2J3-2

Feistel 構造を用いたファイル暗号化アプリケーションの開発

Development of file encryption application using the Feistel structure

小林清哉[*1]
Seiya Kobayashi

井上聡[*1*2]
Satoru Inoue

[*1] 埼玉工業大学大学院　工学研究科
Graduate School of Engineering, Saitama Institute of Technology

[*2] 埼玉工業大学
Saitama Institute of Technology

In this paper, by focusing on the characteristics of the Feistel structure which is a typical example of a common key block encryption method, we propose an encryption method using the Game of Life, which is one of the cellular automaton in the round function sections. Our proposed system can realize not only the encryption but also the decryption of music, image and text file whose format is commonly used.

1. はじめに

近年，情報技術(IT)の発展によりインターネット上での犯罪が増加しており，それに伴うセキリュティ対策の一つとしてファイル暗号化が用いられている．その代表的なもので AES と呼ばれるアメリカの暗号規格が存在する．しかし，その暗号技術がいつまでも安全に使用できるとは限らない．なぜなら暗号の歴史は暗号開発者と暗号解読者との知恵比べであり，時代と共に新しい暗号アルゴリズムの提案がなされているからである．ただし，解読不可能であることが数学的に証明されたワンタイムパッド暗号と呼ばれる暗号方式が存在するが平文と同等もしくはそれ以上の情報量を持った鍵が必要となるため例外とする.私の所属している研究室では Feistel 構造とライフゲームを利用した暗号化の研究が数年にわたり行われてきた．しかし，行われてきた研究はすべて画像やテキストなどデータサイズが小さいものであり，実行には特定の手順を踏まなければならなかった．そこで本研究では先行研究では行われていない，より実用的なデータサイズである音楽ファイル暗号化の検証と誰でもブラウザ上でファイルの暗号化を行えるシステムの作成を目指す．

2. ブロック暗号

ブロック暗号とは固定長のデータを一つのブロックとして暗号化をすることである．代表的な例として Feistel 構造と AES にも用いられている SPN 構造が挙げられる．前者は入力データを 2 つに分け，F 関数を交互に通過させる構造をしている(図 1)．後者は換字変換と転置変換を組み合わせたものをラウンド関数とする構造である(図 2)．

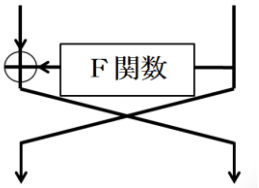

図 1　Feistel 構造

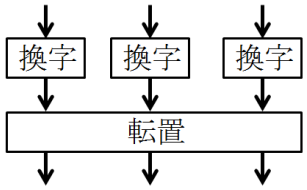

図 2　SPN 構造

連絡先：小林　清哉，埼玉工業大学大学院，埼玉県深谷市普済寺1690，048-585-2521，seiyazyann@gmail.com

3. 研究内容

本研究で行った暗号化の方法は，初めにユーザが指定したファイルの読み込みを行う．この時，読み込んだファイルが既に暗号化済みのファイルなのかをパリティビットを参考に判断する．次にユーザが任意のパスワード(半角数字・最大 17 桁)を指定する．これは後にサブ鍵の生成時に用いられる．その後暗号化を行いユーザに暗号化されたファイルを返す．その際ファイルデータ末尾にパリティビットの追加を行うことでファイルの読み込み開始時，既に暗号化済みのファイルであった場合は復号化プログラムに移行させるシステムの作成を行った．復号化の流れは暗号化時とほぼ同じだが最後にパリティビットの削除を行いユーザにファイルを返す．また，復号化の際にユーザが暗号化時とは異なるパスワードを入力した場合でも復号化処理が行われ，元のデータとはかけ離れた出力がされる．以上が暗号，復号化方法である．ただしこの方法では暗号化した際，ファイルのデータと共にファイル書式の情報までもが暗号化されてしまう．つまり，ファイルを閲覧することが困難になり，攻撃者に対して暗号化済みのファイルだと教えているようなものである．そこで本研究ではファイル形式の情報が記載されているファイルヘッダを残した状態で暗号化を行い，攻撃者に対してのカモフラージュとして暗号化済みファイルの閲覧を可能にした．

4. 暗号化手順

4.1 暗号化アルゴリズムの概要

本研究で用いられている暗号化及び，復号化処理の方法は読み込んだデータから 32×16(512bit)ずつ取り出し順次暗号化を行う．また，そのアルゴリズムはデータ撹拌部と鍵スケジュール部に分かれている．前者は主に Feistel 構造を用いており，処理の中核を担っている．後者はライフゲームを用いてF関数に送るサブ鍵の生成を行っている．

4.2 データ撹拌部

データ撹拌部では取り出した入力データ 64byte を左右 2 つのブロックに分ける．そして鍵スケジュール部で更新されたサブ鍵と右ブロックのデータをF関数にかけ，その出力データと左ブロックのデータを XOR 演算する．これと最初の右ブロックを組み合わせたものを次のラウンドの入力データとする．以上の工程を 10 回繰り返すことにより暗号化，全ての工程を逆に進めることにより復号化を行っている(図 3)．

