秘密分散コマンドの実装及び速度・容量性能の評価

三浦　夢生
（木更津工業高等専門学校 専攻科 制御・情報システム工学専攻）

Evaluation of speed and capacity perform in inplementing secret sharing command

Yu Miura
(Advanced Control and Information Engineering Course, National Institute of Technology, Kisarazu College)

In modern society, spreads cloud or online service and a lot of important information (e.g. personal data) is transmitted and received. Although, there are risks what online management of their information like that loss or theft when the security was vulnerable. In this study, I propose the effective method "Secret Sharing Scheme" against that risks, implement command by using C language on Linux system. In making the program, I use "Shamir's Secret Sharing Scheme" as the main algorithm. This is known as computationally secure. And each operation construct with Galois field what works well with bit operation. After that, I evaluate the implemented program. In an evaluation in the aspect of speed, measuring elapsed time in executing, counting the number of calling system call by using commands. In an evaluation in the aspect of capacity, considering computational complexity and generated file with algorithm and implementation.

Keywords: Secret Sharing, cryptography, Shamir's Secret Sharing Scheme, Galois field, Linux command

1　まえがき

近年，クラウド及びオンラインサービスが企業や一般家庭において広く普及し [1],[2]，個人情報などの多くの重要な情報がやり取りされている．特に，重要な情報をオンライン上に保管する場合はログイン時にパスワードや ID を要求したり，ファイルを閲覧するためにパスワードを要求したり，あるいはファイル自体をパスワードによって暗号化したりする場合が考えられる．実際，重要な情報へのアクセスに対してセキュリティ対策を施す企業は増加している [3]．しかし，このようなケースにおいて，利用されている OS やサーバなどのバージョンが古い場合や，弱いパスワードが設定されている場合など，セキュリティが脆弱である場合に情報が漏洩する可能性が高まる．

本研究では，前述した暗号化では補えないリスクに対して有効な手段である秘密分散に基づくコマンドを実装する．秘密分散アルゴリズムには，計算量的安全性をもつ Shamir の秘密分散法を用い，bit 演算と相性の良い拡大体上でテキストファイルに秘密分散処理を行う実装を試みる．Shamir の秘密分散では多項式補間等を用いるため，一般に連続値を扱うが，拡大体を用いることで全て整数値で演算が可能となり，除算時の誤差を考慮しなくて良い利点も存在する．また，実装したコマンドを実行時間，発行されたシステムコールの回数による速度性能や，生成されたファイルサイズ及び計算量面での容量性能について評価し考察を行う．

先行事例として，エスロジカル社の “マル秘分散” という秘密分散ソフト [4] や，Jon Frisby 氏の “ssss” という秘密分散コマンド [5] が存在する．前者は二つないし三つの分散情報を生成し，任意のファイルについて秘密分散が可能である．後者はコマンドライン上で 128 文字までの標準入力に対して秘密分散が可能である．本研究では先行事例を参考に，テキストファイルに対応したより多くの分散情報を生成できるプログラムの実装を行う．秘密分散コマンドの実装をし，一般に公開することで，秘密分散技術のより身近な普及を狙い，社会全体のデータ管理におけるセキュリティ向上を目指す．

2　前提知識

2.1 Shamir の秘密分散法

秘密分散法とは，まず対象となる秘密情報をあるアルゴリズムを用いてシェアと呼ばれる分散情報を作成し，そのシェアを管理する人・端末に配布し管理・保管する手法である．元の秘密情報を得る際には，管理されたシェアを必要な数だけ集め，復元アルゴリズムによって復元する．

いくつかある秘密分散法のうち，本研究では Shamir の秘密分散法 [6] を用いている．この手法は (k,n) しきい値秘密分散法とも呼ばれ，シェアを n 個生成し，k 個のシェアを集めることで秘密情報の復元が可能であるが，$k-1$ 個以下のシェアからは復元は不可能である．

シェアを生成するアルゴリズムは，まず Eq. 1 のような秘密情報 S を定数項とするランダムな $k-1$ 次多項式を定める．

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

管理に参加する人や端末の数 i に対し，$f(1), f(2), \cdots, f(i)$ を計算し，配布する．

復元の際は，k 個のシェア $(j, f(j))$ を集めて k 個の式を立て，連立方程式を解くことで秘密情報 S を得る．k 個のシェアは同じ多項式から生成されたものであればよく，任意のものを用いればよい．

