GPG 秘密鍵取り扱い方法の提案

吉田 晋

2014 年 6 月 14 日

自己紹介

名前 吉田 晋 (wbcchsyn)
ハンドルネームに特に由来は無い
昔生成したランダム文字列
（日本語では、「うぶちん」と名乗ったりもします。）
どんなサービスであれ、
このアカウントの人がいたら、多分私。

仕事 IT 系の会社でサーバーサイドの事を広く浅く
UI は苦手

言語 いろんな言語を趣味として学ぶのが好き
今年はアセンブリを覚えようと計画中

GPG 4096R/DD08DFA8

自己紹介

セキュリティに関する座右の銘

- 独自実装はするな、必ず穴がある
- 専門家の意見に従え

セキュリティの専門家ではありません

はじめに

みなさん、GPG のキーサインしていますか？

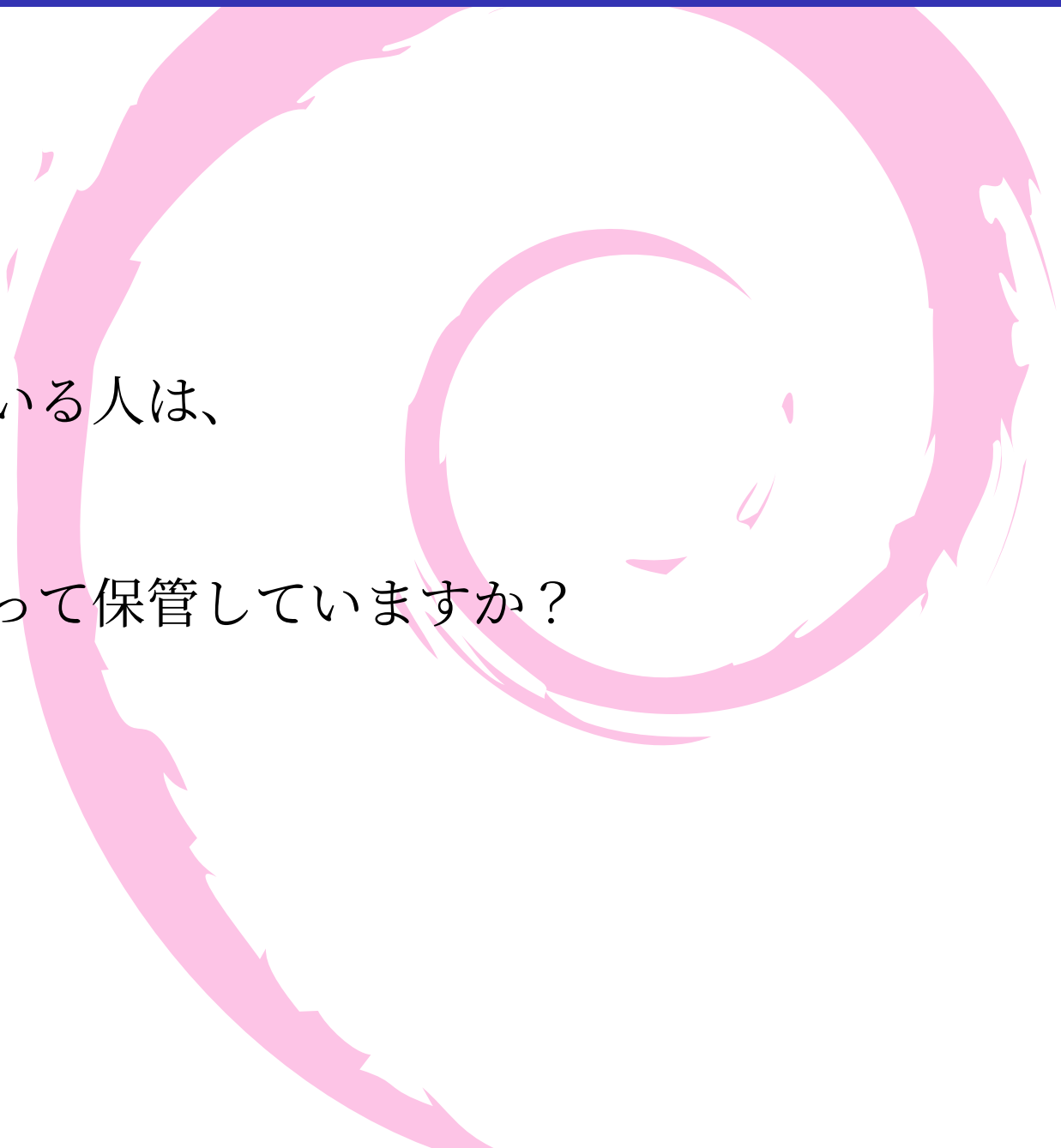
はじめに

私は、

「大統一 Debian 勉強会 2013」

で初めてキーサインパーティーに参加しました

はじめに



キーサインを行っている人は、

その秘密鍵はどうやって保管していますか？

はじめに

再発行出来るとはいえ、秘密鍵が無くなると色々と不便

