秘密分散法の実装における速度・容量性能の評価

三浦夢生

1. まえがき

様々なシステムやサービスがディジタル・オンライン化される流れにある現代社会において，情報の紛失や盗難への対策は重要性を増すばかりである．例として，機密情報の複数人による管理や各地に配置されたストレージで情報の保管をする場合などがある．これらに対する技術として「秘密分散」があげられる．秘密分散とは元の秘密情報をアルゴリズムに基づいて作成した分散情報を管理する者や端末に配布・保管し，必要な数だけ集めて復元する技術のことである．この技術はクラウドサービスやブロックチェーンと相性が良いため広く用いられている．

この技術の代表例としてShamirの提案した(k, n)しきい値秘密分散法 [1]があげられる．これは元の秘密情報からn個のシェアを生成し，k個のシェアから秘密情報の復元ができるが，$k-1$以下の個数のシェアからは元の情報は全く得られない手法である．本研究では，加法的秘密分散法 [2]・(k, n)しきい値秘密分散法 [1]・(k, L, n)しきい値秘密分散法 [3] [4]を実装し，実行速度及び生成されたシェアの容量について評価する．

2. 秘密分散法

2.1 加法的秘密分散法 [2]

加法的秘密分散法はn個のシェアに対してn個のシェアからのみ元の情報が復元できるため，(n, n)しきい値秘密分散法ともいえる．この手法はまず乱数を生成し，シェア$s_2 = r_1, s_3 = r_2, \cdots, s_n = r_{n-1}$とする．次に秘密情報$S$を用いて以下のようにシェア$s_1$を生成する．

$$s_1 = S - (s_2 + s_3 + \cdots + s_n) \tag{1}$$

復元の際には，参加者に配布したシェアを全て集めて足し合わせることで秘密情報Sを得る．

2.2 (k, n)しきい値秘密分散法 [1]

Shamirによる(k, n)しきい値秘密分散法はn個のシェアに対してk個のシェアから元の秘密情報が復元できるが，$(k-1)$個以下のシェアからは元の秘密情報に関する情報は全く得られない．この手法は秘密情報Sを定数項とする，ランダムな$k-1$次多項式を生成する．

$$f(x) = S + a_1 x + a_2 x^2 + \cdots + a_{k-1} x^{k-1} \tag{2}$$

また参加者に番号iを割り振り，$f(i)$を計算し，シェアとして参加者に渡す．復元にはk個のシェア$(i, f(i)), i = 1, 2, \cdots, k$を持ち寄り，$k$個の式を立てて連立方程式を解くことで秘密情報$S$が求められる．このときラグランジュ補間を用いて$x = 0$の場合を計算するとよい．

$$L(x) = \sum_{i=0}^{k} y_i l_i(x) \tag{3}$$

$$l_i(x) = \prod_{j=0, j \neq i}^{k} \frac{x - x_j}{x_i - x_j} \tag{4}$$

$k-1$個以下のシェアからはk個の式は立たず，解が求まらないため元の秘密情報に関する情報が得られないことは容易にわかる．コンピュータ上で連続値は扱えず，またラグランジュ補間において除算を用いているため以上の計算を有限体上で行うと都合がよい．

2.3 (k, L, n)しきい値秘密分散法 [3] [4]

Shamirの(k, n)しきい値秘密分散法を拡張した(k, L, n)しきい値秘密分散法はn個のシェアに対してk個のシェアから元の秘密情報が得られる．$k-L$個以下のシェアからは元の情報に関する情報は得られず(k, n)しきい値秘密分散法と比べて各シェアのサイズが$1/L$になる利点をもつが，$k-l (0 \leq l \leq L-1)$個のシェアからは断片的に元の秘密情報に関する情報が得られてしまう．しかし，各シェアのサイズが$1/L$になる性質は非常に強力であり，実用性に長けているといえる．この手法は秘密情報SをL個に分割し，多項式の係数として用いる．

$$S = s_1 || s_2 || \cdots || s_L \tag{5}$$

またLから$k-1$次の項には乱数を係数として用いて以下の多項式を生成する．

$$\begin{aligned} f(x) = &\, s_1 + s_2 x + \cdots + s_L x^{L-1} \\ &+ a_0 x^L + a_1 x^{L+2} + \cdots + a_{k-L-1} x^{k-1} \end{aligned} \tag{6}$$

次に，参加者に番号iを割り振り，$f(i)$を計算し，シェアとして参加者に渡す．復元も(k, n)しきい値秘密分散法と同様に連立方程式を解くことで秘密情報の断片を得た後，分割した際と逆の手順で結合を行い，元の秘密情報を得る．この手法における計算も有限体上で構成することでコンピュータによる実行が可能となる．

3. 実装

今回はPython3.7.9及びzsh5.8においてソースコードの作成を行った．作成したプログラムの流れを図1に示す．秘密情報として「This is the Secret!」という文章を1万行並べた200KBのファイルを用意し，秘密分散に用いた．

