2009－AL－122 (7)
2009／1／30

DNA 計算による AES 暗号の解読

若月 祐介† 築地 立家†
†東京電機大学 理工学研究科

DNA コンピュータを用いれば，ナノスケールの計算素子として DNA 分子を利用することにより，超並列計算を実現することができる．実際，Adleman らは，DNA コンピュータが DES 暗号を効率的に解読できることを示した．そこで，本論文では，AES 暗号を効率的に解読するような，複数の DNA アルゴリズムを示す．ところで，AES は DES よりもはるかに解読が難しい暗号として知られている．実際，Adleman らの DES 解読アルゴリズムを単純に拡張した AES 解読アルゴリズムでは，DNA メモリコンプレックスの長さが 7 倍以上になってしまう．我々のアルゴリズムを用いれば，DNA を切り離すための制限酵素を使用することにより，高々1.3 倍の長さのメモリコンプレックスで AES コードが解読できることが示される．

DNA Computing The Advanced Encryption Standard

Yuusuke Wakatsuki† Tatsuie Tsukiji†
†Tokyo Denki University. The Science and Engineering graduate course

DNA computer utilizes DNA molecules as nanoscale computational elements to provide efficient computations for currently intractable problems. Actually, Boneh, Dunworth, Lipton showed that a DNA computer can efficiently decrypt DES encryption codes. In this paper, we show a couple of algorithms decrypting given AES codes by a series of parallel DNA operations. Since AES is known as much harder than DES to be decrypted, e.g. AES uses 128 bits, while DES uses 64 bits, of the hidden key, a simple extension of their algorithm requires DNA memory complexes of 7 times longer than the previous ones. One of our results shows that, by inserting restriction enzyme DNA cutting operations, much shorter memory complex is enough to decrypt given AES codes.

1. はじめに

DNA コンピュータを用いれば，ナノスケールの計算素子として DNA 分子を利用することにより，超並列計算を実現することができる． 実際，Adleman[1]の小規模組合せ最適化問題の解法や，Adleman ら[2]や Lipton ら[3]による Data Encryption Standard(DES)暗号の解読などは，ノイマン型計算機では処理時間が莫大になってしまうが，DNA コンピュータが効率的に解いてしまうことが知られている．さらに，近年 DNA 操作技術が飛躍的に進歩し，DNA コンピュータの実現が現実味を帯びてきている．そこで，本研究では，DNA コンピュータによる Advanced Encryption Standard (AES)暗号の解読を行う．

Adleman, Rothemund, Roweis, Winfree [2]は，スティッカーモデルの上で DES 暗号解読アルゴリズムを提案して，その効率性を検証した。本稿でも，スティッカーモデルを採用して，AES 暗号を解読するための幾つかの DNA アルゴリズムを提案して，その効率性について検証する。

2. AES 暗号方式

AES 暗号方式[4]は，コンピュータの高速化と解読ソフトの進歩による DES 暗号方式の脆弱化に伴い，2000 年に新しく採用された暗号技術の米国標準規格である．

AES 暗号の鍵サイズは 128 ビットであり，DES 暗号の 2 倍の長さとなっている．AES 暗号は，図 2.1 の流れに沿って，平分から 10 ラウンドかけて暗号文を作成する．その際，各ラウンドにおいて *RoundKey*（*RK*）と呼ばれる拡張鍵を使用して，中間暗号文 ARK_i を作成する．初期処理では，平分と RK_0 を排他的論理和して，ARK_0 を作成する．

