CRYPTREC 暗号技術ガイドライン
(軽量暗号)

CRYPTREC 軽量暗号ワーキンググループ

2017 年 3 月

CRYPTREC **軽量暗号** WG **委員構成**

主査	本間　尚文	東北大学
委員	青木　和麻呂	日本電信電話株式会社
委員	岩田　哲	名古屋大学
委員	小川　一人	日本放送協会
委員	小熊　寿	株式会社トヨタ IT 開発センター
委員	崎山　一男	電気通信大学
委員	渋谷　香士	ソニーグローバルマニュファクチャリング＆オペレーションズ株式会社
委員	鈴木　大輔	三菱電機株式会社
委員	成吉　雄一郎	ルネサスエレクトロニクス株式会社
委員	峯松　一彦	日本電気株式会社
委員	三宅　秀享	株式会社東芝
委員	渡辺　大	株式会社日立製作所

目次

第 1 章　はじめに 1

第 2 章　軽量暗号とその活用法 3

2.1　軽量暗号とは . . . 3

回路規模 . . . 4

消費電力量 . . . 4

レイテンシ . . . 4

メモリサイズ . . . 4

2.2　軽量暗号はどこに使えるのか . . . 5

2.2.1　家電・スマートテレビ . . . 6

2.2.2　RFID タグ利用のアプリケーション（物流管理等） . . . 6

2.2.3　センサーを利用したスマート農業 . . . 7

2.2.4　医療 . . . 8

2.2.5　産業用システム . . . 9

2.2.6　自動車 . . . 9

2.3　どんな軽量暗号、パラメータを選べばいいか . . . 10

2.3.1　一般的方針 . . . 10

2.3.2　鍵長の選択 . . . 10

2.3.3　ブロック長の選択 . . . 11

2.3.4　処理データ量と鍵更新、その他の対策 . . . 11

2.3.5　利用シナリオ . . . 12

2.3.6　その他の留意点 . . . 12

2.3.7　CRYPTREC 暗号リストの暗号との違い . . . 12

2.4　軽量暗号活用例と効果 . . . 13

2.4.1　家電・スマートテレビ . . . 13

2.4.2　RFID タグ利用のアプリケーション（物流管理等） . . . 13

2.4.3　センサーを利用したスマート農業 . . . 13

2.4.4　医療 . . . 14

2.4.5　産業用システム . . . 14

2.4.6　自動車 . . . 14

第 3 章　軽量暗号の性能比較 17

3.1 ブロック暗号 . . . 17
3.1.1 ハードウェア実装評価 . . . 17
3.1.1.1 性能比較 . . . 18
3.1.1.2 評価方法の概要 . . . 36
3.1.2 ソフトウェア実装評価 . . . 38
3.1.2.1 性能評価 . . . 38
AES . . . 38
Camellia . . . 39
CLEFIA . . . 40
TDES . . . 40
LED . . . 40
PRINCE . . . 41
PRESENT . . . 42
Piccolo . . . 43
TWINE . . . 43
SIMON . . . 44
SPECK . . . 45
Midori . . . 46
3.1.2.2 性能比較 . . . 46
メモリサイズを限定した実装（暗号化のみ） . . . 46
メモリサイズを限定した実装（暗号化・復号） . . . 50
メモリサイズを限定した実装（まとめ） . . . 54
高速実装 . . . 55
最小実装 . . . 56
その他の考察 . . . 57
3.1.2.3 評価方法の概要 . . . 58
組み込みマイコン RL78 と評価環境 . . . 58
実装条件 . . . 58
3.2 認証暗号 . . . 60
3.2.1 ソフトウェア実装評価 . . . 60
3.2.1.1 性能比較 . . . 60
認証暗号の実装評価結果 . . . 60
3.2.1.2 評価方法の概要 . . . 65
コーディングの方針とインターフェース仕様 . . . 65
AES-GCM . . . 67
CLOC . . . 68
SILC . . . 69
Minalpher . . . 70
AES-OTR . . . 71
Ketje . . . 72

ACORN 73
AES-OCB 74
JAMBU 75
Ascon 76

第 4 章　代表的な軽量暗号 79
4.1　ブロック暗号 79
4.2　ストリーム暗号 93
4.3　ハッシュ関数 101
4.4　メッセージ認証コード 108
4.5　認証暗号 112

第 1 章

はじめに

限られた実装環境でも安全で高性能な暗号技術を搭載したいというニーズは古くからあり、これに応えて小型で高速な暗号技術の研究開発が進められてきた。昨今は、IoT (Internet of Things) の発展により、センサーやアクチュエータ等のリソースの限られたデバイスがネットワークに接続され、セキュリティやプライバシー上の脅威が高まっていることから、暗号技術に対してより多様な実装要件が求められている。

従来技術より低コスト、低消費電力等の特徴をもつ「軽量暗号」は、車載機器や、医療機器をはじめとするさまざまな用途での利用が期待されているが、どの軽量暗号を選べばいいのか・運用時にどのようなことに注意すればいいのか等、実際に利用する際には専門家以外では判断が困難な場合も多い。

CRYPTREC では、主として電子政府で利用する暗号技術について検討を行っているが、それに加えて、今後さまざまな領域で利用が想定される暗号技術について技術調査を行い、社会に役立つ形で情報提供を行うことを目指している。特に、軽量暗号技術が求められる製品やサービスにおいて、利用者が最適な暗号方式を選択でき、容易に調達できることを目指し、2013 年度より CRYPTREC 暗号技術評価委員会の下に軽量暗号ワーキンググループ (WG) が設置された。本ガイドラインは、軽量暗号の方式を選択・利用する際の技術的判断に資すること、今後の利用促進を図ることを目的として、軽量暗号 WG が作成したものである。主たる読者としては、情報システムのセキュリティ機能の設計・開発・実装において暗号技術を活用する技術者を想定しているが、軽量暗号技術に興味をもつ方に広く読んで頂ければ幸いである。

1 章は、本ガイドラインの総説である。2 章では、軽量暗号の概説をまとめている。まず、対象とする軽量暗号とその活用例を示し、その上で、軽量暗号を実際に活用する際の手引きを示している。特に、軽量暗号の特徴、代表的なユースケース、方式や、パラメータの選択方法、使用時の留意点などを記載している。3 章では、代表的な軽量暗号の性能を示している。多くの軽量暗号方式が提案されている「ブロック暗号」および「認証暗号」の技術分野において、代表的ないくつかの方式を取り上げ、その性能を比較している。性能比較は、ハードウェア実装とソフトウェア実装で、それぞれ同一の実装環境下で行っている。ハードウェア実装では回路規模、消費電力、レイテンシの比較結果を、ソフトウェア実装では必要メモリサイズの比較結果を示している。4 章では、代表的な軽量暗号の方式の基本情報を、「ブロック暗号」、「ストリーム暗号」、「ハッシュ関数」、「メッセージ認証コード」、「認証暗号」の技術分野別に示している。

本ガイドラインで紹介している軽量暗号技術は、執筆時点までに、主要国際学会で発表されており、有力な攻撃法が発見されておらず、かつ十分な実装性能を持つと考えられる方式を選び、かつ、可能な限り最新の情報に基づき安全性や実装性能、標準化動向等を紹介している。しかしながら、軽量暗号の研究開発は今まさに盛んに行われており、年々新たな方式や評価結果が出ているところであり、記載内容が執筆時点のものであることに留意いただ

きたい。

本ガイドラインは下記に示す軽量暗号 WG 委員および CRYPTREC 事務局で執筆・編集を行った。所属は 2016 年 10 月時点のものである。また、本ガイドラインを執筆する上で、軽量暗号の実装評価において三菱電機株式会社の松井充氏、菅原健氏、村上ユミコ氏、梨本翔永氏に多大なご貢献を頂いた。

主査	本間　尚文	東北大学
委員	青木　和麻呂	日本電信電話株式会社
委員	岩田　哲	名古屋大学
委員	小川　一人	日本放送協会
委員	小熊　寿	株式会社トヨタ IT 開発センター
委員	崎山　一男	電気通信大学
委員	渋谷　香士	ソニーグローバルマニュファクチャリング＆オペレーションズ株式会社
委員	鈴木　大輔	三菱電機株式会社
委員	成吉　雄一郎	ルネサスエレクトロニクス株式会社
委員	峯松　一彦	日本電気株式会社
委員	三宅　秀享	株式会社東芝
委員	渡辺　大	株式会社日立製作所
事務局	盛合　志帆	国立研究開発法人情報通信研究機構
事務局	大久保　美也子	国立研究開発法人情報通信研究機構
事務局	金森　祥子	国立研究開発法人情報通信研究機構

第 2 章

軽量暗号とその活用法

2.1 軽量暗号とは

近年、リソースの限られたデバイスにも実装可能な「軽量暗号」(Lightweight Cryptography) の研究開発が発展しており、多くの方式が学会等で提案されている。欧州では 2004 年から European Commission の第 6-7 次 Framework Programme の研究プロジェクト ECRYPT I, ECRYPT II のテーマとしても取り上げられてきた。日本も小型実装に適した暗号技術等で強みをもっている分野である。軽量暗号の標準化も進んでおり、軽量暗号アルゴリズムを技術分野毎に定めた ISO/IEC 29192 や RFID 向けの暗号技術を定めた ISO/IEC 29167 が策定され、米国 NIST (National Institute of Standards and Technology) も 2015 年より軽量暗号の標準化の検討を開始している。

低コスト・低消費電力で動作可能な軽量暗号技術は、今後も、車載機器や、医療機器などさまざまな機器で利用される可能性があり、IoT (Internet of Things) や CPS (Cyber Physical System) といった次世代のネットワークサービスを構築する上で有効なセキュリティ技術の一つとなることが期待されている。

一方で、これまで提案されてきた軽量暗号技術には、ハードウェア実装のサイズ・消費電力量の観点で軽量性を追求したものや、組み込みソフトウェア実装の必要メモリサイズの軽量性を追求したもの等、さまざまな性能指標で最適化された方式が存在する。「軽量暗号」に対して一般的に合意されている定義はない。また、性能と安全性のトレードオフもあり、実際の性能は多岐に渡っている。以上の状況を鑑み、本ガイドラインでは、「実装性能と安全性のトレードオフを勘案した上で、従来の暗号技術に対して特定の性能指標で優位性（軽量性）を持つように設計された暗号技術」をスコープとし、用途が想定される代表的な性能指標に対して優位性を主張する暗号を主な対象とする。また、現時点で、公開鍵暗号系において「軽量暗号」として広くコンセンサスがとれている方式はほとんどないため、本ガイドラインでは共通鍵暗号系の軽量暗号を対象としている。

本ガイドラインでは、下記を軽量暗号の代表的な性能指標とする。

- ハードウェア実装における
  - 回路規模
  - 消費電力量
  - レイテンシ
- 組み込みソフトウェア実装における
  - メモリサイズ（ROM/RAM）

■回路規模　ハードウェア実装の回路規模は半導体のコストに直結し、また、消費電力 (Power) の指標にもなり得ることが知られている。回路規模の小型化は、RFID をはじめとする回路実装面積の要求条件が厳しいアプリケーションで重要な要件である。また、非接触 IC カードのようにバッテリや外部供給電源がなく、電磁誘導等で駆動するデバイスにおいても重要な要件である。

■消費電力量　消費電力量（Energy）の低減は、人体へ埋め込まれたり密着装備される医療機器をはじめ、バッテリで駆動するあらゆるデバイスで求められる要件である。

■レイテンシ　本ガイドラインでは、レイテンシ（遅延時間）は 1 回の暗号化（復号）処理に必要な時間を意味する。低遅延性はメモリ暗号化や車載機器などのリアルタイム性が求められるアプリケーションで必須の要件である。

■メモリサイズ　組み込みソフトウェア実装では、組み込みマイコン上で実現されるさまざまなアプリケーションの一部として、暗号機能を実装することが多い。組み込みマイコンでは、ROM や RAM のサイズが限られており、小さく実装できる暗号ほど、選択できるマイコンの品種が増えたり、コストを下げられる等の利点がある。組み込みマイコンは家電機器やセンサー、車載向け等で広く利用されており、実装に必要なメモリサイズ (ROM/RAM) が少ないことはこれらのアプリケーションで重要な要件である。

性能指標	アプリケーションの例
回路規模 (消費電力, コスト)	RFID、低コストセンサー
消費電力量	医療機器、バッテリ駆動デバイス
レイテンシ（リアルタイム性能）	メモリ暗号化、車載機器、産業向け I/O デバイス制御
メモリサイズ (ROM/RAM)	家電機器、センサー、車載機器

本ガイドラインでは、軽量暗号の代表的な方式を、ブロック暗号、ストリーム暗号、ハッシュ関数、メッセージ認証コード、認証暗号の技術分野別に分類している。それぞれの技術で実現できる機能 (秘匿、認証など) は既存の暗号技術と同じである。

本ガイドラインで紹介している軽量暗号技術は、執筆時点までに、主要国際学会等で発表されており、有力な攻撃法が発見されておらず、かつ十分な実装性能を持ち、リソースの限られた実装条件下で有用と考えられるアルゴリズムを選んでいる。下記の表に示した軽量暗号技術の概要は 4 章にまとめている。

ブロック暗号	CLEFIA, LED, Midori, Piccolo、PRESENT, PRINCE, SIMON, SPECK, TWINE
ストリーム暗号	ChaCha20, Enocoro, Grain v1, MICKEY 2.0, Trivium
ハッシュ関数	Keccak, PHOTON, QUARK, SPONGENT
メッセージ認証コード	SipHash
認証暗号	ACORN, Ascon, AES-JAMBU, AES-OTR, CLOC and SILC, Deoxys, Joltik, Ketje, Minalpher, OCB, PRIMATEs

2.2 章で軽量暗号がどのようなところで使えるか活用例を示し、2.3 章で方式や鍵長、ブロック長を選ぶ際の留意点を記載している。3 章では、特に多くの方式が提案されているブロック暗号と認証暗号について同じプラットフォーム上で実装した際の性能比較を掲載しているので、軽量暗号の方式を選ぶ際の指針となれば幸いである。

2.2 軽量暗号はどこに使えるのか

「いつでも、どこでも、何でも、誰でも」ネットワークにつながるユビキタスネットワークの構築は「モノのインターネット」(IoT: Internet of Things) というキーワードで、近年表現されるようになっている。IoT のコンセプトの下、パソコンやスマートフォン、タブレットといった従来型の ICT 端末だけでなく、自動車、家電、ロボット、施設などがインターネットにつながることとなる [7]。ただし、現段階では、どのようなデバイスがネットワークに接続されるかは明言することはできず、デバイス上でどのような処理が行われるかも不明である。IoT の時代だからこそ、今までは考えられない状況を考えなければならない。

これら IoT 端末は、先述したユビキタスネットワーク構築のために、我々の生活空間へシームレスに浸透しつつあるため、プライバシーや情報秘匿、また情報の完全性担保を目的としたセキュリティ機能が必須となる。加えてサービス利用者の目的はセキュリティ機能によるメリットを享受することではないため、IoT 端末を利用した円滑なサービス提供をセキュリティ機能が妨げるべきではない。

そして、すべてのデバイスに高機能の CPU が搭載されるとは考えにくい。従来型 ICT 端末に比べてプロセッシング能力や記憶容量、またバッテリによる動作時間の制限などを含め、計算機資源が乏しいデバイスも想定しなければならない。例えば、IoT 端末の 1 つである自動車において、図 2.1 のような要求条件が述べられている。

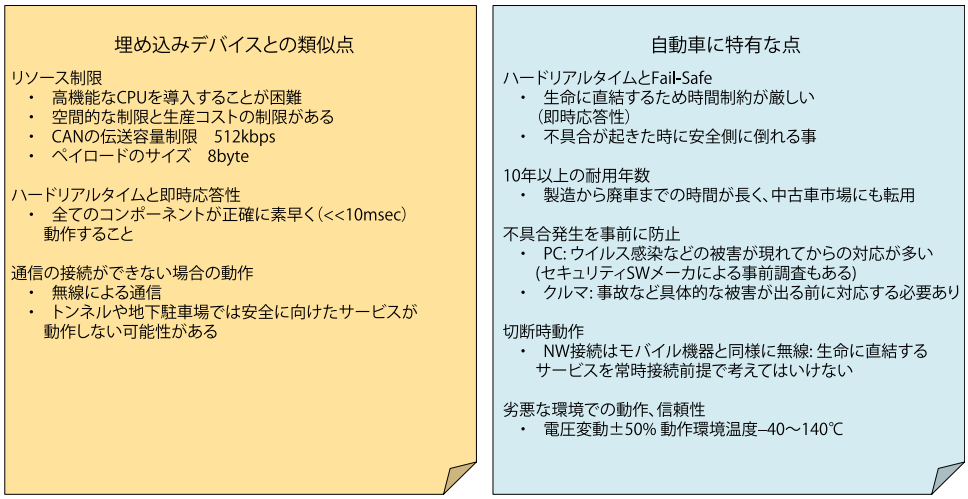

図 2.1 自動車の機能に関する要求条件

そこで、有望となるのが軽量暗号である。軽量暗号には、CPU 負荷が軽い、使用するメモリサイズが小さい、低レイテンシ性などの特徴がある。このような軽量暗号は、これら計算機資源が比較的乏しい IoT 端末に適していると考えられ、特徴に応じて種々の場面での使用が期待されている。

CPU やメモリなどを多くのアプリケーション（以下アプリと記す）で共有しなければならない場合、CPU コストが小さく、メモリ使用量が少ない暗号として期待される場合がある。スマートフォンやタブレット端末、スマートテレビと呼ばれる高機能化されたテレビで使用される暗号がこの例である。

また、装置そのものがバッテリーで動作しているため、消費電力が少ない暗号として期待される場合もある。例えば、環境測定用のデバイスは電気が通っていない場所に置かれることも多々ある。また、埋め込み型の医療機器では、人に埋め込まれるものであれば、バッテリー駆動以外はありえない。さらに、埋め込み機器として、できる

だけ小さいデバイスであり、人体への影響が少ないデバイスであることも望まれる。このような場合に使用される暗号として軽量暗号が期待される。

即時性、低レイテンシ性が求められる場合もある。バッテリーの消費を極力抑えるために、データを送信する場合にのみ電源スイッチが入り、一瞬だけ動作してデータを伝送し、伝送が終了次第電源を切るように運用している機器がある。このような場合には、低レイテンシ性を持つ軽量暗号が役に立つ。また、自動車の制御のように遅い反応では安全性に影響が出るような場合も有用と考えられる。

これら軽量暗号の活用例について、スマートテレビ、RFID、農地での環境測定、医療機器、工場の機器制御、自動車の例をあげて紹介する。

2.2.1 家電・スマートテレビ

スマートフォンやタブレット端末、スマートテレビの CPU はいろいろな種類があり、低機能なものから高機能なものまである。高機能の CPU を使用している場合は問題ないが、低機能の CPU を使用している場合は当然、実現できる機能が制限される。

例えば、テレビ内の処理は、スクランブルの解除、圧縮の復号、映像・音声の提示のみがハードウェア処理であり、それ以外の処理は内蔵されている CPU で処理される。しかも、CPU は実に多くの作業をしており、ほとんど負荷 100% で使用されている。さらに、今後の発展としてネットワークと接続して通信アプリをもテレビ上で動作させようとしている。テレビを少しでも安価にするためには、使用される CPU も安価なものが好ましく、このため高機能な CPU が使われていないのが現状である。そのようなテレビで、テレビ機能だけでなく、アプリをシームレスで動作させる際には、CPU やメモリリソースの取り合いが生じる。その上、暗号化すべきデータが存在するとすれば、暗号化は使用メモリ (ROM/RAM) サイズが小さく、CPU 負荷が小さいものが推奨される。先にも述べたとおり、テレビ内に暗号用に特別なチップを導入することはなく、ソフトウェアでの動作となる。このため、ソフトウェアで CPU 使用時間が短く ROM の使用サイズが小さい暗号の利用価値が高いと考えられる。

IoT 時代では、テレビ以外にも多くの家電がネットワークに接続される。ネットワークを通じてやりとりするデータの中には秘匿しなければならないデータもあり得る。仮に、エアコン、ガスコンロなどがネットワークにつながったとして、その制御信号がネットワーク上を流れるようなサービスでは、制御信号が外部から不正アクセスされ改ざんされたり、勝手なコマンドを入力されることにより異常動作しないようにしなければならない。また、家庭内のロボットが個人データを持っているような場合は、プライバシー保護の観点からデータを秘匿しなければならない。一方、これらのデバイスは 10 年以上使用し続けることも考えなければならない。その上、小さな・安価なデバイスにおいては特殊なハードウェアが搭載されることは考えにくい。デバイスによっては、スマートテレビと同じようにリソースの取り合いが生じることもあり得る。これらのデバイスでは、アップデートが可能なソフトウェア処理がメインとなり、それを司る CPU は低機能で安価なものになるであろう。このような CPU でデータ保護を実現するためには、ソフトウェアで CPU 使用時間が短いタイプの暗号が必要となる。

2.2.2 RFID タグ利用のアプリケーション（物流管理等）

RFID とは無線を利用して、いろいろな物を認識するシステムのことであり、いろいろな用途で使用されている。例えば、倉庫内の在庫管理、物流における物品管理、CD/DVD ショップでの盗難防止、物品の履歴管理、電子マネー、交通用カード、社員証カードなどである。変わったところでは、動物の生態を調べるための追跡用にも使われることがある。同様に、人の居場所を追跡するために利用することなども考えられている。さらには、IoT 時代の

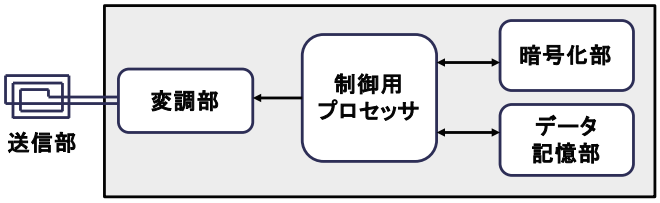

図 2.2　RFID の構造

家庭内の物品識別、例えば、冷蔵庫内に入っているものを冷蔵庫が認識するために利用するなども考えられている。

RFID による無線電波はそれほど強くはないため、近距離伝送で利用される。従って、例えば、倉庫内にどのような品物を揃えられているかを別の同業者が知るために、RFID で発信されるデータを倉庫の外部から盗聴するようなことは困難である。倉庫での在庫管理においては、暗号を実装する必要性は少ないと考えられる。

これに対し、CD ショップでの万引き防止、電子マネー、交通用カード、社員証カードの偽造防止、人の追跡や人の持ち物の追跡におけるプライバシー保護、動物の追跡における密漁防止等の目的で使用される場合は、データは秘匿されることが必須である。

さて、RFID タグは RFID で用いられるチップのことで、そのサイズは数十 μm 角から、数 cm 角のものまである。このチップの中にプロセッサ、メモリなどが凝縮されている。図 2.2 に RFID の構造を示す。

送信部と書かれている部分はアンテナであり、使用されている周波数（kHz〜MHz）に応じてその大きさはまちまちである。送信部以外の部分は小型化が可能であるが、データ記憶容量や使用されるプロセスによって大きさが決まってくる。

RFID では、全体の回路規模に対して暗号機能に使える回路サイズには限界がある。RFID がパッシブタグ、すなわち電池を内蔵していない場合、電波から給電するため消費電力に大きな制約が存在する。このような制約の下でデータを保護するため、ハードウェアが小さく、消費電力の少ない暗号が求められる。

2.2.3　センサーを利用したスマート農業

農作物を栽培する際に、気候の変化に応じた対応をとることで、収穫量を安定させるなど生産性を向上させたり、作物の品質を向上させることが可能になると言われている。これまで多くの農家では、この対応を生産者の経験や勘で補ってきた。例えば、気候（気温、湿度、日照時間、日照量、土壌水分、風向、風速、降水量など）をモニタリングすることで、これまで経験・勘に頼っていた部分を数値化・データ化することができれば、熟練者のみならず経験の浅い人々でも安定した農業を営むことができる。例えば、これらのモニタにより、水まき時間、量を制御したり、ビニールハウスの窓の開け閉めを自動化するなどが可能となる。熟練者にとってもこのようなモニタリングは、気候への対応を容易にし、作業計画・スケジューリング、病害虫駆除対策等が容易になるだけでなく、農地そのものを評価し・改良するための指針を与えることになり、安定した収穫への手助けとなる。

モニタリングするデータは細かければ細かいほど分析精度が高くなる。特に農地の評価を行う上では、広大な農地を区画に分けて区画ごとのデータを取ることが望ましい。農家の人々にしてみれば、少ない労力で長期的に、安価に利用できるモニタリング装置を期待することとなる。このため、徐々に環境センサーネットワークの利用が始まっている。

このセンサーへの条件として、

- 自律駆動

- 小型
- 省電力
- 大量設置

などがある。

細かいデータ取得には、膨大な数のセンサーが必要となる。このため安価な軽量暗号が望まれる。

同じような状況が防災上も考えられる。天候変化だけでなく、地殻の変動をセンシングすることで、地震予知に役立てたり、火山噴火予知、土砂崩れなどの予測に役立てることが考えられている。これらの用途のセンサーも、人の出入りが困難であり、給電がままならない場所に置かれることがしばしばある。しかも、防災関連のデータは重要なデータであり、盗聴はもとより、改ざんなどはあってはならない。データが改ざんされたことで、不正な警報が発せられることがあってはならないためである。このような秘匿性のあるデータの改ざん防止、保護のためには、認証付きの軽量な暗号が有望と考えられる。

2.2.4 医療

入院経験がある人は少なくないと思うが、入院患者には心電図・血圧・脈拍・血糖値・酸素濃度などを測定するいろいろなセンサーが取り付けられる。基本的にはこのようなセンサーは有線で接続されており、電源にも接続されている。ただし、患者が移動する場合、このようなセンサーも患者とともに移動できることが望ましい。特に、帰宅等の外出ができる人、在宅・職場であっても種々のデータをモニタリングする必要のある人もいる。さらには、体に測定装置を埋め込まなければならない場合もある。定期的なデータ測定、投薬時間の連絡などのためである。特にデバイスが埋め込まれている場合は、デバイスを取り出すことは考えにくい。このため、無線利用のデータ通信、バッテリーによる長時間動作が可能なデバイスでなければならない。埋め込み型ではなくとも、有線では移動の自由度が制限されるため無線であることが望ましく、必然的にバッテリーで長時間動作するセンサーであることが必須となる。

また、これらのセンサーデータは個人の情報そのものであり、プライバシー保護の観点からは全てのデータが秘匿されなければならない。現在、特に埋め込み型センサーでは小型化の研究・開発が進められており、nm サイズのデバイスが開発中である。当然、それに付随する暗号機能も小型化が必須である。

さらに、最近ではウェアラブルデバイスの進歩に伴い、mHealth と呼ばれる概念が出てきている。Mobile Health の略とも言われるが、確立した定義は存在していない。例えば mHealth として以下のようなことが考えられている。ウェアラブルデバイスを生体データの観測装置として利用し、日々の自分の健康維持のために心肺活動などのデータを観測する。家庭内で日々のデータを記録しておき、健康診断や病院にお世話になった場合に利用するものである。日常生活の中のデータ保存のみならず、健康維持のために行っている活動の記録としてもしばしば利用する。従来からの健康機器である歩数計も GPS による位置情報の取得など個人情報を取り扱うものが出てきている。

この mHealth では、人の動作により発電する発電デバイスをも組み込んだデバイスもあるが、基本的にはウェアラブルデバイスが測定装置となるため、バッテリー運用である。

使用方法にもよるが、常時データを測定し、観測、集計・分析するのであれば、無線による伝送が必要となる。これらのデータも取扱いに配慮を要する情報であり、プライバシー保護の観点からはデータの暗号化が望まれる。

2.2.5 産業用システム

工場などでは資材の運搬や加工、組み立てなどの工程を自動化し、効率的に運用することが考えられている。例えば、工作機械やロボットを動作させるにあたって、ネットワークを通じて情報を共有したり、センサーを用いての工程管理が行うことが可能となる。さらに、ネットワークを通じて全ての情報を一か所に集めることで、集中管理が可能となる。

もっとも先進的にこのようなオートメーション化に取り組んでいたのが、ドイツではないだろうか。国家プロジェクトとして多額の予算を組み、インダストリー 4.0 として種々の機械・装置がセンサーを持ち、そのデータに応じて考えながら動作するスマートファクトリ（考える工場）という概念を実現しようとしてきた。

また、データを集中管理するにあたって、ネットワークについても EtherCAT と呼ばれる方式が提案された。工場のデータ管理では、個々のデータが重要な意味を持っている。従来のインターネットのプロトコルである TCP/IP ではデータのネットワーク上での衝突等が起こり、データの遅延、消失などが生じてしまうため、工場データの管理に利用するには弊害があった。そこで、EtherCAT では、個々の機器をスレーブ化し、その接続をシリアルにする方式としている。

これらのセンサーは工場内のあちこちに配置される。当然、人の手がはいりにくい箇所もある。装置に備え付けてあるので電源は常に供給される状況にあるが、すべてをシリアルに接続することが困難な状況も生じる。これに対応するためには、無線でデータを送ることも考えられている。無線でデータを送る場合は、工場内で送受信できるようにするため、ある程度の距離を無線伝送することになる。このためには、暗号化してデータを送受することが必要である。

2.2.6 自動車

自動車では、車内でのデータ通信だけでなく、車外との通信も行うようになってきた。この自動車の通信においては、自動車と自動車 (車車間) ならびに自動車と信号機や道路標識などのインフラ (路車間) との相互通信により安全運転の支援をおこなう自動運転支援システム (Car2X communication) に対応するために大量の情報を処理し、かつクラウドと連携してコンテンツなどのサービスと連携する車載情報システム、ならびにレーダなど外部からの情報を各種センサーで取得しボディ系、シャシー系など複数の電子制御ユニット (ECU) 間において互いに情報をやりとりしながら協調制御をおこなう車載制御などがある。

車内の ECU 間の協調制御をおこなうため、車内は CAN (Controller Area network)、LIN (Local Interconnect Network)、Ethernet など各種方式によるネットワークによって張り巡られている。CAN は車載において基幹となるネットワークであり、パワートレイン系、シャシー系、ボディ系などとの協調制御に広く使用されている。例えば車間をミリ波レーダで計測し、車間が狭くなると警告表示あるいは警告ブザーを開始、ブレーキ制御、シートベルト制御といった衝突検知システムを CAN を通じて情報共有をおこなうことで実現している。車載 LAN においても攻撃の実証例がでてきており、CAN に誤ったメッセージが流れるとブレーキ制御が誤るなどの重大な事態につながりかねないため、メッセージを暗号化するとともにが改ざんされていないか認証をおこなう必要がある。車車間、路車間のような非常に高い暗号処理性能、低レイテンシ暗号までは不要もリアルタイム性は求められており、高速な暗号処理の実現が必須である。

車の自動運転支援システムにおいては車車間、路車間と相互通信しなければいけない機器の数が非常に多く、かつ個人につながる情報の漏洩を防ぐため暗号処理にも対応するため、回路規模が小さく、低レイテンシな暗号を実

現する必要がある。

車載情報端末においてはこのほかクラウドを経由した様々なサービス、渋滞予測などの交通情報あるいはコンテンツなども取り扱っており、情報の保護あるいは改ざんの防止が求められる。特にコンテンツ保護においては高スループットの暗号処理が求められる。

規格関連の状況では、欧州 AUTOSAR では車内通信用にメッセージ認証技術の必要性が述べられている。すなわち、多くの自動車では採用されている CAN に、AUTOSAR 規格の R4.2.2 における Secure Onboard Communication (SecOC) としてカウンタとメッセージ認証コード (MAC) によるメッセージ認証が明記されている [1]。車外通信では、欧州の Car 2 Car Communication Consortium (C2C-CC) による車車間通信へ軽量暗号の活用が期待できる [2]。

2.3 どんな軽量暗号、パラメータを選べばいいか

2.3.1 一般的方針

2.1 節で挙げたように軽量暗号は既存暗号と比べて実装時に

- 回路規模
- 消費電力量
- レイテンシ
- メモリサイズ

のいずれか、もしくは複数の性能指標において優位性を持つ暗号である。逆にいえば、ある軽量暗号について既存暗号と比べて回路規模については小さく実装可能であるものの、その回路規模が小さな実装では既存暗号の実装より消費電力量は増えている可能性もあり得る。このように軽量暗号は万能のものではない。暗号利用システムにおいて暗号に対する性能指標の要求条件を明らかにすることは困難であることが多いが、やみくもに軽量暗号を利用するのではなく、まずは要求条件をある程度は明らかにしなければならない。その上で、まずは従来暗号、特に CRYPTREC 暗号リストに掲載されている暗号の利用を検討し、それで上記性能指標に対し達成困難な場合に軽量暗号の利用を検討するのが望ましい。

軽量暗号を使用するにあたり、秘匿のみが必要な場合、データ認証のみが必要な場合、両方が必要な場合、といった場合が考えられ、目的に応じて方式を選択する。例えばブロック暗号を用いる場合、使用する利用モードにブロック暗号の復号が必要ないのであればその分の実装コストを削減できる。秘密鍵をハードコードするような利用形態であれば、その分の実装コストも削減可能である。利用目的と実装からの制約に照らし合わせて、適した方式を選択することが求められる。

2.3.2 鍵長の選択

鍵長は安全性の基準となるもっとも重要なパラメータであり、慎重な選択が求められる。ブロック暗号においては一組、あるいは少数の入出力があれば全数探索が可能であり、次節において述べる多くのユースケースにおいても全数探索のシナリオが成立する。例えば、鍵長を 128 ビットから 80 ビットに減らしたとする。回路集積効率に関する 3 年で 4 ($= 2^2$) 倍というムーアの法則が今後も続くとすると $(128 - 80)/2 \times 3 = 72$ 年寿命が短くなることに注意しなければならない。

2.3.3 ブロック長の選択

ブロック暗号におけるブロック長も安全性に直結する重要なパラメータである。とくにブロック暗号利用モードや認証暗号にブロック長の短いブロック暗号を使用した場合、安全性が保たれるデータ量に厳しい制限が加わり、このための対策が必要となる。

例として CTR モードの安全性の評価法を紹介する。CTR モードの安全性は、利用するブロック暗号のブロック長を n ビット、同じ鍵のもとブロック暗号が呼ばれる回数を σ とおくと一様ランダムなビット列との識別が確率 $\sigma^2/2^{n+1}$ 以下であることが示されている (例えば CRYPTREC 技術報告書 No.2012(2011/3/4 更新版)47 頁)。こ

表 2.1　利用者一人の暗号文がランダムな文字列と識別可能となる最大データ長

ブロック長 n (ビット)	利用者数	データ長 $n\sigma$
64	10^3	1.4Gbyte
	10^6	46.3Mbyte
	10^9	1.4Mbyte
48	10^3	4.3Mbyte
	10^6	139.0Kbyte
	10^9	4.4Kbyte

の確率をもとに、システムの利用者のうち一人程度は暗号文とランダムな文字列との識別が可能となるリスクまで受容出来る場合の同一の鍵で処理できる最大データ長を求めると表 2.1 の通りとなる。それほど大きなデータではないことに注意が必要である。

別の注意点として、ブロック暗号を構成要素として用いてハッシュ関数を構成する方法が知られている。これらは十分に長いブロック長を有するブロック暗号を用いた場合に安全性が保たれるものであり、ブロック長の短い軽量ブロック暗号はこのような用途には適さないと考えられる。

2.3.4 処理データ量と鍵更新、その他の対策

利用形態に応じて、例えば秘密鍵を更新可能な環境であれば、頻繁に更新をするといった対策が考えられる。あるいは秘密鍵をハードウェアとしてハードコードするような実装においては秘密鍵の更新ができないため、処理するデータ量に制限を設け、それを超過する前にデバイス自体を破棄するといった運用が考えられる。

鍵更新のタイミングについて、データを処理するごとに一般に暗号方式の安全性は徐々に低下する。敵の攻撃成功確率が十分に許容できるほど小さい範囲にある間に鍵を更新することが望まれる。例えば [3] にある CMAC では、一般的なアプリケーションにおいて、ブロック長 128 ビットの AES を利用した場合は 2^{48} ブロックのデータ (2^{22}G バイト）を処理する前の鍵更新を、あるいは 64 ビットブロック暗号の TDES を利用した場合は 2^{21} ブロックのデータ（16M バイト）を処理する前の鍵更新を推奨している。これらの制限により、敵の攻撃成功確率は AES の場合は 10 億分の一、TDES の場合は 100 万分の一以下となることが期待される。許容できる攻撃成功確率は暗号方式を使用するアプリケーションに依存し、慎重な選択が求められる。

鍵更新の方法についてもアプリケーションに応じて、それぞれに適した方法を選択することが必要である。鍵共有プロトコルを実行できる環境であれば、鍵更新は問題とはならない。あるいは、マスター鍵からセッション鍵を

生成し、鍵更新をしながら同期をとるような利用形態が考えられる。

いずれにせよあるタイミングで鍵を更新、あるいは破棄する必要がある。この頻度を遅らせることができるような暗号方式も提案されており、例えば MAC であれば SUM-ECBC [5] や PMAC_Plus [6]、暗号化であれば CENC [4] といった例が挙げられる。これらの方式では、利用するブロック暗号のブロック長が n ビット、ブロック暗号が呼ばれる回数が σ であれば、敵の攻撃成功確率はおおよそ $\sigma^3/2^{2n}$ 以下となる。前述の方式と比べ、64 ビットブロック暗号であれば敵の攻撃成功確率は $\sigma/2^{64}$ 倍小さくなり、ある攻撃成功確率の閾値に達するまでにより多くのデータを処理することができるようになる。

2.3.5 利用シナリオ

軽量ブロック暗号は一般的に線形攻撃や差分攻撃などの暗号学的な攻撃に対して、十分な耐性を有するよう設計されている。これらは軽量ブロック暗号に限らず、通常のブロック暗号についても考慮される安全性である。一方で、通常のブロック暗号では関連鍵攻撃や既知鍵攻撃、選択鍵攻撃といった攻撃者側にとりわけ有利な状況を考え、その安全性を評価することが行われている。実装効率の観点から、軽量ブロック暗号では簡素な設計を採用する方式が多くあり、必ずしもこれらの攻撃に対する安全性が十分ではない、あるいは十分な評価がなされていない、といったケースが考えられる。関連鍵攻撃や選択鍵攻撃に対する耐性が十分ではないことが分かっている方式も存在し、これらの方式を採用する場合には攻撃シナリオが成立しないような運用が求められる。

2.3.6 その他の留意点

ソフトウェア実装に適した方式やハードウェアに特化した方式など様々な選択肢があり、実装環境に応じて使用する方式を選択する。このとき、暗号学的な攻撃手法のみならず、サイドチャネル攻撃に対する対策の必要性について検討することが重要である。一般的に多くのユースケースにおいてサイドチャネル攻撃が可能な環境が考えられ、実装レベルでの対策の必要性を検討する必要がある。

2.3.7 CRYPTREC 暗号リストの暗号との違い

ブロック暗号を例にして考える。CRYPTREC 暗号リストに掲載されているブロック暗号はブロック長は 64 ビットもしくは 128 ビット、鍵長は 128 ビット以上となっている。

一方、軽量ブロック暗号については、ブロック長は 32 ビット、また鍵長も 80 ビットなど CRYPTREC 暗号リスト掲載の方式より短いものも数多く提案されている。ブロック長と鍵長は安全性に直結するパラメータであり、ブロック長や鍵長を短くすることにより健在化するリスクが利用するシステムにおいて受容可能かどうかを判断しなければならない。

CRYPTREC 暗号リストにある注釈無しの暗号の利用で安全に利用できる範囲であっても小さなパラメータの軽量暗号を採用する場合には上記例のように利用データ量などについて、もう一度、評価しなおす必要がある。

軽量暗号に限った話ではないが、CRYPTREC 暗号リストの暗号と同様、無条件に永遠に安全である効率的な暗号はない。利用目的とリスク管理を適切に行ない、従来暗号の利用が困難であるが軽量暗号が使える場面では積極的な利用を勧めたい。

2.4 軽量暗号活用例と効果

2.2 節で取り上げた活用例において、軽量暗号、その中でもブロック暗号、あるいはメッセージの改ざんも検出できるよう認証暗号を利用する場合において、どのような点に着目して軽量暗号を選定していけばよいかについて本節にて例示する。以下、選び方の一例として本ガイドラインの 3 章に記載している軽量暗号の性能比較を元に記載する。軽量暗号を適用する際には、2.3 節に記載されている鍵長の選択などの安全性にも配慮すること。

2.4.1 家電・スマートテレビ

スマートテレビのようにリソースの取り合いが生じる場合、例えばソフトウェアで CPU 使用時間が短く ROM の使用サイズが小さい軽量暗号を選択したい場合、3 章の図 3.40 を元に、SPECK、SIMON、Piccolo、及び TWINE のような軽量暗号が候補となりうる。

また、家電におけるデータ保護については、アップデートが可能なソフトウェア処理がメインとなり、それを司る CPU は低機能で安価なものになるであろう。このような CPU でデータ保護を実現するものの一例として、3 章の図 3.34 を元に、SPECK のようなソフトウェアで CPU 使用時間が短いタイプの軽量暗号を選定することも考えられる。

2.4.2 RFID タグ利用のアプリケーション（物流管理等）

RFID では、全体の回路規模に対して暗号機能に使える回路サイズの限界もさることながら、消費電力に大きな制約が存在する。ハードウェア実装 (小型実装) において消費電力の低減が可能な暗号方式として、例えば 3 章の図 3.18 などから SIMON、SPECK、Piccoro、PRINCE などの軽量暗号の選定が考えられる。特に 180nm 以上のレガシープロセスでは、この回路規模の差がクリティカルになる。より新しい 40nm 世代のプロセスであっても、50μm 角クラスの極めて小さなチップであれば、この回路規模の差が暗号機能の搭載可否に影響を与える。

2.4.3 センサーを利用したスマート農業

農作物の生産向上のためには、細かいデータ取得が必要であり、膨大な数のセンサーが必要となる。これらのデータのすべてを暗号化するためには、安価な軽量暗号が望まれる。また、防災上では、データを暗号化し、メッセージ認証コード (MAC) を付与する必要がある場合も考えられる。これらの機能をともに満たす認証暗号を実装する場合は、3 章の図 3.45、図 3.47 によると、ROM サイズの使用量が少なく小型実装に向いている JAMBU-SIMON、SILC-PRESENT、ACORN、Ascon、 Minalpher などの軽量暗号の活用が例として考えられる。ブロック暗号の実装を考えるのであれば、例えば 3 章の図 3.42 から、AES と比較して 1 回あたりの処理 cycle 数が少ないことから消費電力量が小さく、バッテリー寿命を長くすることができる SPECK、SIMON、PRESENT、TWINE ならびに Midori などの軽量暗号の使用も検討に値する。メッセージが短く暗号化を要しない場合は、4.4 節に記載のメッセージ認証コードの利用、あるいはブロック暗号に軽量暗号を適用し CMAC モードを使用することで小型化が図られる。

2.4.4 医療

医療用センサーデータは取扱いに配慮を要する情報を含んでおり、プライバシー保護の観点からはデータが秘匿されることが望まれる。現在、特に体内埋め込み型センサーでは小型化の研究・開発が進められており、nm サイズのデバイスが開発中である。当然、それに付随する部分も小型化が必須である。これらを併せ持つ暗号としては、ハードウェアで省電力の軽量暗号、例えば 3 章の図 3.14 等を参考にすると SIMON、SPECK、Piccolo、PRESENT などが候補となりうる。

また、mHealth の場合は、健常者がウェアラブル端末を利用することが主な想定となっているため、小型化、長時間化という要件は体内埋め込み型センサーほどクリティカルな問題とはならないが、ウェアラブル端末の CPU の小型化・低廉化のために、軽量暗号の利用が望ましい。

2.4.5 産業用システム

工場などの産業用オートメーションにおいて、フィールドネットワークのオープン化が進む中、EtherCAT などの超高速の産業用オープンネットワークが注目されている。例えば、このネットワークにつながるノードの 1000 点デジタル I/O の読み書きに求められる速度は 30μs であり、1 つのイーサネットフレームでは 1486 バイトまでのプロセスデータを交換できる。この通信路の秘匿と改ざん防止を AES で実装しようとした場合、MAC 検証、復号、(解釈、書き換え) 暗号化、MAC 生成で 1 ブロックあたり 4 回の暗号化回路を call する必要がある。AES 1 ブロック暗号化に 100ns、処理速度にして 1.3Gbps かかるとすると 37.2μs 必要になり、AES では厳しい条件となる。これに対して、3 章の図 3.7 によると、例えば軽量暗号の Midori あるいは PRINCE をハードウェアの Unrolled 実装で実装することにより回路規模を AES より抑えた上でそれぞれ 3.9Gbps、3.6Gbps の処理速度で演算が出来るため、リアルタイム性が求められる用途での活用が期待される。

2.4.6 自動車

自動車の車内に暗号および認証機能をハードウェアで実装するにあたり暗号方式を選定する例として、3 章の図 3.14 ならびに図 3.15 を参考に、AES と比較し回路規模を抑えた上で処理速度が高速である Midori、PRINCE、PRESENT、SIMON などが候補として挙げることができる。

車の自動運転支援システムにおける暗号方式としては、レイテンシを低くするため、1 回分のラウンド処理ではなく複数のラウンド処理をハードウェアに実装することになる。このため、例えば 3 章の図 3.6 ならびに図 3.7 を参考に、AES と比較し回路規模を抑えた上で処理速度が高速である Midori、PRINCE、PRESENT、SIMON などの利用が考えられる。

車載情報端末における情報を保護するため、特にコンテンツ保護においては高スループットの暗号処理が求められることから、例えば 3 章の図 3.15 をもとに、軽量ブロック暗号 Midori の実装も考えられる。

参考文献

[1] AUTOSAR Specification of Module Secure Onboard Communication, https://www.autosar.org/fileadmin/files/releases/4-2/software-architecture/safety-and-security/standard/AUTOSAR_SWS_SecureOnboardCommunication.pdf

[2] CAR 2 CAR Communication Consortium, https://www.car-2-car.org/index.php?id=5

[3] Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication. NIST Special Publication 800-38B (2005), National Institute of Standards and Technology.