- その秘密鍵で暗号化したファイルを
  復号化出来なくなる
- 再度、他の人に署名してもらう必要あり
  （貰ったキーサインは資産）

記録メディアの破損等に備え、秘密鍵のバックアップは必要

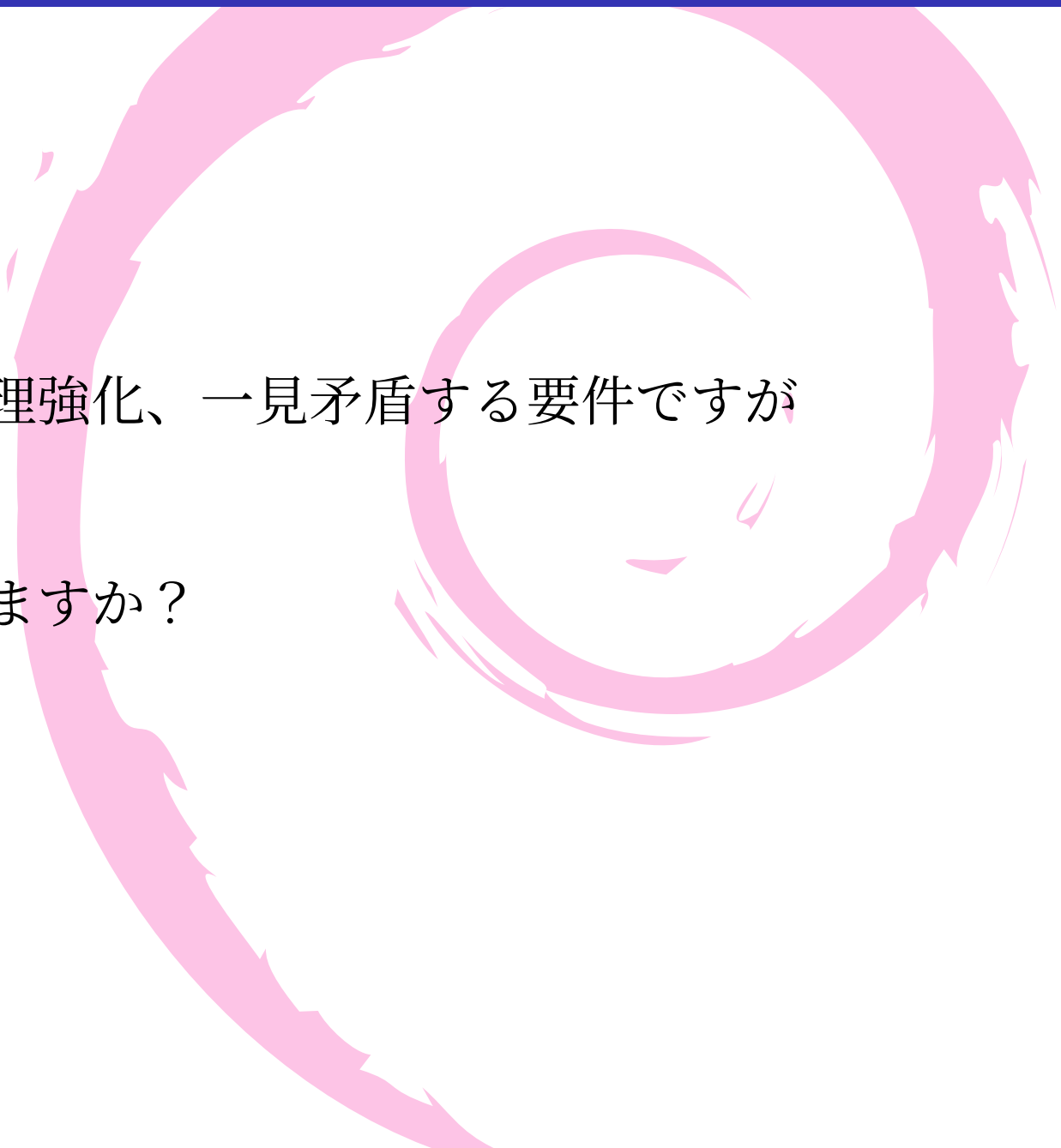
はじめに

一方、いつでも Revoke できるとはいえ、

最低限のマナーとして盗難や盗聴に気をつける必要も有り

管理強化の為には、鍵のバックアップは少ない方が有利

はじめに



データの冗長性と管理強化、一見矛盾する要件ですが

みなさん、どうしてますか？

はじめに

はじめに

~~*Fin*~~

ウソ

はじめに

私は、今の所

「データは 3 カ所以上で保存しない」

という自分ルールで妥協中

はじめに

今回は、この運用方法を変えたいと思って
鍵管理ツールのプロトタイプを作ってみました。

ただ、自分はセキュリティについて詳しく無いので
何かフィードバックがあれば、お願いします

「危険だからそんなツール使うな」
って言う意見でも O.K. です

アイディア

アイディアのベースは RAID3 (RAID5)

