１２８ビットブロック暗号 *Camellia*

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 †

*NTT 情報流通プラットフォーム研究所
〒 239-0847 神奈川県 横須賀市 光の丘 1–1
{maro, kanda, shiho}@isl.ntt.co.jp

†三菱電機株式会社 情報技術総合研究所
〒 247-8501 神奈川県 鎌倉市 大船 5–1–1
{ichikawa, matsui, june15, tokita}@iss.isl.melco.co.jp

あらまし　本稿では，128 ビットブロック暗号アルゴリズム *Camellia* の仕様と設計思想の概要を報告する．本アルゴリズムでは，AES(Advanced Encryption Standard) と同じ，データブロック長が 128 ビット，秘密鍵長が 128 ビット，192 ビット，256 ビットのいずれかを利用できる．*Camellia* は，データランダム化部に 128 ビット鍵では 18 段 Feistel 構造を，また 192 ビット鍵と 256 ビット鍵では 24 段 Feistel 構造を採用している．ラウンド関数の構造は SPN 構造であり，さらに 6 段ごとに補助変換関数 FL およびその逆関数 FL^{-1} が挿入されている．鍵生成部では，ハードウェア規模を削減するために，データランダム化部と同じラウンド関数を利用した 2 段 Feistel 構造を用いて中間鍵を生成している．各ラウンド関数に挿入される拡大鍵は，この中間鍵と元の秘密鍵とを指定された手順にしたがって変換したものである．
このアルゴリズムの特徴は，安全性の観点では差分攻撃や線形攻撃に対して十分な安全性を有することを示した点，実装の高速性や柔軟性の観点ではバイト単位の処理を基本に実現可能なようにした点が挙げられる．この結果，十分な安全性を有しながら，ソフトウェア・ハードウェア問わずに高速かつ柔軟な実装が可能なマルチプラットフォーム対応型の暗号となっている．具体的には，ソフトウェアでは Pentium III (800MHz) 上で約 300Mbps 以上（Assembly）の処理速度を有しており，またハードウェアでは暗号化回路を約 10Kgate で実現できる．

和文キーワード　ブロック暗号, 暗号設計, *Camellia*

Camellia — A 128-bit Block Cipher

Kazumaro AOKI * Tetsuya ICHIKAWA †
Masayuki KANDA * Mitsuru MATSUI †
Shiho MORIAI * Junko NAKAJIMA † Toshio TOKITA †

*NTT Information Sharing Platform Laboratories
1–1 Hikarinooka, Yokosuka-shi, Kanagawa 239-0847, Japan

†Information Technology R&D Center
Mitsubishi Electric Corporation
5–1–1 Ofuna, Kamakura-shi, kanagawa 247-8501, Japan

Abstract　In this paper, we propose a 128-bit block cipher *Camellia*, with a key size of 128/192/256-bits. *Camellia* is an 18-round Feistel cipher for 128-bit keys, and 24-round for 192/256-bit keys. In *Camellia*, the round function uses the SPN structure, and additional FL/FL^{-1}-functions are inserted every 6 rounds. To reduce hardware size, the key schedule shares 2-round Feistel structure with data randomizing part for generating intermediate keys. All subkeys are produced from the intermediate keys and the original secret key.
We confirm that *Camellia* has enough security against differential and linear cryptanalysis, in particular. Furthermore, *Camellia*, which is a byte-oriented cipher, realizes high speed encryption and excellent flexibility on both any software environments and hardware implementations. *Camellia* achieves more than 300 Mbits/sec on Pentium III (800 MHz) using assembly language, and approximately 10 Kgates for hardware.

