Twofish Technical Report #7

Key Separation in Twofish

John Kelsey*

April 7, 2000

Abstract

In [Mur00], Murphy raises questions about key separation in Twofish. We discuss this property of the Twofish key schedule, and compare it with other block ciphers. While every block cipher has this property in some abstract sense, the specific structure of Twofish makes it an interesting property to consider. We explain why we don't believe this property leads to any interesting attacks on Twofish, and describe what such attacks will have to look like if they do exist.

1 Introduction

In [Mur00], Murphy raises questions about a property in Twofish he calls "key separation." Murphy explains the property as the ability for an attacker to guess 64 bits of a 128-bit Twofish key, and to then be faced with only 2^{64} possible keys with those 64 guessed bits. This property is present in any block cipher with a 128-bit key. The difference between other block ciphers and Twofish is that Twofish uses key-dependent S-boxes, defined by only half of the total key material. Since the quality of the S-boxes is so critical to the security of Twofish, we need to consider attacks in which either:

1. The attacker first guesses S, and then mounts an attack based on his guess.
2. The attacker waits for an especially weak S, and detects this weak S in some way.

In this note, we discuss this property in more detail, first for all block ciphers, and then for Twofish. We discuss the way the property comes about in Twofish, and the necessary mechanics of any attack based on the property. We explain why we don't expect any attack to come from this property, and review some analysis we've done in other technical reports involving this class of attack.

2 The Twofish Key Schedule and Key Separation

To understand Murphy's argument, it is necessary to know something about the Twofish key schedule. In Twofish, the key schedule has three phases:

1. Split the N bit key into three $N/2$ bit pieces: M_e, M_o, and S.
   (a) M_e is taken from the even-numbered key words.
   (b) M_o is taken from the odd-numbered key words.

*Counterpane Internet Security, Inc. kelsey@counterpane.com

(c) S is the result of an RS code applied to M_e, M_o, whose properties make both partial-guessing attacks against the cipher and related-key attacks quite difficult.

2. Use M_e and M_o together to generate the whitening keys and the round subkeys.

3. Use S to define the key-dependent S-boxes used by the cipher during encryption and decryption.

Now, S is half the size of the full key. When a 128-bit key is used, there are 2^{64} possible sets of key-dependent S-boxes available. When a 256-bit key is used, there are 2^{128} possible sets of S-boxes available. Further, one quarter of the bits of S are used for each of the four key-dependent S-boxes. Thus, for Twofish-128, each S-box has 2^{16} possibilities, while for Twofish-256, each S-box has 2^{32} possibilities.

2.0.1 Why Use Key-Dependent S-boxes?

Making S half the size of the full key was a deliberate design decision. Using the full key to define S would have been easy enough, but would have approximately halved throughput in Twofish implementations with zero keying, and would have doubled the time necessary to do the full key schedule precomputation for bulk encryption. Assuming some restrictions on minimal allowed throughput for highly key-agile implementations, this would have required a decrease in the number of rounds of the cipher.

The key-dependent S-boxes in Twofish are part of our security margin, much the same way as the large number of rounds in Serpent or the heterogeneous structure of MARS are part of theirs. Key-dependent S-boxes provide two different kinds of security margin:

1. Any attack that takes note of the specific inputs and outputs (or input and output differences) to the g function must apparently also take note of the S-box contents. This forces attackers into either guessing S (and thus beginning their attack by guessing half the key) or into ignoring the specifics of g for most of the attack.

2. There may be properties of S-boxes we know nothing about, which are as important as differential and linear properties. Knowing nothing about these, we have no way of knowing we haven't chosen especially bad S-boxes with respect to these properties. But unless the majority of random S-boxes are especially bad with respect to these properties, the key-dependent S-boxes will probably ensure that most keys aren't vulnerable to attacks based on these properties. (Note, however, that our key-dependent S-boxes are constructed from fixed S-boxes and XORed-in key bytes. However, these involve far more S-box lookups (three times as many for 128-bit keys), and so may be expected to be less vulnerable to attacks.)

Since the purpose of the key-dependent S-boxes is to provide additional security margin, we saw no reason why the whole key must be involved in defining them. Additionally, the key dependent S-boxes help frustrate related-key attacks.

