Twofish Technical Report #5

Impossible differentials in Twofish

Niels Ferguson[*]

October 19, 1999

Abstract

We show how an impossible-differential attack, first applied to DEAL by Knudsen, can be applied to Twofish. This attack breaks six rounds of the 256-bit key version using 2^{256} steps; it cannot be extended to seven or more Twofish rounds.

Keywords: Twofish, cryptography, cryptanalysis, impossible differential, block cipher, AES.

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

1 Introduction

Twofish is one of the finalists for the AES [SKW$^+$98, SKW$^+$99]. In [Knu98a, Knu98b] Lars Knudsen used a 5-round impossible differential to attack DEAL. Eli Biham, Alex Biryukov, and Adi Shamir gave the technique the name of 'impossible differential', and applied it with great success to Skipjack [BBS99]. In this report we show how Knudsen's attack can be applied to Twofish. We use the notation from [SKW$^+$98] and [SKW$^+$99]; readers not familiar with the notation should consult one of these references.

2 The attack

Knudsen's 5-round impossible differential works for any Feistel cipher where the round function is invertible. Twofish has some additional rotations in the datapath which make things a bit more complicated. It turns out that these extra rotations do not complicate the attack, but only its description. To avoid these complications we first rewrite Twofish as a pure Feistel cipher.

2.1 Twofish as a pure Feistel cipher

As mentioned in [SKW$^+$98, section 7.9] and [SKW$^+$99, section 7.9.3] we can rewrite Twofish to be a pure Feistel cipher. We will demonstrate how this is done. The main idea is to save up all the rotations until just before the output whitening, and apply them there. We will use primes to denote the values in our new representation. We start with the round values:

$$\begin{aligned} R'_{r,0} &= \mathrm{ROL}(R_{r,0}, \lfloor (r+1)/2 \rfloor) \\ R'_{r,1} &= \mathrm{ROR}(R_{r,1}, \lfloor (r+1)/2 \rfloor) \\ R'_{r,2} &= \mathrm{ROL}(R_{r,2}, \lfloor r/2 \rfloor) \\ R'_{r,3} &= \mathrm{ROR}(R_{r,3}, \lfloor r/2 \rfloor) \end{aligned}$$

To get the same output we update the rule to compute the output whitening.

$$\begin{aligned} C_0 &= \mathrm{ROR}(R_{n,2}, \lfloor n/2 \rfloor) \oplus K_4 \\ C_1 &= \mathrm{ROL}(R_{n,3}, \lfloor n/2 \rfloor) \oplus K_5 \\ C_2 &= \mathrm{ROR}(R_{n,0}, \lfloor (n+1)/2 \rfloor) \oplus K_6 \\ C_3 &= \mathrm{ROL}(R_{n,1}, \lfloor (n+1)/2 \rfloor) \oplus K_7 \end{aligned}$$

[*] Counterpane Internet Security, Inc.

where n is the number of rounds. Of course, $n = 16$ for the full cipher but we will be attacking reduced-round versions and that would affect the rotation amounts.

We now have to adjust the round function to the new representation. The input to the round function of round r consists of $(R'_{r,0}, R'_{r,1})$. At the beginning of the round function we undo the rotates to get the original $(R_{r,0}, R_{r,1})$, and apply the original F-function to get $(F_{r,0}, F_{r,1})$. We then apply appropriate rotates to the outputs, namely:

$$\begin{aligned} F'_{r,0} &= \mathrm{ROL}(F_{r,0}, \lfloor r/2 \rfloor) \\ F'_{r,1} &= \mathrm{ROR}(F_{r,1}, \lfloor (r+2)/2 \rfloor) \end{aligned}$$

We can now write the update rule:

$$\begin{aligned} R'_{r+1,0} &= \mathrm{ROL}(R_{r+1,0}, \lfloor (r+2)/2 \rfloor) \\ &= \mathrm{ROL}(\mathrm{ROR}(R_{r,2} \oplus F_{r,0}, 1), \lfloor (r+2)/2 \rfloor) \\ &= \mathrm{ROL}(R_{r,2}, \lfloor r/2 \rfloor) \oplus \mathrm{ROL}(F_{r,0}, \lfloor r/2 \rfloor) \\ &= R'_{r,2} \oplus F'_{r,0} \\ R'_{r+1,1} &= \mathrm{ROR}(R_{r+1,1}, \lfloor (r+2)/2 \rfloor) \\ &= \mathrm{ROR}(\mathrm{ROL}(R_{r,3}, 1) \oplus F_{r,1}, \lfloor (r+2)/2 \rfloor) \\ &= R'_{r,3} \oplus \mathrm{ROR}(F_{r,1}, \lfloor (r+2)/2 \rfloor) \\ &= R'_{r,3} \oplus F'_{r,1} \\ R'_{r+1,2} &= R'_{r,0} \\ R'_{r+1,3} &= R'_{r,1} \end{aligned}$$

As we can see from these formulas, the combining function used to mix the output of the round function and the right half is now a simple XOR.

