of R_n as the next word of the key-stream. See Figure 4 for a pictorial illustration of the cipher.

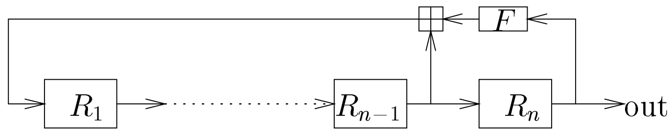

Fig. 4. The WAKE-ROFB stream cipher

The parameters k and n may be varied to suit performance and security needs. However, [3] suggests two concrete proposals: $(n, k) = (5, 8)$ and $(n, k) = (4, 4)$. For the $n = 5$ proposal, a concrete scheme for loading an initialization vector is proposed: the 64-bit IV (A, B) is loaded into the registers as $R_1 = R_4 = R_5 = A, R_2 = R_3 = B$, and then 8 words of output are generated and discarded. For the $n = 4$ proposal, no scheme for loading an IV was suggested.

Note that, to support re-synchronization, WAKE-ROFB is built around a mode of operation that is somewhat unusual for a stream cipher. Many stream cipher constructions use a public feedback function and load their initial state from the key. In contrast, WAKE-ROFB is keyed solely by the choice of the key-dependent function F, and the initial state of the register is loaded from a publicly-known IV[6]. Re-synchronization is easily accomplished by choosing a new IV.

The main observation is that this can be viewed as roughly an unbalanced Feistel cipher (with round function F) that outputs one word every k rounds. From this viewpoint, there is no round-dependence in the round transformation. Since Feistel ciphers with no round-dependence are susceptible to slide attacks, it seems natural to suspect that slide attacks may also prove useful against the WAKE-ROFB stream cipher. This is indeed the case.

First, we note that when the attacker has full control over the initial state of the stream cipher, it is often easy to break with a simple slide attack. The attack is the same as a chosen-plaintext slide attack on a Feistel cipher with constant round subkeys. We fix $r_1, \ldots, r_{n-1}$, and generate 2^{16} IV's of the form $IV_X = (r_1, \ldots, r_{n-1}, X)$ by varying X. We also generate 2^{16} IV's of the form $IV_Y = (Y, r_1, \ldots, r_{n-1})$ by varying Y. Note that if $r_{n-1} + F(X) = Y$, we will have a successful slide relation between the key-stream generated by IV_X and the key-stream generated by IV_Y. For such X, Y, the resulting internal states

[6] But note that slide attacks do not always require knowledge of the initial state of the register. For instance, some of our attacks would still be possible even if the construction were modified to load the initial state of the register as e.g. the Triple-DES-CBC decryption of the IV under some additional keying material.

will be closely related: if we let $S_\alpha[t] = (R_{1,\alpha}[t], \ldots, R_{n,\alpha}[t])$ be the $32n$-bit state generated from IV_α by stepping the cipher t times, then $S_Y[t] = S_X[t+1]$ for all t.

In many cases, this condition can be easily recognized, because the key-streams will be highly related to each other. For instance, for the $(n,k) = (4,4)$ proposal, if we know the key-stream outputs from IV_X at times $jk, (j+1)k$ and the key-stream output from IV_Y at time jk, we can deduce one input-output pair for the F function for each time step; this property allows us to easily recognize slid pairs with about 8 known outputs for the F proposed in [3][7]. Analysis is apparently more difficult when $\gcd(n,k) = 1$, but attacks are still available (albeit with increased data requirements) by choosing $n \cdot 2^{32}$ IV's of the form $(Y, \ldots, Y, r, \ldots, r)$; the crucial observation is that $(r, \ldots, r)$ forms a slid pair with $(F(r)+r, r, \ldots, r)$, which forms a slid pair with $(F(r)+r, F(r)+r, r, \ldots, r)$, and so on.

We conclude that a slide attack may be possible with as few as 2^{17} streams (each containing at least 8 known outputs), when the attacker has full control over the initial state of the register. This situation might occur if, for instance, the IV-loading mechanism simply loaded the initial state of the register directly as the value of a n-word IV, since then an attacker would be able to control the initial state directly with a chosen-IV chosen-ciphertext attack. One corollary is that the IV-loading mechanism must be carefully designed for WAKE-ROFB type stream ciphers.

Even when the attacker has no control over the initial state of the register, known-IV slide attacks may still be possible. By analogy to the standard known-text attacks on block ciphers, we expect to find one successful slide relation after examining about $2^{32n/2}$ known text streams, and in some cases this might enable successful cryptanalysis of the cipher. One defense is to increase the size of the internal state enough so that the data requirements become infeasible.

