Twofish Technical Report #6

A Twofish Retreat: Related-Key Attacks Against Reduced-Round Twofish

Niels Ferguson[*] John Kelsey[†] Bruce Schneier[‡] Doug Whiting[§]

February 14, 2000

Abstract

The Twofish AES submission document contains a partial chosen-key and a related-key attack against ten rounds of Twofish without whitening, using 256-bit keys. This attack does not work; it makes use of a postulated class of weak key pairs which has the S-box keys and eight successive round keys equal, but no such pairs exist. In this report we analyze the occurrence of this kind of weak key pair and describe how such pairs may be used both to mount attacks on reduced-round Twofish and to find properties of reduced-round Twofish that are not present in an ideal cipher. We find that related-key and chosen-key attacks are considerably less powerful against Twofish than was previously believed.

Keywords: Twofish, cryptography, cryptanalysis, related-key, block cipher, AES.

Current web site: `http://www.counterpane.com/twofish.html`

1 Introduction

In the Twofish paper [SKW$^+$98]—written as part of the AES process—and in the Twofish book [SKW$^+$99], we described a related-key attack against 9 rounds of Twofish with whitening, or 10 rounds without whitening, when using 256-bit keys.

The basic idea of the attack is that it is possible to find two keys, K and K^*, which produce the same set of S-boxes (and thus the same S) *and* the same set of round keys K_i for a few rounds. We call such a key pair a weak key pair, and use n to denote the number of equal round subkeys. For the rounds in question, the two keys implement the same function. An attacker can use this fact to distinguish the reduced-round cipher from a perfect cipher, or attempt to mount a related-key attack that recovers the key.

Our original analysis of this type of attack against Twofish was incomplete, and the results reported in [SKW$^+$98, SKW$^+$99] are erroneous. In this report, we investigate this attack in more detail.

We will use the notation from [SKW$^+$98, SKW$^+$99]; readers not familiar with the notation should consult one of these references.

1.1 Summary of Our Results

Our new results are as follows:

[*] Counterpane Internet Security, Inc., 3031 Tisch Way, 100 Plaza East, San Jose, CA 95128, USA; `niels@counterpane.com`

[†] Counterpane Internet Security, Inc. `kelsey@counterpane.com`

[‡] Counterpane Internet Security, Inc. `schneier@counterpane.com`

[§] Hi/fn, Inc., 5973 Avenida Encinas Suite 110, Carlsbad, CA 92008, USA; `dwhiting@hifn.com`

1. We have determined that there is no pair of keys which yield $n = 6$ successive identical round subkeys while also yielding identical S-boxes. This implies that the originally postulated case of $n = 8$ identical round subkeys is also impossible.

2. We show that it is unlikely that there is a pair of keys which yield $n = 5$ successive identical round subkeys while also yielding identical S-boxes, although we have not yet been able to verify this exhaustively.

3. Weak key pairs occur in groups of 2^{72} key pairs all with the same XOR difference. Any related-key attack based on these weak key pairs must be able to distinguish a weak key pair from a non-weak pair in less then 2^{71} steps.

   We are able to detect such weak key pairs with less than 2^{71} work for:

   (a) Up to $n + 3$ rounds without whitening. (Thus, if $n = 5$, this would result in an eight round attack.)

   (b) Up to $n + 1$ rounds with whitening. (Thus, if $n = 5$, this would result in a six round attack.)

4. Reduced-round Twofish with $n + 2$ rounds (with whitening) has some undesirable properties which might lead to an attack under some circumstances, although we have been unable to find such a circumstance.

We emphasize that the case $n = 5$ is improbable to exist at all, and thus that our current best attacks of this type is against 7 rounds without whitening and 6 rounds with whitening. We also note that we use more powerful attacks on these weak key pairs, especially on the reduced-round Twofish variants without whitening, than appeared in the original paper [SKW$^+$98, SKW$^+$99].

These new result represents a "retreat" from the related-key attack we believed we had previously; hence the name of the paper.

1.2 Implications of the Results

Twofish is substantially less susceptible to related-key attacks of the kind considered here than we previously suspected. The way the key material is used to derive the round subkeys interacts with the RS code used to derive the S-box keys in a way that makes this kind of related-key attack impossible for more than a small number of rounds. This interaction has important implications for attacks on hashing modes of Twofish, which appear to be extremely difficult even when the attacker has full control over the key.

In more theoretical terms, these results demonstrate the advantages of the reuse of the same key material in two "orthogonal" ways within the structure of the cipher. We note, however, that the Twofish key schedule is required to do two orthogonal things with the same key material, due to the cipher's key-dependent S-boxes. This kind of design may not be reasonable for other ciphers.

