Cryptanalysis of FROG

David Wagner* Niels Ferguson† Bruce Schneier‡

March 9, 1999

Abstract

We examine some attacks on the FROG cipher. First we give a differential attack which uses about 2^{58} chosen plaintexts and very little time for the analysis; it works for about $2^{-33.0}$ of the keyspace. Then we describe a linear attack which uses 2^{56} known texts and works for $2^{-31.8}$ of the keyspace. The linear attack can also be converted to a ciphertext-only attack using 2^{64} known ciphertexts. Also, the decryption function of FROG is a lot weaker than the encryption function. We show a differential attack on the decryption function that requires 2^{36} chosen ciphertexts and works on $2^{-29.3}$ of the keyspace. Using our best attack an attacker with a sufficient number of cryptanalytical targets can expect to recover his first key after $2^{56.7}$ work.

Taken together, these observations suggest that FROG is not a very strong candidate for the AES.

1 Introduction

FROG [3] is a block cipher submitted to the AES competition with a novel internal structure. It uses 8 cycles, where each cycle consists of 16 rounds. Round r ($r = 0, \ldots, 15$) of the q-th cycle ($q = 0, \ldots, 7$) modifies the 16 bytes $X_{0\ldots15}$ of the internal block according to the three (sequential) operations

$$\begin{aligned} X_r &\leftarrow S_q(X_r \oplus K_{q,r}) \\ X_{r+1} &\leftarrow X_{r+1} \oplus X_r \\ X_{\pi_q(r)} &\leftarrow X_{\pi_q(r)} \oplus X_r, \end{aligned}$$

*U.C. Berkeley
†Counterpane Systems
‡Counterpane Systems

where the indices are to be taken modulo 16. Here the round subkeys consist of a bijective byte-wide S-box S_q, a so-called "bomb" permutation π_q of the symbols $\{0, 1, \ldots, 15\}$, and an array of subkey bytes $K_{q,r}$. In all, the cipher contains 128 rounds, and so the key schedule expands a k-bit master key ($k = 128, 192, 256$) into a large amount of subkey material.

The FROG expanded key has a redundancy in it. The input to the S-box consists of a data byte XORed with a key byte. If we take any value u we can XOR each of the key bytes with u and update the S-box to include an extra XOR with u at the input. This results in an equivalent expanded key. During an attack we can just set one key byte or one S-box entry in each round to an arbitrary value without loss of generality.

One aspect of FROG that complicates the analysis is the key-dependent nature of the internal structure: the quality of internal diffusion depends heavily on the (key-dependent) choice of the "bomb" permutations π_q. We deal with this issue by characterizing every attack with three parameters. The first one is the fraction of keys F for which the attack is successful. The second parameter is the complexity D of detecting if the key is within that fraction. The third parameter is the complexity C of the rest of the attack. Often C, D, and F can be traded off against each other.

This brings up the question how to compare attacks with different values of C, D, and F. We assume that the attacker has a ready supply of encryption black boxes with different keys, and use the amount of work an attacker has to do before finding the first key as a measure of the quality of the attack. The attacker has to about D/F operations to find a key that is weak plus C operations to recover the key which gives us the total work $D/F + C$. Normally the expression $D/F + C$ is either dominated by D/F (when finding a suitable weak key is the biggest problem) or by C. It is not worth considering attacks which have $D/F > 2^k$ or $C > 2^k$ where k is the key size as these are less useful than a simple exhaustive search.

In this paper, we show that diffusion failures in FROG can lead to real attacks on the full cipher. The key-dependent diffusion structure of FROG is in this respect a weakness as there is a fraction of the keys for which diffusion is weak. In particular, Section 2 describes several useful differential characteristics for FROG, and in Section 3 we give a simple differential attack which needs about 2^{58} chosen plaintexts and very little time for the analysis. The attack works for about $2^{-33.0}$ of the keyspace. Section 4 examines the application of linear cryptanalysis to FROG, showing how to break FROG with about 2^{56} known texts for about $2^{-31.8}$ of the keyspace. Section 5 extends those results to a ciphertext-only setting. In Section 6 we give a differential attack against the decryption function that requires 2^{36}

chosen plaintexts and works for $2^{-29.3}$ of the keyspace. Finally, Section 8 discusses FROG's performance as compared with other block ciphers and its suitability as an AES candidate.