またこのときラグランジュ補間を用い，式 (2), (3) において $x=0$ の場合を計算することで．多項式のうち定数項のみ，つまり元の秘密情報を得ることができる．

$$L(x) = \sum_{i=0}^{k} y_i l_i(x) \tag{2}$$

$$l_i(x) = \prod_{j=0, j\neq i}^{k} \frac{x - x_j}{x_i - x_j} \tag{3}$$

2.2 拡大体

体とは集合に対して加法，乗法の二つの二項演算を定めた代数的構造のことであり，加法，乗法のどちらに関しても結合法則・交換法則・分配法則が成り立ち，単位元及び逆元が存在する．また前提として，集合の二つの要素に対して，その二項演算の結果が集合の中に存在することも条件である．これらのことから，体の要素は加減乗除に閉じているといえる [7]．これを踏まえて拡大体 K' とは，ある体 K に代数的構造を損なわないように元 $\alpha \notin K$ を追加して得られる体のことである．このとき，K' は K を含んでおり，K は K' の部分体ともいう．また，K' では K で定義された演算をそのまま用いることができる [8]．

本研究では $GF(2) = \{0, 1\}$ という二つの要素をもつ体を拡大し $GF(2^8)$ という集合を扱う．まず $GF(2)$ における演算を Table 1 及び Table 2 のように定義する．この体は整数を 2 で割った剰余によって構成された集合と見ることもできる．

この演算を踏まえて $GF(2)$ を拡大し $GF(2^8)$ を構成する．このとき，既約多項式という概念を導入すると，$GF(2)$ において 2 の剰余の集合と扱えたように，$GF(2^8)$ は既約多項式による剰余の集合とすることができる．本研究では既約多項式 $x^8 + x^4 + x^3 + x^2 + 1$ を用いたため，$GF(2^8)$ の各元は係数が 0 または 1 の 7 次式である．

Table1. 加法の演算表　Table2. 乗法の演算表

+	0	1
0	0	1
1	1	0

×	0	1
0	0	1
1	1	0

$GF(2^8)$ において，加法は表 1 より各次数の項の排他的論理和となるため，7 次までの項の係数を 8bit の列として見ることで，コンピュータで扱いやすい演算となる．減法について，$1 \equiv -1 \bmod 2$ であるため，加法と同じ演算を用いることができる．

乗法は各元の算術的な積をとり，既約多項式による剰余を求めることで定義される．除法についても 0 除算を除いて体の定義から逆元が存在するため剰余の計算をすることで求めることができる．

ここで仮想的に既約多項式の根 α を考える．この根は体上には存在しないが，この根は体の原始元であり，体の要素は原始元のべき乗によって巡回的に生成される．このことから，多項式による表記と原始元のべき乗による表記を対応させることで多項式の積を実装せずに体上の乗除が可能となる [9]．

3 実装

実装・計測した環境は表 3 の通りである．

Table3. 実装・計測環境

OS	linux 5.14.14-arch1-1(x86_64)
CPU	Intel Core i7-8565U 1.80GHz
メモリ	16GB
コンパイラ	gcc 11.1.0
シェル	GNU bash 5.1.8(1)-release

コマンドはオプション解析部，ファイル入出力を含む分散・復元処理部，体上の演算部に分けて実装している．今回はテキストファイルのみに対応した秘密分散を実装した．

オプション解析部では分散・復元モードの選択，パラメータの入力及びファイル名を受け取り，解析を行っている．モード選択には文字列"split"または"combine"が利用可能であり，パラメータには生成シェア数 $n(1 \leq n \leq 255)$ 及びしきい値 $k(1 \leq k \leq n)$ を数値とともに指定する．分散モードでは秘密分散にかけるファイルを一つ指定し，復元モードでは復元するために必要な数だけシェアファイルを指定する．オプションの文字列・パラメータの数値が適切に入力されている場合に分散・復元処理

が呼び出される．

分散処理部ではパラメータの生成シェア数 n で指定された数だけファイルを生成し，入力されたテキストファイルを 1byte ずつ読み込みシェア生成のため体上の演算部に渡す．1 文字を 8bit とみなし，$GF(2^8)$ 上の要素として扱うことで前述の演算が行える．生成されたシェアは 2 文字の 16 進数としてファイルに出力される．

復元処理部では入力されたシェアのテキストファイルを 2 文字ずつ読み込み，分散時と同様 8bit の列とみなし，ラグランジュ補間によって方程式の解を求めることで秘密情報を得る．

