2008 年度版リストガイド
（秘匿の暗号利用モード）

平成 21 年 3 月

独立行政法人情報通信研究機構
独立行政法人情報処理推進機構

目 次

1　秘匿の暗号利用モード

1.1　本文書の位置付け

1.1.1　文書の目的

本文書は、共通鍵ブロック暗号アルゴリズムとともに用いるべき利用モードを記述する。ここでは、5つのデータ秘匿を目的とした利用モードを扱う。ともに用いるべきブロック暗号はCRYPTRECが定めた電子政府推奨暗号リストとし、そのうちであれば任意のものを使うことができる。紹介する5つのモードは、Cipher Block Chaining (CBC)、Cipher Feedback(CFB)、Counter (CTR)、Electronic Codebook (ECB)、Output Feedback (OFB) の各モードである。これらを適切に用いることでデータの秘匿を実現することができる。

1.1.2　対象とする利用目的

情報の秘匿とは、情報へのアクセス権限所持者以外に情報を漏らさないことである。本技術では、暗号化処理により元の情報 (平文と呼ぶ) から暗号文を生成することで、情報の秘匿を実現する。情報へのアクセス可能者は、暗号文を平文に戻すことができるが、その他の者は暗号文から平文のいかなる部分情報も有意に推測することができない。そのため、情報を漏洩することなく情報の記憶領域への保存や通信路への送信が可能となる。

本技術は、ブロック暗号を用いて任意の長さ[1]の平文データを暗号化する方法、ならびにその結果生成された暗号文を復号する方法を提供する。

情報の秘匿を実現するためには、本技術は別途紹介する関連技術とともに適切に実装し、運用する必要がある。これらを怠った場合には、本文書で説明する安全性は達成されない場合がある。

本文書の暗号技術による情報の秘匿には攻撃者による暗号文の改ざんや暗号文のビットエラーを検出する機能はない。攻撃者による改ざんの脅威が考えられ、これらを検出する必要がある場合にはMACを別途利用しなければならない。また、脅威が攻撃によらないビットエラーなどの場合にはチェックサムやエラー訂正符号による検出や訂正機能を暗号文に実施する。これらに違反し、平文にチェックサムなどを実施し、復号後でチェックの結果を攻撃者が知ることができるような場合、そのことが脆弱性となり攻撃者が故意にノイズを挿入し、その改ざん検知の挙動を見ることにより情報が漏洩する場合がある [V02]。

また、本文書の暗号技術では、情報の存在そのものや、情報の長さを秘匿することはできないので、これらを秘匿したい場合には別のメカニズムを利用する必要がある。

1.1.3　本文書の構成

本文書は以下からなる。1.2節で用語や記法、数学的記述などを示し、1.3節で扱う技術について以下の順で解説する: まず、1.3.1節で一般的な枠組みとしてどのようなモデルで利用させるかを示し、次に1.3.2節で関連する他の技術やその運用につ

[1]安全性の理由により、許容されるデータ長に上限 ($b \times 2^{b/2}$ ビット) がある。これは例えば、64ビットブロック暗号の場合32ギガバイト、128ビットブロック暗号の場合256エクサバイト (1エクサバイトは約100万テラバイト) である。

いて示し、最後に 1.3.3 節で本文書で扱う 5 つのモードの違いや選択の方法を示す。そして、1.4 節にて個々のモードの仕様を記述し、1.5 節では動作をより正確に理解するための具体的なデータ例 (テストベクトル) を示す。

1.2 定義

1.2.1 用語定義

ビット 2 値文字、0 または 1 の値をとる。

ビット列 順序つきビットの列。

ビットエラー ビット列に対し、どこかの “0” 値のビットが “1” となること、またその逆。あるいはこれらが複数発生すること。

ブロック 予め決められた長さのビット列。

セグメント CFB モードにおいて、平文データを処理するデータ単位 (ブロック暗号の「ブロック」と異なる場合があるため、混乱なきよう用語を変えている)。

ブロック暗号 固定長の入力 (64 ビットや 128 ビットなど) をとり、同じ長さの出力を生成する関数群が別途鍵入力によりパラメータ化されたもの。暗号として利用するため、入出力から鍵が求まらない、逆関数が存在する、鍵入力なしには出力から入力を有意に推定できない、などの性質がある。

ブロックサイズ ブロック暗号における主となる入出力データのビット数。64 ビット、128 ビットが一般的。

鍵 ブロック暗号の入力ビット列であり、暗号化する者と復号する者のみで共有する秘密情報。

初期値 暗号化における鍵、平文以外のもう一つの入力。

平文 暗号化する前の保護対象のデータ。

暗号文 暗号化により生成されたビット列。

暗号化 保護対象のデータを入力とし、この内容がわからないような別のデータへ変換すること。

復号 暗号化されたデータを、元のデータに変換すること。ただし、暗号文自身がエラーやデータ欠損など変化した場合にはその限りでなく、規定された処理の結果であるデータ (壊れたデータやランダムなもの) が出てくる可能性がある。

利用モード 共通鍵ブロック暗号を如何に利用することで、期待するデータ変換を得るかを規定した方式。

CBC モード　Cipher Block Chaining、暗号文ブロック連鎖モード。

CFB モード　Cipher Feedback、暗号文フィードバックモード。

CTR モード　Counter、カウンタモード。

ECB モード　Electronic Codebook、電子辞書モード。

OFB モード　Output Feedback、出力フィードバックモード。

排他的論理和　繰り上がりのない 2 進数の加算 ($0 \oplus 0 = 0$、$0 \oplus 1 = 1$、$1 \oplus 1 = 0$)。また、本文書では、長さが同じ 2 つのビット列に対する排他的論理和を次のように考える; 同じ位置のビット対で排他的論理和をとり、その結果をその順でならべたものを結果とする (例: $0011 \oplus 1010 = 1001$)。

ナンス　逐次的に生成または入力されるデータを考え、常に生成されるデータは過去に出現したことがない、という性質を持つもの。

1.2.2　記号、および記法の定義

$X \oplus Y$	:	ビット列 X, Y に対するビット毎の排他的論理和。X や Y が不等長の場合、短いほうの文字列の上位に必要な “0” 値があるものとし、下位ビットに揃えて計算するものとする
$X \ll l$	:	長さが明示的なバイナリ文字列 X を上位 (左) へ l ビットシフトする。溢れた上位 l ビットは値が破棄され、下位 l ビットは 0 が挿入される
$\|X\|$	:	ビット列 X のビット長
b	:	内部で利用するブロック暗号のブロック長 (ビット)
$Enc_K(P)$	:	内部で利用するブロック暗号の暗号化関数に鍵 K を使うときの入力データ (平文ブロック)P を処理したときの出力データ (P の K による暗号文ブロック)
$Dec_K(C)$	:	内部で利用するブロック暗号の復号関数に鍵 K を使うときの入力データ (暗号文ブロック)C を処理したときの出力データ (C の K による復号ブロック)
$inc(C)$	:	カウンタ C を規定に従い更新した値
IV	:	暗号利用モードにおける初期値
K	:	内部で利用するブロック暗号にさす鍵
k	:	内部で利用するブロック暗号の鍵長 (ビット)
$\mathrm{msb}_n(X)$	:	ビット列 X の上位 n ビット
n	:	CFB モードが処理するデータ単位 (セグメント) の長さ (ビット)

1.3　技術概要

本技術は、情報の秘匿を実現する暗号化処理のための技術である。

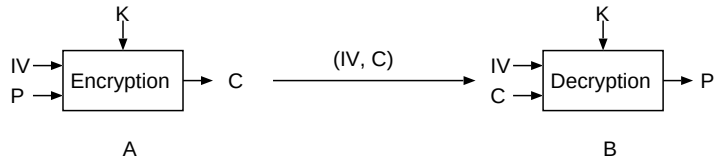

図 1: ブロック暗号秘匿モードを利用した秘匿通信のモデル

1.3.1 暗号技術の利用モデル

暗号技術による情報の秘匿では、暗号化のための鍵を用いて平文を暗号化、または、復号のための鍵を用いて暗号文を平文に復号する。本技術を含めた共通鍵暗号技術による情報の秘匿では、暗号化および復号において、互いに同一の鍵を用いる。

共通鍵暗号技術の代表的な構成要素として、ブロック暗号がありこれは二つの関数の対からなる。一方は暗号化関数と呼ばれ、鍵を用いてある決まった長さの平文ブロック (典型的には 64 ビットや 128 ビット) を同じ長さの暗号文ブロックに暗号化処理するものである。もう一つはこの逆関数である復号関数である。これは暗号化関数と同じ鍵を用いて暗号文ブロックからもとの平文ブロックに復号する。本技術は、このブロック暗号を用いて情報の秘匿を実現するものであり、共通鍵ブロック暗号の秘匿モードと呼ばれる。