英文 key words　Block cipher, Design strategy, *Camellia*

1 はじめに

本稿では，128ビット共通鍵ブロック暗号アルゴリズム *Camellia* の仕様と設計思想の概要を報告する．このアルゴリズムでは，AES (Advanced Encryption Stardard) の基本要求条件 [13] と同じ，データブロック長が128ビットであり，秘密鍵長が128ビット，192ビット，256ビットのいずれかを利用できる．

Camellia の設計目標は，(1) 安全性，特に差分攻撃 [2] および線形攻撃 [9] に対して十分な耐性を有することを示すこと，(2) 高速性，並列処理が可能で簡素な構造および演算を採用することにより暗号化速度の高速性を実現すること，(3) 柔軟性，特にマルチプラットフォーム対応型を目指し，ソフトウェア・ハードウェアとも効率的かつ柔軟な実装可能性を有すること，の3点を満たすことである．

本アルゴリズムの概要および特徴は以下のとおりである．

- データランダム化部は，128ビット鍵では18段Feistel構造，192/256ビット鍵では24段Feistel構造である．

- ラウンド関数は，64ビットの入力データに対し64ビット拡大鍵を用いて64ビットのデータを出力するような全単射関数である．この関数は，すべてバイト単位処理が可能であるようなSPN構造として構成されており，また s-box には8ビット入出力の全単射テーブルを使用している．

- 補助変換として，64ビットの入力データに対し64ビット拡大鍵を用いて64ビットのデータを出力するような全単射関数 FL/FL^{-1} を6段ごとに挿入する．この関数は，論理積，論理和，ビットローテーション, 排他的論理和で構成される．なお，FL 関数とその逆関数である FL^{-1} 関数を並列位置に置くことによって，Feistel型暗号の特徴である暗号化処理と復号処理の同一性[1]を失わないようにしている．

- 鍵生成部では，ラウンド関数を共用した2段Feistel構造を用いて中間鍵を生成する．具体的には，128ビット鍵からは128ビット中間鍵を，192[2]/256ビット鍵からは2つの128ビット中間鍵を生成する．各段のラウンド関数に挿入される拡大鍵は，この中間鍵と元の秘密鍵とを指定された単純な手順にしたがって変換したものである．

- 安全性に関しては以下のとおりである (第2.3節及び第2.4節を参照のこと)．
  - データランダム化部は，少なくとも16段で差分攻撃・丸め差分攻撃 (Truncated differential attack)[6, 12]・線形攻撃に対して，十分な安全性を有するように設計されている．
  - s-box では，差分攻撃・線形攻撃・高階差分攻撃 [6, 5] に対して最良の安全性を有するように設計されている．また，補間攻撃 [5] などに対する耐性も考慮している．
  - ラウンド関数では，差分攻撃・線形攻撃に対してバイト単位処理の構造の中では最良の安全性を有するように線形変換層が設計されている．
  - 補助変換では，スライド攻撃 [3] や様々な解読法に対する安全性が向上するように期待して設計されている．
  - 鍵生成部では，関連鍵攻撃 [1] に対しても安全であるように考慮して設計されている．

Camellia の暗号化処理速度については， Pentium III (800MHz) 上で300 Mbits/sec以上（アセンブラ言語）を達成した．ハードウエア実装では，暗号化回路を約10 Kgateで実現できる．このサイズは128ビットブロック暗号では世界最小クラスである．

2 設計方針

2.1 設計基準

Camellia を設計するにあたり，我々は以下のような設計目標を立てた．

1. 安全性，特に差分攻撃・丸め差分攻撃・線形攻撃に対して十分な耐性を有することを示すこと
2. あらゆるプラットフォーム上で高速かつ柔軟なソフトウェア実装が可能であること
3. ハードウェア実装で十分な高速性と小型化が可能であること

新しいブロック暗号を設計する上で最大の要求条件は “安全である” ことである．ブロック暗号に対する解読法はいくつか知られており，その中でも汎用的で最も強力な解読法である差分攻撃と線形攻撃に対しては，設計者が必ず対策を考慮しなければならない．そこで，我々は，まず設計時において差分攻撃・丸め差分攻撃・線形攻撃に対して十分な耐性を有するようにラウンド関数の設計および全体構造の段数の決定などを行ない，その後，それ以外の解読法，例えば高階差分攻撃や補間攻撃などに対する安全性の検証を行なうという2段階の設計手順を取った．

