A Note on the Block Cipher Camellia

Yiqun Lisa Yin

NTT Multimedia Communication Laboratories
250 Cambridge Avenue, Palo Alto, CA 94306
yiqun@nttmcl.com

Version 1.1[1] — August 18, 2000

Abstract. Camellia is a block cipher jointly developed by NTT and Mitsubishi in 2000. In this note, we describe some observations on the design of the cipher.

1 Introduction

Camellia [1] is a block cipher jointly developed Nippon Telegraph and Telephone Corporation and Mitsubishi Electric Corporation in 2000. It is one of the proposals submitted to ISO/IEC JTC 1/SC 27 for consideration as an international encryption standard. The joint effort combines expertise in cipher design from both companies. As a result, Camellia certainly bears some nice features of E2 [9] (designed by NTT) and MISTY [5] (designed by Mitsubishi).

In this note, we describe some observations on Camellia in terms of its security. Our analysis is based on the specification of Camellia [1], a short proposal to ISO [11] as well as existing analysis on E2 [10] and MISTY [5]. The main objectives of our analysis is to give some outsider's opinions on the design of Camellia and to contribute to the continued analysis of the cipher.

The note is organized as follows: In Sections 2 through 4 we make some observations and ask a few questions regarding the major components in Camellia, and in Section 5 we draw some concluding remarks. Throughout the note, we use the same terminology and notation as in [1] without restating the precise definition.

2 The *F*-function, *P*-function, and *S*-function

The structure of Camellia is "almost" Feistel. Its round function F adopts the so-called "1-round substitution-and-permutation structure" (or 1-round SPN): The S layer consists of 8 s-boxes each operating on a single byte, and the P layer is a linear transformation from 8 input bytes to 8 output bytes.

[1] This version is based on the earlier note dated April 6, 2000.

2.1 A different view of the s-boxes

There are four different s-boxes called s_1, s_2, s_3, s_4 in Camellia. They are closely related to each other as follows:

$$\begin{aligned}
s_1(x) &= h(g(f(x \oplus a))) \oplus b, \\
s_2(x) &= s_1(x) \lll 1, \\
s_3(x) &= s_1(x) \ggg 1, \\
s_4(x) &= s_1(x \lll 1).
\end{aligned}$$

The functions f and h are linear mappings, g is an inverse over $GF(2^8)$, and a, b are two fixed constants.

The above definition can be rewritten in the following way so that the one-bit rotations are "implicit."

$$s_i(x) = h_i(g(f_i(x \oplus a_i))) \oplus b_i, \text{ for } i = 1, 2, 3, 4. \tag{1}$$

Such a different view might be useful in facilitating certain analysis.

Below, we describe some observations based on Murphy and Robshaw's recent work on one of the AES finalists Rijndael [2]. Rijndael has a single s-box defined as

$$s(x) = L(G(x)) \oplus c,$$

where L is a linear mapping, G is an inverse over $GF(2^8)$, and c is a constant. Murphy and Robshaw [8] showed that the constant c can be incorporated into the key schedule and the linear component L can be re-grouped with the linear transformation layer in Rijndael. In effect, the Rijndael s-box is just an inverse function over the finite field.

The above arguments on Rijndael also apply to Camellia if we use Equation 1 as the formulas for the s-boxes in Camellia. More specifically, the 8 constants a_i, b_i (for $i = 1, 2, 3, 4$) can be incorporated into the key schedule, and the linear mappings f_i, h_i (for $i = 1, 2, 3, 4$) can be re-grouped with the linear transformation P. The re-grouping of linear layers in Camellia is a little more complicated than that of Rijndael. In the case of Rijndael, byte-oriented operations (over $GF(2^8)$) are preserved. For Camellia, the original linear layer P can be described as only exclusive-or of *bytes*, but after re-grouping P needs to be specified using exclusive-or of *bits*, since f_i, h_i are defined at the bit level.

