

# PGP Public Keyserver のコンテンツ同期についての考察

---

鈴木裕信  
OpenPKSD プロジェクト  
平成 14 年 2 月 26 日

はじめに .....	3
PGP PUBLIC KEYSERVER における鍵数の考察.....	4
PGP.NIC.AD.JP のデータ .....	4
SURFNET のデータ .....	4
REDIRIS のデータ .....	5
ジョージア工科大学のデータ.....	5
大量の鍵の差についての考察.....	6
大量の鍵が異なる場合のアルゴリズム.....	7
アルゴリズム .....	7
処理量 .....	8
極めて少数の違いがある場合 .....	9
アルゴリズム .....	9
処理量 .....	10
関連資料 .....	12

## はじめに

本文は異なる PGP Public Keyserver 間でコンテンツ同期を行なうにはどのような方法が考えられるかをまとめたものである。

まず現状における PGP Public Keyserver における鍵数の考察を行ない、次に同期するためのアルゴリズムを2つ提案する。大きな2つの集合において、異なる要素を効率良くみつけるというのがポイントである。そこにネットワークの転送能力、CPU パワー、あるいはメモリ容量といった制約用件が加わる。また大量の要素が異なる場合、少量の要素が異なる場合といった条件でもアルゴリズムの選択が異なる。

これらの考察、およびアルゴリズムのアイデアのディスカッションは主にメーリングリスト上で行なわれた。本文は、そのまとめとしての位置付けとなる。

## PGP Public Keyserver における鍵数の考察

### PGP.NIC.AD.JP のデータ

アジア圏唯一の PGP Public Keyserver である `pgp.nic.ad.jp` の鍵サーバは 1 月 23 日時点での全鍵数が 1,595,374 鍵であった。公表されている統計情報によると[1]、1 年間(2001 年 1 月 7 日 ~ 2002 年 1 月 7 日まで)に追加された公開鍵は約 31 万鍵である。他サイトからのトラザクションは約 82 万回発生している。

URL <http://pgp.nic.ad.jp/docs/2001.txt>

各種統計情報	
総保持鍵数	1595374
サーチリクエスト受付数	53070
鍵 ID を使ったのサーチ回数	29522
サーバへのコネクション回数	134521
鍵の登録/更新回数	826321
鍵同期メール数	823163
公開鍵の追加	312373
公開鍵への署名の追加	61437
署名の入れ換え	7397
ユーザ ID の追加	39756
プライマリ UID の変更	16668
鍵のリボケーション	5608
鍵のリボケーションを変更	661

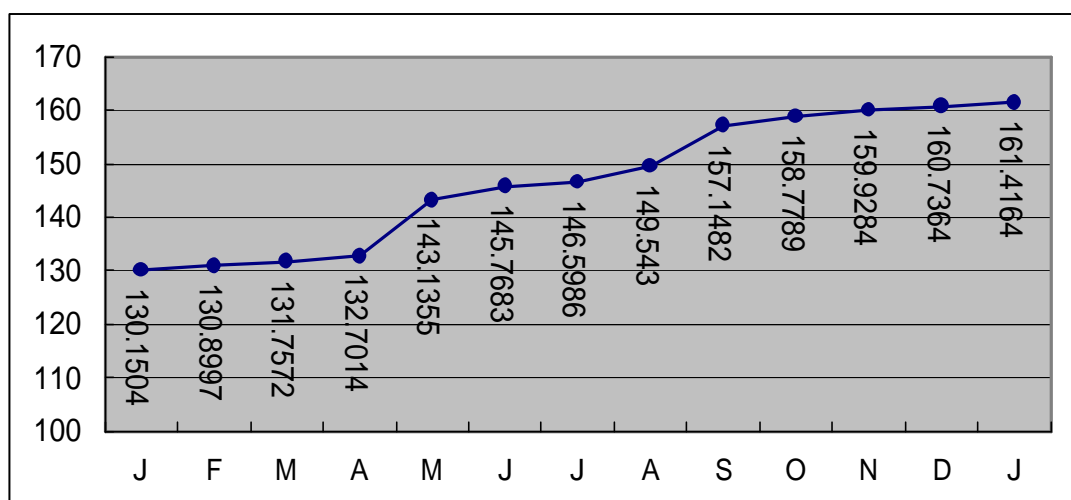
注) 解析ツールは Roman Pavlik <rp@tns.cz> が作成した