[4] Iwata, T.: New Blockcipher Modes of Operation with Beyond the Birthday Bound Security. In: FSE. Lecture Notes in Computer Science, vol. 4047, pp. 310–327. Springer (2006)

[5] Yasuda, K.: The Sum of CBC MACs Is a Secure PRF. In: CT-RSA. Lecture Notes in Computer Science, vol. 5985, pp. 366–381. Springer (2010)

[6] Yasuda, K.: A New Variant of PMAC: Beyond the Birthday Bound. In: CRYPTO. Lecture Notes in Computer Science, vol. 6841, pp. 596–609. Springer (2011)

[7] 総務省: 平成 27 年度版 情報通信白書 (2015), http://www.soumu.go.jp/johotsusintokei/whitepaper/ja/h27/pdf/index.html

第 3 章

軽量暗号の性能比較

本章では、同一プラットフォーム上で、同一の実装者または統一的な実装ポリシーによりハードウェア実装・ソフトウェア実装を行い、統一的な評価環境で性能比較を行った結果を示す。ハードウェア実装については軽量ブロック暗号を、ソフトウェア実装については軽量ブロック暗号及び認証暗号を評価対象とした。既存文献の調査では評価環境や実装者が異なることから暗号アルゴリズム間での比較が難しかった。

評価対象とした軽量暗号は、表 3.1 に示す 12 種類の軽量ブロック暗号と表 3.2 に示す 10 種類の認証暗号である。

表 3.1　評価対象とした軽量ブロック暗号

評価対象ブロック暗号	評価対象としたブロック長/鍵長	参照仕様書
AES	128/128	[8]
Camellia	128/128	[9]
CLEFIA	128/128	[25]
TDES	64/168	[11]
LED	64/128	[17]
PRINCE	64/128	[16]
PRESENT	64/80	[19]
Piccolo	64/80	[24]
TWINE	64/80	[26]
SIMON	32/64, 64/96, 64/128, 128/128	[15]
SPECK	32/64, 64/96, 64/128, 128/128	[15]
Midori	64/128, 128/128	[14]

3.1　ブロック暗号

3.1.1　ハードウェア実装評価

暗号回路の実装方式は用途に応じて様々な形態が考えられるが、本ガイドラインでは図 3.1 記載の「Unrolled 実装」、「Round 実装」、「Serial 実装」の 3 つの基本実装方式を採る。図 3.1 中、Round Function は、各暗号アルゴリズムで規定される基本関数の演算を行う組み合わせ回路を指す。また、12 通りのアルゴリズムに対するハードウェア実装評価では、暗号化演算のみと暗号化・復号演算の双方を同一のモジュールで実行し、その切り替えは制御信号でのみ行う実装の 2 通りに対して実装評価を行う。本ガイドラインでは前者の実装を「Enc」と後者を「Enc/Dec」

表 3.2　評価対象とした認証暗号

評価対象の認証暗号	参照仕様書
ACORN	[27]
AES-GCM	[10]
AES-OTR	[2]
Ascon	[1]
CLOC	[18]
SILC	[12]
JAMBU	[7]
Ketje	[3]
Minalpher	[4]
OCB	[5]

と表記する。

ブロック暗号アルゴリズムは一般に鍵スケジューリング機能と、暗号化・復号機能に分割できる。本ガイドラインにおける評価では、前記の両機能を持つ暗号回路を構成する。また、当該暗号化回路は、鍵スケジューリング機能、暗号化・復号機能ともに同一のクロックで動作する仕様で構成する。また、鍵スケジューリングにレジスタが不要なアルゴリズムでは、当該レジスタを削除して実装する。

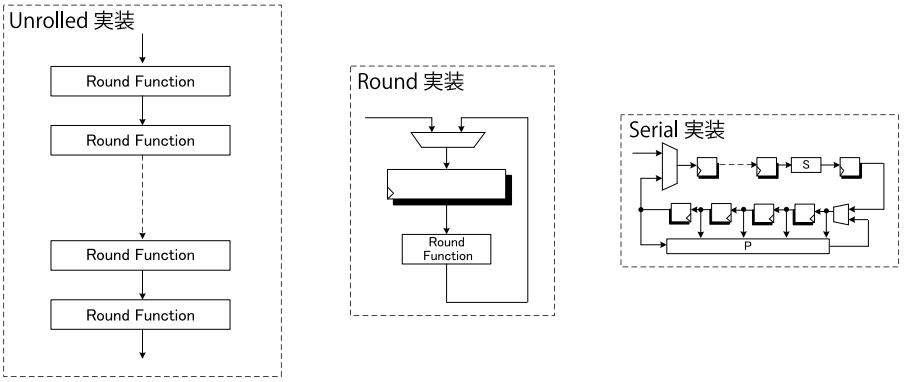

図 3.1　基本実装方式

3.1.1.1　性能比較

表 3.3 から表 3.5 として実装評価結果のデータを示す。また、インターフェースの回路規模を除いた各実装における回路規模の比較結果を表 3.6 に示す。また、各実装に対する回路規模、処理速度、ピーク電流、リーク電流についてグラフによる比較結果を図 3.2 から図 3.25 に示す。表中、(comp) はブロック暗号の構成要素である S-box を合成体上の演算回路として実装したことを表す。また、(table) は S-box をテーブル参照として記述し、合成ツール依存で回路を構成したことを表す。

まず、表 3.3 の注目点について述べる。Unrolled 実装の回路規模は AES を含む CRYPTREC のブロック暗号が軽量暗号・低遅延暗号 (LED は除く) に対して突出して大きい。この要因は表 3.7 に示されるように、S-box の性能差が支配的である。8bit S-box は遅延優先の Table 実装の場合、PRESENT や PRINCE などの 4bit S-box

に対して 100 倍以上大きくなる一方、遅延は 5 倍程度大きい。仮に 8bit S-box で 4bit S-box と同等の遅延性能を達成しようとした場合、4bit S-box よりもラウンド数を 1/4 程度にすることができないと遅延の観点からは効率が悪く、遅延が同等になっても S-box に関する回路規模は 100 倍近く大きくなることを意味する。この視点から PRINCE はほぼ AES と同じラウンド数となっているため、S-box の性能差がダイレクトに全体性能の差になって表れている。PRINCE は PRESENT に対してラウンド数がおよそ 1/3 であるが、表 3.8 に示すように P 層は PRESENT の方が高速であるため、3 倍の差はなく、回路規模、遅延ともに 2 倍程度 PRINCE が優れる結果となる。復号の影響については、GFN (Generalized Feistel Networks) 型や α-reflection property の効果により PRINCE、TWINE、Piccolo は PRESENT や LED の回路規模の差はさらに広がる。また、SIMON と Midori について PRINCE と同程度の処理性能を持つ。

Unrolled 実装の遅延についてもう一つ着目すべき点は、AES、LED など復号鍵を一旦生成しないと復号できないアルゴリズムや Camellia や CLEFIA のような中間鍵を生成するアルゴリズムはクリティカルパスにその分の遅延が乗るため、Unrolled 実装では不利になる。Piccolo や PRINCE は暗号化と暗号化・復号回路の最大動作周波数がほぼ同じように構成できる。

次に、Round 実装と Serial 実装との性能差を比較する。AES では 9kgate 程度の削減が可能であるのに対して、PRESENT、PRINCE は数百 gate から 1kgate の削減に留まる。S-box や P 層などの演算器を削減しても、表 3.7、表 3.8 からわかるようにその削減効果が限定的であるためである。その一方で処理性能は 1/10 以下となるため、例えば処理速度 / 回路規模などの指標を導入すれば効率が悪い。コードとしての可読性も悪いため、回路規模に対してなんらかの強い実装制約がない限り、軽量暗号で Serial 実装を採用する必要はないであろう。

最後に、Serial 実装について述べる。AES は文献 [22] では 2.4kgate で実装されていたが、本実装では 1kgate 程度増加している。その要因は、フリップフロップ 1 つあたりのゲート換算が文献 [22] よりもおよそ 1 ゲート程度大きいことや、合成によって挿入されるバッファ、制御回路の構成などが差分として挙げられる。文献 [22] に記載されるような Scan-FF を積極的に利用するような最適化を実施していないことも差分になる。PRESENT、PRINCE については本ガイドラインの結果でも AES より 1〜2kgate 程度小さい回路となっている。PRESENT、PRINCE 間に回路規模としての差はないが、サイクル数は PRINCE が PRESENT の 1/2 程度で実装できる。

表 3.3　Unrolled 実装の評価結果

アルゴリズム	ブロック長 [bit]	鍵長 [bit]	1ブロックあたりの処理サイクル数	最大動作周波数 [MHz]	処理速度 [Gbps]	回路規模 [kgate]	ピーク電力 [mW]	リーク電力 [uW]
Unrolled, Enc								
AES(table)(128/128)	128	128	1	25.7	3.3	112.4	-	-
AES(comp)(128/128)	128	128	1	13.4	1.7	78.8	175.6	939.6
Camellia(comp)(128/128)	128	128	1	7.8	1.0	60.2	136.5	706.7
CLEFIA(128/128)	128	128	1	5.7	0.7	74.6	195.5	891.0
SIMON(128/128)	128	128	1	24.7	3.2	63.2	172.2	685.9
SPECK(128/128)	128	128	1	3.2	0.4	44.4	73.0	417.0
Midori(128/128)	128	128	1	38.5	4.9	34.6	118.2	446.1
TDES(64/168)	64	168	1	10.0	0.6	55.4	111.9	652.2
LED(64/128)	64	128	1	6.9	0.4	74.5	99.1	824.0
PRINCE(64/128)	64	128	1	57.1	3.7	9.8	28.1	107.4
SIMON(64/128)	64	128	1	27.8	1.8	23.8	71.5	260.4
SPECK(64/128)	64	128	1	7.3	0.5	19.5	35.6	183.0
Midori(64/128)	64	128	1	46.5	3.0	12.3	34.9	149.0
SIMON(64/96)	64	96	1	41.3	2.6	20.3	56.7	218.1
SPECK(64/96)	64	96	1	7.6	0.5	18.6	35.4	174.7
PRESENT(64/80)	64	80	1	34.3	2.2	23.9	57.8	259.6
Piccolo(64/80)	64	80	1	18.0	1.2	19.1	61.0	224.8
TWINE(64/80)	64	80	1	24.8	1.6	19.5	43.8	221.2
SIMON(32/64)	32	64	1	39.4	1.3	9.0	30.5	97.4
SPECK(32/64)	32	64	1	15.3	0.5	8.2	17.3	78.0
Unrolled, Enc/Dec								
AES(table)(128/128)	128	128	1	11.4	1.5	208.4	337.2	2612.0
AES(comp)(128/128)	128	128	1	6.4	0.8	144.2	294.3	1734.3
Camellia(comp)(128/128)	128	128	1	7.7	1.0	63.4	133.8	754.9
CLEFIA(128/128)	128	128	1	5.7	0.7	74.3	195.5	891.0
SIMON(128/128)	128	128	1	13.0	1.7	74.1	187.0	803.7
SPECK(128/128)	128	128	1	1.1	0.1	69.1	127.1	672.5
Midori(128/128)	128	128	1	30.7	3.9	55.6	123.7	720.2
TDES(64/168)	64	168	1	9.6	0.6	56.5	112.9	673.3
LED(64/128)	64	128	1	3.1	0.2	215.4	103.1	815.6
PRINCE(64/128)	64	128	1	56.1	3.6	10.1	29.1	108.2
SIMON(64/128)	64	128	1	16.8	1.1	27.5	83.2	299.1
SPECK(64/128)	64	128	1	2.7	0.2	29.9	62.3	290.8
Midori(64/128)	64	128	1	37.7	2.4	20.6	37.1	256.4
SIMON(64/96)	64	96	1	21.5	1.4	23.8	62.9	255.3
SPECK(64/96)	64	96	1	2.9	0.2	28.6	57.8	278.0
PRESENT(64/80)	64	80	1	26.8	1.7	43.8	127.8	505.4
Piccolo(64/80)	64	80	1	16.3	1.0	22.8	64.8	264.0
TWINE(64/80)	64	80	1	13.1	0.8	25.6	50.9	292.2
SIMON(32/64)	32	64	1	23.6	0.8	10.4	30.9	111.8
SPECK(32/64)	32	64	1	6.9	0.2	12.4	27.5	121.7

表 3.4　Round 実装の評価結果

アルゴリズム	ブロック長 [bit]	鍵長 [bit]	1ブロックあたりの処理サイクル数	最大動作周波数 [MHz]	処理速度 [Gbps]	回路規模 [kgate]	ピーク電力 [mW]	リーク電力 [uW]
Round, Enc								
AES（comp)(128/128)	128	128	11	108.2	1.259	15.4	36.1	152.6
Camellia(comp)(128/128)	128	128	23	103.0	0.573	10.8	46.6	107.7
CLEFIA(128/128)	128	128	19	145.8	0.982	10.1	39.8	99.6
SIMON(128/128)	128	128	68	371.7	0.700	7.0	17.4	69.9
SPECK(128/128)	128	128	32	50.3	0.201	7.2	11.4	66.2
Midori(128/128)	128	128	20	386.1	2.471	7.1	11.9	79.7
TDES(64/168)	64	168	48	164.2	0.219	7.9	13.9	76.2
LED(64/128)	64	128	48	208.3	0.278	6.3	5.3	52.5
PRINCE(64/128)	64	128	13	234.2	1.153	5.1	16.4	47.1
SIMON(64/128)	64	128	44	371.7	0.541	5.3	12.4	51.1
SPECK(64/128)	64	128	27	95.8	0.227	5.3	10.5	48.3
Midori(64/128)	64	128	16	340.1	1.361	4.7	11.4	49.1
SIMON(64/96)	64	96	42	392.2	0.598	4.5	11.8	44.1
SPECK(64/96)	64	96	26	95.8	0.236	4.6	10.0	42.4
PRESENT(64/80)	64	80	33	326.8	0.634	4.1	4.7	33.4
Piccolo(64/80)	64	80	27	262.5	0.622	3.5	3.4	34.2
TWINE(64/80)	64	80	36	311.5	0.554	4.4	4.6	40.0
SIMON(32/64)	32	64	32	369.0	0.369	2.9	9.8	28.0
SPECK(32/64)	32	64	22	175.1	0.255	2.9	8.4	26.8
Round, Enc/Dec								
AES（comp)(128/128)	128	128	11	107.0	1.245	18.7	44.1	193.6
Camellia(comp)(128/128)	128	128	23	103.0	0.573	11.8	44.6	121.9
CLEFIA(128/128)	128	128	19	143.1	0.964	9.9	38.1	99.0
SIMON(128/128)	128	128	68	310.6	0.585	7.8	17.2	78.4
SPECK(128/128)	128	128	32	49.9	0.200	9.6	11.2	92.7
Midori(128/128)	128	128	20	271.0	1.734	8.4	11.9	96.9
TDES(64/168)	64	168	48	161.6	0.215	10.6	13.9	114.0
LED(64/128)	64	128	48	188.7	0.252	7.2	6.5	66.6
PRINCE(64/80)	64	128	13	224.7	1.106	5.3	18.7	50.3
SIMON(64/128)	64	128	44	342.5	0.498	6.0	12.4	58.2
SPECK(64/128)	64	128	27	93.5	0.222	6.7	10.6	63.2
Midori(64/128)	64	128	16	266.7	1.067	5.3	11.4	57.5
SIMON(64/96)	64	96	42	342.5	0.522	5.1	11.6	49.9
SPECK(64/96)	64	96	26	93.5	0.230	5.9	9.9	55.7
PRESENT(64/80)	64	80	33	280.9	0.545	4.7	4.9	44.8
Piccolo(64/80)	64	80	27	261.8	0.621	3.8	3.3	38.5
TWINE(64/80)	64	80	36	302.1	0.537	4.7	4.5	42.8
SIMON(32/64)	32	64	32	359.7	0.360	3.3	9.9	31.8
SPECK(32/64)	32	64	22	167.5	0.244	3.6	8.7	34.1

表 3.5　Serial 実装の評価結果

アルゴリズム	ブロック長 [bit]	鍵長 [bit]	1ブロックあたりの処理サイクル数	最大動作周波数 [MHz]	処理速度 [Gbps]	回路規模 [kgate]	ピーク電力 [mW]	リーク電力 [uW]
Serial, Enc								
AES(comp)(128/128)	128	128	226	112.2	63.6	6.3	18.5	76.8
Camellia(comp)(128/128)	128	128	360	109.5	38.9	6.6	14.4	66.1
CLEFIA(128/128)	128	128	175	114.2	83.5	6.2	13.1	61.3
SIMON(128/128)	128	128	4481	269.5	7.7	4.8	8.2	47.1
SPECK(128/128)	128	128	2177	291.5	17.1	5.0	8.2	48.4
Midori(128/128)	128	128	489	254.5	66.6	4.9	11.9	49.2
LED(64/128)	64	128	1872	344.8	11.8	5.6	2.2	50.0
PRINCE(64/128)	64	128	247	246.3	63.8	3.9	8.7	40.0
SIMON(64/128)	64	128	1537	309.6	12.9	3.7	4.8	36.2
SPECK(64/128)	64	128	993	339.0	21.8	3.9	5.4	37.4
Midori(64/128)	64	128	393	253.2	41.2	3.5	11.4	35.3
SIMON(64/96)	64	96	1441	328.9	14.6	3.3	4.5	31.7
SPECK(64/96)	64	96	929	314.5	21.7	3.4	5.1	33.1
PRESENT(64/80)	64	80	563	186.9	21.2	3.9	3.4	36.4
Piccolo(64/80)	64	80	433	300.3	44.4	3.5	2.0	28.5
TWINE(64/80)	64	80	324	277.8	54.9	4.1	2.8	29.6
SIMON(32/64)	32	64	577	389.1	21.6	2.2	3.7	20.8
SPECK(32/64)	32	64	417	390.6	30.0	2.3	5.5	21.9
Serial, Enc/Dec								
AES(comp)(128/128)	128	128	226	108.6	61.5	7.2	14.5	61.2
Camellia(comp)(128/128)	128	128	360	108.3	38.5	7.3	14.8	63.1
CLEFIA(128/128)	128	128	175	113.1	82.7	6.8	12.5	59.3
SIMON(128/128)	128	128	4481	277.8	7.9	5.6	9.7	57.4
SPECK(128/128)	128	128	2177	316.5	18.6	5.9	8.3	57.2
Midori(128/128)	128	128	489	204.1	53.4	5.3	11.9	53.9
LED(64/128)	64	128	1872	303.0	10.4	6.9	1.4	34.5
PRINCE(64/128)	64	128	247	245.7	63.7	3.8	8.4	36.2
SIMON(64/128)	64	128	1537	277.0	11.5	4.5	5.6	45.3
SPECK(64/128)	64	128	993	317.5	20.5	4.8	7.6	46.2
Midori(64/128)	64	128	393	220.3	35.9	3.8	11.4	37.7
SIMON(64/96)	64	96	1441	298.5	13.3	3.9	5.1	39.0
SPECK(64/96)	64	96	929	280.1	19.3	4.1	7.6	40.1
PRESENT(64/80)	64	80	563	170.9	19.4	4.5	2.4	25.8
Piccolo(64/80)	64	80	433	292.4	43.2	3.7	2.0	23.4
TWINE(64/80)	64	80	324	270.3	53.4	4.2	2.6	28.4
SIMON(32/64)	32	64	577	299.4	16.6	2.6	4.1	25.7
SPECK(32/64)	32	64	417	295.9	22.7	2.8	6.3	27.3

表 3.6　各実装の回路規模

	インターフェースの除いた暗号回路のみの回路規模[kgate]					
アルゴリズム	Unrolled, Enc	Unrolled, Enc/Dec	Round, Enc	Round, Enc/Dec	Serial, Enc	Serial, Enc/Dec
AES(table)(128/128)	109.7	205.6	—	—	—	—
AES（comp)(128/128)	76.1	141.5	12.4	15.6	3.2	4.1
Camellia(comp)(128/128)	57.4	60.6	8.0	9.0	4.1	4.3
CLEFIA(128/128)	71.5	71.5	7.3	7.1	3.6	3.8
SIMON(128/128)	60.4	71.3	4.3	5.0	2.1	2.9
SPECK(128/128)	41.6	66.4	4.4	6.8	2.2	3.1
Midori(128/128)	31.8	52.9	4.3	5.6	2.2	2.6
TDES(64/168)	52.8	53.8	5.3	7.9	—	—
LED(64/128)	71.9	212.9	3.8	4.7	3.0	4.3
PRINCE(64/128)	7.8	8.1	2.7	3.0	1.6	1.8
SIMON(64/128)	21.8	25.4	3.2	3.9	1.7	2.5
SPECK(64/128)	17.4	27.8	3.2	4.6	1.8	2.7
Midori(64/128)	10.2	18.5	2.6	3.2	1.5	1.7
SIMON(64/96)	18.4	21.9	2.7	3.2	1.4	2.0
SPECK(64/96)	16.8	26.8	2.8	4.1	1.6	2.3
PRESENT(64/80)	22.0	42.1	2.2	2.9	2.0	2.8
Piccolo(64/80)	17.4	21.1	1.6	1.9	1.1	1.3
TWINE(64/80)	17.8	23.9	2.7	2.9	2.4	2.5
SIMON(32/64)	7.8	9.2	1.7	2.1	1.0	1.4
SPECK(32/64)	7.0	11.2	1.7	2.4	1.1	1.6

表 3.7　S-box の比較

Module	Area [gate]	Path delay [ns]
AES 8bit S-box (Table)	3,194	2.43
AES 8bit S-box (Composite)	315	5.75
PRESENT 4bit S-box (Table)	26	0.57
PRINCE 4bit S-box (Table)	18	0.48

表 3.8　P 層の比較

Module	Area [gate]	Path delay [ns]
AES 128bit permutation	864	0.89
PRESENT 64bit permutation	0	0
PRINCE 64bit permutation	192	0.51