ソフトウェアでの性能において，最近提案される多くのブロック暗号，特に128ビットブロック暗号では，特

[1] 拡大鍵の挿入順序を変えるだけで暗号化処理も復号処理も同一回路で実行できること

[2] 192ビットの時は元の秘密鍵のうち64ビットを利用してパディングデータを作り，256ビット鍵として扱う

定の仕様のプロセッサ（例えば Pentium II/III/ Celeron 等）で最高の速度を達成するように設計されているため，しばしばそれ以外のプロセッサでは性能が大きく低下することがある．これに対して，我々は，暗号の使用環境が急速に広がっていることに鑑み，あらゆるプロセッサで適度な高速性を実現できるマルチプラットフォーム対応型の暗号を目指す観点から，特定のプロセッサのみで高速な命令は *Camellia* に採用しない方針を取った．

ハードウェアに関しては，従来の 128 ビットブロック暗号の多くはソフトウェア実装を前提として実現されているために，ハードウェア実装で十分な高速性と小型化を両立させることは困難である．そこで，我々は，十分な高速性を出しながら，128 ビットブロック暗号のハードウェア実装としては世界最小クラスとなる暗号化回路約 10 Kgate という小型化を実現するという目標を立てた．このサイズを達成するには，データランダム化部と鍵生成部とを共有化した上で，かつラウンド関数を 5 Kgate 程度以下で実現するというほどの極めて厳しい制約条件を満たさなければならない．

2.2 基本演算要素の検討

採用すべき基本演算要素を決定するために，演算要素ごとに分類し，それぞれの項目について安全性やソフトウェア・ハードウェアでの実用性を検討した．

- 論理演算
  AND, OR, XOR のような論理演算は，ソフトウェアでもハードウェアでも高速に実現できるため，実用上大変都合がよい．しかし，安全性をこれらの演算単体に求めることは困難である．
- 算術演算
  データランダム化に一定の寄与があるため，多くのブロック暗号に採用されている演算要素である．特に，32 ビットの加減算と乗算は現在主流の 32 ビットプロセッサで高速に実行でき，かつデータの撹拌効果が論理演算よりも優れているため，従来の 128 ビットブロック暗号の多くに採用されている．一方，これをハードウェアで実現するとなると，回路規模が大きく，かつ処理速度が遅いという欠点がある．また，ソフトウェア実装でも，実装するプロセッサに含まれていない算術演算（例えば 8 ビットプロセッサ上での 32 ビット算術演算など）が使われると途端に性能が低下する．さらに，差分攻撃や線形攻撃に対する安全性評価が難しくなる場合がある．
- テーブル参照
  テーブル参照の効率はソフトウェアで行なう場合メモリアクセススピードに大きく依存する．このことは，ソフトウェアにおいては，テーブルの中身ではなく，テーブルのサイズに実装効率が依存することを意味する．一方，ハードウェアでのテーブル参照は，テーブルの内容によっては直接論理回路を組むことによって高速かつ小型化が実現できる場合がある [10]．このことは，ハードウェアにおいては，テーブルの中身が実装効率に依存することを意味する．また，安全性の点からは，一度のテーブル参照でランダム性を大きく上げることができるという長所がある．