1.3 Guide to the Paper

The remainder of the paper is ordered as follows: First, we give an overview of the kinds of related-key and chosen-key attacks described in the remainder of the paper. We then consider the class of weak key pairs which yield the same S-box and the same round subkeys for n successive rounds, focusing on how common and easily found such key pairs are. Next, we discuss ways to use this class of weak key pairs to attack reduced-round Twofish versions, or at least to distinguish their behavior from that of an ideal cipher. We conclude with a number of interesting open questions raised by this research.

2 Related-Key Attacks using Weak Key Pairs

Twofish derives both its key-dependent S-boxes and its round and whitening subkeys from its key. When we considered related-key and chosen-key attacks on Twofish in [SKW$^+$98, SKW$^+$99], we were unable to find any useful related-key differential attack without using identical S-boxes for the pair of keys attacked. This is due to the use of the RS code when deriving the S-box keys from the input key. The result is a complex set of constraints on sets of keys which produce the same S-box keys.

In our attacks, the most useful property we have been able to find for a pair of keys is for both keys to generate the same S-boxes, and also the same round subkeys for some sequence of n successive rounds. This allows a number of attacks on the dissimilar rounds. In [SKW$^+$98, SKW$^+$99] we describe an attack of this kind based on a postulated large class of key pairs yielding eight identical round subkeys as well as identical S-boxes.

Because of the way the S-box keys are derived, requiring that two keys generate the same S-boxes puts very stringent constraints on the pair. In [SKW$^+$98, SKW$^+$99] we assumed that the only constraint on a pair was in the number of different S-boxes used for subkey generation that had to be active. This led to a calculation of the number of related-key pairs which we might expect to generate n successive identical round subkeys which, for each n considered, was much larger than the correct number. The calculation was inaccurate because we failed to consider the complicated set of linear relations imposed on the keys by requiring that the S-boxes remain identical. Also, rather than verifying whether a pairs of keys with some number n of colliding round subkeys existed, we used a very pessimistic (from the cipher designer's standpoint) estimate. This sufficed for the goal of that analysis, which was to check whether Twofish was vulnerable to related-key attacks, but seriously overestimated the impact of related key attacks on Twofish.

Below, we discuss our current estimates of how many pairs of keys may be expected to give a run of n identical round subkeys while also giving identical S-boxes. Given n identical round subkeys we discussed a number of attacks which could be mounted.

2.1 Attacks and Properties

The results we have on Twofish currently define a class of weak key pairs. Given access to encryptions under a pair of such keys, we can recover the effective keys for some reduced-round versions of Twofish with less work than we would need to simply try all members of the class of weak key pairs. This is the main sense in which it makes sense to consider an attack.

We can also consider attacks in which we request many pairs of keys with some chosen XOR difference, or other simple relationship, and wait for one of these weak key pairs against which to run one of our attacks on reduced-round Twofish. These attacks are necessarily very expensive, since they must be carried out on many different key pairs until a weak key pair is found. It is thus quite hard to find a related-key attack using these properties which is less work than brute-force searching the whole key.

Finally, we can consider properties of some reduced-round Twofish versions, based on weak key pairs, that differ from those expected from an ideal cipher. It is not clear what computational limits ought to be considered when looking at these properties. We use the rule of thumb that any property that can be detected with less work than a full key search is interesting. This even makes properties that require the entire plaintext/ciphertext mapping under the two keys worth investigating.

3 Determining n

This section concerns itself with dermining a value for n: the maximum number of rounds for which a pair of keys can share successive subkeys while at the same time producing identical S-boxes. If we look at the Twofish key schedule, we see that each round key (and each half of a whitening key) is computed from the key using eight S-boxes. The outputs from these 8 S-boxes are combined using two MDS matrixes, a PHT and some rotations but these are all invertible. To get identical round keys, we need to get identical outputs of the eight S-boxes.

3.1 The Eight S-boxes

The eight S-boxes are defined by

$$\begin{aligned}
u_0[i] &= q_1[q_0[q_0[q_1[q_1[i] \oplus m_{24}] \oplus m_{16}] \oplus m_8\ ] \oplus m_0] \\
u_1[i] &= q_0[q_0[q_1[q_1[q_0[i] \oplus m_{25}] \oplus m_{17}] \oplus m_9\ ] \oplus m_1] \\
u_2[i] &= q_1[q_1[q_0[q_0[q_0[i] \oplus m_{26}] \oplus m_{18}] \oplus m_{10}] \oplus m_2] \\
u_3[i] &= q_0[q_1[q_1[q_0[q_1[i] \oplus m_{27}] \oplus m_{19}] \oplus m_{11}] \oplus m_3] \\
u_4[i] &= q_1[q_0[q_0[q_1[q_1[i] \oplus m_{28}] \oplus m_{20}] \oplus m_{12}] \oplus m_4] \\
u_5[i] &= q_0[q_0[q_1[q_1[q_0[i] \oplus m_{29}] \oplus m_{21}] \oplus m_{13}] \oplus m_5] \\
u_6[i] &= q_1[q_1[q_0[q_0[q_0[i] \oplus m_{30}] \oplus m_{22}] \oplus m_{14}] \oplus m_6] \\
u_7[i] &= q_0[q_1[q_1[q_0[q_1[i] \oplus m_{31}] \oplus m_{23}] \oplus m_{15}] \oplus m_7]
\end{aligned}$$