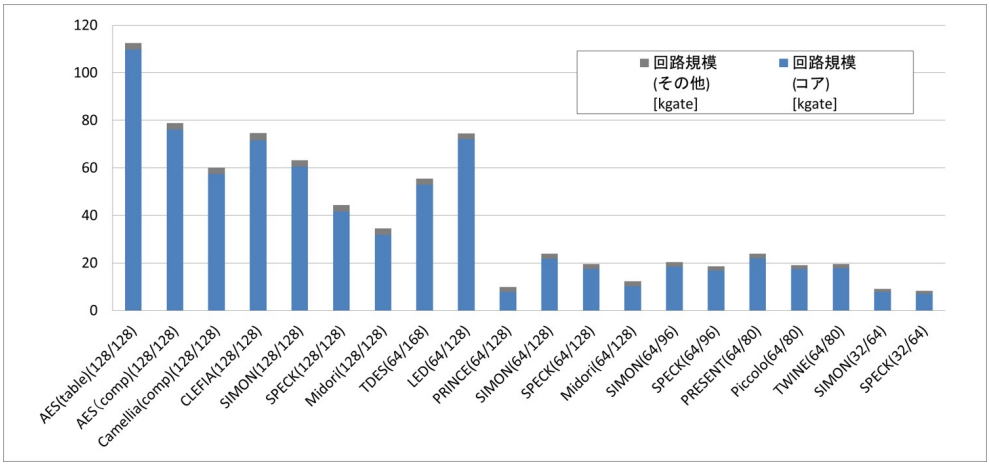

図 3.2　Enc,Unrolled 実装の回路規模

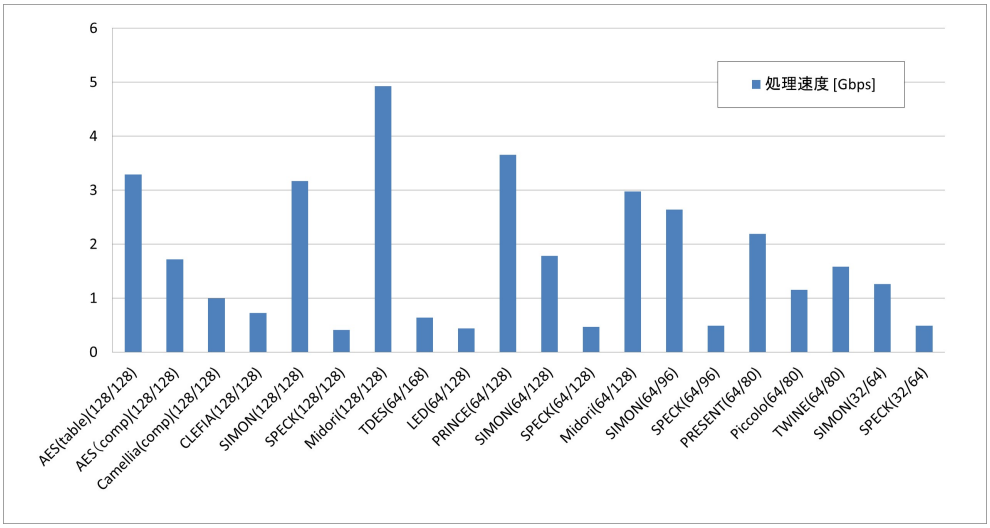

図 3.3　Enc,Unrolled 実装の処理速度

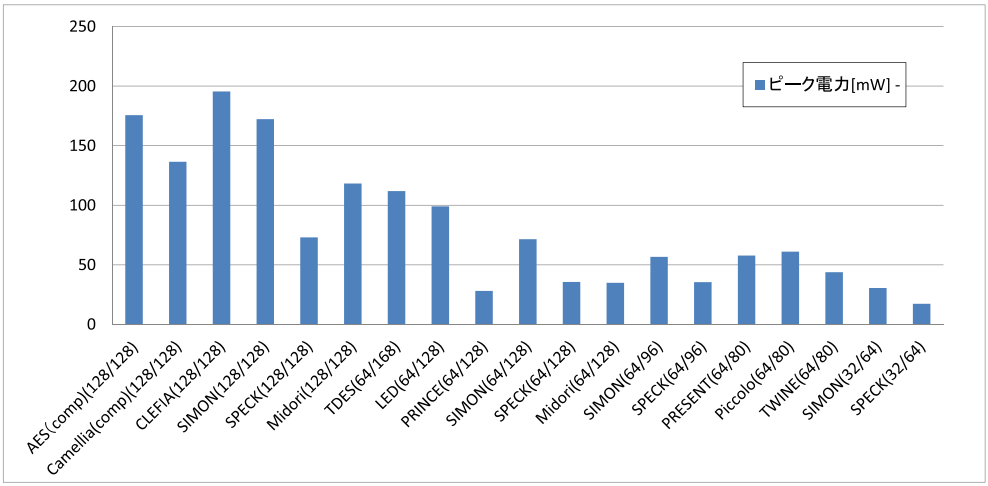

図 3.4　Enc,Unrolled 実装のピーク電流

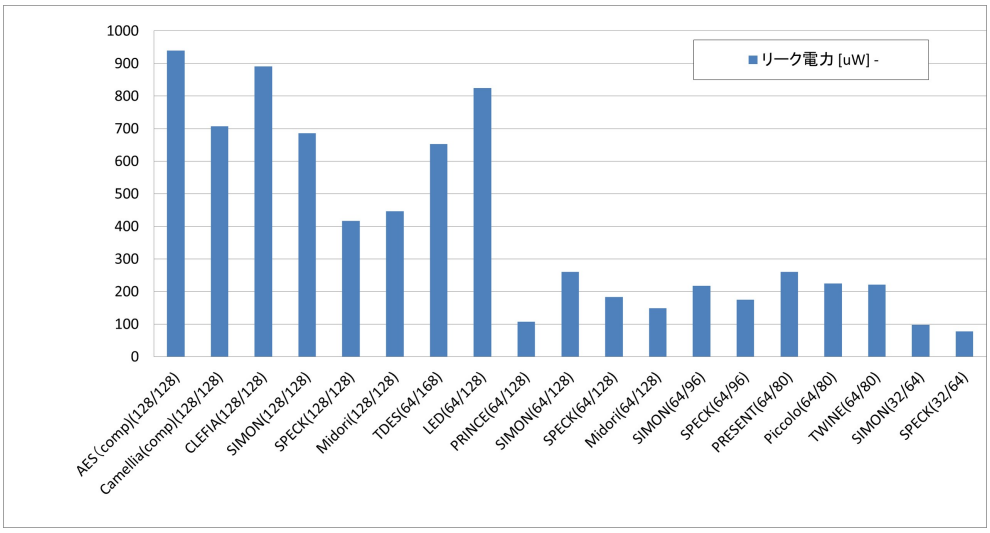

図 3.5　Enc,Unrolled 実装のリーク電流

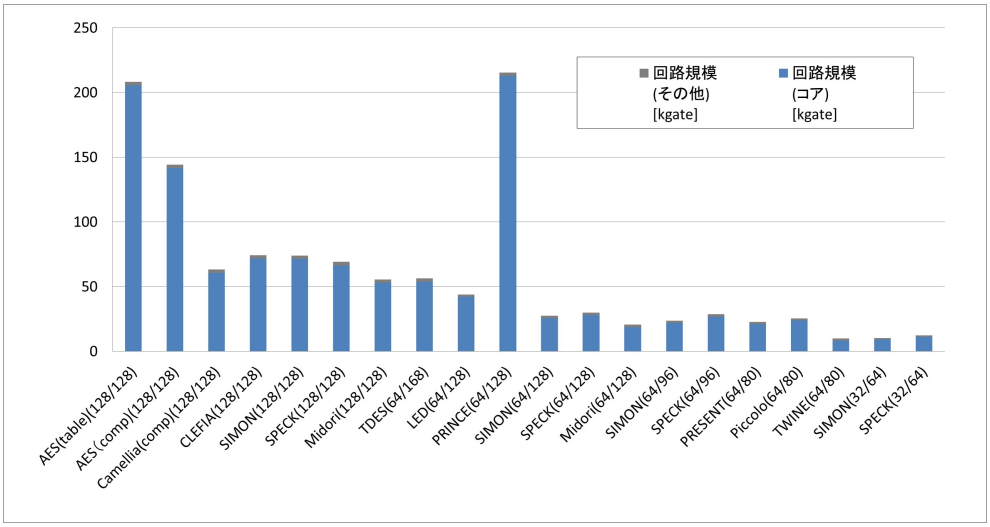

図 3.6　Enc/Dec,Unrolled 実装の回路規模

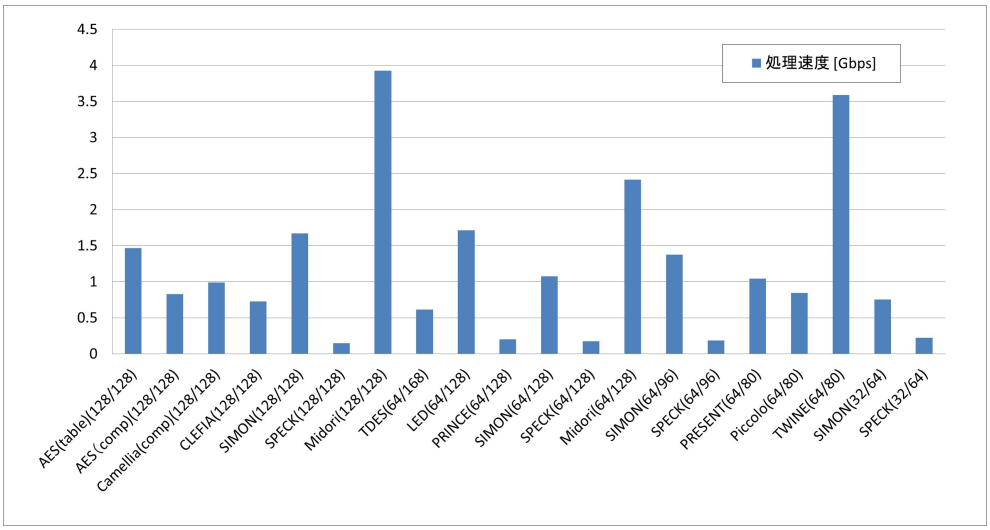

図 3.7　Enc/Dec,Unrolled 実装の処理速度

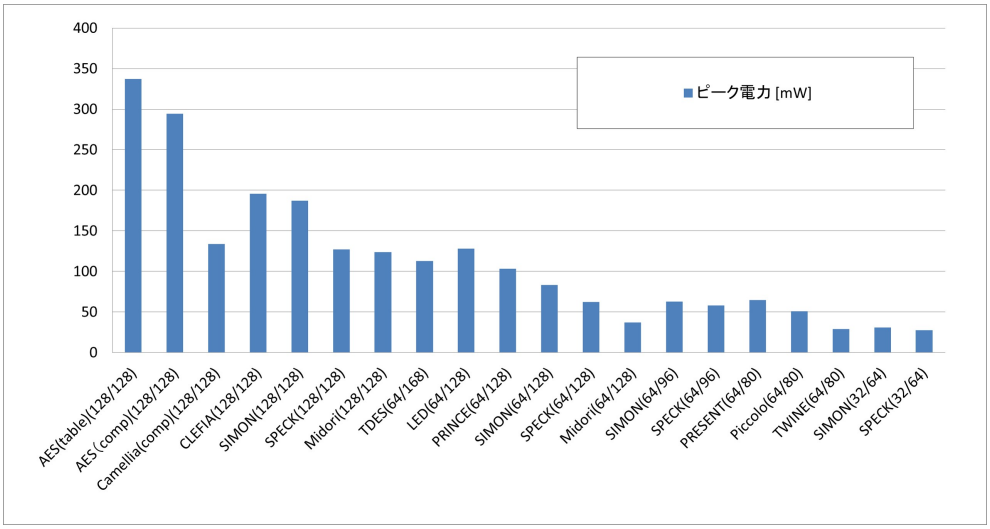

図 3.8　Enc/Dec,Unrolled 実装のピーク電流

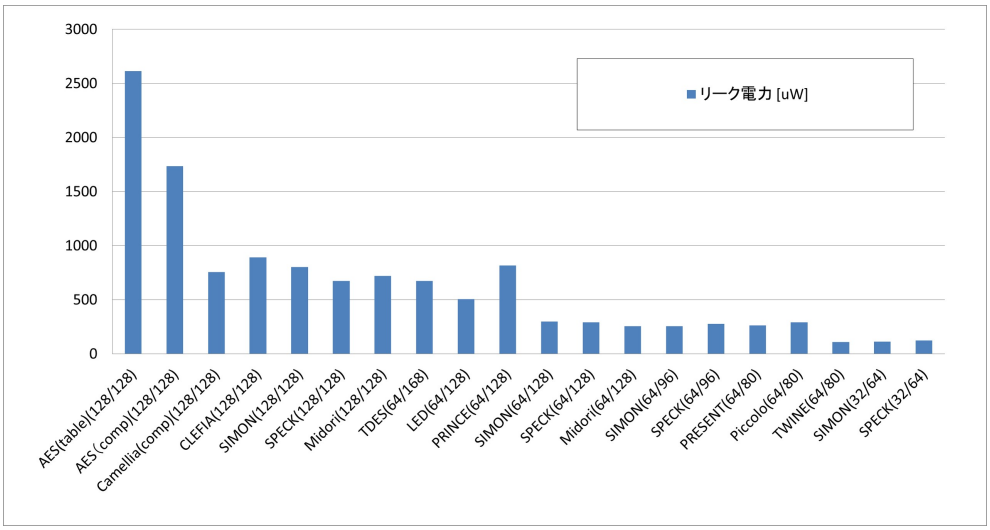

図 3.9　Enc/Dec,Unrolled 実装のリーク電流

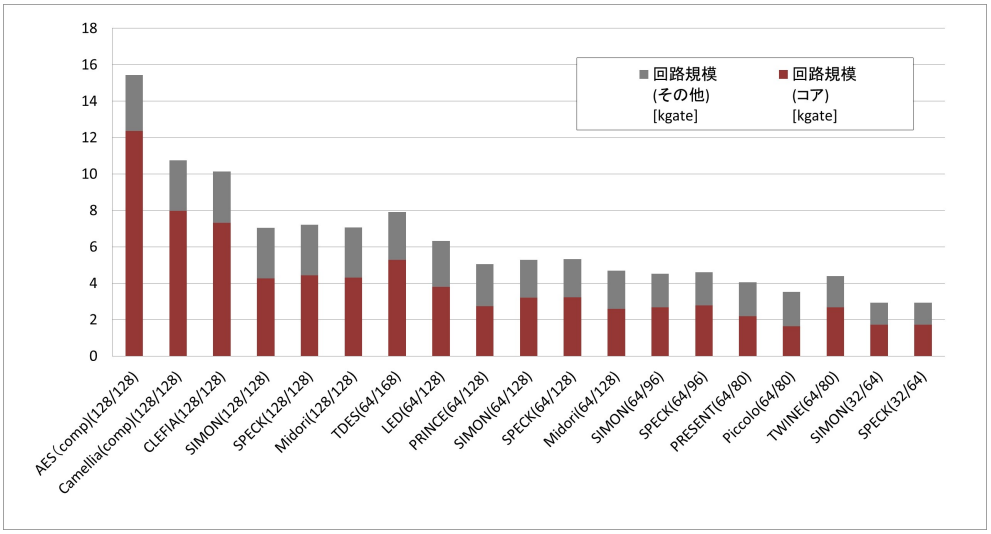

図 3.10　Enc,Round 実装の回路規模

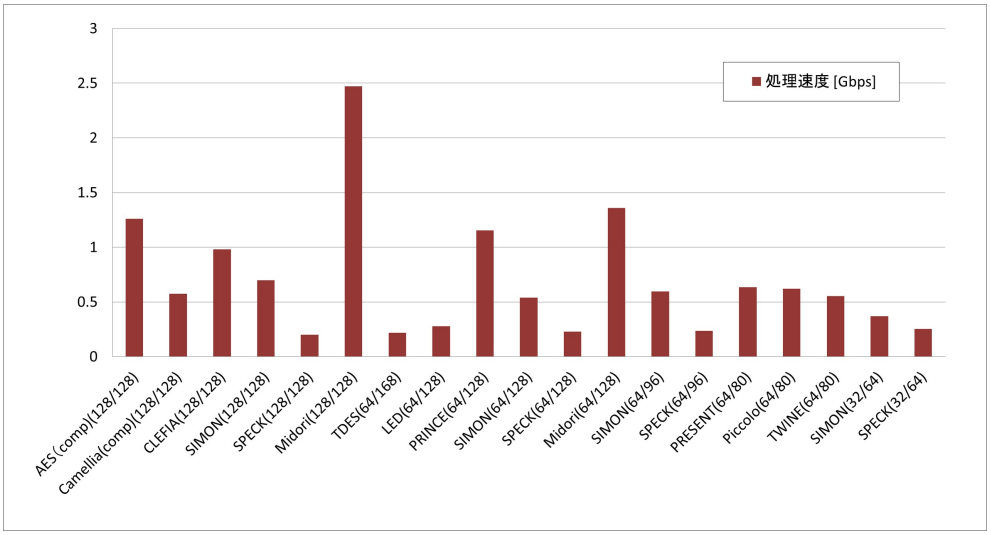

図 3.11　Enc,Round 実装の処理速度

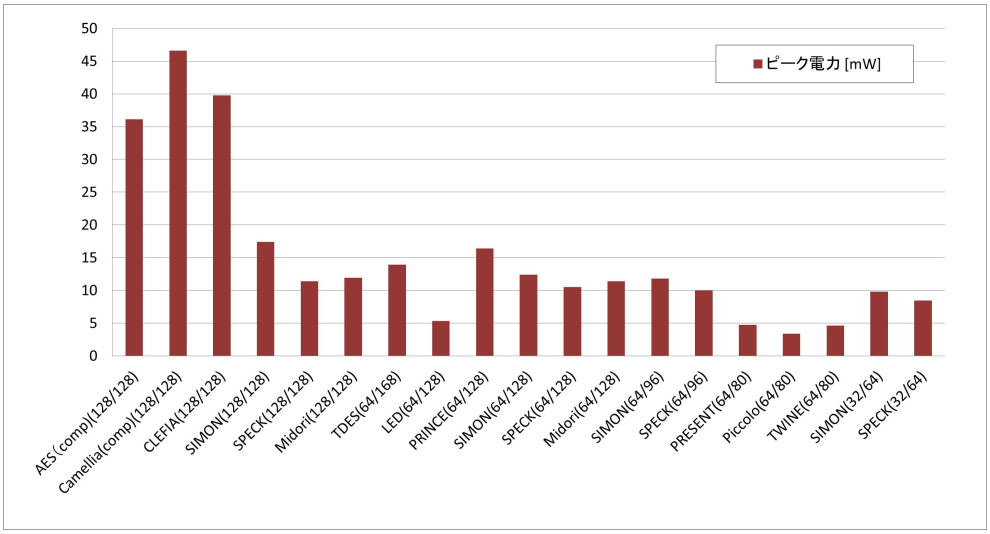

図 3.12　Enc,Round 実装のピーク電流

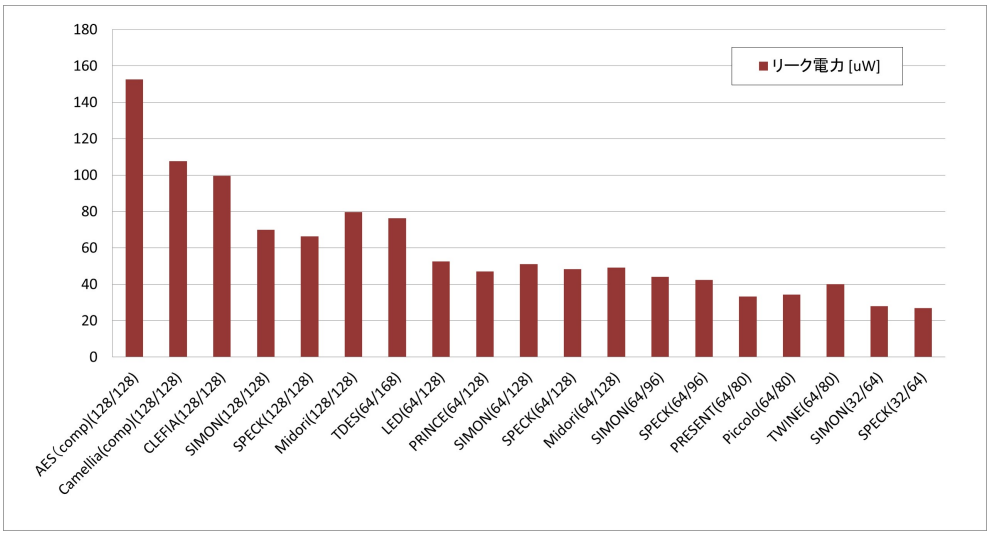

図 3.13　Enc,Round 実装のリーク電流

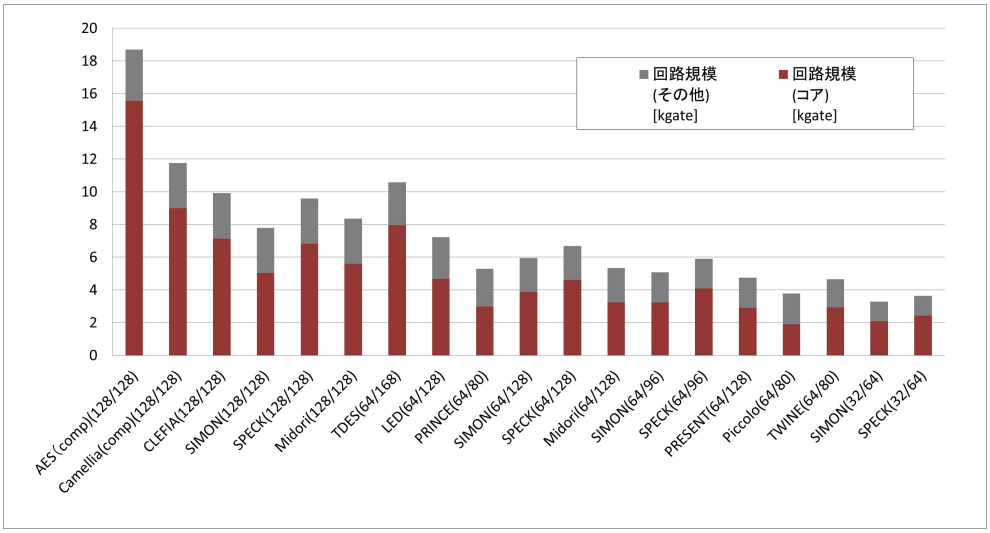

図 3.14　Enc/Dec,Round 実装の回路規模

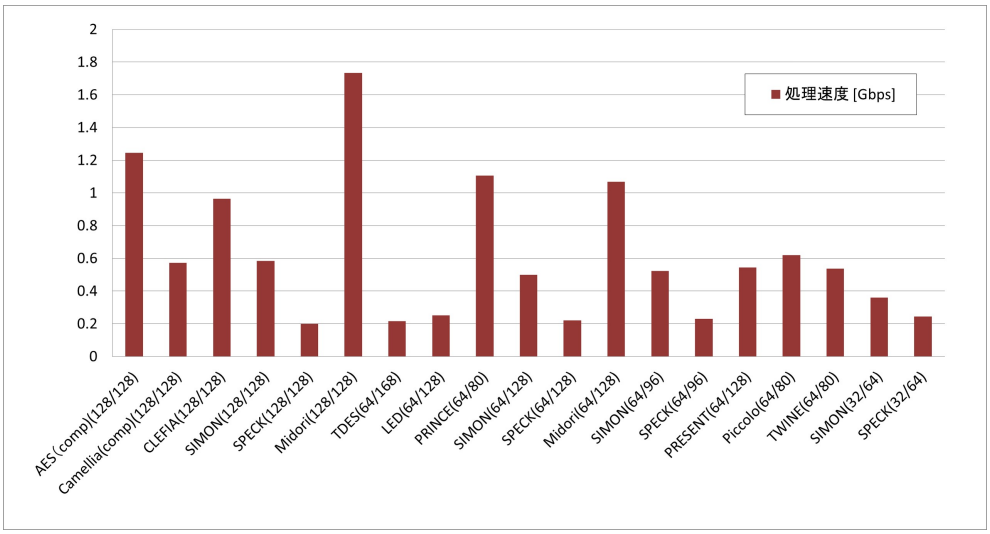

図 3.15　Enc/Dec,Round 実装の処理速度

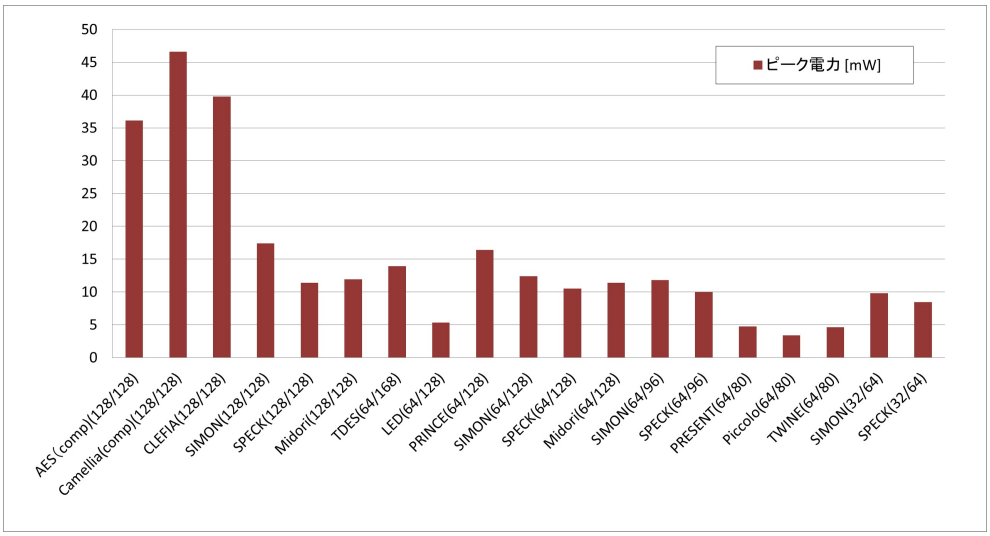

図 3.16　Enc/Dec,Round 実装のピーク電流

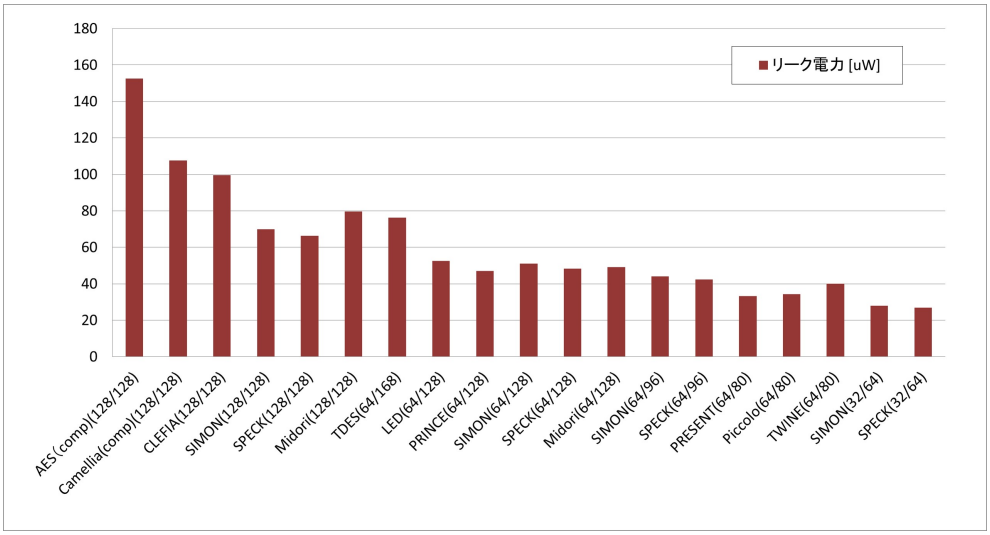

図 3.17　Enc/Dec,Round 実装のリーク電流

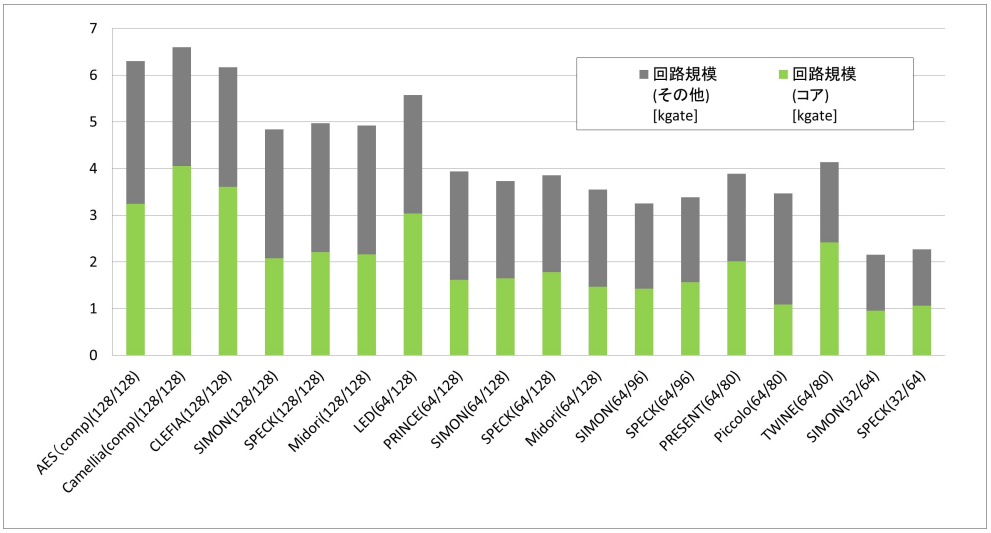

図 3.18　Enc,Serial 実装の回路規模

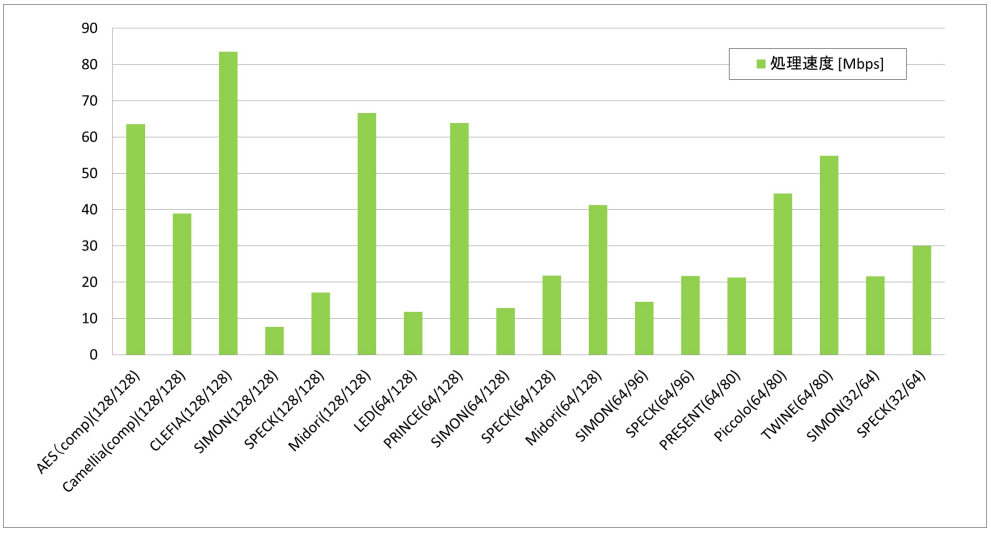

図 3.19　Enc,Serial 実装の処理速度

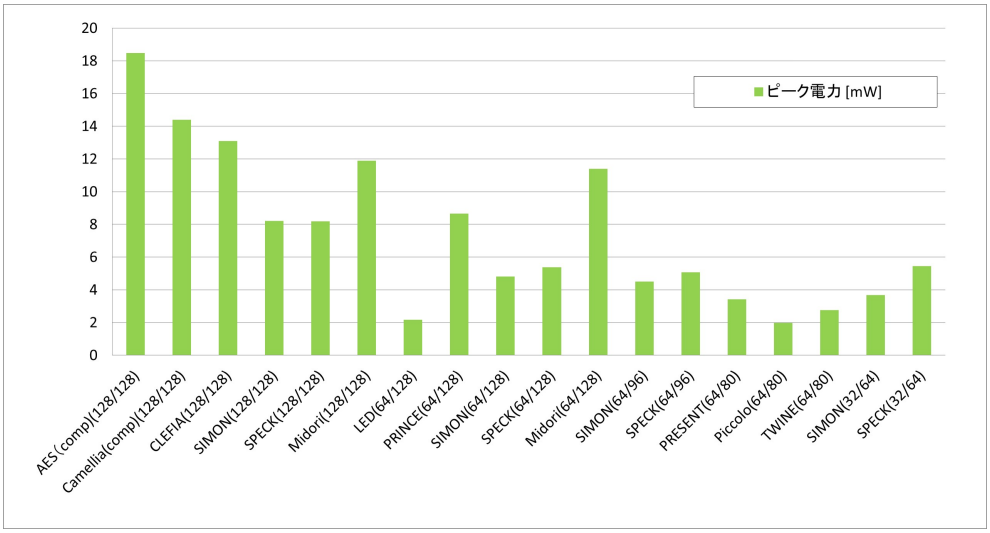

図 3.20　Enc,Serial 実装のピーク電流

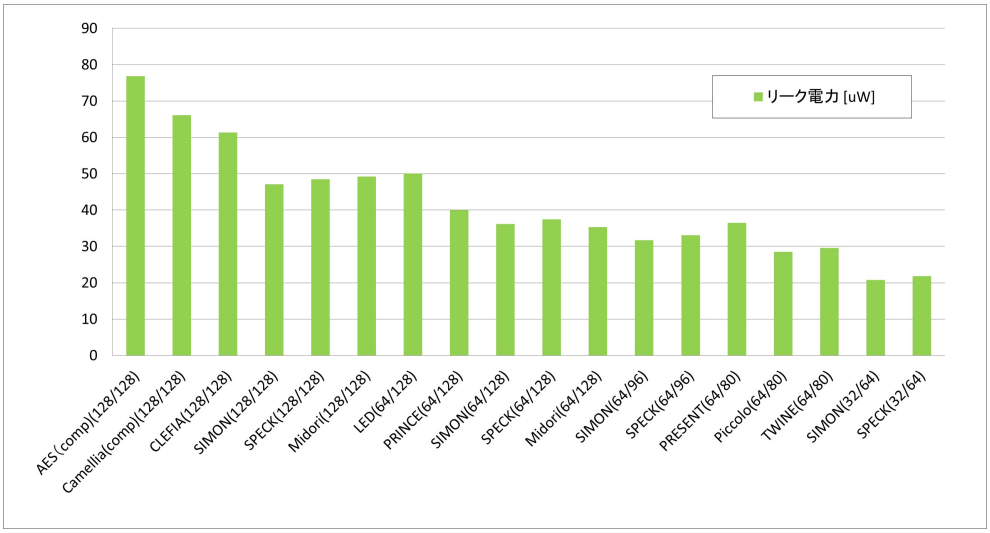

図 3.21　Enc,Serial 実装のリーク電流

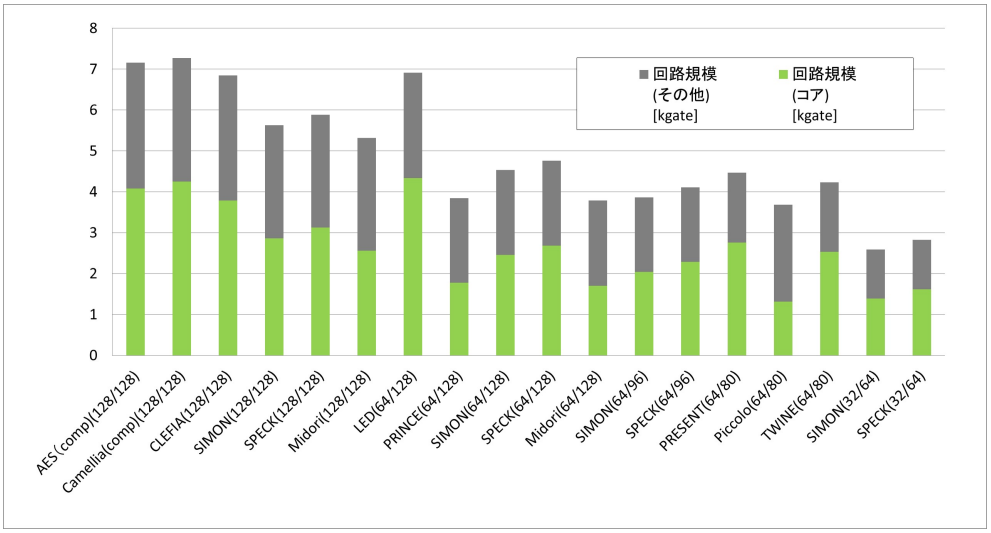

図 3.22　Enc/Dec,Serial 実装の回路規模

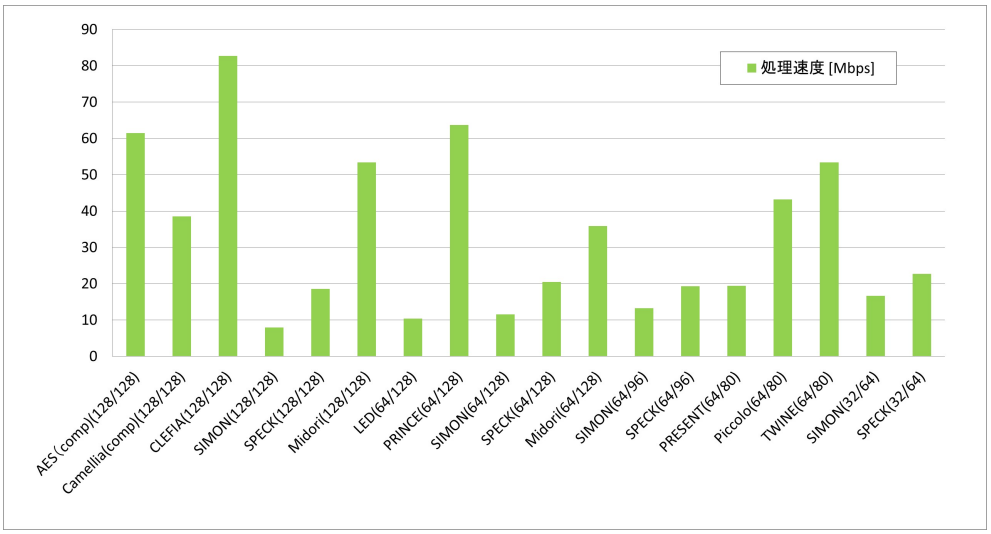

図 3.23　Enc/Dec,Serial 実装の処理速度

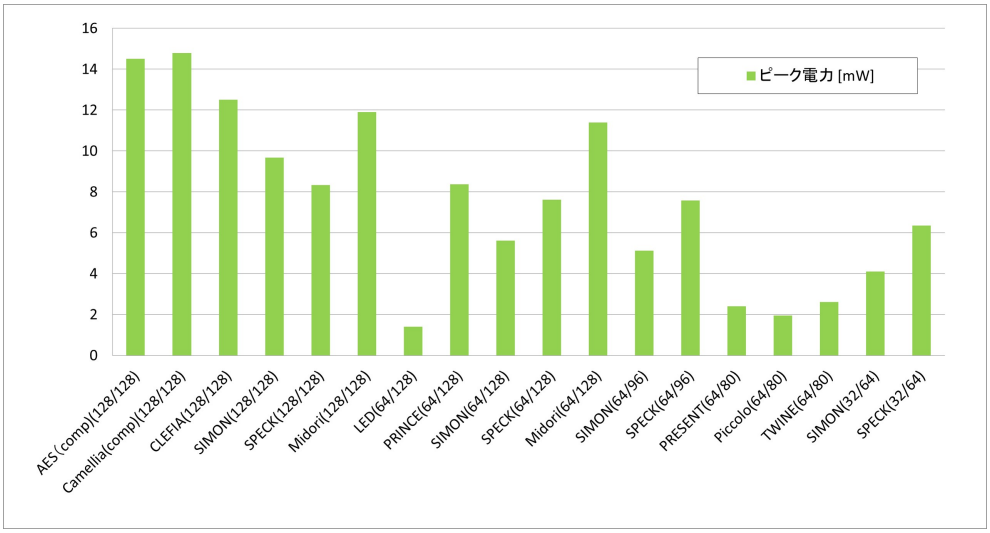

図 3.24　Enc/Dec,Serial 実装のピーク電流

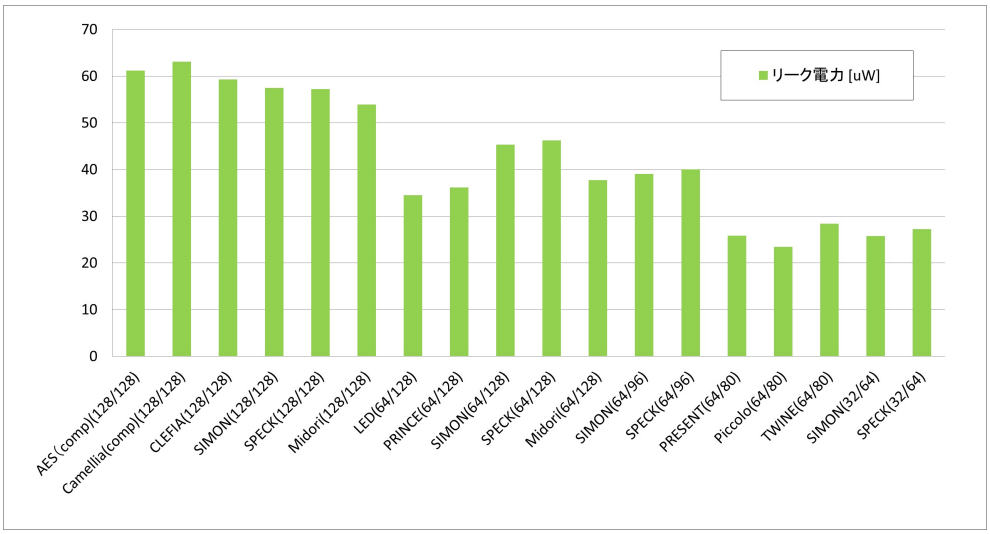

図 3.25　Enc/Dec,Serial 実装のリーク電流

3.1.1.2　評価方法の概要

本節では、評価方法の概要を示す。今回の評価では、各種軽量暗号を、オープンソースの CMOS セルライブラリを利用して、回路リソースの使用量及び最大動作周波数などに関するデータを計測した。実装環境を表 3.9 に示す。

表 3.9　実装環境

論理合成ツール	Design Compiler (Version G-2012.06-SP5)
パワー解析ツール	PrimeTime PX (Version G-2012.06-SP3-2)
合成制約	面積最小
ライブラリ	NANGATE Open Cell Library (45-nm CMOS) `http://www.nangate.com/`
遅延条件	NangateOpenCellLibrary_slow (最悪条件の仮想遅延)

以下に実装する論理回路の機能概略を述べる。

F1. 鍵長は 80bit 以上でかつ規定される最小のパラメータで評価を行う。但し LED に関してはテストベクタが提供されている 128bit 鍵長で評価する。
F2. 暗号化、暗号化・復号回路の実装とする。
F3. CPU のコプロセッサとしての利用を想定し、コンパクトで低電力とされる APB バス [13] 接続が可能な設計とする。

APB バスと暗号回路のブロック図を図 3.26 に示す。図中の信号の意味は以下の通りである。

* PCLK:バスクロック信号
* PRESETn:非同期リセット信号
* PADDR[31:0]:アドレス信号
* PSEL:IP 選択信号
* PENABLE:イネーブル信号

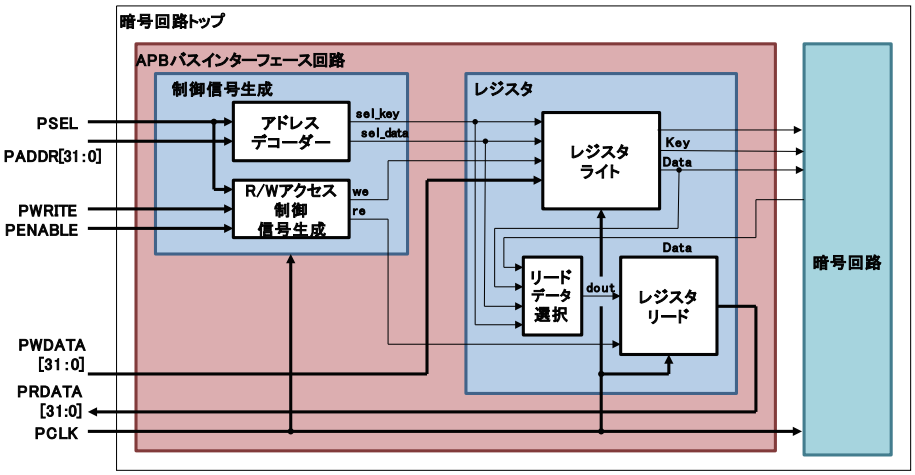

図 3.26　APB バスと暗号回路

* PWRITE:ライト信号、1: ライト, 0: リード
* PWDATA[31:0]:ライトデータ
* PRDATA[31:0]:リードデータ

このほか、APB 信号規定としては、PSTRB[3:0](ライトストローブ信号) や PREADY(APB 転送延長信号) があるが、今回の評価では使用していない。

次に設計方針を述べる。

P1. 各アルゴリズムに対して 3 種類の実装を行う: (i) 典型的な Round ベースの実装に加え、(ii) 1 サイクルで処理が完了する Unrolled 実装、(iii) データパスを S-box のサイズとする Serial 実装を行う。

P2. 鍵スケジュールは on-the-fly で実装する。

P3. CMOS セルライブラリを直接インスタンスするような最適化は行わず、ライブラリ非依存で合成可能な記述とする。

以上の方針に基づき設計した論理回路に対してサイクル数、最大動作周波数 (最大遅延)、スループット、ゲート数、ピーク電力、リーク電力を評価している。

3.1.2 ソフトウェア実装評価

本節では、軽量ブロック暗号を、組み込みマイコン上の限定されたメモリリソースのもとで、ソフトウエア実装した場合の速度性能について計測、比較した結果を示す。

3.1.2.1 性能評価

本節ではルネサスエレクトロニクス社製 16 ビットマイコン RL78 において、9 つのブロック暗号を実装した結果を示す。以下の表の Category の欄は、ROM-Min が ROM サイズを最小化する実装を行ったもの、(n, m) は、ROM サイズ n バイト以下（n=512, 1024)、RAM サイズ m バイト以下（m=64, 128) で実装を行ったもの、また Fast は、おおむね ROM サイズ 2K バイト以下で高速化を目指して実装を行ったものである。すべての実装は、暗号化・復号と並行して鍵スケジュールを行う実装方式 (on-the-fly key scheduling) を採用している。

■AES

表 3.10 に、RL78 にて AES を実装した結果を示す。暗号化のみを実装する場合、RAM サイズは 64 バイトで十分であったため、RAM 128 バイトの実装は省略した。また暗号・復号の両方を実装する場合は、S-box だけで ROM 512 バイトを消費するので、ROM 1024 バイトのカテゴリのみ実装を行っている。

実装方法は基本的に、メモリサイズの制約が厳しくなるにつれ、ラウンド内のループを増やしていくものであるが、参考までにコードサイズに大きく影響する MixColumns の実装方法を一番右の欄に示した。ここで M4 は MixColumns の行列乗算 4 つを独立にコードとしてもっているもの、M1 は行列乗算 1 つだけのコードをもち、各ラウンドこれを 4 回ループさせて MixColumns を実行するもの、また MQ は行列の 1 行分だけの演算をするコードをもち、この二重ループ合計 16 回により 1 ラウンド分の MixColumns 演算をするものである。

ROM 最小化に関しては、[20] よりさらに小型化されており、暗号化で ROM 430 バイトは現在知られているもっとも小さい実装であると思われる。一方 AES の暗号化・復号を実装する場合に ROM 1024 バイトというのは大きな制約であり、内部でループを多用することによる性能低下が避けられない。実際 ATtiny での既存実装がほとんど ROM サイズ 1500 バイト以上であることも、このような事情が背景にある。一方、高速化に関しては、ROM 2K バイトあれば、ほぼすべてのループがアンロールできるので、3500 cycles/block 程度が RL78 プロセッサにおける AES の最高性能であると考えられる。

表 3.10　RL78 での AES の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed	Method
AES (E)	ROM-Min	430	66 + 14	$8753n$	-	MQ
AES (E)	(512,64)	510	48 + 10	$5302n$	-	M1
AES (E)	(1024,64)	926	48 + 8	$3554n$	-	M4
AES (ED)	(1024,64)	1020	48 + 14	$8193n$	$9719n$	M1
AES (ED)	(1024,128)	1020	66 + 14	$6946n$	$1380+8490n$	M1
AES (ED)	Fast	2044	50 + 10	$3554n$	$753+5527n$	M4

■Camellia

表 3.11 に、RL78 にて Camellia を実装した結果を示す。Camellia には 128 ビットの回転シフト演算が多数含まれているが、その回転数に規則性がないため、ROM サイズが大きくなる。また FL 関数や定数 Σ のサイズも小さくなく、S-box を 1 個だけ持った場合でも最小 ROM サイズは 749 バイトであった。また暗号化と復号両方を実装する場合には、コードサイズを減少させるためにサブルーチン化が必要となり、この結果利用するスタックが増加し RAM 64 バイトでは実装が困難であった。

一方、ROM サイズが 2K バイトという条件で暗号化だけを行う場合には、AES と同程度の速度が得られている。したがって、さらに ROM を利用できるならば、復号においては Feistel 構造を持つ Camellia は AES よりも高速となることが期待される。このように Camellia はメモリに比較的余裕がある場合に高性能となる方式である。

これらの実装のうち、ROM-Min 以外の暗号化のみを実装したコードは、必要な回転シフトサブルーチン群を個別に内部で持っているが、暗号化・復号の両方を実装したコードは、8 ビット回転シフトルーチンと 1 ビット回転シフトルーチンだけをコードとしてプログラムの内部に持ち、オンラインで必要なビット数の回転シフトを実現するものである。また ROM-Min 実装は 1 ビット回転シフトルーチンだけを内部にコードとして持ち、オンラインで必要なビット数の回転シフトを実現するものである。参考までに、表の一番右の欄には、それぞれの実装が何ビットの回転シフトルーチンを独立に持っているかを示した。

表 3.11　RL78 での Camellia の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed	Method
Camellia (E)	ROM-Min	749	56 + 16	58382n	-	1
Camellia (E)	(1024,64)	1024	54 + 10	889+4709n	-	17,15
Camellia (E)	(1024,128)	1018	56 + 14	884+4520n	-	17,15
Camellia (E)	Fast	1995	66 + 12	740+3638n	-	34,30,17,15
Camellia (ED)	(1024,128)	1021	58 + 22	3034+25470n	3907+25498n	8,1

■CLEFIA

表 3.12 に、RL78 にて CLEFIA を実装した結果を示す。CLEFIA は鍵スケジュール部において、中間鍵を格納するメモリサイズが多いため RAM サイズ 64 バイトで実装することは困難である。また S-box や MixColumns が２つあることに加え、定数が 384 バイトあるなど ROM サイズも大きく、ROM 512 バイトで実装することは不可能である。

一方、2 つの S-box のうちの 1 つと定数は実行中に動的に生成することも可能であり、ROM-Min 実装と暗号・復号両方を実装したものについては、これらを実際動的に生成させている。また暗号のみの実装で (1024,128) のものは、定数だけを動的に生成させ、S-box は 2 つ ROM にもつ実装を行っている。この結果 ROM 最小実装でも 800 バイト必要であった。参考のため、表の一番右の欄にどの実装方法を採用したかを記載している。S は S-box を動的生成、C は定数を動的生成させたことを意味している。

表 3.12　RL78 での CLEFIA の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed	Method
CLEFIA (E)	ROM-Min	800	58 + 22	23854n	-	SC
CLEFIA (E)	(1024,128)	1021	58 + 16	12351n	-	C
CLEFIA (E)	Fast	1681	74 + 14	3010+5899n	-	
CLEFIA (ED)	(1024,128)	1018	90 + 26	19879n	20797n	SC

■TDES

表 3.13 に、RL78 にて TDES を実装した結果を示す。TDES は RAM サイズは 64 バイトで十分であるものの、不規則なビット演算が中心のアルゴリズムであるため、S-box やビット位置を示す表などだけで 400 バイト以上の ROM を占有し、したがって ROM 512 バイトで全体を実装することはできない。

ROM 1832 バイトの実装は、速度にかかわる部分はほぼアンロールしているので、このプロセッサでのほぼ最高性能に近い速度が出ていると考えられる。この結果から最高性能で比較すると TDES は AES の 1/15 程度の速度性能と言うことができる。

表 3.13　RL78 での TDES の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed
TDES (E)	ROM-Min	958	50 + 14	111183n	-
TDES (E)	(1024,64)	1024	50 + 14	77708n	-
TDES (E)	Fast	1832	50 + 8	26697n	-
TDES (ED)	(1024,64)	1019	50 + 14	87879n	87543n

■LED

表 3.14 に、RL78 にて LED を実装した結果を示す。LED は 4 ビットを 1 ワードとする AES の構造に近い。このような構造の暗号の場合 8 ビットの平文や鍵をどこかの段階で 2 つの 4 ビットデータに分割する必要がある。これをどの段階で行うか（暗号化前に行っておくか、実行時に分割するか）でメモリサイズと速度のトレードオフが存在する。また 4 ビット S-box をあらかじめ ROM から RAM に転送しておくと、コードサイズと RAM サイズ

の増加と引き換えに速度が向上するという別のトレードオフがある。これは、RL78 は ROM データの読み出しに 4 サイクルかかるのに対して、RAM データの読み出しは 1 サイクルで済むからである。さらに GF(16) 上の 2 倍算を実行時に行うのか RAM に搭載した表で行うかのトレードオフも存在する。

このようなさまざまのトレードオフの中で、それぞれの与えられたメモリサイズ条件に対してどれが最も高速になるかは複雑なパズルである。参考までに、表の一番右の欄にどの実装を採用したかを示す記号を示した。ここで S は S-box を RAM 転送していること、G は GF(16) 上の二倍算のテーブルを RAM 転送していること、T は平文 (暗号文) を最初に 4 ビット分割していること、K は鍵を最初に 4 ビット分割していることをそれぞれ示している。

なお暗号化のみの (1024,128) 実装は、主要な部分をすべてアンロールしているものなので、これが RL78 での最高速性能に近いと考えられる。

表 3.14　RL78 での LED の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed	Method
LED (E)	ROM-Min	298	54 + 12	$36779n$	-	T
LED (E)	(512,64)	510	54 + 10	$18055n$	-	S
LED (E)	(512,128)	504	100 + 12	$17207n$	-	SGTK
LED (E)	(1024,64)	956	54 + 10	$15899n$	-	S
LED (E)	(1024,128)	1023	100 + 8	$14478n$	-	SGTK
LED (ED)	(512,64)	508	54 + 10	$35726n$	$32219n$	T
LED (ED)	(512,128)	508	54 + 14	$33950n$	$31787n$	T
LED (ED)	(1024,64)	1007	54 + 10	$17717n$	$17788n$	S
LED (ED)	(1024,128)	1023	100 + 8	$16753n$	$17472n$	SGT

■PRINCE

表 3.15 に、RL78 にて PRINCE を実装した結果を示す。このうち RAM 128 バイトの実装はすべて 2 つの S-box の合計 32 バイトを RAM 転送することにより高速化を目指したもので、表の一番右に記号 S で示している。また高速実装のものは S-box 2 つを並列化した 256 バイトのテーブルを 2 つもつことにより、高速化を目指した実装であり記号 S8 で示している。

PRINCE は鍵スケジュール処理がほとんどなく、しかも暗号化と復号がほとんど同じ処理で実現できるという特長をもっているが、一方で定数が少なくないことと Matrix 演算のコードのオーバーヘッドから、最小実装の ROM サイズは他の lightweight 64 ビットブロック暗号に比べると大きくなっている。