All in all we can replace the one-bit rotations in Twofish by round-dependent rotations at the input and output of the round function, and by some suitable rotations just before the output whitening.

To make it easier to follow the differentials through the cipher we make one more change. Instead of having a swap after each round, we apply the round functions alternating from left to right and from right to left. This is the representation that we will use for the rest of this paper. If the number of rounds is even we need a swap just before the output whitening, but we will ignore this as it does not affect any of the attacks.

2.2 The impossible differential

We will now look at a particular differential for five rounds. The XOR input difference (written as two 64-bit quantities) is of the form $(0, \alpha)$ for some nonzero α, and the output difference after 5 rounds is also $(0, \alpha)$. We can draw the following conclusions: the difference after round 1 $(0, \alpha)$, the difference after round 4 is $(0, \alpha)$, the difference after round 2 is (β, α) for some nonzero β, and the difference after round 3 is (γ, α) for some nonzero γ. We trivially get that $\beta = \gamma$, but more interesting is the output difference of the round function in the third round. As the difference in the right half after round 2 is equal to the difference in the right half after round 3, we conclude that the output difference of the round function in the third round must be zero. As the input difference to the round function in the third round is nonzero and the round function is bijective, we get a contradiction. We conclude that the differential $(0, \alpha) \mapsto (0, \alpha)$ is a 5-round impossible differential.

2.3 Finding the last round key

To use this impossible differential we search for plaintext pairs that exhibit a 6-round differential of the form $(0, \alpha) \mapsto (\delta, \alpha)$ for any $\alpha \neq 0$.

Our first task is to find pairs of this form. Fix an 8-byte value A, and let B_i take on all 2^{64} possible 8-byte values. Encrypt each of the plaintexts (A, B_i) to produce (C_i, D_i), and sort them by the value $B_i \oplus D_i$. Any pair of encryptions that has the same $B_i \oplus D_i$ produces a difference of the correct form. As there are 2^{127} pairs and a 64-bit restriction, we expect to get about 2^{63} pairs with the right form of differential.

If we ignore the output whitening for a moment, we can now try each possible round key for the last round with each of the differentials that we found, and compute what the differential would be after the fifth round. We know that this cannot be of the form $(0, \alpha)$, so any key value that produces this difference after five rounds must be wrong. We keep track of which key values we have determined are wrong until the key space is sparse enough to apply other methods.

Each 6-round differential of the form $(0, \alpha) \mapsto (\delta, \alpha)$ eliminates about one in 2^{64} of the possible keys for the last round, so a single structure of 2^{64} plaintexts that generates 2^{63} of these differentials will reduce the key space by about one bit. We can of course generate many more structures to generate more suitable differential pairs.

2.4 128-bit keys

To make this attack work for 128-bit keys we have to arrange these computations in a proper way; otherwise we end up decrypting 2^{64} ciphertexts for one round for each of the 2^{128} possible keys of the last round, which is obviously more work than an exhaustive search.

The best way we see to arrange this is to guess the 64-bit S-box key S. For the two ciphertexts of the differential pair we decrypt part of the last round, stopping at the point where the round keys get added in. We now know that adding the round key to these two outputs is not allowed to produce a specific XOR-differential. We assume that we can generate the set of disallowed round keys very fast in constant time (on average).

All in all we have to compute the F'-function (i.e. F without the key additions) of the last round about 2^{128} times, and have to generate the forbidden set of round keys 2^{127} times to get a 1-bit reduction in the key space. The remaining 2^{127} keys can be exhaustively searched. With a bit of luck this will be marginally faster than an exhaustive key search over all 2^{128} keys as the attack requires a total of $2^{128} + 6 \cdot 2^{127}$ computations of the round function whilst the exhaustive key search requires $6 \cdot 2^{128}$ computations of the round function plus the generation of the round keys through the key schedule.

The output whitening completely destroys this attack, as we now need to guess both the S-box key and 64 bits of the whitening key to get any data. Effectively we have to guess the entire cipher key. Even after this 128-bit guess every differential pair gives us only a 2^{-64} probability of eliminating that key, whereas we could eliminate it with probability nearly 1 if we just did a full trial encryption (which of course corresponds to an exhaustive key search).

2.5 192-bit keys

For 192-bit keys the S-box key is 96 bits long. Using the same algorithm as in the 128-bit key we can reduce the overall key space by 1 bit in 2^{160} steps. Repeating the operation for a few more structures gives us an overall complexity of about 2^{160} for the entire attack.

