2J2-1

# ライフゲームの性質を利用したファイルの暗号化に関する研究

## Proposal of file encryption method using dynamics of the Game of Life

井上 聡[*1]
Satoru Inoue

[*1] 埼玉工業大学 工学部 情報システム学科
Department of Information Systems, Faculty of Engineering, Saitama Inst. of Tech.

In our study, we propose the file encryption method based on the complex dynamics of the Game of Life. The Game of Life shows the remarkable complex but fascinated dynamics in spite of its quite simple rule. We apply its complexity to the file encryption method for constructing the strong encrypting algorithm. We confirm that encryption and decryption using our system are completely reversible. Moreover we make sure that even slight difference in encryption key is not allowed to complete decryption. It means that our encryption method has remarkable strength in encryption.

## 1. はじめに

ライフゲームは，1970 年にイギリスの数学者 John Horton Conway によって考案されたシミュレーションゲームである[Berlekamp 1982]. ゲームのフィールドは 2 次元の格子状のセルと呼ばれるマス目で構成されており，各セルは生(1)または死(0)のどちらかの状態をもつ. ある 1 つのセルに注目したとき，そのセルの次世代の状態は，次の 3 つのルールによって決定される.

・自身が死の状態のとき，周囲に生のセルが 3 つあるなら次世代では生となる.

・自身が生の状態のとき，周囲に生のセルが 2～3 つあるなら次世代でも生となる.

・上記以外なら，次世代では死となる.

ライフゲームには，「簡単なルールで更新するにも拘らず，多様に変化していく」という性質がある. 本研究では，この性質を利用する暗号化方法を提案する.

## 2. 暗号化システムについて

### 2.1 暗号化アルゴリズムの概要

本研究で提案する暗号化アルゴリズムのブロックダイアグラムを図1に示す. このシステムは鍵スケジュール部とデータ攪拌部に分かれている.

鍵スケジュール部では暗号化処理に先立ち，鍵を拡張しサブ鍵の生成を行う. ここで生成したサブ鍵は暗号化処理の際に使用する.

データ攪拌部では，サブ鍵を使って平文を暗号文に変換する処理を行う. 具体的には，入力データを左右半分に分け，右半分と鍵スケジュール部で生成したサブ鍵を F 関数に通す. F 関数の出力と左半分の XOR を行う. この操作を左右入れ替えながら 10 回繰り返すことで暗号化処理が終了する. 復号するときは，全く逆の操作を行う. なお，F 関数については 2.3 で述べるが，入力された右半分とサブ鍵のデータを混ぜ合わせる処理を行っている. このように，2 分割したデータを交互に F 関数を通過させる処理の方式を Feistel 構造という.

連絡先:埼玉工業大学 工学部 情報システム学科，
埼玉県深谷市普済寺 1690，inoue@sit.ac.jp

### 2.2 鍵スケジュール部

鍵スケジュール部では，128bit の鍵をもとにして 64bit のサブ鍵を 10 個生成する.

鍵データを 16×8 のライフゲームとして表現し，これをライフゲームのルールに従って 10 ステップ分更新する. このとき，1 ステップ更新するごとに以下の操作を行い，サブ鍵を作っていく. ライフゲームを中央で半分に区切り，8×8 の平面を 2 つ作る. この 2 つの平面の XOR を行った結果をサブ鍵として暗号化処理に用いる.

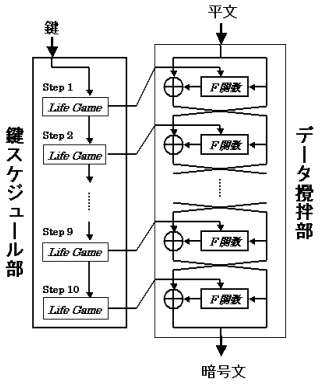


**図1. 暗号化システムの全体像**

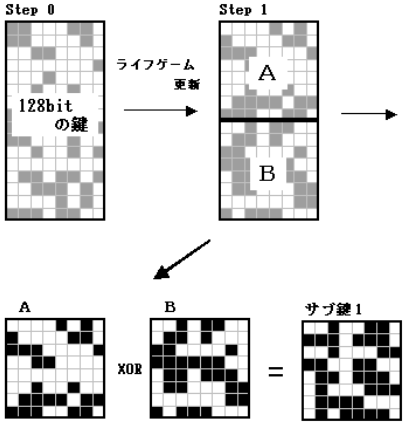


**図 2. 鍵スケジュール部の流れ**

### 2.3 F 関数

F関数は，データの攪拌を行う暗号化処理の中核となる部分である．入力データの右半分とラウンドごとのサブ鍵との XOR をとり，それをライフゲームのルールに従って 1 ステップ更新する．これを F 関数の出力とし，入力データの左半分と XOR 演算を行う．

## 3. ファイルを暗号化した結果について

本研究で提案している暗号化方法は，ファイルのバイナリデータを直接暗号化しているため暗号化可能なファイルの種類は特に問わない．ここではテキストファイルとビットマップファイルの 2 種類のファイルを暗号化した結果を示す．また，この方法での暗号化はファイルヘッダ部すら暗号化されているため，元ファイルの種別等の基礎データも秘匿される．ここではビットマップデータ部分がどのように攪拌されているかを示すため，あえてファイルヘッダは暗号化前のものと書き換えて表示している．