表 3.15　RL78 での PRINCE の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed	Method
PRINCE (E)	ROM-Min	424	42 + 22	9905n	-	
PRINCE (E)	(512,64)	512	42 + 12	7611n	-	
PRINCE (E)	(512,128)	511	74 + 12	7320n	-	S
PRINCE (E)	(1024,64)	1019	42 + 12	4928n	-	
PRINCE (E)	(1024,128)	1020	74 + 12	4541n	-	S
PRINCE (E)	Fast	1789	42 + 8	3307n	-	S8
PRINCE (ED)	(512,64)	511	44 + 20	9925n	10050n	
PRINCE (ED)	(512,128)	511	76 + 24	9541n	9810n	S
PRINCE (ED)	(1024,64)	1007	42 + 12	5117n	5214n	
PRINCE (ED)	(1024,128)	1017	74 + 12	4745n	4832n	S

■PRESENT

表 3.16 に、RL78 にて PRESENT を実装した結果を示す。PRESENT は規則正しい構造をもつため、この構造を利用して極めて小さいコードを作ることが可能である。暗号化での ROM 最小実装は 164 バイトを達成した。これは [23] や [20] よりはるかに小さいものである上、速度的にもこれらの結果よりも優れている。

PRESENT の実装方法はいずれも基本的には、入力レジスタのデータを一ビットシフトし、そのキャリービットを出力レジスタに取り込むという簡単な処理の繰り返しである。RL78 の 16 ビット命令を使うことで、このキャリービットの移動が一命令でできることが小型化に貢献している。

PRESENT もテーブルの作り方、またそれを RAM に転送するかどうかでサイズと速度のトレードオフが存在する。表の一番右の欄の Sn-m は、16 バイトのテーブルを ROM に n 個持ち、そのうち m 個を RAM に転送する実装であることを意味している。また S8 はこの表 2 つを並列に参照する 256 バイトの表を 2 つもつ実装であることを示している。

暗号化における (1024,64) 実装は 1 段を完全にアンロールしているものであり、RL78 における速度の限界を示していると考えられる。このように PRESENT は速度は遅いものの、メモリサイズの小型化で優位性のあるアルゴリズムである。

表 3.16　RL78 での PRESENT の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed	Method
PRESENT (E)	ROM-Min	164	38 + 22	93412n	-	S1-0
PRESENT (E)	(512,64)	491	44 + 16	11344n	-	S2-1
PRESENT (E)	(512,128)	499	60 + 16	10560n	-	S2-2
PRESENT (E)	(1024,64)	952	28 + 10	9007n	-	S8
PRESENT (ED)	(512,64)	512	42 + 18	16924n	3736+19131n	S2-0
PRESENT (ED)	(512,128)	509	74 + 18	16407n	3643+18614n	S2-2
PRESENT (ED)	(1024,64)	989	38 + 18	12048n	1996+12367n	S4-0
PRESENT (ED)	(1024,128)	1003	102 + 18	10691n	1903+11010n	S4-4

■Piccolo

表 3.17 に、RL78 にて Piccolo を実装した結果を示す。表の右端の見方は PRESENT の場合と同様である。Piccolo は RAM メモリの使用が少なく実装できるため、すべてのカテゴリにおいて 64 バイトの RAM メモリがあれば十分である。

最小実装のサイズでは PRESENT に及ばないものの、全体的に Piccolo は高速に実装できるアルゴリズムと言える。

表 3.17　RL78 での Piccolo の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed	Method
Piccolo (E)	ROM-Min	275	24 + 18	$12220n$	-	S1-0
Piccolo (E)	(512,64)	498	52 + 8	$5779n$	-	S2-2
Piccolo (E)	(1024,64)	1018	40 + 8	$4961n$	-	S8
Piccolo (E)	Fast	1172	40 + 8	$4636n$	-	S8
Piccolo (ED)	(512,64)	512	54 + 8	$6186n$	$6084n$	S2-2
Piccolo (ED)	(1024,64)	966	52 + 8	$5779n$	$5779n$	S2-2

■TWINE

表 3.18 に、RL78 にて TWINE を実装した結果を示す。一番右の欄の見方は PRESENT、Piccolo と同様である。TWINE はソフトウエアでオーバーヘッドが少なく、きわめて小型化が可能なアルゴリズムである。また Piccolo と同じく RAM メモリの使用が少なく実装できるため、すべてのカテゴリにおいて 64 バイトの RAM メモリがあれば十分である。

暗号化のみの場合、ROM 512 バイト、RAM 64 バイトですでにほぼ最高速の実装が可能となっている。速度的には、TWINE は Piccolo とほぼ同程度を達成している。

表 3.18　RL78 での TWINE の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed	Method
TWINE (E)	ROM-Min	232	52 + 8	$11043n$	-	S1-0
TWINE (E)	(512,64)	468	52 + 6	$4957n$	-	S1-1
TWINE (ED)	(512,64)	510	54 + 10	$6132n$	$2463+5570n$	S1-1
TWINE (ED)	(1024,64)	972	54 + 6	$4957n$	$1727+4892n$	S1-1

■SIMON

表 3.19 に、RL78 にて SIMON を実装した結果を示す。小型実装は、共通部分のループ化やサブルーチン化を積極的に行って、できる限り ROM サイズを小さくしたものである。SIMON は Feistel 構造であるため、暗号化モジュールと復号モジュールを両方もつプログラムにおいては、データ暗号化部の暗号化／復号共用を行っている。表 3.19 の Category で One と表記する一段実装は、ラウンド関数 1 つの内部をアンロールし、これをラウンド回ループさせた実装を行ったものである。データ暗号化部と鍵スケジュール部の共用は行っていないが、暗号化モジュールと復号モジュールを両方もつプログラムにおいては、初期化と鍵スケジュール部はサブルーチン化して暗号化と復号で共用している。

高速実装は、複数ラウンドをまとめてアンロールし、これを必要回数ループさせる実装を行うこととともに、一切の共用化やサブルーチン化を抑制することにより、さらなる高速化を目指したものである。高速実装においてまとめてアンロールする最適なラウンド数は、m = (key size) / (word size) とするとき、SIMON の場合 LCM(2, m) である。これは、SIMON がデータ暗号化部は 2 ラウンド周期、鍵スケジュール部が m ラウンド周期であるためである。

表 3.19　RL78 での SIMON の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed
SIMON(32/64)(E)	ROM-Min	127	20 + 8	$3706n$	-
SIMON(32/64)(E)	One	171	20 + 6	$2480n$	-
SIMON(32/64)(E)	Fast	413	20 + 6	$1872n$	-
SIMON(64/96)(E)	ROM-Min	112	32 + 8	$7354n$	-
SIMON(64/96)(E)	One	243	32 + 6	$4598n$	-
SIMON(64/96)(E)	Fast	859	32 + 6	$3450n$	-
SIMON(64/128)(E)	ROM-Min	128	40 + 8	$9094n$	-
SIMON(64/128)(E)	One	303	40 + 6	$6404n$	-
SIMON(64/128)(E)	Fast	753	40 + 6	$4688n$	-
SIMON(128/128)(E)	ROM-Min	111	48 + 8	$21050n$	-
SIMON(128/128)(E)	One	415	48 + 6	$13148n$	-
SIMON(128/128)(E)	Fast	629	48 + 6	$10836n$	-
SIMON(32/64)(ED)	ROM-Min	273	20 + 14	$4227n$	$6586n$
SIMON(32/64)(ED)	One	310	30 + 10	$2777n$	$4473n$
SIMON(32/64)(ED)	Fast	1035	20 + 6	$1872n$	$3069n$
SIMON(64/96)(ED)	ROM-Min	244	32 + 14	$8035n$	$12063n$
SIMON(64/96)(ED)	One	436	32 + 10	$4985n$	$7559n$
SIMON(64/96)(ED)	Fast	1888	32 + 6	$3450n$	$5217n$
SIMON(64/128)(ED)	ROM-Min	277	40 + 14	$9807n$	$15408n$
SIMON(64/128)(ED)	One	546	40 + 10	$6809n$	$11057n$
SIMON(64/128)(ED)	Fast	1883	40 + 6	$4688n$	$7551n$
SIMON(128/128)(ED)	ROM-Min	203	48 + 14	$22147n$	$34005n$
SIMON(128/128)(ED)	One	506	48 + 10	$13767n$	$21023n$
SIMON(128/128)(ED)	Fast	1457	48 + 6	$10836n$	$16116n$

■SPECK

表 3.20 に、RL78 にて SPECK を実装した結果を示す。小型実装は、共通部分のループ化やサブルーチン化を積極的に行って、できる限り ROM サイズを小さくしたものであり、データ暗号化部と鍵スケジュール部の共用を行っている。一段実装は、ラウンド関数 1 つの内部をアンロールし、これをラウンド回ループさせた実装を行ったものである。データ暗号化部と鍵スケジュール部の共用は行っていないが、暗号化モジュールと復号モジュールを両方もつプログラムにおいては、初期化と鍵スケジュール部はサブルーチン化して暗号化と復号で共用している。高速実装は、複数ラウンドをまとめてアンロールし、これを必要回数ループさせる実装を行うとともに、一切の共用化やサブルーチン化を抑制することにより、さらなる高速化を目指したものである。高速実装においてまとめてアンロールする最適なラウンド数は、m = (key size) / (word size) とするとき、SPECK の場合 m-1 である。これは SPECK がデータ暗号化部に周期はなく、鍵スケジュール部が m-1 ラウンド周期であることから導き出される。

表 3.20　RL78 での SPECK の実装結果

Algorithm	Category	ROM	RAM static+stack	Enc Speed	Dec Speed
SPECK(32/64)(E)	ROM-Min	96	24 + 8	$1817n$	-
SPECK(32/64)(E)	One	115	20 + 6	$1249n$	-
SPECK(32/64)(E)	ROM-Min	261	20 + 6	$1006n$	-
SPECK(64/96)(E)	ROM-Min	90	44 + 8	$6645n$	-
SPECK(64/96)(E)	One	185	32 + 6	$2335n$	-
SPECK(64/96)(E)	Fast	308	32 + 6	$2062n$	-
SPECK(64/128)(E)	ROM-Min	89	52 + 8	$7448n$	-
SPECK(64/128)(E)	One	205	40 + 6	$2644n$	-
SPECK(64/128)(E)	Fast	451	40 + 6	$2122n$	-
SPECK(128/128)(E)	ROM-Min	71	67 + 8	$11432n$	-
SPECK(128/128)(E)	One	205	64 + 6	$5662n$	-
SPECK(128/128)(E)	Fast	309	48 + 6	$4793n$	-
SPECK(32/64)(ED)	ROM-Min	211	24 + 10	$2308n$	$3684n$
SPECK(32/64)(ED)	One	283	20 + 6	$1249n$	$1918n$
SPECK(32/64)(ED)	Fast	623	20 + 6	$1006n$	$1392n$
SPECK(64/96)(ED)	ROM-Min	211	44 + 10	$6600n$	$10837n$
SPECK(64/96)(ED)	One	447	32 + 6	$2335n$	$3585n$
SPECK(64/96)(ED)	Fast	742	32 + 6	$2062n$	$3088n$
SPECK(64/128)(ED)	ROM-Min	210	52 + 10	$7078n$	$11690n$
SPECK(64/128)(ED)	One	499	40 + 6	$2644n$	$4152n$
SPECK(64/128)(ED)	Fast	1087	40 + 6	$2122n$	$3165n$
SPECK(128/128)(ED)	ROM-Min	157	67 + 10	$11471n$	$18074n$
SPECK(128/128)(ED)	One	391	64 + 10	$5702n$	$8726n$
SPECK(128/128)(ED)	Fast	746	48 + 6	$4793n$	$7343n$

■Midori

表 3.21 に、RL78 にて Midori を実装した結果を示す。Midori64 については 4 ビットの S-box で構成されるため、実装方法は PRESENT などと同様の方針をとる。したがって、一番右の欄の見方は PRESENT、Piccolo、TWINE と同様である。Midori128 については速度優先の実装については 8 ビット S-box のテーブル実装とループ展開を行う。また、小型実装においては、Midori128 における 8 ビット S-box の 4 ビット S-box から計算処理による作成、関数共通化と関数呼び出しの多用、ループ処理化を行っている。

表 3.21　RL78 での Midori の実装結果

Algorithm		Category	ROM	RAM static+stack	Enc Speed	Dec Speed	Method
Midori64	(E)	Fast	871	64 + 8	6768n	-	S2-0
Midori64	(E)	ROM-Min	232	96 + 8	16979n	-	S2-0
Midori64	(ED)	Fast	1576	64 + 10	6768n	8360n	S2-0
Midori64	(ED)	ROM-Min	374	96 + 6	17867n	27966n	S2-0
Midori128	(E)	Fast	1346	64 + 8	9217n	-	-
Midori128	(E)	ROM-Min	560	64 + 8	31794n	-	-
Midori128	(ED)	Fast	1745	64 + 10	9217n	10166n	-
Midori128	(ED)	ROM-Min	605	64 + 6	32495n	45586n	-

3.1.2.2　性能比較

これまでの実装結果をもとに、評価対象アルゴリズムをいくつかの軸で比較する。

■メモリサイズを限定した実装（暗号化のみ）

図 3.27 は ROM サイズ 1024 バイト以下、RAM サイズ 128 バイト以下の条件で、暗号化のみを実装した場合の速度性能の比較である。図 3.28 は、見やすさのため図 3.27 から TDES を除いたものである。この程度のメモリリソースがある場合には、AES が最も高速となり、SPECK がそれに続くとの結果を得た。

図 3.29 は ROM サイズ 1024 バイト以下、RAM サイズ 64 バイト以下の条件で、暗号化のみを実装した場合の速度性能の比較である。図 3.30 は、見やすさのため図 3.29 から TDES を除いたものである。CLEFIA を除けば ROM サイズ 1024 バイト以下、RAM サイズ 128 バイト以下の場合と同様であるが、CLEFIA だけは RAM64 バイトでの実装が不可能である。これを図では値 0 として示している。

図 3.31 は ROM サイズ 512 バイト以下、RAM サイズ 128 バイト以下の条件で、暗号化のみを実装した場合の速度性能の比較、図 3.32 は ROM サイズ 512 バイト以下、RAM サイズ 64 バイト以下の条件で暗号化のみ実装した場合の速度性能の比較である。ROM サイズが 512 バイト以下になると CLEFIA 以外にも Camellia や TDES も実装が不可能となる。その他のアルゴリズムでは AES、SPECK が依然高速である。

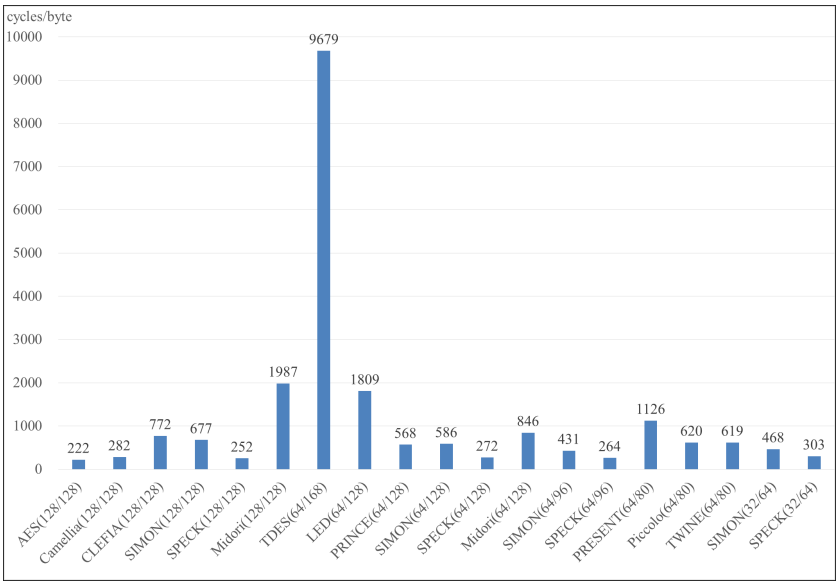

図 3.27　ROM 1024 バイト、RAM 128 バイトでの速度性能

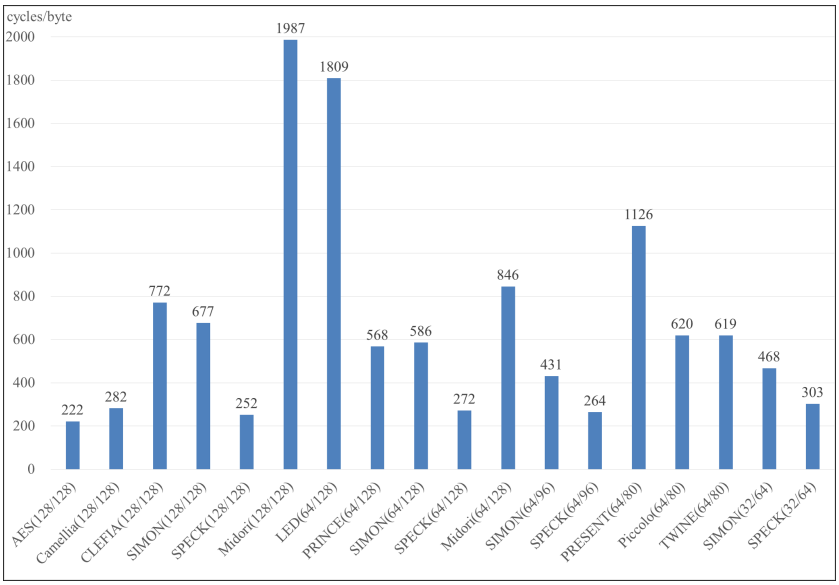

図 3.28　ROM 1024 バイト、RAM 128 バイトでの速度性能（TDES を除いた図）

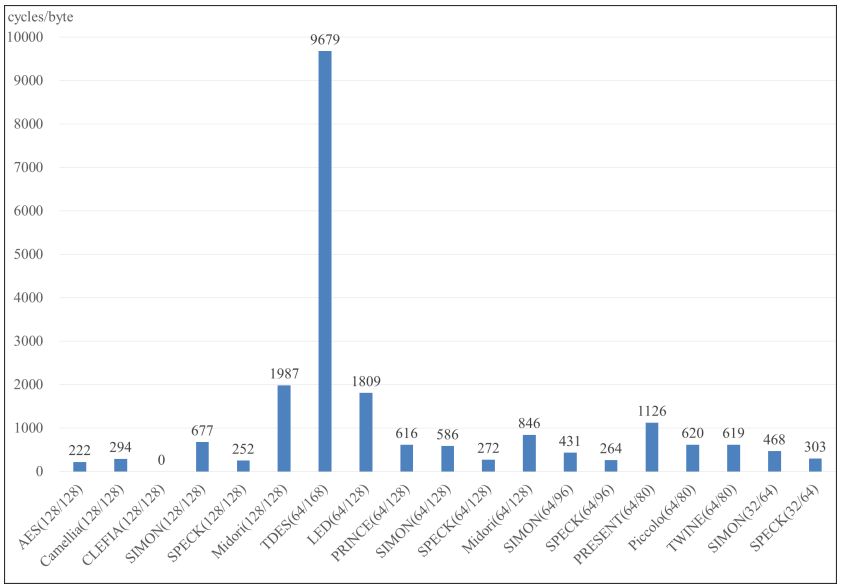

図 3.29　ROM 1024 バイト、RAM 64 バイトでの速度性能

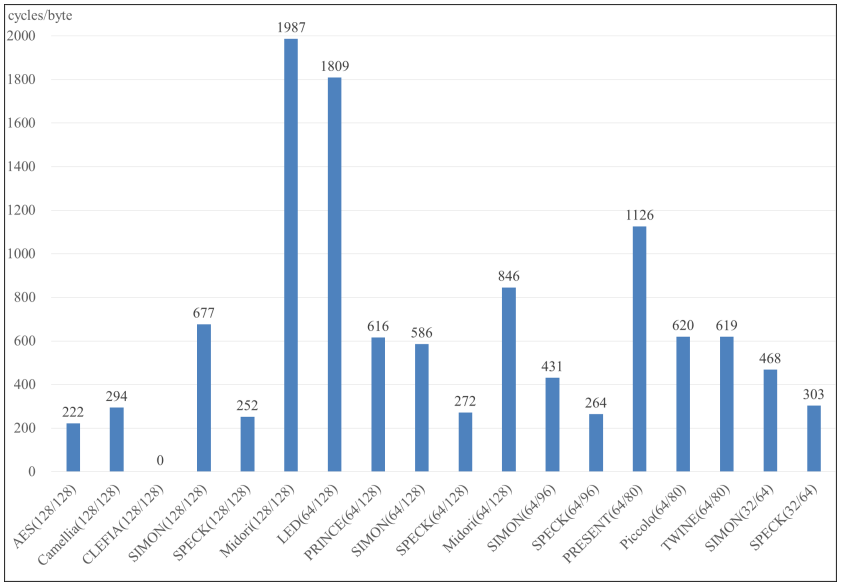

図 3.30　ROM 1024 バイト、RAM 64 バイトでの速度性能（TDES を除いた図）

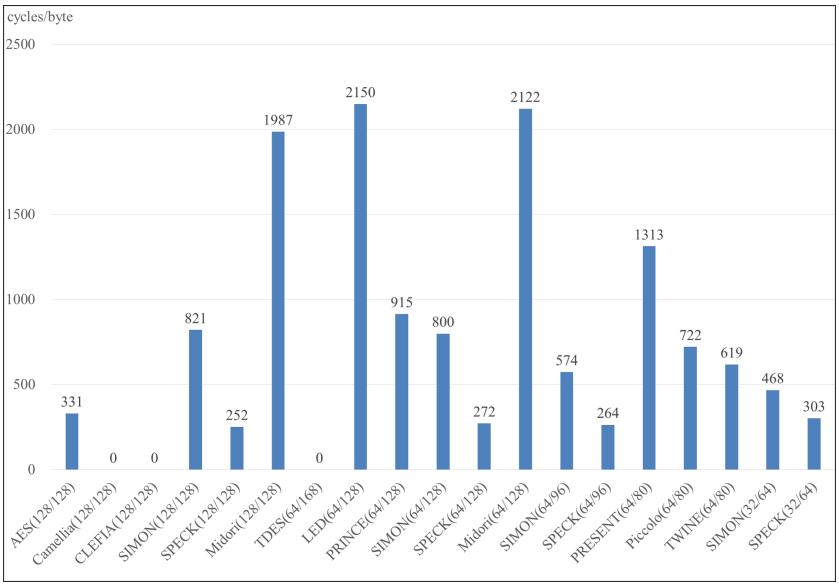

図 3.31　ROM 512 バイト、RAM 128 バイトでの速度性能

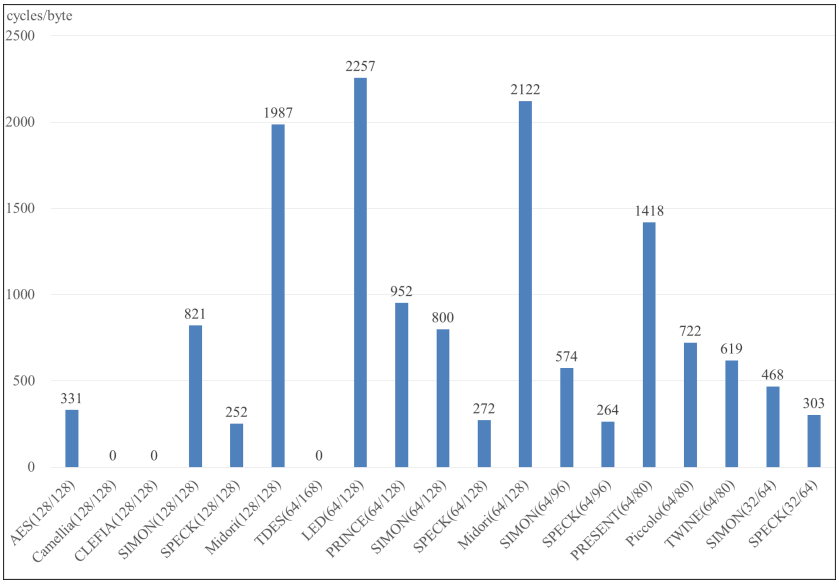

図 3.32　ROM 512 バイト、RAM 64 バイトでの速度性能

■メモリサイズを限定した実装（暗号化・復号）

図 3.33 は ROM サイズ 1024 バイト以下、RAM サイズ 128 バイト以下の条件で、暗号化・復号両方を実装した場合の速度性能の比較である。図 3.34 は、見やすさのため図 3.33 から TDES を除いたものである。ここでも SPECK が最高速であるが、暗号化のみの実装の場合と比べて Piccolo、PRINCE、TWINE との差は縮まっている。

図 3.35 は ROM サイズ 1024 バイト以下、RAM サイズ 64 バイト以下の条件で、暗号化・復号両方を実装した場合の速度性能の比較である。図 3.36 は、見やすさのため図 3.35 から TDES を除いたものである。このカテゴリでは CLEFIA と Camellia、Midori128 が実装不可能となった。また、AES の速度と Piccolo、PRINCE、TWINE との差があまりなくなっている。これは AES の速度がメモリリソース不足のため低下していることを示している。ここでも SPECK が最高速である。

図 3.37 は ROM サイズ 512 バイト以下、RAM サイズ 128 バイト以下の条件で暗号化・復号両方を実装した場合の速度性能の比較、図 3.38 は ROM サイズ 512 バイト以下、RAM サイズ 64 バイト以下の条件で暗号化・復号両方を実装した場合の速度性能の比較である。ここでは AES も実装不可能となり、結果的に 5 つのアルゴリズムだけが生き残るという結果となった。なかでも SPECK は ROM サイズの制限によってほとんど影響を受けない高速性能を示している。

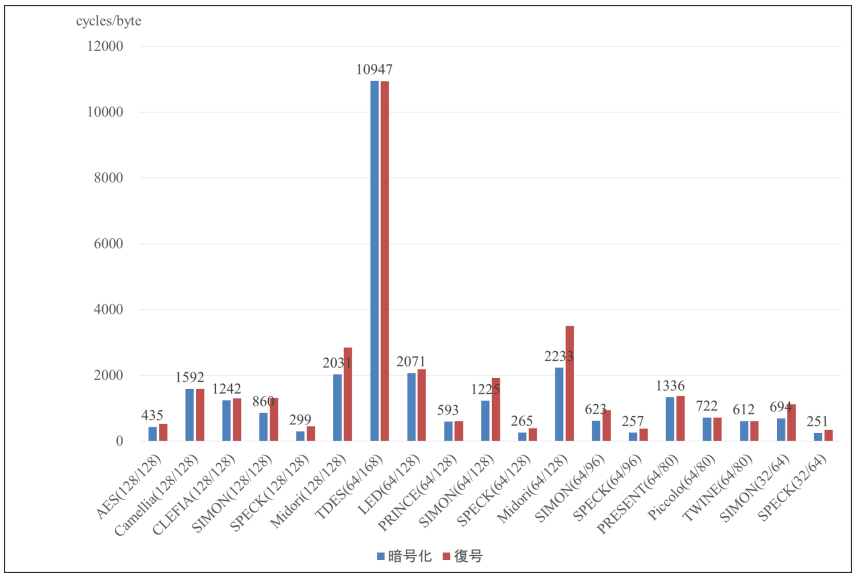

図 3.33　ROM 1024 バイト、RAM 128 バイトでの速度性能

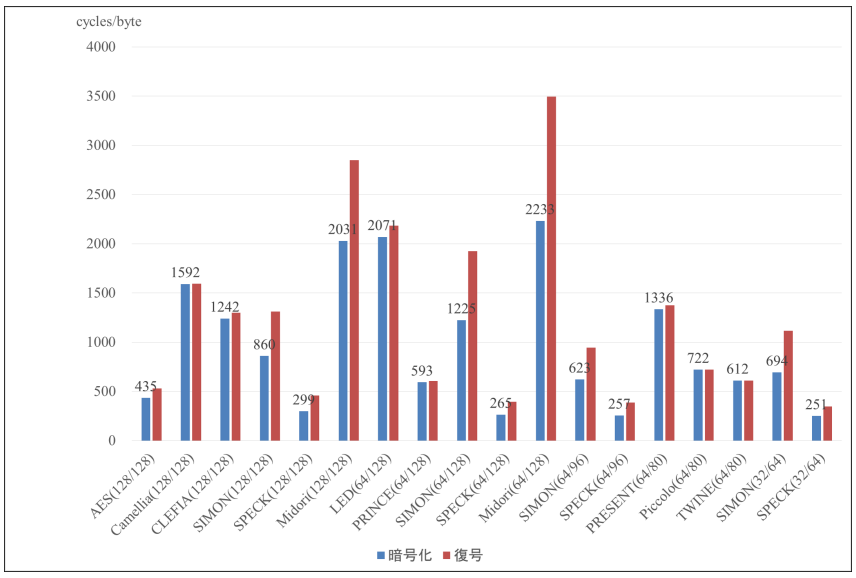

図 3.34　ROM 1024 バイト、RAM 128 バイトでの速度性能（TDES を除いた図）

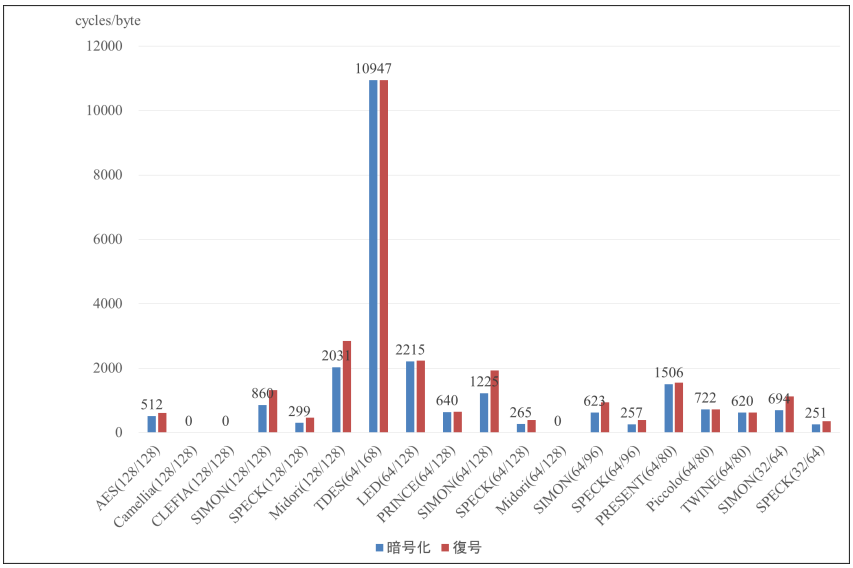

図 3.35　ROM 1024 バイト、RAM 64 バイトでの速度性能

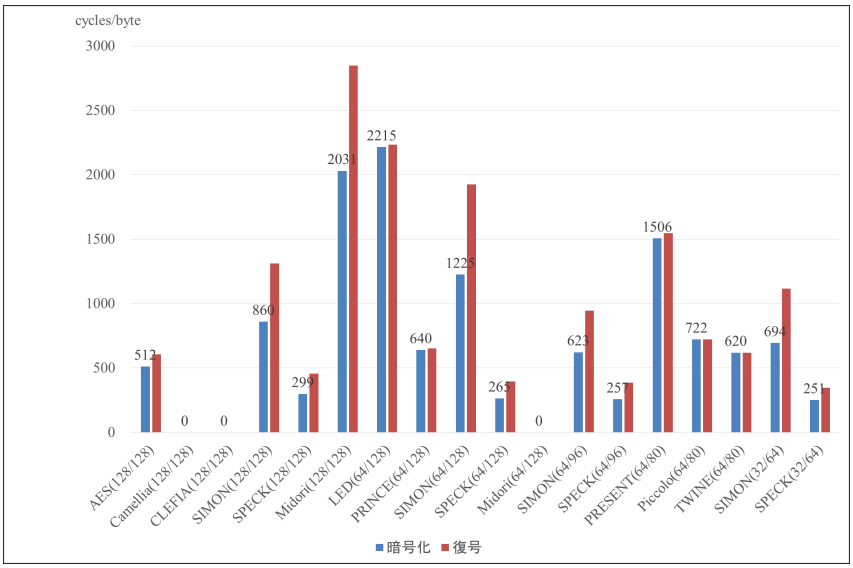

図 3.36　ROM 1024 バイト、RAM 64 バイトでの速度性能（TDES を除いた図）

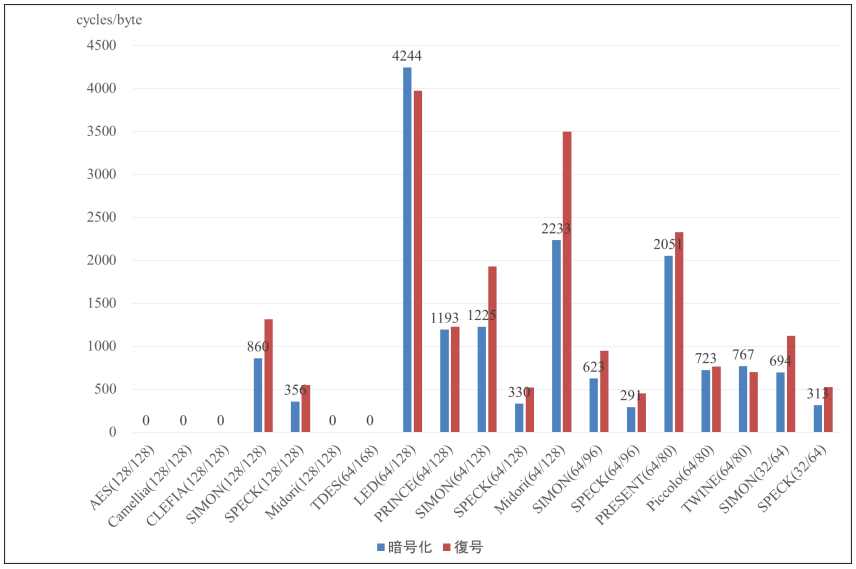

図 3.37　ROM 512 バイト、RAM 128 バイトでの速度性能

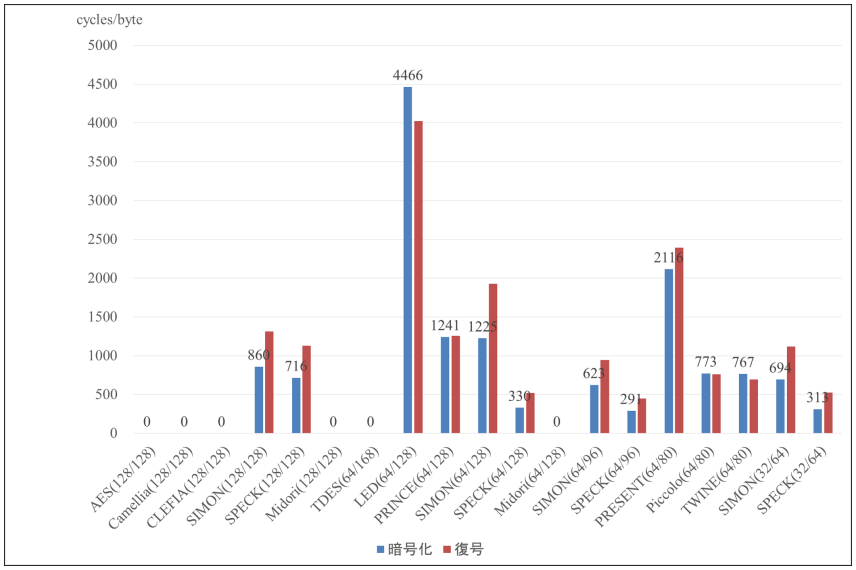

図 3.38　ROM 512 バイト、RAM 64 バイトでの速度性能

■メモリサイズを限定した実装（まとめ）

以上の結果をひとつの図にまとめたものを図 3.39 ならびに図 3.40 に示す。後者は前者から TDES を除いたものである。一般にメモリサイズの制約が厳しくなる右側に行けばいくほど性能は低下する。この性能低下があまり見られない SIMON、SPECK、Piccolo 及び TWINE はメモリサイズにまだ余裕があることを示している。

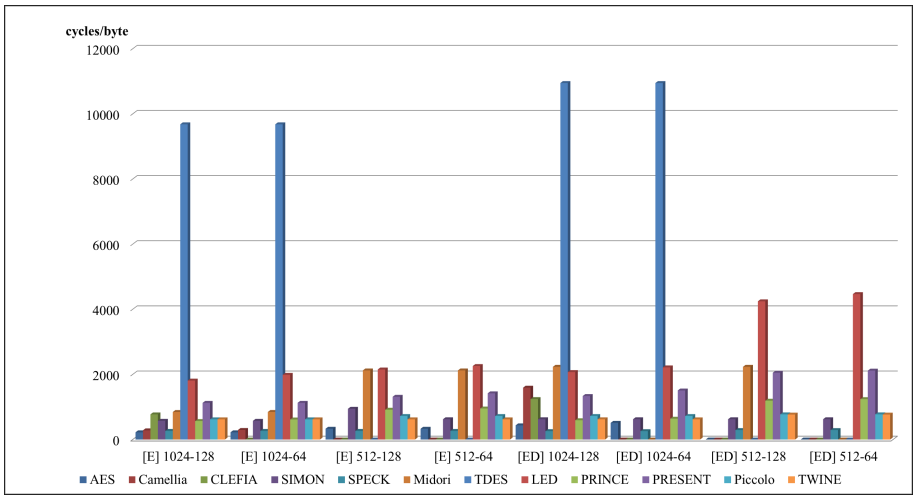

図 3.39　メモリサイズ限定速度性能一覧

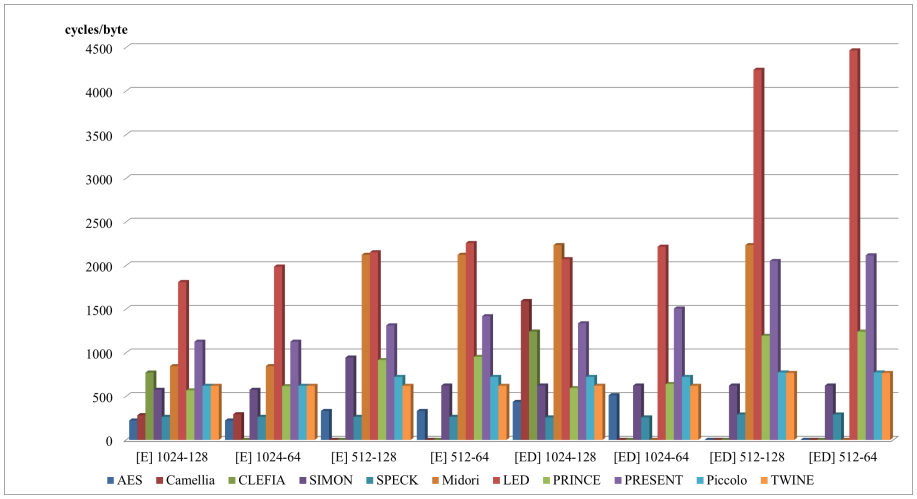

図 3.40　メモリサイズ限定速度性能一覧（TDES を除いた図）

■高速実装

図 3.41 は ROM 2KB 程度で高速実装を目指した場合の速度性能の比較である。これは RL78 プロセッサで達成できる各アルゴリズムの最高性能に近い値と考えられる。この評価では AES、Camellia、SPECK がほぼ同等の性能となることが知られる。

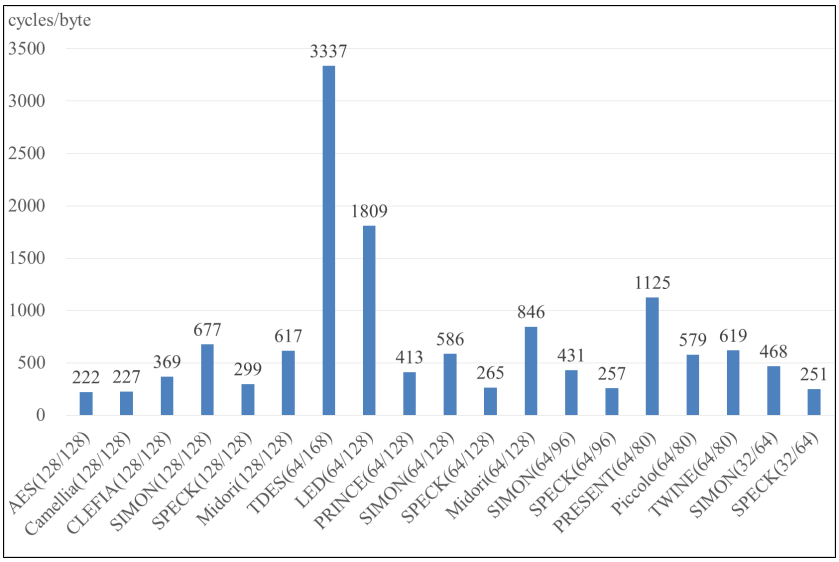

図 3.41　高速実装速度性能一覧

■最小実装

図 3.42 は ROM サイズを最小にする実装（暗号化のみ）のサイズがどこまで小さくなるかを評価したものである。ここでは、最近の軽量ブロック暗号アルゴリズムと、それ以外のものの差がはっきりあらわれている。