In the specification of Camellia [1], it states that one of the major design rationales for the two linear functions f, h is to make the s-boxes more complicated. However, we have seen that the linear functions can be deprived from the s-boxes to combine with the linear layer instead. The implication of such a different view of the overall S-P layer in Camellia is not yet clear, and perhaps some further analysis following techniques in [8] should be considered.

Finally, it seems that the two constants a, b in the s-box of Camellia are not needed in terms of the security of the encryption function of Camellia since they can be integrated into the key schedule.

2.2 The choice of the 1-round SPN structure

One of the major changes from E2 [9] to Camellia is that the 2-round SPN structure for the round function is replaced by the 1-round SPN structure. Here we consider some pros and cons for such a choice by analyzing the upper bound on the maximum differential and linear probabilities. To simplify discussions, we will just focus on differential probabilities.

Recently, Kanda presented some new results on how to evaluate 1-round SPN round functions in terms of differential and linear attacks [4]. Let $y = P(x)$ be the linear layer in an SPN round function. He defined the *differential branch number* P_d of P as

$$P_d = \min_{\text{all } (x,x') \text{ s.t. } x \neq x'} (\#\text{changed input bytes} + \#\text{changed output bytes}).$$

He showed that for Feistel ciphers with 1-round SPN round function, the minimum number of active s-boxes in r rounds is roughly proportional to $\frac{r}{4} \times P_d$. The analysis was actually quite involved.

The minimum number of s-boxes in Feistel ciphers with 2-round SPN structure can be quite easily analyzed using two simple facts. First, in any three consecutive rounds of a Feistel cipher, there are at least two rounds with non-zero input difference for any plaintext difference. Second, if input difference to a round is non-zero in 2-SPN, then the minimum number of active s-boxes is P_d in that round. Combining the above two facts, we can conclude that the minimum number of active s-boxes in r rounds is roughly proportional to $\frac{2r}{3} \times P_d$ in Feistel ciphers with 2-round SPN structure.

Based on the above discussion, we know that the minimum number of active s-boxes over r rounds for the 2-round SPN structure is about $\frac{8}{3} = 2.33$ times as much as that for the 1-round SPN structure. On the other hand, the total number of s-boxes over r rounds the former is only twice as much as the latter. So it seems that choosing a 2-round SPN structure may be better than a 1-round SPN in a Feistel cipher, at least asymptotically. In practice, though, some boundary conditions and implementation issues might make the two comparable the latter more favorable.

2.3 Camellia vs. E2

The most effective existing attack on E2 is truncated differential cryptanalysis [7]. We'd like to know how effective the attack is on Camellia.

It would also be interesting to consider variants of Camellia. For example, we can analyze Camellia without the irregular rounds, or Camellia with the P function being the same as that in E2, or Camellia with a 2-round SPN structure, or Camellia with all the s-boxes set to s_1. In terms of truncated differential cryptanalysis, would the above variants be stronger, comparable, or weaker than the original Camellia? Such a study can help to better understand each components of the cipher.

3 The FL-function and FL^{-1}-function

The pair of functions FL and FL^{-1} are "inserted" between every six rounds of the encryption routine to provide some non-regularity across rounds. One of the obvious goals for such a design is to thwart future unknown attacks.

We observe that the function FL can be viewed as a two-round 64-bit Feistel network with the first round function set to $F_1 = \cap\lll 1$ and the second one set to $F_2 = \cup$, and similarly for FL^{-1}. We will often refer to the pair (FL, FL^{-1}) as an *irregular round*. Such an irregular round is much simpler compared with the regular round function F of Camellia.

Below, we analyze the effect of irregular rounds in terms of several known attacks. In our analysis, we assume that there is an attack of a certain type on Camellia when the irregular rounds are omitted and try to see what strength the irregular rounds can add to the cipher.

Let us first consider the two bitwise logical operations used in FL and FL^{-1}.

$$\begin{aligned} y &= x \cap roundkey, \text{ and} \\ y &= x \cup roundkey. \end{aligned}$$

For an n-bit vector x, we use $x[i]$ to denote the ith bit of x. Some basic properties for the two operations are summarized in Table 1 and Table 2.

