暗号アルゴリズム「ＭＡＲＳ」
詳細評価（攻撃評価）レポート

２００１年1月１２日

目次

１．ＭＡＲＳの概要

ＭＡＲＳは Carolynn Burwick、Don Coppersmith、Edward D'Avignon、Rosario Gennaro、Shai Halevi、Charanjit Jutla、Stephen M. Matyas Jr.、Luke O'Connor、Mohammad Peyravian、David Safford、Nevenko Zunic により１９９７年に設計された１２８ビットブロック暗号であり、ＡＥＳの最終審査対象５候補のひとつに選定されている。鍵サイズは１２８ビット以上の可変長をサポートしている。

ＭＡＲＳは比較的複雑なアルゴリズム構成をとっている。データの暗号化の流れは、鍵の whitening に引き続いて、８段の unkeyed forward mixing、８段の keyed forward core、８段の keyed backward core、８段の unkeyed backward mixing、鍵の whitening となっている。またこれらの中で利用されるコンポーネントもさまざまである。unkeyed mixing rounds では、８ビット入力３２ビット出力の S-boxes と算術加算と排他的論理和演算が用いられている。keyed core rounds ではこれに加えて、鍵との３２ビット乗算、鍵との３２ビット加算ならびにデータ依存回転シフト演算が用いられる。

これらの演算は３２ビットビット単位のソフトウエア実装に適している。一方３２ビット乗算やデータ依存回転シフトの利用は、プラットフォームによっては速度の低下をまねくことがある。特にハードウエアでは、MARS のスループット（平文入力から暗号文出力までの時間）は非常に低速であり、しかも実装面積が大きくなる。このようにＭＡＲＳは結果的にＡＥＳの標準プラットフォームであった３２ビットプロセッサにターゲットを絞った暗号であるということができる。

なお、ＭＡＲＳはＡＥＳ投稿後に一度アルゴリズムの変更をおこなっている。これは変更前の版に等価鍵が存在したこと、またＩＣカード実装には向かないことに対応したもので、例えば必要ＲＡＭサイズを削減する工夫がなされた。これからみてもＭＡＲＳはその設計時にはＩＣカードでの実装が意図されていなかったことがわかる。さらに最近になって、ＭＡＲＳの S-boxes は設計者が設定した設計基準を満たしていないことが発見されている。この事実がＭＡＲＳのセキュリティに与える影響は少ないようであるが、設計ミスであったことは否定できない。

ＭＡＲＳはＡＥＳ候補として、その安全性についてはこれまで数多くの研究がなされてきた。そこでここでは安全性の評価項目ごとに、設計者の評価内容とこれまでの研究結果をふまえて、評価者の考察をのべることとする。

２．ＭＡＲＳの安全性評価

２．１　差分解読法に対する安全性

【自己評価書における記述】

自己評価書（技術仕様書）には、詳細にＭＡＲＳの差分解読法に対する強度評価の検討がなされている。これによればＭＡＲＳの差分特性確率は以下のとおりである。

ＭＡＲＳの差分特性確率　　2^{-196}以下

ここから類推すると、ＭＡＲＳを差分解読法で解読するの必要な平文数が理論上得ることの出来る平文数を超えることになり、したがってＭＡＲＳは十分安全であると結論を導き出している。

【考察】

ＭＡＲＳは比較的複雑なアルゴリズムであり、その解析も容易ではないため、自己評価書（技術仕様書）に書かれた内容は完全には遠いとはいえ、信頼に足ると考えられる。

なおこの評価結果は、完全な形での差分確率の上からの評価(いわゆる差分解読法に対するprovable security)を実現したものではないことに注意する必要がある。すなわち、真の差分確率の最大値はこの評価結果よりも大きくなり、したがって解読に必要な選択平文数は実際にはこれより少ない可能性が残っている。しかしながら、一般に与えられたブロック暗号アルゴリズムの差分確率の最大値を求めることはきわめて困難であり、差分確率の最大値が求められていないことを理由に評価が不足しているということはできない。