where $m_0, \ldots, m_{31}$ are the 32 bytes of the key and the q_i's are the two q-boxes. This form is not the easiest to work with. First we observe that in a single round the boxes $u_0, \ldots, u_3$ get an argument of the form $2j$, while $u_4, \ldots, u_7$ get an argument of $2j + 1$. Furthermore, as we are only interested in equality, we can strip off the outer q-box. We now get:

$$\begin{aligned}
v_0[j] &= q_0[q_0[q_1[q_1[2j\quad] \oplus m_{24}] \oplus m_{16}] \oplus m_8\ ] \oplus m_0 \\
v_1[j] &= q_0[q_1[q_1[q_0[2j\quad] \oplus m_{25}] \oplus m_{17}] \oplus m_9\ ] \oplus m_1 \\
v_2[j] &= q_1[q_0[q_0[q_0[2j\quad] \oplus m_{26}] \oplus m_{18}] \oplus m_{10}] \oplus m_2 \\
v_3[j] &= q_1[q_1[q_0[q_1[2j\quad] \oplus m_{27}] \oplus m_{19}] \oplus m_{11}] \oplus m_3 \\
v_4[j] &= q_0[q_0[q_1[q_1[2j{+}1] \oplus m_{28}] \oplus m_{20}] \oplus m_{12}] \oplus m_4 \\
v_5[j] &= q_0[q_1[q_1[q_0[2j{+}1] \oplus m_{29}] \oplus m_{21}] \oplus m_{13}] \oplus m_5 \\
v_6[j] &= q_1[q_0[q_0[q_0[2j{+}1] \oplus m_{30}] \oplus m_{22}] \oplus m_{14}] \oplus m_6 \\
v_7[j] &= q_1[q_1[q_0[q_1[2j{+}1] \oplus m_{31}] \oplus m_{23}] \oplus m_{15}] \oplus m_7
\end{aligned}$$

3.2 Finding Collisions in a Single S-box

We are trying to find a collision for n different values of j. Let J be the set of values of j for which we try to find a collision, thus $|J| = n$. That is, we try to find two keys which generate the same values for $v_i[j]$ for $i = 0, \dots, 7$ and $j \in J$. In our original analysis we observed that there are 2^{32} different sets of keys for each of these S-boxes, and that therefore we can expect to find a collision for eight different values of j due to the birthday paradox. This is inaccurate; the structure of v_i is not purely random. Instead, the last key byte is a simple XOR at the output. This reduces the expected number of rounds for which we can find a collision for this S-box by one.

The first observation is that the sets of key bytes used for the eight S-boxes are disjunct. Therefore we can construct the collisions for each of the eight S-boxes separately. As there are only 4 bytes that define an S-box, we could do a 2^{32} search and find all collisions on the set J. We can do it even faster than that. Let $j_0 \in J$ and let $J' := J \backslash \{j_0\}$. (We assume that $|J| > 2$ which is the case we are interested in.) The key bytes $(m_{24}, m_{16}, m_8, m_0)$ define v_0. If there are two different values for the key bytes $(m_{24}, m_{16}, m_8, m_0)$ and $(m'_{24}, m'_{16}, m'_8, m'_0)$ which produce a collision on all $v_0[j]$ for $j \in J$, then they produce a collision on $v_0[j] \oplus v_0[j_0]$ for $j \in J'$. This leads us to define

$$\begin{aligned} v'_0[j] &:= v_0[j] \oplus v_0[j_0] = \\ &\quad q_0[q_0[q_1[q_1[2j] \oplus m_{24}] \oplus m_{16}] \oplus m_8] \oplus \\ &\quad\quad q_0[q_0[q_1[q_1[2j_0] \oplus m_{24}] \oplus m_{16}] \oplus m_8] \end{aligned} \tag{1}$$

in which the effect of the m_0 has been canceled out and we are left with a function affected by three key bytes. We need to find a triple of key bytes m_{24}, m_{16}, m_8, such that both keys produce equal $v'_0[j]$ values when $j \in J'$. There are 2^{24} different triples, and thus a total of 2^{47} pairs of triples. As each of the $n-1$ elements of J' imposes an 8-bit restriction, we expect that there are $2^{47-8(n-1)} = 2^{55-8n}$ pairs of triples which produce a suitable collision.