Finally, we consider the concrete IV-loading scheme proposed in [3] for the $(n,k) = (5,8)$ cipher. There the 64-bit IV (A,B) is loaded into the registers as $(R_1, \ldots, R_5) = (A, B, B, A, A)$, and then 8 words of output are generated and discarded.

We note that a slide attack on this scheme is still possible, when 2^{32} chosen-IV queries are available. We obtain known key-stream output for the 2^{32} IV's of the form (A, A). This loads the initial state of the registers with $(R_1, \ldots, R_5) = (A, \ldots, A)$. Note that when $F(A) = 0$, we will have $R_1' = A$, and so stepping the initial state $(A, \ldots, A)$ gives the state $(A, \ldots, A)$. In other words, for $A = F^{-1}(0)$, we obtain a cycle of period one. This can be easily recognized from a

[7] This is because [3] constructs the T table from two 4×16-bit lookup tables, and by the birthday paradox after 7 observations of a 4-bit value we expect to see a collision or two. But even for more sophisticated constructions of the F function, the number of known outputs needed would not increase substantially. With a randomly generated T table, about 40 known outputs would suffice; even if the entire function F were chosen randomly, $2^{16.5}$–$2^{17.5}$ known outputs should be enough to detect slid pairs.

short stretch of known key-stream output, and allows allows us to obtain 32 bits of information on the key.

It is clear that the design of a secure IV-loading mechanism for WAKE-ROFB-like stream ciphers is non-trivial. Certainly running the cipher for $8k$ time steps and discarding the outputs helps stop some attacks, but as we have shown, it is not always sufficient.

Therefore, we propose the following design principle for such stream ciphers:

Whenever possible, the feedback function should include some form of round-dependence.

7 Key-dependent S-boxes: A Variant of Blowfish

The following was inspired by a paper due to Grossman and Tucherman [5] from 1978. In this section we show by using a more modern techniques that if the only strength of a cipher comes from key-dependent S-boxes (with no round dependence) then such cipher can be attacked easily using slide attacks. This shows that slide attacks are not restricted to ciphers with weak key-scheduling algorithms.

For an example of how this might work consider a cipher called Blowfish, which was designed by Bruce Schneier [12]. This is a Feistel cipher with 64-bit block, 16 rounds and up to 448 bits of the secret key. These are expanded into a table consisting of four S-boxes from 8 to 32 bits (4096 bytes total). S-boxes are key-dependent and unknown to the attacker. Also in each round a 32-bit subkey P_i is XORed to one of the inputs. At the end two 32-bit subkeys P_{17} and P_{18} are XORed to the output of a cipher. See Figure 5 for a picture of one round of Blowfish. So far no attacks are known on a full version of this cipher. The best previous result [14] is a differential attack on Blowfish with known S-boxes which can find the P_i array using 2^{8r+1} chosen plaintexts, where r stands for the number of rounds. For certain weak keys that generate bad S-boxes (1 out 2^{14} keys) the same attack requires 2^{4r+1} chosen plaintexts (still completely ineffective against 16-round Blowfish).

Assume that all the P_i's are equal to zero. In this case one may notice that all rounds of a cipher perform the same transformation which is data-dependent. Thus given a 32-bit input to the F-function the output of the F function is uniquely determined. Also only 16 bytes out of 4096 take part in each evaluation of the F-function. Thus one naive approach will be to fix a plaintext P, guess all these 128-bits of the key and partially encrypt P with the guessed keys one Feistel-round, and then perform a slide attack for P and for the guessed text. A less naive approach is to guess the 32-bit output of the F-function and thus obtain a correct encryption with one Feistel round in 2^{32} steps, checking if the guess was correct with a usual sliding technique. An even better approach is to encrypt two pools of chosen plaintexts (X, P_R) and (P_R, Y), where X and Y both receive 2^{16} random values and P_R is fixed. Thus with high probability there is an element (P_R, Y_i) in the second pool which is an exact one-round encryption

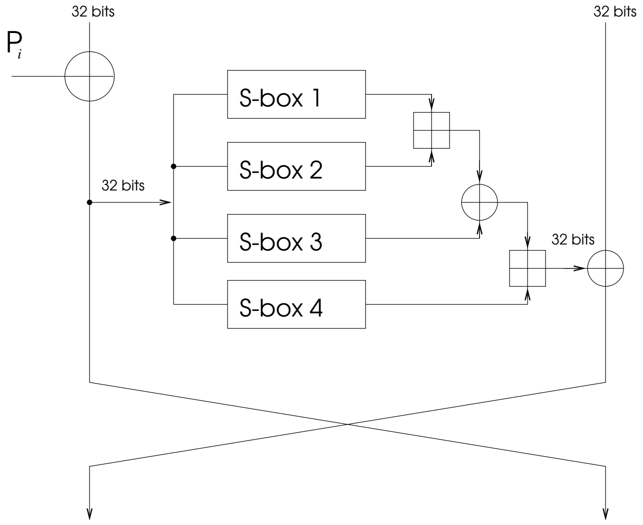

Fig. 5. One round of Blowfish.

of some element (X_j, P_R) from the first pool. Such pair can be easily detected by sliding considerations (especially if we repeat this experiment with the same value of P_R and other random values of X and Y). Each slid pair provides us with about 64 bits of key-dependent S-box information (two equations for F-function).