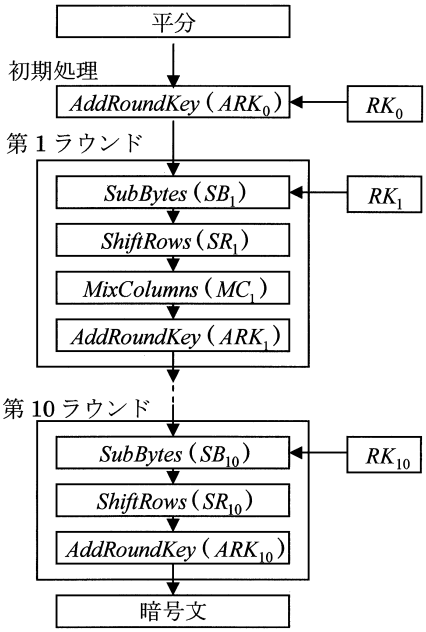


図 2.1　AES 暗号

各ラウンドでは，128 ビットの状態値ベクトルを 8 ビット毎のユニットベクトルに 16 分割した上で，以下の処理を行う．*AddRoundKey* 処理は，状態値ベクトルに対して，128 ビットの *RoundKey* 値を排他的論理和する．*SubBytes* 処理は，各ユニットベクトルを SBOX 表により変換する．*ShiftRows* 処理は，ユニットベクトル間でシフト操作する．これらの処理は，いずれも DES 暗号でも用いられている．一方，*MixColumns* 処理は，AES 暗号に特有の処理であり，多項式環 $Z_2(x)/(x^8+x^7+\cdots+x+1)$ 上の次の行列計算を実行する．ただし，$s_{i,j}$（$j=0,1,2,3$）は入力ユニットベクトル，$s'_{i,j}$ は出力ユニットベクトルを表す．また，ユニットベクトルの値である 8 ビット値は，7 次多項式の係数を表現するものとする．係数の足し算は排他的論理和で行う．

$$\begin{bmatrix} s'_{0,j} \\ s'_{1,j} \\ s'_{2,j} \\ s'_{3,j} \end{bmatrix} = \begin{bmatrix} 02 & 03 & 01 & 01 \\ 01 & 02 & 03 & 01 \\ 01 & 01 & 02 & 03 \\ 03 & 01 & 01 & 02 \end{bmatrix} \begin{bmatrix} s_{0,j} \\ s_{1,j} \\ s_{2,j} \\ s_{3,j} \end{bmatrix}$$

図 2.2 の流れに従って，鍵 RK_i から RK_{i+1} を作成する．RK_i を $RK_{i,j}$（$j=0,1,2,3$）に 4 等分割し，$RK_{i,3}$ に対して *RotWord*（シフト変換），*SubWord*（SBOX 変換），および $Rcon_i$（定数ベクトル）との排他的論理和を行い，その結果を元に各 $RK_{i,j}$ とリレー式に排他的論理和を行い，RK_{i+1} を作成する．

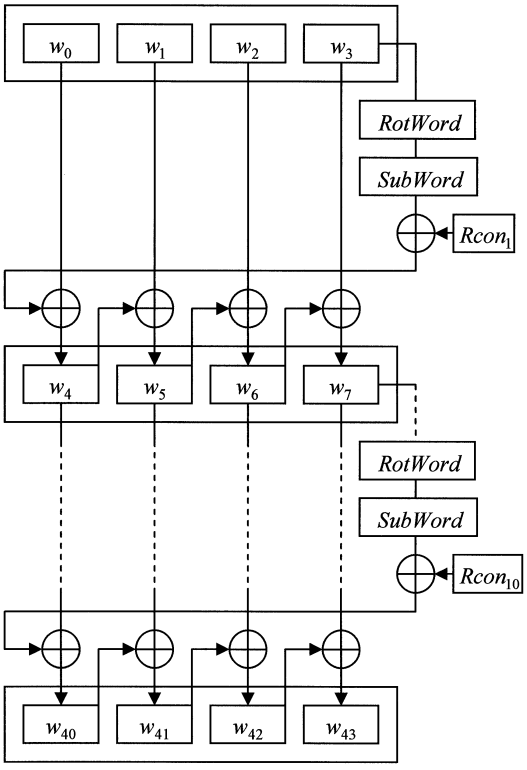


図 2.2　鍵作成の流れ

3. DNA 分子の構造

1本鎖 DNA はデオキシリボヌクレオチド(deoxyribonucleotides acid)と呼ばれるモノマーが一列に連なったポリマーのことである. デオキシリボヌクレオチドは, 図3.1のように, 1′～5′と番号付けられた一連の糖にP(リン酸基), OH(ヒドロキシル基)および塩基が結合しており, 塩基の種類に応じて, アデニン(A : adenine)とグアニン(G : guanine), シトシン(C : cytosine)とチミン(T : thymine)の4種類がある.