$\cap$	$roundkey[i]$	input/output difference	affect analysis
	0	$\Delta x[i] = 1 \rightarrow \Delta y[i] = 0$	yes
	1	$\Delta x[i] = 1 \rightarrow \Delta y[i] = 1$	no
$\cup$	$roundkey[i]$	input/output difference	affect analysis
	0	$\Delta x[i] = 1 \rightarrow \Delta y[i] = 1$	no
	1	$\Delta x[i] = 1 \rightarrow \Delta y[i] = 0$	yes

Table 1. How $\cap$ and $\cup$ affect differential cryptanalysis

$\cap$	$roundkey[i]$	input/output value	affect analysis
	0	$y[i] = 0$, independent of $x[i]$	yes
	1	$y[i] = x[i]$	no
$\cup$	$roundkey[i]$	input/output value	affect analysis
	0	$y[i] = x[i]$	no
	1	$y[i] = 1$, independent of $x[i]$	yes

Table 2. How $\cap$ and $\cup$ affect linear cryptanalysis

3.1 Differential and truncated differential cryptanalysis

For a random round key, the bit $roundkey[i]$ is either 0 or 1 with probability 1/2. From Table 1 we know that a difference $\Delta x[i]$ may be canceled by the corresponding key bit $roundkey[i]$. In particular, if Δx has Hamming weight h then we have $\Delta y = 0$ with probability 2^{-h} for a random round key.

In the case of truncated differentials, there is a similar effect. Suppose Δx is n-bit long. If $\Delta x \neq 0$, then $\Delta y = 0$ with probability $\frac{1}{2^n} \sum_{i=1}^{n} \binom{n}{i} 2^{-i}$ for a random round key. For example, if we choose $n = 8$ (for byte characteristics), then the above probability is about 0.1.

Therefore, the differential paths as well as their associated probabilities vary from one key to another when irregular rounds are inserted. On one hand, this could make the analysis more difficult since the differential paths are key dependent. On the other hand, one might be able to take advantage of such key dependencies and try to deduce the round keys used in the irregular rounds by detecting which differential paths are valid.

Overall, it is not very clear whether such effects from the two logical operations will make it easier or harder to construct useful differential characteristics or truncated differentials.

3.2 Impossible differential cryptanalysis

As we have just seen, when an irregular round is inserted between regular rounds, the $\cap roundkey$ and the $\cup roundkey$ operation can change existing differential paths and their probabilities. So the irregular rounds also make impossible differential paths key dependent. Since the associate probability for an impossible differential path is *zero*, any change in the probability makes it no longer an impossible differential path. Therefore, for most round keys, the use of FL and FL^{-1} will destroy impossible differential paths, making impossible differential cryptanalysis infeasible.

3.3 Linear cryptanalysis

Now we take a look at Table 2. Depending on the value of $roundkey[i]$ and the logical operation, the value of the output bit $y[i]$ can be *independent* of the input bit $x[i]$. If this happens, the pair of bits $(x[i], y[i])$ cannot be included in any linear approximation. Therefore, for a random round key on average about half of the $(x[i], y[i])$ pairs cannot be use to form linear approximations. This makes linear approximations highly key dependent, and more importantly, it really limits the ways that linear approximations can be constructed.

By similar argument, the use of both $\cap$ and $\cup$ will also reduce the effect of linear hulls (if any) since many of the linear hull paths are destroyed.

Overall, the irregular rounds should provide some extra strength for the cipher against linear cryptanalysis.

3.4 Small avalanche effect

Here we want to point out some mathematical properties of the FL function. Following the notation in [1], let X_L, X_R be the two 32-bit inputs to FL, and let Y_L, Y_R be the two 32-bit outputs from FL. We use $0^t x$ and $1^t x$ to denote 32-bit binary vectors in which the first $t \leq 32$ bits are set to all 0s or 1s, respectively.

