プロジェクト　最終レポート

14k0114　　木下 和也

◆ 1．秘密分散法とは

情報を暗号化する際、複数のシェアと呼ばれるものにファイルを分割し、情報が漏れないようにする。復元の際は一定数以上のシェアを集めることにより復元する。また、この一定数より少ない数のシェアが漏洩しても元の情報は漏洩していないことが保証されているのも特徴である。

◆ 2．今回実装する秘密分散法

今回は(k, n)閾値型秘密分散法を java で実装する。(k, n)閾値型秘密分散法とは暗号化するシェアの数が n 個、復元する際に必要な数が k 個である秘密分散法である。

今回(3, 5)閾値型秘密分散法を考える。まず 1 個のファイルが 5 個のシェアに分割される。復元する際には 3 個以上のシェアを集める必要がある。つまり、5 個のシェアのうち 2 個のシェアが漏洩しても元のファイルを復元することはできない。

実装するにあたり$\mathrm{k}=3,\mathrm{n}=5,\mathrm{p}=257$、秘密は 0 以上 p 未満の値とする。

◆ 3．分割方法と復元方法

シェアの作成方法は以下のとおりである。

1. 0 以上 257 未満の数で$\mathrm{k}-1$個の定数(a_1, a_2)をランダムに選択する。(今回$\mathrm{k}=3$なので 2 個の定数を設定する。)
2. int 型の配列 share に次の値を計算し、保存する。

$$\mathrm{share[i]} = s + a_1 * i + a_2 * i^2 \ (mod\ p)$$

また、share[0]は秘密がそのまま保存されるため例外として 0 とする。

復元方法は share[0]以外の任意の 3 個のシェアから求める。

1. ラグランジュ補間を利用するためシェアの番号に対応する値を用意する。
この時、i 番目のシェアに対応する値は以下の式により求まる。

$$\mathrm{F_i} = \frac{(0-x_1)(0-x_2)\dots(0-x_{i-1})(0-x_{i+1})\dots(0-x_k)}{(x_i-x_1)(x_i-x_2)\dots(x_i-x_{i-1})(x_i-x_{i+1})\dots(x_i-x_k)}$$

2. 答えとして値を用意し(ans)そこにシェアと 1. で用意したシェア番号に対応する値を乗算した結果を加算する。
3. 3 回(k 回)加算したのち、ans を mod p する。これにより、秘密が復元される。

◆ 4. java での実装

シェアの作成方法から実装していく。

1. については乱数を利用することにより、容易に書くことが可能である。
2. については再帰を利用することにより、汎用性を高めることができるが、今回実装するプログラムはあらかじめ k と n の値を固定しているので重くなる再帰を利用せず直接式を入力した。

次は復元方法である。

1. はまず、分母と分子に分けて計算していく。その際、掛ける数が負にならないか確認し、負になるようであれば p を加算し、p の mod の値となるようにする。これは計算で mod を用いているためである。最後に分子の mod p と分母の乗法逆元を掛ける。この値がシェア番号に対応する値である。
2. については直接式を書き込むだけである。注意する点はシェアに乗算する値はシェアに対応した番号の値であるということである。
3. は計算した値を mod p するだけである。

◆ 5. 実行結果

```
<終了> Project [Java
share[1] = 33
share[2] = 218
share[3] = 63
share[4] = 82
share[5] = 18
s =     22
ans =   22
```

図 1　実行結果

◆ 6. 考察

図 1 は 1, 2, 3 番のシェアを用いて計算結果を出力した。s が秘密であり、ans が計算した結果である。実装したプログラムでは連続した 3 つのシェアしか選択できないが書き換えにより、任意の 3 つのシェアから求められるように変更は可能であると考える。