以下では、共通鍵ブロック暗号の秘匿モードを利用する際の具体的なモデルの一例を記述する。情報を秘匿した通信を A と B の間で行う場合を考える。ここでは、A が送信者、B が受信者とする。A と B は共通の鍵 K を共有しており、A は暗号化装置、B は復号装置を持っている。暗号装置は平文、鍵と初期値を入力として受け取り、暗号文を出力する。復号装置は、暗号文、鍵と初期値を入力として受け取り、平文を出力する。これら、暗号装置および復号装置は、それぞれ、本文書で説明するブロック暗号の秘匿モードでの暗号化および復号を実装した装置である。

秘匿通信の手順は、次の通りである。

1. 送信者 A は、初期値 IV を生成し、平文 P、鍵 K、初期値 IV を暗号装置に入力し、暗号文 C を得る。
2. 送信者 A は、暗号文 C と初期値 IV を、受信者 B に送信する。
3. 受信者 B は、受け取った暗号文 C、初期値 IV とあらかじめ保持している鍵 K を復号装置に入力し、平文 P を得る。

手順 2 において、暗号文 C および初期値 IV は共に通信路上で他者に暴露されてもよい。手順の概略を、図 1 に示してある。

B から A に向けて、秘匿された通信を行う場合は、上記の A, B の役割を交代させることになる。従って、双方向で秘匿通信を行う場合には、A, B は双方とも暗号装置と復号装置の両方を持つことになる。双方向の鍵を共有することは一般的ではあるが、IV の管理は一方向の管理と差異なく、同じ IV が使われないような注意が必要である。

1.3.2 技術の基本的構成

共通鍵ブロック暗号の秘匿モードにおける基本的構成要素として、ブロック暗号、鍵、初期値、パディングがある。これらを説明する。

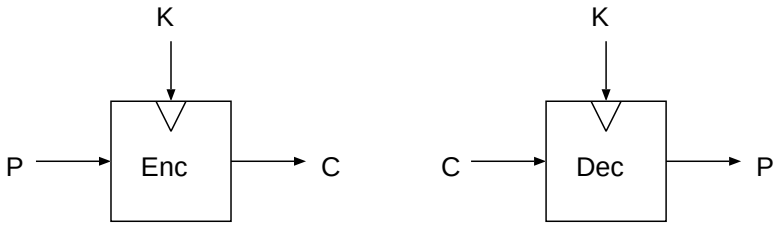

図 2: ブロック暗号での暗号化 (左) と復号 (右)

ブロック暗号　ブロック暗号は、鍵長 k ビットの鍵を用いて、長さ b ビットの平文 (暗号文) を暗号化 (復号) し、長さ b ビットの暗号文 (平文) に変換するものである。長さ b ビットはブロック長と呼ばれ、鍵長とブロック長は、各ブロック暗号に固有に定められた値である。

鍵 K で平文 P を暗号文 C に暗号化した場合、同じ鍵 K で暗号文 C を復号することにより平文 P が得られる。これを、暗号化関数 Enc、復号関数 Dec を使って記述すると、

$$C = Enc(K, P), \tag{1}$$

$$P = Dec(K, C), \tag{2}$$

と記述できる (図 2)。

ブロック暗号のアルゴリズムの仕様や、その選択方法は本文書では取り扱わない。達成すべき安全性レベルや実行環境、入手可能な方式などを考慮し、CRYPTREC が定めた電子政府推奨暗号リストから、相応しい方式を選択する。

鍵　ブロック暗号に入力する k ビットのビット列であり、暗号化する者と復号する者のみで共有する秘密情報である。秘密情報を共有する者以外に、知られてはならない。鍵長は、用いるブロック暗号により規定される。

秘密情報を共有する者の組毎に鍵は独立に生成する。また、複数の秘密情報に対して同一の鍵を使うことができる (その場合一般には異なる情報には個々に異なる初期値を使う)。ただし、一つ鍵を用いて処理される累計のデータ量には安全性の観点から上限 ($b \cdot 2^{b/2}$ ビット) があり、この上限を越えてはならない [BDJR97]。

初期値　初期値は CBC、CFB、CTR、OFB モードで暗号化・復号する際に必要となるビット列である。これらのモードでは、暗号化の際に鍵と平文に加えて、初期値が必要となる。復号の際には鍵と暗号文に加えて、暗号化で用いた初期値が必要になる。初期値は一般に秘密情報である必要はないため、第三者に知られても良い。このため初期値は暗号文に結合されて送信されたり、あるいは暗号文とともに保存されることが多い。ただし、暗号文に関する補足情報などから初期値を復号処理側が決定できる場合は、初期値を暗号文に付加しなくてもよい。

各モード毎に、初期値が満たすべき性質は異なる。CBC モードにおいては予測不可能性が必要であり、CFB、OFB モードにおいてはナンス性が必要とされる。CTR モードには、内部カウンタの動作も理解した上での特殊なナンス性が必要とされる。以下では、これらの性質を説明する。

- ナンス性 (nonce)　ナンス性とは、ある一つの鍵の下で以前に初期値として使われた値は、同じ鍵の下では二度と初期値として使われることはない性質のこ

とである。
例えば、使用する度に値が増えるカウンタはこの性質を満たしており、ナンス性を持つ初期値として利用できる。他の具体的な例として、ファイルシステム上のファイルを暗号化する場合には、ファイルの更新時刻とファイルのパスを結合したものを初期値として用いても、ナンス性を満す。
乱数生成を用いることは厳密にはナンス性を持たないが、同一のビット列が生成される確率は $b \geq 64$ などでは極めて低く、乱数値の衝突の事象を無視して考えることができるため、乱数もナンス性があると考えることができる。
CTR モードに関しては、モード内部のカウンタ値が処理ブロック毎に異なる必要があるため、初期値により細かい条件がかかる。これは次の 2 つの条件から成る。まず、1 つのメッセージ内でのカウンタ値の更新はそのメッセージ内で同じカウンタ値が繰り返されることのないようなものでなければならない。次に、メッセージの先頭ブロックに対するカウンタ値は、全てのメッセージに渡って、カウンタ値の繰り返しがないように更新されなければならない。このようなカウンタ値の更新方法としてここでは 2 つの例を挙げる。1 つ目の例として、複数のメッセージを直列に連続的に処理する場合、メッセージ内のブロック毎の更新、メッセージの先頭ブロックに対する更新のいずれでも、カウンタ値を 1 だけ増加させる。この処理は、一連のメッセージを 1 つのメッセージとみなして処理することに相当する。2 つ目の例として、カウンタ値のブロックを、m ビットと $b - m$ ビットの 2 つの部分に分け、m ビット部分をメッセージ内のブロック毎に更新する部分、$b - m$ ビット部分をメッセージの先頭ブロック毎に更新する部分とする方法がある。各更新では、各カウンタ値を 1 ずつ増加させる。
上記では、カウンタ値を 1 ずつ増加させて更新しているが、カウンタ値が繰り返されない更新方法が他にあれば、その方法で更新してもよい。例えば、線形フィードバックシフトレジスタ等での更新が考えられる。

- 予測不可能性　平文の第 1 ブロックが決定される時点で、初期値の部分情報が有意に予測できない場合、その初期値は予測不可能であるという。この性質を満たすために、次の 2 つの方法を例として挙げる。
一つの方法は、平文が決定した後に、鍵を用いてナンスをブロック暗号の暗号化関数で変換し、出力を初期値とする方法である。ここでナンスは上で説明したものと同様の性質であり、初期値生成のために入力される値はこれまでに一度も使われていない、という性質になる。二つ目の方法として、平文が決定した後に、乱数生成器から乱数を出力し、その値を初期値として利用する方法が挙げられる。

サイズの大きなデータの場合、再同期やランダムアクセスの容易さのためから、平文を適切な長さで区切り、分割された各平文に対して個々に異なる初期値を用いて暗号化処理を実施することが考えられる。その際にも、初期値は各モードに要求される性質を満たすように、適切に変化させなければならない。

パディング　暗号化関数に入力される平文の長さは、CBC、ECB モードではブロック長 b の倍数、CFB モードではパラメータ n の倍数でなければならない。そうでない場合は、平文にパディングを施して、長さがモードの入力に合致するように調整してから、パディングされた平文を暗号化関数に入力する必要がある。パディングで付加されたビットは、復号時曖昧さ無しに取り除くことができる必要がある。

パディングの一例を紹介する。ここでは揃えるべき境界条件を b とするが、n の場合も同様である。まず、平文の終端にビット値 “1” を結合し、その後に平文の長さがブロック長の倍数になるのに必要な最小個の “0” 値を結合する方法である。このパディング方法を “10(いち・ぜろ) パディング” と呼ぶ。このパディングの付け方の手順は、以下の通りとなる。平文の長さは M ビットとする。

1. 平文の終端にビット 1 を付け加える。
2. $M+1+l \equiv 0 \mod b, 0 \leq l \leq b-1$ を満たす l を求め、手順 1 でビット “1” を加えた平文の終端にさらにビット “0” の列を l ビット分付加する。