２．２　線形解読法に対する安全性

【自己評価書における記述】

自己評価書（技術仕様書）には、詳細にＭＡＲＳの線形解読法に対する強度評価の検討がなされている。これによればＭＡＲＳの線形バイアスは以下のとおりである。

ＭＡＲＳの線形バイアス　　2^{-69}以下

ここから類推すると、ＭＡＲＳを線形解読法で解読するの必要な平文数が理論上得ることの出来る平文数を超えることになり、したがってＭＡＲＳは十分安全であると結論を導き出している。

【考察】

ＭＡＲＳは比較的複雑なアルゴリズムであり、その解析も容易ではないため、自己評価書（技術仕様書）に書かれた内容は完全には遠いとはいえ、信頼に足ると考えられる。

なおこの評価結果は、完全な形での線形確率の上からの評価(いわゆる線形解読法に対するprovable security)を実現したものではないことに注意する必要がある。すなわち、真の線形確率の最大値はこの評価結果よりも大きくなり、したがって解読に必要な既知平文数は実際にはこれより少ない可能性が残っている。しかしながら、一般に与えられたブロック暗号アルゴリズムの線形確率の最大値を求めることはきわめて困難であり、線形確率の最大値が求められていないことを理由に評価が不足しているということはできない。

２．３　差分線形解読法に対する安全性

【自己評価書における記述】

記述なし。

【考察】

差分線形解読法とは、アルゴリズムの前半を差分解読法で差分の波及を考え、後半では線形近似を考えることによって解読に結びつけるといったもので、８段版のＤＥＳにおいて良好な結果が得られている。しかしながら差分線形解読法に必要な選択平文数は、前半でえられた差分確率の２乗および後半でえられた線形偏差の４乗に反比例してしまう。したがって、前半の差分確率および後半の線形偏差がそれぞれ非常に大きな（１に近い）値でなければ、差分線形解読法の優位性はあらわれない。このため通常、差分線形解読法は、段数の少ないアルゴリズムでのみ成立する。このような理由から、ＭＡＲＳに対する差分線形解読法が、差分解読法や線形解読法よりも効果があるとは考えられない。

２．４　高階差分解読法／補間攻撃に対する安全性

【自己評価書における記述】

記述なし。

【考察】

いかなるブロック暗号においても、平文と暗号文と鍵（あるいは副鍵）との関係を代数式(algebraic form)であらわすことが理論上可能である。そしてその代数式が簡易であればこの式を数学的に解くことによって解読できる可能性がある。これが、高階差分解読法や補間攻撃とよばれる解読法の基本的な考え方である。すなわち、これらの解読法が成立するためには、まず暗号アルゴリズムを記述する代数式が、明示的な形で書き下せる程度の複雑さであるかどうかが、第１の関門となる。

普通これらの解読法が成立するのは、「数学的に美しく」あるいは「簡単な論理式で」作られた参照テーブルと、論理演算だけから構成されるような暗号アルゴリズムの場合である。これに対してＭＡＲＳには、３２ビット乗算、データ依存回転シフト、排他的論理和演算が含まれている。このように、算術演算と論理演算を組み合わせた暗号アルゴリズムを代数式で書き下すのは、式の項数が多すぎて困難である。したがって自己評価書にあるように、ＭＡＲＳは高階差分解読法や補間攻撃で解読されるとは考えにくい。

２．５　短縮差分解読法に対する安全性

【自己評価書における記述】

記述なし。

【考察】

短縮差分解読法とは、特性(characteristic)の遷移状態をビットよりも大きい単位で（例えばバイト単位で）扱うことによって、差分(differentials)を効率よく求め、これにより通常の差分解読法よりも効率のよい解読をめざすものである。この解読法が成立するためには、アルゴリズムの主要な部分がこの単位を基本とする演算で構成されており、しかも１ブロック内におけるこの単位の数が多いなど、強い構造をもつことが必要である。

現在知られているＭＡＲＳの攻撃は、その名称は異なっていても、多かれ少なかれ短縮差分解読法のバリエーションである。ここではＭＡＲＳの安全性評価に関する２つの主要な論文の結果を示す。いずれも Bruce Schneier らの論文からの引用である。

発表先	Rounds	必要平文数	必要メモリー	必要ステップ数
ＦＳＥ2000	11 Core	2^{65}	2^{70}	2^{229}
第３回AES会議	16 Mixed 5 Core	8	2^{236}	2^{232}
	16 Mixed 5 Core	2^{50}	2^{197}	2^{247}
	6 Mixed 6 Core	2^{69}	2^{73}	2^{197}