2 Differential characteristics

Suppose the differential characteristic $a \to a$ holds with probability p_q for the S-box S_q, where $a \neq 0$ is an arbitrary byte-wide difference. Set

$$\beta_0 = (a, a, 0, \ldots, 0)$$

and let β_j be the result of rotating β_0 to right by j byte positions. Then the simple differential characteristic $\beta_1 \to \beta_0$ can be used to approximate one cycle of FROG with probability p_q when the key is favorable. We can classify the favorable keys as those where $\pi_q(1) = 0$, so it follows that $1/15$ of the keyspace is favorable.[1]

We can also obtain 13 more characteristics of a similar form. Namely, $\beta_{j+1} \to \beta_j$ also holds (for $j = 0, \ldots, 13$) with the same probability, when $\pi_q(j+1) = j$. Of course, these characteristics can be pieced together nicely. This provides an easy way to build a n-cycle differential characteristic $\beta_{j+n} \to \beta_j$ with probability $\prod_q p_q$; the characteristic will work for $1/15^n$ of the keyspace.

A useful truncated differential characteristic is $\alpha \to \beta_{13}$, where

$$\alpha = (0, \ldots, 0, b, a)$$

Here b may be any non-zero byte difference. For each a, this holds with average probability 2^{-8} for $1/15$ of the keyspace. The advantage of using this characteristic is that it enables us to bypass the first cycle by using structures.

These characteristics can be concatenated to obtain the differential characteristic $\alpha \to \beta_6$ for the whole 8-cycle cipher. This characteristic will hold with probability $p = 2^{-8} \cdot p_1 \cdot \ldots \cdot p_7$ for $1/15^8 \approx 2^{-31.3}$ of the keyspace. We shall describe in the next section how to use this to break FROG in a 0-R attack.

But first, we focus on analyzing the probability p of this characteristic. The propagation probability p is dependent on the key and on the choice

[1] The way in which π_q is generated results in a distribution very close to $\Pr(\pi_q(x) \in \{x, x+1\}) = 0$, $\Pr(\pi_q(x) = x+2) = 2/15$, and $\Pr(\pi_q(x) = y) = 1/15$ for all other values of y.

of the byte difference a, so we must resort to probabilistic approximations. First of all, modeling S_q as a random permutation suggests that the distribution of p_q is likely to be Poisson with parameter 1/2. In other words,

$$\Pr[p_q = \frac{2j}{256}] = \frac{e^{-1/2}\, 2^{-j}}{j!}$$

Since we need $p > 0$, we require $p_r > 0$ for $r = 1, \ldots, 7$. Now $\Pr[p_q > 0] = 1 - e^{-1/2}$, so $\Pr[p > 0] = (1 - e^{-1/2})^7 \approx 0.00146$, under the heuristic assumption that the round subkeys behave as though they were chosen independently. There are 255 choices for a, so heuristically we expect that $p > 0$ for at least one of them with probability $1 - (1 - 0.00146)^{255} \approx 2^{-1.7}$. Again assuming the independence of S_q and π_q, this suggests that about $2^{-1.7}/15^8 \approx 2^{-33.0}$ of the keyspace is favorable. A key is said to be favorable if there is some a such that $\pi_q(15 - q) = 14 - q$ (for $q = 0, \ldots, 7$) and $p_q > 0$ (for $q = 1, \ldots, 7$). For a favorable key and a suitable a, the probability of the differential $\alpha \rightarrow \beta_6$ is at least $2^{-8} \cdot (2/256)^7 = 2^{-57}$.

Summarizing, we have found that the 8-cycle characteristic $\alpha \rightarrow \beta_6$ is expected to hold with probability at least 2^{-57} for $2^{-33.0}$ of the keyspace.

3 The attack

In the first phase of our attack on FROG, we search for a value of a such that the 8-cycle characteristic has probability $p \geq 2^{-57}$. We use a total of 2^{65} plaintext pairs, or 2^{57} pairs for each of the 256 possibilities for a. The required pairs can be generated with 2^{50} chosen plaintext queries by using structures.

Very efficient filtering is possible, since we can eliminate all but those ciphertexts with difference β_6. We should be very surprised if we see even one wrong pair.