パディングされた平文からパディング部分を取り除く方法の手順は、次の通りである。

1. 復号で生成されたデータ列の終端から先頭方向にビットを見ていき、初めてビット値 “1” が出てくるところで、そのビット値 “1” とそれより終端側にあるビット値 “0” を全て取り除く。

また、注意すべきパディングの実施例を示す。この例では平文がバイト単位のデータで、ブロック長は $b/8$ バイトとする。メッセージ長を M バイトとしたとき、パディングは次の通りである。

1. $M+l \equiv 0 \mod b/8, 1 \leq l \leq b/8$ を満たす l を求める。
2. 平文の終端に l 個のバイト値 “l” を付け加える。

このパディングの取り除き方は、次の手順である。

1. 復号で生成されたデータ列の終端の 1 バイトを読み込む。このバイト値を z とする。$l' = z \mod b/8, 1 \leq l' \leq b/8$ を満たす l' を求める。データ列の終端から l' 個のバイトを取り除く[2]。

CBC モードに関しては、適切なパディング処理をしなければ、安全性が損われることが知られている [V02]。具体的には、安全性を保つためには、暗号文を復号して得られるデータがパディングのフォーマットに合致するかどうかを攻撃者に伝えることのないパディング処理方法である必要がある。例えば、上記の後者のパディング法において、パディングフォーマットに合わないデータが復号で得られた時にエラーを返すように実装している場合、エラーの有無により攻撃者に復号されたデータがフォーマットと合致しているか否かの情報が漏れてしまう。一方、“10 パディング” では、復号されたデータはほぼ全てが、それが正しい平文であるかどうかによらずに、パディングフォーマットに合致したものとなる。従って、“10 パディング” を用いる限りでは、上記脆弱性はないと考えられるため、CBC モード利用時には “10 パディング” の採用を推奨する。

インターリーブ　CBC、CFB、OFB モードには並列処理性はない。しかし、これらのモードで処理を行う際には、以下に説明する方法で処理をインターリーブすることにより並列処理が可能となる。これは、実行環境におけるハードウェアやソフトウェアの条件により複数の独立な処理を並列に処理できる場合にメリットがある。インターリーブとは、処理する平文を仮想的に複数のメッセージからなるものとし、

[2]正しくパディングされていれば、終端バイトは 1 から $8/b$ までの数になっているはずである。ただし、暗号文にノイズがのった場合等、パディングのフォーマットに合わないデータ列となることがある。この場合でも、後述の CBC モードで安全性を保つためには、エラー等を返さずに、決まった方法でパディング処理を進め、平文を出力する必要がある。

メッセージを並列数だけ分割し、それらを独立に処理することである。一般にブロック毎に次々と別の処理系に割り振る。よって単一の処理系は、並列数分だけのブロックを飛ばしながら処理を行うことになる。
例えば、ブロック暗号の段関数の繰り返し構造を利用して、パイプライン処理[3]を行うようにブロック暗号をハードウェアで実装している場合に、インターリーブは効果的である。なぜならば、この実装では、複数の独立なデータストリームを暗号化/復号することができるため、インターリーブされたモードを処理するのに適しているからである。

1.3.3 評価観点と比較

本節では、まず、本文書でのノイズとエラー伝播の定義を説明してから、モード間の比較を行い、どのような場合に各モードの利用が推奨されるかを記載する。そして、各モード毎の特徴についても、個別に記載する。

ノイズとエラー伝播 ブロック暗号の処理内部や暗号文の転送に障害が発生する可能性がある。これらが発生すると処理結果 (暗号文や復号した結果) に影響が出る。これらの影響の出方を含めて適切なモードを選択することが必要となる場合がある。一般には、ブロック暗号の処理内部に発生したエラーやその影響を考えてモードを選択することは稀であり、それ以外の暗号文の転送や記録、読み出しにおいて発生する、データの書き換えやデータの欠損、不正データの挿入を考える。データサイズに変更のない書き換えに相当するエラーの影響は、そのエラーがどう復号結果に伝播するか、という観点で “エラー伝播” と本文書では記載する。暗号文の長さが変化するようなデータの一部欠損や挿入は、同期が取れなくなったと考え、自己同期回復できるかどうかをモードの選択における扱いとすることが一般的であり、モードの適切な選択により、望ましくない性質を回避するなどできる場合がある。

モード選択のためのガイド モードの特徴を表 1 にまとめる。特別な理由が無い限りは、CBC または CTR モードの利用を推奨する。CFB モードは、ブロック長より短いデータ単位での自己同期回復が暗号機能自身にも必要とされる場合に利用することを推奨する。OFB モードは、CTR モードと比べ積極的に採用する理由は少ないが、CTR モードでのモード内部のカウンタの管理が障害となる場合には、採用が考えられる。ECB モードは、基本的に利用しないことを推奨するが、利用する場合は他のモードが適用できない特別な理由がある時に限る。

各モードの特徴

- CBC モード CBC モードは安全なモードであり、一般によく使われているモードである。復号時のエラー伝播は、ある暗号文ブロックの 1 ビットが反転した場合、対応する平文ブロック全体と次の平文ブロックの 1 ビットが損壊する。自己同期性に関しては、欠損した暗号文が処理ブロック長に一致する場合は、同期を保持できるが、それ以外の場合は自動では回復しない。暗号化処理において、並列処理性はない。復号処理においては、ブロック暗号の処理部

[3] 一般にブロック暗号では同一の処理 (処理単位を「段」と呼ぶ) を複数回繰り返す。このことを使って、依存関係のない複数のブロックを並列に処理するために、専用のハードの構成として、複数の段関数を直列に実装し、これらの間にレジスタを配置するものがある。このような構成による処理をパイプライン処理という。

表 1: モードの特徴の比較

モード	安全性	処理速度	並列処理	自己同期	エラー伝播	パディング	初期値
CBC	○	○	欄外 3	欄外 4	欄外 5	欄外 7	予測不可能性
CFB	○	欄外 2	欄外 3	○	欄外 5	欄外 8	ナンス性
CTR	○	○	○	×	欄外 6	不要	特殊なナンス性
ECB	欄外 1	○	○	欄外 4	欄外 5	欄外 7	(初期値入力なし)
OFB	○	○	欄外 3	×	欄外 6	不要	ナンス性

欄外 1 平文がブロック毎に異なることが保証されるなど、安全に用いるためには条件が必要となる。
欄外 2 ブロック暗号の呼び出し回数が b/n 倍となるため、処理速度はほぼ n/b 倍となる。
欄外 3 インターリーブを参照。
欄外 4 データの欠損・挿入がブロック境界に一致したにのみ同期が保たれるが一般には同期は回復しない。
欄外 5 暗号文のエラーは 1,2 ブロックに跨る範囲で平文に影響するが、伝播し続けることはない。
欄外 6 暗号文上でエラー (ビット) パターンがそのまま平文上のエラーパターンとなる。
欄外 7 メッセージ長がブロック長の倍数である保証があればパディングが不要だが、そうでないメッセージを処理する可能性がある場合には、パディングが必要。
欄外 8 メッセージ長がパラメータ n の倍数ビット長である保証があればパディングが不要だが、そうでないメッセージを処理する可能性がある場合には、パディングが必要。

分は並列処理が可能である。初期値には、予測不可能性が必要とされる。ただし、不適切な初期値生成により、初期値が予測可能であったような場合にも、暗号文から漏れる平文の情報は限定的である。また、パディング方法によっては安全性が保たれない可能性がある。“10 パディング” の採用が推奨される。

- CFB モード　CFB モードは安全なモードである。セグメントの長さを 1 から b までの間で選べ、例えば、バイト、ワードといった短い単位に設定できる。エラー伝播は、ある暗号文セグメントの 1 ビットが反転した場合、対応する平文セグメントの 1 ビットが損壊し、これ以降の平文セグメントでは、該当エラーがブロック暗号の暗号化関数の入力レジスタから消えるまで、損壊し続ける。自己同期性は、欠損した暗号文がセグメント長の倍数である場合は、自動的に同期を回復する。暗号化・復号処理ともに並列処理性はない。セグメント長を n とした場合、他のモードに比べ、処理速度が約 n/b に落ちる。初期値には、ナンス性が必要とされる。