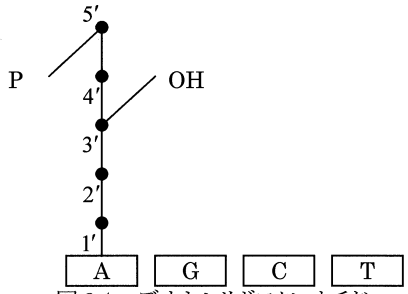


図 3.1 デオキシリボヌクレオチド

DNA の結合には, 5′-リン酸基と3′-OH 基によるホスホジエステル結合と, 塩基同士による水素結合があり, 前者が後者よりも強力である. 水素結合では, A は T, G は C とのみ結合し, この性質をワトソン・クリック相補性という.

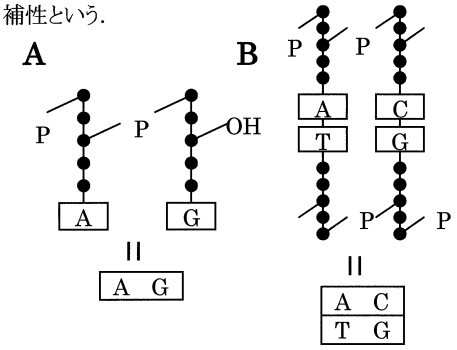


図 3.2 A:ホスホジエステル結合 B:水素結合

さらに, ワトソン・クリック相補性を利用した水素結合において, ミスマッチングを許すことにより結合力を弱めることができる. また, DNA 分子の代わりに, マッチング結合力およびミスマッチング反発力がより強い PNA 分子を使うことにより, 以下の順に結合を強化することができる. ミスマッチング結合 PNA/DNA→ミスマッチング結合 DNA/DNA → DNA/DNA → PNA/DNA→PNA/PNA. 融解温度も, この順番で高くなる.

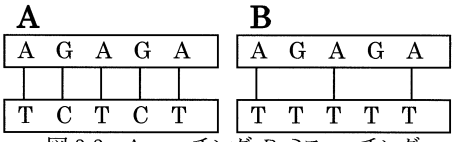


図 3.3 A:マッチング B:ミスマッチング

4. スティッカーモデル

DNA 分子に情報を格納するための手法として, 図4.1で表わされるようなメモリコンプレックスとスティッカーと呼ばれる一本鎖DNA分子を用いる[6,7]. 図4.1のメモリコンプレックスは, 20 個のヌクレオチドがホスホジエステル結合されており, A と G からなるプリンブロックと, C と T からなるピリミジンブロックが交互に配置されている. 各ブロックと相補的な関係にあるブロックをスティッカーと呼ぶ. スティッカーが結合している部分は1, そうでない部分は0を表す. したがって, 図4.1の例は0101を表す.

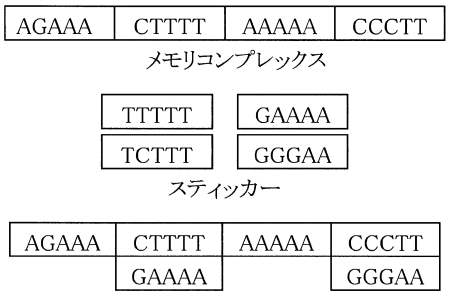


図 4.1 メモリコンプレックス例

5. DNA 分子操作のための準備

実験道具として用意するものは, 莫大な量の DNA 分子が入った多数のチューブとそれらを支えるラック, チューブを並列操作するロボットアーム, およびロボットアームに指令をだすマイクロプロセッサーである.