As a result, when the key is favorable, we expect to be able to identify the useful value of a where $\alpha \rightarrow \beta_6$. (If the key is unfavorable, we can give up at this point.) For the remainder of the attack, we restrict our attention to pairs with this value of a.

In the second phase of our attack, we use knowledge of this value for a to generate 2^{65} more plaintext pairs with this value of a. By using structures, we can form the necessary pairs with about 2^{58} more chosen plaintext queries.

We expect to find about 256 right pairs where the characteristic is followed. These right pairs can be used to deduce the contents of the inverse of the last-cycle S-box S_7^{-1} with some simple linear algebra. We may treat the 256 entries of S_7^{-1} as 256 formal unknowns. Then each pair of ciphertexts C, C' with $C \oplus C' = \beta_6$ gives us one linear equation on the entries of S_7:

$$S_7^{-1}(C_7) \oplus S_7^{-1}(C_7') = a.$$

With 256 right pairs, we obtain 256 linear equations on 256 unknowns, which is enough to solve for S_7^{-1} nearly uniquely (up to an unknown XOR at the output of the S-box). We can ignore the remaining XOR freedom at the input since that only selects between equivalent expanded keys. Of course, this gives us most of the entries of S_7 by inversion[2].

Actually, there is a complication. If $\pi_7(j) = 7$ for some $j > 7$, then the previous approach will not work. However, a simple modification usually will. If the previous technique fails, we note that

$$S_7^{-1}(C_7 \oplus C_j) \oplus S_7^{-1}(C_7' \oplus C_j) = a,$$

and try the linear algebra approach again eight more times for each of $j = 8, \ldots, 15$. With this modification, S_7 can be recovered with excellent probability.

In this way we can recover S_7 with very little work. At this point, we may continue by peeling off the outer cycles (making a few guesses where necessary) and repeating the attack, though a bit of care is required to make this work.

Using out terminology from section 1 this attack has $D = 2^{50}$, $F = 2^{-33.0}$ and $C = 2^{58}$. We thus expect that an attacker will recover a key after 2^{83} operations.

As stated, this is an adaptive chosen plaintext attack. However, the adaptivity may be easily removed by requesting all 2^{58} chosen plaintexts in advance. The only reason we stated the attack in its adaptive form is that the chosen text complexity is somewhat reduced when the key is not favorable.

Besides the differential $\alpha \to \beta_6$ there are six more differentials with the same properties. These are constructed by rotating the differential $\alpha \to \beta_6$ between 1 and 6 bytes positions to the left. Using structures we can run the attack for all seven differentials using the same plaintexts. This gives an improvement of a factor of 7 on the number of favorable keys without an increase in work.

[2]If necessary, we may continue to eliminate the few remaining gaps in our knowledge of S_7 by analyzing a few of the wrong pairs where the characteristic held in the first seven cycles but failed in the last one.

j	F	j	F	j	F	j	F	j	F
0	0.076	3	0.060	6	0.044	9	0.027	12	0.011
1	0.071	4	0.055	7	0.038	10	0.022	13	0.005
2	0.065	5	0.049	8	0.033	11	0.016	14	0.000

Table 1: Fraction F of the keyspace where $\Gamma_{j+1} \to \Gamma_j$ holds

4 Linear cryptanalysis

FROG also is susceptible to linear cryptanalysis. The linear attacks are not as clean or elegant as the differential attacks, but they allow one to relax the chosen-plaintext assumption required by a differential attack.

Suppose that $a \to a$ with bias b_q by the S-box S_q, where the bias is defined as

$$b_q = 4\,|\Pr[S_q(x \cdot a) = x \cdot a] - 1/2|^2 .$$

Take $\Gamma_0 = (a, 0, \ldots, 0)$ and let Γ_j be the result of rotating Γ_0 right j positions.

Then $\Gamma_{j+1} \to \Gamma_j$ (for $j = 1, \ldots, 15$) forms a useful one-cycle linear characteristic with bias b_q. It is expected to work on between 1/15 and 1/30 of the keyspace, with the exact fraction depending on j; see Table 1 for empirical results.