- CTR モード　CTR モードは安全なモードであり、一般によく使われているモードである。エラー伝播は、ある暗号文ブロックの 1 ビットが反転した場合、対応する平文ブロックの 1 ビットが損壊する。自己同期性はなく、自動的に同期を回復することはない。暗号化・復号処理ともに並列処理が可能である。鍵と初期値を含めたモード内部のカウンタさえあれば、平文または暗号文を得る前でも事前計算によりブロック暗号の計算そのものは行える。安全性の観点から、ある一つの鍵におけるすべての処理において、ブロック単位の内部カウンタ値が常にどれとどれを取っても異なっていなければならない。これを満たす方法として、ここでは 2 つの例を挙げる。1 つ目の例として、複数のメッセージを直列に処理する場合、メッセージ内のブロック毎の更新、メッセージの先頭ブロックに対する更新のいずれでも、カウンタ値を 1 だけ増加させる。この処理は、一連のメッセージを 1 つのメッセージとみなして処理することに相当する。2 つ目の例として、カウンタ値のブロックを、m ビットと $b-m$ ビットの 2 つの部分に分け、m ビット部分をメッセージ内のブロック毎に更新する部分、$b-m$ ビット部分をメッセージの先頭ブロック毎に更新する部分とする方法がある。各更新では、各カウンタ値を 1 ずつ増加させる。逆に不適切な初期値と内部カウンタ値を用いた場合の危険性は高い。例えば、以前使われた内部カウンタ値が再び使われた場合、対応する暗号文ブロックから平文ブロックの全ての情報が漏れることがある。また、カウンタ値の更新方法やカウンタ値の

初期値の仕様が曖昧であると、相互接続性を保てないため、利用時には、詳細に初期値運用やカウンタ値更新方法を確認する必要がある。

- ECB モード　ECB モードは安全性上の問題があるため、他のモードが利用できない場合を除いては、このモードの利用は推奨されない。具体的には、ECB モードは平文ブロックが同じならば、対応する出力ブロックも同じものとなる。従って、ECB モードは平文ブロックの一致の存在とその一致場所が漏れるという脆弱性が存在する。安全に利用するためには、同じ値の平文ブロックが出現しないことを保証する必要があるが、このような条件を平文に課することは、実システムでの入力データを注意深く詳細に把握することが必要である。この条件が保持されない限りは、なるべく ECB モードの利用は避けるべきである。復号時のエラー伝播性は、ある暗号文ブロックの 1 ビットが反転した場合、対応する平文ブロック全体が損壊する。自己同期性に関しては、欠損した暗号文がブロック長の倍数の場合に限っては、自動的に同期を回復する。暗号化、復号処理ともに、並列処理が可能である。

- OFB モード　OFB モードは安全なモードである。エラー伝播は、ある暗号文ブロックの 1 ビットが反転した場合、対応する平文ブロックの 1 ビットが損壊する。自己同期性はなく、自動的に同期を回復することはない。暗号化、復号処理どちらにおいても並列処理性はない。鍵と初期値さえあれば、平文または暗号文を得る前でも事前計算によりブロック暗号の計算そのものは行える。初期値には、ナンス性が必要とされる。不適切な初期値を用いた場合の危険性は高い。例えば、以前使われた初期値が再び使われた場合、暗号文から平文の情報が漏れることがある。なお、以前の標準文書 [FIPS-81] で定義されていた k-OFB は致命的な安全性の欠陥を持ちうるため [W02]、使われてはならず、該当標準文書では利用推奨しない。

1.4　実装仕様

この章では、秘匿に関する利用モードのうち、CBC, CFB, CTR, ECB, OFB の各モードの仕様を定義する。

1.4.1　CBC

CBC モードは、平文長が n の倍数であるような平文に対して暗号化を行なう利用モードである。手法は、平文を b ビット毎のブロックに分割し (それぞれを $M_i, i = 1, 2, \ldots, m$ とする)、中間値 $H_i = M_i \oplus C_{i-1}, C_0 = IV$ を生成したあと、それをブロック暗号の暗号化関数の入力とする。その結果えられた出力が暗号文ブロック (C_i) となり、暗号文はそれらを接続したものである。

$$\begin{aligned} C_0 &= IV, \\ C_i &= Enc_K(M_i \oplus C_{i-1}). \end{aligned}$$

復号はその逆関数である。

$$M_i = Dec_K(C_i) \oplus C_{i-1}.$$

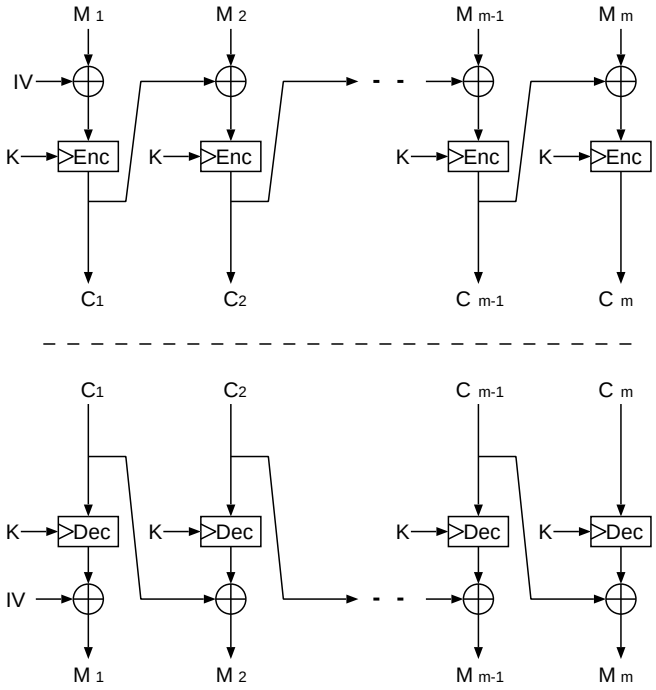

図 3: CBC モードの暗号化と復号のブロック図

他のバリエーション　ISO/IEC 10116:2005 では、インターリーブされた処理も含めた定義となっている。インターリーブする次数 m は $1 \leq m \leq 1024$ から選択できる。また、ISO/TR 19038 では、TDEA が DES を 3 重に処理することに注目し、上記インターリーブ仕様の $b = 64$, $m = 3$ のものを TCBC-I として定義している。

1.4.2　CFB

CFB モードは、パラメータ n を持つブロック暗号利用モードである。便宜的に内部レジスタ R を考えながら処理を説明する。

n ビットの倍数であるメッセージ M を、n ビット毎のブロック列に分割する。これを $M_i, i = 1, \ldots, m$ とする。内部レジスタ R の初期値 R_0 を IV とする。各ブロックでのブロック処理では、まず中間値 $H_i = Enc_K(R_{i-1})$ を生成し、このうち上位 n ビットの値 $\hat{H}_i$ を平文ブロック M_i と排他論理和することで暗号文ブロック $C_i = M_i \oplus \hat{H}_i$ を得る。最後に次の手順で R を更新する; R を上位へ n ビットシフトし、シフトの際、0 が埋められた下位 n ビットに C_i を埋め込む。$n = b$ であるような場合は、$R_i = C_i$ となる。

$$\begin{aligned} R_0 &= IV, \\ C_i &= M_i \oplus \mathrm{msb}_n(Enc_K(R_{i-1})). \\ R_i &= ((R_{i-1}) \ll n) \oplus C_i. \end{aligned}$$

復号はその逆関数である。

$$\begin{aligned} M_i &= C_i \oplus \mathrm{msb}_n(Enc_K(R_{i-1})), \\ R_i &= ((R_{i-1}) \ll n) \oplus C_i. \end{aligned}$$

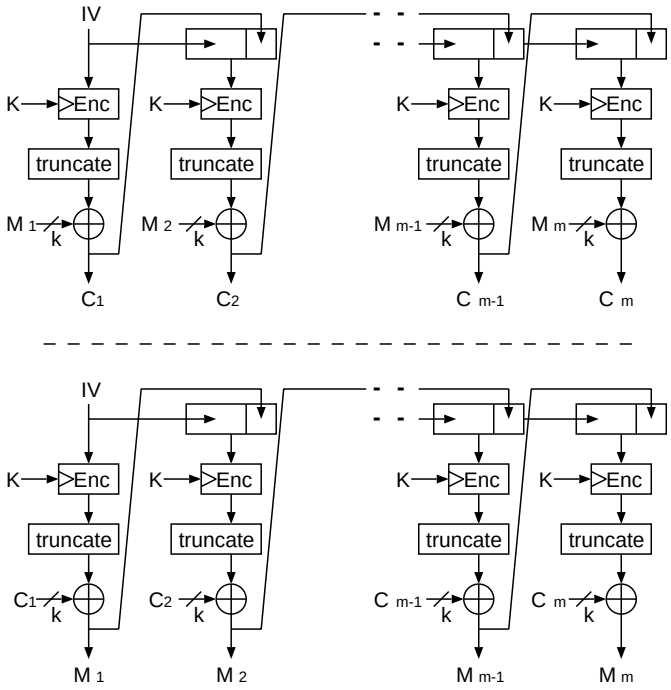

図 4: n-CFB モードの暗号化と復号のブロック図

他のバリエーション　ISO/IEC 10116:2005 では、次の 2 点の拡張が盛り込まれている; (1) インターリーブ可能 (次数 $m, 1 \leq m \leq 1024$)、(2) 柔軟なレジスタ更新サイズ $(n', n \leq n' \leq b)$。また、ISO/TR 19038 では、TDEA が DES を 3 重に処理することに注目し、上記インターリーブ仕様の $b = 64$, $m = 3$ のものを TCFB-I として定義している。