6. DNA アルゴリズム

本論文では, 表6.1に示される特徴をもつ3種類のメモリコンプレックス上で, DNA 分子を操作するアルゴリズムを提案する. モデル2および3のブロック数は,

Adlemanらのモデル[2]の579ブロックと同程度である. そこで, ブロック長もAdlemanらのモデルに合わせて20 塩基とする. 一方, モデル1においては, ブロック数が約 10 倍となるため, ブロック長も長くする必要がある.

また, 128ビットの鍵の作成についても, Liptonらのアルゴリズムと同様である. すなわち, 初期状態のメモリコンプレックスが入っているチューブをAとBの二つに分けて, チューブBにスティッカーを加えることにより, 最初の128個のブロック $B_1, B_2, \cdots, B_{128}$ に1を書き込む. 一方, チューブAの対応するブロックは0のままである. その後, チューブBにチューブAを加えて溶液を加熱・冷却をすることで, スティッカーをランダムにリアニールさせればよい. これにより, 2^{128} 通りの鍵をもつメモリコンプレックスが生成される.

AES メモリコンプレックス		
	ブロック数	特徴
モデル 1	5378 ブロック	DES メモリコンプレックスの拡張モデル.
モデル 2	641 ブロック	PNA_2 / DNA の 3 重鎖結合を利用してクリアするモデル
モデル 3	645 ブロック	制限酵素を利用してクリアするモデル

表 6.1 各モデルの特徴

6.1. モデル 1 の構成と操作

モデル1のメモリコンプレックスの構成とメモリー値の割り当てを図 6.1.1 に示す. B_0 は, メモリコンプレックスとスティッカーを区別するために使用する. $B_1, \ldots, B_{5377}$ の全てのブロックを DNA 分子で構成する. 最後の 128 ブロックに対しては, ミスマッチング DNA スティカーを用意し, それ以外のブロックに対してはマッチング DNA スティッカーを用意する. このとき, 温度操作により, マッチングスティッカーをはがすことなく, ミスマッチングスティッカーだけをクリアすることが可能となる. 本稿では, このクリア方法をミスマッチングクリアと呼ぶ.

図 6.1.1 モデル1のメモリー値割り当て

次に, DNA 操作アルゴリズムの説明を行う. 初期ラウンドが終了した時点で, 鍵 RK_0 と初期中間暗号文 ARK_0 のみが保存されており他のブロックは 0 である.

各第 i ラウンド($i = 1, \ldots, 10$)においては, 以下の5ステップの処理を行う. ただし, 第 10 ラウンドでは, ステップ 2 の *MixColumns* 処理は行わず, ステップ 5 では *MixColumns* 値の代わりに $SB \cdot SR$ 値を用いる.

ステップ 1. ARK_{i-1} 値に対する *SubByte* 処理と *ShiftRows* 処理を行い, 結果を保存する.

ステップ 2. ステップ1の結果に対する *MixColumns* 処理を行って, 結果を保存する. その際の *MC* 計算途中値をメモ用ブロックに一時保存する.

ステップ 3. RK_{i-1} 値の最後の 32 ビットに対して *RotWord*, *SubWord*, 及び⊕*Rcon* 処理を行い, 結果をメモに保存する.

ステップ 4. RK_{i-1} 値に対してステップ 3 の結果から始めてリレー式に排他的論理和計算を行うことにより, RK_i 値を逐次的に作成して, 保存する.

ステップ 5. *MixColumns* 値と RK_i 値の排他的論理和により ARK_i 値を作成して, 保存する.

最終ラウンド終了時に暗号文 ARK_{10} が作成されている. 各ラウンドにおいて逐次作成される中間暗号文, 中間鍵, *SubByte* · *ShiftRows* 値および *MixColumns* は, ラウンドが終了してもクリアされることがないため, 結果として, メモリコンプレックスは非常に長くなる.