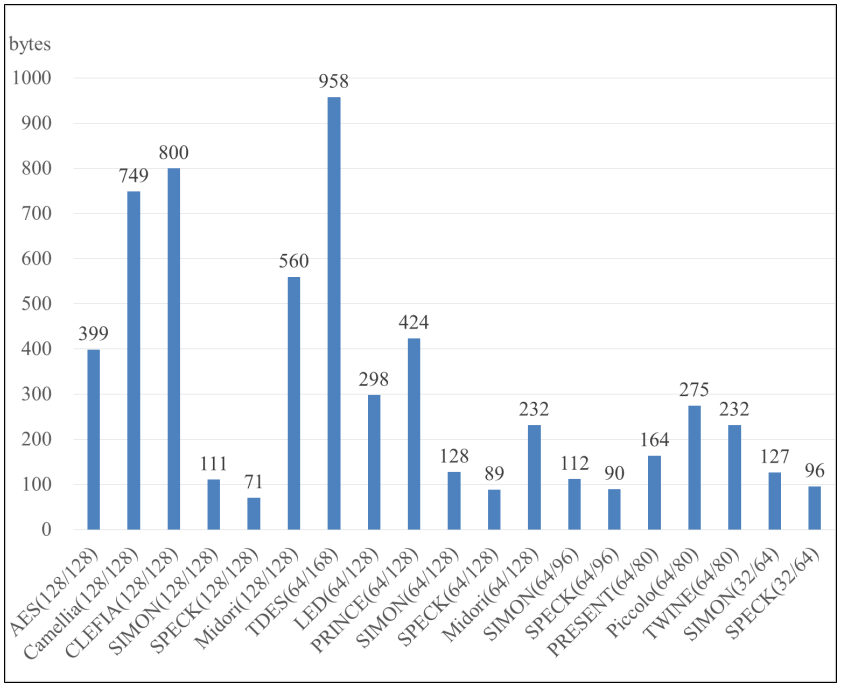

図 3.42　最小実装速度性能一覧

■その他の考察

図 3.43 は今回暗号化のみを実装したもののうち、ROM サイズが 512 バイト以下になるものについてすべてをプロットしたものである。ここで横軸は ROM サイズ、縦軸は速度を示している。AES については S-box テーブルをもつ限り ROM サイズ 400 バイト前後が限界であり、それ以下の領域は、より小さい S-box をもつアルゴリズムあるいは S-box をもたないアルゴリズムで可能となる。またこの図が示すように、ROM が 200 バイト以下で 2000 サイクル/バイト以下は、SIMON, SPECK だけが達成できる域であり、今後のソフトウエア軽量暗号が目指すひとつの方向性を示していると思われる。

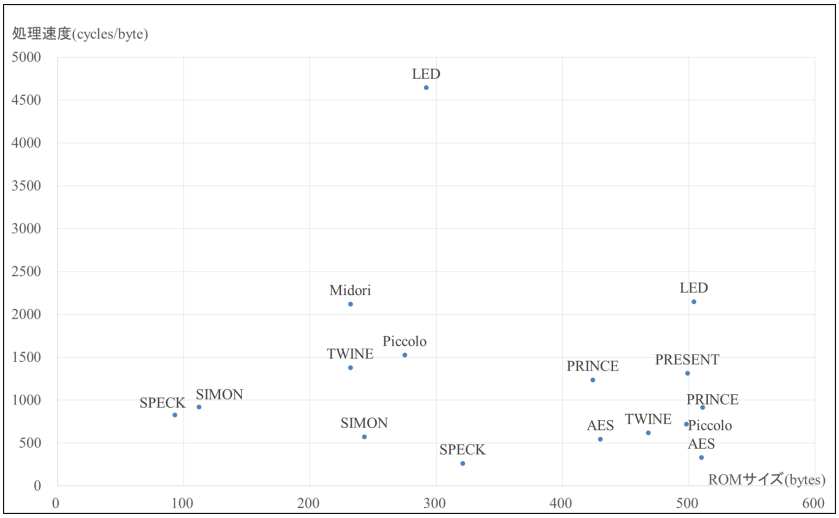

図 3.43　サイズと速度のトレードオフ

一方、今回の実装をアルゴリズム構造の観点から見ると、小型化するには、アルゴリズム全体が数少ない単純な繰り返し構造をもっている必要がある。小型化実装を行う場合、単なるデータの移動や定数も無視できないオーバーヘッドとなる。また RAM サイズの制約がある場合 on-the-fly で鍵スケジュールを行う必要があるが、アルゴリズムによってはこの鍵スケジュールが小型化のボトルネックになることも少なくない。

さらにプロセッサ構造の観点から見ると、回転シフト命令の効率はプロセッサに大きく依存すること、また最近のプロセッサはほとんどが little endian なので big endian を前提としたアルゴリズムはバイト順の変更に伴うオーバーヘッドが発生する可能性があることなどは、軽量暗号設計において留意すべき点であると言える。

3.1.2.3 評価方法の概要

本節では、軽量ブロック暗号のソフトウェア実装性能評価において採用した評価環境と、実装条件について述べる。

■組み込みマイコン RL78 と評価環境

本評価では、ルネサスエレクトロニクス社製の組み込みマイコン RL78 上で評価対象ブロック暗号の実装を行った [6]。RL78 はアキュムレータベースの 16 ビット CISC プロセッサである。その命令セットには多くの 1 バイト命令が存在し、また Load-Modify 命令がサポートされているなど、ROM サイズの小型化に適したプロセッサであると言える。一方 RL78 はすべての命令で 16 データを扱えるわけではなく、例えばブロック暗号で頻繁に必要となる論理演算や回転シフト演算は 8 ビット命令だけがサポートされている。

RL78 にはいくつかのシリーズが存在するが、たとえばそのうち汎用である G1x シリーズのローエンド製品は ROM サイズが 1KB、RAM サイズは 128B である。また、車載用に開発された F1x シリーズのローエンド製品は ROM サイズ 8KB、RAM サイズ 512B を搭載している。

RL78 の命令セットはシリーズ共通であり（ただし乗算命令のサポートの有無は品種により異なる）、その命令長も同じである。したがって乗算命令以外の汎用命令を使う限り、RL78 のコードはすべての品種で動作し、またそのコードが占有するメモリサイズは同じである。実行速度については一部採用するハードウエアコア（S1、S2、S3 の 3 種類が存在する [6]）によって若干異なるが、ここではほとんどの品種で利用されている S2 コアでの速度を計測した。

なお開発環境はルネサスエレクトロニクス社製の CubeSuite+ を用いた。

■実装条件

さまざまなメモリリソース環境での性能を評価するため、ROM サイズは 512B と 1024B の 2 種類、RAM サイズは 64B と 128B の 2 種類の合計 4 通りのメモリ制約条件のもので、暗号化機能だけを実装した場合と、暗号化機能と復号機能の両方を実装した場合の速度性能を調べた。さらにこれに加え、速度性能は考慮せずに ROM サイズを最小化する実装や、逆に ROM サイズを 2KB 程度まで許した場合に速度がどこまで向上するかなどの観点でも評価を行った。

評価対象とするブロック暗号アルゴリズムによっては、特定のメモリ制約条件ではそもそも実装が不可能なものや、逆に少ないリソースで充分な性能が出ており、それ以上のメモリ容量が与えられてもそれ以上の性能向上が認められないものも存在する。そのような場合は個別の実装を省略した。なお RAM サイズが最大 128B での評価であるので、結果的にすべての評価対象アルゴリズムで、鍵スケジュールは on-the-fly 実装になっている。

本ライブラリのプログラムインターフェースや ROM、RAM サイズの計算方法は文献 [20] と同様である。すなわちプログラムはアセンブリ言語で記述し、1 ブロックのデータを暗号化および復号する機能をもつ。C 言語から呼び出し可能なサブルーチン形式とし、このサブルーチンは引数を一つだけ取り、その引数が指すアドレスに、平文、暗号文、鍵、一時利用データを格納する。また RAM メモリ最小化ならびに実用的な観点から、以下の条件での実装を行っている。

- 平文と暗号文の領域は共用する。
- 鍵の領域はサブルーチン呼び出し時と終了時で同じ内容とする（ただし実行中に一時的に変更してもよい）。
- システムが主に利用するゼロページ領域（絶対アドレス 256 未満の領域）は利用しない。
- リロケータブルなプログラムとする（絶対アドレスをハードコーディングしない）。

- システムに依存するコーディングは行わない（例えばレジスタバンクの切り替えをしない、特殊レジスタを直接操作することはしないなど)。

ROM、RAM サイズの計算には、このサブルーチンを実行するために必要なすべてのリソースを含めている。具体的には、ROM サイズはコードならびに固定データテーブルを含み、RAM サイズは、平文（暗号文と領域を共用する)・鍵・一時データ・スタックをすべて含んでいる。したがって例えば 128 ビットブロック、128 ビット鍵のブロック暗号においては、平文と鍵の領域だけで 32 バイトを占有する。また関数呼び出し時に必要となるスタックフレームは 6 バイト（call 命令 4 バイト＋ callee save レジスタの保存 2 バイト）であるので、例えば RAM サイズ 64 バイト以下で実装する場合、自由に使える RAM はスタックを含めて 26 バイトしか存在せず、かなり厳しい制約となる。

3.2 認証暗号

3.2.1 ソフトウェア実装評価

本章では、認証暗号を、組み込みマイコン上の限定されたメモリリソースのもとで、ソフトウエア実装した場合の速度性能について計測、比較した結果を示す。評価対象とした認証暗号を図 3.44 に示す。

	セキュリティレベル	鍵長	ブロック長	ナンス長	タグ長
CLOC-TWINE	32	80	64	48	32
SILC-PRESENT	32	80	64	48	32
JAMBU-SIMON	48	96	96	48	48
CLOC-AES	64	128	128	96	64
SILC-AES	64	128	128	96	64
AES-OTR	64	128	128	96	128
AES-OCB	64	128	128	96	128
JAMBU-AES	64	128	128	96	128
AES-GCM	64	128	128	96	128
ACORN	128	128	128	128	128
Minalpher	128	128	256	128	128
Ketje-SR	128	128	32	128	128
Ascon	128	128	128	128	128

図 3.44　評価対象暗号パラメータ比較

なお、ハードウェア実装については、ジョージメイソン大学の研究チームによって同一プラットフォームによる評価が行われているため、本ガイドラインでは扱わない。詳細は以下の URL を参照されたい。

Authenticated Encryption FPGA Ranking

`https://cryptography.gmu.edu/athenadb/fpga_auth_cipher/rankings_view`

3.2.1.1 性能比較

■認証暗号の実装評価結果

図 3.45 に認証暗号を小型実装した場合の評価結果を示す。上の表は左から、アルゴリズム名、セキュリティレベル (bit)、ROM サイズ (bytes)、RAM サイズ (bytes)、関連データ処理に関する各関数の処理速度 (赤字が cycles/byte、黒字は cycles)、内部で利用されている Core Function 名、Core Function の ROM サイズ (bytes)、Core Functions の速度 (cycles/byte) である。下の表は暗号化処理ならびに復号処理に関する各関数の処理速度を表している。

なお、CLOC-TWINE における ENC_NULL、DEC_NULL の速度はそれぞれ 11160、11099 cycles、CLOC-AES における ENC_NULL、 DEC_NULL の速度はそれぞれ 8895、 8790 cycles である。

図 3.46 に認証暗号を高速実装した場合の評価結果を示す。図の見方は図 3.45 と同様である。

なお、CLOC-TWINE における ENC_NULL、DEC_NULL の速度はそれぞれ 5074、5013 cycles、CLOC-AES における ENC_NULL、 DEC_NULL の速度はそれぞれ 3748、 3643 cycles である。

以上は暗号化（とタグ生成）モジュールならびに復号（とタグ検証）モジュールをともに含むプログラムの評価結果であったが、ここから復号部の関数群（DEC_0, DEC_1, DEC_2, DEC_3, DEC_4）を取り除いて暗号化（とタグ生成）機能だけにした場合は、その他の関数の性能は変わらず、ROM サイズだけが以下にあげるバイト数だけ減少

	Sec Level	Size		Associate Data Processing					Core Function		
		ROM	RAM	AD_0	AD_1	AD_2	AD_3	AD_4		ROM	Speed
CLOC-TWINE	32	811	108	-	11199	1409		365	TWINE	234	1380
SILC-PRESENT		537	98	93413	11706			312	PRESENT	164	11677
		600	96	38357	4796			312	PRESENT	227	4768
JAMBU-SIMON	48	522	90	12407	1050			12602	Simon (96/96)	109	1030
CLOC-AES	64	963	150	-	8966	570		637	AES	399	544
SILC-AES		772	144	9071	569			544			
AES-OTR		1202	178	142/9638	594		10134	-			
JAMBU-AES		803	128	8885	563		9010	-			
AES-OCB		1705	224	9252	564		8921		AES	1020	512/607
AES-GCM		760	172	8943	2557			-	Mul128	64	2534
ACORN	128	589	129	109075	489			15522	State	327	478
Minalpher		665	193	607/41361	1276		41220	-	P	295	1241
Ketje-SR		724	114	46827	988		3955	-	F	385	969
Ascon		617	132	40332	2491		19919	11	p^6	299	2475

	Plaintext Data Processing					Ciphertext Data Processing				
	ENC_0	ENC_1	ENC_2	ENC_3	ENC_4	DEC_0	DEC_1	DEC_2	DEC_3	DEC_4
CLOC-TWINE	11160	22365	2799	22593	-	11160	22365	2799	22532	64
SILC-PRESENT	93816	187276	23424		93786	93816	187276	23424		93778
	38343	76730	9606		38513	38543	76730	9606		38505
JAMBU-SIMON	-	1057		12619	25048	-	1057		12633	25099
CLOC-AES	9373	18211	1151	18096	-	9373	18211	1151	17991	108
SILC-AES	9443	18210	1151		9353	9443	18210	1151		9345
AES-OTR	9708	616		19839	10166	9708	609		19841	10158
JAMBU-AES	-	573		9035	17952	-	573		9042	17960
AES-OCB	9736	585		8804	8861	9736	680		8804	8885
AES-GCM	182	3124			49823	182	3124			50008
ACORN		490			62053		493			62085
Minalpher	40729	2560		81942	-	40729	2549		81525	196
Ketje-SR	-	992		23298	11724	-	993		23302	11747
Ascon		2503		206	40220		2517		300	40173

図 3.45　認証暗号の実装結果（小型実装）

する。このサイズは Core Function の種類や小型版、高速版に依存せず一定である（そのように設計されている）。

表 3.22　認証暗号の暗号化モジュールのみのプログラムの ROM 削減バイト数

アルゴリズム名	AES-GCM	CLOC	SILC	AES-OTR	Ketje	Minalpher
削減されるバイト数	47	102	59	255	108	93

ここで Core Function で利用されるブロック暗号の暗号・復号機能は、認証暗号としての暗号・復号機能とは異なることに注意されたい。今回実装評価対象とした認証暗号のうち、Minalpher と AES-OCB 以外はすべて、認証暗号としての復号機能においても Core Function の逆関数（Core Function がブロック暗号の場合はブロック暗号としての復号機能）は必要ではない。また、Minalpher の Core Function は、その逆関数とは厳密には異なるが、Involution 構造をもつので非常に類似した構造をもっている。

また、本評価におけるインターフェースの作り方から、認証暗号としての暗号化部の関数群と復号部の関数群は明確に分けられているので、暗号化（とタグ生成）機能だけをもつ認証暗号モジュールのサイズは、単純に復号部の関数群のサイズを全体から引くだけで計算でき、速度性能については変わることはない。

図 3.47 に小型実装を行った場合の ROM サイズを比較したグラフを示す。図 3.48 はこの ROM サイズを、Core Function の ROM サイズ（下部）と Mode に相当する、それ以外の部分の ROM サイズ（上部）に分割したものである。また、図 3.49 に高速実装を行った場合の速度（正確には平文サイズが十分大きい時の漸近的速度）を比較したグラフを示す。図 3.50 はこの速度を、Core Function の速度（下部）と Mode に相当する、それ以外の部分の速

	Sec Level	Size		Associate Data Processing					Core Function		
		ROM	RAM	AD_0	AD_1	AD_2	AD_3	AD_4		ROM	Speed
CLOC-TWINE	32	1049	106	-	5113	649		365	TWINE	470	620
SILC-PRESENT		896	118	10777	1349			262	PRESENT	499	1320
JAMBU-SIMON	48	1709	80	6342	531		6435	-	Simon (96/96)	477	527
CLOC-AES	64	1521	128	-	3819	248		637	AES	928	222
SILC-AES		1369	124	3924	248			438			
AES-OTR		1958	156	142/4081	239		4163	-			
JAMBU-AES		2466	102	3589	227		3710	-			
AES-OCB		2962	216	3809	238		3837		AES	2007	222/391
AES-GCM		1489	150	3669	1034			-	Mul128	239	1011
ACORN	128	871	116	55296	246			7752	State	446	237
Minalpher		1926	169	49/9580	308		9898	-	P	1455	289
		1977	425	866/8669	254		8170	-	P	1455	235
Ketje-SR		1482	114	15699	340		1361	-	f	927	321
Ascon		1966	116	11173	696		5639	11	p^6	1015	691

	Plaintext Data Processing					Ciphertext Data Processing				
	ENC_0	ENC_1	ENC_2	ENC_3	ENC_4	DEC_0	DEC_1	DEC_2	DEC_3	DEC_4
CLOC-TWINE	5074	10193	1278	10421	-	5074	10193	1278	10307	64
SILC-PRESENT	10657	21485	2689		10883	10657	21485	2689		10875
JAMBU-SIMON	-	532		6455	12699	-	532		6460	12767
CLOC-AES	3728	7585	476	7971	-	3728	7585	476	7866	108
SILC-AES	3694	7751	487		4100	3694	7751	487		4092
AES-OTR	4151	245		8140	4147	4151	243		8142	4139
JAMBU-AES	-	228		3738	7203	-	228		3745	7293
AES-OCB	4923	243		3893	3737	4923	412		3893	3888
AES-GCM	173	1271			20172	173	1271			20357
ACORN		247			31034		250			31082
Minalpher	9713	616		19739	-	9713	602		19554	196
	7985	508		16283	-	7985	494		16098	196
Ketje-SR	-	343		7734	3942	-	344		7738	3965
Ascon		698		188	11184		699		273	11312

図 3.46　認証暗号の実装結果（高速実装）

度（上部）に分割したものである。

なお、これらのグラフでは、いずれも認証暗号アルゴリズムのセキュリティレベルを（32 ビット、 64 ビット、128 ビット）色で表現している。

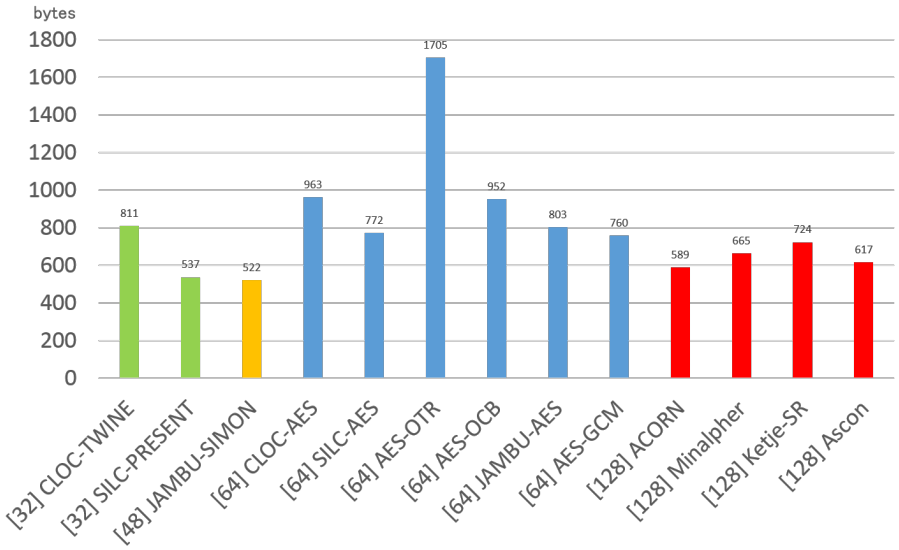

図 3.47　小型実装における ROM サイズ比較

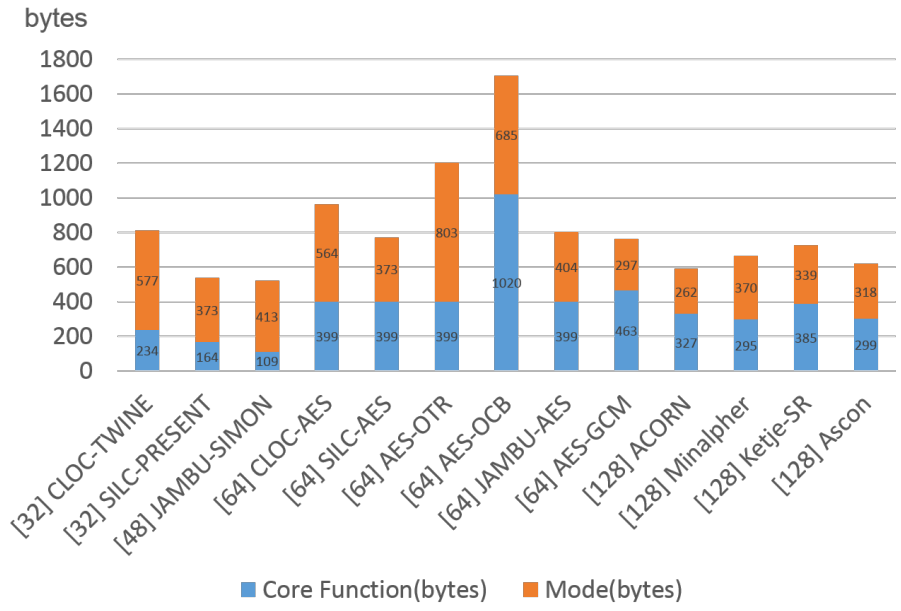

図 3.48　小型実装における ROM サイズ比較（Core Function と Mode に分割）

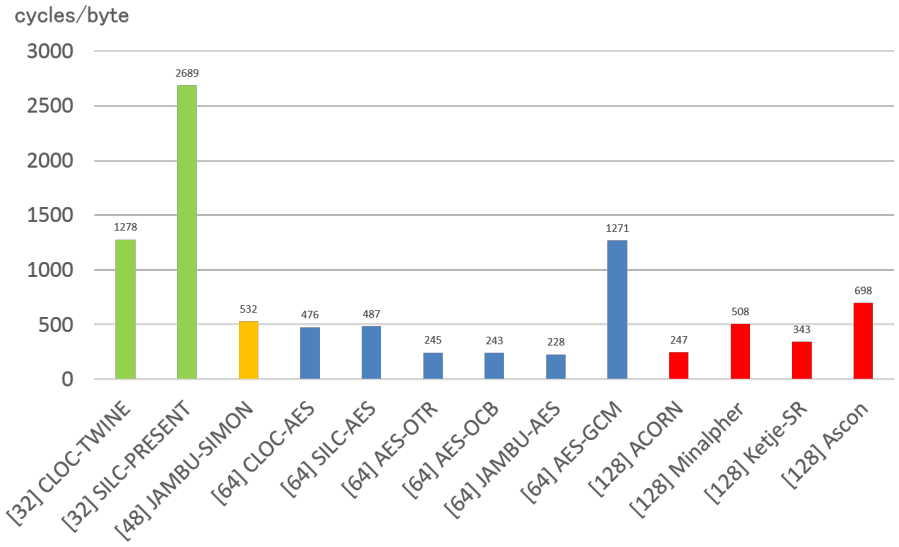

図 3.49　高速実装における暗号化速度（漸近速度）比較

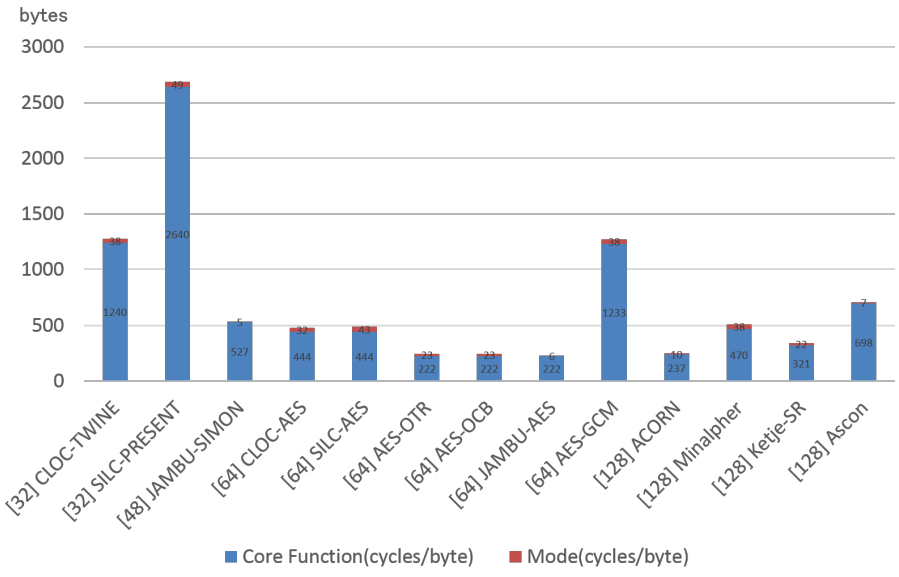

図 3.50　高速実装における暗号化速度（漸近速度）比較（Core Function と Mode に分割）

3.2.1.2 評価方法の概要

■コーディングの方針とインターフェース仕様

認証暗号は平文、暗号文、鍵に加え、関連データ (Associate Data)、ナンス (Nonce) 等、入出力パラメータが多く、このため軽量暗号のように速度性能を一次元的に評価することは困難である。そこで本評価では、文献 [21] で示されたコーディングの方針に沿いつつ、各々の認証暗号アルゴリズムをいくつかの部分に分割し、その分割単位ごとに速度性能を評価する方針とした。これにより平文や関連データの長さが与えられれば、本報告で作成された表から、誰もが実際の計算にかかるサイクル数を自分で計算できる。

一方でこのようにアルゴリズムを細かく分割すると、認証暗号全体の処理を行うためには全体のフローをコントロールする上位プログラムが必要であり、そのプログラムのオーバーヘッドが大きくなると、分割単位ごとの性能データを集積しても全体の速度を正しく見積もることができなくなる。

そこで、ここでは上位プログラムのオーバーヘッドができるだけ少なくなるような、ブロック単位での分割の方法を提案する。具体的には認証暗号のアルゴリズムを、関連データ処理部（以後 AD)、暗号化部（以後 ENC)、復号部（以後 DEC）に分割し、この 3 つの部分それぞれについて、さらに次の 5 つの機能に分割を行った。

1 初期計算（関連データや平文、暗号文を入力する前の処理）
2 第 1 ブロック計算
3 中間ブロック計算（第 2 ブロックから最終ブロックの前までの各 1 ブロックの処理）
4 最終ブロック計算
5 終了計算（関連データや平文、暗号文を入力し終わった後の処理）

これら各機能をひとつの関数としてコーディングしたが、実際には不必要な関数や同じ機能を持つ複数の関数があるため、すべての認証暗号アルゴリズムについて合計 15 個の関数を別々に実装する必要はない。また、一つの機能と別の機能との境界は一意的ではないが、アルゴリズムごとに、もっとも自然と考えられる境界を設定した。

以降これらの関数を次のように記述する。

AD_0　関連データ処理部の初期計算関数
ENC_123　暗号化部の第 1 、中間、最終ブロック計算関数 (これは ENC_1、ENC_2、ENC_3 が共通化できることを意味する)

この方法で記述した関数群を用いて認証暗号全体を記述した上位プログラムを、AES-GCM の暗号化モードを例として図 3.51 に示す。ここで alen と mlen はそれぞれ関連データ、平文のバイト数である。BLEN はブロック長で、この場合 16 である。赤字が今回作成した関数に対応する。

```
int crypto_aead_encrypt(int mlen,  int alen)
{
        int   clen=0,size;

        // Associate Data Handling

        AESGCM_AD_0();                          // Initialization

        while(alen > 0){
                size = (alen >= BLEN) ? BLEN : alen;
                AESGCM_AD_123(size);            // one block processing
                aadr += BLEN;                   // aadr = address of AD
                alen -= BLEN;
        }

        // Message Handling

        AESGCM_ENC_0();                         // Initialization

        while(mlen > 0){
                size = (mlen >= BLEN) ? BLEN : mlen;
                clen += AESGCM_ENC_123(size);   // one block processing
                madr += BLEN;                   // madr = address of MSG
                cadr += BLEN;                   // cadr = address of OUT
                mlen -= BLEN;
        }

        AESGCM_ENC_4(alen, clen);               // Tag Generation

        return clen;
}
```

図 3.51　AES-GCM の暗号化モードのコード例

■AES-GCM

本評価では、文献 [10] に記述されている AES-GCM アルゴリズムを、もっとも一般的と考えられる次のパラメータで実装した。

表 3.23　本評価で用いた AES-GCM のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
AES-GCM	128 ビット	128 ビット	96 ビット	64 ビット

また、図 3.52 に示すように AES-GCM アルゴリズムを機能分割した。

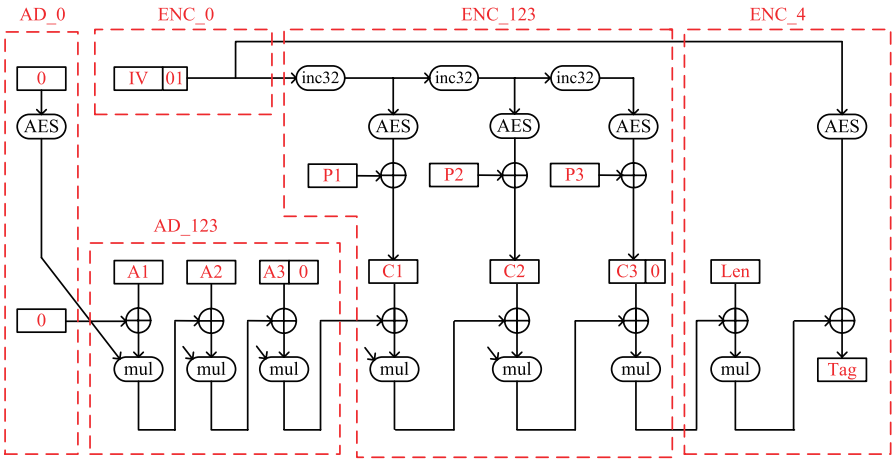

図 3.52　本評価で用いた AES-GCM の機能分割

■CLOC

CLOC アルゴリズム v2[18] には推奨パラメータが 3 つ示されている。本評価ではこのうち 2 つを実装した。このうちひとつは Core Function として TWINE-64-80 を用いるもの、もうひとつは Core Function として AES-128-128 を用いるものである。

表 3.24　本評価で用いた CLOC のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
CLOC-TWINE	80 ビット	64 ビット	48 ビット	32 ビット
CLOC-AES	128 ビット	128 ビット	96 ビット	64 ビット

また、図 3.53 に示すように CLOC アルゴリズムを機能分割した。なお、CLOC は平文のサイズが 0 の時に特別の処理を行う仕様となっている。この特別な処理を ENC_NULL と名付けている。

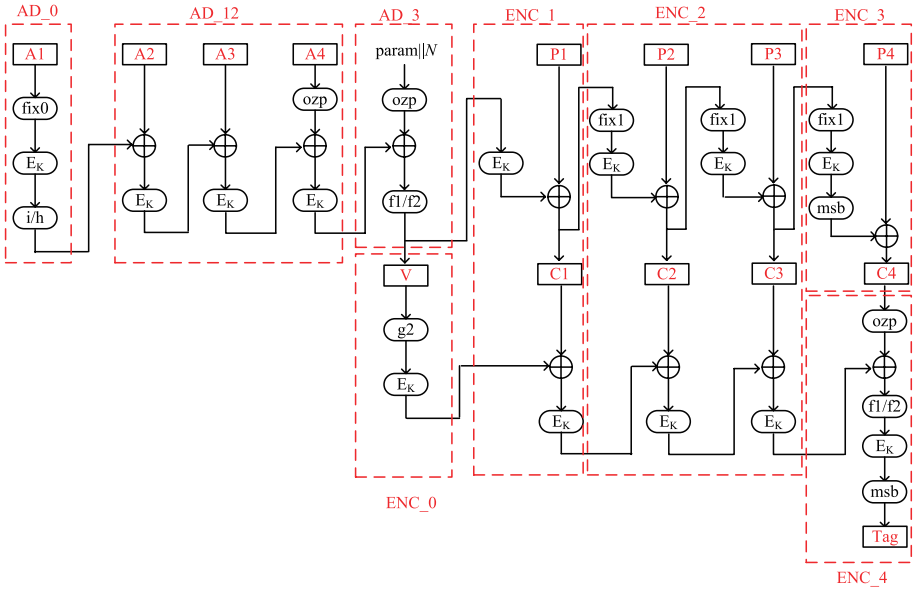

図 3.53　本評価で用いた CLOC の機能分割

■SILC

SILC アルゴリズム v2[12] には推奨パラメータが 4 つ示されている。本評価ではこのうち 2 つを実装した。このうちひとつは Core Function として PRESENT-64-80 を用いるもの、もうひとつは Core Function として AES-128-128 を用いるものである。

表 3.25　本評価で用いた SILC のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
SILC-PRESENT	80 ビット	64 ビット	48 ビット	32 ビット
SILC-AES	128 ビット	128 ビット	96 ビット	64 ビット

また、図 3.54 に示すように SILC アルゴリズムを機能分割した。

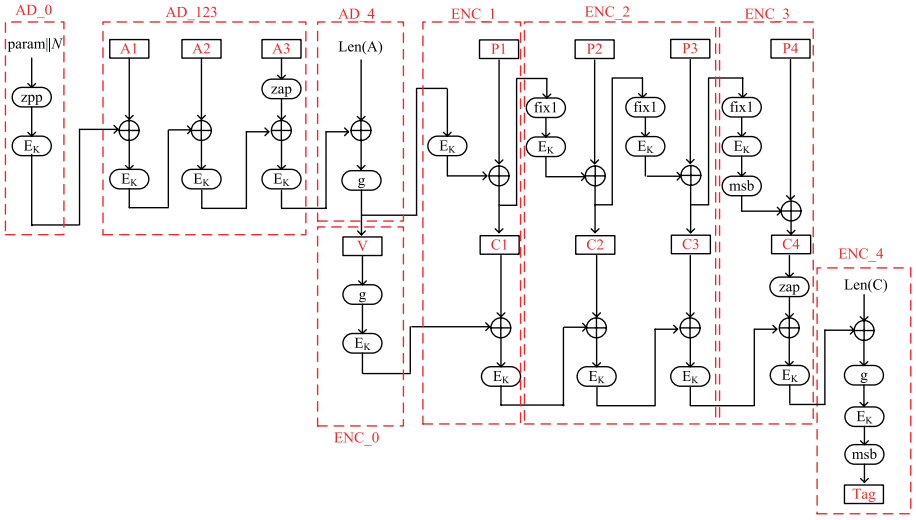

図 3.54　本評価で用いた SILC の機能分割

■Minalpher

Minalpher アルゴリズム [4] は Core Function として、Minalpher-P と呼ばれる置換が用いられている。パラメータは以下の一種類である。

表 3.26　本評価で用いた Minalpher のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
Minalpher	128 ビット	256 ビット	104 ビット	128 ビット

また、図 3.55 に示すように Minalpher アルゴリズムを機能分割した。

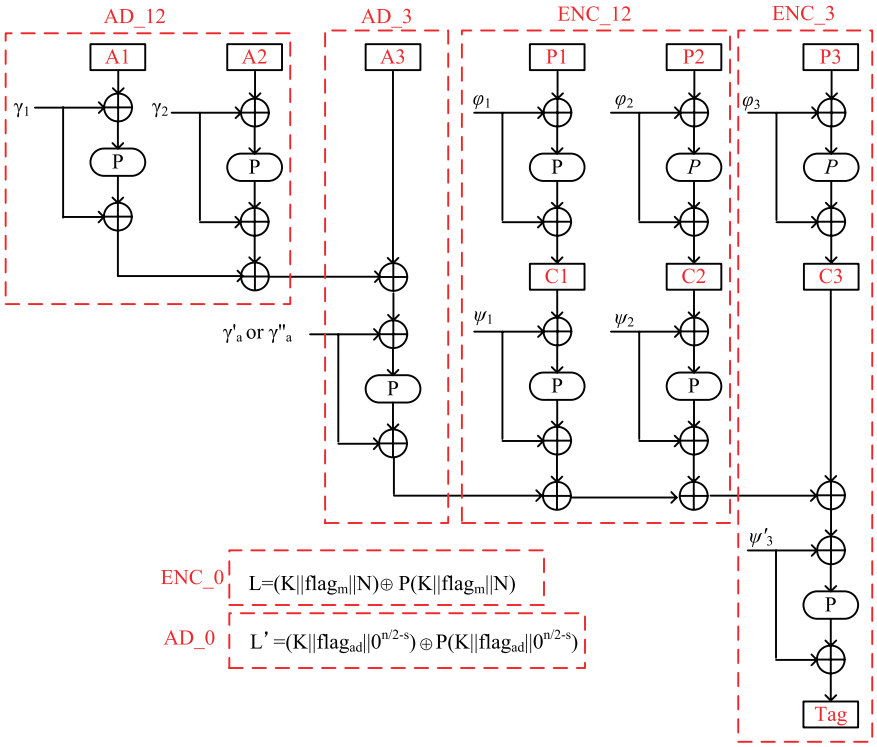

図 3.55　本評価で用いた Minalpher の機能分割

■AES-OTR

AES-OTR アルゴリズム [2] は Core Function として AES-128-128 あるいは AES-128-256 が用いられる。AES-OTR にはいくつかのパラメータが定義されているが、今回実装したのは Primary Parameter と呼ばれる以下のものであり、Core Function として AES-128-128 が使われている。

表 3.27　本評価で用いた AES-OTR のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
AES-OTR	128 ビット	128 ビット	96 ビット	128 ビット

また、図 3.56 に示すように AES-OTR アルゴリズムを機能分割した.