A useful linear characteristic for the last cycle is $\Gamma_1 \to \Gamma'$, where $\Gamma'_0 = a$, $\Gamma'_1 = c$, $\Gamma'_{\pi_7^{-1}(0)} = a$, $\Gamma'_{\pi_7^{-1}(1)} = c$, and Γ' is zero elsewhere. Suppose $a \to c$ with bias b_7 by S_7; then the characteristic $\Gamma_1 \to \Gamma'$ for the last cycle has bias b_7 and holds for about 21% of the keyspace (according to empirical tests).

We can combine these one-cycle characteristic to obtain a eight-cycle linear characteristic for the whole cipher of the form $\Gamma_8 \to \Gamma'$. It will hold for $0.038 \cdot 0.044 \cdot \ldots \cdot 0.071 \cdot 0.21 \approx 2^{-31.8}$ of the keyspace. For a single fixed value of a, c, the bias will be $b(a, c) = b_0 \cdot \ldots \cdot b_7$.

The technique of multiple linear approximations [6] may be applied with some success here. We sum over all values a and c; according to [6], the equivalent bias for the multiple linear approximation will be approximately $B = \sum_{a,c} b(a, c)$. The multiple linear approximation involves eight bits of key material, namely $K_{0,8} \oplus \ldots \oplus K_{7,1}$, so we will need to guess all possible values for those eight key bits. We will also need to guess $\pi_7^{-1}(0)$ and $\pi_7^{-1}(1)$ so that we know which form of Γ' to use. According to Matsui's rule of thumb, we expect to need about $N = 32/B$ known texts to have a good chance of success.

The number N of texts needed may be estimated empirically with probabilistic methods. In our experiments, we found that N has expected value $E[N] \approx 2^{54.4}$. The number of texts needed is often not too much more than the expected value, and quite frequently is substantially less: for instance, $\Pr[N \leq 2^{48}] \approx 2^{-7}$ and $\Pr[N \leq 2^{51}] \approx 0.12$, while $\Pr[N > 2^{57}] < 2^{-10}$.

The attack proceeds as follows. We obtain about 2^{56} known plaintexts. Using the technique of multiple linear approximations, we expect to recover $K_{0,8} \oplus \ldots \oplus X_{7,1}$ with very good probability if the key is favorable. (If the key is unfavorable, that will also be detected, and we may halt the attack at once.)

Next, we attempt to derive information about the S-box S_0 in a second phase of the attack. We repeat the following procedure for each of the 256 possible inputs x to S_0. Without loss of generality we set $K_{0,8} = 0$. To learn the S-box entry $S_0(x)$, we restrict our attention temporarily to those plaintexts P where $P_8 = x$, so that the input to S_0 is x in the eighth round of the first cycle[3]. We then guess $S_0(x)$ and use the linear characteristic $\Gamma_7 \rightarrow \Gamma'$ for the last seven cycles to verify our guess at $S_0(x)$. Based on our simulations, we expect that only about 2^{43} texts are needed to find $S_0(x)$; since $2^{56}/2^8 = 2^{48}$ known texts should be available for each x, the second phase of the attack should succeed with very good probability. At the end of the second phase, we expect to have recovered S_0.

With similar techniques (and a bit more work), we can recover the entries of the S-box S_7 used in the last cycle. At this point, it will be helpful that we derived the values of $\pi_7^{-1}(0)$ and $\pi_7^{-1}(1)$ earlier in the attack, since that will help us to identify the output of S_7 in the first and second rounds of the last cycle.

Once S_0 and S_7 are known, we may peel off the outer cycles and repeat the attack iteratively to recover the remainder of the key. In practice, peeling off the last cycle is expected to be easier, so that is what we recommend.

Summarizing, our linear attack uses about 2^{56} known plaintexts, and is expected to work for about $2^{-31.8}$ of the keyspace. The time complexity is expected to be small compared to the number of texts needed in the attack.

This linear attack is only the result of a preliminary investigation, and it may be possible to improve it significantly. For example, one simple avenue

[3] Actually, there is a complication. This can fail when $\pi_0^{-1}(8) < 8$, which occurs with probability 0.57. However, the failure will be detected, and we may try each of the eight possibilities for $\pi_0^{-1}(8)$ in this case. Then $P_8 \oplus P_{\pi_0^{-1}(8)} = x$ implies that the input to S_0 in round 8 is x (with good probability), so we may still separate out the plaintexts as needed. Of course, we only need to search for $\pi_0^{-1}(8)$ once, and then it will be known for all the other values of x, so this should not be a significant burden in practice.

for improvement is to repeat the attack with other linear characteristics, such as rotated versions of $\Gamma_8 \to \Gamma'$; it should be possible to break a somewhat larger class of weak keys in this way. Alternatively, one could reduce the number of known texts needed at the cost of some reduction in the number of favorable keys. As another example, we suspect that it may be fruitful to mount a systematic search for other linear approximations; the ones given here were the result of a limited search by hand.