第 1～9 ラウンドにおいては, メモ用ブロックが 5 回

のクリアされるため，合計で 45 回のミスマッチングクリアが発生することになる．最終ラウンドでは，メモ用ブロックをクリアする必要はない．

6.2. モデル 2 の構成と操作

モデル 2 のメモリコンプレックスの構成とメモリー値の割り当てを図 6.2.1 に示す．このように，同一のブロック領域に異なる種類のメモリー値を割り当てる．メモリコンプレックスとスティッカーのブロック構成はモデル 1 と同様である．

さらに，中央の各 $B_{129},\cdots,B_{512}$ ブロックに対するスティッカーの切り離しのために，PNA_2 / DNA の 3 重鎖結合[5,6]を利用する．すなわち，切り離したい DNA に対して，それと相補的な一本鎖 PNA を投入すると，PNA_2 / DNA の 3 重鎖結合が形成されてスティッカーの結合力が弱まり，切り離すことができる．本稿では，このクリア方法を PNA クリアと呼ぶ．最後の 128 ブロックに対しては，ミスマッチングスティッカーを利用した温度操作クリアをおこなう．

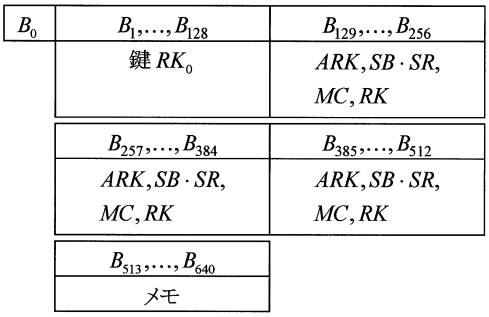


図 6.2.1 モデル 2 のメモリー値割り当て

次に，DNA 操作アルゴリズムの説明を行う．初期ラウンド処理までは，モデル 1 と同様である．また，第 10 ラウンドも同様なので，以下では，第 i ラウンド ($i=1,\ldots,9$)の処理について述べる．

ステップ 1. ARK_{i-1} 値に対する *SubByte* 処理と *ShiftRows* 処理を行い，結果を保存する．最後に，ARK_{i-1} 値を PNA クリアする．

ステップ 2. ステップ 1 の結果に対する *MixColumns* 処理を行って，結果を保存する．その際の計算途中値をメモ用ブロックに一時保存する．最後に，ステップ 1 で保存した $SB\cdot SR$ 値を PNA クリアする．

ステップ 3. RK_{i-1} 値の最後の 32 ビットに対して *RotWord*，*SubWord*，及び⊕*Rcon* 処理を行い，結果をメモ用ブロックに一時保存する．

ステップ 4. RK_{i-1} 値に対してステップ 3 の結果から始めてリレー式に排他的論理和計算を行うことにより，RK_i 値を逐次的に作成して，保存する．最後に，RK_{i-1} 値を PNA クリアする．ただし，初期鍵である RK_0 値はクリアしない．

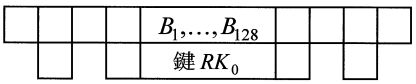
ステップ 5. *MixColumns* 値と RK_i 値の排他的論理和により ARK_i 値を作成して，保存する．最後に，*MixColumns* 値を PNA クリアする．

各ラウンドにおいて作成される中間暗号文，中間鍵，*SubByte*·*ShiftRows* 値および *MixColumns* 値 は，上記のように必要がなくなった時点でクリアされるので，同一ブロック領域に異なる種類の値を次々と保存することができる．結果として，メモリコンプレックスは最適に短くすることができる．

ミスマッチングクリアの実施個所および回数は，モデル 1 と同様である．PNA クリアの回数は，第 1 ラウンド 3 回，第 2～9 ラウンドは各 4 回，第 10 ラウンドは 2 回となり，計 37 回である．