Once we have a a pair of triples (m_{24}, m_{16}, m_8) and (m'_{24}, m'_{16}, m'_8), we can extend them to quadruples to provide a full collision on $v_0[j]$ for all $j \in J$. We define

$$\begin{aligned} \delta := &\, q_0[q_0[q_1[q_1[2j] \oplus m_{24}] \oplus m_{16}] \oplus m_8] \\ &\oplus q_0[q_0[q_1[q_1[2j] \oplus m'_{24}] \oplus m'_{16}] \oplus m'_8] \end{aligned}$$

for any $j \in J$. (The value of j that we choose is irrelevant due to the fact that we have a collision on Equation 1). We can choose a m_0 at random, and compute $m'_0 := m_0 \oplus \delta$. The two quadruples, $(m_{24}, m_{16}, m_8, m_0)$ and $(m'_{24}, m'_{16}, m'_8, m'_0)$, now produce the same set of values $v_0[j]$ for $j \in J$.

Note that we can easily compute the set of all pairs of quadruples with this property. It is 2^{24} steps of work to find all pairs of triples, and each of these leads to 256 pairs of quadruples. We do not even bother to store all 256 pairs of quadruples. All we store are the values m_{24}, m_{16}, m_8, m'_{24}, m'_{16}, m'_8, and δ; generating the structure of 256 pairs of quadruples they represent is trivial.

We can perform the equivalent calculations for each of the eight S-boxes and store all the results. For each S-box we expect to store 2^{55-8n} structures which is quite reasonable for the values of n we are interested in. Table 1 shows for different n how many of these structures we found for the set $J = \{5, 6, \dots, n+4\}$, which are the values of j used to compute the round keys of the second round onwards. The exact numbers for different choices of J will be slightly different, but we expect no major deviations. Note that the actual number of structures that we found tends to be slightly larger than our expected value. We have no theoretical explanation for this effect; it might relate to the structure of the q-boxes or the way the S-boxes are built out of the q-boxes.

3.3 The RS Restriction

As Table 1 shows, there is no collision on 8 rounds for our chosen set J. For $n < 8$ we can construct two keys, K and K^*, that generate the same set of round keys. But this is only half the condition; they also have to generate the same S, and thus the same set of S-boxes for the round function.

The S-box key S can be written as a 4×4 matrix over GF(2^8) defined by

$$M := \begin{pmatrix} m_{24} & m_{16} & m_8 & m_0 \\ \vdots & \vdots & \vdots & \vdots \\ m_{31} & m_{23} & m_{15} & m_7 \end{pmatrix}$$

$$S := \mathrm{RS} \times M$$

where M is a 8×4 matrix filled with the key byte and RS is the 4×8 Reed-Solomon (RS) matrix. Note that this is not the usual representation of S, but it is an equivalent one. The order of the columns of M was chosen to simplify the discussion to follow.

	n=4	n=5	n=6	n=7	n=8
v_0	8589897	34138	133	1	0
v_1	8595400	34810	122	0	0
v_2	8603230	33720	139	0	0
v_3	8594247	33984	132	0	0
v_4	8597633	34354	135	0	0
v_5	8596388	34510	124	0	0
v_6	8602423	34010	125	0	0
v_7	8601218	34121	137	0	0

Table 1: Number of collision structures found for each S-box

We are not interested in the actual value of S, only in the fact that it should be the same between the two keys K and K^*. Since S is a linear function of M, it is natural to look at the differences in M. Looking back at the results of Section 3.2, we are now only interested in the difference between two quadruples. We call such a difference a *row*, and each structure naturally leads to the row $(m_{24} \oplus m'_{24}, m_{16} \oplus m'_{16}, m_8 \oplus m'_8, \delta)$. Note that we get one difference value for each structure of 256 quadruples. Let R_i be the set of all rows that we found for S-box i, plus the all-zero row $(0, 0, 0, 0)$. (Two different structures might result in the same row, in which case $|R_i|$ is less than the number of structures we found.)

If we have two keys K and K^* that produce the same S-boxes, then we can look at the two matrixes M and M' that they define. In particular, we define $R := M - M'$. (We can also see this as $R = M \oplus M'$ with byte-wise XOR operations as these matrixes live in GF(2^8) in which both addition and subtraction are equivalent to a byte-wise XOR.) The condition that K and K^* produce the same S is now equivalent to saying that RS × $R = 0$. Furthermore we know that the first row of R must be a member of R_0 otherwise K and K^* would not produce a collision on the round keys. Similarly, each row of R must be an element of the corresponding R_i.

3.4 Finding a Key Difference

We are now close to our goal. All that is left is to select one row from each of our R_i to make up a matrix R with $\mathrm{RS} \times R = 0$. The trivial solution $R = 0$ is not interesting, as that corresponds to $K = K^*$. For any nontrivial solution we know that every nonzero column of R must have at least five nonzero elements. (This is a property of the RS matrix). As we must have at least one nonzero column, we must use collisions on at least five of the S-boxes. This argument immediately shows that the case $n = 7$ cannot occur as there are not enough S-boxes for which we have a collision.