We observe that no matter what the values of the two round keys are in FL, the inputs $(X_L, X_R) = (0^{32}, 1^{32})$ always produce the outputs $(Y_L, Y_R) = (1^{32}, 1^{32})$. That is, for such input values, the two round keys have no effect to the round.

More generally, we have the property that any inputs of the form $(X_L, X_R) = (0^t x, 1^t y)$ always produce outputs of the form $(Y_L, Y_R) = (0^{t-1}x', 1^{t-1}y')$. So only the lower $(31 - t)$ bits of two round keys have an impact on the results of the outputs.

The above properties of FL show that for certain input values, the function FL can have a very small avalanche effect. It is not clear yet whether such properties can be used in some attacks.

3.5 Remarks

¿From the above analysis, we find some quite interesting properties of the irregular round (FL, FL^{-1}). Sometimes, the round key bits can cancel the corresponding input bits, making certain differential and linear paths key *dependent*. On the flip side, the input bits can also cancel the corresponding round key bits, making certain output bits key *independent*.

Such properties are generally not possessed by the regular round function F in Camellia for which the round keys are exclusive-ored with the inputs. It can be expected that the different ways of how round keys are used in regular and irregular rounds will help prevent future attacks.

4 The key schedule

The key schedule of Camellia is quite simple and resembles that of MISTY [5] to a certain degree. The procedure can be divided into two steps. First, derive one (or two) 128-bit key materials K_A (and K_B) from the original secret key K. (Let us call K, K_A, K_B as the *subkeys*.) Second, generate all round keys by rotating the subkeys K, K_A, K_B by various amounts. The key schedule is slightly different for 128-bit keys and 192/256-bit keys.

Here, we will focus on the key schedule of Camellia for 128-bit keys. In this case, the size of the subkeys (K, K_A) is 256 bits, and 26 64-bit round keys are generated — 13 of them are rotations of K and the other 13 the rotation of K_A.

4.1 Structures in round keys

The round keys are rotations of K or K_A with rotation amounts differing by 15 or 17. The rotation can result in some structures among the round keys.

First, let G be some bit pattern (e.g., consecutive 0s of a certain length) that occurs with probability p for a random 64-bit word. If both K and K_A have pattern G (which happens with probability p^2), then *all* of the round keys will have patten G (or its rotated form). Note that if the 26 round keys are independent, then the probability that all of them possess the same pattern G is only p^{26}, which is much smaller than p^2. We are not sure if one can take advantage of such a structure in the round keys in a real attack yet. But if one can find some pattern G of the round key for which attacking a single round function with such a key is easier than random round keys, then from the above analysis we know that a fair percentage of the keys will be weak under this attack.

Second, we observe that if K or K_A has a repetitive pattern of length 15 or 17, the some of the round keys can be *exactly* the same. That is, the same round key may be used in different round. The property does not seem to cause any immediate problem especially because it happens with very small probability.

4.2 Deriving the secret key from round keys

As we can see, the major design goals for the key schedule in Camellia is fast speed, small RAM requirement, and compact hardware design. So "one-wayness" was not a concern, but let us just quickly analyze the amount of work needed to derive the original secret key K from some round keys.

Assume that some bits of a round key k_i are obtained (from an attack). If k_i is a rotation from (half of) K, then obviously the key space of K is immediately reduced. Or, if we can obtain two round keys k_i, k_j that allow us to recover all bits of K_A, then we can find K much faster than exhaustive search. In particular, it is not hard to see that one can derive K from K_A in 2^{64} steps by guessing either the left half or the right half of K.

4.3 Related-key attacks

There are probably two problems to consider here. First, given two related secret keys K and K', what relation can one get for the corresponding K_A and K'_A? Second, given two sets of round keys that are related to each other, how can one attack Camellia?