### 3.1 テキストファイルを暗号化した結果

図 3(上)のテキストファイルに対して暗号化処理を行った結果が図 3(下)である．暗号化後のデータから暗号化前のデータを読み取ることは極めて困難である．

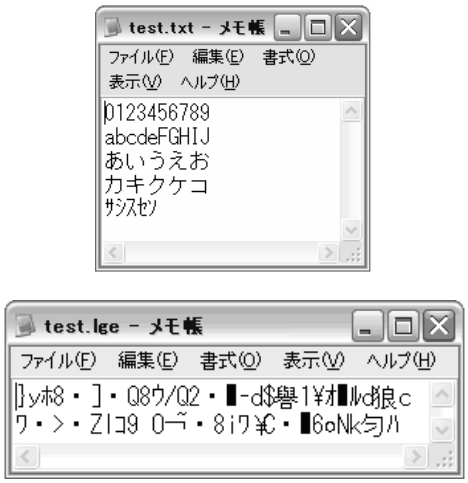


**図 3. テキストファイルの暗号化例**

### 3.2 ビットマップファイルを暗号化した結果

もう 1 つの例としてビットマップファイルを暗号化した例を閉めず．図 4(左)は暗号化前のファイルで，図 4(右)が暗号化後のファイルである．図 4 が示すように，双方を比較すると暗号化された画像から元のファイルの画像を推測することは不可能といえる．

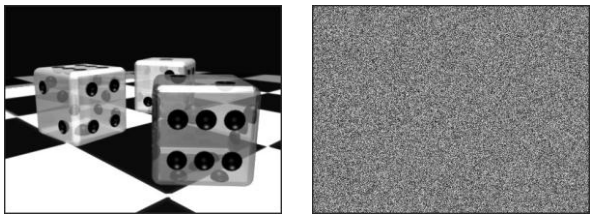

**図 4. ビットマップファイルの暗号化例**

## 4. ファイルを復号化した結果について

暗号化されたファイルも復号ができなければ，そのシステムはまったく意味のないものといえる．そこで暗号化されたデータを正規鍵で正しく復号することができるかを検証した．また，システムの秘匿性を確認するために，正規鍵から 1 ビットだけ反転させた鍵をつかって復号させた場合に，暗号化前ファイルが読み取られてしまう可能性がないかを確認するためである．

図 5(左下)を見ると正規鍵をつかえば，暗号化したデータを復号できることが確認できる．

また，暗号化したデータを 127 ビット一致している鍵を用いて復号化しても元のデータには復号できず秘匿性が確保できていることがわかる(図 5(右下))．

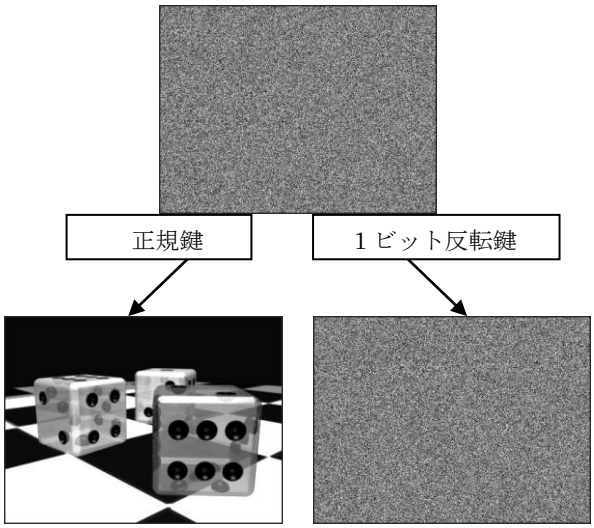


**図 5.暗号化データを復号した例**

## 5. まとめと考察

本研究ではライフゲームを利用したファイル暗号化システムというものを提案，構築した．そして暗号化に利用した正規鍵を利用すれば暗号-復号が可能である可逆性があることを確認した．本研究でのデータ暗号化は元データと鍵のビットを XOR 演算をすることによって行っているが，この場合暗号化元データと同じデータ長の鍵が必要となってしまう．ファイルのサイズが肥大化している今日，あらかじめ元データと同サイズの鍵を乱数を利用して用意することは現実的とはいえない．その解消のためにライフゲームを利用して，元々少ないデータサイズから必要なサイズの鍵を作成したことがシステムとしては新規性のある部分である．ただ，その手法により作成された鍵データのビットに偏在が存在すれば，暗号の脆弱性につながりかねない．

そのような想定にもとづき，計算量的観点，さらには統計量的観点からその安全性の検証を進めている．

### 参考文献

[Berlekamp 1982] Elwyn Berlekamp, John Conway and Richard Guy : Winning Ways for your Mathematical Plays, Academic Press ,1982.

[Poundstone 1987] William Poundstone :Recursive Universe, Oxford Paperbacks , 1987.