2.1 Key Separation

The basic argument of Murphy is as follows: An attacker can guess S. For Twofish-128, this leaves only 2^{64} possible subkey sequences that are consistent with this guess. That is, having guessed half the key bits, we're left with half the key bits. This is no different than any other cipher. The concern here is that Twofish uses half its effective key bits in a way that's extremely important for the strength of the rest of the cipher. This compares in some sense to IDEA, whose multiplicative keys are much more important for its ultimate security than its additive keys.

It is important to note the implications of this property. An attacker cannot guess the round subkey sequence with less than a guess of the full Twofish key. The round subkey sequence is derived from M_e and M_o, and it is easy to see that every round subkey is dependent on the full value of both M_e and M_o. However, S is derived from M_e and M_o in a way that ensures that knowledge of any two is sufficient to learn the third. Thus, an attacker who has guessed S doesn't know M_e or M_o, but for each possible value of M_e, he knows what M_o will have to be.

The property does not allow any straightforward division of the key space. For example, it's easy to see that a meet-in-the-middle attack based on this property can't work, since the attack requires guessing first one part of the key, and then another *based on the correctness of the guessed first part.* A bit more analysis, based on the specific way the RS code is used to derive S from M_e, M_o, shows that the best an attacker can do against Twofish-128 is to guess 15/16 bytes. This leaves him with one byte of uncertainty about each round subkey, and two bytes of uncertainty about each of the two whitening keys. Note that this doesn't mean he knows the bytes; at best, he can know 4/8 bytes of each subkey, but he will also know that there are only 256 possible values for any round subkey that are not already specified by his previous guesses.

3 Applying Key Separation to the Key-Dependent S-Boxes

The real relevance of key separation to Twofish involves the great importance of the S-box key, S. Each of the three "halves" into which the Twofish key is broken (M_e, M_o, S) radically affects every round. (For example, a single byte change to M_e or M_o is guaranteed to change every round subkey.) But S determines the S-boxes, the basic nonlinear elements upon which the cipher is based. In [**?**, DGV94], the concept of key-dependent differential and linear characteristics was introduced. It is known that the differential and linear characteristics through the Twofish S-boxes are key-dependent; we have done considerable statistical analysis of the S-boxes for all three key sizes to determine the average and worst-case differential and linear properties of the S-boxes [SKW+98, SKW+99].

The use of key-dependent S-boxes raises one possibility for attack: Wait for users to choose keys with especially weak S-boxes, and mount attacks based on these weak keys. The decision to use only half the key material to define the S-boxes raises another possibility: First guess the S-box key, then attempt an attack based on the guess. These two possibilities need to be addressed.

3.1 Weak S-box Keys

It is known that some S-box keys yield S-boxes with higher probability differential and linear characteristics than others. It is known, for example, that the worst S-box for 192-bit Twofish keys (of all four S-boxes) had one differential with probability 24/256, though the average was about 12/256. This leads to the question of how weak the key can be; that is, what's the best set of S-boxes available for an attacker? This question must ultimately be answered in terms of the difficulty of distinguishing the cipher with this set of S-boxes from a random permutation.

Suppose there is a single S value for 256-bit keys that allows an attack that distinguishes Twofish with that S-box from a random permutation with only 128 basic operations and a few chosen plaintexts. This allows an attack on Twofish-256 which works on 2^{-128} of all keys, and so gives a very small advantage over brute-force search. However, the same advantage is available to an attacker who precomputes encryptions for 2^{128} different keys; he is able to request a few chosen plaintexts, look up the results in a sorted list, and quickly determine the key used, if it is in his list. The difference is only in terms of memory requirements; a table of 2^{128} ciphertexts from the different keys requires larger storage than anything available on Earth. (On the other hand, the precomputation also takes more resources than are currently available. It is unclear which requirement will be the constraining factor on some attacker in the future.)

This places the importance of weak keys of this kind into context. Only a very large class of such weak keys can threaten the security of a cipher *in the context of normal use.* However, such weak keys could raise security problems in some hashing modes, for example. It is therefore worth briefly reviewing our current knowledge about the range of S-boxes available.