これらの結果からみても、すでに必要メモリーあるいは必要ステップ数が膨大となっており、現実的には完全仕様のＭＡＲＳの安全性を脅かすにはいたっていないことがわかる。

２．６　不能差分攻撃に対する安全性

【自己評価書の記述】

記述なし。

【考察】

通常の差分解読法では差分特性確率が小さいという性質は安全性の観点から望ましいことであるが、不能差分攻撃では逆に確率０の差分経路がある場合、すなわち絶対に起こらない差分波及パターンが多ラウンドにわたって存在する場合に、このことを手がかりに拡大鍵の情報を絞りこむことができるというものである。例えばラウンド関数が全単射であるような５段のＦｅｉｓｔｅｌ暗号には、ラウンド関数の構造によらず必ず不能差分が存在することが知られている。

ＭＡＲＳ は Feistel とは異なった暗号構造をもっており、その不能差分攻撃に対する強度をしらべることは意味のないことではないと思われる。ただし現時点では ＭＡＲＳ の不能差分について考察された研究は知られていない。直感的には ＭＡＲＳ の一段の複雑さを考えると、１０段以上で不能差分があるとは考えにくいが、正確には今後の研究をまたなければならない。自己評価書の内容も、著者らが不能差分攻撃に対する安全性評価をまだ本格的には行っていないことを示しているように思われる。

２．７　ブーメラン攻撃に対する安全性

【自己評価書における記述】

記述なし。

【考察】

ブーメラン攻撃は、差分解読法を応用した適応的選択暗号文攻撃手法であり、その原理は、ある差分値をもつ選択平文から得られた暗号文をもとに、そこから解読者が適当な差分値をもつ選択暗号文を新たに生成し、それを今度は復号して得られた平文の差分値を観測するというものである。この解読法は暗号アルゴリズムを例えば上半分と下半分に分割した場合に、上半分の最大平均差分確率と下半分の最大平均差分確率がともに非常に大きい場合に成立する。

実際この解読法に必要な平文の数は少なく見積もってもこれらの確率の積の逆数程度になり、この点差分線形解読と似ているともいえる。結論としては、ＭＡＲＳに対してこのブーメランの攻撃が成立するということは考えにくいと言ってよい。なお Bruce Schneier らが行った MARS のアタックで導入した amplified boomerang attack は、実質的には短縮差分解読法の一種であるので、本レポートでは短縮差分解読法の節で示している。

２．８　χ^2攻撃に対する安全性

【自己評価書における記述】

記述なし。

【考察】

χ^2攻撃は、差分解読法や線形解読法のある種の一般化であり、平文の部分情報と暗号文の部分情報との間に成立する統計的相関性を手がかりに解読をおこなうというものである。これらの解読法のフレームワークは非常に一般的なものであるため、与えられた暗号アルゴリズムに対してこれらの解読法に対する安全性を完全な形で証明することは通常困難とされている。ただし現実にはこれらの解読法が効果を生むためには、平文や暗号文を分割して考えることが（解読者にとって）有効となるような特別な構造が暗号アルゴリズムに必要となり、一般には特定のビット長単位だけで演算が行なわれているなどの場合がこれにあたる。

ＭＡＲＳはその構造上３２ビット演算を基本とはしているものの、８ビット入力３２ビット出力の S-Boxes を用いるなど、その演算要素は多種多様で、χ^2攻撃が簡単に成立するとは考えにくい。

２．９　mod n 攻撃に対する安全性

【自己評価書における記述】

記述なし。

【考察】

mod ｎ 攻撃は、暗号アルゴリズムの中間データの分布を mod n で見たときに出現する偏りを手がかりに拡大鍵の情報を推定するという手法である。この解読法は算術演算を多く用いているアルゴリズムにおいて有効になるが、算術演算と論理演算がともに利用されているアルゴリズムでは急速にその効果が失われることが知られている。この mod n 攻撃は特殊な暗号アルゴリズムでのみ成立する解読法と考えたほうがよい。