5 A ciphertext-only attack

The linear attack can also be generalized to a ciphertext-only attack. In the ciphertext-only attack, we assume that the plaintext is formed of ASCII text, so that the high bit of each byte is always zero. This is expected to be a relatively realistic model in practice.

Only slight modifications to the linear attack are needed to work in this model. Instead of considering all of the 2^{16} linear characteristics that result from allowing a, c to take on all possible values, we now fix $a = 128 =$ `0x80`, so that the mask Γ_8 applied to the plaintext selects the high bit from the byte in position 8. We expect the overall bias to decrease by a factor of 2^8, so that the number of texts needed increases to about 2^{64}.

The analysis phase requires only cosmetic changes. It will no longer be possible to recover all of S_0 in the same way; in particular, we can only learn the high bit of each S-box entry. However, since we still allow c to vary over all 255 possibilities, we do expect to be able to recover S_7 as before. In this way, we can repeat the attack iteratively until we have recovered the entire key.

In summary, the ciphertext-only attack is expected to require about 2^{64} known ciphertexts on average and comparable time complexity. The ciphertext-only attack is applicable not only to ECB mode, but also to some chaining modes, including CBC mode.

6 Decryption

We now turn to cryptanalysis of the decryption. FROG exhibits surprisingly poor diffusion behavior in the reverse direction. In other words, if we decrypt two ciphertexts which differ in only one byte position, it will often take many (48–64) rounds before full avalanche is achieved. In contrast, in the forward direction, avalanche is expected to be achieved relatively quickly (about 16–24 rounds).

Ciphers with an asymmetrical internal structure (e.g., unbalanced Feistel networks [9]) often require extra care to avoid this sort of pitfall. Several other recent ciphers contain special precautions to avoid weaknesses in the reverse direction. As a notable example, Skipjack [7, 10] alternates between one round structure (Rule A) and its inverse (Rule B) to avoid asymmetries between encryption and decryption. Additionally, the MARS submission [2] explicitly steers clear of this pitfall; the CAST-256 submission [1] also manages to avoid this pitfall; and an early design considered by the Twofish team was rejected due to asymmetry concerns.

We present a variation on our differential attack that works on the decryption function of FROG. Observe that if $A \to B$ is a differential characteristic of function f with probability p, then $B \to A$ is a differential of f^{-1} with the same probability. (This is easily seen when looking at the set of all pairs of plaintexts and ciphertexts.) We will use 5 cycles of the differential used in section 2. This gives us the differential $\beta_0 \to \beta_5$ after 5 cycles. In the sixth cycle we assume that $\pi_5(6) = 5$. This gives us the 6-cycle differential of

$$\beta_0 \to (0, 0, 0, 0, 0, 0, X, a, 0, \ldots, 0)$$

where X represents an arbitrary value. We now hope for a favorable propagation through the last two cycles. This propagation depends only on the values of π_6 and π_7. To make the attack easy we look for cases where the input differential of S_7 in the first byte is always a. This is the very last S-box computation of the decryption. We also need to have some redundancy for filtering. Every output byte with a difference of 0 or a provides redundancy that is useful for filtering. We say that the pair (π_6, π_7) is favorable if it provides at least 6 bytes of redundancy and the input difference a to the last S_7 computation.

The fraction of favorable keys can be determined as follows. For the first 6 cycles we require $\pi_q(x) = x - 1$ for some x. This holds for $1/15^6$ of all keys. We ran empirical tests on the last two permutation choices. We generated 2^{20} pairs of (π_6, π_7) and found that 0.031 of all pairs were favorable. Together this implies that about $2^{-28.5}$ of the keys are favorable.