これらの考察と前節の設計基準から，我々は以下のような設計方針を採用した．

A) テーブル参照と論理演算を使用することとし，算術演算は使わない

B) ソフトウェアでの実装効率性およびマルチプラットフォーム性の観点から 8 ビットを基本単位とする

C) ハードウェアでの実装柔軟性の観点からテーブルはハードウェア用に設計されたものを使用する

D) 暗号文と平文のバイト単位の関連を崩すため補助変換において固定ビットローテーションを使用する

2.3 データランダム化部

本節では，*Camellia* のデータランダム化部の設計概要を述べる．

2.3.1 ラウンド関数

ラウンド関数の設計にあたっては，*E2*[7] のラウンド関数で使用した 2 段 SPN 構造から，より構造が単純な SPN 構造に変更したものの，両者に同じように使われる線形変換層の設計において *E2* の設計指針をそのまま踏襲した．すなわち，*Camellia* の線形変換層 (以下，P 関数と呼ぶ) も *E2* のときと同じように全てバイト単位の排他的論理和のみで構成された簡素な構造で構成されている．安全性に関して，s-box の最大差分・線形確率を p_s, q_s とすると，連続する 8 段のラウンド関数における最大差分特性・線形特性確率 p, q は，$p \leq p_s^{11}$, $q \leq q_s^{11}$ を満たす．

P 関数の具体的な設計法などについては文献 [8] を参照されたい．なお，P 関数の選定にあたっては，*E2* での実装・設計経験を生かし，マルチプラットフォーム対応型暗号を目指すため，32 ビットプロセッサ用，ハイエンド IC カード用，ローエンド IC カード用，小型ハードウェア用と異なるタイプの実装が可能となるようなものを選択した．これにより，実装環境に応じて最適な実装方法を使い分けることが可能になる．なお，この詳細については別稿にて報告する．

2.3.2 s-box

8 ビット入出力の s-box であるならば，テーブル参照でソフトウェア実装する限り，テーブルの中身によっ

て処理速度が変わることはない．しかし，ハードウェア実装においては，回路規模や処理速度はテーブルの中身に大きく依存する．そこで，*Camellia* では，安全でかつハードウェア向きの s-box という観点から，GF(2^8) 上の逆数関数とアフィン変換の組合せにより s-box を実現した．

GF(2^8) 上の逆数関数は，安全性の面では最大差分・線形確率に対して最良と予想されている $p_s = q_s = 2^{-6}$ を，また高階差分攻撃に対しても最良次数である 7 次を実現している全単射関数である．また，ハードウェア実装の面では，演算基底の取り方を工夫することによって，部分体 GF(2^4) 上の演算を用いて構成することも可能である [11]．これによって，コンパクトなハードウェアを実現することができる．

アフィン変換は，逆数関数の持つ GF(2^8) 上の代数的な性質を壊す性質があるため，逆数関数とアフィン変換とを組み合わせることによって，補間攻撃などの代数攻撃に対する安全性の向上が期待できる．さらに，アフィン変換を複数用意することによって，同じ逆数関数を使っていても，実効的に複数の s-box を構成しているようにすることが可能である．*Camellia* では，GF(2^8) 上の逆数関数と 4 種類のアフィン変換を組み合わせることによって，4 つの s-box として，あるいは 1 つの s-box と 4 つのアフィン変換の組合せとして実装するなど実装環境に応じて柔軟に対処することができる．

2.3.3 FL/FL^{-1} 関数

MISTY[10] で使われている FL 関数と同じ発想に基づく補助変換部である．すなわち，鍵によって最良差分・線形表現を変化させるとともに，差分攻撃や線形攻撃以外の解読の可能性をも軽減させることを期待している．また，構造として論理演算及び固定ローテーションだけによって実現することで，性能に大きな影響を与えないような処理としている．

ラウンド関数以外のところに FL 関数などの全単射関数を単純に挿入すると，Feistel 型暗号の大きな特徴の一つでもある暗号化処理と復号処理の同一性を損なうことになる．そこで，FL 関数とその逆関数 FL^{-1} を並列に配置することによって，暗号化処理と復号処理の同一性を損なうことがないようにしている．また，1 ビットの固定ローテーションを組み込むことによって，純粋なバイト単位処理を崩す役割も担っている．

2.3.4 全体構造

すでに述べたように，*Camellia* の 16 段での最大差分・線形特性確率の上界値は、s-box の最大差分・線形確率が $p_s = q_s = 2^{-6}$ を満たしていることから，$2^{-132} = (p_s^{11})^2 < 2^{-128}$，$2^{-132} = (q_s^{11})^2 < 2^{-128}$ となる．このことは，少なくとも 16 段で差分攻撃・線形攻撃に対して攻撃に有用なパスが存在しないことを示している．