ＭＡＲＳにおいてはルックアップテーブルが用いられているため、mod n 攻撃は成立しない。

２．１０　弱鍵の存在について

【自己評価書における記述】

（自己評価書（技術仕様書からの引用））
我々が認識している限り、MARS には弱い鍵はない。鍵拡大処理を行うことによって、乗算に使用される鍵ワードが明らかな弱点を持たないようにしており (例えば、鍵ワードは偶数ではない)、弱い鍵のその他の存在要因も認識していない。そのため、我々は鍵の選択にいかなる制約も設けていない。

【考察】

ＭＡＲＳの鍵スケジュール部ならびに暗号化部の構造をみるかぎり、ＤＥＳの弱鍵という意味での弱い鍵が存在するとは考えられない。しかしながら広い意味での弱鍵、すなわち特定の鍵集合を用いたときにある特定の解読法に対して弱くなるという可能性を完全に否定することはできない。例えばＭＡＲＳでは拡大鍵とデータとの乗算演算が含まれているが、このような演算はしばしば鍵の値に依存した強度のばらつきを招くことがある。しかしながら、そのようなばらつきが仮にあったとしても、そのことが直ちに完全版ＭＡＲＳの安全性に大きな影響を与えるとは考えにくい。

２．１１　関連鍵攻撃に対する安全性

【自己評価における記述】

記述なし。

【考察】

関連鍵攻撃は鍵スケジュール部が非常に簡単な暗号アルゴリズムに対して成立する（あるいは意味をもつ）ものであって、鍵スケジュールが複雑なアルゴリズムに対する関連鍵攻撃は、それが仮に可能であったとしてもきわめて作為的な関連性を解読者が与えられなければならなくなり、解読としての意味が薄れてしまうと考えられる。

ＭＡＲＳは、関連鍵攻撃を無効化するに十分な複雑性をもったロジックを鍵スケジュール部にもっている。したがって関連鍵攻撃がＭＡＲＳで成立するということはきわめて考えにくい。

２．１２　スライド攻撃に対する安全性

【自己評価における記述】

記述なし。

【考察】

スライド解読法はデータ暗号化部のラウンドを一段あるいはそれ以上の段数をずらしたものと、もとのアルゴリズムとの間に、拡大鍵を含めた等価なロジックが現れる場合に有効となる解読法であり、一種の関連鍵解読法と考えることができる。この解読法が成立するためには、１段あるいはそれ以上の段数ごとに同じ拡大鍵が周期的に各ラウンド関数に入力されることが必要となる。ＭＡＲＳの鍵スケジュール部はこのような構造をもっていないので、ＭＡＲＳにスライド解読法を適用するのは困難と考えられる。

２．１３　タイミング攻撃に対する安全性

【自己評価における記述】

（自己評価書（技術仕様書）からの引用）
適切な実装を行えば、MARS はタイミング攻撃および差分fault 解読に対して耐性がある。旧式のマシンでは乗算に要する時間が入力によってかなり変わるが、MARS の鍵拡大ルーチンは、乗算が高速になるような鍵 (例えば 0や 1が多く連続している鍵)を排除している。

【考察】

タイミング攻撃とは、暗号鍵などのセキュリティパラメータ値によって暗号の演算時間が異なるような実装が行われた場合に、この演算時間を観測することによってこのセキュリティパラメータの値を逆算する攻撃法である。この攻撃法に対処するためには、セキュリティパラメータの値に依存せずに常に同じ時間で暗号化、復号が完了する実装をする必要がある。

自己評価書には「ＭＡＲＳはタイミング攻撃に耐性がある」と書かれているが、これは書きすぎであると考えられる。実際には「ＭＡＲＳは、不用意な実装ではタイミング攻撃で破られる危険性がある。」ということであり、そのように読まれるべきである。実際、ＭＡＲＳのような乗除算やデータの回転シフトを多用するアルゴリズムでは、その入力値によって演算時間が異なる場合があるため、プラットフォームによっては（特にＩＣカード用プロセッサなど）実装時にこの攻撃法に対する注意が特に必要である。