1.4.3　CTR

CTR モードは、任意のビット長のメッセージを暗号化できる。まず、メッセージを b ビット毎のブロック列と末端ブロックに分割する。これらを $M_i, i = 1, \ldots, m$ とする。$|M_i| = b, i = 1, \ldots, m-1$ であり、末端ブロックについてはそのビット長を $b' = |M_m|$ $(\leq b)$ とおく。内部レジスタ R の初期値 R_0 を IV とする。R_i をブロック暗号入力とし、暗号化処理の結果を H_i とする。これより暗号文ブロック $C_i = M_i \oplus H_i$ を生成する。次のブロックでは、内部レジスタ R を規定の方法で更新する。

$$\begin{aligned} R_0 &= IV, \\ C_i &= M_i \oplus Enc_K(R_{i-1}), \\ R_i &= inc(R_{i-1}), \\ C_m &= M_m \oplus \mathrm{msb}_{b'}(Enc_K(R_{m-1})). \end{aligned}$$

復号はその逆関数である。

$$\begin{aligned} M_i &= C_i \oplus Enc_K(R_{i-1}), \\ R_i &= inc(R_{i-1}), \\ M_m &= C_m \oplus \mathrm{msb}_{b'}(Enc_K(R_{m-1})). \end{aligned}$$

カウンタの更新手法、初期値の決定方法については第 1.3.2 節を参照せよ。

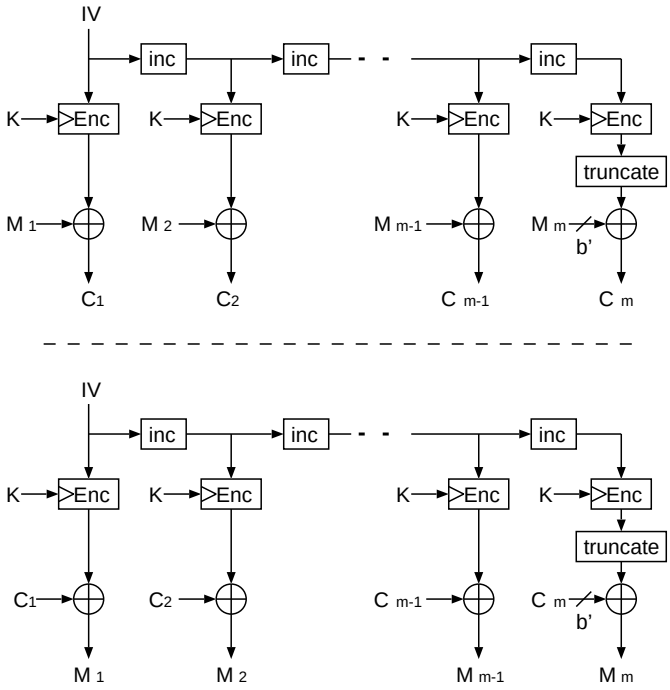

図 5: CTR モードの暗号化と復号のブロック図

他のバリエーション　ISO/IEC 10116:2005 では、次の 3 点において本稿で記述した CTR モードとの違いがある; (1) 末端ブロックが端数であることを許さず、メッセージ長が n(後述) ビットの倍数である, (2) カウンタは b ビットの整数カウンタであり、$\mathrm{mod}\, 2^b$ における算術 1 加算だけに限定している, (3) パラメータ n が定義するメッセージのセグメントサイズを定義し、1 回のブロック暗号化処理に対して n ビットのメッセージセグメントの暗号化を行う。

1.4.4　ECB

ECB モードは、平文長が b の倍数であるような平文に対して暗号化を行なう利用モードである。手法は、平文を b ビット毎のブロックに分割し (それぞれを M_i とする)、それぞれ独立にブロック暗号の暗号化関数の入力とする。その結果えられた出力が暗号文ブロック (C_i) となり、暗号文はそれらを接続したものである。

$$C_i = Enc_K(M_i).$$

この利用モードには初期値がない。平文と鍵のみから暗号文が生成される。復号はその逆関数である。

$$M_i = Dec_K(C_i).$$

1.4.5　OFB

OFB モードは、任意のビット長のメッセージを暗号化できる。まず、メッセージを b ビット毎のブロック列と末端ブロックに分割する。これらを $M_i, i = 1, \ldots, m$ とする。$|M_i| = b, i = 1, \ldots, m-1$ であり、末端ブロックについてはそのビット長を $b' = |M_m|$ $(\leq b)$ とおく。

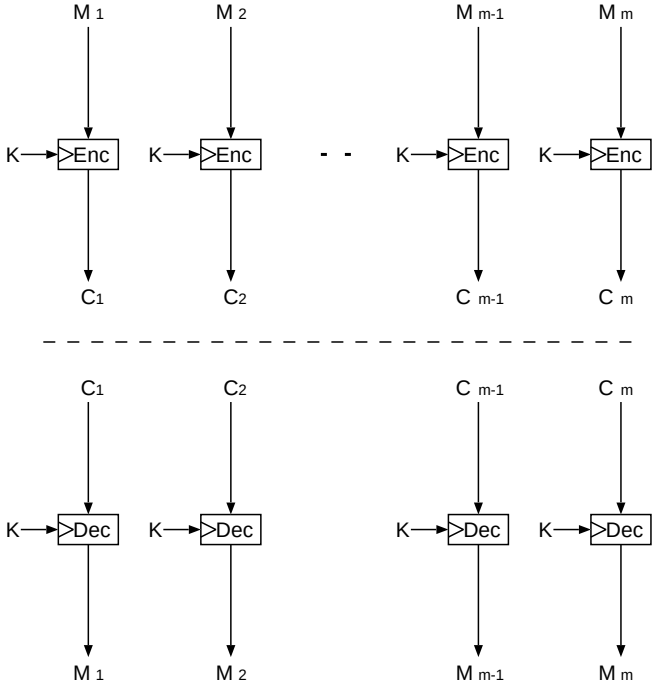

図 6: ECB モードの暗号化と復号のブロック図

内部レジスタ R の初期値 R_0 を IV とする。R_i をブロック暗号入力とし、暗号化処理の結果を R_{i+1} とする。これより暗号文ブロック $C_i = M_i \oplus R_i$ を生成する。

$$\begin{aligned} R_0 &= IV, \\ R_i &= Enc_K(R_{i-1}), \\ R_m &= \mathrm{msb}_{b'}(Enc_K(R_{i-1})), \\ C_i &= M_i \oplus R_i. \end{aligned}$$

復号はその逆関数である。

$$\begin{aligned} R_i &= Enc_K(R_{i-1}), \\ R_m &= \mathrm{msb}_{b'}(Enc_K(R_{i-1})), \\ M_i &= C_i \oplus R_i. \end{aligned}$$

他のバリエーション　ISO/IEC 10116:2005 では、次の 2 点において本稿で記述した OFB モードとの違いがある; (1) 末端ブロックが端数であることを許さず、メッセージ長が n(後述) ビットの倍数である, (2) パラメータ n が定義するメッセージセグメントサイズを定義し、1 回のブロック暗号処理に対して n ビットのメッセージの暗号化を行う。

1.5　テストベクトル

ここでは、参考文献 [SP800-38A] に記載されるテストベクトルのうち 128 ビット鍵 AES を使ったブロック暗号について、各モードの処理結果例を示す。