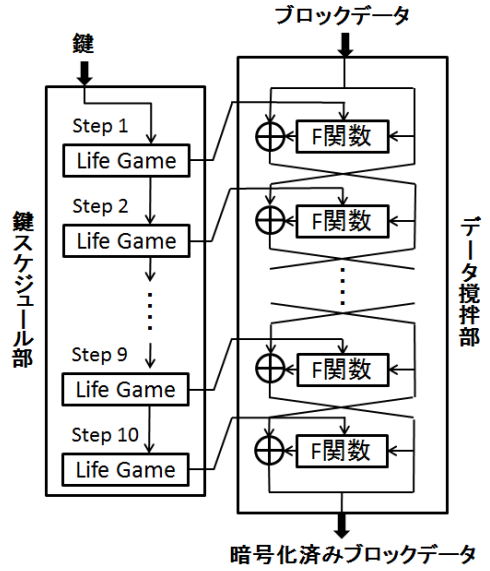

図 3 暗号化システムの模式図

4.3 鍵スケジュール部

ユーザが指定したパスワードを 2 進数に変換させたものと，こちら側であらかじめ設定した 32×16 の 2 値の数列(512bit)との XOR 演算を行うことで仮のサブ鍵を作成する．その仮のサブ鍵をライフゲームとして表現し，ライフゲームのルールに従い 10 ステップの更新をする．この時 1 ステップ毎にサブ鍵の作成を行う．図 4 では例として 8×8 のサブ鍵作成時の流れを示す．

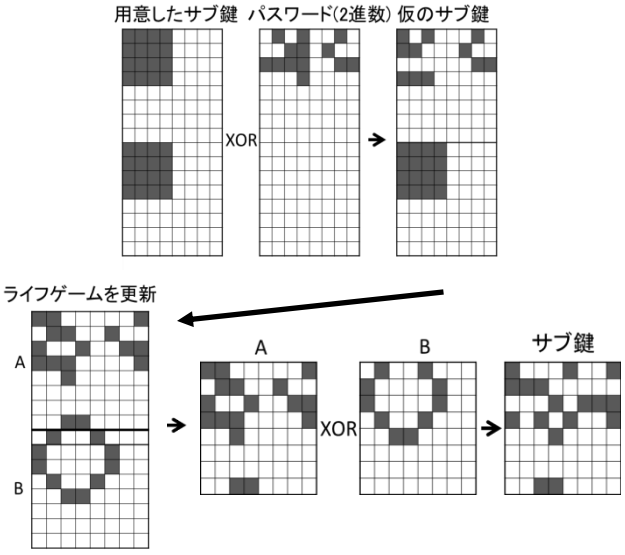

図 4 サブ鍵(8×8)作成時の流れ

4.4 F関数

F関数は，データ撹拌を行う暗号処理の核となる部分である．入力データの右ブロックとラウンドごとのサブ鍵を XOR 演算し，それをライフゲームの規則により 1 ステップ更新する．これがF関数の出力となる．

5. 研究結果

ファイル形式は様々なものがあるが今回は特に，音楽ファイルの中でも一般的なフォーマットの一つである WAV ファイルを暗号化，復号化できるシステムの作成を行った．図 5 が暗号化前のバイナリデータで図 6 が暗号化後のバイナリデータである．図 6 のファイルヘッダ(44byte)以降のバイナリが図 5 の暗号化前のファイルと比べて異なった配列になっていることからファイルの形状を維持したまま暗号化されていることが確認できた．また，そのデータを復号化した結果が図 7 である．図 5 と比較して同じ配列になっていることから復号化が成功していることが分かる．

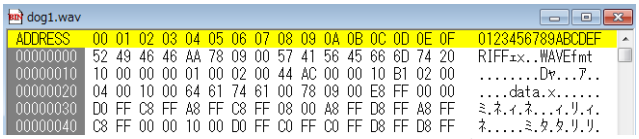

図 5 ファイル暗号化前のバイナリデータ

図 6 ファイル暗号化後のバイナリデータ

図 7 ファイル復号化後のバイナリデータ

6. 考察

本研究で提案するシステムは，暗号化の際，カモフラージュとしてファイル形式の情報を保持したままにしているが，暗号化ファイルを再生すると雑音が流れるだけで攻撃者に対しての効果が薄いことが問題として挙げられる．そこで暗号化後のファイルを特定の音情報に変化させることで秘匿性を上げることが可能だと考えられる．また現在，ファイルヘッダを残した暗号化プログラムは WAV ファイルのみに対応しており，各書式に対応した暗号化，復号化プログラムの作成が今後の課題である．

参考文献

[岡本 02] 岡本栄司: 暗号理論入門(第 2 版)，共立出版株式会社，2002.

[結城 03] 結城浩: 暗号技術入門-秘匿のアリス，ソフトバンクパブリッシング株式会社，2003.

[辻井 02] 辻井重男，岡本栄治: 暗号のすべてユビキタス社会の暗号技術，電波新聞社，2002.

[William 03] William Poundstone，有沢誠: ライフゲイムの宇宙[新装版]，日本評論社，2003.

[辻 00] 辻秀一: 暗号技術利用ハンドブック(第 2 版)，電子商取引実証推進協議会セキュリティ WG，2000.

[角尾 01] 角尾幸保，太田良二，宮内宏，中村勝洋: 統計的手法に基づく暗号強度評価支援システム，電子情報通信学会論文誌，2001.