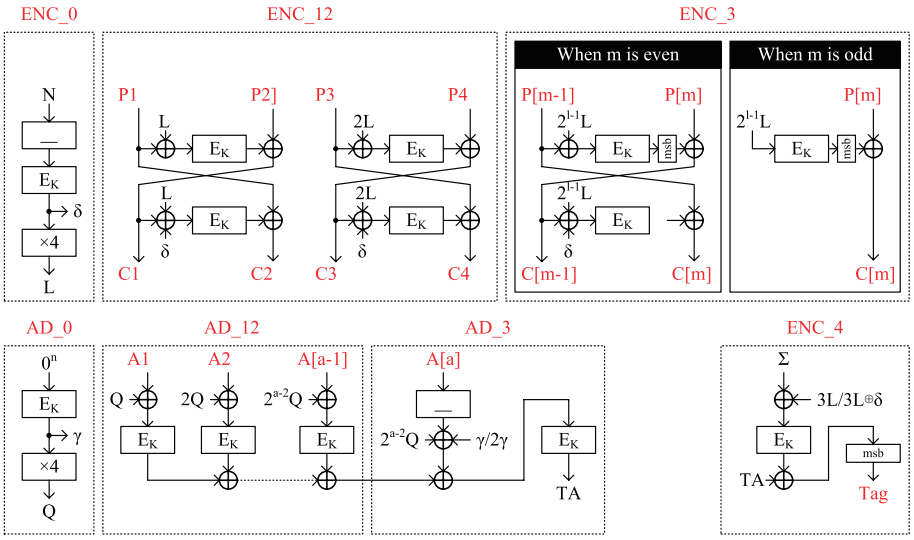

図 3.56　本評価で用いた AES-OTR の機能分割

■Ketje

Ketje アルゴリズム [3] は、Sponge 型認証暗号であり、Core Function として独自の関数 f が用いられている。Ketje には Ketje-SR と Ketje-JR の 2 つのパラメータが定義されているが、今回実装したのは Primary recommendation とされている、50 バイトの入出力を持つ関数 f を用いたものである。

表 3.28　本評価で用いた Ketje のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
Ketje-SR	128 ビット	32 ビット	128 ビット	128 ビット

また、図 3.57 に示すように Ketje アルゴリズムを機能分割した。なお、AD_12 と AD_3 は定数が異なる以外は同じ機能である。また、f の下に書かれた数字は f 内部の繰り返し回数を表している。

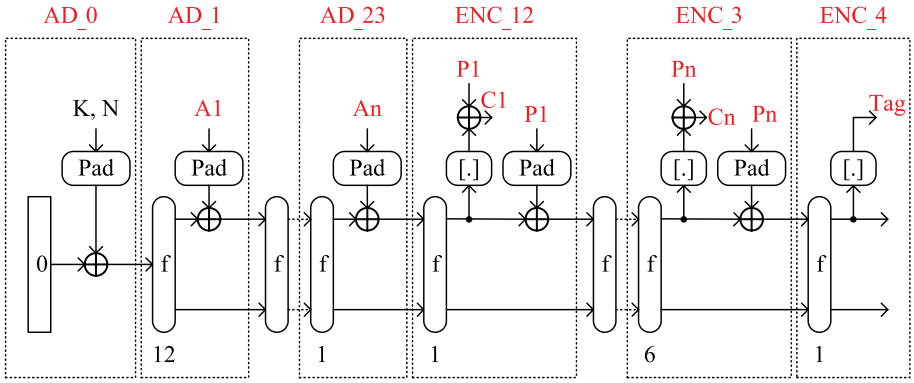

図 3.57　本評価で用いた Ketje の機能分割

■ACORN

ACORN は、ストリーム暗号型の認証暗号である。ACORN では、Core function となる StateUpdate を、制御ビットと入力を変更しながら繰り返し実行することにより、293 ビットの内部状態 state を変更しながら暗号処理を実行する。本評価では、ACORN アルゴリズムを、図 3.58 に示すパラメータで実装した。

表 3.29　本評価で用いた ACORN のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
ACORN	128 ビット	128 ビット	128 ビット	128 ビット

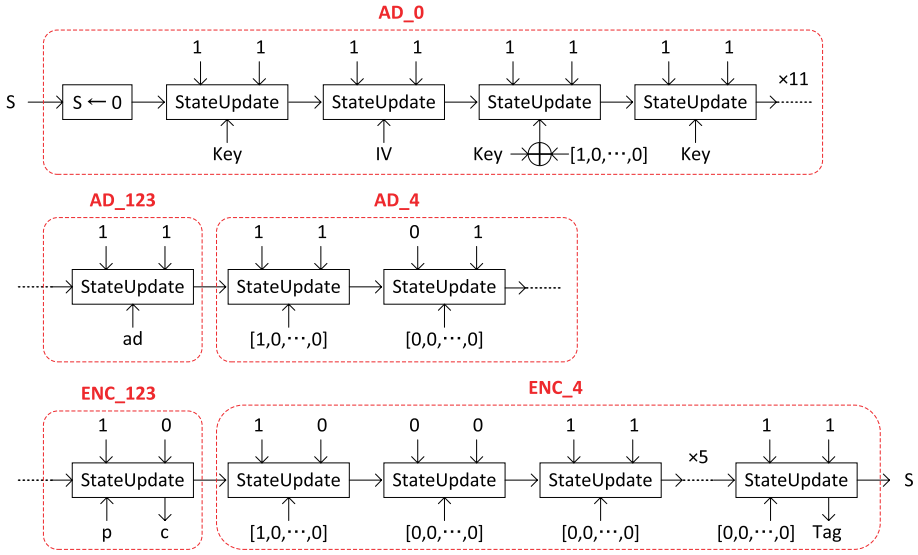

図 3.58　本評価で用いた ACORN の機能分割

■AES-OCB

AES-OCB アルゴリズム [5] は Core Function として AES-128, AES-192, あるいは AES-256 が用いられる。ここでは Primary recommendation と考えられる、Core Function として AES-128 を用いた以下のパラメータのものを実装した。なお、AES-OCB は、本ガイドラインでとりあげた他の AES ベースの認証暗号の中で唯一、(認証暗号としての) 復号時に AES の復号機能を必要とする。

表 3.30　本評価で用いた AES-OCB のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
AES-OCB	128 ビット	128 ビット	96 ビット	128 ビット

また、図 3.59 に示すように AES-OCB アルゴリズムを機能分割した。

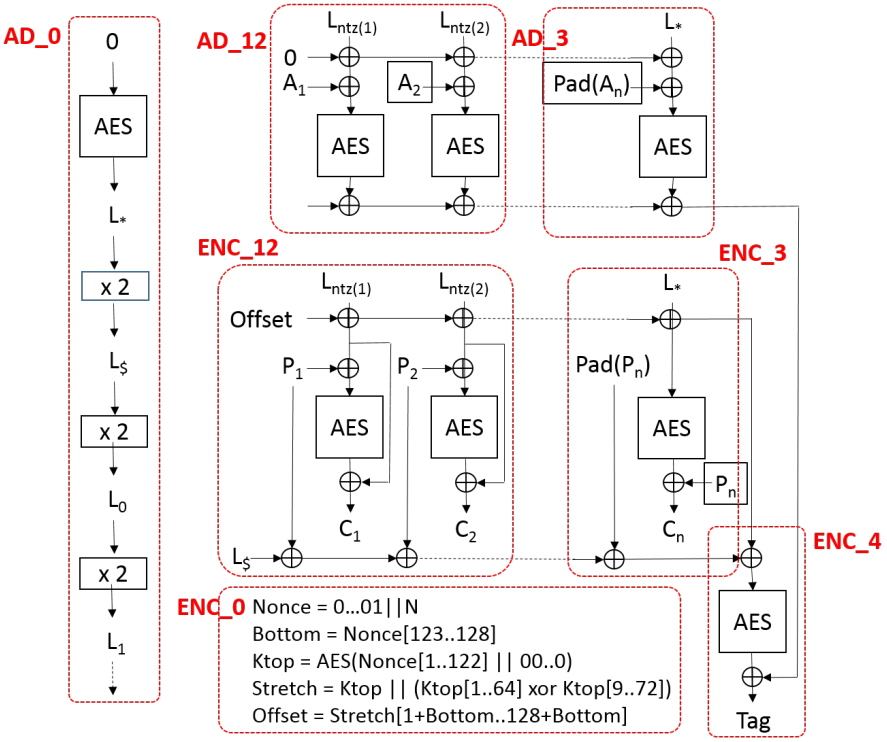

図 3.59　本評価で用いた AES-OCB の機能分割

■JAMBU

Jambu アルゴリズム [7] は、Core Function として SIMON-96-96, SIMON-64-96, SIMON-128-128 あるいは AES-128-128 が用いられる。ここでは Primary recommendation とされている SIMON-96-96 を用いたものに加えて、AES-128-128 を用いたものの 2 種類を実装した。これらは、それぞれ以下のパラメータを持つものである。

表 3.31　本評価で用いた JAMBU のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
JAMBU-SIMON	96 ビット	96 ビット	48 ビット	48 ビット
JAMBU-AES	128 ビット	128 ビット	64 ビット	64 ビット

また、図 3.60 に示すように JAMBU アルゴリズムを機能分割した。

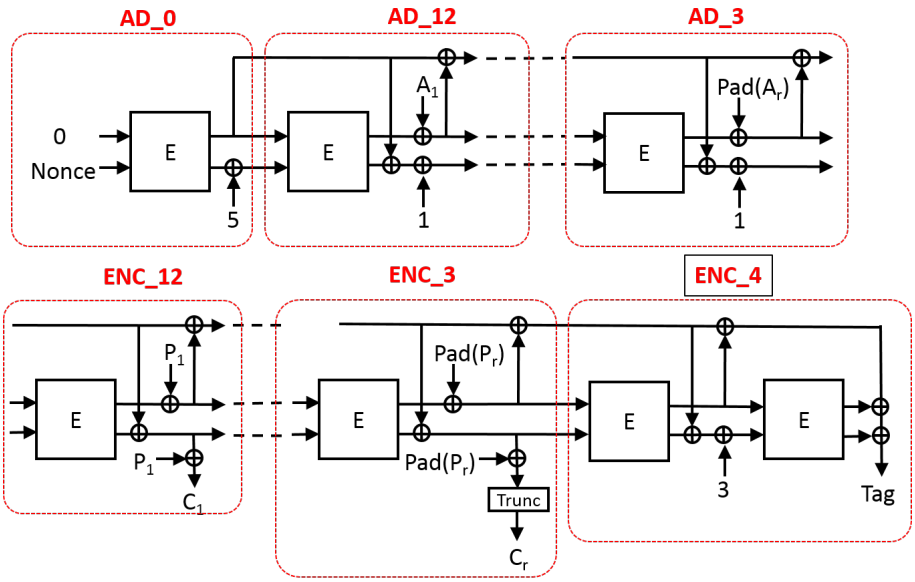

図 3.60　本評価で用いた JAMBU の機能分割

■Ascon

Ascon アルゴリズム [1] は Ketje と同じく Sponge 型認証暗号であり、独自の関数 p が内部で用いられている。ここでは Ascon の Primary recommendation である、以下のパラメータをもつものを実装した。なお、関数 p は 40 バイトの入出力をもつものである。

表 3.32　本評価で用いた Ascon のパラメータ

名称	鍵サイズ	ブロックサイズ	ナンスサイズ	タグサイズ
Ascon	128 ビット	128 ビット	96 ビット	128 ビット

また、図 3.61 に示すように Ascon アルゴリズムを機能分割した。ここで p^{12}, p^{6} と示されている関数は、それぞれ関数 p を 12 回、6 回実行させることを示している。

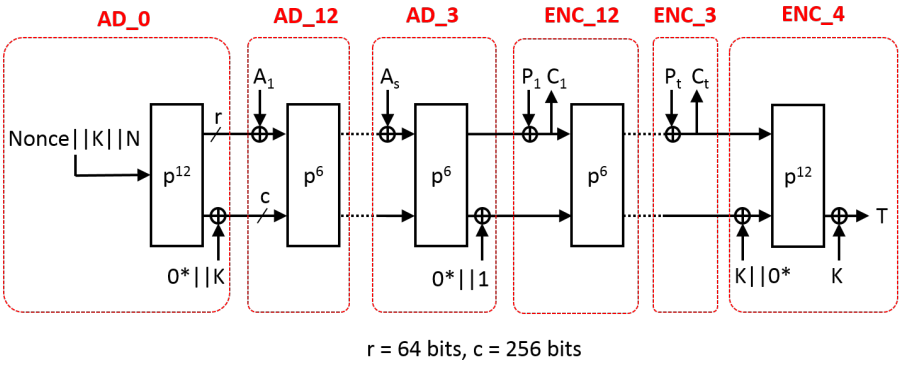

図 3.61　本評価で用いた Ascon の機能分割

参考文献

[1] ACORN v2. https://competitions.cr.yp.to/round3/asconv12.pdf.

[2] AES-OTR v1. http://competitions.cr.yp.to/round1/aesotrv1.pdf.

[3] Ketje. http://ketje.noekeon.org/.

[4] Minalpher Homepage. http://info.isl.ntt.co.jp/crypt/minalpher/index.html.

[5] OCB (v1.1).

[6] RL78 ファミリ. http://japan.renesas.com/products/mpumcu/rl78/index.jsp?campaign=tb_prod.

[7] The JAMBU Lightweight Authentication Encryption Mode. https://competitions.cr.yp.to/round3/jambuv21.pdf.

[8] Specification for the Advanced Encryption Standard (AES), 2001. http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf.

[9] Specification of Camellia – a 128-bit Block Cipher, 2001. http://info.isl.ntt.co.jp/crypt/camellia/dl/01espec.pdf.

[10] NIST SP 800-38D, Recommendation for Block Cipher Modes of Operation: Galois/Counter Mode (GCM) and GMAC, 2007. http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf.

[11] Recommendation for the Triple Data Encryption Algorithm (TDEA) Block Cipher, 2012. http://nvlpubs.nist.gov/nistpubs/Legacy/SP/nistspecialpublication800-67r1.pdf.

[12] CLOC and SILC—Authenticated Encryption Schemes for Constrained Devices, 2014. http://www.nuee.nagoya-u.ac.jp/labs/tiwata/AE/.

[13] ARM. AMBA 3 APB Protocol Specification v2.0, 2008. http://infocenter.arm.com/help/index.jsp?topic=/com.arm.doc.ihi0024c/index.html.

[14] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In *Advances in Cryptology – ASIACRYPT 2015 – 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, pages 411–436, 2015.

[15] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. SIMON and SPECK: block ciphers for the internet of things. *IACR Cryptology ePrint Archive*, 2015:585, 2015.

[16] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications

- extended abstract. In *Advances in Cryptology – ASIACRYPT 2012 – 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, pages 208–225, 2012.

[17] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In *Cryptographic Hardware and Embedded Systems – CHES 2011 – 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 326–341, 2011.

[18] Tetsu Iwata, Kazuhiko Minematsu, Jian Guo, and Sumio Morioka. CLOC: authenticated encryption for short input. In *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, pages 149–167, 2014.

[19] Lars R. Knudsen and Gregor Leander. PRESENT - block cipher. In *Encyclopedia of Cryptography and Security, 2nd Ed.*, pages 953–955. 2011.

[20] Mitsuru Matsui and Yumiko Murakami. Minimalism of software implementation - extensive performance analysis of symmetric primitives on the RL78 microcontroller. In *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, pages 393–409, 2013.

[21] Mitsuru Matsui and Yumiko Murakami. AES smaller than s-box - minimalism in software design on low end microcontrollers. In *Lightweight Cryptography for Security and Privacy – Third International Workshop, LightSec 2014, Istanbul, Turkey, September 1-2, 2014, Revised Selected Papers*, pages 51–66, 2014.

[22] Amir Moradi, Axel Poschmann, San Ling, Christof Paar, and Huaxiong Wang. Pushing the limits: A very compact and a threshold implementation of AES. In *Advances in Cryptology – EUROCRYPT 2011 – 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, May 15-19, 2011. Proceedings*, pages 69–88, 2011.

[23] Konstantinos Papagiannopoulos and Aram Verstegen. Speed and size-optimized implementations of the PRESENT cipher for tiny AVR devices. In *Radio Frequency Identification - Security and Privacy Issues 9th International Workshop, RFIDsec 2013, Graz, Austria, July 9-11, 2013, Revised Selected Papers*, pages 161–175, 2013.

[24] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An ultra-lightweight blockcipher. In *Cryptographic Hardware and Embedded Systems – CHES 2011 – 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 342–357, 2011.

[25] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, pages 181–195, 2007.

[26] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE : A lightweight block cipher for multiple platforms. In *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, pages 339–354, 2012.

[27] Hongjun Wu. ACORN v2. `http://competitions.cr.yp.to/caesar-submissions.html/`.

第 4 章

代表的な軽量暗号

4.1 ブロック暗号

本章では、主要な軽量ブロック暗号として CLEFIA、LED、Midori、Piccolo、PRESENT、PRINCE、SIMON、SPECK、TWINE の調査結果をまとめる。調査対象とした軽量ブロック暗号アルゴリズムは、主要国際学会で発表されており、現段階で有力な攻撃法が発見されておらず、かつ十分な実装性能を持つと考えられるアルゴリズムとした。各アルゴリズムのブロック長、鍵長といった基本入出力情報に加え、全体構造、および構成段数を記載している。鍵長やブロック長によって、個別の名称が与えられているアルゴリズムについてはそれぞれ個別の名称も記載した。アルゴリズムの特徴としては、主に提案論文で述べられている設計者らの主張をできる限りそのまま記載した。安全性解析状況は、設計仕様での安全性解析状況を記載するとともに、設計仕様における効率的な攻撃が知られていないアルゴリズムに関しては、参考情報として簡略化したアルゴリズムにおける最良の解析状況を記載している。なお、一般的に設計上の安全性は、鍵長、およびブロック長によって決められるが、PRINCE は設計上の安全性が通常の安全性より低く設定されていることに注意が必要である。また、設計者らが想定している攻撃者の能力についても、通常の Single-key setting での攻撃者に加え、アルゴリズムによってはさらに強力な関連鍵攻撃、弱鍵攻撃などに耐性を持つように設計されているものもあるため、それらのアルゴリズムについては特徴欄にその旨を記載している。ハードウェア実装性能調査としては主に十分な評価が行われていると考えられる ASIC での実装性能評価を調査し、実装ゲート規模 (GE)、1 ブロックの演算に必要なサイクル数 (cycles/block)、および 100kHz におけるスループットを記載している。また、ソフトウェア実装性能調査としてはハイエンド CPU での実装結果として 1 バイトの処理に必要なサイクル数 (cycles/byte) を記載し、ローエンド CPU での実装結果としては cycles/byte に加えて ROM、RAM 使用量を記載した。

<table>
<tr><td>技術分野</td><td>ブロック暗号</td></tr>
<tr><td>名称</td><td>CLEFIA</td></tr>
<tr><td>設計者</td><td>白井 太三、渋谷 香士、秋下 徹、盛合 志帆 (ソニー, 日本)、岩田 哲 (名古屋大学, 日本)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2007 (FSE 2007 [31])</td></tr>
<tr><td>仕様参照先</td><td>http://www.sony.net/Products/cryptography/clefia/[31]</td></tr>
<tr><td>特徴</td><td>設計者らは、高い安全性を保ちつつ、ハードウェア、ソフトウェアの両実装形態で高い実装性能を持つと主張している。また、AES と同じインタフェースに対応している点も特長である。
<table>
<tr><td>全体構造</td><td colspan="3">4-line type-II 一般化 Feistel 型</td></tr>
<tr><td>ブロック長 [bit]</td><td colspan="3">128</td></tr>
<tr><td>鍵長 [bit]</td><td>128
(CLEFIA-128)</td><td>192
(CLEFIA-192)</td><td>256
(CLEFIA-256)</td></tr>
<tr><td>構成段数 [段]</td><td>18</td><td>22</td><td>26</td></tr>
</table></td></tr>
<tr><td>安全性解析状況</td><td>多くの解析論文が発表されているが、仕様段数においては全数探索より効率的な攻撃は知られていない。
13 段に簡略化した CLEFIA-128、14 段に簡略化した CLEFIA-192、15 段に簡略化した CLEFIA-256 に対してはそれぞれ全数探索より効率的な攻撃が知られている [22, 9]。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Area
[GE]</th><th>Cycles/block</th><th>Throughput
@100kHz [kbps]</th><th>Ref.</th></tr>
<tr><td>CLEFIA-128 (Enc)</td><td>2,488</td><td>328</td><td>39.0</td><td>[2]</td></tr>
<tr><td>CLEFIA-128 (Enc/Dec)</td><td>2,604</td><td>328/320</td><td>39.0/40.0</td><td>[2]</td></tr>
<tr><td>CLEFIA-128 (Enc/Dec)</td><td>5,979</td><td>18</td><td>711.1</td><td>[31]</td></tr>
</table>
ソフトウェア実装評価結果 (ローエンド CPU)
<table>
<tr><th>Algorithm</th><th>ROM
[byte]</th><th>RAM
[byte]</th><th>Cycles/byte
[Enc/Dec]</th><th>Platform</th><th>Ref.</th></tr>
<tr><td>CLEFIA-128</td><td>1,309</td><td>78</td><td>39,357/152,023</td><td>RL78</td><td>[24]</td></tr>
<tr><td>CLEFIA-128</td><td>2,026</td><td>64</td><td>4,337/4,477</td><td>RL78</td><td>[24]</td></tr>
</table></td></tr>
<tr><td>標準化状況</td><td>ISO/IEC 29192-2</td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ブロック暗号</td></tr>
<tr><td>名称</td><td>LED</td></tr>
<tr><td>設計者</td><td>Jian Guo (Institute for Infocomm Research, シンガポール)、Thomas Peyrin、Axel Poschmann (南洋理工大学, シンガポール)、Matt Robshaw (Orange Labs, フランス)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2011 (CHES 2011 [19])</td></tr>
<tr><td>仕様参照先</td><td><code>http://led.crypto.sg/home-1</code> [19]</td></tr>
<tr><td>特徴</td><td>設計者らは、鍵スケジュールがなく、関連鍵攻撃耐性を持ち、ハードウェア実装での軽量性に特化しながらも十分なソフトウェア実装性能を持つと主張している。軽量ハッシュ関数 PHOTON と同様、serialized MDS を内部構造として採用している。
<table>
<tr><td>全体構造</td><td colspan="2">SPN 型</td></tr>
<tr><td>ブロック長 [bit]</td><td colspan="2">64</td></tr>
<tr><td>鍵長 [bit]</td><td>64 (LED-64)</td><td>128 (LED-128)</td></tr>
<tr><td>構成段数 [段]</td><td>32 (8 ステップ)</td><td>48 (12 ステップ)</td></tr>
</table></td></tr>
<tr><td>安全性解析状況</td><td>いくつかの解析論文が発表されているが、仕様段数においては全数探索より効率的な攻撃は知られていない。
12 段に簡略化した LED-64、および 32 段に簡略化した LED-128 に対しては全数探索より効率的な攻撃が知られている [17]。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Area [GE]</th><th>Cycles/block</th><th>Throughput@100kHz [kbps]</th><th>Ref.</th></tr>
<tr><td>LED-64 (Enc)</td><td>966</td><td>1,248</td><td>5.1</td><td>[19]</td></tr>
<tr><td>LED-64 (Enc)</td><td>2,695</td><td>32</td><td>200.0</td><td>[1]</td></tr>
<tr><td>LED-128 (Enc)</td><td>1,265</td><td>1,872</td><td>3.4</td><td>[19]</td></tr>
<tr><td>LED-128 (Enc)</td><td>3,036</td><td>48</td><td>133.3</td><td>[1]</td></tr>
</table>
ソフトウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Type</th><th>Cycles/byte</th><th>Platform</th><th>Ref.</th></tr>
<tr><td>LED-64</td><td>Table/VPI/Bitslice</td><td>76.0/48.1/13.1</td><td>Core i3 2367M</td><td>[6]</td></tr>
<tr><td>LED-128</td><td>Table/VPI/Bitslice</td><td>113.3/54.6/17.6</td><td>Core i3 2367M</td><td>[6]</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ブロック暗号</td></tr>
<tr><td>名称</td><td>Midori</td></tr>
<tr><td>設計者</td><td>Subhadeep Banik、Andrey Bogdanov (デンマーク工科大学, デンマーク)、五十部 孝典、渋谷 香士、樋渡 玄良、秋下 徹 (ソニー, 日本)、Francesco Regazzoni (ルガーノ大学, スイス)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2015 (ASIACRYPT 2015 [3])</td></tr>
<tr><td>仕様参照先</td><td>[3]</td></tr>
<tr><td>特徴</td><td>設計者らは、ハードウェア実装における小型実装性能、低レイテンシ性能に加え、低エネルギー消費性能に優れたアルゴリズムであると主張している。
<table>
<tr><td>全体構造</td><td colspan="2">SPN 型</td></tr>
<tr><td>ブロック長 [bit]</td><td>64 (Midori64)</td><td>128 (Midori128)</td></tr>
<tr><td>鍵長 [bit]</td><td colspan="2">128</td></tr>
<tr><td>構成段数 [段]</td><td>16</td><td>20</td></tr>
</table></td></tr>
<tr><td>安全性解析状況</td><td>いくつかの解析論文が発表されているが、仕様段数においては全数探索より効率的な攻撃は知られていない。
Midori64 については弱鍵が存在し、その弱鍵を使用している場合、仕様段数であっても効率的な攻撃が行える [18]。また、12 段に簡略化した Midori64 に対してはより効率的な攻撃が知られている [21]</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Area [GE]</th><th>Cycles/block</th><th>Throughput@100kHz [kbps]</th><th>Ref.</th></tr>
<tr><td>Midori64 (Enc)</td><td>1,542</td><td>16</td><td>400.0</td><td>[3]</td></tr>
<tr><td>Midori64 (Enc/Dec)</td><td>2,450</td><td>16</td><td>400.0</td><td>[3]</td></tr>
<tr><td>Midori128 (Enc)</td><td>2,522</td><td>20</td><td>640.0</td><td>[3]</td></tr>
<tr><td>Midori128 (Enc/Dec)</td><td>3,661</td><td>20</td><td>640.0</td><td>[3]</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ブロック暗号</td></tr>
<tr><td>名称</td><td>Piccolo</td></tr>
<tr><td>設計者</td><td>渋谷 香士、五十部 孝典、樋渡 玄良、三津田 敦司、秋下 徹、白井 太三 (ソニー, 日本)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2011 (CHES 2011 [30])</td></tr>
<tr><td>仕様参照先</td><td>[30]</td></tr>
<tr><td>特徴</td><td>設計者らは、従来の攻撃に加え、関連鍵攻撃、中間一致攻撃に対して十分な安全性を持ち、特にハードウェア実装での性能が高く、構造上、復号関数を実装したとしても大きなオーバーヘッドはなく、軽量性のみならずエネルギー効率も高いと主張している。
<table>
<tr><td>全体構造</td><td colspan="2">4-line 変形一般化 Feistel 型</td></tr>
<tr><td>ブロック長 [bit]</td><td colspan="2">64</td></tr>
<tr><td>鍵長 [bit]</td><td>80 (Piccolo-80)</td><td>128 (Piccolo-128)</td></tr>
<tr><td>構成段数 [段]</td><td>25</td><td>31</td></tr>
</table></td></tr>
<tr><td>安全性解析状況</td><td>いくつかの解析論文が発表されているが、仕様段数においては全数探索より効率的な攻撃は知られていない。
14 段に簡略化した Piccolo-80、および 21 段に簡略化した Piccolo-128 に対してはより効率的な攻撃 [30]、および関連鍵攻撃 [25] が知られている。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Area [GE]</th><th>Cycles/block</th><th>Throughput@100kHz [kbps]</th><th>Ref.</th></tr>
<tr><td>Piccolo-80 (Enc)</td><td>1,048</td><td>432</td><td>14.8</td><td>[30]</td></tr>
<tr><td>Piccolo-80 (Enc)</td><td>1,499</td><td>27</td><td>237.0</td><td>[30]</td></tr>
<tr><td>Piccolo-80 (Enc/Dec)</td><td>1,109</td><td>432</td><td>14.8</td><td>[30]</td></tr>
<tr><td>Piccolo-128 (Enc)</td><td>1,338</td><td>528</td><td>12.1</td><td>[30]</td></tr>
<tr><td>Piccolo-128 (Enc)</td><td>1,776</td><td>33</td><td>193.9</td><td>[30]</td></tr>
<tr><td>Piccolo-128 (Enc/Dec)</td><td>1,397</td><td>528</td><td>12.1</td><td>[30]</td></tr>
</table>
ソフトウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Type</th><th>Cycles/byte</th><th>Platform</th><th>Ref.</th></tr>
<tr><td>Piccolo-80</td><td>Bitslice</td><td>4.57</td><td>Core i7 870</td><td>[23]</td></tr>
<tr><td>Piccolo-128</td><td>Bitslice</td><td>5.52</td><td>Core i7 870</td><td>[23]</td></tr>
<tr><td>Piccolo-80</td><td>Table/VPI/Bitslice</td><td>89.3/33.3/9.2</td><td>Core i3 2367M</td><td>[6]</td></tr>
<tr><td>Piccolo-128</td><td>Table/VPI/Bitslice</td><td>103.6/41.6/10.9</td><td>Core i3 2367M</td><td>[6]</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ブロック暗号</td></tr>
<tr><td>名称</td><td>PRESENT</td></tr>
<tr><td>設計者</td><td>Andrey Bogdanov[1]、Lars R. Knudsen[2]、Gregor Leander[1]、Christof Paar[1]、Axel Poschmann[1]、Matthew J. B. Robshaw[3]、Yannick Seurin[3]、C. Vikkelsoe[2]
(1: ルール大学ボーフム, ドイツ、2: デンマーク工科大学, デンマーク、3:フランステレコム, フランス)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2007 (CHES 2007 [10])</td></tr>
<tr><td>仕様参照先</td><td>[10]</td></tr>
<tr><td>特徴</td><td>軽量ブロック暗号の草分け的アルゴリズムであり、特にハードウェアの小型実装において高い実装性能を持つ。
<table>
<tr><td>全体構造</td><td colspan="2">SPN 型</td></tr>
<tr><td>ブロック長 [bit]</td><td colspan="2">64</td></tr>
<tr><td>鍵長 [bit]</td><td>80 (PRESENT-80)</td><td>128 (PRESENT-128)</td></tr>
<tr><td>構成段数 [段]</td><td colspan="2">31</td></tr>
</table></td></tr>
<tr><td>安全性解析状況</td><td>多くの解析論文が発表されているが、仕様段数においては全数探索より効率的な攻撃は知られていない。
26 段に簡略化した PRESENT-80/128 に対してはより効率的な攻撃が知られている [14, 8]</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Area [GE]</th><th>Cycles/block</th><th>Throughput@100kHz [kbps]</th><th>Ref.</th></tr>
<tr><td>PRESENT-80 (Enc)</td><td>1,000</td><td>563</td><td>11.4</td><td>[29]</td></tr>
<tr><td>PRESENT-80 (Enc)</td><td>1,570</td><td>32</td><td>200.0</td><td>[10]</td></tr>
<tr><td>PRESENT-128 (Enc)</td><td>1,391</td><td>559</td><td>11.4</td><td>[27]</td></tr>
<tr><td>PRESENT-128 (Enc)</td><td>1,886</td><td>32</td><td>200.0</td><td>[10]</td></tr>
</table>
ソフトウェア実装評価結果 (ハイエンド CPU)
<table>
<tr><th>Algorithm</th><th>Type</th><th>Cycles/byte</th><th>Platform</th><th>Ref.</th></tr>
<tr><td>PRESENT-80/128</td><td>Bitslice</td><td>5.79</td><td>Core i7 870</td><td>[23]</td></tr>
<tr><td>PRESENT-80</td><td>Table/VPI/Bitslice</td><td>72.6/35.0/17.4</td><td>Core i3 2367M</td><td>[6]</td></tr>
<tr><td>PRESENT-128</td><td>Table/VPI/Bitslice</td><td>72.5/35.0/18.9</td><td>Core i3 2367M</td><td>[6]</td></tr>
</table>
ソフトウェア実装評価結果 (ローエンド CPU)
<table>
<tr><th>Algorithm</th><th>ROM [byte]</th><th>RAM [byte]</th><th>Cycles/byte [Enc/Dec]</th><th>Platform</th><th>Ref.</th></tr>
<tr><td>PRESENT-80</td><td>512</td><td>62</td><td>61,634/60,834</td><td>RL78</td><td>[24]</td></tr>
<tr><td>PRESENT-80</td><td>1,855</td><td>48</td><td>9,007/8,920</td><td>RL78</td><td>[24]</td></tr>
</table></td></tr>
<tr><td>標準化状況</td><td>ISO/IEC 29192-2</td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ブロック暗号</td></tr>
<tr><td>名称</td><td>PRINCE</td></tr>
<tr><td>設計者</td><td>Julia Borghoff[1]、Anne Canteaut[1,2]、Tim Guneysu[3]、Elif Bilge Kavun[3]、Miroslav Knezevic[4]、Lars R. Knudsen[1]、Gregor Leander[1]、Ventzislav Nikov[4]、Christof Paar[3]、Christian Rechberger[1]、Peter Rombouts[4]、Soren S. Thomsen[1]、Tolga Yalcin[3] (1: デンマーク工科大学, デンマーク、2: INRIA, フランス、3: ルール大学ボーフム, ドイツ、4: NXP セミコンダクター, オランダ)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2012 (ASIACRYPT 2012 [11])</td></tr>
<tr><td>仕様参照先</td><td>[11]</td></tr>
<tr><td>特徴</td><td>設計者らは、ハードウェア実装における小型実装性能に加え、低レイテンシ性能にも優れたアルゴリズムであると主張している。
α-reflection と呼ばれる対称性を持つことにより、通常のブロック暗号とは異なり、128-bit 鍵を利用していても攻撃者が 2^n の平文暗号文ペアを使える場合、$(127-n)$-bit の安全性しか主張できていない。
<table>
<tr><td>全体構造</td><td>SPN 型</td></tr>
<tr><td>ブロック長 [bit]</td><td>64</td></tr>
<tr><td>鍵長 [bit]</td><td>128</td></tr>
<tr><td>構成段数 [段]</td><td>12</td></tr>
</table></td></tr>
<tr><td>安全性解析状況</td><td>いくつかの解析論文が発表されているが、仕様段数においては全数探索より効率的な攻撃は知られていない。
10 段に簡略化した PRINCE に対してはより効率的な攻撃が知られている [12].</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Area [GE]</th><th>Cycles/block</th><th>Throughput@100kHz [kbps]</th><th>Ref.</th></tr>
<tr><td>PRINCE (Enc/Dec)</td><td>2,953</td><td>12</td><td>533.3</td><td>[4]</td></tr>
<tr><td>PRINCE (Enc/Dec)</td><td>8,577</td><td>1</td><td>6400.0</td><td>[4]</td></tr>
</table>
ソフトウェア実装評価結果 (ローエンド CPU)
<table>
<tr><th>Algorithm</th><th>ROM [byte]</th><th>RAM [byte]</th><th>Cycles/byte [Enc/Dec]</th><th>Platform</th><th>Ref.</th></tr>
<tr><td>PRINCE</td><td>2,382</td><td>220</td><td>225.4</td><td>ATtiny85</td><td>[26]</td></tr>
</table></td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ブロック暗号</td></tr>
<tr><td>名称</td><td>SIMON</td></tr>
<tr><td>設計者</td><td>Ray Beaulieu、Douglas Shors、Jason Smith、Stefan Treatman-Clark、Bryan Weeks、Louis Wingers (NSA(アメリカ国家安全保障局), 米国)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2013 (Cryptology ePrint Archive [5])</td></tr>
<tr><td>仕様参照先</td><td>[5]</td></tr>
<tr><td>特徴</td><td>設計者らは、様々なブロック長、鍵長に対応したアルゴリズムであり、軽量性において、ハードウェア、ソフトウェア両方で高い実装性能を持つが、特にハードウェアでの実装性能に優れると主張している。ブロック長 $2n$-bit、鍵長 m-word の SIMON を SIMON$2n/mn$ と表記する。例えば、SIMON64/128 はブロック長 64-bit、鍵長 128-bit の SIMON を表す。
<table>
<tr><td>全体構造</td><td colspan="10">Feistel 型</td></tr>
<tr><td>ブロック長 [bit]</td><td>32</td><td colspan="2">48</td><td colspan="2">64</td><td colspan="2">96</td><td colspan="3">128</td></tr>
<tr><td>鍵長 [bit]</td><td>64</td><td>72</td><td>96</td><td>96</td><td>128</td><td>96</td><td>144</td><td>128</td><td>192</td><td>256</td></tr>
<tr><td>構成段数 [段]</td><td>32</td><td colspan="2">36</td><td>42</td><td>44</td><td>52</td><td>54</td><td>68</td><td>69</td><td>72</td></tr>
</table></td></tr>
<tr><td>安全性解析状況</td><td>多くの解析論文が発表されているが、仕様段数においては全数探索より効率的な攻撃は知られていない。23, 25, 31, 38, 53 段に簡略化したブロック長 32, 48, 64, 96, 128-bit の SIMON に対してはより効率的な攻撃が知られている [13]。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Area [GE]</th><th>Cycles/block</th><th>Throughput@100kHz [kbps]</th><th>Ref.</th></tr>
<tr><td>SIMON64/96</td><td>809</td><td>1,455</td><td>4.4</td><td>[5]</td></tr>
<tr><td>SIMON64/128</td><td>958</td><td>1,524</td><td>4.2</td><td>[5]</td></tr>
<tr><td>SIMON128/128</td><td>1,234</td><td>4,414</td><td>2.9</td><td>[5]</td></tr>
<tr><td>SIMON128/256</td><td>1,782</td><td>4,923</td><td>2.6</td><td>[5]</td></tr>
</table>
ソフトウェア実装評価結果 (ローエンド CPU)
<table>
<tr><th>Algorithm</th><th>ROM [byte]</th><th>RAM [byte]</th><th>Cycles/byte</th><th>Platform</th><th>Ref. [Enc/Dec]</th></tr>
<tr><td>SIMON64/96</td><td>274</td><td>0</td><td>239</td><td>ATtiny45</td><td>[5]</td></tr>
<tr><td>SIMON64/128</td><td>282</td><td>0</td><td>250</td><td>ATtiny45</td><td>[5]</td></tr>
<tr><td>SIMON128/128</td><td>732</td><td>0</td><td>376</td><td>ATtiny45</td><td>[5]</td></tr>
<tr><td>SIMON128/256</td><td>764</td><td>0</td><td>398</td><td>ATtiny45</td><td>[5]</td></tr>
</table></td></tr>
<tr><td>オープンソース情報</td><td>設計者チームよりさまざまなプラットフォーム用のパブリックドメインソフトウェアがリリースされている [1]。[2] には AVR, MSP430, ARM Cortex M 向けに最適化されたブロック長 64bit の実装が、[3] には X86 や ARM Cortex A などのハイエンド CPU 向けに最適化されたブロック長 64, 128bit の実装が提供されている。他にも多くの小さなオープンソースプロジェクトに実装があるが [4, 5, etc.]、OpenSSL のように広く使われている暗号ライブラリには組み込まれていない。
[1]https://iadgov.github.io/simon-speck/implementations/
[2]https://www.cryptolux.org/index.php/FELICS
[3]https://bench.cr.yp.to/supercop.html
[4]https://rweather.github.io/arduinolibs/crypto.html
[5]http://cppcrypto.sourceforge.net/</td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ブロック暗号</td></tr>
<tr><td>名称</td><td>SPECK</td></tr>
<tr><td>設計者</td><td>Ray Beaulieu、Douglas Shors、Jason Smith、Stefan Treatman-Clark、Bryan Weeks、Louis Wingers (NSA(アメリカ国家安全保障局), 米国)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2013 (Cryptology ePrint Archive [5])</td></tr>
<tr><td>仕様参照先</td><td>[5]</td></tr>
<tr><td>特徴</td><td>設計者らは、様々なブロック長、鍵長に対応したアルゴリズムであり、軽量性において、ハードウェア、ソフトウェア両方で高い実装性能を持つが、特にソフトウェアでの実装性能に優れると主張している。SIMON 同様、ブロック長 $2n$-bit、鍵長 m-word の SPECK を SPECK$2n/mn$ と表記する。例えば、SPECK64/128 はブロック長 64-bit、鍵長 128-bit の SPECK を表す。
<table>
<tr><td>全体構造</td><td colspan="10">変形 Feistel 型</td></tr>
<tr><td>ブロック長 [bit]</td><td>32</td><td colspan="2">48</td><td colspan="2">64</td><td colspan="2">96</td><td colspan="3">128</td></tr>
<tr><td>鍵長 [bit]</td><td>64</td><td>72</td><td>96</td><td>96</td><td>128</td><td>96</td><td>144</td><td>128</td><td>192</td><td>256</td></tr>
<tr><td>構成段数 [段]</td><td colspan="2">22</td><td>23</td><td>26</td><td>27</td><td>28</td><td>29</td><td>32</td><td>33</td><td>34</td></tr>
</table></td></tr>
<tr><td>安全性解析状況</td><td>多くの解析論文が発表されているが、仕様段数においては全数探索より効率的な攻撃は知られていない。14, 15, 19, 17, 19 段に簡略化したブロック長 32, 48, 64, 96, 128-bit の SPECK に対してはより効率的な攻撃が知られている [7, 16]。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Area [GE]</th><th>Cycles/block</th><th>Throughput@100kHz [kbps]</th><th>Ref.</th></tr>
<tr><td>SPECK64/96</td><td>860</td><td>1,778</td><td>3.6</td><td>[5]</td></tr>
<tr><td>SPECK64/128</td><td>996</td><td>1,882</td><td>3.4</td><td>[5]</td></tr>
<tr><td>SPECK128/128</td><td>1,280</td><td>4,267</td><td>3.0</td><td>[5]</td></tr>
<tr><td>SPECK128/256</td><td>1,840</td><td>4,571</td><td>2.8</td><td>[5]</td></tr>
</table>
ソフトウェア実装評価結果 (ローエンド CPU)
<table>
<tr><th>Algorithm</th><th>ROM [byte]</th><th>RAM [byte]</th><th>Cycles/byte</th><th>Platform</th><th>Ref. [Enc/Dec]</th></tr>
<tr><td>SPECK64/96</td><td>182</td><td>0</td><td>144</td><td>ATtiny45</td><td>[5]</td></tr>
<tr><td>SPECK64/128</td><td>186</td><td>0</td><td>150</td><td>ATtiny45</td><td>[5]</td></tr>
<tr><td>SPECK128/128</td><td>396</td><td>0</td><td>167</td><td>ATtiny45</td><td>[5]</td></tr>
<tr><td>SPECK128/256</td><td>412</td><td>0</td><td>177</td><td>ATtiny45</td><td>[5]</td></tr>
</table></td></tr>
<tr><td>オープンソース情報</td><td>設計者チームよりさまざまなプラットフォーム用のパブリックドメインソフトウェアがリリースされている [1]。[2] には AVR, MSP430, ARM Cortex M 向けに最適化されたブロック長 64bit の実装が、[3] には X86 や ARM Cortex A などのハイエンド CPU 向けに最適化されたブロック長 64, 128bit の実装が提供されている。他にも多くの小さなオープンソースプロジェクトに実装があるが [4, 5, etc.]、OpenSSL のように広く使われている暗号ライブラリには組み込まれていない。
[1]https://iadgov.github.io/simon-speck/implementations/
[2]https://www.cryptolux.org/index.php/FELICS
[3]https://bench.cr.yp.to/supercop.html
[4]https://rweather.github.io/arduinolibs/crypto.html
[5]http://cppcrypto.sourceforge.net/</td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ブロック暗号</td></tr>
<tr><td>名称</td><td>TWINE</td></tr>
<tr><td>設計者</td><td>洲崎 智保、峯松 一彦、森岡 澄夫、小林 栄太 (NEC, 日本)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2011 (ECRYPT Workshop on Lightweight Cryptography、SAC 2012 [32])</td></tr>
<tr><td>仕様参照先</td><td>http://jpn.nec.com/rd/crl/code/research/image/twine_SAC_full_v5.pdf [32]</td></tr>
<tr><td>特徴</td><td>設計者らは、ハードウェアでの軽量性のみならず、ローエンド CPU からハイエンド CPU までの幅広いソフトウェアにおいても高い実装性能を持つと主張している。FSE 2010 で設計者らにより提案された改良ブロックシャッフルを採用し、安全性を高めている。
<table>
<tr><td>全体構造</td><td colspan="2">16-line 変形一般化 Feistel 型</td></tr>
<tr><td>ブロック長 [bit]</td><td colspan="2">64</td></tr>
<tr><td>鍵長 [bit]</td><td>80 (TWINE-80)</td><td>128 (TWINE-128)</td></tr>
<tr><td>構成段数 [段]</td><td colspan="2">36</td></tr>
</table></td></tr>
<tr><td>安全性解析状況</td><td>いくつかの解析論文が発表されているが、仕様段数においては全数探索より効率的な攻撃は知られていない。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装評価結果
<table>
<tr><th>Algorithm</th><th>Area [GE]</th><th>Cycles/block</th><th>Throughput@100kHz [kbps]</th><th>Ref.</th></tr>
<tr><td>TWINE-80 (Enc)</td><td>1,503</td><td>36</td><td>177.8</td><td>[32]</td></tr>
<tr><td>TWINE-80 (Enc)</td><td>1,011</td><td>393</td><td>16.3</td><td>[32]</td></tr>
<tr><td>TWINE-80 (Enc/Dec)</td><td>1,799</td><td>36</td><td>177.8</td><td>[32]</td></tr>
<tr><td>TWINE-128 (Enc)</td><td>1,866</td><td>36</td><td>177.8</td><td>[32]</td></tr>
<tr><td>TWINE-128 (Enc/Dec)</td><td>2,285</td><td>36</td><td>177.8</td><td>[32]</td></tr>
</table>
ソフトウェア実装評価結果 (ハイエンド CPU)
<table>
<tr><th>Algorithm</th><th>Type</th><th>Cycles/byte</th><th>Platform</th><th>Ref.</th></tr>
<tr><td>TWINE-80/128</td><td>Bitslice (Single/Double)</td><td>11.10/5.55</td><td>Core i7 2600S</td><td>[32]</td></tr>
</table>
ソフトウェア実装評価結果 (ローエンド CPU)
<table>
<tr><th>Algorithm</th><th>ROM [byte]</th><th>RAM [byte]</th><th>Cycles/byte [Enc/Dec]</th><th>Platform</th><th>Ref.</th></tr>
<tr><td>TWINE-80</td><td>2,294</td><td>386</td><td>163/163</td><td>ATmega163</td><td>[32]</td></tr>
<tr><td>TWINE-80</td><td>792</td><td>191</td><td>2,350/2,337</td><td>ATmega163</td><td>[32]</td></tr>
</table></td></tr>
<tr><td>利用実績等</td><td>NEC インダストリアル IoT ソリューション　加速度・温度センサ データ収集システム
http://jpn.nec.com/engsl/pro/securesensor/index.html
http://www.nec-eng.co.jp/press/161024press.html
NEC クラウド型防犯カメラサービス
http://jpn.nec.com/press/201604/20160421_02.html
NEC ExpEther 40G
http://jpn.nec.com/press/201511/20151109_01.html</td></tr>
</table>

参考文献

[1] The LED block cipher, December 2013. Available from `https://sites.google.com/site/ledblockcipher/hardware`.