6.3. モデル 3 の構成と操作

モデル 3 のメモリコンプレックスの構成とメモリー値の割り当てを図 6.3.1 に示す．また，操作の途中で追加投入されるメモリコンプレックスを図 6.3.2 に示す．



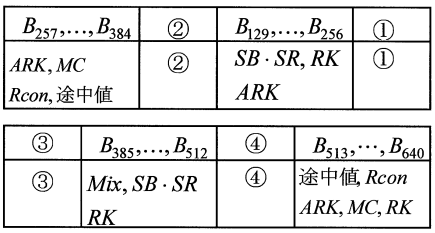


図 6.3.1 モデル 3 の構成とメモリー値の割り当て

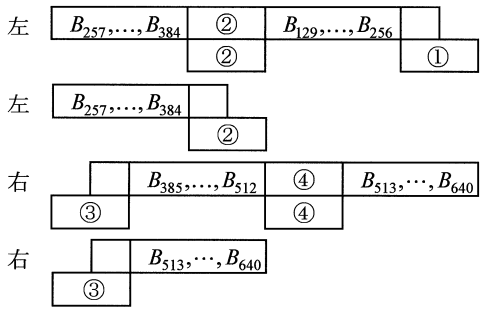


図 6.3.2　追加投入用のメモリコンプレックス

制限酵素	認識部位	切断
EcoRI ①	5′－GAATTC－3′ 3′－CTTAAG－5′	5′－G　　AATTC－3′ 3′－CTTAA　　G－5′
BamHI ②	5′－GGATCC－3′ 3′－CCTAGG－5′	5′－G　　GATCC－3′ 3′－CCTAG　　G－5′
KpnI ③	5′－GGTACC－3′ 3′－CCATGG－5′	5′－GGTAC　　C－3′ 3′－C　　CATGG－5′
SacI ④	5′－GAGCTC－3′ 3′－CTCGAG－5′	5′－GAGCT　　C－3′ 3′－C　　TCGAG－5′

表 6.3.1 制限酵素([7])

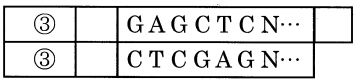
図中の全てのブロックはDNA分子で構成する.

スティカーモデル用のブロック以外に, 切断用のブロック(①～④)を用いる. 表 6.3.1 に指定された制限酵素を投入することにより, 対応する切断箇所で下記のように切断と結合を行うことにより, メモリコンプレックスをクリアすることができる. 本稿では, このクリア方法を, ①クリア($i=1,2,3,4$)と呼ぶ.

酵素④により, 対応する分離ブロックを切断する.

相補的なメモリコンプレックスを投入して, 温度を下げることにより, 水素結合させる.

DNAリガーゼを投入して, 接合部分でホスホジエステル結合させれば, 完成.



次に, DNA 操作アルゴリズムの説明を行う. 初期ラウンド処理までは, モデル 1 と同様である.

第 i ラウンド($i=2,\ldots,9$)においては, 以下の 5 ステップの処理を行う. 偶数ラウンドと奇数ラウンドでは, 切断クリアタイプが異なるため, 奇数ラウンドにおける切断タイプを()内に表記する. なお, 第1ラウンドと第10ラウンドについては, 切断クリアの箇所が他のステージと異なるため, 相違点を別個に付記する.

ステップ 1.　ARK_{i-1} 値に対する *SubByte* 処理と *ShiftRows* 処理を行い, 結果を保存する. 最後に, ARK_{i-1} 値を②(④)クリアする.

ステップ 2.　ステップ 1 の結果に対する *MixColumns* 処理を行い, 途中結果の保存とクリアを繰り返し, 最後に *SB*・*SR* 値と最終 MC 値をクリアする. ③(①)クリアが 4 回行われる.

ステップ 3.　RK_{i-1} 値の最後の 32 ビットに対して *RotWord*, *SubWord*, 及び⊕*Rcon* 処理を行い, 結果(*Rcon* 値と呼ぶ)を保存する.