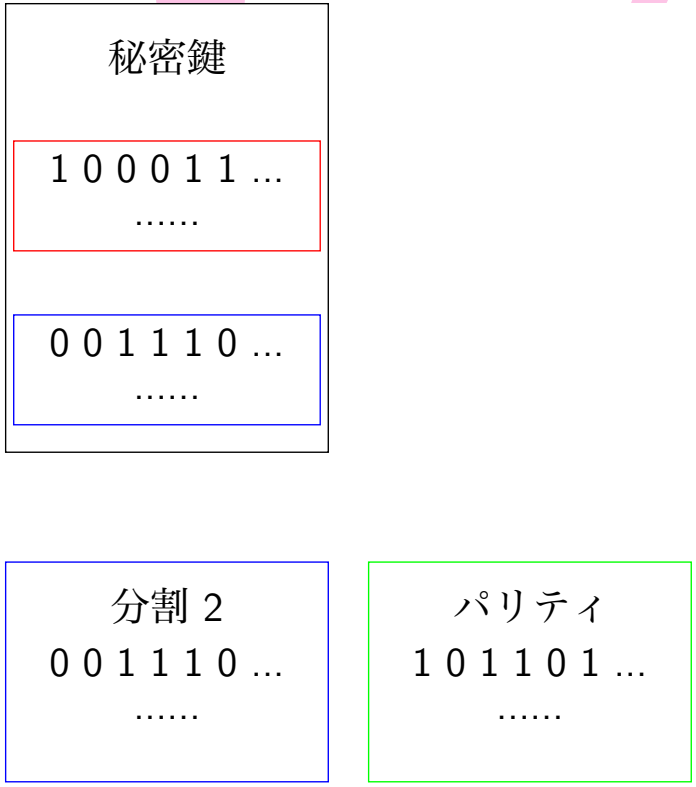
アイディア

RAID3 ディスクアレイの特徴

- ディスク 1 本破損してもデータ復元可能
- ディスク 1 本盗難にあっても盗難者はデータ復旧不可能

アイディア

同じ要領でファイルを分散、保存
メディアが 1 個破損してもデータ復旧可能
メディアが 1 個盗まれても、しばらく大丈夫



アイディア

例えば、3 個のファイルを
PC、USB メモリ、DVD@金庫
の 3 カ所に保存

普段は PC と USB メモリからデータを復旧させて使う
（毎回、データ復旧、使い終わったら削除）
なんて使い方を考え中

アイディア

USB メモリに鍵情報を保存するのは少し怖いが、
4096 bit 長の鍵ならば、ファイルが 1 個盗まれたとしても
まだ 2048 bit 残っている

現役で 2048 bit 長の鍵を使っている人が居る事を思えば
まだ大丈夫かなと。。。

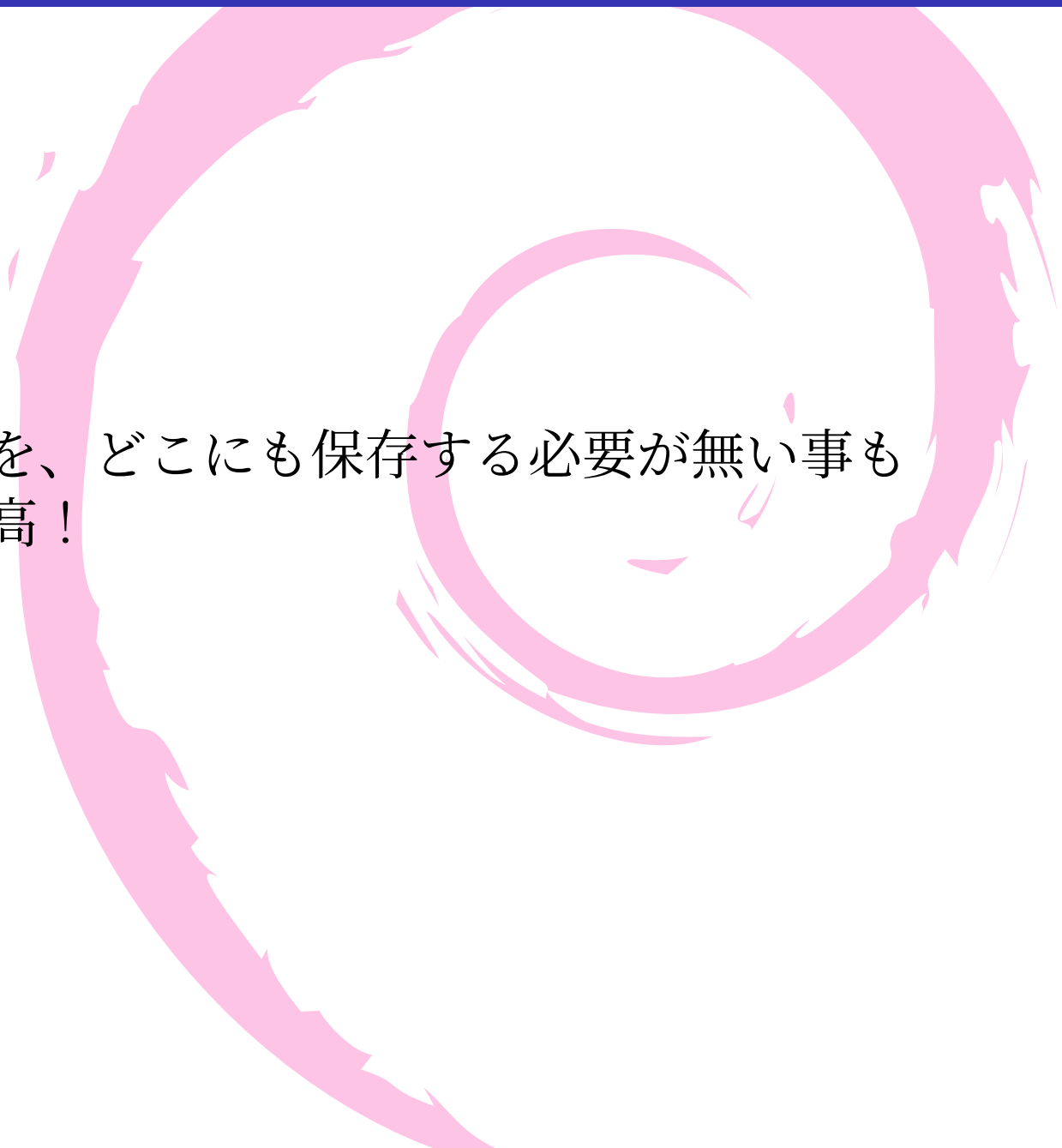
アイディア

ちなみに、分散するファイル数とパリティの数を
それぞれ 2 倍にすると （4 分割 + パリティ 2 個）

ファイル 1 個あたりの情報量 減
メディア破損時の冗長性 増

ちゃんと、盗難時のリスク管理と冗長性の向上を両立出来
てます。

アイディア



完全な形のファイルを、どこにも保存する必要が無い事も
個人的にはポイント高！

実装紹介

ファイルを分割、分割したファイルを結合する
プロトタイプを実装

`https://github.com/wbcchsyn/dVault_prototype.git`

dVault (distribute vault) と命名

実装紹介

注意

これはプロトタイプです。
絶対に、自分の秘密鍵で実験しないでください！

例えば、

- ファイルのアクセス権とか考えていません
- 問答無用で既存のファイルを上書きします
  失敗して、既存のファイルを壊すかも

実装紹介

分割したファイルの形式

1. ファイルの先頭に json 形式でメタデータを記載
2. メタデータの後に空行を 1 行挟む
3. その後、各ファイルにデータを保存

実装紹介

現状では、下記の 3 個のメタデータを定義
（このデータが無いと、復旧できない！）

- united_length
  元ファイルのファイルサイズ
- split_count
  ファイルを何分割したか
  (最低 2, パリティの数は含まれません)
- index
  分割したフィアルの何個目か
  0 スタート
  index >= split_count の場合、パリティである事を示す