For a favorable key, the probability of getting a useful pair through the entire cipher is determined by the S-boxes. We need the differential $a \to a$ for the $S_0, \ldots, S_4$. For any a the probability of this differential having a positive probability for all these boxes is $(1 - e^{-1/2})^5 \approx 0.00943$. The chance of this happening for at least one a is over 90% (for simplicity we ignore this factor in our analysis). The expected number of a for which the approximation has a positive probability is more than 2. For such an a, the probability of the differential is at least $(2/256)^5 = 2^{-35}$.

The attack now works as follows. Using structures we generate suitable input differentials and filter the output differences using the redundancy. This requires about 2^{37} chosen ciphertexts to generate 2^{36} pairs for each value of a. As there are on average more than 2 suitable values for a we expect to find two right pairs for each such useful value of a.

Identifying the right pairs and thus the useful values of a is not difficult. We look for an output pair with at least 6 bytes that have difference 0 or a. For each useful value of a, two right pairs will have the same redundancy pattern. For a wrong value of a, we expect to see about $2^{36} \cdot \binom{16}{6} \cdot 2^6/2^{48} \approx 2^7$ wrong pairs; according to the following calculation, they should all fall into different redundancy patterns. By the birthday paradox, the chance of seeing two wrong pairs with the same redundancy pattern is $1 - \exp -2^{7\cdot 2}/(2 \cdot \binom{16}{6} \cdot 2^6) \approx 0.016$. Thus, over all 255 values of a, we expect about $255 \cdot 0.016 \approx 4$ wrong values of a (in addition to the two right values of a) which contain at least two pairs falling in the same redundancy pattern. Then the two right values of a can be recognized because they both induce the same redundancy pattern.

From these right pairs we learn a suitable value for a and the pattern of the redundancy. Unfortunately there is a single input difference we are now interested in, so we cannot use structures for the remaining of the attack. We generate 2^{42} more pairs using 2^{43} chosen inputs. Given the known output redundancy pattern these can be filtered very easily. This gives us about 256 output pairs where the input difference to the last S-box computation was the known value a. We recover S_7 in the same way as the earlier differential attack.

We conclude that the FROG decryption has an attack with $F = 2^{-28.5}$, $D = 2^{37}$ and $C = 2^{43}$. Thus an attacker can be expected to recover a key after about $2^{65.5}$ work.

There are some obvious extensions. Instead of $\beta_0 \to \beta_5$ we can start with any of the shifts that don't wrap around. Furthermore, we can use additional differentials for the first round. For example $(a, 0, a, 0, \ldots, 0) \to \beta_2$. The further the input pattern shifts to the right the more differentials can be used for the first round.

We ran our analysis of the permutations in the last two rounds for each of the shifted versions. The overall result is that this attack can be made to work with more or less the same complexity against about 2^{-24} of all keys. We have not investigated the details of how attacks against multiple cases can be combined using a single input structure, but we expect that this will lead to an improvement of about an order of magnitude.

There are several more improvements that can be made to this attack. We can reduce the number of cycles for which we use the differential, and leave the last 3 cycles free of restrictions. There is a large enough fraction of the keys that produce the right kind of input difference to the last S-box computation, but the avalanche is strong enough to limit the very simple filtering rules that we use to all but a small fraction of the keys. Using 5 bytes for filtering, we get a differential with probability 2^{-28} for $2^{-29.3}$ of all keys. We now expect that there are about 6 values of a for which the differential $a \rightarrow a$ has a nonzero probability in the first four S-boxes. Thus, using $2^{27.4}$ ciphertexts we can generate $2^{26.4}$ pairs for each a. As we expect 6 suitable values of a we now have a good chance of finding two right pairs. The two right pairs are easily recognized, as they have the same redundancy pattern; in comparison, only about $2^{7.9}$ wrong pairs are expected, which by the birthday paradox stands only a 18% chance of generating any false alarms. This improves the attack to $D = 2^{27.4}$, $F = 2^{-29.3}$ and $C = 2^{36}$, which means that the attacker can expect to recover his first key after $2^{56.7}$ work. Again there are several similar differentials; together they cover about $2^{-26.6}$ of the keyspace.

A better filtering method, or a better way of solving for S_7 could improve the overall attack greatly. If we drop the requirement of having at least 5 filtering bytes, the fraction of useable keys goes up to $2^{-22.6}$ for the most likely position, and the whole family of differentials cover up to 2^{-18} of the keyspace.