もちろんこの自己評価書に書かれているように、それを防ぐような実装は可能ではある。しかしながらこの防御のために、速度やプログラムサイズを犠牲にしなければならないことにも同時に注意する必要がある。またタイミング攻撃はソフトウエア実装だけでなく、ハードウエアでの実装でも起こりうる。結論としては、ＭＡＲＳは実装時にタイミング攻撃に注意すべきであり、それを避けるために注意深い実装が要求されるということである。

２．１４　電流解析に対する安全性

【自己評価における記述】

記述なし。

【考察】

電流解析攻撃、差分電流解析攻撃はおもにＩＣカードをターゲットとして、暗号鍵などのセキュリティパラメータに依存して電流消費量が変化することに着目して解読を行うというものである。電流消費量はビットデータの値そのもの（あるいはビットデータが変更されたかどうか）によって変化するということがこの解読法の本質的な点である。この攻撃はＩＣカードの物理特性とアルゴリズムの実装方法に大きく依存するため、自己評価書にあるように暗号アルゴリズムそのものの構造によってあらゆる差分電流解析攻撃を防御するのは困難であると考えるべきであろう。

実際ＡＥＳ候補すべてに対し何らかの差分電流攻撃が成立することが（理論的には）知られている。この攻撃法は現在の暗号研究におけるホットトピックのひとつであり、攻撃と防御の両面から研究が進められている。差分電流解析攻撃の防御方法のうち有力なものとして、FSE2000 で発表された Messerges による “Securing the AES Finalists Against Power Analysis Attacks” があり、ここではＡＥＳ最終選考５候補について、差分電流解析攻撃を防御した実装法が示されている。

しかしながら CHES2000 では、この方法は Rijndael と Serpent 以外のアルゴリズム、すなわち算術演算をもちいた暗号には実は適用できないことが示されており、またおなじ CHES2000 の別論文では、この防御方法そのものも、高次の差分電流解析攻撃には必ずしも有効ではないことが示されているなど、この種の研究はまだ途上であり、いかなる暗号アルゴリズムに対しても差分電流解析攻撃について結論的なことが言える段階ではないと考えられる。

３．　まとめ

ＡＥＳ仕様の（改定後の）ＭＡＲＳについては、理論的なものを含め、鍵の全数探索よりも高速であるようないかなる解読法も発見されていない。その意味では、ＭＡＲＳは現時点では十分安全な暗号アルゴリズムといってよい。しかしながら、暗号理論的あるいは実用的な観点から以下の点に注意すべきである。

（１）　ＭＡＲＳのＳｂｏｘは設計者の意図した通りには作られていなかったことが最近になって判明している。この事実に関する詳細な検討はまだあまり行われていない。このことがＭＡＲＳの安全性に大きく影響することは現状考えにくいが、注意しておく必要がある。

（２）　ＭＡＲＳは３２ビットプロセッサ上のソフトウエアでは優秀な性能を発揮するが、ＩＣカード向けやハードウエア向けには作られていない。ＭＡＲＳを利用する場合にはその利用環境に関する注意が必要である。

（３）　ＭＡＲＳのコンポーネントである、乗算とデータ依存回転シフト演算が、タイミング攻撃や差分電流解析に対してどのように作用するか、すなわちこれらの解読法に対する防御実装にどの程度のコストがかかるかは、いまだ研究途上であるが、その情報はＭＡＲＳを特定の環境で利用する上で重要になる可能性がある。

以上

付録　ＭＡＲＳの仕様　（技術仕様書の抜粋）

2.5　擬似コード

以下では擬似コードを使用してこの暗号を説明する。ここでは、次の表記法を使用する。
この暗号で使用されている演算は、符号のない整数とみなされる 32 ビット・ワードに適用される。各ワードのビットには 0 から 31 まで番号をつける。ここで 0 番目のビットは下位のビット、31 番目のビットは上位のビットである。

2 つのワード、a と b のビットごとの排他的論理和を $a \oplus b$ で表記し、2 つのワードのビットごとの OR (論理和) および AND (論理積) を $a \vee b$ および $a \wedge b$ で示す。また、2^{32} を法とする加算を $a+b$、2^{32} を法とする減算を $a-b$、さらに 2^{32} を法とする乗算を $a \times b$ で示す。