使用方法

origin というファイルを split0, split1, parity の 3 個に分割

```
$ dvault split -u origin -s split0 -s split1 -s parity
```

split0, parity の 2 個のファイルから origin を復旧

```
$ dvault unite -u origin -s split0 -s parity
```

実演

実演

中身が abcedf\n の 7 byte のファイルを 3 個に分割してみる

実演

準備

```
$ ls -l

total 4
-rw-r--r-- 1 wbcchsyn wbcchsyn 7 Jun 14 07:08 origin


$ cat origin

abcdef
```

実演

分割
split0 には、ヘッダと origin の偶数バイト目が記載される

```
$ dvault split -u origin -s split0 -s split1 -s parity

$ cat split0

{"index": 0, "united_length": 7, "split_count": 2}

ace
```

実演

split0 と parity から origin を復活させる

```
$ rm origin split1

$ ls -l

total 8
-rw-r--r-- 1 wbcchsyn wbcchsyn 56 Jun 14 07:15 parity
-rw-r--r-- 1 wbcchsyn wbcchsyn 56 Jun 14 07:15 split0

$ dvault unite -u origin -s split0 -s parity

$ ls -l

total 12
-rw-r--r-- 1 wbcchsyn wbcchsyn  7 Jun 14 07:20 origin
-rw-r--r-- 1 wbcchsyn wbcchsyn 56 Jun 14 07:15 parity
-rw-r--r-- 1 wbcchsyn wbcchsyn 56 Jun 14 07:15 split0

$ cat origin

abcdef
```

その他、現行仕様

- パリティは 1 個で固定
- ファイル分割時は、元ファイルを 1 byte ずつ分割
- 分割時に元ファイルが split_count で割り切れない場合、0 でフィリング
  結合時には、フィリングの 0 は落とします
- 最終的なファイル書き出し時以外は
  全てオンメモリで処理
  一時ファイルは作りません

現状の悩み

Python で実装したのは、正しかったのか？

Debian の最小構成では、Python が入っていない
パッケージ化を目指すなら、C で実装した方が良かったかも

現状の悩み

メタデータは json で保存して良かったのか？

今後の拡張を考え、型を分かりやすくしたり
文字列のエスケープが分かりやすいようにと思い json にした
しかし、HTTP ヘッダのように改行区切りの方が
良かったかもしれないと
思い直している最中

- ANSII C で実装する場合、json のパースは面倒
- 現行のツールでは、json 中に改行 x 2 があるとバグをふむ

現状の悩み

メタデータにファイルの Hash 値や、ツールのバージョン等を
加えても良いか？

ツールの安定開発の為にはメタデータを増やしたいが、
1 個盗まれた際の強度が減る事が弱点

現状の悩み

オンメモりの処理にこだわる必要があったのか？

当初は、ディスクに何らかの情報が残る事が嫌だった
でも、良く考えると SWAP したら意味が無い
秘密鍵を生成、復旧する際に、ディスクに保存したら意味が無い

せっかくなら、秘密鍵以外のデータも管理出来た方が良い
大容量のファイルを扱う際にそなえて、名無しの一時ファイル（開いた直後に OS で削除）程度は
使ってよいかもしれない

現状の悩み

使い勝手、悪すぎないか？

毎回秘密鍵を生成して、使い終わったら削除するという運用は面倒すぎる
なんとか、もう少し自動化したい所
例えば、

- 削除忘れ対策として、ファイルは必ず tmpfs 上に生成する
- 生成したファイルを自動で使用するように、gpg コマンドのラッパーを作る

など

現状の悩み

複数パリティに対応するべきか？

対応するとしたら、どの程度の数のパリティまでサポートするか？

同じアルゴリズムで多数決のツールとして使う事も可能
（1 人が分割したファイルを 1 個ずつ持つ）
パリティの数に制限は付けない方が良い事は確か
でも、パリティの数を増やしたり、破損チェック機構を加えると
計算量が劇的に増加する

現状の悩み

ファイル分割の仕方を 1 byte ずつ固定ではなく、可変長にするべきか？

分割方法が 1 byte ずつ固定の場合、
分割されたファイルが 1 個盗まれると、急に強度が弱くなる
（2 分割　＋パリティの場合、もう 1 個盗まれたらアウト）

しかし、分割長の長さを返る事ができれば、少しはマシ

現状の悩み

自分以外の人は使ってくれるだろうか？

自分も使ってよいのか？
（安全か？）