Another interesting point to look at is the first decryption cycle; maybe there are other differentials that we can use. This could make our structures more efficient, and thus reduce the complexity of the attack.

7 Further work

We have not implemented any of our attacks. Practical demonstrations of all of our attacks can be given by applying the attack against keys that are known to be favorable.

These attacks are the result of a preliminary analysis of FROG. A more thorough analysis will no doubt turn up better attacks.

8 Conclusions

As a result of a preliminary analysis of the FROG cipher, we have found several attacks that allow one to recover the key significantly more quickly than brute force, for a small portion of the keyspace. One of these attacks even works under a ciphertext-only model. This indicates that FROG has a significant problem with weak keys.

The internal diffusion structure of FROG is weak, especially in the decryption direction. One would probably have to double the number of rounds to eliminate the attacks described here.

Furthermore, the performance of FROG is significantly slower than many of the other AES contenders. One back-of-the-envelope estimate [12] suggests that FROG requires at least 48 clocks/byte (as a theoretical minimum) on a Pentium, and probably closer to about 70 clocks/byte in practice due to potential instruction pairing problems, for hand-tuned assembly-language implementations. This is faster than triple-DES (at about 120 clocks/byte), but much slower than some other modern ciphers such as Blowfish, Square, and RC5, which operate at 20–25 clocks/byte [5, 4, 11, 12]. Even DES (at 43 clocks/byte) is faster than FROG. Increasing the number of rounds by a factor of 2 to counteract our attacks would only widen the gap.

Finally, FROG seems ill-suited for smartcards, since it requires at least 2500 bytes of RAM for the keying material [3], as opposed to about a tenth of that for other AES submissions (e.g., Twofish [8]). Smart card suitability is likely to become extremely important for any general-purpose encryption algorithm.

In our opinion, this is enough to suggest that FROG is not a very strong candidate for the AES.

9 Acknowledgements

We are very grateful to Doug Whiting for his help with performance estimates. Of course, the authors are responsible for any errors in this document.

We would also like to thank Don Coppersmith for pointing out the importance of symmetry between encryption and decryption.

References

[1] C. Adams, "The CAST-256 Encryption Algorithm," AES submission, 1998.

[2] C. Burwick, D. Coppersmith, E. D'Avignon, R. Gennaro, S. Halevi, C. Jutla, S. Matyas Jr., L. O'Connor, M. Peyravian, D. Safford, and N. Zunic, "MARS — A Candidate Cipher for AES," AES submission, 1998.

[3] D. Georgoudis, D. Leroux, and B.S. Chaves, "The 'FROG' Encryption Algorithm," AES submission, 1998.

[4] C. Hall, J. Kelsey, V. Rijmen, B. Schneier, and D. Wagner, "Cryptanalysis of SPEED," *Proceedings of SAC 98*, Springer-Verlag, to appear.

[5] C. Hall, J. Kelsey, B. Schneier, and D. Wagner, "Cryptanalysis of SPEED," *Financial Cryptography '98 Proceedings*, Springer-Verlag, 1998, to appear.

[6] B. Kaliski Jr., and M. Robshaw, "Linear Cryptanalysis Using Multiple Approximations," *Advances in Cryptology — CRYPTO '94 Proceedings*, Springer-Verlag, 1994, pp. 26–39.

[7] National Security Agency, "Skipjack and KEA algorithm specifications," May 1998. `http://csrc.ncsl.nist.gov/encryption/skipjack-1.pdf`

[8] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Fergusen, "Twofish: A 128-Bit Block Cipher," AES Submission, 1998.

[9] B. Schneier and J. Kelsey, "Unbalanced Feistel Networks and Block Cipher Design," *Fast Software Encryption, 3rd International Workshop Proceedings*, Springer-Verlag, 1996, pp. 121–144.

[10] National Security Agency, "NSA Releases Fortezza Algorithms," Press Release, June 24, 1998. `http://csrc.ncsl.nist.gov/encryption/nsa-press.pdf`

[11] B. Schneier and D. Whiting, "Fast Software Encryption: Designing Encryption Algorithms for Optimal Speed on the Intel Pentium Processor," *Fast Software Encryption, 4th International Workshop Proceedings*, Springer-Verlag, 1997, pp. 242–259.

[12] D. Whiting, personal communications, July 30 1998.