3.5 Exhaustive Search Algorithm

In the case $n = 6$ we have approximately 128 different row values for each of the eight rows of R. We performed an exhaustive search to find any nontrivial R that satisfies the equation. The simple method is to try all 2^{28} ways of choosing the first four rows. Given the first four rows, the last four rows are uniquely determined by the linear equation. These are thus easy to compute, and it is easy to check whether each row value occurs in the respective R_i set. Note that only about 2^{28} of the 2^{128} sets of four row values for the last four rows are possible, so the chances of any one matching is approximately 2^{-100} and the chances of finding a suitable R are about 2^{-72}. Our search found no suitable R, which rules out the case $n = 6$ for the sets J we are discussing. Solutions for other sets J for the case $n = 6$ are so unlikely that they do not seem to be worth looking for.

3.6 An Improved Algorithm

The algorithm above cannot handle the case $n = 5$ as it would take about 2^{60} steps. An improved algorithm can be used which is faster. Let N be the average number of row values for each row. We want to guess the first three rows and then quickly eliminate many of our guesses to reduce the set of possible choices for the first three rows. We choose three of the remaining row positions which we will call the filtered rows; we will use several of these choices but we will describe the case where we choose the last three row positions. We now choose p, a 1×4 row vector over GF(2^8), such that $p \times \mathrm{RS}$ has 0 coordinate values in columns 6, 7 and 8. Observe $x := p \times \mathrm{RS} \times R$, which must of course be zero. The choices for the last three rows of R do not affect this value. The choices for rows 4 and 5 together generate $N^2 \approx 2^{30}$ different values. We build a bitmap of 2^{32} bits; each bit represents a possible value for x and we set those bits that correspond to a value that can be generated by the choices of rows 4 and 5.

Any choice that we make for rows 1–3 results in a contribution to x which has to be cancelled by rows 4 and 5. With our bitmap we can very quickly discard about one-quarter of the guesses for rows 1–3. We can make similar bitmaps for other initial choices of our three filtered rows. There are ten suitable choices for which rows to filter. If we use eight of them, and 2^{32} bytes of memory to store the bitmaps, then we expect 1 in 2^{16} of our guesses to pass the eight tests.

The algorithm now proceeds as follows. We run over all $N^3 \approx 2^{45}$ guesses for the first three rows and apply eight of the bitmap tests. We expect that about 2^{29} guesses survive these tests. For these cases we guess an additional row. We now have four of the rows which together with the linear restriction fully specify R, and can therefore quickly determine whether this is a solution. The overall running time of this algorithm is dominated by the first phase, which takes 2^{45} steps for the case $n = 5$.

We have not had the resources to implement this algorithm and run it for $n = 5$. However, we can perform the same probability analysis that we did before. The total number of matrixes R that we can construct from our eight set of rows is about $2^{120.5}$. Each R has a probability of 2^{-128} of satisfying the linear restriction, so we expect to find about $2^{-7.5}$ solutions for $n = 5$. This is well below one, and based on this information we estimate that the case $n = 5$ is not possible. To be quite sure we would have to test all "interesting" sets J as well. There are 12 sets of J which correspond to 5 consecutive round keys. Then there are some attacks that require equal whitening keys, which produce a few more interesting sets J. All in all we still expect that there is no solution for $n = 5$. Therefore we believe that at best it is possible to find an R for $n = 4$.

We assume that any attacker has the resources to construct suitable weak key pairs. For $n = 5$ this is certainly technically feasible using our improved algorithm. For $n = 4$ it is much easier. The attacker simply chooses the first four rows of R arbitrarilly. The RS restriction uniquely determines the value for the last four rows, and there is a 2^{-36} chance that these four rows are possible values.

3.7 If We Find an R

If we find an R with the desired properties, then we immediately get a huge number of key pairs (K, K^*) that satisfy all our restrictions. If R has eight nonzero rows (which seems to be the most likely case), then we have two key values for each row that we can use. One of each is given to K and K^*, which provides us with 2^8 different key pairs. Furthermore, our rows only specify the difference between the last key bytes in every quadruple. Given any pair (K, K^*) we can XOR any constant into the first eight bytes of both keys and get another pair of keys with the same property. (This corresponds to the 256 pairs of quadruples that each row generates.) All in all we expect to get about 2^{72} different key pairs, all of which share the same property. Note that we are counting ordered pairs, so both (K, K^*) and (K^*, K) are counted separately. We expect that these 2^{72} key pairs consist of 2^{71} different keys, and generate 2^{32} different values of S, with each S being generated 2^{40} times.

As we mentioned above, for the case $n = 5$ we expect no valid R at all, but if one exists, we get a set of 2^{72} weak key pairs. For the case $n = 4$ we expect to get about 2^{56} different solutions for R, each of which generates a set of 2^{72} weak key pairs.[1] We therefore expect that for $n = 4$ there are about 2^{128} weak key pairs made up out of 2^{127} different keys.

