Twofish Technical Report #2

Empirical Verification of Twofish Key Uniqueness Properties

Doug Whiting* David Wagner†

September 22, 1998

Abstract

Twofish is a 128-bit block cipher submitted as an AES candidate. Each distinct Twofish key generates a unique sequence of subkeys K_i, and each round function F is unique for a distinct value of the S bits used to generate the S-boxes. Thus, no two distinct keys result in an identical sequence of round functions.

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

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

1 Introduction

In the Twofish encryption key schedule for an N-bit key ($N = 128, 192, 256$), three different sets of key material are used, each consisting of $N/2$ bits, as described in section 4.3 of the Twofish AES submission paper. Two of these sets, M_e and M_o in the notation of the paper, consist of the even and odd 32-bit words of key material, respectively, and are used to generate the round subkeys K_j. The third set, S, is derived by applying a Reed-Solomon parity code matrix (RS) to the entire key, and the bits of S are used to define the four key-dependent S-boxes s_i of Twofish. Any two of these three sets is sufficient to determine the entire key.

Given the structures used for generating subkeys and S-boxes, it appears intuitively that each distinct Twofish key results in a distinct cipher. In this tech note we describe some empirical work performed to verify certain uniqueness properties of Twofish keys that strongly support our intuition. In particular, we have proven that:

- No two distinct keys produce an identical sequence of subkeys K_j. In fact, we actually have proven an even stronger result; namely, each distinct M_e results in a unique sequence of A_i values, and each distinct M_o results in a unique sequence of B_i values.

- Each distinct value for S results in a unique round function F. This fact splits all keys into equivalence classes of size $2^{N/2}$ with respect to the round functions.

Given these two results, it is clear that no two keys produce the same sequence of round functions. Unfortunately, this result stops short of proving conclusively that each distinct Twofish key results in a distinct cipher, since it is not impossible theoretically that two ciphers with different round functions could produce an identical cipher. However, such an occurrence seems incredibly unlikely.

*Hi/fn, Inc., 5973 Avenida Encinas, Suite 110, Carlsbad, CA 92008, USA; dwhiting@hifn.com.
†University of California Berkeley, Soda Hall, Berkeley, CA 94720, USA; daw@cs.berkeley.edu.

2 Distinct Subkey Sequences

Let us examine the A_i sequences used in generating subkeys. Similar arguments apply to the B_i sequence, and the same empirical verification has been performed for both sequences. Note that, since the operations used to compute K_{2i} and K_{2i+1} from A_i and B_i are reversible, proving that the A_i and B_i sequences are distinct is more than sufficient to prove that the K_j sequences are distinct.

Note that $A_i = h(2\rho i, M_e)$, where in this case the h function consists of applying the MDS matrix multiplication to the four values $(y0, y1, y2, y3)$ obtained by running the value i through $1 + N/64$ levels of q_0 and q_1, xoring with bits from M_e. Since the MDS matrix is nonsingular, it is easily seen that, for example, a unique sequence for the y_0 values results in a unique sequence for the A_i values, with similar results for y_1, y_2, and y_3.

The sequence of y_0 values for 256-bit keys is

$$q_1[q_0[q_0[q_1[q_1[i] \oplus k_3] \oplus k_2] \oplus k_1] \oplus k_0]$$

where the k_j are different bytes from M_e. Smaller keys result in similar equations with fewer key bytes, so the analysis is almost identical but the empirical search time is much smaller. The question at hand is whether two distinct sets of four key bytes can result in an identical sequence of twenty y_0 values. In fact, we exclude the input and output whitening values to concentrate solely on the round subkeys, so only sixteen values are included in our test (one per round). Clearly, there are 2^{32} such sequences. However, this number can be reduced for the purposes of our search by noting that, since the "outer" mapping q_1 is bijective, it can be removed from the equation, leaving us with

$$q_0[q_0[q_1[q_1[i] \oplus k_3] \oplus k_2] \oplus k_1] \oplus k_0$$

Now the outer xor term k_0 can be effectively removed by creating a related sequence with only fifteen values, where the y_0 values for $i = 5..19$ are xored with the $i = 4$ value. The k_0 terms thus cancel out, so there are only 2^{24} equivalence classes of sequences, speeding up the search dramatically. If all these sequences (each with $15*8 = 120$ bits) are unique, then the y_0 sequence is also guaranteed to be unique.

For 192-bit keys, the y_0 sequence is

$$q_1[q_0[q_0[q_1[i] \oplus k_2] \oplus k_1] \oplus k_0]$$

and for 128-bit keys, it is

$$q_1[q_0[q_0[i] \oplus k_1] \oplus k_0].$$

In a manner identical to that discussed for 256-bit keys, the outer q_1 permutation can be removed for these smaller keys, and the outer k_0 term can similarly be removed by creating the related xor sequence. The number of the remaining sequences is 2^{16} and 2^8, respectively. All these sequences can then be compared across key sizes (with a total of $N_s = 2^{24} + 2^{16} + 2^8$ sequences) to verify uniqueness.