In [SKW+98], we reported statistical tests run on our set of possible S-boxes. The worst known S-box had a differential characteristic of 24/256, but there may be slightly worse S-boxes in the 256-bit key case; we were unable to exhaustively search the set of S-boxes for these long keys. We also considered possible bytewise difference patterns; the best difference patterns available required (as a rule) no fewer than six active S-boxes per three rounds. If we imagine a distinguishing attack mounted against the full 16-round Twofish, we may thus consider 30 active S-boxes to be a lower bound. (In fact, the real number is almost certainly much higher in all cases.) Getting the S-box differences to connect involves an interaction of the MDS matrix, PHT, and key addition; the probabilities for individual byte differences are quite low. If the best set of four S-boxes were to all have differential characteristics with probability 30/256, the active S-boxes alone would account for a probability for a 15-round characteristic of about 2^{-92}. If the probability of getting the required pattern of byte differences is around 2^{-10} for each three rounds, then we end up with a total differential characteristic with probability about 2^{-132}. If we also assume a 1/2 probability of getting the right difference in each S-box (which basically means that each S-box's best difference would have to have a Hamming weight of one), the total differential characteristic would have probability of about 2^{-162}. This appears to be the best possible case for an attacker (and thus the worst for Twofish). Note that this isn't a truncated differential; to get the best differential probabilities for this set of S-boxes, we must specify actual differences into the S-boxes. To get the output differences from those S-boxes to align through the MDS matrix to be reused in the next round, we must again specify actual differences, rather than large classes of differences.

Naturally, there are many possible properties that might raise problems with S-boxes, and there's no way to consider all of them. It is, therefore, useful to recall two lessons from analysis we've done on Twofish:

1. The use of MDS matrix multiplication spreads out any property among all the S-boxes fairly quickly. Thus, it seems likely that an attacker must find interesting properties in all four S-boxes to find a real attack.

2. The combination of operations in Twofish makes many simple attacks just fall apart. For example, the one-bit rotations prevent attacks based on lining up byte-oriented differentials cleanly through many rounds; the combination of addition and XOR operations in the F-function means that there are more operations in the cipher that can't be bypassed "for free"; e.g., without decreasing probabilities or biases.

3.1.1 Using a Weak Key If Found

Consider a differential with probability 2^{-100} for Twofish-256 with the worst possible S key. An attacker who wishes to distinguish Twofish from a random permutation using this differential expects to need about 2^{100} chosen plaintext pairs. The attack is expected to work 2^{-128} of the time. The attacker thus expects to request about $2^{100} \times 2^{128} = 2^{228}$ chosen plaintext pairs before he expects his first success. This is enormously less efficient than simply doing the precomputation of the 2^{128} 256-bit Twofish keys with S as their S-box keys, storing the encryptions of the same four chosen plaintexts under each key in a sorted list, and retrieving the key as described above. Thus, while such a weak key, if it existed, might be of some value in attacking hashing modes, it would have no value in recovering encrypted messages.

In general, weak keys that are selected with reasonably low probability (e.g., less than 2^{-32}) are not very interesting in the normal use of ciphers.

3.1.2 Conclusions

Naturally, many other attacks are possible, and we cannot prove that there is not *some* unknown property, for which, by immense bad luck, Twofish has an especially weak S key available. However, we note that the diffusion properties of Twofish make any attack based on such a property likely to require it to occur in all four key-dependent S-boxes. We also note that Twofish already mixes addition, XOR, and rotation along with the S-boxes. Any property in the S-boxes must survive all three operations with some reasonable probability to be worthwhile. We don't expect such a property to exist; there appears to be no more reason to expect such a property to exist for Twofish than for Serpent, RC6, MARS, or Rijndael.

3.2 Attacking Twofish by Guessing S

The use of key-dependent S-boxes in defining the g function means that attackers have essentially two choices:

1. Ignore nearly all specific properties of the g function in an attack.

2. Guess S before mounting the rest of the attack.

Without guessing S, an attacker must base his attack only upon the properties of the rest of the structure, such as the use of the PHT to mix outputs of the two parallel g functions in each round, or the balanced Feistel construction, or the general properties of the MDS matrix multiply step applied after the S-boxes in the g function. Indeed, to backtrack past a single round, an attacker must guess S. We should thus expect that attackers will start by guessing S, and then will mount their attacks based on that guess.

It is interesting to consider the mechanics of such an attack. There are three interesting constraints:

1. Once the attacker has guessed S, the rest of the attack must distinguish the full cipher from a random permutation using less than $2^{k/2}$ work, where k is the key length in bits.

