Camellia オープンソース解説資料 (C版 Version 1.1)

NTT 情報流通プラットフォーム研究所

平成 18 年 7 月 12 日

目 次

1 オープンソースコード

1.1 はじめに

本オープンソースコードは 1block(128bit) のみを暗号化・復号することが出来ます。また、仕様書の Appendix C1.1, C1.2, C1.2, C1.8, C2.1, C2.4, C2.7 の最適化手法が実装されています。この文書はオープンソースコードのそれぞれ関数の簡単な使用法をまとめます。

uint32_t が定義されていない環境では下記の部分にご注意ください。

```
#if defined(_MSC_VER)
typedef unsigned char uint8_t;
typedef unsigned int uint32_t;
typedef unsigned __int64 uint64_t;
#else
#include <inttypes.h>
#endif
```

コンパイルを行う環境と実行する環境の ENDIAN が異なる場合は、下記の部分にご注意ください。

```
#if __i386__ || __alpha__ || _M_IX86 || __LITTLE_ENDIAN || __LITTLE_ENDIAN__
#define L_ENDIAN
#else
#undef L_ENDIAN
#endif
```

1.2 Camellia 関数概要

この Camellia ライブラリでは、

- 128bit
- 192bit
- 256bit

の三種類の鍵長を選択することが出来ます。また、「入力する鍵」、「拡大鍵」、「平文」、「暗号文」、「復号文」の全ては unsigned char 型の配列で宣言されています。

拡大鍵の生成　暗号化・復号を行う前に、必ず拡大鍵の生成を行わなくてはなりません。　3
一度拡大鍵を生成すれば、同じ鍵で暗号化・復号を行う限り、再度拡大鍵の生成を行う必要はありません。Camellia_Ekeygen で、鍵長 keyBitLength の鍵のポインタ rawKey を入力し、拡大鍵 keyTable に代入します。

プロトタイプ宣言：

```
void Camellia_Ekeygen
(
    const int keyBitLength,
    const unsigned char *rawKey,
    KEY_TABLE_TYPE keyTable
);
```

```
        keyBitLength : 鍵長 = {128,192,256}
        rawKey       : 鍵データへのポインタ
        keyTable     : 拡大鍵 (KEY_TABLE_TYPE 型)
```

keyBitLength で指定した長さの鍵データのポインタを rawKey に与えてください。

暗号化　暗号化を行う前に使用する鍵で Camellia_Ekeygen を行っている必要があります。Camellia_EncryptBlock で、Camellia_Ekeygen で作られた拡大鍵 keyTable を使用して、平文 plaintext を暗号化し、暗号 cipherText に代入します。

プロトタイプ宣言：

```
void Camellia_EncryptBlock
(
    const int keyBitLength,
    const unsigned char *plaintext,
    const KEY_TABLE_TYPE keyTable,
    unsigned char *cipherText
);
```

```
        keyBitLength : 鍵長 = {128,192,256}
        plaintext    : 平文のポインタ
        keyTable     : 拡大鍵 (KEY_TABLE_TYPE 型)
        cipherText   : 暗号化された文のポインタ
```

Camellia_EncryptBlock は plaintext から 128bit(16byte) の領域を暗号化した結果を、ciphertext から 128bit(16byte) の領域に格納します。必ず plaintext および ciphertext から 128bit(16byte) の領域を確保してから動作させて下さい。

復号　復号を行う前に使用する鍵で Camellia_Ekeygen を行っている必要があります。Camellia_EncryptBlock で、Camellia_Ekeygen で作られた拡大鍵 keyTable を使用して、暗号文 cipherText を復号し、復号文 plaintext に代入します。

プロトタイプ宣言：

```
void Camellia_DecryptBlock
(
    const int keyBitLength,
    const unsigned char *cipherText,
```

```
    const KEY_TABLE_TYPE keyTable,
    unsigned char *plaintext
);
```

```
keyBitLength : 鍵長 = {128,192,256}
cipherText   : 暗号文のポインタ
keyTable     : 拡大鍵 (KEY_TABLE_TYPE 型)
plaintext    : 復号された文のポインタ
```

必ず plaintext および ciphertext から 128bit(16byte) の領域を確保してから動作させて下さい。

1.3 サンプルプログラム

```
const int keyBitLength = 128;
KEY_TABLE_TYPE keyTable;
unsigned char rawKey[] = {0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef,
                0xfe, 0xdc, 0xba, 0x98, 0x76, 0x54, 0x32, 0x10,
                0x00, 0x11, 0x22, 0x33, 0x44, 0x55, 0x66, 0x77,
                0x88, 0x99, 0xaa, 0xbb, 0xcc, 0xdd, 0xee, 0xff};
unsigned char pt[]     = {0x01, 0x23, 0x45, 0x67, 0x89, 0xab, 0xcd, 0xef,
                0xfe, 0xdc, 0xba, 0x98, 0x76, 0x54, 0x32, 0x10};
unsigned char ct_buf[16];
unsigned char pt_buf[16];
/* 拡大鍵の生成 */
Camellia_Ekeygen(keyBitLength, rawKey, keyTable);
/* 暗号化 */
Camellia_EncryptBlock(keyBitLength, pt,     keyTable, ct_buf);
/* 復号 */
Camellia_DecryptBlock(keyBitLength, ct_buf, keyTable, pt_buf);
```

参考文献

[1] 青木 和麻呂, 市川 哲也, 神田 雅透, 松井 充, 盛合 志帆, 中嶋 純子, 時田 俊雄, “128 ビットブロック暗号 Camellia アルゴリズム仕様書 (第二版)” , http://info.isl.ntt.co.jp/crypt/camellia/dl/01jspec.pdf , 2001