### Surfnet のデータ

ヨーロッパの中心的 PGP Public Keyserver であるオランダ Surfnet の鍵サーバは 1 月 2 日時点での全鍵数が 1,614,164 鍵であった[2]。過去 1 年間の増加は以下の通りであった。

URL <http://pki.surfnet.nl/>

単位は万



pki.surfnet.nl での1年間の鍵の増加

年間約 31 万鍵の増加というのは pgp.nic.ad.jp での計測データと合致する。pki.surfnet.nl と pgp.nic.ad.jp は相互に同期している。

## RedIRIS のデータ

スペインの CSIRT チーム IRIS-CERT が運営している鍵サーバ(pgp.rediris.es)は、1月25日時点での全鍵数は 1,605,783 鍵であった[3]。

URL <http://www.rediris.es/cert/servicios/keyserver/>

## ジョージア工科大学のデータ

アメリカのジョージア州にあるジョージア工科大学(cc.gatech.edu)で運営している鍵サーバは2月1日時点での全鍵数は 1,623,342 鍵であった[4]。

## 大量の鍵の差についての考察

今回の考察に用いたデータであるが、同じタイミングで各々のサーバ上の鍵数をカウントしたわけではない、それゆえ正しい鍵数の違いの比較とはならない。トランザクション数が多いため鍵数の比較をする時は明示的かつ厳密に日時を指定し、その時点での鍵のカウントを行わなければならない。

鍵数	サーバ名
1595374	pgp.nic.ad.jp (2002/1/23)
1605783	pgp.rediris.es (2002/1/25)
1614164	pki.surfnet.nl (2002/1/02)
1623342	cc.gatech.edu (2002/2/01)

このデータからわかることは pgp.nic.ad.jp は pki.surfnet.nl と同期しているにもかかわらず、最低で約 2 万近い鍵が正しく同期していない。数万というオーダーの大量な鍵に対するリクエストが必要な場合も考慮に入れなければならないということがわかる。

pgp.nic.ad.jp や pki.surfnet.nl のデータから見ると一カ月に 2 万~3 万鍵程度の増加が発生するので、1 日メンテナンスのためにサーバが停止するだけでも数千鍵の違いが発生する可能性がある。たとえば、なんらかの理由で数日間サーバが停止した場合などは、数万というオーダーの違いを処理する場合も考慮に入れなければならない。

## 大量の鍵が異なる場合のアルゴリズム

大量の鍵が異なる場合、あるいはまだ一度も同期をしていないためどれだけの鍵が違うかわからない場合に用いるアルゴリズムは、両者の鍵を参照した時に複数の鍵が見つかるかどうかという極めて単純なアルゴリズムを利用する [5]。

```
"But you can't look up all those license
  numbers in time", Drake objected
"We don't have to, Paul. We merely arrange a list
  and look for duplications."
-- PERRY MASON, in The Case of the Angry Mourner (1951)
```

## アルゴリズム

Step 1: 両方で DB レコードの更新時間順に OpenPGP の鍵 ID をソートする。  
マスター側からの値をアップデート側が取るような形になる

Step 2: マスター側鍵( $M_k$  と表記)の値をハッシュテーブルのインデックスとしたテーブルに対し値を 1 にセットする。

```
For all  $M_k$ 
do
  Table[H( $M_{k_i}$ )] = 1
done
```

Step 3: アップデート側鍵( $U_k$  と表記)の値をハッシュテーブルのインデックスとしたテーブルに対し値が既に 1 であれば 2 を加える。値が nil の場合、2 をセットする。

```
For all  $U_k$ 
do
  if Table[H( $U_{k_i}$ )] is not nil
    Table[H( $U_{k_i}$ )] = 3
  else
    Table[H( $U_{k_i}$ )] = 2
done
```