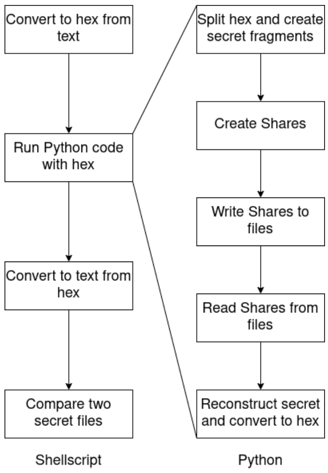

図 1 実装したプログラムの流れ．

3.1 シェルスクリプト部

シェルスクリプトにおいてはテキストファイルを16進数の文字列に変換し，Pythonのプログラムに渡す．その後，Pythonのプログラムによって生成された16進数のファイルをテキストに変換し，元のテキストファイルと再構築された秘密ファイルの差分を比較する．

3.2 Python部

ファイル入出力や秘密情報の分割については秘密分散と切り離すため別のプログラムを作成し，それぞれのアルゴリズムにインポートして使用している．各秘密分散には16進数を4文字ずつ読み込み，一つの10進数の秘密情報として渡している．

加法的秘密分散法については生成するシェア数を$n = 11$とし，用いる乱数の範囲を$-(2^{16}-1) \leq r \leq 2^{16}-1$とした．$(k,n)$しきい値秘密分散法ではパラメータを$k = 4, n = 11$に設定，有限体の標数は$p = 65537$とし，復元にはラグランジュ補間を用いた．$(k,L,n)$しきい値秘密分散法ではパラメータを$k = 4, L = 2, n = 11$に設定，有限体の標数は$p = 65537$とし，復元にはヴァンデルモンド行列による逆行列計算を用いた．

4. 結果・考察

Pythonのみの実行時間を計測した結果と平均シェアサイズを表1に示す．加法的秘密分散法はシェアを$-(2^{16}-1) \leq r \leq 2^{16}-1$の範囲で生成しているため，演算を有限体上で行う他の二つと比べて平均シェアサイズが大きくなっている．(k,L,n)しきい値秘密分散法は，今回秘密情報の断片を取り出すのに逆行列計算を用いているため非常に時間がかかった結果となった．

またシェアのサイズが(k,n)しきい値秘密分散法とほぼ同等であるが，これは元の秘密ファイルを一度断片に分割し，それぞれで秘密分散にかけたからだと考える．一つの秘密情報の断片は64ビットで分割され，どのアルゴリズムにも同様に配布されるため，今回の比較手法およびパラメータでは効果が見えなかったと考えられる．

表 1 実行時間及びシェアの平均サイズ．

アルゴリズム	実行時間[s]	平均サイズ[KB]
加法的 秘密分散法	3.154	約637
(k,n)しきい値 秘密分散法	6.956	約583
(k,L,n)しきい値 秘密分散法	2145.62	約583

5. まとめ

本研究では，加法的秘密分散法・(k,n)しきい値秘密分散法・(k,L,n)しきい値秘密分散法を実装し，実行速度及び生成されたシェアの容量について評価した．本研究の実装手法・評価項目においては(k,n)しきい値秘密分散法が最も優れているという結果となった．

今後の展望として，まず(k,L,n)しきい値秘密分散法に適した秘密情報の断片作成方法及び秘密情報を取り出すアルゴリズムの模索をし，より適切な評価を行うことが大きな課題の一つといえる．与えられた秘密情報を再度分割して秘密分散を行うのではなく，秘密情報をL個ずつ用いて秘密分散することでシェアのサイズを削減できると考える．

次に，比較対象・評価項目を増やし，様々な観点から考察することも今後の課題といえる．比較対象として，AONT秘密分散 [5]やKrawczykの方式 [6]などを実装したり，評価項目として，シェアから得られる秘密情報の情報量や，秘密の分散・再構築時にかかった時間の項目，各秘密分散について乱数の範囲やパラメータを変更した際の項目を設けたりすることで，より多角的に評価することができると考える．

参考文献

[1] A Shamir, “How to Share a Secret”, Communications of the ACM, Vol.22 pp612–613, 1979.
[2] 大原一真, “秘密分散を用いた秘密計算”, システム/制御/情報, Vol.63, pp71–76, 2019.
[3] 山本博資, “(k,L,n)しきい値秘密分散システム”, 電子通信学会論文誌, Vol.J68-A, pp945–952, 1985.
[4] 千田浩司・五十嵐大・菊池亮・濱田浩気, “計算量的秘密分散およびランプ型秘密分散のマルチパーティ計算拡張”, 研究報告コンピュータセキュリティ, Vol.2012-CSEC-58, pp1–5, 2012.
[5] Ronald L. Rivest, “All-or-Nothing Encryptionand the Package Transform”, Fast Software Encryption, Vol.LNCS 1267, pp210–218 1997.
[6] Hugo Krawczyk, “Secret Sharing Made Short”, CRYPTO’93, Vol.LNCS 773, 136–146, 1994.