演算部では $GF(2^8)$ の要素のベクトル表記を生成する関数や体上における加減乗除，またそれらを用いたラグランジュ補間を行う．

$GF(2^8)$ の要素のベクトル表記はプログラム実行時に一度だけ呼び出される，既約多項式による割り算回路の関数によって全て生成される．加減は二項の排他的論理和を行う．乗除は各元のベクトル表記に対応したべき表現に変換し，指数の，体の大きさによる剰余の計算を行った後にベクトル表記へと戻すことで実現している．

4　評価方法

各モードの選択，時間またはシステムコール計測の選択，秘密ファイルのサイズ選択をし，それらに基づいてコマンドの実行・計測をするスクリプト及び分散・復元モードにおいて，同一の秘密ファイルについてパラメータ n, k をそれぞれ変化させ，その時のそれぞれの実行時間を計測するスクリプトを用いて速度性能について評価する．

また，生成されたシェアのサイズ及び実装や使用したアルゴリズムにおける計算量について理論的考察を行い，容量性能について評価する．

5　結果・考察

生成シェア数 255，復元可能数 255 にパラメータを設定し，49KB のテキストファイルを秘密分散した結果を Table 4 及び Table 5 に示す．

Table4. 実行結果

	分散時	復元時
実行時間	73m50.112s	73m59.957s
生成されたシェアのサイズ	98KB	-

分散した秘密情報 1 文字に対して 16 進数 2 文字を出力しており，生成されたシェアのサイズが 2 倍になっていることからプログラムが正当に動作していることがわかる．また分散時の write 関数や

Table5. 主なシステムコールと発行回数

分散時	復元時
write 6120 回	read 6122 回
openat 258 回	newfstat 258 回
close 258 回	openat 258 回
newfstatat 258 回	write 12 回
read 14 回	brk 11 回
brk 13 回	mmap 8 回

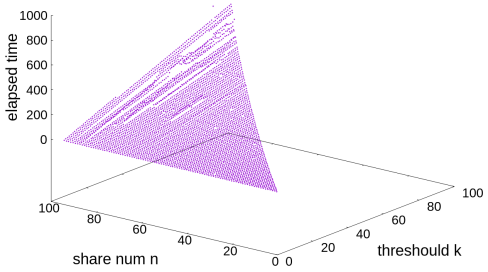

Fig.1. パラメータの変化に伴う分散処理実行時間の変化

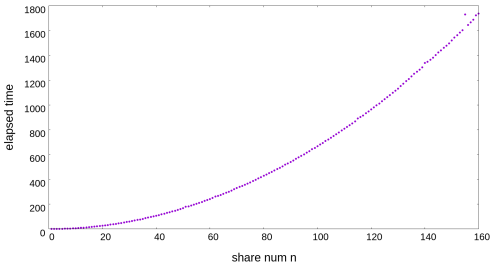

Fig.2. パラメータの変化に伴う復元処理実行時間の変化

復元時の read 関数が増えていることからも同様なことが言える．

分散処理はランダムな多項式を生成した後にシェアを生成，16 進数の 2 文字として各シェアに出力する．これをファイルの文字数分だけ行う．つまり少なく見積もってもこの場合 4.9 万×255 回のループ処理を行っているため，多くの実行時間がかかったと考える．

復元処理においては，ファイルへの出力はないが，ラグランジュ補間が $O(n^2)$ の計算であるためおおよそ 255^2×ファイルサイズ 程度のループ処理を行っていることからこのような実行時間になったと考える．

計測は実行終了までに月単位の時間がかかるため，現時点での実行結果を示す．分散処理においてパラメータを変化させた際の実行時間の変化を Fig. 1 に示す．グラフより，生成するシェア数や，しきい値が変化すると実行時間は線形的に増加することがわかる．

加えて，復元処理においてパラメータを変化させた際の実行時間の変化を Fig. 2 に示す．復元処理

においては $O(n^2)$ の計算を行っていることから, 実行時間も指数関数的に増加していることがわかる.

今回のプログラムではファイルのサイズもとい文字数に大きく依存しているが, 一般的なファイルはより大きなファイルサイズであることが多いため, もしテキストファイル以外に対応したとしても現状では実用における課題が多いことがわかった.

6 まとめ

本研究では, 暗号化等のセキュリティ対策とは異なる, 情報を紛失・窃取から守る技術として秘密分散を挙げ, Shamir の秘密分散を拡大体上で行う実装をし, 実行時間やシステムコールの発行回数を計測した. その後. 実行時間や計算量などの面から評価・考察を行い, 本実装における課題を明らかにした.