[2] Toru Akishita and Harunaga Hiwatari. Very compact hardware implementations of the blockcipher CLEFIA. In Ali Miri and Serge Vaudenay, editors, *Selected Areas in Cryptography - 18th International Workshop, SAC 2011, Toronto, ON, Canada, August 11-12, 2011, Revised Selected Papers*, volume 7118 of *Lecture Notes in Computer Science*, pages 278–292. Springer, 2011.

[3] Subhadeep Banik, Andrey Bogdanov, Takanori Isobe, Kyoji Shibutani, Harunaga Hiwatari, Toru Akishita, and Francesco Regazzoni. Midori: A block cipher for low energy. In Tetsu Iwata and Jung Hee Cheon, editors, *Advances in Cryptology - ASIACRYPT 2015 - 21st International Conference on the Theory and Application of Cryptology and Information Security, Auckland, New Zealand, November 29 - December 3, 2015, Proceedings, Part II*, volume 9453 of *Lecture Notes in Computer Science*, pages 411–436. Springer, 2015.

[4] Lejla Batina, Amitabh Das, Baris Ege, Elif Bilge Kavun, Nele Mentens, Christof Paar, Ingrid Verbauwhede, and Tolga Yalçin. Dietary recommendations for lightweight block ciphers: Power, energy and area analysis of recently developed architectures. In Michael Hutter and Jörn-Marc Schmidt, editors, *Radio Frequency Identification - Security and Privacy Issues 9th International Workshop, RFIDsec 2013, Graz, Austria, July 9-11, 2013, Revised Selected Papers*, volume 8262 of *Lecture Notes in Computer Science*, pages 103–112. Springer, 2013.

[5] Ray Beaulieu, Douglas Shors, Jason Smith, Stefan Treatman-Clark, Bryan Weeks, and Louis Wingers. The SIMON and SPECK families of lightweight block ciphers. *IACR Cryptology ePrint Archive*, 2013:404, 2013.

[6] Ryad Benadjila, Jian Guo, Victor Lomné, and Thomas Peyrin. Implementing lightweight block ciphers on x86 architectures. In Lange et al. [20], pages 324–351.

[7] Alex Biryukov, Arnab Roy, and Vesselin Velichkov. Differential analysis of block ciphers SIMON and SPECK. In Cid and Rechberger [15], pages 546–570.

[8] Céline Blondeau and Kaisa Nyberg. Links between truncated differential and multidimensional linear properties of block ciphers and underlying attack complexities. In Phong Q. Nguyen and Elisabeth Oswald, editors, *Advances in Cryptology - EUROCRYPT 2014 - 33rd Annual International Conference on the Theory and Applications of Cryptographic Techniques, Copenhagen, Denmark, May 11-15, 2014. Proceedings*, volume 8441 of *Lecture Notes in Computer Science*, pages 165–182. Springer, 2014.

[9] Andrey Bogdanov, Huizheng Geng, Meiqin Wang, Long Wen, and Baudoin Collard. Zero-correlation

linear cryptanalysis with FFT and improved attacks on ISO standards Camellia and CLEFIA. In Lange et al. [20], pages 306–323.

[10] Andrey Bogdanov, Lars R. Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew J. B. Robshaw, Yannick Seurin, and C. Vikkelsoe. PRESENT: an ultra-lightweight block cipher. In Pascal Paillier and Ingrid Verbauwhede, editors, *Cryptographic Hardware and Embedded Systems - CHES 2007, 9th International Workshop, Vienna, Austria, September 10-13, 2007, Proceedings*, volume 4727 of *Lecture Notes in Computer Science*, pages 450–466. Springer, 2007.

[11] Julia Borghoff, Anne Canteaut, Tim Güneysu, Elif Bilge Kavun, Miroslav Knezevic, Lars R. Knudsen, Gregor Leander, Ventzislav Nikov, Christof Paar, Christian Rechberger, Peter Rombouts, Søren S. Thomsen, and Tolga Yalçin. PRINCE - A low-latency block cipher for pervasive computing applications - extended abstract. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology - ASIACRYPT 2012 - 18th International Conference on the Theory and Application of Cryptology and Information Security, Beijing, China, December 2-6, 2012. Proceedings*, volume 7658 of *Lecture Notes in Computer Science*, pages 208–225. Springer, 2012.

[12] Anne Canteaut, Thomas Fuhr, Henri Gilbert, María Naya-Plasencia, and Jean-René Reinhard. Multiple differential cryptanalysis of round-reduced PRINCE. In Cid and Rechberger [15], pages 591–610.

[13] Huaifeng Chen and Xiaoyun Wang. Improved linear hull attack on round-reduced simon with dynamic key-guessing techniques. In Thomas Peyrin, editor, *Fast Software Encryption - 23rd International Conference, FSE 2016, Bochum, Germany, March 20-23, 2016, Revised Selected Papers*, volume 9783 of *Lecture Notes in Computer Science*, pages 428–449. Springer, 2016.

[14] Joo Yeon Cho. Linear cryptanalysis of reduced-round PRESENT. In Josef Pieprzyk, editor, *Topics in Cryptology - CT-RSA 2010, The Cryptographers' Track at the RSA Conference 2010, San Francisco, CA, USA, March 1-5, 2010. Proceedings*, volume 5985 of *Lecture Notes in Computer Science*, pages 302–317. Springer, 2010.

[15] Carlos Cid and Christian Rechberger, editors. *Fast Software Encryption - 21st International Workshop, FSE 2014, London, UK, March 3-5, 2014. Revised Selected Papers*, volume 8540 of *Lecture Notes in Computer Science*. Springer, 2015.

[16] Itai Dinur. Improved differential cryptanalysis of round-reduced speck. In Antoine Joux and Amr M. Youssef, editors, *Selected Areas in Cryptography - SAC 2014 - 21st International Conference, Montreal, QC, Canada, August 14-15, 2014, Revised Selected Papers*, volume 8781 of *Lecture Notes in Computer Science*, pages 147–164. Springer, 2014.

[17] Itai Dinur, Orr Dunkelman, Nathan Keller, and Adi Shamir. Key recovery attacks on 3-round even-mansour, 8-step led-128, and full AES^2. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013 - 19th International Conference on the Theory and Application of Cryptology and Information Security, Bengaluru, India, December 1-5, 2013, Proceedings, Part I*, volume 8269 of *Lecture Notes in Computer Science*, pages 337–356. Springer, 2013.

[18] Jian Guo, Jérémy Jean, Ivica Nikolic, Kexin Qiao, Yu Sasaki, and Siang Meng Sim. Invariant subspace attack against full midori64. *IACR Cryptology ePrint Archive*, 2015:1189, 2015.

[19] Jian Guo, Thomas Peyrin, Axel Poschmann, and Matthew J. B. Robshaw. The LED block cipher. In Preneel and Takagi [28], pages 326–341.

[20] Tanja Lange, Kristin E. Lauter, and Petr Lisonek, editors. *Selected Areas in Cryptography - SAC 2013 - 20th International Conference, Burnaby, BC, Canada, August 14-16, 2013, Revised Selected Papers*, volume 8282 of *Lecture Notes in Computer Science*. Springer, 2014.

[21] Li Lin and Wenling Wu. Meet-in-the-middle attacks on reduced-round midori-64. *IACR Cryptology ePrint Archive*, 2015:1165, 2015.

[22] Hamid Mala, Mohammad Dakhilalian, and Mohsen Shakiba. Impossible differential attacks on 13-round CLEFIA-128. *J. Comput. Sci. Technol.*, 26(4):744–750, 2011.

[23] Seiichi Matsuda and Shiho Moriai. Lightweight cryptography for the cloud: Exploit the power of bitslice implementation. In Emmanuel Prouff and Patrick Schaumont, editors, *Cryptographic Hardware and Embedded Systems - CHES 2012 - 14th International Workshop, Leuven, Belgium, September 9-12, 2012. Proceedings*, volume 7428 of *Lecture Notes in Computer Science*, pages 408–425. Springer, 2012.

[24] Mitsuru Matsui and Yumiko Murakami. Minimalism of software implementation - extensive performance analysis of symmetric primitives on the RL78 microcontroller. In Shiho Moriai, editor, *Fast Software Encryption - 20th International Workshop, FSE 2013, Singapore, March 11-13, 2013. Revised Selected Papers*, volume 8424 of *Lecture Notes in Computer Science*, pages 393–409. Springer, 2013.

[25] Marine Minier. On the security of Piccolo lightweight block cipher against related-key impossible differentials. In Goutam Paul and Serge Vaudenay, editors, *Progress in Cryptology - INDOCRYPT 2013 - 14th International Conference on Cryptology in India, Mumbai, India, December 7-10, 2013. Proceedings*, volume 8250 of *Lecture Notes in Computer Science*, pages 308–318. Springer, 2013.

[26] Kostas Papapagiannopoulos. High throughput in slices: The case of present, PRINCE and KATAN64 ciphers. In Nitesh Saxena and Ahmad-Reza Sadeghi, editors, *Radio Frequency Identification: Security and Privacy Issues - 10th International Workshop, RFIDSec 2014, Oxford, UK, July 21-23, 2014, Revised Selected Papers*, volume 8651 of *Lecture Notes in Computer Science*, pages 137–155. Springer, 2014.

[27] Axel Poschmann. Lightweight cryptography - cryptographic engineering for a pervasive world. *IACR Cryptology ePrint Archive*, 2009:516, 2009.

[28] Bart Preneel and Tsuyoshi Takagi, editors. *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, volume 6917 of *Lecture Notes in Computer Science*. Springer, 2011.

[29] Carsten Rolfes, Axel Poschmann, Gregor Leander, and Christof Paar. Ultra-lightweight implementations for smart devices - security for 1000 gate equivalents. In Gilles Grimaud and François-Xavier Standaert, editors, *Smart Card Research and Advanced Applications, 8th IFIP WG 8.8/11.2 International Conference, CARDIS 2008, London, UK, September 8-11, 2008. Proceedings*, volume 5189 of *Lecture Notes in Computer Science*, pages 89–103. Springer, 2008.

[30] Kyoji Shibutani, Takanori Isobe, Harunaga Hiwatari, Atsushi Mitsuda, Toru Akishita, and Taizo Shirai. Piccolo: An ultra-lightweight blockcipher. In Preneel and Takagi [28], pages 342–357.

[31] Taizo Shirai, Kyoji Shibutani, Toru Akishita, Shiho Moriai, and Tetsu Iwata. The 128-bit blockcipher CLEFIA (extended abstract). In Alex Biryukov, editor, *Fast Software Encryption, 14th International Workshop, FSE 2007, Luxembourg, Luxembourg, March 26-28, 2007, Revised Selected Papers*, volume 4593 of *Lecture Notes in Computer Science*, pages 181–195. Springer, 2007.

[32] Tomoyasu Suzaki, Kazuhiko Minematsu, Sumio Morioka, and Eita Kobayashi. TWINE : A lightweight

block cipher for multiple platforms. In Lars R. Knudsen and Huapeng Wu, editors, *Selected Areas in Cryptography, 19th International Conference, SAC 2012, Windsor, ON, Canada, August 15-16, 2012, Revised Selected Papers*, volume 7707 of *Lecture Notes in Computer Science*, pages 339–354. Springer, 2012.

4.2 ストリーム暗号

この節では、軽量なストリーム暗号を取り上げる。ストリーム暗号はデータの秘匿を行うための関数だが、同じ機能を提供するブロック暗号と異なり、機能を秘匿のみに絞っている。したがって、(ソフトウェア実装での) プログラム領域や、(ハードウェア実装での) 回路規模などリソースに関する制約が強く、複数の暗号化関数を実装できないような場合、たとえば単一の暗号化関数を用いてメッセージ認証子も生成したい、といった場合には、ブロック暗号を選択するのが適当である。また、多くのストリーム暗号アルゴリズムは、初期化に要する時間が長くなりがちであり、短いデータを処理する場合にはブロック暗号が適する場合が多い。逆に、秘匿機能だけで良いが、省リソースで高速な処理を行いたい、といった場合にはストリーム暗号が適する。

軽量暗号の中では、ストリーム暗号の提案／評価が先行して実施された経緯もあり、他に比べて成熟しているアルゴリズムが多い。そこで、この節では、安全性評価が十分に行われたと考えられる eStream portfolio [21]、および ISO/IEC 29192-3 [13] に掲載されたアルゴリズムを中心に紹介する。eStream プロジェクトでは、ソフトウェア向けの Profile 1 とハードウェア向けの Profile 2 に分けてアルゴリズムの公募、評価を行っているが、この節では Profile 2 で portfolio に掲載された 3 つのアルゴリズム Grain v1 [17], MICKEY 2.0 [4], Trivium [9] を取り上げる。ソフトウェア向けの Profile 1 では 4 つのアルゴリズム HC-128, Rabbit, Salsa20/12, SOSEMANUK が portfolio に掲載されている。eBACS の PC、サーバ向け CPU を使った処理性能比較によれば、長いメッセージの処理については Salsa20/12 がもっとも優れている。ここでは、Salsa20/12 のかわりに、2015 年に RFC 化された ChaCha20 [7] を紹介する。ChaCha20 は、Salsa20/12 の改良版として開発されたソフトウェア向けのストリーム暗号であり、Salsa20/12 に比べると処理速度の点で幾分劣る。しかし、多くのオープンソースに採用されており、利用しやすさという点で優れている。ISO/IEC 29292-3 は 2 つのアルゴリズム Trivium と Enocoro を掲載している。そこで、本節では、上記のアルゴリズムに加えて Enocoro を取り上げる。

なお、軽量暗号に関する ISO 標準としては、RFID 向けの暗号技術を取り扱う ISO/IEC 29167 があるが、安全性の観点で望ましくない「XOR 暗号」が含まれており、CRYPTREC では一般に使用を奨めない。

技術分野	ストリーム暗号
名称	ChaCha20
設計者	Daniel J. Bernstein (イリノイ大学シカゴ校, 米国)
発表年 (発表学会等)	2008 (SASC 2008) [7]
仕様参照先	[6, 20]
特徴	鍵長 256-bit、IV 長 96-bit のストリーム暗号であり、秘密鍵、IV に加えて 128-bit の定数、32-bit のブロックカウンタを入力として 512-bit の疑似乱数を出力する。アルゴリズムは 32-bit ワードの算術加算、排他的論理和、巡回シフトで構成されており、ソフトウェア実装に適する。特に、初期化処理がほとんど無いため、短いメッセージに対しても高速であるという特徴を持つ。また、アルゴリズム中でテーブル参照を行わないため、素直に実装してもキャッシュタイミング攻撃に対して安全である。
安全性解析状況	2016 年時点では、ChaCha20 の安全性を脅かす結果は知られていない。Aumasson らは、差分攻撃を用いて 20 ラウンド中 7 ラウンドについて、鍵の総当たりよりも効率良く解読可能であることを示した [3]。この他にも、Shi らによる評価 [22]、Maitra による評価 [18] があり、それぞれ Aumasson らの結果を改良しているが、攻撃可能なラウンド数は 7 ラウンドに留まっている。
主な実装評価結果	eBASC [1] での評価 (2016 年 6 月現在) によれば、Intel Core i5 上で 1.2 cycles/byte を達成している。また、FELICS [2] での評価 (2016 年 6 月現在) によれば、ARM Cortex-M3 上の評価で、初期化に 144 cycles を要し、スループットは 54.3 cycles/byte である。
標準化状況	IETF RFC 7539 [20] で標準化されている。
利用実績等	主に多項式型メッセージ認証子の Poly-1305 と組み合わせた認証暗号として利用される。Google が提供するサービスの通信路 (https) 保護に利用されている [8]。
オープンソース情報	OpenSSL, Google Chrome, Mozilla Firefox, OpenSSH など。

技術分野	ストリーム暗号
名称	Enocoro
設計者	日立製作所, 日本
発表年 (発表学会等)	2008 (WAIS 2008), 2010 (ISITA 2010)
仕様参照先	`http://www.hitachi.co.jp/rd/yrl/crypto/enocoro/`
特徴	鍵長 80-bit の Enocoro-80、鍵長 128-bit の Enocoro-128v2 の 2 つのアルゴリズムから成る。いずれのアルゴリズムについても鍵長相当の安全性を謳っているが、鍵、IV を固定するごとに出力するデータはそれぞれ 2^{32} bytes、2^{64} bytes に制限されている。軽量ストリーム暗号には珍しく、8-bit 単位の処理で構成されており、ソフトウェア実装でも AES と同等の処理速度が得られる。
安全性解析状況	Enocoro-80, Enocoro-128v2 のいずれについても、single-key setting での脆弱性は知られていない。その一方で、Enocoro-80、および Enocoro-128v2 の前身である Enocoro-128v1.1 は関連鍵攻撃に対して脆弱であることが知られている [12, 24, 26, 10]。たとえば、Ding らの評価 [10] によれば、Enocoro-80 の秘密鍵は確率 2^{-8} で弱鍵であり、2^{17} 個の選択 IV を用いることで、計算量 2^{48} で秘密鍵を復元することができる。なお、Enocoro-128v2 については、関連鍵攻撃の有効性は確認されていない。
主な実装評価結果	Enocoro-80 [23]: Pentium 4 上での実装では、初期化に 1335 cycles を要し、スループットが 27 cycles/byte である。また、ハードウェア実装 (ASIC) での性能は回路規模が 2.7K GE, 処理速度が 2197.6 Mbps (180nm プロセス、最大動作周波数 274.7 MHz) である。 Enocoro-128v2 [25]: Intel Core2 Duo 上での実装では、初期化に 1530 cycles を要し、スループットが 14.8 cycles/byte である。また、ハードウェア実装 (ASIC) での性能は回路規模が 2.4K GE, 処理速度が 6250 Mbps (90 nm プロセス、最大動作周波数 781.3 MHz) である。
標準化状況	CRYPTREC 推奨候補暗号 (Enocoro-128v2), ISO 29192-3 (Enocoro-80, -128v2)

技術分野	ストリーム暗号
名称	Grain v1
設計者	Martin Hell, Thomas Johansson (ルンド大学, スウェーデン), Willi Meier (FH Aargau)
発表年 (発表学会等)	2005 (eSTREAM Project)
仕様参照先	`http://www.ecrypt.eu.org/stream/e2-grain.html`
特徴	eSTREAM portfolio に選ばれた鍵長 80 ビット（IV 長 64-bit)、鍵長 128 ビット（IV 長 80-bit）の 2 つのアルゴリズムから成る、ハードウェア実装向けのストリーム暗号である。ビット単位で処理を行う線形フィードバックシフトレジスタ 1 個と非線形フィードバックシフトレジスタ 1 個を組み合わせている。軽量ストリーム暗号の中でも特にハードウェア実装の軽量性に優れている。ある程度の並列処理が可能であり、ソフトウェア実装でも実用に耐える。
安全性解析状況	Berbain らは、Grain v1 の前身である Grain について評価を行い、2^{38} bits の鍵ストリームと 2^{43} の計算量で 80-bit の秘密鍵を復元できることを報告している [5]。Grain v1 はこの報告を受けてアルゴリズムを改良したものであり、Berbain らの攻撃をそのまま適用することはできない。 また、Dinur らは、Grain v1 の鍵長 128-bit のアルゴリズムについて、初期化が不十分であり 118 bits 分の弱鍵空間があると指摘している [11]。この弱鍵の空間は大きいが、この弱鍵に対して鍵復元攻撃を行うためには 2^{103} 程度の計算量が必要である。
主な実装評価結果	Grain v1 のハードウェア実装性能については、Good らの評価 [16] が詳しい。この評価では、0.13 μm スタンダードセルライブラリを用いて評価を行っている。 80-bit　回路規模 1,294 GE で最大動作周波数は 724.6 MHz、スループットは 724.6 MHz である。また、回路を 16 個まで並列処理することが可能である。 128-bit　回路規模 1,857 GE で最大動作周波数は 925.9 MHz、スループットは 925.9 MHz である。また、回路を 32 個まで並列処理することが可能である。
標準化状況	Grain v1 をベースにした認証暗号 Grain-128a が ISO/IEC 29167-13 [14] として標準化されている。

技術分野	ストリーム暗号
名称	MICKEY 2.0
設計者	Steve Babbage (Vodafone), Matthew Dodd (Independent consultant)
発表年 (発表学会等)	2005 (eSTREAM Project)
仕様参照先	`http://www.ecrypt.eu.org/stream/mickeypf.html`
特徴	eSTREAM portfolio に選ばれた鍵長 80-bit、IV 長 80-bit のハードウェア実装向けストリーム暗号である。1 つの鍵に対して、利用可能な IV の数は最大 2^{40} に制限されている。また、鍵と IV のペアに対して、利用可能な鍵ストリームは 2^{40} bits に制限されている。線形フィードバックシフトレジスタ 1 個と非線形フィードバックシフトレジスタ 1 個を組み合わせており、不規則なクロック制御を行うことを特徴としている。このクロック制御機構が原因で、並列処理は困難である。
安全性解析状況	MICKEY 2.0 に対する安全性評価結果は自己評価以外に知られておらず、脆弱性は見つかっていない。
主な実装評価結果	MICKEY 2.0 のハードウェア実装性能については、Good らの評価 [16] が詳しい。この評価では、0.13 μm スタンダードセルライブラリを用いて評価を行っている。MICKEY 2.0 の回路規模は 3,188 GE であり、最大動作周波数は 454.5 MHz、スループットは 454.5 MHz である。

技術分野	ストリーム暗号
名称	Trivium
設計者	Christophe De Cannière, Bart Preneel (ルーヴェン・カトリック大学, ベルギー)
発表年 (発表学会等)	2005 (eSTREAM Project)
仕様参照先	`http://www.ecrypt.eu.org/stream/e2-trivium.html`
特徴	eSTREAM portfolio に掲載された鍵長 80-bit、IV 長 80-bit のハードウェア実装向けのストリーム暗号である。鍵と IV のペアごとに生成される鍵ストリームは 2^{64} bits に制限される。3 つの非線形フィードバックシフトレジスタを直列した特徴的なアルゴリズムである。ビット単位の処理を基本としながら、高い並列性を持ち、ハードウェア実装での軽量性とソフトウェア実装での高速性を両立している。ただし、初期化に要する時間が長いため、短いデータの処理には適さない。
安全性解析状況	2016 年時点では、Trivium の安全性を脅かす結果は知られていない。Maximov らは、鍵ストリームから内部状態を推定する攻撃について、$2^{61.5}$ bits の鍵ストリームから、鍵の総当たりを $2^{89.5}$ 回行うのと同等の計算量で内部状態を復元できることが攻撃が可能であることを報告している [19]。また、Fouque らは、Trivium の初期化に対して cube attack を適用しており、2^{40} 個の IV(と対応する鍵ストリーム) および 2^{62} の総当たりを行うことで、1152 ステップ中 799 ステップまで鍵復元が可能であると報告している [15]。
主な実装評価結果	Trivium のハードウェア実装性能については、Good らの評価 [16] が詳しい。この評価では、0.13 μm スタンダードセルライブラリを用いて評価を行っている。Trivium の回路規模は 2,580 GE であり、最大動作周波数は 327.9 MHz、スループットは 327.9 Mbps である。また、Trivium のアルゴリズムは最大で 64 並列で実行することが可能であり、このときの回路規模は 4,921 GE、スループットは 22,299.6 Mbps である。 また、ソフトウェア実装については、FELICS [2] で評価が行われている。ARM Cortex-M3 上でのスループットは 49.4 cycles/byte であり、ChaCha20 よりも高速である。ただし、初期化に 7,195 cycles を要するため、短いデータの処理には適さない。
標準化状況	ISO/IEC 29192-3 [13]

参考文献

[1] ebacs: Ecrypt benchmarking of cryptographic systems. http://bench.cr.yp.to/results-stream.html.

[2] Felics stream ciphers brief results. https://www.cryptolux.org/index.php/FELICS_Stream_Ciphers_Brief_Results.

[3] Jean-Philippe Aumasson, Simon Fischer, Shahram Khazaei, Willi Meier, and Christian Rechberger. New features of latin dances: Analysis of salsa, chacha, and rumba. In Kaisa Nyberg, editor, *Fast Software Encryption FSE 2008*, volume 5086 of *Lecture Notes in Computer Science*, pages 470–488. Springer Berlin Heidelberg, 2008.

[4] Steve Babbage and Matthew Dodd. The mickey stream ciphers. In M. Robshaw and editors O. Billet, editors, *New Stream Cipher Designs: The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 191–209. Springer, 2008.

[5] C. Berbain, H. Gilbert, and A. Maximov. Cryptanalysis of grain. In M. Robshaw, editor, *Fast Software Encryption, FSE'06*, volume 4047 of *Lecture Notes in Computer Science*, pages 15–29. Springer-Verlag, 2006.

[6] Daniel J. Bernstein. Chacha, a variant of salsa20. https://cr.yp.to/chacha/chacha-20080128.pdf.

[7] Daniel J. Bernstein. Chacha, a variant of salsa20. In *The State of the Art of Stream Ciphers, SASC 2008*. ECRYPT, 2008.

[8] Elie Bursztein. Google security blog: Speeding up and strengthening https connections for chrome on android (april 24, 2014), 2014. https://security.googleblog.com/2014/04/speeding-up-and-strengthening-https.html.

[9] Christophe De Cannière. Trivium: A stream cipher construction inspired by block cipher design principles. In Sokratis K. Katsikas, Javier López, Michael Backes, Stefanos Gritzalis, and Bart Preneel, editors, *Information Security, ISC 2006*, volume 4176 of *Lecture Notes in Computer Science*, pages 171–186. Springer Berlin Heidelberg, 2006.

[10] Lin Ding, Chenhui Jin, and Jie Guan. Slide attack on standard stream cipher enocoro-80 in the related-key chosen iv setting. In *Pervasive and Mobile Computing, Special Issue on Secure Ubiquitous Computing*, volume 24, pages 224–230, 2015.

[11] I. Dinur and A. Shamir. Breaking grain-128 with dynamic cube attack. In A. Joux, editor, *Fast Software Encryption, FSE 2011*, volume 6733 of *Lecture Notes in Computer Science*, pages 167–187. Springer, 2011.

[12] D.Watanabe, T.Owada, K.Okamoto, Y.Igarashi, and T.Kaneko. Update on enocoro stream cipher. In

2010 International Symposium on Information Theory and its Applications, ISITA 2010, 2010.

[13] International Oragnization for Standards. *ISO/IEC 29192-3:2012 Information technology – Security techniques – Lightweight cryptography – Part 3: Stream ciphers*. October 2012.

[14] International Oragnization for Standards. *ISO/IEC 29167-13:2015 Information technology – Automatic identification and data capture techniques – Part 13: Crypto suite Grain-128A security services for air interface communications*. May 2015.

[15] Pierre-Alain Fouque and Thomas Vannet. Improving key recovery to 784 and 799 rounds of trivium using optimized cube attacks. In Shiho Moriai, editor, *Fast Software Encryption, FSE 2013*, volume 8424 of *Lecture Notes in Computer Science*, pages 502–517. Springer Berlin Heidelberg, 2013.

[16] Tim Good and Mohammed Benaissa. Hardware performance of estream phase-iii stream cipher candidates. In *The State of the Art of Stream Ciphers, SASC 2008*, 2008.

[17] Martin Hell, Thomas Johansson, Alexander Maximov, and Willi Meier. The grain family of stream ciphers. In M. Robshaw and editors O. Billet, editors, *New Stream Cipher Designs: The eSTREAM Finalists*, volume 4986 of *Lecture Notes in Computer Science*, pages 179–190. Springer, 2008.

[18] Subhamoy Maitra. Chosen iv cryptanalysis on reduced round chacha and salsa, July 2015. `https://eprint.iacr.org/2015/698`.

[19] Alexander Maximov and Alex Biryukov. Two trivial attacks on trivium. In Carlisle Adams and Ali Miri Michael Wiener, editors, *Selected Areas in Cryptography, 14th International Workshop, SAC 2007*, volume 4876 of *Lecture Notes in Computer Science*, pages 36–55. Springer Berlin Heidelberg, 2007.

[20] Y. Nir and A. Langley. *ChaCha20 and Poly1305 for IETF Protocols*, volume RFC4493 of *Request For Comments*. May. `https://tools.ietf.org/html/rfc7539`.

[21] ECRYPT Network of Excellence. The estream project. `http://www.ecrypt.eu.org/stream/`.

[22] Zhenqing Shi, Bin Zhang, Dengguo Feng, and Wenling Wu. Improved key recovery attacks on reduced-round salsa20 and chacha. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *ICISC'12 Proceedings of the 15th international conference on Information Security and Cryptology*, volume 7839 of *Lecture Notes in Computer Science*, pages 337–351. Springer-Verlag Berlin Heidelberg, 2011.

[23] D. Watanabe, K. Ideguchi, J. Kitahara, K. Muto, H. Furuichi, and T. Kaneko. Enocoro-80: A hardware oriented stream cipher. In *Second International Workshop on Advances in Information Security*, 2008.

[24] 五十嵐保隆, 岡本和人, 金子敏信. 関連鍵攻撃による Enocoro の弱鍵復元の検討. 電子情報通信学会技術研究報告, ISEC, pages 275–280, 2010.

[25] 三上 修吾, 渡辺 大. ストリーム暗号 Enocoro-128v2 のソフトウェアおよびハードウェア実装と評価. コンピュータセキュリティシンポジウム 2012 論文集, pages 742–748, 2012.

[26] 五十嵐 保隆, 岡本 和人, 金子 敏信. 関連鍵攻撃による Enocoro-128v1.1 の弱鍵復元の検討 (ii). 電子情報通信学会総合大会講演論文集, 2010.

4.3 ハッシュ関数

本節では軽量ハッシュ関数について記載する。ここで取り上げるアルゴリズムは、軽量ハッシュ関数として主要国際会議等に採録実績のある Keccak、PHOTON、QUARK、SPONGENT の 4 方式とする。 ただし、軽量という観点から、Keccak は SHA-3 として選定されたフルスペックのものではなく、置換関数のビット幅が小さいもののみを対象とする。また、実装評価結果は基本的に提案論文から抽出しており、それぞれが同一環境で評価されたものではないことに注意されたい。

<table>
<tr><td>技術分野</td><td>ハッシュ関数</td></tr>
<tr><td>名称</td><td>Keccak</td></tr>
<tr><td>設計者</td><td>Guido Bertoni (STMicroelectronics), Joan Daemen (STMicroelectronics), Michaël Peeters (NXP Semiconductors), Gilles Van Assche (STMicroelectronics)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2008 (NIST SHA-3 Competition)</td></tr>
<tr><td>仕様参照先</td><td>http://keccak.noekeon.org/</td></tr>
<tr><td>特徴</td><td>Keccak はスポンジ構造から成り、内部パラメータにより様々なバリエーションを構成できる。置換関数は 7 種類が定義されており、それぞれ Keccak-f[b] (b ∈ 25, 50, 100, 200, 400, 800, 1600) と表される。ここでは、軽量暗号の観点から、Keccak-f[100]、Keccak-f[200]、Keccak-f[400] を利用した方式について掲載する。
<table>
<tr><td>Keccak-f[b]</td><td>n</td><td>r</td><td>r’</td></tr>
<tr><td>Keccak-f[100]</td><td>80</td><td>20</td><td>20</td></tr>
<tr><td>Keccak-f[200]</td><td>64</td><td>72</td><td>72</td></tr>
<tr><td>Keccak-f[400]</td><td>128</td><td>144</td><td>144</td></tr>
</table>
∗n:出力長、r:入力ブロック長、r’:出力ブロック長</td></tr>
<tr><td>安全性解析状況</td><td>SHA-3 として標準化された方式に関する解析論文が多数存在するが、現時点では致命的な脆弱性は報告されていない。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装 [4](130nm process)
<table>
<tr><td></td><td>Area [GE]</td><td>Latency [clk]</td><td>Throughput [kbps]</td></tr>
<tr><td>Keccak-f[100]</td><td>1250</td><td>800</td><td>2.5</td></tr>
<tr><td>Keccak-f[200]</td><td>2520</td><td>900</td><td>8.00</td></tr>
<tr><td>Keccak-f[400]</td><td>5090</td><td>1000</td><td>14.40</td></tr>
</table></td></tr>
<tr><td>標準化状況</td><td>Keccak-f[1600] を利用した方式は SHA-3(FIPS 202) に採用されている。</td></tr>
<tr><td>利用実績等</td><td>SHA-3 としては多くのアプリケーションで導入されつつある。
http://csrc.nist.gov/groups/STM/cavp/documents/sha3/sha3val.html
http://www.3gpp.org/DynaReport/35-series.htm</td></tr>
<tr><td>オープンソース情報</td><td>http://keccak.noekeon.org/files.html
https://github.com/gvanas/KeccakCodePackage</td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ハッシュ関数</td></tr>
<tr><td>名称</td><td>PHOTON</td></tr>
<tr><td>設計者</td><td>Jian Guo (Institute for Infocomm Research, シンガポール)、Thomas Peyrin (南洋理工大学, シンガポール)、Axel Poschmann (南洋理工大学, シンガポール)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2011 (CRYPTO2011)</td></tr>
<tr><td>仕様参照先</td><td>https://sites.google.com/site/photonhashfunction/</td></tr>
<tr><td>特徴</td><td>PHOTON はスポンジ構造から成り、内部パラメータにより様々なバリエーションを構成できる。ISO/IEC 29192-5 では 5 種のバリエーションが示されている（下表)。置換関数は AES と似た構成で、AddConstants、SubCells、ShiftRows、MixColumnsSerial の 4 ステップを 12 ラウンド繰り返す。SubCells での変換には PRESENT の Sbox を利用する。
<table>
<tr><td>PHOTON-n/r/r’</td><td>n</td><td>r</td><td>r’</td></tr>
<tr><td>PHOTON-80/20/16</td><td>80</td><td>20</td><td>16</td></tr>
<tr><td>PHOTON-128/16/16</td><td>128</td><td>16</td><td>16</td></tr>
<tr><td>PHOTON-160/36/36</td><td>160</td><td>36</td><td>36</td></tr>
<tr><td>PHOTON-224/32/32</td><td>224</td><td>32</td><td>32</td></tr>
<tr><td>PHOTON-256/32/32</td><td>256</td><td>32</td><td>32</td></tr>
</table>
∗n:出力長、r:入力ブロック長、r’:出力ブロック長</td></tr>
<tr><td>安全性解析状況</td><td>現時点では致命的な脆弱性は報告されていないが、解析論文が少ないため潜在的な脆弱性を含んでいる可能性があることに注意。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装 [3](180nm process)
<table>
<tr><td></td><td>Area [GE]</td><td>Latency [clk]</td><td>Throughput [kbps]</td></tr>
<tr><td>PHOTON-80/20/16</td><td>865/1168</td><td>708/132</td><td>2.82/15.15</td></tr>
<tr><td>PHOTON-128/16/16</td><td>1122/1708</td><td>996/156</td><td>1.61/10.26</td></tr>
<tr><td>PHOTON-160/36/36</td><td>1396/2117</td><td>1332/180</td><td>2.70/20.00</td></tr>
<tr><td>PHOTON-224/32/32</td><td>1735/2786</td><td>1716/204</td><td>1.86/15.69</td></tr>
<tr><td>PHOTON-256/32/32</td><td>2177/4362</td><td>996/156</td><td>3.21/20.51</td></tr>
</table>
ソフトウェア実装 [3] (Intel Core i7 @1.6GHz)
<table>
<tr><td></td><td>32bit optimized implementation [cycles/byte]</td></tr>
<tr><td>PHOTON-80/20/16</td><td>95</td></tr>
<tr><td>PHOTON-128/16/16</td><td>156</td></tr>
<tr><td>PHOTON-160/36/36</td><td>116</td></tr>
<tr><td>PHOTON-224/32/32</td><td>227</td></tr>
<tr><td>PHOTON-256/32/32</td><td>135</td></tr>
</table></td></tr>
<tr><td>標準化状況</td><td>ISO/IEC 29192-5 (PHOTON-100, 140, 196, 256, 288)</td></tr>
<tr><td>オープンソース情報</td><td>https://sites.google.com/site/photonhashfunction/downloads</td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ハッシュ関数</td></tr>
<tr><td>名称</td><td>QUARK</td></tr>
<tr><td>設計者</td><td>Jean-Philippe Aumasson (Nagravision SA, Cheseaux, スイス)、Luca Henzen (ETH Zurich, スイス)、Willi Meier (FHNW, Windisch, スイス)、Maria Naya-Plasencia (FHNW, Windisch, スイス)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2010 (CHES2010)</td></tr>
<tr><td>仕様参照先</td><td>http://aumasson.jp/quark/</td></tr>
<tr><td>特徴</td><td>QUARK はスポンジ構造から成り、内部パラメータにより様々なバリエーションを構成できる。パラメータの違いにより、U-QUARK、D-QUARK、S-QUARK の 3 種類が示されている。置換関数はストリーム暗号 Grain とブロック暗号 KATAN の利点を組み合わせた構成となっている。ラウンド数はそれぞれ 544、704、1024 である。
<table>
<tr><td></td><td>n</td><td>r</td><td>r’</td></tr>
<tr><td>U-QUARK</td><td>128</td><td>8</td><td>8</td></tr>
<tr><td>D-QUARK</td><td>160</td><td>16</td><td>16</td></tr>
<tr><td>S-QUARK</td><td>224</td><td>32</td><td>32</td></tr>
</table>
∗n:出力長、r:入力ブロック長、r’:出力ブロック長</td></tr>
<tr><td>安全性解析状況</td><td>現時点では致命的な脆弱性は報告されていないが、解析論文が少ないため潜在的な脆弱性を含んでいる可能性があることに注意。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装 [1](180nm process)
<table>
<tr><td></td><td>Area [GE]</td><td>Latency [clk]</td><td>Throughput [kbps]</td></tr>
<tr><td>U-QUARK</td><td>1379/2392</td><td>544/68</td><td>1.47/11.76</td></tr>
<tr><td>D-QUARK</td><td>1702/2819</td><td>704/88</td><td>2.27/18.18</td></tr>
<tr><td>S-QUARK</td><td>2296/4640</td><td>1024/64</td><td>3.13/50.00</td></tr>
</table></td></tr>
<tr><td>オープンソース情報</td><td>http://aumasson.jp/quark/
https://github.com/veorq/Quark</td></tr>
</table>

<table>
<tr><td>技術分野</td><td>ハッシュ関数</td></tr>
<tr><td>名称</td><td>SPONGENT</td></tr>
<tr><td>設計者</td><td>Andrey Bogdanov[1]、Miroslav Knežević[2]、Gregor Leander[3]、Deniz Toz[1]、Kerem Varıcı[1]、Ingrid Verbauwhede[1] (1: ルーヴェン・カトリック大学, ベルギー、2: NXP Semiconductors, ベルギー、3: デンマーク工科大学, デンマーク)</td></tr>
<tr><td>発表年 (発表学会等)</td><td>2011 (CHES2011)</td></tr>
<tr><td>仕様参照先</td><td>https://sites.google.com/site/spongenthash/</td></tr>
<tr><td>特徴</td><td>SPONGENT は PRESENT タイプの置換関数を用いたスポンジ構造から成り、内部パラメータにより様々なバリエーションを構成できる。提案者により 13 種のバリエーションが示されており（下表)、そのうちの 5 種が ISO/IEC 29192-5 として標準化されている（表中の*印)。
<table>
<tr><td>SPONGENT-n/c/r</td><td>n</td><td>c</td><td>r</td></tr>
<tr><td>SPONGENT-88/80/8*</td><td>88</td><td>80</td><td>8</td></tr>
<tr><td>SPONGENT-88/176/88</td><td>88</td><td>176</td><td>88</td></tr>
<tr><td>SPONGENT-128/128/8*</td><td>128</td><td>128</td><td>8</td></tr>
<tr><td>SPONGENT-128/256/128</td><td>128</td><td>256</td><td>128</td></tr>
<tr><td>SPONGENT-160/160/16*</td><td>160</td><td>160</td><td>16</td></tr>
<tr><td>SPONGENT-160/160/80</td><td>160</td><td>160</td><td>80</td></tr>
<tr><td>SPONGENT-160/320/160</td><td>160</td><td>320</td><td>160</td></tr>
<tr><td>SPONGENT-224/224/16*</td><td>224</td><td>224</td><td>16</td></tr>
<tr><td>SPONGENT-224/224/112</td><td>224</td><td>224</td><td>112</td></tr>
<tr><td>SPONGENT-224/448/224</td><td>224</td><td>448</td><td>224</td></tr>
<tr><td>SPONGENT-256/256/16*</td><td>256</td><td>256</td><td>16</td></tr>
<tr><td>SPONGENT-256/256/128</td><td>256</td><td>256</td><td>128</td></tr>
<tr><td>SPONGENT-256/512/256</td><td>256</td><td>512</td><td>256</td></tr>
</table>
∗n:出力長、c:capacity、r:rate(入力ブロック長)</td></tr>
<tr><td>安全性解析状況</td><td>現時点では致命的な脆弱性は報告されていないが、解析論文が少ないため潜在的な脆弱性を含んでいる可能性があることに注意。</td></tr>
<tr><td>主な実装評価結果</td><td>ハードウェア実装 [2](130nm process)
<table>
<tr><td></td><td>Area [GE]</td><td>Latency [clk]</td><td>Throughput [kbps]</td></tr>
<tr><td>SPONGENT-88/80/8</td><td>738/1127</td><td>990/45</td><td>0.81/17.78</td></tr>
<tr><td>SPONGENT-128/128/8</td><td>1060/1687</td><td>2380/70</td><td>0.34/11.43</td></tr>
<tr><td>SPONGENT-160/160/16</td><td>1329/2190</td><td>3960/90</td><td>0.40/17.78</td></tr>
<tr><td>SPONGENT-224/224/16</td><td>1728/2903</td><td>7200/120</td><td>0.22/13.33</td></tr>
<tr><td>SPONGENT-256/256/16</td><td>1950/3281</td><td>9520/140</td><td>0.17/11.43</td></tr>
</table></td></tr>
<tr><td>標準化状況</td><td>ISO/IEC 29192-5:2016 (SPONGENT-88, 136, 176, 240, 272)
http://www.iso.org/iso/catalogue_detail.htm?csnumber=67173</td></tr>
<tr><td>オープンソース情報</td><td>https://sites.google.com/site/spongenthash/downloads</td></tr>
</table>

参考文献

[1] Jean-Philippe Aumasson, Luca Henzen, Willi Meier, and María Naya-Plasencia. Quark: A lightweight hash. In *Cryptographic Hardware and Embedded Systems, CHES 2010, 12th International Workshop, Santa Barbara, CA, USA, August 17-20, 2010. Proceedings*, pages 1–15, 2010.