4 Attacks Based on Weak Key Pairs

For the remainder of this paper, we pretend that Twofish is a pure Feistel cipher (without the rotations in the datapath). We can in fact rewrite Twofish as a pure Feistel cipher [Fer99], although that introduces additional rotations at the input and output of the round function. Taking the rotations into account complicates the description of our attacks, but it does not seem to affect their complexity.

As our type of weak key pairs cover more rounds for the 256-bit key size then for shorter keys, and the permissable workload for a 256-bit key size attack is higher, we will only consider the case of the 256-bit key cipher. The 128-bit key and 192-bit key versions are much harder to attack using this type of weak keys.

[1] In this estimate we ignore minor effects, such as what happens when two colisions in a single S-box generate the same row difference for R.

There are three different kinds of attacks that we look at:

1. Related-key attacks, in which the attacker requests a pair of keys with a certain relationship, and tries to use this to recover both keys.

2. Weak key pair detection attacks, in which the attacker tries to detect a weak key pair with less work than he would require to simply try all possible weak key pairs.

3. Properties that reduced-round Twofish has, but which an ideal cipher is not expected to have, using these weak key pairs.

4.1 Related-Key Attacks

In a related-key attack, the attacker is permitted to request encryptions under a number of different keys. He is allowed to choose the relationship between the keys, but not the keys themselves. An attacker can exploit a weak key pair to mount a related key attack. He requests a large set of keys, with some simple relationship (such as a fixed XOR between pairs of keys), and hopes to end up with a weak key pair. He then mounts an attack to detect this weak key pair.

Related-key attacks using our type of weak key pairs are not very powerfull. There is no general simple relationship between two keys in a pair. The best an attacker can do is to choose a XOR difference between two keys given by the matrix R. If the first key happens to be one of the set of 2^{71} keys that occur in weak key pairs generated by this R, then the attacker gets a weak key pair. The chances of this happening are 2^{-185}, so the attacker has to detect the weak key pair in less than 2^{71} work. Even for the case $n = 4$, where there are far more weak key pairs, the same bound holds because the attacker restricts himself to the weak keys generated by a single R at the moment that he chooses his difference. In other words, this type of related key attack boils down to detecting weak key pairs in less than 2^{71} steps. This is our second type of attack, which we will discuss in more detail in section 4.2.

As noted in [KSW96], there is a general related-key attack based on the time-memory tradeoff; an attacker given 2^{128} related-key queries and 2^{64} chosen plaintext queries per key, as well as a lot of memory, can break any cipher with a 256-bit key and 128-bit plaintext.

4.2 Detecting Weak Key Pairs

We will now discuss attacks which allow an attacker to detect whether two keys unknown to him form a weak key pair. If the attacker can do this in less than 2^{71} steps, it can be used to create a related-key attack.

In the case $n = 4$ there are about 2^{128} weak key pairs made up out of 2^{127} keys, but that does not mean that detecting a weak key pair in less than 2^{127} steps makes an attack. After all, out of the total of 2^{512} possible key pairs only 2^{128} are weak. An attacker is much better off doing an exhaustive key search than waiting for a random pair of keys to be weak. In a related-key attack the attacker can use the relation between the two unknown keys to improve the chances of getting a weak key pair, but as we discussed this still requires the weak key pair to be detected within 2^{71} steps.

4.2.1 Detecting a Weak Key Pair with $n+2$ rounds

Consider a Twofish variant with $n+2$ rounds and no whitening, and a weak pair of keys which give identical round subkeys for the first n rounds. When we encrypt X_i under these two keys to get Y_i and Y_i^*. The value $Y_i \oplus Y_i^*$ has two bits that are constant and independent of X_i.

To see why this property exists we look in more detail at the Twofish encryption. If we compare the two encryptions, the first difference occurs when the round keys for the nex-to-last round are added to the output of the F'-function. This results in a fixed XOR difference of the two least significant bits of the two 32-bit words. This difference is XORed into the right half of the encryption state, and appears in the left half of the ciphertext.

By observing these two bits, we can detect a weak key pair on $n+2$ rounds. Rejecting a key pair that is not weak requires $2\frac{1}{3}$ identical plaintexts under each key on average. As there are about 2^{512} key pairs that are not weak, we need about 2^{8} plaintexts to fully detect the weak key pair property.

Another way of looking at this attack is to split the cipher into two parts. The first n rounds are an unknown but fixed mapping. The last 2 rounds is what we are attacking. Our attack on these 2 rounds cannot use any properties of the plaintext other then equality, as all other useful properties are destroyed by the first n rounds.

4.2.2 Detecting a Weak Key Pair with n rounds and whitening

Consider a pair of keys, (K, K^*), which produce identical input whitening keys, and identical round subkeys for $n-2$ rounds. This corresponds to a set J of size n that includes the indexes for the input whitening key generation. We can detect such a key pair using the same property. The XOR of two ciphertexts generated by encrypting the same plaintext under the two keys will have two fixed bits. Therefore the same attack applies.