ステップ 4.　RK_{i-1} 値に対して, *Rcon* 値から始めてリレー式に排他的論理和計算を行うことにより, RK_i 値を逐次的に作成して保存する. 最後に, *Rcon* 値を④(②)切断クリアする.

ステップ 5.　*MixColumns* 値と RK_i 値の排他的論理和により ARK_i 値を作成して, 保存する. 最後に, *MixColumns* 値と RK_{i-1} を①(③)切断クリアする.

第 1 ラウンドでは, ステップ1で ARK_0 値を①切断クリアする代わりに, ステップ 2 の最後に ARK_0 値と *SB*・*SR* 値を同時に①クリアする. また, ステップ 4 で *Rcon* 値を切断クリアする代わりに, ステップ 5 の最後に *Rcon* 値と *MC* 値を同時に③クリアする. さらに, ステップ 2 では, *MC* 途中値を 4 回だけ④クリアする.

第 10 ラウンドでは, ステップ 1 で ARK_9 値を②クリ

アし，ステップ 2 は全く実行せず，ステップ 3 はそのまま実行し，ステップ 4 で切断クリア以外を実行し，ステップ 5 では $Rcon$ 値と RK_9 値を同時に①クリアしてから，ARK_{10} 値を保存する．

6.4. 初期鍵の検知

モデル 1～3 のどのアルゴリズムを用いても，初期鍵と，それを使用して作成した AES 暗号文のペアを，DNA 分子上に出力することができた．ただし，同一試験管内に，全パターンの初期鍵に対応する DNA 分子が存在するので，正しい暗号文をもとにして，正しい初期鍵をもつ DNA を検出する必要がある．そのためには，次節で述べる分離演算を行うことにより，DNA 上の暗号文と正しい暗号文のマッチングを行えばよい．

7. 分離演算・結合演算・まとめ演算

6 節の DNA 操作法は，分離・結合・まとめ演算とよばれる DNA 操作の基本演算から成り立っている．本節では，これらの各操作法について，概略を述べる．

分離演算． 図 7.1 の分離演算例では，ブロック B_1 と B_2 の値を分離している．分離演算子チューブは，1 回目の分離で 1 本，2 回目の分離で 2 本使用されており，合計 3 本である．分離されたデータを保管するためのデータチューブは 6 本使用される．演算回数は，2 回目の分離が並列操作のため，2 回となる．一般に，n ビットの分離のためには，2^n-1 本の分離演算子チューブ，$3\times 2^{n-1}$ 本のデータチューブ（再利用可能）が必要となる．演算回数は，n 回である．

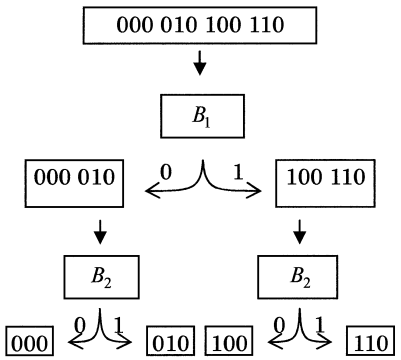


図 7.1 分離演算

結合演算． 関数演算を行うために，メモリコンプレックスに適切なスティッカーを結合させることにより，関数グラフを作成する演算．図 7.2 の結合演算では，最初の 2 ビットの排他的論理和を最後のビットに書き込んでいる．