A computer search has been performed over y_0, y_1, y_2, and y_3, for all sequences across all key lengths. It turns out that only 64 bits of each sequence were sufficient to distinguish the sequences. This fact is actually quite encouraging, since there are 120 bits in each sequence, giving some heuristic comfort that the sequences are in fact quite different. Each search requires slightly more than 128 MB memory (i.e., $8N_s$ bytes) to hold all the sequences, which are then sorted and compared to adjacent values to guarantee uniqueness. The test was run on a Pentium computer with only 32 MB of memory, so the search was actually performed in multiple passes, running through all the N_s values several times, on each pass selecting only those values falling into certain bins. For example, using approximately 8 MB of memory, there are sixteen passes, with the m^{th} pass discarding all sequence values for which a fixed four-bit field of the sequence does not equal to m. In addition, the same test was run for the B_i using M_o, with similar results.

The definitive result from these tests is that there are no two distinct keys of any size for which the same sequence of A_i and B_i values is obtained. Thus, the round subkey sequence K_j is unique across keys.

3 Distinct Round Functions

The round function F takes a 64-bit input and produces a 64-bit output, and F is characterized by the four S-boxes s_i and the round keys K_{2r+8}, K_{2r+9}. The S-boxes depend on $N/2$ bits of key material S. Our test for distinctness of the round functions concentrates on only a single bit of the output, which will be sufficient to prove uniqueness.

In particular, consider the value

$$F_1 = (T_0 + 2T_1 + K_{2r+9}) \bmod 2^{32}$$

where $T_0 = g(R_0)$ and $T_1 = g(\mathrm{ROL}(R_1, 8))$. Because the g function involves an MDS matrix multiply, which uses the xor operation, it is difficult to analyze the full F_1 value due to the interaction between operations of different algebraic groups (i.e., xor and addition). However, if we examine only the least significant bit (lsb) of F_1, we can ignore carries, so the operation can be analyzed entirely using xor. Note that this bit does not depend on R_1, due to the multiplication by two in the PHT. Further, the MDS matrix element that maps the S-box output is a simple linear transformation (i.e., multiplication by a GF(256) field element). Thus, for each S-box, the affect of the final fixed permutation (q_0 or q_1) of the S-box and the MDS multiply on the lsb of F_1 is a simple fixed mapping from eight bits to one bit.

Now, consider two keys with distinct values for S. Since the S values are different, at least one of the four S-boxes must have different key material under the two different keys. To prove uniqueness, we simply fix the inputs to the remaining three S-boxes, so that their xor "contribution" to the lsb of F_1 is fixed. Up to a fixed xor constant (based on K_{2r+9} and the other three S-boxes), we are then left with a simple function involving a single S-box that maps eight bits to one bit, with $N/8$ bits of key material used in the S-box. We can remove dependence on the fixed constant by constructing sequences consisting of the xor of two bits in the S-box/MDS lsb output sequence, similar to the method discussed in the previous section. If this sequence of bits is unique across the all $2^{N/8}$ possible key material values for the S-box, then the F function is unique with respect to that S-box. If all four S-boxes are unique in this way, then the F function is unique for each distinct value of S.

To remove the dependence on the constant bit, we performed our search on a modified sequence in which each bit was the xor of two lsbs of the S-box/MDS output values. Since the s_i mapping has 256 inputs, this limits the sequence down to only 255 values, which is still easily sufficient to distinguish between F functions. Let us first consider the 256-bit key case, which obviously involves the longest search. Note that, for example,

$$s_0(x) = q_1[q_0[q_0[q_1[q_1[x] \oplus k_3] \oplus k_2] \oplus k_1] \oplus k_0]$$

where the k_j bytes are a subset of the bytes in S. There are 2^{32} such functions, but, unfortunately, since we are dealing only with the lsb, there is no obvious method to cut down the search time by a factor of 256, as we were able to do in the previous section. Similar sequences were produced for the smaller key sizes, with all sequences for each S-box combined across key sizes in the test, for a total of $N_F = 2^{32} + 2^{24} + 2^{16}$ sequences per S-box.

To avoid a "birthday surprise" collision with N_F sequences, we require more than 64 bits of each sequence. Even with only 64 bits, however, a total of over 32 GB of memory would be required, a size far beyond the budget of this experiment. Thus, this test was also performed in multiple passes, with each pass generating all N_F sequences but discarding those values not falling into the selected bin for the particular pass, as in the previous section. It was found empirically that the performance time was roughly proportional to the number of passes; in other words, the sort/compare time for a given pass was considerably shorter than the $O(2^{32})$ time requires to generate the "filtered" list. For example, on a Pentium computer with slightly over 256MB of available RAM, 128 passes are required, which empirically were completed for a single S-box in slightly less than three days on a 200 MHz Pentium. Each sequence value in the list consisted of 64 bits, with additional sequence bits used to filter out values not to be used in a given pass. For a 128-pass test, this means that seven extra bits were used to help avoid a birthday surprise collision. The filtering code (in C) was carefully optimized so that most of the values to be filtered on each pass were quickly rejected. Since 127 of every 128 values were filtered out on each pass, this simple optimization sped up performance considerably, without requiring particularly fast generation of the 64-bit sequence values to be included.

The test was run on several Pentium computers with various amounts of RAM over a period of about ten days. The results showed that, for each of the four S-boxes, all N_F mappings have a unique lsb sequence. Thus, each F function is unique for each distinct value of S.

4 Conclusion and Future Work

We have empirically proven that every Twofish key leads to a unique set of round constants, and that every string S used to define the S-boxes results in a unique F-function. Although these properties seemed intuitively true, having an exhaustive proof is nonetheless reassuring. Given the structure of Twofish, it appears unlikely that there are any weak keys or significant problems with related keys.

The results in this paper are part of the ongoing work by the Twofish team to analyse Twofish in more detail.