4.2.3 Detecting a Weak Key Pair with $n+1$ rounds and whitening

Consider a pair of keys which has n identical 64-bit subkeys, arranged as:

- The second input whitening key.
- The first $n-1$ rounds.

We now guess the 64-bit difference in the left half of the input whitening key. For each guess, we can create pairs of plaintexts (X_i, X_i^*) which, when encrypted with their respective keys, result in the same input to the first round. The encryption states remain identical until the last two rounds. As above, we get two fixed bits in the XOR of the two ciphertexts if our guess was correct.

As we noted above, it takes on average $2\frac{1}{3}$ plaintext to discard a wrong guess, and each plaintext requires two encryptions to be performed. The overall complexity of this attack is therefore about 2^{66}, which is just below the threshold of 2^{71}.

4.2.4 Detecting some Weak Key Pair with $n+3$ rounds

We now turn to the case of $n+3$ rounds where the first n rounds have the equal round keys. We generate a large number of arbitrarilly chosen X_i values and encrypt them under both keys. After about 2^{64} tries we expect to find an X_i such that Y_i and Y_i^* have identical left halves. We call such a pair of ciphertexts a matching pair.

Recall that $R_{r,i}$ with $i \in \{0, \ldots, 3\}$ is word i of the encryption state after round r, and that the rounds are numberd $0, 1, \ldots n+2$ for an $n+3$ round Twofish version. As we discussed earlier, the encryptions of a matching pair have a fixed difference in bit 0 (the least significant bit) of $R_{n,1}$. (The R values are taken after the swap, so the fixed difference after $n+1$ rounds is now in the left half of the state.) This propagates to a fixed difference in bit 0 of $R_{n+1,3}$. We then encounter a one-bit rotation to the left, which moves the fixed difference to bit 1 of the word.

We now look at the output of the F-function of the last round. Note that the inputs to this F-function in the two encryptions are identical, as they are formed by the two left halves of the ciphertexts of a matching pair. Therefore, the outputs of the F'-function (without the key addition) are identical too. The only differences in the last round between the two encryptions are the round keys. We are only interested in bit 1 of the second output word of the F-function. If the two keys of the weak key pair have the same least-significant bit in the second word of the round key of the last round, then the difference between the encryptions in bit 1 of the second output word of the F-function in the last round is fixed. This occurs for about half of all weak key pairs.

The cipher now combines our two bits with a fixed difference in the Feistel-XOR of the last round. The result is that bit 1 of the third word of the ciphertext must have a fixed difference, irrespective of the value X_i.

The end effect is that, for a matching pair and for a weak key pair that matches our additional restriction, one bit of $Y_i \oplus Y_i^*$ has a fixed value. On average we will need 3 matching pairs to reject a non-weak key pair. Thus, we can detect about half of the weak key pairs using less than 2^{67} encryptions on average.

4.3 Unexpected properties

We now turn to our third category of attacks: distinguishable properties that reduced-round Twofish has that result from our weak key pairs. In this type of attack we assume that the attacker can force the use of a weak key pair, and is looking for some property that relates the two resulting encryption functions. Attacks like this can occur in settings where the block cipher is used to create a hash function. Note that the purpose of the attack is not to recover the key. Rather, there might be a situation in which the property of the cipher leads to a more efficient attack on the entire system then would be possible with a perfect cipher.

4.3.1 Correlations on $n+2$ rounds with whitening

Let (K, K^*) be a weak key pair which generated identical round keys for the first n rounds, and let

Δ be the XOR difference between the input whitening keys. If we compare the encryptions of $E_K(X)$ with $E_{K^*}(X \oplus \Delta)$ then we get the same result after n rounds. As we saw in section 4.2.1 this leads to two bits after $n+2$ rounds that have a fixed difference between the two encryptions.[2] The output whitening does not change this property.

In other words, after one try to determine the actual difference value, encrypting X with key K gives us direct information about the result of the encryption of $X \oplus \Delta$ with key K^*. This is a property that an ideal cipher would not have, and one that might be useful to an attacker under some circumstance.

By itself it is not useful to try and use it to predict the result of encryption by K^*. We are working in a setting where the attacker already knows the two keys, so the attacker can compute the encryption under K^* himself. We have not found a situation in which this correlation property leads to a weakness of the system. Nevertheless, we consider this property to be so serious that we do not want to use Twofish with $n+2$ rounds.

4.3.2 Correlations on $n+3$ rounds with whitening

We can go one round further and show a very slight correlation on $n+3$ rounds with whitening. We use the property from section 4.2.4. Let (K, K^*) be a weak key pair, let Δ_i be the XOR difference between the input whitening keys and let Δ_o be the XOR difference between the output whitening keys. We define

$$C(X) := E_K(X) \oplus E_{K^*}(X \oplus \Delta_i) \oplus \Delta_o$$

If, for any X, the left half of $C(X)$ is zero, then there is a particular bit in the right half that has a fixed value.