また、$a \lll b$ および $a \ggg b$ は、32 ビット・ワード a を b ビットだけ、それぞれ左と右に巡回シフトすることを示す。左へ b ビット巡回シフトした場合、位置 i のビットは位置 $i+b \bmod 32$ へ移動する (例えば、下位のビットは 0 番目から b 番目の位置へ移動する)。同様に、右へ b ビット巡回シフトした場合は、位置 i のビットは位置 $i-b \bmod 32$ へ移動する。

最後に、$x_1,...x_n$ を 32 ビット・ワードの 1 から n までの各ビットとしたとき、n ビット幅でのスワップ演算を $(x_n,...,x_2,x_1) \leftarrow (x_1,...,x_3,x_2)$ で示す。

例えば、$(D[3],D[2],D[1],D[0]) \leftarrow (D[0],D[3],D[2],D[1])$ は 4 ワード配列 $D[\,]$ を右へ 1 ワード分巡回シフトすることを示している。

注記：　以下に示す擬似コードは図 2 および図 5 とはスタイルがいくらか異なっている。特に、擬似コードを短くするために、8 つの混合ラウンドを 1 つのループで実装している。

E 関数 (入力: $in, key1, key2$)

```
---------------------------------------------
1. // 3 つの仮変数 L 、 M 、 R を使用
2. M = in + key1                  // 第 1 鍵ワードを加算
3. R = (in <<< 13) × key2         // 第 2 鍵ワード (かならず奇数) を掛ける
4. i = M の下位 9 ビット
5. L = S[i]                       // S-box ルックアップ
6. R = R <<< 5
7. r = R の下位 5 ビット          // これらのビットが巡回シフトの量を指定
8. M = M <<< r                    // 1 回目のデータ依存型巡回シフト
9. L = L ⊕ R
10. R = R <<< 5
11. L = L ⊕ R
12. r = R の下位 5 ビット         // これらのビットが巡回シフトの量を指定
13. L = L <<< r                   // 2 回目のデータ依存型巡回シフト
14. 出力 ( L 、 M 、 R )
```

MARS 暗号化 (入力: $D[\,], K[\,]$)

フェーズ 1: 前方混合

1. // 最初に副鍵をデータに加算
2. for $i = 0$ to 3 do
3. $\quad D[i] = D[i] + K[i]$
4. // つぎに前方混合の 8 ラウンドを実行
5. for $i = 0$ to 7 do // $D[0]$ を使用して $D[1]$、$D[2]$、$D[3]$ を変換
6. $\quad$ // S-box ルックアップ 4 回
7. $\quad D[1] = D[1] \oplus S0[\,D[0]\,]$ の下位バイト]
8. $\quad D[1] = D[1] + S1[\,D[0]\,]$ の第 2 バイト]
9. $\quad D[2] = D[2] + S0[\,D[0]\,]$ の第 3 バイト]
10. $\quad D[3] = D[3] \oplus S1[\,D[0]\,]$ の上位バイト]
11. $\quad$ // つぎにソース・ワードを右に巡回シフト
12. $\quad D[0] = D[0] >>> 24$
13. $\quad$ // 続いて追加の混合演算を実行
14. $\quad$ if $i = 0$ or 4 then
15. $\quad\quad D[0] = D[0] + D[3]$ // $D[3]$ をソース・ワードに加算して戻す
16. $\quad$ if $i = 1$ or 5 then
17. $\quad\quad D[0] = D[0] + D[1]$ // $D[1]$ をソース・ワードに加算して戻す
18. $\quad$ // つぎのラウンドのために $D[\,]$ を右へ 1 ワード巡回シフトする
19. $\quad (D[3], D[2], D[1], D[0]) \leftarrow (D[0], D[3], D[2], D[1])$
20. end-for

フェーズ 2: 鍵使用変換

21. // 鍵使用変換の 16 ラウンドを実行
22. for $i = 0$ to 15 do
23. $\quad (out1, out2, out3)$ = E-Function($D[0], K[2i+4], K[2i+5]$)
24. $\quad D[0] = D[0] <<< 13$
25. $\quad D[2] = D[2] + out2$
26. $\quad$ if $i < 8$ then // 最初の 8 ラウンドは前方モード
27. $\quad\quad D[1] = D[1] + out1$
28. $\quad\quad D[3] = D[3] \oplus out3$
29. $\quad$ else // 後半の 8 ラウンドは後方モード
30. $\quad\quad D[3] = D[3] + out1$
31. $\quad\quad D[1] = D[1] \oplus out3$
32. $\quad$ end-if
33. $\quad$ // つぎのラウンドのために $D[\,]$ を右へ 1 ワード巡回シフト
34. $\quad (D[3], D[2], D[1], D[0]) \leftarrow (D[0], D[3], D[2], D[1])$
35. end-for