Again, the output whitening makes this attack impossible. We have to guess 64 bits of the whitening key in addition to the 96-bit S-box key before we can even start to eliminate the round keys of the last round. To halve the key space we have to compute the F'-function of the last round for 2^{64} ciphertexts for each of the 2^{96+64} key guesses, which clearly exceeds the complexity of an exhaustive key search.

2.6 256-bit keys

For 256-bit keys the S-box key is 128 bits long. The basic attack thus has an overall complexity of about 2^{192} steps.

Even with the whitening we can perform this attack. All that is required is to guess 64 bits of the post-whitening key. This brings the attack complexity to 2^{256} steps, but as we mentioned before each step might very well be faster than a single encryption.

We can also push the attack through one more round without whitening. In the 7-round cipher we use the 5-round impossible differential $(\alpha, 0) \mapsto (\alpha, 0)$ in the middle five rounds. We look for differential pairs for the 7-round cipher that have the form $(\alpha, \epsilon) \mapsto (\alpha, \delta)$. These can be generated by encrypting random plaintexts, and sorting them by the XOR of the first half of the plaintext with the first half of the ciphertext. We guess the S-box key S (128 bits). We find out which of the round keys in the first round would produce a difference after the first round of $(\alpha, 0)$, and which of the round keys of the last round produce a difference before the last round of $(\alpha, 0)$. The cross product of these two sets can all be eliminated from the set of possible round keys for the first and last round. The probability of any differential of the form $(\alpha, \epsilon) \mapsto (\alpha, \delta)$ eliminating any one choice of the first and last round key is about 2^{-128}. We therefore need 2^{128} plaintext/ciphertext pairs with the appropriate differential. As a random pair has a chance of 2^{-64} of matching the differential, we need 2^{192} pairs in total, which we can get using 2^{96} plaintext/ciphertext pairs.

All in all we do the following computations for each choice of S-box keys: We compute the first and last round of the cipher for 2^{96} plaintext/ciphertext pairs. We then take the 2^{128} pairs with the proper differential and generate the set of forbidden round keys for the first and last round. We hope that we can implement each of these 2^{256} steps significantly faster than the cipher itself. After these steps the keyspace has been halved, and we can recover the entire key by searching the remaining keyspace.

As discussed in [WKS$^+$99] any pair of 64-bit round keys (or whitening-key halves) has only about 117 bits of entropy. This outlaws some sets of the first and last round keys already, but it does not seem to speed up the attack in any way.

An alternative way of doing a 7-round attack is to use the impossible differential $(0, \alpha) \mapsto (0, \alpha)$ in the first five rounds. By guessing the S-box key and the last round key (a total of 192 bits) we can decrypt the last round. We can now mount the 6-round attack using only 2^{64} fast steps just like we did in the 128-bit key case as the S-box key is already known. Again the fact that each pair of round keys has only 117 bits of entropy does not really help us as we are not iterating over the remaining round keys but instead are using 2^{64} differentials to eliminate half of the remaining keys space.

Neither of these extensions to seven rounds work when whitening is present.

3 Conclusions

Without whitening, the impossible differential attack on 6-round Twofish has complexity 2^{128} for 128-bit keys (but is still faster than exhaustive search), complexity 2^{160} for a 192-bit key, and complexity 2^{192} for a 256-bit key. We can also attack seven rounds in 2^{256} steps for a 256-bit key, again without whitening.

With whitening, the best impossible-differential attack on Twofish is on six rounds, has complexity 2^{256} and works only for a 256-bit key. We see no way to extend this attack to seven or more rounds.

References

[BBS99] Eli Biham, Alex Biryukov, and Adi Shamir. Cryptanalysis of Skipjack reduced to 31 rounds using impossible differentials. In Jacques Stern, editor, *Advances in Cryptology—EUROCRYPT '99*, volume 1592 of *Lecture Notes in Computer Science*. Springer-Verlag, 1999.

[Knu98a] Lars R. Knudsen. DEAL—a 128-bit block cipher. Technical report 151, Department of Informatics, University of Bergen, Norway, February 1998.

[Knu98b] Lars R. Knudsen. DEAL—a 128-bit block cipher. In *AES Round 1 Technical Evaluation CD-1: Documentation*. NIST, August 1998. See `http://www.nist.gov/aes`.

[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.

[WKS+99] Doug Whiting, John Kelsey, Bruce Schneier, David Wagner, Niels Ferguson, and Chris Hall. Further observations on the key schedule of Twofish. Twofish Technical Report 4, Counterpane Systems, March 1999. See `http://www.counterpane.com/twofish.html`.