[2] Andrey Bogdanov, Miroslav Knezevic, Gregor Leander, Deniz Toz, Kerem Varici, and Ingrid Verbauwhede. spongent: A lightweight hash function. In *Cryptographic Hardware and Embedded Systems - CHES 2011 - 13th International Workshop, Nara, Japan, September 28 - October 1, 2011. Proceedings*, pages 312–325, 2011.

[3] Jian Guo, Thomas Peyrin, and Axel Poschmann. The PHOTON family of lightweight hash functions. In *Advances in Cryptology - CRYPTO 2011 - 31st Annual Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2011. Proceedings*, pages 222–239, 2011.

[4] Elif Bilge Kavun and Tolga Yalçin. A lightweight implementation of keccak hash function for radio-frequency identification applications. In *Radio Frequency Identification: Security and Privacy Issues - 6th International Workshop, RFIDSec 2010, Istanbul, Turkey, June 8-9, 2010, Revised Selected Papers*, pages 258–269, 2010.

4.4 メッセージ認証コード

本節では、軽量なメッセージ認証コード (Message Authentication Code: MAC) を取り上げる。汎用的に用いられている MAC は、主にブロック暗号のモード (CMAC [10]) やハッシュ関数のモード (HMAC [11, 5]) である。CMAC や HMAC は、モード自体のオーバーヘッドがそれほど大きくないので、4.1 節に掲載されている軽量ブロック暗号や、4.3 節に掲載されている軽量ハッシュ関数と組み合せることで、軽量な MAC を構成することができる。HMAC はハッシュ関数を 2 回呼び出すため、処理するメッセージ長が非常に短い場合には、ブロック暗号ベースの CMAC を用いる方が効率的である可能性が高い。

ソフトウェア実装の性能に限れば、軽量暗号のベンチマークを行っている FELICS プロジェクト [4] (Fair Evaluation of Lightweight Cryptographic Systems) がブロック暗号やハッシュ関数を選択する参考になる。FELICS では、Atmel AVR ATmega128 (8-bit)、Texas Instruments MSP430F1611 (16-bit)、Arduino Due ARM Cortex-M3 (32-bit) 上で多数のアルゴリズム (ブロック暗号、ストリーム暗号、ハッシュ関数) を比較している。

ブロック暗号やハッシュ関数のモードではない、専用に設計された軽量 MAC はそれほど多く知られていないが、短いメッセージの処理に特価した擬似ランダム関数 SipHash [2, 1] は MAC として利用可能であり、多くの利用実績がある。ただし、SipHash は内部処理で 64-bit 加算などを利用しているので、「比較的ハイエンドの CPU 上で高速」という意味での軽量 MAC であり、8〜32-bit CPU での使用には適さない。

この他に、ローエンド CPU 向けの軽量 MAC として Chaskey [9, 7] がある。FELICS では Chaskey は軽量ブロック暗号に分類されており、多くの項目で最も優秀な成績を収めている。その一方で、8 ラウンド中 7 ラウンドについて鍵復元攻撃が可能であることが報告されており [6]、安全性の観点ではセキュリティマージンが小さい。この点を改善するために、ラウンド数を 8 から 12 に増やした Chaskey-12 [8] が提案されている。

本節では、現時点で十分に安全と考えられる MAC として、SipHash のみを取り上げる。CMAC, HMAC については、本節では特に取り上げない。また、Chaskey についても、現時点では十分にレビューを受けたアルゴリズムとは言えないため、本節では取り上げない。

技術分野	メッセージ認証コード
名称	SipHash
設計者	Jean-Philippe Aumasson (Kudelski Security, スイス), Daniel J. Bernstein (イリノイ大学シカゴ校, 米国)
発表年 (発表学会等)	2012 (INDOCRYPT 2012)
仕様参照先	[2, 1]
特徴	SipHash は、連想配列に用いるハッシュ関数として開発された鍵長 128-bit、出力長 64-bit の鍵付きハッシュ関数である。入力されるメッセージ長の上限は 2039 bytes であり、汎用のハッシュ関数に比べて短い。SipHash のアルゴリズムは c ラウンドの圧縮フェーズと d ラウンドの最終処理フェーズからなり、SipHash-c-d と表される。一般に利用されているのは、$c = 2, d = 4$ の SipHash-2-4 である。64-bit ワードを単位とし、算術加算、排他的論理和、巡回シフトを組み合わせたアルゴリズムであり、64-bit 演算をサポートする CPU 上で高速に動作する。また、アルゴリズム中でテーブル参照を行わないため、素直に実装してもキャッシュタイミング攻撃に対して安全である。
安全性解析状況	2016 年時点において、SipHash の脆弱性は知られていない。第三者による評価として、Dobraunig らによる結果 [3] がある。彼らは、差分特性確率が $2^{-236.3}$ の差分パスを発見している。しかし、SipHash の鍵長は 128-bit であり、鍵の総当たり攻撃の方がはるかに効率的であるので、この結果は SipHash の安全性を脅かすものではない。
主な実装評価結果	提案論文 [2] によれば、SipHash のスループットは amd64 アーキテクチャ上で 1.5〜3.0 cycles/byte である。メッセージ長が短い場合には最終処理のオーバーヘッドが大きく、8 bytes のデータでは 10〜30 cycles/byte となる。
標準化状況	現時点では、(デジュールの) 標準化には提案されていない。しかし、多くのオープンソースライブラリに実装されており、デファクト標準の地位を固めつつある。
利用実績等	オープンソース、特に軽量プログラミング言語 (Perl, Python, Ruby 等) の連想配列で用いるハッシュ関数として広く採用されている。この他にも、[1] によれば Wireguard, Bloomberg, OpenBSD, Shardmap, SoundHound, FreeBSD, Hashable, Rubinius, JRuby, Redis, OpenDNS, Rust, Sodium が SipHash を採用している。
オープンソース情報	前項を参照のこと。

参考文献

[1] Jean-Philippe Aumasson. Siphash: a fast short-input prf. https://131002.net/siphash/.

[2] Jean-Philippe Aumasson and Daniel J. Bernstein. Siphash: A fast short-input prf. In Steven Galbraith and Mridul Nandi, editors, *Progress in Cryptology – INDOCRYPT 2012*, volume 7668 of *Lecture Notes in Computer Science*, pages 489–508. Springer-Verlag Berlin Heidelberg, 2012.

[3] Christoph Dobraunig, Florian Mendel, and Martin Schläffer. Differential cryptanalysis of siphash. In Antoine Joux and Amr Youssef, editors, *Selected Areas in Cryptography – SAC 2014*, volume 8781 of *Lecture Notes in Computer Science*, pages 165–182. Springer International Publishing, 2014.

[4] FELICS. Fair evaluation of lightweight cryptographic systems. https://www.cryptolux.org/index.php/FELICS.

[5] H. Krawczyk, M. Bellare, and R. Canetti. *HMAC: Keyed-Hashing for Message Authentication*, volume RFC2104 of *Request For Comments*. February. https://tools.ietf.org/html/rfc2104.

[6] Gaëtan Leurent. Improved differential-linear cryptanalysis of 7-round chaskey with partitioning. In Marc Fischlin and Jean-Sébastien Coron, editors, *Advances in Cryptology – EUROCRYPT 2016*, volume 9665 of *Lecture Notes in Computer Science*, pages 344–371. Springer, 2016.

[7] Nicky Mouha. Chaskey. http://mouha.be/chaskey/.

[8] Nicky Mouha. Chaskey: a mac algorithm for microcontrollers – status update and proposal of chaskey-12 –. http://eprint.iacr.org/2015/1182.

[9] Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel, , and Ingrid Verbauwhede. Chaskey: An efficient mac algorithm for 32-bit microcontrollers. In Antoine Joux and Amr Youssef, editors, *Selected Areas in Cryptography – SAC 2014*, volume 8781 of *Lecture Notes in Computer Science*, pages 306–323. Springer, 2014.

[10] National Institute of Standards and Technology. *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. NIST Special Publication 800-38B. May 2005. http://csrc.nist.gov/publications/nistpubs/800-38B/SP_800-38B.pdf.

[11] National Institute of Standards and Technology. *The Keyed-Hash Message Authentication Code (HMAC)*. Federal Information Processing Standards Publication FIPS 198-1. July 2008. http://csrc.nist.gov/publications/fips/fips198-1/FIPS-198-1_final.pdf.

4.5 認証暗号

本節では、CAESAR コンペティション [6] において提案された方式のうち、軽量性を謳い、かつ安全性の観点で問題が見つかっていない（2016 年 7 月 28 日の時点）方式を中心に概要を示す。なお上記の日付以降に公開されたいくつかの結果については参考として記載する。ここで示す方式には、ブロック暗号ないし Tweakable ブロック暗号を用いているものも多い。これらの方式については理論的速度を測る指標としてレートを導入する。 レートは 1 ブロック暗号で処理可能な入力ブロック数を表す。ソフトウェアの実装評価値は特に断りのない限り eBACS 内の Supercop ベンチマークシステム [7] での十分長いメッセージ処理の結果、ハードウェアの実装評価値も同様に特に断りのない限り ATHENA ベンチマークシステム [12] の結果である。ソフトウェアの評価尺度は十分長いメッセージでのバイトあたりの処理サイクル数（Cycles/Byte、 C/B と略す)、ハードウェアでの評価は FPGA のスライス数 (slices）と最大動作周波数 (fmax)。ASIC ハードウェア実装の場合はサイズの評価尺度は Gate equivalent (GE) を用いるものとする。その他、これらの公式ベンチマークに当てはまらない注目すべき実装についても適宜報告する。いずれの場合も最適化実装の有無・最適化の度合いにより結果は大きく変わりうるため注意が必要である。著者の所属については提案時点のものである。なお 2016 年 8 月 15 日に CAESAR の 2 次選考結果が発表され、3 次選考の候補が 15 方式に絞られた。3 次選考に残った方式についてはその旨を追記している。

技術分野	認証暗号
名称	ACORN
設計者	Hongjun Wu (南洋理工大学, シンガポール)
発表年 (発表学会等)	2014 (DIAC 2014 [41])
仕様参照先	CAESAR ウェブサイト [6]
特徴	LFSR と単純な非線形処理を利用した方式。 ハードウェア向けのストリーム暗号である Grain や Trivium と類似したシンプルな構造を持つ。 鍵は 128 ビットであり、LFSR 6 つを組合せ 293 ビットを内部状態として保持する。 Grain や Trivium と同様にハードウェアに向いている。 CAESAR 3 次選考候補である。
安全性解析状況	Salam らによる、初期攪拌回数を短くした場合の攻撃と、Nonce 重複を許した場合のフルスペックの解析が存在する [36]。後者は $2^{72.8}$ の計算量と報告されている。そのほかいくつかの解析結果が存在する（例えば [26]）が、いずれも鍵総当たりより大きい計算量、あるいは鍵を既知とした場合の構造解析などであり、直接的な攻撃とはなっていない。 2016 年 12 月に、Roy と Mukhopadhyay により、2^{40} という現実的な計算量における内部状態回復攻撃が提案された [35]。ただしこの結果は ePrint への投稿であり、査読などの検証を未だ経ていないものであることを付記しておく。
主な実装評価結果	(SW) Intel Core i5-6600 (Skylake 3.31 GHz) で 8.46 C/B。 (HW) Virtex 6 で 135 slices、fmax 389 MHz。

技術分野	認証暗号
名称	ASCON
設計者	Christoph Dobraunig、 Maria Eichlseder、 Florian Mendel、 Martin Schlaffer (グラーツ工科大学, オーストリア)
発表年 (発表学会等)	2014 (DIAC 2014 [14])
仕様参照先	CAESAR ウェブサイト [6]、 および公式ウェブサイト `http://ascon.iaik.tugraz.at`
特徴	SHA-3 ハッシュ関数で導入された Sponge 構造をもつ。利用モードは MonkeyDuplex [13] をベースとする。 内部の暗号学的置換は 320 ビット幅を持つ。5 ビット S-box ベースであり、Bitslice 実装を意識した構造である。ソフト・ハード両面で軽量性を謳っている。 CAESAR 3 次選考候補である。
安全性解析状況	内部の置換の段数短縮に対する提案者らによる評価が存在する [15]。初期撹拌を 5 段 (フルスペックは 12 段) にした場合、現実的な計算量 $O(2^{35})$ で鍵導出が可能、 同様に 6 段であれば $O(2^{66})$ の攻撃計算量と報告されている。フルスペックに対する攻撃報告はない。
主な実装評価結果	(SW) Intel Core i5-6600 (Skylake 3.31 GHz) で 11.51 C/B。 (HW) Virtex 6 で 413 slices、fmax 347.0 MHz。 ASIC 実装、I/O 回路含め 7,950 GE、 5.5 Mbps [16]。

技術分野	認証暗号
名称	AES-JAMBU
設計者	Hongjun Wu、 Tao Huang (南洋理工大学, シンガポール)
発表年 (発表学会等)	2014 (DIAC 2014 [42])
仕様参照先	CAESAR ウェブサイト [6]
特徴	ブロック暗号利用モードである。ブロック暗号として AES-128 と SIMON [4] を採用している。SIMON はブロックサイズ/鍵長 (ビット) で 64/96、 96/96、 128/128 の 3 バージョンを指定。 ブロック暗号の入出力以外にブロックサイズの半分を状態変数として用いてシリアルに処理を行う。ブロック暗号 1 回ごとにブロックサイズの半分の暗号化を行う。状態変数のサイズが小さいため小規模ハードウェアに向いている。 CAESAR 3 次選考候補である。
安全性解析状況	一般的な暗号利用モードとは異なり、ブロック暗号の計算量的安全性に基づく安全性帰着を提案者は示していない。 提案者の主張では k ビット鍵、$2n$ ビット ブロック暗号のときに、暗号化の安全性で k ビット、 認証の安全性で n ビット としている。Peyrin ら [31] により、Nonce-misuse シナリオにおいて $2^{n/2}$ 回の暗号化による攻撃と、Nonce-respecting シナリオにおける CCA2 安全性 [5] に対する計算量 $2^{3n/2}$ の攻撃が報告されている。
主な実装評価結果	(SW) AES-128 利用、 Intel Core i5-6600 (Skylake 3.31 GHz) で 5.71 C/B。 (HW) Virtex 6 で 453 slices、fmax 209.8 MHz。

技術分野	認証暗号
名称	AES-OTR
設計者	峯松　一彦 (NEC)
発表年 (発表学会等)	2014 (Eurocrypt 2014 [28])
仕様参照先	CAESAR ウェブサイト [6]、公式ウェブサイト `http://jpn.nec.com/rd/crl/code/research/otr_jp.html/`
特徴	ブロック暗号利用モードである。 CAESAR 提案は AES を利用している。 OCB と類似した構造を持つ。2 ラウンドの Feistel 置換を用いており、ほぼ暗号化のみの計算量で処理が可能である。並列処理も可能。OCB と異なり、認証暗号としての復号処理も AES 暗号化関数のみで実行可能であり、AES 復号を用いない。 CAESAR 3 次選考候補である。
安全性解析状況	提案論文にて、OTR の安全性がブロック暗号の擬似ランダム性 (Pseudorandomness) へ帰着可能なことが示されている。 n ビットブロック暗号の利用において $n/2$ ビットの証明可能安全性を有する。 なお Bost ら [11] により内部のマスク生成における安全性証明との齟齬が指摘され、提案者の修正版が提案されている。
主な実装評価結果	(SW) AES-128 利用、 Intel Core i5-6600 (Skylake 3.31 GHz) で 0.68 C/B。 (HW) Virtex 6 で 1,385 slices、fmax 256.9 MHz。 ARMv7 実装 [29]：1GHz Cortex-A8 マイコンボード 上で 23.5 C/B (42.5 MByte/sec)。 Banik らによる ASIC 実装 [3]：部分的に外部メモリ利用、入力長の制約など加えた特殊条件下で実装し 6,000 GE 台
利用実績等	NEC ソリューションイノベータ 車載系組み込みソリューション（試作） http://www.nec-solutioninnovators.co.jp/sl/emb/automotive/

技術分野	認証暗号
名称	CLOC and SILC
設計者	岩田　哲（名古屋大学, 日本)、峯松　一彦（NEC)、Jian Guo （南洋理工大学, シンガポール)、森岡　澄夫（NEC ヨーロッパ)、小林　栄太（NEC）
発表年 (発表学会等)	2014 (FSE 2014 [18]、 DIAC 2014 [19])
仕様参照先	CAESAR ウェブサイト [6]、公式ウェブサイト `http://www.nuee.nagoya-u.ac.jp/labs/tiwata/AE/`
特徴	ブロック暗号利用モードである。 CFB と CBC-MAC をベースにしたレート 1/2 の方式。 鍵以外に必要なメモリ量が小さいのが特徴（n ビット ブロック暗号利用で約 $2n$ bits)。CLOC は処理のオーバーヘッドを削減し短い入力での性能向上を狙っており、組み込みソフトウェア向き。SILC は CLOC の処理を簡素化したハードウェア向けの方式である。 CAESAR 3 次選考候補である。 CLOC、SILC ともに 128 ビットブロック暗号として AES を採用。 64 ビットブロック暗号として CLOC は TWINE [39] を採用。 SILC は PRESENT [10] および LED [17] を採用。
安全性解析状況	提案論文にて、CLOC と SILC の安全性が用いるブロック暗号の疑似ランダム性に帰着可能であることが示されている。n ビットブロック暗号を用いたとき $n/2$ ビットの安全性を有する。 Nonce を誤って暗号化で重複させた場合でも暗号文の改ざんに対する安全性が保証されている。
主な実装評価結果	(SW) Intel Core i5-6600 (Skylake 3.31 GHz) で CLOC が 2.82 C/B。 SILC は 2.78 C/B。 (HW) Virtex 6、 CLOC が 891 slices、fmax 280.9 MHz。SILC が 989 slices、fmax 280.7 MHz。 CLOC の 8 ビット マイコン実装 [18]: AVR ATmega128 (16 Mhz)。 初期化に 2,000 サイクル、32 バイト暗号化に 550 C/B。 Banik らによる ASIC 実装 [3]：部分的に外部メモリ利用、入力長の制約など加えた特殊条件下で実装し、 CLOC-AES、 SILC-AES ともに約 3,100 GE。

技術分野	認証暗号
名称	Deoxys
設計者	Jérémy Jean、 Ivica Nikolić、 Thomas Peyrin (南洋理工大学, シンガポール)
発表年 (発表学会等)	2014 (DIAC 2014 [22]、 Asiacrypt 2014 [23])
仕様参照先	CAESAR ウェブサイト [6]、公式ウェブサイト `http://www1.spms.ntu.edu.sg/~syllab/Deoxys/`
特徴	専用 tweakable ブロック暗号 Deoxys-BC を利用するブロック暗号利用モード。 Deoxys-BC は 128 ビットブロック、 256 ビット tweak+key、 ラウンド関数は AES そのものであり、段数は 14 から 16 のいずれかをとる。 ブロック暗号利用モードは TAE [27] と SCT [30] の 2 種類がある。TAE モードを用いる場合は 128 ビット安全性を有する。 TAE モードでは OCB 同様の実装面の特徴を有し、レート 1 での並列処理が可能である。一方の SCT は 2 パス、レート 1/2 のオフライン処理だが、SCT モードが Deterministic AE (あるいは Misuse-resistant AE) [34] の機能を有することにより、Nonce の重複に対する安全性を持つ。 CAESAR 3 次選考候補である。
安全性解析状況	提案書 [20] にて Deoxys-BC の安全性に帰着可能であり、TAE モードで 128 ビット 安全性を持つことが示されている。SCT モードの安全性は [30] にて証明されている。
主な実装評価結果	(SW) Intel Core i5-6600 (Skylake 3.31 GHz) で 0.87 C/B。 (HW) Virtex 6 で 993 slices、fmax 330 MHz。 Deoxys-BC 単体の ASIC 実装が 2,860 GE [24]。

技術分野	認証暗号
名称	Joltik
設計者	Jérémy Jean、 Ivica Nikolić、 Thomas Peyrin (南洋理工大学, シンガポール)
発表年 (発表学会等)	2014 (DIAC 2014 [22]、 Asiacrypt 2014 [23])
仕様参照先	CAESAR ウェブサイト [6]、公式ウェブサイト `http://www1.spms.ntu.edu.sg/~syllab/Joltik/`
特徴	専用 tweakable ブロック暗号 Joltik-BC を利用。 Joltik-BC は 64 ビットブロック、 128 ビット tweak+key、 ラウンド関数は 4 ビット S-box を用いた SPN 構造、段数は 24 から 32 のいずれかをとる。 Deoxys 同様、モードは TAE と SCT の 2 種類である。
安全性解析状況	提案書 [21] にて Joltik-BC の安全性に帰着可能、TAE モードで 64 ビット 安全性を有することが示されている。
主な実装評価結果	(SW) Intel Core i5-6600 (Skylake 3.31 GHz) で 13.32 C/B。 (HW) Virtex 6 で 494 slices、fmax 430 MHz。

技術分野	認証暗号
名称	Ketje
設計者	Guido Bertoni[1]、Joan Daemen[1]、 Michael Peeters[2]、Gilles Van Assche[1]、 Ronny Van Keer[1] (1: STMicroelectronics、2: NXP Semiconductors)
発表年 (発表学会等)	2014 (DIAC 2014 [9])
仕様参照先	CAESAR ウェブサイト [6]、 および公式ウェブサイト `http://ketje.noekeon.org/`
特徴	Sponge 構造を持つ。利用モードは MonkeyDuplex [13] がベースとなる。 内部の暗号学的置換は Keccak-p と呼ばれ、SHA-3 関数で用いられる Keccak-f 置換 [8] をベースとしたものである。200 ビット幅の置換を利用するものを Ketje-JR、400 ビット幅のものを Ketje-SR と呼ぶ。 メモリサイズの小ささと計算量の少なさによる、ハード・ソフト両面での軽量性を謳っている。 CAESAR 3 次選考候補である。
安全性解析状況	内部の暗号学的置換が公開ランダム置換とした場合の安全性証明が存在する [13]。 Keccak-p 置換に対しては、提案者らはベースとする Keccak-f の安全性評価結果の大部分が引き継げるため安全としている。
主な実装評価結果	(SW) Ketje-SR、 Intel Core i5-6600 (Skylake 3.31 GHz) で 42.57 C/B。 (HW) Virtex 6 で 456 slices、fmax 229.5 MHz。

技術分野	認証暗号
名称	Minalpher
設計者	佐々木 悠 [1]、藤堂 洋介 [1]、青木 和麻呂 [1]、内藤 祐介 [2]、 菅原 健 [2]、村上 ユミコ [2]、松井 充 [2]、廣瀬 勝一 [3] (1: NTT, 日本、2: 三菱電機, 日本、3: 福井大学, 日本)
発表年 (発表学会等)	2014 (DIAC 2014 [38])
仕様参照先	CAESAR ウェブサイト [6]、公式ウェブサイト `http://info.isl.ntt.co.jp/crypt/minalpher/index-j.html/`
特徴	256 ビットの暗号学的置換 Minalpher-P を用いた専用 256 ビット Tweakable Even-Mansour ブロック暗号 (TEM) を利用、モードは独自方式である。 TEM が用いる内部の置換は 4 ビット S-box 利用の SPN 構造、暗号化・復号関数の統合が容易となる構造を採用している。Nonce の重複に対し部分的な安全性を有する。
安全性解析状況	TEM ブロック暗号の安全性に帰着可能であり、128 ビット 安全性を持つことが示されている。 Minalpher-P は全部で 17.5 段の構造であるが、 安全性評価として、CAESAR 提案書 [37] にて 5.5 段まで短縮したときに攻撃が可能であることが示されている
主な実装評価結果	(SW) Intel Core i5-6600 (Skylake 3.31 GHz) で 5.81 C/B。 (HW) Virtex 6 で 1,104 slices、fmax 280.9 MHz。 SIMD 実装 [37] : Intel CPU Core i7 (Haswell) で 5.6 C/B。 8 ビット RL78 マイコン実装 [37]: 510 ROM、214 RAM バイトの利用で約 2,800 C/B、1275 ROM、470 RAM バイトの利用で 514 C/B。

技術分野	認証暗号
名称	OCB
設計者	Ted Krovetz (カリフォルニア州立大学, 米国)、 Phillip Rogaway (カリフォルニア大学デービス校, 米国)
発表年 (発表学会等)	2001 (ACM CCS 2001 [33])、2004 (ASIACRYPT 2004 [32])、2011 (FSE 2011 [25])
仕様参照先	CAESAR ウェブサイト [6]、および公式ウェブサイト `http://web.cs.ucdavis.edu/~rogaway/ocb/`
特徴	ブロック暗号利用モードである。 AES を利用したバージョンが RFC 7253 にて規定されている。CAESAR 提案は RFC 7253 と同じである。ECB モードと類似した構造だが、メッセージ認証は平文ブロックのチェックサム、実際は排他的論理和をとり、これを暗号化するのみで実現しており、全体の計算量はほぼ暗号化のみの計算量と同等である。さらにブロックごとの並列処理が可能である。 基本的な構造は 2001 年に提案されており、マスク生成の違いなどで後に複数のバージョンが提案されている。 AES を用いるケースでは、特に AES-NI 命令が利用可能な CPU において顕著な高速性を有する。 CAESAR 3 次選考候補である。
安全性解析状況	提案論文 [33, 32, 25] にて、OCB の安全性がブロック暗号の強擬似ランダム性 (Strong Pseudorandomness) へ帰着可能なことが示されている。n ビットブロック暗号の利用において $n/2$ ビットの証明可能安全性を有する。
主な実装評価結果	(SW) AES-128 利用、 Intel Core i5-6600 (Skylake 3.31 GHz) で 0.64 C/B。 (HW) Virtex 6 で 1,348 slices、fmax 292.7 MHz。 その他多様な CPU 上での実装結果が報告されている [25]。
標準化状況	RFC 7253

技術分野	認証暗号
名称	PRIMATEs
設計者	Elena Andreeva[1]、 Begul Bilgin[1]、Andrey Bogdanov[2]、 Atul Luykx[1]、Florian Mendel[3]、Bart Mennink[1]、Nicky Mouha[1]、Qingju Wang[1]、Kan Yasuda[4] (1: ルーヴェンカトリック大学, ベルギー、 2: デンマーク工科大学, デンマーク、3: グラーツ工科大学, オーストリア、4: NTT, 日本)
発表年 (発表学会等)	2014 (DIAC 2014 [1]、 FSE 2014 [2])
仕様参照先	CAESAR ウェブサイト [6]、公式ウェブサイト `http://primates.ae/`
特徴	Sponge 構造をもつ。具体的には、それぞれ異なる利用モードを持つ HANUMAN、GIBBON、APE の 3 つの方式で構成される。いずれの方式も 200 ないし 280 ビットの置換を内部要素とし、この置換を公開ランダム置換と仮定した場合に 80 ビットないし 120 ビットセキュリティを持つことが保証されている。内部の置換は AES ないし Rijndael と類似した SPN だが S-box は 5 ビットである。HANUMAN、 GIBBON はそれぞれ既知の利用モード (SpongeWrap, MonkeyWrap) をベースとするが、APE は Nonce 重複など考慮した独自の利用モードである。
安全性解析状況	HANUMAN に対して、Associated Data がないときの処理の問題点を利用した現実的な偽造作成攻撃が報告されている [40]。 提案者による修正が提案されている。
主な実装評価結果	GIBBON について、 (SW) Intel Core i5-6600 (Skylake 3.31 GHz) で 1,712 C/B。 (HW) Virtex 6 で 419 slices、fmax 333.4 MHz。

参考文献

[1] Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mendel, F., Mennink, B., Mouha, N., Wang, Q., Yasuda, K.: CAESAR candidates PRIMATES. DIAC - Directions in Authenticated Ciphers (2014), http://2014.diac.cr.yp.to/

[2] Andreeva, E., Bilgin, B., Bogdanov, A., Luykx, A., Mennink, B., Mouha, N., Yasuda, K.: APE: Authenticated Permutation-Based Encryption for Lightweight Cryptography. In: FSE. Lecture Notes in Computer Science, vol. 8540, pp. 168–186. Springer (2014)

[3] Banik, S., Bogdanov, A., Minematsu, K.: Low-area hardware implementations of CLOC, SILC and AES-OTR. In: HOST. pp. 71–74. IEEE Computer Society (2016)

[4] Beaulieu, R., Shors, D., Smith, J., Treatman-Clark, S., Weeks, B., Wingers, L.: The SIMON and SPECK Families of Lightweight Block Ciphers. Cryptology ePrint Archive, Report 2013/404 (2013), http://eprint.iacr.org/2013/404

[5] Bellare, M., Desai, A., Jokipii, E., Rogaway, P.: A Concrete Security Treatment of Symmetric Encryption. In: FOCS. pp. 394–403. IEEE Computer Society (1997)

[6] Bernstein, D.J.: CAESAR: Competition for Authenticated Encryption: Security, Applicability, and Robustness, http://competitions.cr.yp.to/caesar.html/

[7] Bernstein, D.J.: eBACS: ECRYPT Benchmarking of Cryptographic Systems, http://bench.cr.yp.to/results-caesar.html/

[8] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V.: The Keccak reference (2011), http://keccak.noekeon.org/

[9] Bertoni, G., Daemen, J., Peeters, M., Assche, G.V., Keer, R.V.: CAESAR candidates Ketje + Keyak. DIAC - Directions in Authenticated Ciphers (2014), http://2014.diac.cr.yp.to/

[10] Bogdanov, A., Knudsen, L.R., Leander, G., Paar, C., Poschmann, A., Robshaw, M.J.B., Seurin, Y., Vikkelsoe, C.: PRESENT: An Ultra-Lightweight Block Cipher. In: CHES. Lecture Notes in Computer Science, vol. 4727, pp. 450–466. Springer (2007)

[11] Bost, R., Sanders, O.: Trick or Tweak: On the (In)security of OTR' s Tweaks. Cryptology ePrint Archive, Report 2016/234 (2016), http://eprint.iacr.org/2016/234

[12] Cryptographic Engineering Research Group at George Mason University: ATHENa: Automated Tools for Hardware EvaluatioN, https://cryptography.gmu.edu/athena/

[13] Daemen, J.: Permutation-based encryption, authentication and authenticated encryption. DIAC - Directions in Authenticated Ciphers (2012), http://hyperelliptic.org/DIAC/

[14] Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: CAESAR candidates ASCON. DIAC - Direc-

tions in Authenticated Ciphers (2014), http://2014.diac.cr.yp.to/

[15] Dobraunig, C., Eichlseder, M., Mendel, F., Schläffer, M.: Cryptanalysis of Ascon. In: CT-RSA. Lecture Notes in Computer Science, vol. 9048, pp. 371–387. Springer (2015)

[16] Groß, H., Wenger, E., Dobraunig, C., Ehrenhofer, C.: Suit up! - Made-to-Measure Hardware Implementations of ASCON. In: DSD. pp. 645–652. IEEE Computer Society (2015)

[17] Guo, J., Peyrin, T., Poschmann, A., Robshaw, M.J.B.: The LED Block Cipher. In: CHES. Lecture Notes in Computer Science, vol. 6917, pp. 326–341. Springer (2011)

[18] Iwata, T., Minematsu, K., Guo, J., Morioka, S.: CLOC: Authenticated Encryption for Short Input. In: FSE. Lecture Notes in Computer Science, vol. 8540, pp. 149–167. Springer (2014)

[19] Iwata, T., Minematsu, K., Guo, J., Morioka, S., Kobayashi, E.: CAESAR candidates SILC. DIAC - Directions in Authenticated Ciphers (2014), http://2014.diac.cr.yp.to/

[20] Jean, J., Nikolić, I., Peyrin, T.: Deoxyx v1.3, http://competitions.cr.yp.to/caesar-submissions.html/

[21] Jean, J., Nikolić, I., Peyrin, T.: Joltik v1.3, http://competitions.cr.yp.to/caesar-submissions.html/

[22] Jean, J., Nikolić, I., Peyrin, T.: CAESAR candidates DEOXYS + Joltik. DIAC - Directions in Authenticated Ciphers (2014), http://2014.diac.cr.yp.to/

[23] Jean, J., Nikolic, I., Peyrin, T.: Tweaks and Keys for Block Ciphers: The TWEAKEY Framework. In: ASIACRYPT (2). Lecture Notes in Computer Science, vol. 8874, pp. 274–288. Springer (2014)

[24] Jean, J., Nikolić, I., Peyrin, T.: Deoxys and Joltik. DIAC - Directions in Authenticated Ciphers (2015), http://www1.spms.ntu.edu.sg/~diac2015/

[25] Krovetz, T., Rogaway, P.: The Software Performance of Authenticated-Encryption Modes. In: FSE. Lecture Notes in Computer Science, vol. 6733, pp. 306–327. Springer (2011)

[26] Lafitte, F., Lerman, L., Markowitch, O., Heule, D.V.: SAT-based cryptanalysis of ACORN. Cryptology ePrint Archive, Report 2016/521 (2016), http://eprint.iacr.org/2016/521

[27] Liskov, M., Rivest, R.L., Wagner, D.: Tweakable Block Ciphers. In: Yung, M. (ed.) CRYPTO. Lecture Notes in Computer Science, vol. 2442, pp. 31–46. Springer (2002)

[28] Minematsu, K.: Parallelizable Rate-1 Authenticated Encryption from Pseudorandom Functions. In: EUROCRYPT. Lecture Notes in Computer Science, vol. 8441, pp. 275–292. Springer (2014)

[29] Minematsu, K., Shigeri, M., Kubo, H.: AES-OTR v2. DIAC - Directions in Authenticated Ciphers (2015), http://www1.spms.ntu.edu.sg/~diac2015/

[30] Peyrin, T., Seurin, Y.: Counter-in-Tweak: Authenticated Encryption Modes for Tweakable Block Ciphers. In: Robshaw, M., Katz, J. (eds.) Advances in Cryptology - CRYPTO 2016 - 36th Annual International Cryptology Conference, Santa Barbara, CA, USA, August 14-18, 2016, Proceedings, Part I. Lecture Notes in Computer Science, vol. 9814, pp. 33–63. Springer (2016), http://dx.doi.org/10.1007/978-3-662-53018-4_2

[31] Peyrin, T., Sim, S.M., Wang, L., Zhang, G.: Cryptanalysis of JAMBU. In: FSE. Lecture Notes in Computer Science, vol. 9054, pp. 264–281. Springer (2015)

[32] Rogaway, P.: Efficient Instantiations of Tweakable Blockciphers and Refinements to Modes OCB and PMAC. In: ASIACRYPT. Lecture Notes in Computer Science, vol. 3329, pp. 16–31. Springer (2004)

[33] Rogaway, P., Bellare, M., Black, J., Krovetz, T.: OCB: a block-cipher mode of operation for efficient authenticated encryption. In: ACM Conference on Computer and Communications Security. pp. 196–205. ACM (2001)

[34] Rogaway, P., Shrimpton, T.: A Provable-Security Treatment of the Key-Wrap Problem. In: Vaudenay, S. (ed.) EUROCRYPT. Lecture Notes in Computer Science, vol. 4004, pp. 373–390. Springer (2006)

[35] Roy, D., Mukhopadhyay, S.: Some results on ACORN. Cryptology ePrint Archive, Report 2016/1132 (2016), http://eprint.iacr.org/2016/1132

[36] Salam, M.I., Bartlett, H., Dawson, E., Pieprzyk, J., Simpson, L., Wong, K.K.H.: Investigating Cube Attacks on the Authenticated Encryption Stream Cipher ACORN. Cryptology ePrint Archive, Report 2016/743 (2016), http://eprint.iacr.org/2016/743

[37] Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M., Hirose, S.: Minalpher v1.1, http://competitions.cr.yp.to/caesar-submissions.html/

[38] Sasaki, Y., Todo, Y., Aoki, K., Naito, Y., Sugawara, T., Murakami, Y., Matsui, M., Hirose, S.: CAESAR candidates Minalpher. DIAC - Directions in Authenticated Ciphers (2014), http://2014.diac.cr.yp.to/

[39] Suzaki, T., Minematsu, K., Morioka, S., Kobayashi, E.: TWINE : A Lightweight Block Cipher for Multiple Platforms. In: Selected Areas in Cryptography. Lecture Notes in Computer Science, vol. 7707, pp. 339–354. Springer (2012)

[40] Vizár, D.: Ciphertext Forgery on HANUMAN. Cryptology ePrint Archive, Report 2016/697 (2016), http://eprint.iacr.org/2016/697

[41] Wu, H.: CAESAR candidates Acorn + MORUS. DIAC - Directions in Authenticated Ciphers (2014), http://2014.diac.cr.yp.to/

[42] Wu, H., Huang, T.: CAESAR candidates AEGIS + Jambu. DIAC - Directions in Authenticated Ciphers (2014), http://2014.diac.cr.yp.to/

CRYPTREC
暗号技術ガイドライン (軽量暗号)
[CRYPTREC-GL-0001-2016-J]
不許複製　禁無断転載

発行日：2017 年 6 月 30 日 (第 1 版)

発行者

- 〒184-8795
  東京都小金井市貫井北町四丁目 2 番 1 号
  国立研究開発法人 情報通信研究機構
  (サイバーセキュリティ研究所 セキュリティ基盤研究室)
  NATIONAL INSTITUTE OF INFORMATION AND COMMUNICATIONS TECHNOLOGY
  4-2-1 NUKUI-KITAMACHI, KOGANEI
  TOKYO, 184-8795 JAPAN
- 〒113-6591
  独立行政法人 情報処理推進機構
  (技術本部 セキュリティセンター 暗号グループ)
  INFORMATION-TECHNOLOGY PROMOTION AGENCY, JAPAN
  2-28-8 HONKOMAGOME, BUNKYO-KU
  TOKYO, 113-6591 JAPAN