フェーズ 3: 後方混合

36. // 後方混合の 8 ラウンドを実行
37. for $i = 0$ to 7 do
38. // 追加の後方混合
39. if $i = 2$ or 6 then
40. $D[0] = D[0] - D[3]$ // ソース・ワードから $D[3]$ を減算
41. if $i = 3$ or 7 then
42. $D[0] = D[0] - D[1]$ // ソース・ワードから $D[1]$ を減算
43. // S-box ルックアップ 4 回
44. $D[1] = D[1] \oplus S1[D[0]$ の下位バイト]
45. $D[2] = D[2] - S0[D[0]$ の上位バイト]
46. $D[3] = D[3] - S1[D[0]$ の第 3 バイト]
47. $D[3] = D[3] \oplus S0[D[0]$ の第 2 バイト]
48. // つぎにソース・ワードを左に巡回シフト
49. $D[0] = D[0] <<< 24$
50. // つぎのラウンドのために $D[\,]$ を右へ 1 ワード巡回シフト
51. $(D[3], D[2], D[1], D[0]) \leftarrow (D[0], D[3], D[2], D[1])$
52. end-for
53. // さらにデータから副鍵を減算
54. for $i = 0$ to 3 do
55. $D[i] = D[i] - K[36 + i]$

鍵拡大 (入力： $k[\,], n$、 出力 : $K[\,]$)

---

1. // n は鍵バッファ $k[\,]$, $(4 \le n \le 14)$ のワード数
2. // $K[\,]$ は拡大鍵配列で、40 ワードで構成
3. // $T[\,]$ は一時的配列で、15 ワードで構成
4. // $B[\,]$ は 4 ワードの固定テーブル

5. // $B[\,]$ を初期化する
6. $B[\,] = \{$ 0xa4a8d57b , 0x5b5d193b , 0xc8a8309b , 0x73f9a978 $\}$

7. // $T[\,]$ を鍵データで初期化
8. $T[0..n-1] = k[0..n-1]$, $T[n] = n$, $T[n+1...14] = 0$

9. // 繰返しを 4 回、各回で $K[\,]$ の 10 個のワードを計算
10. for $j = 0$ to 3 do
11. for $i = 0$ to 14 do // 線形変換
12. $T[i] = T[i] \oplus \left(\left(T[i - 7 \bmod 15] \oplus T[i - 2 \bmod 15] \right) <<< 3 \right) \oplus (4i + j)$

13. 4 回繰返す (repeat four times) // 攪拌 4 ラウンド
14. for $i = 0$ to 14 do
15. $T[i] = \left(T[i] + S[T[i - 1 \bmod 15]\text{の下位 9 ビット}] \right) <<< 9$
16. end-repeat

17. for $i = 0$ to 9 do // つぎの 10 個のワードを $K[\,]$ に保存
18. $K[10j + i] = T[4i \bmod 15]$
19. end-for

20. // 乗算鍵を変更
21. for $i = 5,7,...35$ do
22. $j = K[i]$ の下位 2 ビット
23. $w = K[i]$ (ただし下位 2 ビットを 1 にセット)

24. // ビット・マスク M を生成
25. $M_\ell = 1$ ただし w_ℓ が 10 ビット連続の 0 または 1 に属し、
26. さらに $2 \le \ell \le 30$ および $w_{\ell-1} = w_\ell = w_{\ell+1}$ の場合

27. // パターンを固定テーブルから選択して巡回シフト

28. $r = K[i-1]$ の下位 5 ビット // 巡回シフト量
29. $p = B[j] <<< r$

30. // マスク M の制御下で $K[i]$ を p で変更
31. $K[i] = w \oplus (p \wedge M)$
32. end-for