また，*Camellia* や *E2* などのようなバイト単位の処理を基本とする byte-oriented cipher では，特に丸め差分攻撃に対する耐性も検証しておく必要がある．詳細については別稿にて報告するが，現状の結果では FL/FL^{-1} 関数の有無によって多少異なるものの，10 段程度でランダム関数との識別が不可能になる．

FL/FL^{-1} 関数の挿入位置に関して，最大差分特性・線形特性・丸め差分特性確率の段数による変化と，挿入に伴う処理速度に対するペナルティ，スライド攻撃やその他の攻撃などに対する耐性向上への期待などを考慮した．その結果，6 段ごとに挿入するのが効果が高いとの判断に達した．

以上の点を考慮し，最終的に段数を 128 ビット鍵では 18 段，192/256 ビット鍵では 24 段とし，6 段ごとに FL/FL^{-1} 関数を挿入することとした．

2.4 鍵生成部

Camellia では，128 ビット鍵の時は 24 個[3]の，また 192/256 ビット鍵の時は 30 個の 64 ビット拡大鍵を使用する．そこで，鍵生成部の設計にあたり，安全性と実用性の観点から以下のような構成条件を立てた．これらの条件は，大きく分けて，(A) 安全性に関するもの (R5, R6)，(B) 実装上，特にハードウェア実装に関する優位点となるもの (R1, R2, R4)，(C) アプリケーションへの適用可能範囲を広げるなど実用上の優位点となるもの (R3, R7) に分けられる．

[R1] データランダム化部の構成要素と共用可能であること

[R2] On-the-fly[4]で拡大鍵生成が実現できること

[R3] DES[4] の拡大鍵生成と同じように，on-the-fly 適用時に暗号化用の拡大鍵生成時間と復号用の拡大鍵生成時間が同じであること[5]

[R4] 128 ビット鍵での鍵生成は，192/256 ビット鍵での鍵生成の一部分として構成され，それ自体が単体として動作可能であること

[R5] 等価鍵が発生しないこと

[R6] 関連鍵攻撃が困難であること

[R7] 拡大鍵生成時間は暗号化時間に比べて短いものにすること

[3] 各段に 1 個の 64 ビット拡大鍵と FL/FL^{-1} 関数 1 つにつき 1 個の 64 ビット拡大鍵を必要とする

[4] 拡大鍵を生成しながら暗号化を行なう手法．メモリ量の制約上，拡大鍵をあらかじめ展開しておくことができない時に使われる

[5] 何らかの方法によってある特定の段 (最終段が多い) の拡大鍵を求められたとしても，それから芋蔓式に別の段の拡大鍵が算出されることを防ぐ目的で，鍵生成に一方向性を持たせている最近の暗号も多い．このタイプの欠点の一つに，暗号化の最終段で使用する拡大鍵まで鍵生成した (=暗号化用の拡大鍵を全て生成する) 後でないと復号処理に入れない点が挙げられる

Camellia の鍵生成部の設計において，最優先条件としたのが可能な限りコンパクトなハードウェア実装が実現できるようにすることである．そこで，我々は，ランダム関数を共用した 2 段 Feistel を用いて中間鍵を生成し，その中間鍵と元の秘密鍵とを指定された単純な手順にしたがって変換することによって拡大鍵を生成することにした．具体的には，秘密鍵と中間鍵を指定された固定量のローテーションを行ない，その結果から決められた位置のデータを抜き出して拡大鍵とする方式 (rotation and choice 方式) である．さらに，rotation and choice 方式によって全ての拡大鍵を生成した後に最初の秘密鍵と中間鍵に戻るようにローテーション量を決定することによって，復号用の拡大鍵生成も，暗号化用の拡大鍵生成と同様に，秘密鍵と中間鍵から直接 rotation and choice 方式で生成することが可能となるようにした．