Given a quadruple $(K, K^*, \Delta_i, \Delta_o)$ we can detect this property in 2^{80} steps or so. Once again, this does not translate to any type of attack. (If we know the keys we can detect them much easier.) For this property to be an attack there would have to be a system which is weakened by this property. Given that this correlation is very small, and we have not even found a useful attack using the much stronger property of section 4.3.1, we do not consider this correlation to be significant.

4.3.3 Even more rounds

There is a trivial and general way to extend this type of property to more rounds. In the previous sections we already used the Δ-constants to eliminate the whitening differences. This is the start of taking the cipher apart round by round. Given an encryption function $E_K(X)$, we can find a pair (K, f) such that $X \mapsto f(E_K(X))$ is a very simple function, for example by letting f undo the last few rounds of the encryption. This is obviously not an interesting property as it applies to all block ciphers.

4.4 Reflection Attacks

Another idea is to use the equal round keys to cancel each other. Suppose we have $n+d$ rounds without whitening, and a weak key pair (K, K^*) where the last n round keys are equal. Consider $f(m) := D_{K^*}(E_K(m))$. In other words, we encrypt using K and then decrypt using K^*. As the last n round keys are the same, they cancel in this construction and we are left with a cipher that has $2d$ rounds. This cipher is very similar to Twofish, except that there is a swap missing in the middle, so that the two middle rounds both operate in the same direction. The two Feistel round functions in these rounds have the same inputs, and except for the round keys produce the same output. Thus they more or less cancel each other out. We have not studied this Twofish variant with a swap missing, but we will be very generous and assume that $2d$ rounds of this variant is as hard to attack as $2d-2$ rounds of ordinary Twofish. Our best direct cryptanalytical attack on Twofish is on 6 rounds, so we expect that we can make this atack work for at most $n+4$ rounds without whitening.[3]

Whitening spoils much of the effectiveness of this attack. If the output whitening generated by K and K^* are different, then the cancellation of the rounds which have equal round keys no longer takes place. Thus, to allow this construction to work at all we have to require that the output whitening keys are identical. If we take n to be the number of equal round keys or equal whitening-key halves then we get only $n-2$ rounds with identical keys. We therefore expect at most to be able to attack $n+2$ rounds with whitening.

[2] To be more precise: there are at least two fixed difference bits. There are many weak key pairs that generate more fixed difference bits.

[3] This is of course a very rough estimate. Further analysis on this type of attack will probably give us a better bound. Note however, that we have to improve an attack on the missing-swap variant by two rounds to increase d by one. This seems to be a very difficult and unrewarding way of attacking Twofish.

5 Conclusions and Open Questions

In this paper, we have reconsidered the difficulty of related-key and partial chosen-key attacks on reduced-round Twofish. By experimentally verifying the number of successive round subkeys for which it was possible to get pairs of keys which collide while also colliding in the S-box keys, we have determined that our previous related-key attack cannot work. Related-key attacks of the kind we described in [SKW$^+$98, SKW$^+$99] cannot cover more than five successive rounds with identical subkeys, and most likely cover only four successive round keys.

We have made improvements in the attacks that make use of weak key pairs. Even with these improvements related-key attacks do not seem to be more powerfull against Twofish then other types of attack.

There is a large set of possible related-key attacks which we have not considered here. For example, related-key attacks using keys that derive different S-boxes have not been considered. Likewise, attacks which interleave identical and different round subkeys, instead of putting the different subkeys at one or the other end of the cipher, have not been considered. We have also not closely considered attacks in which a pair of keys is directly chosen, and large sets of colliding plaintext/ciphertext pairs are sought. This kind of attack would likely be applied to a Twofish-based hashing mode.

References

[Fer99] Niels Ferguson. Impossible differentials in Twofish. Twofish Technical Report 5, Counterpane Systems, October 1999. See `http://www.counterpane.com/twofish.html`.

[KSW96] John Kelsey, Bruce Schneier, and David Wagner. Key-schedule cryptanalysis of IDEA, G-DES, GOST, SAFER, and triple-DES. In Neal Koblitz, editor, *Advances in Cryptology—CRYPTO '96*, volume 1109 of *Lecture Notes in Computer Science*, pages 237–251. Springer-Verlag, 1996.

[SKW$^+$98] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. Twofish: A 128-bit block cipher. In *AES Round 1 Technical Evaluation CD-1: Documentation.* NIST, August 1998. See `http://www.nist.gov/aes`.

[SKW$^+$99] Bruce Schneier, John Kelsey, Doug Whiting, David Wagner, Chris Hall, and Niels Ferguson. *The Twofish Encryption Algorithm, A 128-Bit Block Cipher.* Wiley, 1999.