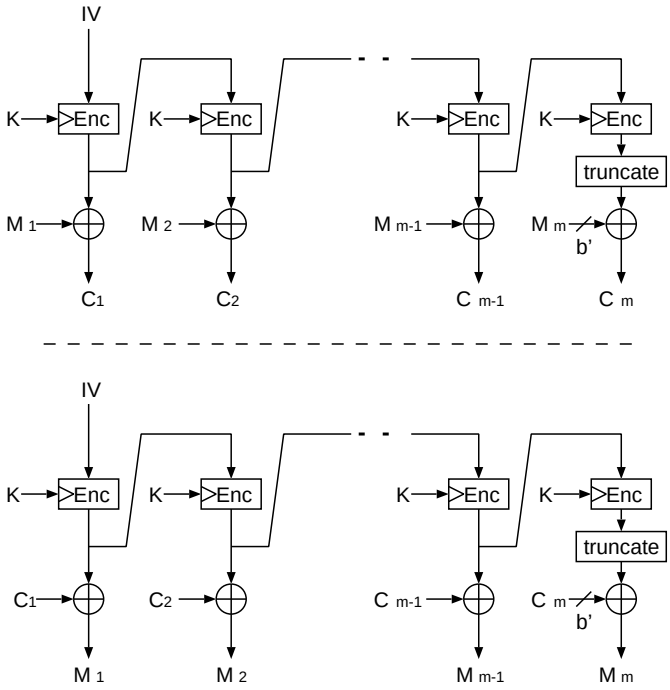

図 7: OFB モードの暗号化と復号のブロック図

1.5.1 CBC 暗号化

```
F.2.1 CBC-AES128.Encrypt
Key           2b7e151628aed2a6abf7158809cf4f3c
IV            000102030405060708090a0b0c0d0e0f
Block #1
Plaintext     6bc1bee22e409f96e93d7e117393172a
Input Block   6bc0bce12a459991e134741a7f9e1925
Output Block  7649abac8119b246cee98e9b12e9197d
Ciphertext    7649abac8119b246cee98e9b12e9197d
Block #2
Plaintext     ae2d8a571e03ac9c9eb76fac45af8e51
Input Block   d86421fb9f1a1eda505ee1375746972c
Output Block  5086cb9b507219ee95db113a917678b2
Ciphertext    5086cb9b507219ee95db113a917678b2
Block #3
Plaintext     30c81c46a35ce411e5fbc1191a0a52ef
Input Block   604ed7ddf32efdff7020d0238b7c2a5d
Output Block  73bed6b8e3c1743b7116e69e22229516
Ciphertext    73bed6b8e3c1743b7116e69e22229516
Block #4
Plaintext     f69f2445df4f9b17ad2b417be66c3710
Input Block   8521f2fd3c8eef2cdc3da7e5c44ea206
Output Block  3ff1caa1681fac09120eca307586e1a7
Ciphertext    3ff1caa1681fac09120eca307586e1a7
```

1.5.2 CBC 復号

```
F.2.2 CBC-AES128.Decrypt
Key           2b7e151628aed2a6abf7158809cf4f3c
IV            000102030405060708090a0b0c0d0e0f
Block #1
Ciphertext    7649abac8119b246cee98e9b12e9197d
Input Block   7649abac8119b246cee98e9b12e9197d
Output Block  6bc0bce12a459991e134741a7f9e1925
Plaintext     6bc1bee22e409f96e93d7e117393172a
Block #2
Ciphertext    5086cb9b507219ee95db113a917678b2
Input Block   5086cb9b507219ee95db113a917678b2
Output Block  d86421fb9f1a1eda505ee1375746972c
Plaintext     ae2d8a571e03ac9c9eb76fac45af8e51
Block #3
Ciphertext    73bed6b8e3c1743b7116e69e22229516
Input Block   73bed6b8e3c1743b7116e69e22229516
Output Block  604ed7ddf32efdff7020d0238b7c2a5d
Plaintext     30c81c46a35ce411e5fbc1191a0a52ef
Block #4
Ciphertext    3ff1caa1681fac09120eca307586e1a7
Input Block   3ff1caa1681fac09120eca307586e1a7
Output Block  8521f2fd3c8eef2cdc3da7e5c44ea206
Plaintext     f69f2445df4f9b17ad2b417be66c3710
```

1.5.3 1-CFB 暗号化

```
F.3.1 CFB1-AES128.Encrypt
Key           2b7e151628aed2a6abf7158809cf4f3c
IV            000102030405060708090a0b0c0d0e0f
Segment #1
Input Block   000102030405060708090a0b0c0d0e0f
Output Block  50fe67cc996d32b6da0937e99bafec60
Plaintext     0
Ciphertext    0
Segment #2
Input Block   00020406080a0c0e10121416181a1c1e
Output Block  19cf576c7596e702f298b35666955c79
Plaintext     1
Ciphertext    1
Segment #3
Input Block   0004080c1014181c2024282c3034383d
Output Block  59e17759acd02b801fa321ea059e331f
Plaintext     1
Ciphertext    1
```

```
Segment #4
Input Block  00081018202830384048505860687078
Output Block 71f415b0cc109e8b0faa14ab740c22f4
Plaintext    0
Ciphertext   0
Segment #5
Input Block  00102030405060708090a0b0c0d0e0f6
Output Block 3fb76d3d1048179964597a0f64d5adad
Plaintext    1
Ciphertext   1
Segment #6
Input Block  0020406080a0c0e10121416181a1c1ed
Output Block 4c943b4bac54ab974e3e52326d29aaa1
Plaintext    0
Ciphertext   0
Segment #7
Input Block  004080c1014181c2024282c3034383da
Output Block c94da41eb3d3acf1993a512ab1e8203f
Plaintext    1
Ciphertext   0
Segment #8
Input Block  00810182028303840485058606870784
Output Block e07f5e98778f75dbb2691c3f582c3953
Plaintext    1
Ciphertext   0
Segment #9
Input Block  01020304050607080a0b0c0d0e0f68
Output Block 02ef5fc8961efcce8568bc0731262dc7
Plaintext    1
Ciphertext   1
Segment #10
Input Block  020406080a0c0e10121416181a1c1ed1
Output Block 9f5a30367065efbe914b53698c8716b7
Plaintext    1
Ciphertext   0
Segment #11
Input Block  04080c1014181c2024282c3034383da2
Output Block d018cfb81d0580edbff955ed74d382db
Plaintext    0
Ciphertext   1
Segment #12
Input Block  08101820283038404850586068707b45
Output Block 81272ab351e08e0b695b94b8164d86f4
Plaintext    0
Ciphertext   1
Segment #13
```

```
Input Block  102030405060708090a0b0c0d0e0f68b
Output Block 094d33f856483d3fa01ba94f7e5ab3e7
Plaintext    0
Ciphertext   0
Segment #14
Input Block  20406080a0c0e10121416181a1c1ed16
Output Block 609900ad61923c8c102cd8d0d7947a2c
Plaintext    0
Ciphertext   0
Segment #15
Input Block  4080c1014181c2024282c3034383da2c
Output Block 9e5a154de966ab4db9c88b22a398134e
Plaintext    0
Ciphertext   1
Segment #16
Input Block  810182028303840485058606870 7b459
Output Block 7fe16252b338bc4de3725c4156dfed20
Plaintext    1
Ciphertext   1
```

1.5.4 1-CFB 復号

```
F.3.2 CFB1-AES128.Decrypt
Key          2b7e151628aed2a6abf7158809cf4f3c
IV           000102030405060708090a0b0c0d0e0f
Segment #1
Input Block  000102030405060708090a0b0c0d0e0f
Output Block 50fe67cc996d32b6da0937e99bafec60
Ciphertext   0
Plaintext    0
Segment #2
Input Block  00020406080a0c0e10121416181a1c1e
Output Block 19cf576c7596e702f298b35666955c79
Ciphertext   1
Plaintext    1
Segment #3
Input Block  0004080c1014181c2024282c3034383d
Output Block 59e17759acd02b801fa321ea059e331f
Ciphertext   1
Plaintext    1
Segment #4
Input Block  000810182028303840485058606870 7b
Output Block 71f415b0cc109e8b0faa14ab740c22f4
Ciphertext   0
Plaintext    0
Segment #5
```

```
Input Block  00102030405060708090a0b0c0d0e0f6
Output Block 3fb76d3d1048179964597a0f64d5adad
Ciphertext   1
Plaintext    1
Segment #6
Input Block  0020406080a0c0e10121416181a1c1ed
Output Block 4c943b4bac54ab974e3e52326d29aaa1
Ciphertext   0
Plaintext    0
Segment #7
Input Block  004080c1014181c2024282c3034383da
Output Block c94da41eb3d3acf1993a512ab1e8203f
Ciphertext   0
Plaintext    1
Segment #8
Input Block  00810182028303840485058606870 7b4
Output Block e07f5e98778f75dbb2691c3f582c3953
Ciphertext   0
Plaintext    1
Segment #9
Input Block  0102030405060708090a0b0c0d0e0f68
Output Block 02ef5fc8961efcce8568bc0731262dc7
Ciphertext   1
Plaintext    1
Segment #10
Input Block  020406080a0c0e10121416181a1c1ed1
Output Block 9f5a30367065efbe914b53698c8716b7
Ciphertext   0
Plaintext    1
Segment #11
Input Block  04080c1014181c2024282c3034383da2
Output Block d018cfb81d0580edbff955ed74d382db
Ciphertext   1
Plaintext    0
Segment #12
Input Block  0810182028303840485058606870 7b45
Output Block 81272ab351e08e0b695b94b8164d86f4
Ciphertext   1
Plaintext    0
Segment #13
Input Block  10203040506070 8090a0b0c0d0e0f68b
Output Block 094d33f856483d3fa01ba94f7e5ab3e7
Ciphertext   0
Plaintext    0
Segment #14
Input Block  20406080a0c0e10121416181a1c1ed16
```

```
Output Block 609900ad61923c8c102cd8d0d7947a2c
Ciphertext   0
Plaintext    0
Segment #15
Input Block  4080c1014181c2024282c3034383da2c
Output Block 9e5a154de966ab4db9c88b22a398134e
Ciphertext   1
Plaintext    0
Segment #16
Input Block  810182028303840485058606870 7b459
Output Block 7fe16252b338bc4de3725c4156dfed20
Ciphertext   1
Plaintext    1
```