大規模な秘密分散の際に用いられるような大きなパラメータでは, 分散・復元ともに実用的でない結果となった. また, 分散処理よりも復元処理のほうが, パラメータが増えるにつれて実行時間がかかることが明らかとなった.

7 今後の展望

短中期的な目標としてまず実行時間の高速化があげられる. 具体的に, 計算結果をバッファリングしてファイルへの出力回数を減らす方法や, 体として $GF(2^{16})$ を用い, 2byte ずつの処理を行う方法などが考えられる.

前者について, 今回の実装では各文字に対応したシェアを生成するたびにファイル出力を行っているため, 2 文字, 4 文字ごとに出力を行う実装とすることでファイルへの出力回数も二分の一, 四分の一となるため, 実行時間の短縮が見込める.

後者では 16 次既約多項式を用いる必要があり, 生成可能なシェアの個数も $2^{16}-2$ 個に増えるため, それだけシェアの重複がなくなることからよりセキュアな秘密分散となることが予想される. しかし, 分散処理実行時の, 体の要素を生成する関数における実行時間が大幅に増加すると考えられる. その分ファイル入出力も増えてしまうため 16 次既約多項式を求めるプログラムや, それによる体の要素生成を行うプログラムをコマンド利用の前に行うような実装とすることでコマンド自体の実行時間が短縮できると考える.

また生成されるシェアのサイズ削減があげられる. 今回は, 秘密情報 1 文字に対して 16 進数 2 文字を出力する実装だが, なんらかの 1 文字を出力する形に変えることで単純にシェアのサイズが現状の半分, 元の秘密情報と同じサイズとなる.

次に, 他の秘密分散アルゴリズムの模索を行い, 実行時間の短縮や, シェアのサイズ削減などを目指すことがあげられる. たとえば, (k, L, n) ランプ型秘密分散法 [10] という手法である. この手法は Shamir の秘密分散とは異なり, 新たに L というパラメータが利用でき, n 個のシェアを生成し, しきい値 k の数のシェアから復元が可能である. また, $k-L$ 個以下のシェアからは秘密情報に関する情報は一切得られず, $k-l(1 \leq l \leq L-1)$ 個のシェアからは秘密情報のどの部分も明確には得られない特徴をもつ. シェアのサイズは $1/L$ となる性質をもつため, シェアの容量が比較的小さくなることから, より実用的であるといえる.

また, 関数呼び出しの観点から, 今回はシステムコールのみのカウントをしたが, 標準関数や, 自作関数の呼び出し回数などのデータをとることで高速化への手がかりが得られると考える.

しかし, 秘密分散コマンド単体では得られないセキュリティ対策もある. 例えば, 通信路が暗号化されていない場合などは, しきい値分だけの通信路を盗聴することで復元が可能となってしまう [11]. このことから, 長期的な目標として秘密分散をより効果的に活用できるようなシステムの考察・実装を行うことが, 秘密分散コマンドの実用へ繋がると考える.

参考文献

(1) 総務省, “デジタルで支える暮らしと経済”, 情報通信白書, pp.156-159, July 2020.

(2) 総務省, “デジタルで支える暮らしと経済”, 情報通信白書, pp.313-315, July 2020.

(3) 独立行政法人情報処理推進機構, “企業における営業秘密管理に関する実態調査2020”, 調査実施報告書, pp40-47, March 2020.

(4) 株式会社エスロジカル, “マル秘分散”, http://www.ma-bu.com/, 閲覧日Dec 2021.

(5) Jon Frisby, “ssss”, https://github.com/MrJoy/ssss, 閲覧日Dec 2021.

(6) A Shamir, “How to share a secret”, Communications of the ACM, Vol.22, pp.612-613, April 1979.

(7) 野崎昭弘, “なっとくする群・環・体”, 講談社, 第1版, pp.136-137, May 2016.

(8) 汐崎陽, “情報・符号理論の基礎”, オーム社, 第2版, pp.90-91, May 2019.

(9) 野崎昭弘, “なっとくする群・環・体”, 講談社, 第1版, pp.160-176, May 2016.

(10) 高荒亮, 岩村惠市, “XORを用いた高速な(k, L, n)ランプ型秘密分散法に関する研究”, コンピュータセキュリティシンポジウム2009 (CSS2009)論文集, Vol.2009, pp.1-6, October 2011.

(11) 青野成俊, 岩村惠市, “実用観点からみた秘密分散法に関する一考察”, コンピュータセキュリティシンポジウム2009 (CSS2009)論文集, Vol.2009, pp.1-6, October 2011.