2. If some attack requires specific chosen plaintexts or ciphertexts for each different guess of S, then an attacker must choose a different set of plaintexts or ciphertexts for each S value considered. This can easily require the whole codebook to mount the attack for all possible S values, and will probably require that the whole set of texts requested be stored.

3. The attack needs to actually work for most S values. If it works for only a tiny portion of the possible S values, then it is properly considered as a class of weak keys, rather than as an attack on the cipher as generally used.

Consider an attacker who tries to guess S, and then mount an attack based on his guess. To break Twofish, the attacker must distinguish Twofish with this S-box from a random permutation with less work than is required to brute-force search half the keyspace. And he must do this for reasonably common keys, rather than waiting for the worst possible key to happen to be used. Thus, to attack Twofish-256, he must guess 128 bits, and then distinguish Twofish with these S-boxes from a random permutation in less than 2^{128} work.

Once again, let's consider differential attacks. The average-case differential characteristic for Twofish S-boxes is 12/256. If a distinguishing attack on Twofish requires 30 active S-boxes with probability 12/256 each, the total differential characteristic probability is about 2^{-132} to get through the active S-boxes alone, and so will take more than 2^{128} work or input pairs to detect. This ignores all the other requirements for such a differential characteristic to exist, involving carry-bit propagation, byte difference patterns, etc.

3.2.1 How the Attack Would Be Mounted If It Worked

Suppose we have some property by which we can distinguish Twofish-256 from a random permutation, given knowledge of the S-box key, with 2^{127} work and 2^{70} chosen plaintexts determined by the S-box key. In this case, an attack would require essentially the full plaintext/ciphertext mapping for the key being attacked, to allow the right set of 2^{100} chosen plaintexts for each key. An attack of this kind, if it worked, would give 2^{255} work to break Twofish-256. However, we know of no property by which 16-round Twofish can be distinguished from a random permutation with less work then keysearch, even given the S-box key.

3.2.2 Conclusions

We have been unable to find a way to mount this class of attack successfully against Twofish. However, we would very much like to see more research into this kind of attack. The important requirement to remember is that the attacker, once he has guessed the S-box key (half the total entropy), must now mount an attack on the cipher with less work than guessing the other half of the key. Alternatively, the attacker may guess individual S-boxes, each defined by 1/8 of the total Twofish keyspace, and try to mount an attack on the cipher requiring less work than searching the remaining keyspace.

4 Conclusions

In this note, we have discussed the key separation property of Twofish. We have discussed what this property is, its existence in all block ciphers, and its specific relevance to Twofish. We have discussed the possibility of weak S-box keys making an attack on Twofish possible, and shown some of our reasons for believing no such weak S-box keys exist. We have also discussed the possibility of partial guessing attacks in which the attacker guesses the S-box key first, then tailors an attack to the specific set of S-boxes he's guessed. Again, we have discussed some of our reasons for not believing that such an attack will work.

However, we want to encourage further investigation into the partial-guessing attacks described above. It is our belief that, if an attack is to be found against Twofish, this kind of attack is the most likely place to find it. We expect that any such attack will need to find some serious weakness in the rest of the Twofish structure, or some property of the fixed S-boxes q_0 and q_1 that survives the process of generating key-dependent S-boxes. We have no idea what such a property would look like, if it does exist.

5 Acknowledgements

The author wishes to thank Susan Langford, Sean Murphy, Bruce Schneier, and Doug Whiting for useful conversations and comments.

References

[BB93] I. Ben-Aroya and E. Biham, "Differential Cryptanalysis of Lucifer," *Advances in Cryptology—CRYPTO '93 Proceedings*, Springer-Verlag, 1994.

[DGV94] J. Daemen, R. Govaerts, J. Vanderwalle, "Weak Keys for IDEA," *Advances in Cryptology—EUROCRYPT '93 Proceedings*, Springer-Verlag, 1994.

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

[Mur00] Sean Murphy, "The Key Separation of Twofish," available on NIST AES3 Web site, `http://www.nist.gov/aes`, April 2000.

[SKW+98] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, "Twofish: A 128-bit Block Cipher," *AES Round 1 Technical Evaluation CD-1: Documentation*, National Institute of Standards and Technology, Aug 1998.

[SKW+99] B. Schneier, J. Kelsey, D. Whiting, D. Wagner, C. Hall, and N. Ferguson, *The Twofish Encryption Algorithm, A 128-Bit Block Cipher*, John Wiley & Sons, 1999.