1.5.5 8-CFB 暗号化

```
F.3.7 CFB8-AES128.Encrypt
Key          2b7e151628aed2a6abf7158809cf4f3c
IV           000102030405060708090a0b0c0d0e0f
Segment #1
Input Block  000102030405060708090a0b0c0d0e0f
Output Block 50fe67cc996d32b6da0937e99bafec60
Plaintext    6b
Ciphertext   3b
Segment #2
Input Block  0102030405060708090a0b0c0d0e0f3b
Output Block b8eb865a2b026381abb1d6560ed20f68
Plaintext    c1
Ciphertext   79
Segment #3
Input Block  020304050607080 90a0b0c0d0e0f3b79
Output Block fce6033b4edce64cbaed3f61ff5b927c
Plaintext    be
Ciphertext   42
Segment #4
Input Block  030405060708090a0b0c0d0e0f3b7942
Output Block ae4e5e7ffe805f7a4395b180004f8ca8
Plaintext    e2
Ciphertext   4c
Segment #5
Input Block  0405060708090a0b0c0d0e0f3b79424c
Output Block b205eb89445b62116f1deb988a81e6dd
Plaintext    2e
Ciphertext   9c
Segment #6
Input Block  05060708090a0b0c0d0e0f3b79424c9c
```

```
Output Block 4d21d456a5e239064fff4be0c0f85488
Plaintext    40
Ciphertext   0d
Segment #7
Input Block  060708090a0b0c0d0e0f3b79424c9c0d
Output Block 4b2f5c3895b9efdc85ee0c5178c7fd33
Plaintext    9f
Ciphertext   d4
Segment #8
Input Block  0708090a0b0c0d0e0f3b79424c9c0dd4
Output Block a0976d856da260a34104d1a80953db4c
Plaintext    96
Ciphertext   36
Segment #9
Input Block  08090a0b0c0d0e0f3b79424c9c0dd436
Output Block 53674e5890a2c71b0f6a27a094e5808c
Plaintext    e9
Ciphertext   ba
Segment #10
Input Block  090a0b0c0d0e0f3b79424c9c0dd436ba
Output Block f34cd32ffed495f8bc8adba194eccb7a
Plaintext    3d
Ciphertext   ce
Segment #11
Input Block  0a0b0c0d0e0f3b79424c9c0dd436bace
Output Block e08cf2407d7ed676c9049586f1d48ba6
Plaintext    7e
Ciphertext   9e
Segment #12
Input Block  0b0c0d0e0f3b79424c9c0dd436bace9e
Output Block 1f5c88a19b6ca28e99c9aeb8982a6dd8
Plaintext    11
Ciphertext   0e
Segment #13
Input Block  0c0d0e0f3b79424c9c0dd436bace9e0e
Output Block a70e63df781cf395a208bd2365c8779b
Plaintext    73
Ciphertext   d4
Segment #14
Input Block  0d0e0f3b79424c9c0dd436bace9e0ed4
Output Block cbcfe8b3bcf9ac202ce18420013319ab
Plaintext    93
Ciphertext   58
Segment #15
Input Block  0e0f3b79424c9c0dd436bace9e0ed458
Output Block 7d9fac6604b3c8c5b1f8c5a00956cf56
```

```
Plaintext    17
Ciphertext   6a
Segment #16
Input Block  0f3b79424c9c0dd436bace9e0ed4586a
Output Block 65c3fa64bf0343986825c636f4a1efd2
Plaintext    2a
Ciphertext   4f
Segment #17
Input Block  3b79424c9c0dd436bace9e0ed4586a4f
Output Block 9cff5e5ff4f554d56c924b9d6a6de21d
Plaintext    ae
Ciphertext   32
Segment #18
Input Block  79424c9c0dd436bace9e0ed4586a4f32
Output Block 946c3dc1584cc18400ecd8c6052c44b1
Plaintext    2d
Ciphertext   b9
```

1.5.6 8-CFB 復号

```
F.3.8 CFB8-AES128.Decrypt
Key          2b7e151628aed2a6abf7158809cf4f3c
IV           000102030405060708090a0b0c0d0e0f
Segment #1
Input Block  000102030405060708090a0b0c0d0e0f
Output Block 50fe67cc996d32b6da0937e99bafec60
Ciphertext   3b
Plaintext    6b
Segment #2
Input Block  0102030405060708090a0b0c0d0e0f3b
Output Block b8eb865a2b026381abb1d6560ed20f68
Ciphertext   79
Plaintext    c1
Segment #3
Input Block  02030405060708090a0b0c0d0e0f3b79
Output Block fce6033b4edce64cbaed3f61ff5b927c
Ciphertext   42
Plaintext    be
Segment #4
Input Block  030405060708090a0b0c0d0e0f3b7942
Output Block ae4e5e7ffe805f7a4395b180004f8ca8
Ciphertext   4c
Plaintext    e2
Segment #5
Input Block  0405060708090a0b0c0d0e0f3b79424c
Output Block b205eb89445b62116f1deb988a81e6dd
```

```
Ciphertext   9c
Plaintext    2e
Segment #6
Input Block  05060708090a0b0c0d0e0f3b79424c9c
Output Block 4d21d456a5e239064fff4be0c0f85488
Ciphertext   0d
Plaintext    40
Segment #7
Input Block  060708090a0b0c0d0e0f3b79424c9c0d
Output Block 4b2f5c3895b9efdc85ee0c5178c7fd33
Ciphertext   d4
Plaintext    9f
Segment #8
Input Block  0708090a0b0c0d0e0f3b79424c9c0dd4
Output Block a0976d856da260a34104d1a80953db4c
Ciphertext   36
Plaintext    96
Segment #9
42
Input Block  08090a0b0c0d0e0f3b79424c9c0dd436
Output Block 53674e5890a2c71b0f6a27a094e5808c
Ciphertext   ba
Plaintext    e9
Segment #10
Input Block  090a0b0c0d0e0f3b79424c9c0dd436ba
Output Block f34cd32ffed495f8bc8adba194eccb7a
Ciphertext   ce
Plaintext    3d
Segment #11
Input Block  0a0b0c0d0e0f3b79424c9c0dd436bace
Output Block e08cf2407d7ed676c9049586f1d48ba6
Ciphertext   9e
Plaintext    7e
Segment #12
Input Block  0b0c0d0e0f3b79424c9c0dd436bace9e
Output Block 1f5c88a19b6ca28e99c9aeb8982a6dd8
Ciphertext   0e
Plaintext    11
Segment #13
Input Block  0c0d0e0f3b79424c9c0dd436bace9e0e
Output Block a70e63df781cf395a208bd2365c8779b
Ciphertext   d4
Plaintext    73
Segment #14
Input Block  0d0e0f3b79424c9c0dd436bace9e0ed4
Output Block cbcfe8b3bcf9ac202ce18420013319ab
```

```
Ciphertext   58
Plaintext    93
Segment #15
Input Block  0e0f3b79424c9c0dd436bace9e0ed458
Output Block 7d9fac6604b3c8c5b1f8c5a00956cf56
Ciphertext   6a
Plaintext    17
Segment #16
Input Block  0f3b79424c9c0dd436bace9e0ed4586a
Output Block 65c3fa64bf0343986825c636f4a1efd2
Ciphertext   4f
Plaintext    2a
Segment #17
Input Block  3b79424c9c0dd436bace9e0ed4586a4f
Output Block 9cff5e5ff4f554d56c924b9d6a6de21d
Ciphertext   32
Plaintext    ae
Segment #18
Input Block  79424c9c0dd436bace9e0ed4586a4f32
Output Block 946c3dc1584cc18400ecd8c6052c44b1
Ciphertext   b9
Plaintext    2d
```

1.5.7 CTR 暗号化

```
F.5.1 CTR-AES128.Encrypt
Key           2b7e151628aed2a6abf7158809cf4f3c
Init. Counter f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff
Block #1
Input Block   f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff
Output Block  ec8cdf7398607cb0f2d21675ea9ea1e4
Plaintext     6bc1bee22e409f96e93d7e117393172a
Ciphertext    874d6191b620e3261bef6864990db6ce
Block #2
Input Block   f0f1f2f3f4f5f6f7f8f9fafbfcfdff00
Output Block  362b7c3c6773516318a077d7fc5073ae
Plaintext     ae2d8a571e03ac9c9eb76fac45af8e51
Ciphertext    9806f66b7970fdff8617187bb9fffdff
Block #3
Input Block   f0f1f2f3f4f5f6f7f8f9fafbfcfdff01
Output Block  6a2cc3787889374fbeb4c81b17ba6c44
Plaintext     30c81c46a35ce411e5fbc1191a0a52ef
Ciphertext    5ae4df3edbd5d35e5b4f09020db03eab
Block #4
Input Block   f0f1f2f3f4f5f6f7f8f9fafbfcfdff02
Output Block  e89c399ff0f198c6d40a31db156cabfe
```

```
Plaintext     f69f2445df4f9b17ad2b417be66c3710
Ciphertext    1e031dda2fbe03d1792170a0f3009cee
```