Thus with about 500 probes of this type it is possible to find all four S-boxes. Data can be packed into structures efficiently. Thus we have a very simple slide attack with only about $2^9 \cdot 2^{18} = 2^{27}$ chosen plaintexts on this variant of Blowfish.

This attack is independent of the number of rounds of a cipher be it 16 or 16000 rounds, of the exact structure of the F-function, and of the key-schedule, no matter how complex is the S-box generation process [8]. This shows that slide attacks are not restricted to ciphers with weak key-scheduling.

8 Acknowledgments

Many thanks to Bruce Schneier for suggesting the name "slide attack", and for encouragement to write up these results. Also, we are indebted to Craig Clapp and to Bruce Schneier for assistance in obtaining solid information about the M6 algorithm. Finally, we are grateful to the FSE6 program committee for detailed comments on the paper.

[8] Notice also, that it is possible to find a 448-bit key which will force $P_1, \ldots, P_{14}$ to be zero; however, $P_{15}, .., P_{18}$ will remain uncontrolled.

References

1. E. Biham, A. Shamir, *Differential Cryptanalysis of the Data Encryption Standard*, Springer-Verlag, 1993.
2. E. Biham, *New Types of Cryptanalytic Attacks Using Related Keys*, J. of Cryptology, Vol.7, pp.229–246, 1994.
3. C. Clapp, *Joint Hardware / Software Design of a Fast Stream Cipher*, Fast Software Encryption'98, pp.75–92, 1998.
4. J. Daemen, R. Govaerts, J. Vanderwalle, *Re-synchronization weaknesses in stream ciphers*, EUROCRYPT'93, pp.159–169, 1994.
5. E. K. Grossman, B. Tucherman, *Analysis of a Weakened Feistel-like Cipher*, 1978 International Conference on Communications, Alger Press Limited, 1978, pp.46.3.1–46.3.5.
6. G. Yuval, *Reinventing the Travois: Encryption/ MAC in 30 ROM Bytes*, LNCS'1267, Fast Software Encryption'97, pp.205–209, 1997.
7. G. Yuval, *Private communication, August 1998.*
8. J. Kelsey, B. Schneier, D. Wagner, *Key-Schedule Cryptanalysis of IDEA, G-DES, GOST, SAFER, and Triple-DES*, CRYPTO'96, pp.237–251, 1996.
9. J. Kelsey, B. Schneier, D. Wagner, *Related-Key Cryptanalysis of 3-WAY, Biham-DES, CAST, DES-X, New DES, RC2, and TEA*, Proceedings of the 1997 International Conference on Information and Communications Security, Beijing.
10. L. R. Knudsen, *Cryptanalysis of LOKI91*, proceedings of AUSCRYPT'92, LNCS 718, pp.196–208, 1993.
11. M. Matsui, *Linear Cryptanalysis Method of DES Cipher*, Lecture Notes in Computer Science 765, Advances in Cryptology – EUROCRYPT'93, pp.386–397, Springer-Verlag, 1994.
12. B. Schneier, *Description of a New Variable-Length Key, 64-Bit Block Cipher (Blowfish)*, proceedings of FSE'94, pp.191–204, 1994.
13. *Content Protection for Digital Transmission System*, 5C compromise proposal version 0.91, 17-Feb-1998, Hitachi, Intel, Matsushita, Sony, and Toshiba.
Editors: Brendan Traw (Intel) `brendan_traw@ccm.jf.intel.com` and Scott Smyers (Sony) `scotts@lsi.sel.sony.com`.
14. S. Vaudenay, *On the Weak Keys in Blowfish*, proceedings of FSE'96, pp.27–32, 1996.
15. D. Wagner, *Cryptanalysis of some recently-proposed multiple modes of operation*, proceedings of FSE'98, pp.254–269, 1998.
16. *Response for Data Protection System for Digital Transmission of Copy Protected Information*, Version 0.99, pp. 8–12, Hitachi, Matsushita, and Sony.

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