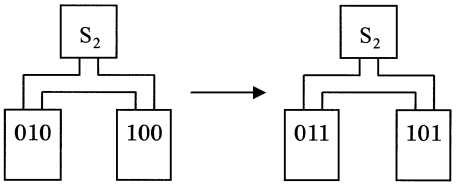


図 7.2 結合演算

まとめ演算． 複数の試験管の中身を一本の試験管にまとめる，試験管操作．図 7.3 では，図 7.2 の結合演算の結果のデータに対するまとめ演算である．

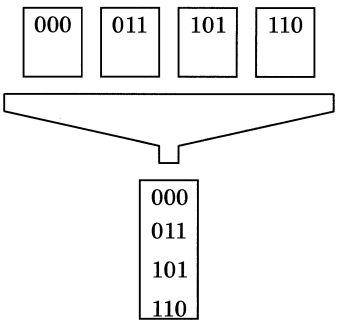


図 7.3 まとめ演算

8. チューブ数とステップ数

6 節で提案した各アルゴリズムを，7 節で述べた基本演算によって組み上げると，下記の表 8.1 のようなチューブ数とステップ数が得られる（詳細は，紙面の都合上，割愛する）．なお，Lipton らの DES 解読アルゴリズムも，同じ基本演算に基づいて評価されており，チューブ数の総数は 1369 本，ステップ数は 6719 となっている．モデル 2 及び 3 のアルゴリズムでは，チューブ数が同程度，ステップ数が 20 倍程度となっている．一般に，チューブ数は，DNA 演算装置の規模に直結するため，ステップ数より重要なファクターと考えられている．

AES メモリコンプレックス				
	ステップ	分離演算子チューブ	データチューブ	スティッカーチューブ
モデル 1	111853 ステップ	6880 本	384 本	4280 本
モデル 2	111890 ステップ	1048 本	384 本	640 本
モデル 3	111872 ステップ	1600 本	384 本	640 本

表 8.1　チューブ数とステップ数

9. まとめ

モデル 1 は, メモリコンプレックスが非常に長いため, 現状の DNA 操作技術では, 壊れやすい, 試験管にはいらない, などの欠点をもつ. 一方, モデル 2 で用いた PNA_2 / DNA の 3 重鎖結合によるクリア技術[5,6]は, 近年急速に精度を向上させているが, 他のスティカーの状態に影響を与えずに, 必要なスティカーを高い精度で何度もクリアすることは, 未だに困難である. モデル 3 では, 数百ブロックの長さをもつ DNA 分子を制限酵素によって切断した後に, 接合している. 巨大な DNA の切断に関する技術は確立されているが, 巨大な DNA を一連の DNA 操作に耐えうる程に強力に接合する技術は, 未だに確立されていない.

AES 暗号方式は, DES 暗号方式よりはるかに強度が高く, 未だに解読されていない. 本稿では, DNA 操作技術の進歩により, ある程度大きな DNA 分子の切断や結合を十分に高い精度で操作できるようになれば, AES 暗号方式もある程度効率的に解読可能であることを示すことができた.

参考文献

[1] L. M. Adleman : Molecular Computation of Solutions to Combinatorial Problem. Science, volume 266(5187), pages 1021-1024, 1994.

[2] L. M. Adleman, P. W. K. Rothemund, S. Roweis, E. Winfree : On Applying Molecular Computation To The Data Encryption Standard. Journal of Computational Biology, volume 6(1), pages 53-64, 1999.

[3] D. Boneh, C. Dunworth, R. J. Lipton : Breaking {DES} Using a Molecular Computer. DNA based computers, DIMACS 27, pages 37-66, 1995.

[4] J. Daemen and V. Rijmen : Rijndael for AES. AES Candidate Conference, pages 343-348, 2000.

[5] P. E. Nielsen, M. Egholm, R. H. Berg, O. Buchardt : Sequence-Selective Recognition of DNA by Strand Displacement with a Thymine-Substituted Polyamide. Science, volume 254(5037), pages 1497-1500, 1991.

[6] S. Roweis, E. Winfree, R. Burgoyne, N. V. Nickolas, M. F. Goodman, P. W .K. Rothemund, L. M. Adlman : A Sticker Based Model for DNA Computation. Journal of Computational Biology. Journal of Computational Biology, volume 5(4), pages 615-630, 1998.

[7] G. Paun, A. Salomaa, G Rozenberg : DNA Computing, Springer, 1998.