安全性の面では，拡大鍵を生成するための中間鍵は Feistel 構造を用いて生成されており，さらに元の秘密鍵も使用することから，等価鍵非存在を証明することができる．また，rotation and choice 方式を採用し，かつローテーション量を一定にしない [4] ことによって，同じ値を取るような自明な拡大鍵は存在しなくなるので，関連鍵攻撃が困難であることが期待される．

3 まとめ

本稿では，128 ビットブロック暗号アルゴリズム *Camellia* の仕様と設計思想の概要を報告した．本アルゴリズムでは，AES と同じ，データブロック長が 128 ビット，秘密鍵長が 128 ビット，192 ビット，256 ビットのいずれかを利用できる．

このアルゴリズムの特徴は，安全性の観点では差分攻撃や線形攻撃に対して十分な安全性を有することを示した点，実装の高速性や柔軟性の観点ではバイト単位の処理を基本に実現可能なようにした点が挙げられる．この結果，十分な安全性を有しながら，ソフトウェア・ハードウェア問わずに高速かつ柔軟な実装が可能なマルチプラットフォーム対応型の暗号となっている．

なお，紙面の都合上，詳細な安全性の評価手法及び評価結果，ソフトウェアでの効率的な実装方法，ハードウェアでの実装結果などは今回報告できなかったので，今後の報告を参照していただきたい．

著作権に関する注意

[6] 著作権の帰属先はソフトウェア中に Copyright xxx の形で明記されています

References

[1] Biham, "New Types of Cryptanalytic Attacks Using Related Keys," *Journal of Cryptology*, Vol. 7 No. 4, pp. 229–246, 1994. (The extended abstract appeared at *EUROCRYPTO'93*)

[2] Biham, Shamir, "Differential Cryptanalysis of DES-like Cryptosystems," *Journal of Cryptology*, Vol. 4 No. 1, pp. 3–72, 1991. (The extended abstract appeared at *CRYPTO'90*)

[3] Biryukov, Wagner, "Slide Attacks," *Fast Software Encryption — 6th International Workshop FSE'99*, LNCS **1636**, 1999.

[4] *Data Encryption Standard*, FIPS-PUB-46, 1977.

[5] Jakobsen, Knudsen, "The Interpolation Attack on Block Ciphers," *Fast Software Encryption — 4th International Workshop FSE'97*, LNCS **1267**, 1997.

[6] Knudsen, "Truncated and Higher Order Differentials," *Fast Software Encryption — Second International Workshop*, LNCS **1008**, 1995.

[7] Kanda, Moriai, Aoki, Ueda, Takashima, Ohta, Matsumoto, "*E2* — A New 128-Bit Block Cipher," *IEICE Transactions Fundamentals of Electronics, Communications and Computer Sciences*, Vol. E83-A, No. 1, pp. 48–59, 2000.

[8] Kanda, Takashima, Matsumoto, Aoki, Ohta, "A Strategy for Constructing Fast Round Functions with Practical Security Against Differential and Linear Cryptanalysis," *5th Annual International Workshop on Selected Areas in Cryptography SAC'98*, LNCS **1556**, 1999.

[9] Matsui, "Linear Cryptanalysis Method for DES Cipher," *Advances in Cryptology - EUROCRYPT'93*, LNCS **765**, 1994.

[10] Matsui, "New Block Encryption Algorithm MISTY," *Fast Software Encryption — 4th International Workshop FSE'97*, LNCS **1267**, 1997.

[11] Matsui, Inoue, Yamagishi, Yoshida, "A note on calculation circuits over GF(2^{2^n})," *Technical Report*, IT88-14, 1988, (in Japanese).

[12] Matsui, Tokita, "Cryptanalysis of a Reduced Version of the Block Cipher *E2*," *Fast Software Encryption — 6th International Workshop FSE'99*, LNCS **1636**, 1999.

[13] NIST, "Announcing request for candidate algorithm nominations for the Advanced Encryption Standard (AES)," `http://csrc.nist.gov/encryption/aes/aes_9709.htm`, 1997.