1.5.8 CTR 復号

```
F.5.2 CTR-AES128.Decrypt
Key           2b7e151628aed2a6abf7158809cf4f3c
Init. Counter f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff
Block #1
Input Block   f0f1f2f3f4f5f6f7f8f9fafbfcfdfeff
Output Block  ec8cdf7398607cb0f2d21675ea9ea1e4
Ciphertext    874d6191b620e3261bef6864990db6ce
Plaintext     6bc1bee22e409f96e93d7e117393172a
Block #2
Input Block   f0f1f2f3f4f5f6f7f8f9fafbfcfdff00
Output Block  362b7c3c6773516318a077d7fc5073ae
Ciphertext    9806f66b7970fdff8617187bb9fffdff
Plaintext     ae2d8a571e03ac9c9eb76fac45af8e51
Block #3
Input Block   f0f1f2f3f4f5f6f7f8f9fafbfcfdff01
Output Block  6a2cc3787889374fbeb4c81b17ba6c44
Ciphertext    5ae4df3edbd5d35e5b4f09020db03eab
Plaintext     30c81c46a35ce411e5fbc1191a0a52ef
Block #4
Input Block   f0f1f2f3f4f5f6f7f8f9fafbfcfdff02
Output Block  e89c399ff0f198c6d40a31db156cabfe
Ciphertext    1e031dda2fbe03d1792170a0f3009cee
Plaintext     f69f2445df4f9b17ad2b417be66c3710
```

1.5.9 ECB 暗号化

```
F.1.1 ECB-AES128.Encrypt
Key           2b7e151628aed2a6abf7158809cf4f3c
Block #1
Plaintext     6bc1bee22e409f96e93d7e117393172a
Input Block   6bc1bee22e409f96e93d7e117393172a
Output Block  3ad77bb40d7a3660a89ecaf32466ef97
Ciphertext    3ad77bb40d7a3660a89ecaf32466ef97
Block #2
Plaintext     ae2d8a571e03ac9c9eb76fac45af8e51
Input Block   ae2d8a571e03ac9c9eb76fac45af8e51
Output Block  f5d3d58503b9699de785895a96fdbaaf
Ciphertext    f5d3d58503b9699de785895a96fdbaaf
Block #3
Plaintext     30c81c46a35ce411e5fbc1191a0a52ef
Input Block   30c81c46a35ce411e5fbc1191a0a52ef
```

```
Output Block 43b1cd7f598ece23881b00e3ed030688
Ciphertext   43b1cd7f598ece23881b00e3ed030688
Block #4
Plaintext    f69f2445df4f9b17ad2b417be66c3710
Input Block  f69f2445df4f9b17ad2b417be66c3710
Output Block 7b0c785e27e8ad3f8223207104725dd4
Ciphertext   7b0c785e27e8ad3f8223207104725dd4
```

1.5.10 ECB 復号

```
F.1.2 ECB-AES128.Decrypt
Key          2b7e151628aed2a6abf7158809cf4f3c
Block #1
Ciphertext   3ad77bb40d7a3660a89ecaf32466ef97
Input Block  3ad77bb40d7a3660a89ecaf32466ef97
Output Block 6bc1bee22e409f96e93d7e117393172a
Plaintext    6bc1bee22e409f96e93d7e117393172a
Block #2
Ciphertext   f5d3d58503b9699de785895a96fdbaaf
Input Block  f5d3d58503b9699de785895a96fdbaaf
Output Block ae2d8a571e03ac9c9eb76fac45af8e51
Plaintext    ae2d8a571e03ac9c9eb76fac45af8e51
Block #3
Ciphertext   43b1cd7f598ece23881b00e3ed030688
Input Block  43b1cd7f598ece23881b00e3ed030688
Output Block 30c81c46a35ce411e5fbc1191a0a52ef
Plaintext    30c81c46a35ce411e5fbc1191a0a52ef
Block #4
Ciphertext   7b0c785e27e8ad3f8223207104725dd4
Input Block  7b0c785e27e8ad3f8223207104725dd4
Output Block f69f2445df4f9b17ad2b417be66c3710
Plaintext    f69f2445df4f9b17ad2b417be66c3710
```

1.5.11 OFB 暗号化

```
F.4.1 OFB-AES128.Encrypt
Key          2b7e151628aed2a6abf7158809cf4f3c
IV           000102030405060708090a0b0c0d0e0f
Block #1
Input Block  000102030405060708090a0b0c0d0e0f
Output Block 50fe67cc996d32b6da0937e99bafec60
Plaintext    6bc1bee22e409f96e93d7e117393172a
Ciphertext   3b3fd92eb72dad20333449f8e83cfb4a
Block #2
Input Block  50fe67cc996d32b6da0937e99bafec60
Output Block d9a4dada0892239f6b8b3d7680e15674
```

```
Plaintext    ae2d8a571e03ac9c9eb76fac45af8e51
Ciphertext   7789508d16918f03f53c52dac54ed825
Block #3
Input Block  d9a4dada0892239f6b8b3d7680e15674
Output Block a78819583f0308e7a6bf36b1386abf23
Plaintext    30c81c46a35ce411e5fbc1191a0a52ef
Ciphertext   9740051e9c5fecf64344f7a82260edcc
Block #4
Input Block  a78819583f0308e7a6bf36b1386abf23
Output Block c6d3416d29165c6fcb8e51a227ba994e
Plaintext    f69f2445df4f9b17ad2b417be66c3710
Ciphertext   304c6528f659c77866a510d9c1d6ae5e
```

1.5.12 OFB 復号

```
F.4.2 OFB-AES128.Decrypt
Key          2b7e151628aed2a6abf7158809cf4f3c
IV           000102030405060708090a0b0c0d0e0f
Block #1
Input Block  000102030405060708090a0b0c0d0e0f
Output Block 50fe67cc996d32b6da0937e99bafec60
Ciphertext   3b3fd92eb72dad20333449f8e83cfb4a
Plaintext    6bc1bee22e409f96e93d7e117393172a
Block #2
Input Block  50fe67cc996d32b6da0937e99bafec60
Output Block d9a4dada0892239f6b8b3d7680e15674
Ciphertext   7789508d16918f03f53c52dac54ed825
Plaintext    ae2d8a571e03ac9c9eb76fac45af8e51
Block #3
Input Block  d9a4dada0892239f6b8b3d7680e15674
Output Block a78819583f0308e7a6bf36b1386abf23
Ciphertext   9740051e9c5fecf64344f7a82260edcc
Plaintext    30c81c46a35ce411e5fbc1191a0a52ef
Block #4
Input Block  a78819583f0308e7a6bf36b1386abf23
Output Block c6d3416d29165c6fcb8e51a227ba994e
Ciphertext   304c6528f659c77866a510d9c1d6ae5e
Plaintext    f69f2445df4f9b17ad2b417be66c3710
```

参考文献

[BDJR97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation," *Proceedings of the 38th Symposium on Foundations of Computer Science*, IEEE, 1997.

[V02] S. Vaudenay, "Security Flaws Induced by CBC Padding — Applications to SSL, OPSEC, MTLS...," *Advances in Cryptology — EUROCRYPT 2002, International Conference on the Theory and Application of Cryptographic Techniques, Amsterdam, the Netherlands, April 28–May 2, 2002, proceedings*, ed. L. Knudsen, pp. 534–546, Lecture Notes in Computer Science vol. 2332, Springer-Verlag, 2002.

[W02] D. Wagner, "New attacks on t-bit OFB and CFB modes: A cautionary note regarding IV selection", CRYPTO 2002 rump session, August 20, 2002.

[SP800-38A] M. Dworkin, National Institute of Standards and Technology, Special Publication 800-38A, 2001 Edition, Recommendation for Block Cipher Modes of Operation, Methods and Techniques, December 2001.

[ISO/IEC10116] ISO/IEC 10116:2006 (3rd edition): Information technology — Security techniques — Modes of operation for an n-bit block cipher, 2006.

[ISO/TR19038] ISO/TR19038:2006 (1st edition): Banking and related financial services — Triple DEA — Modes of operation — Implementation guidelines, 2005.

[FIPS-81] National Institute of Standards and Technology, Federal Information Processing Standards Publication 81, Des Modes of Operation, December 1980.

発行日　2009年5月14日　第1版　第1刷
発行者
・〒184-8795
東京都小金井市貫井北四丁目2番1号
独立行政法人 情報通信研究機構
（情報通信セキュリティ研究センター セキュリティ基盤グループ）
NATIONAL INSTITUTE OF
INFORMATION AND COMMUNICATIONS TECHNOLOGY
4-2-1 NUKUI-KITAMACHI,KOGANEI
TOKYO,184-8795 JAPAN

・〒113-6591
東京都文京区本駒込二丁目28番8号
独立行政法人 情報処理推進機構
（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