In Camellia, the subkey K_A is derived from K using a 4-round Feistel network. The round function is just the F-function that is used in encryption and the round keys are fixed constants. We observe that if ΔK is set to $(0, \Delta)$ for a 64-bit Δ, then after 2 rounds and xoring K, the input difference to the third round becomes $(\Delta', 0)$ where $\Delta' = F(\Delta)$. Hence, the effect 4-round Feistel network is really just 2 rounds. It seems that for any $\Delta' \neq 0$, most of the bytes (exactly how many?) of K_A will differ even after 2 rounds of F. So even when K and K' are closely related, K_A and K'_A are likely to differ a lot.

Overall, we have not been able to come up with a related-key attacks even on the reduced round version of Camellia.

4.4 Camellia vs. MISTY

For MISTY, four criteria were set up for designing the key schedule.

1. The size of secret key K is 128 bits.
2. The size of subkey materials is 256 bits.
3. Every round is affected by *all* bits of K.
4. Every round is affected by as many subkey bits.

Now let us consider again the key schedule of Camellia for 128-bit secret key. We see that criteria 1 and 2 above are also satisfied, but criteria 3 and 4 are not. In particular, half of the round keys (rotations of K) only depend on half of the bits of K. Even though we cannot see any problem with this, but it seems to us that criterion 3 is a good property for key schedule to have in general. Since Camellia certainly adopts some design ideas from MISTY in key schedule, we wonder if this criterion was intentionally ignored due to efficiency concerns.

5 Conclusion

We have described our initial observations on the major components of Camellia. We have focused our analysis mostly on the key schedule and on the FL, FL^{-1} functions, since we think they are relatively new components. We have tried to compare the components in Camellia with existing components in E2 and MISTY and raise some questions on the design rationale.

We have not yet considered the implementation aspect of Camellia. We note that in [11] it is mentioned that Camellia achieves the smallest hardware design (only 10K gates) among all existing block ciphers. In comparison, the most hardware-compact AES finalist still needs about 400K gates according to [3]. So we feel that Camellia can be a nice complement cipher to the AES winner(s). (This could be a good marketing point to make.)

Finally, we wish Camellia the best in various standardization effort.

References

1. K. Aoki, T. Ichikawa, M. Kanda, M. Matsui, S. Moriai, J. Nakajima, T. Tokita. *Camellia – a 128-bit block cipher suitable for multiple platforms (Extended abstract).* To appear in SAC 2000.
2. J. Daemen and V. Rijmen. *AES proposal: Rijndael.* Version 2. 1999.
3. T. Ichikawa, T. Kasuya, M. Matsui. *Hardware Evaluation of the AES Finalists.* In Proceedings of the third AES conference. March 2000.
4. M. Kanda. *Practical security evaluation of Feistel ciphers with SPN round function against differential and linear attacks.* To appear in SAC 2000.
5. M. Matsui. *New Block Encryption Algorithm MISTY.* In Proceedings of the 4th Fast Software Encryption Workshop, 1997.
6. M. Matsui and T. Tokita. *Cryptanalysis of a reduced version of the block cipher E2.* In Proceedings of the 6th Fast Software Encryption, March, 1999.

7. S. Moriai, M. Sugita, K. Aoki, and M. Kanda. *Security of E2 against truncated differential cryptanalysis.* In Proceedings of the 6th International Workshop on Selected Areas in Cryptography. August 1999.
8. S. Murphy and M. Robshaw. *New observations on Rijndael.* Research manuscript. August 7, 2000. Available at `http://isg.rhbnc.ac.uk/mrobshaw/`.
9. Nippon Telegraph and Telephone Corporation. *Specification of E2 – a 128-bit Block Cipher.* 1999. Available at `http://info.isl.ntt.co.jp/e2/`.
10. Nippon Telegraph and Telephone Corporation. *Supporting document on E2.* 1999. Available at `http://info.isl.ntt.co.jp/e2/`.
11. Nippon Telegraph and Telephone Corporation and Mitsubishi Electric Corporation. *A Contribution on New Work Item Proposal on "Encryption Algorithms".* March 10, 2000.