Step 4: ハッシュテーブルの内容を参照し値を見ることによってマスター側のみの鍵か、アップデート側のみ鍵か、両方にある鍵がわかる。

```
For all Table elements
do
  if a TableElement is 1 then it is in master only key
  if a TableElement is 2 then it is in update only key
  if a TableElement is 3 then it is in both
done
```

Step 5: マスター側から"master only key"の鍵の束(集合)を受け取る。

Step 5': アップデート側から"update only key"の鍵の束(集合)を受け取る。

Step 5'は、双方向に鍵を同期する場合

## 処理量

マスター側の鍵の集合 M、アップデート側の鍵の集合 U、ハッシュテーブルの要素集合 H、ハッシュテーブル参照コスト Hcost と表すと、以下のような関係になる。

$$Hcost * (size(M) + (3 * size(U)) + size(H))$$

関数 size(X)は集合のエレメント数を返す

また

$$size(H) = size(uniq(M + U))$$

関数 uniq(X+Y) 集合 X と集合 Y のユニークな要素を返す

M、U 各々が 100 万の鍵を持ち、U の要素に 10%の違いがあるとした場合、このコストは (100 万 + 300 万 + 110 万 =) 530 万回のハッシュテーブル参照コストになる。

事前のデータ移動のコストは M を U のある場所へ移動するコストである。参照に鍵 ID をつかうならば、1 鍵につき 8 バイト(64 ビット)、100 万鍵で約 8M バイト程度である。現在のインターネット上では 8M 程度の転送は負担にならない。

ruby を使いハッシュテーブル参照を 500 万回行なうプログラムを書いた所、Linux 2.4、Althron 1GHz、Mem 1GB 上では約 69 秒で終了した。この程度の時間であれば十分に実用的である。

## 極めて少数の違いがある場合

実用上単純なマッチングで十分であるが、僅かな違いのための高速なアルゴリズムを考えてみる。

## アルゴリズム

Step 1: 両方で DB レコードの更新時間順に OpenPGP の鍵 ID をソートする。マスター側からの値をアップデート側が取るような形になる。

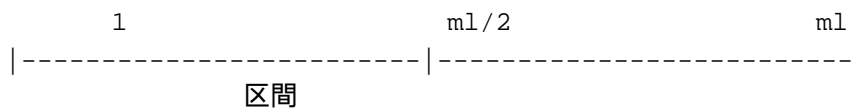
Step 2: マスター側鍵( $M_k$  と表記)、アップデート側鍵( $U_k$  と表記)とも要素  $M_{k_1}$ 、 $U_{k_1}$  が最も古い側とする。最新は(最後は) $M_{k_{m_1}}$ 、 $U_{k_{u_1}}$  と表記する。 $M_k$  と  $U_k$  の要素数が同じとは限らないので、 $m_1$  と  $u_1$  が同じ数とは限らない。

1 から  $m_1/2$  までの  $M_k$  と  $U_k$  のハッシュ値を計算し、比較する。

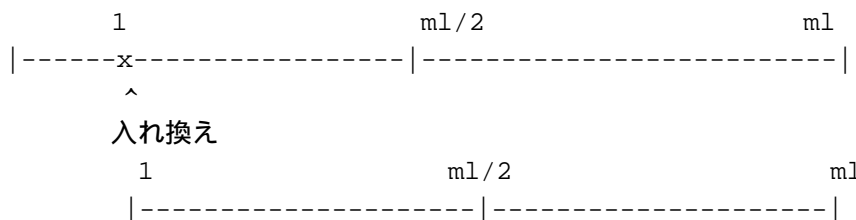
$H(x)$  はハッシュ関数

$$M_{mid} = H ( M_{k_1} + M_{k_2} + \dots + M_{k_{m_1/2}} )$$

$$U_{mid} = H ( U_{k_1} + U_{k_2} + \dots + U_{k_{m_1/2}} )$$



Step 3: もし区間の値が同じでない場合、 $m_1/2$  を 新しい  $m_1$  として Step 2 へもどる。最後の 1 つになったら、それは最も古い違いであるので、それを足りない方へ加える。 $m_1$  は動かさず、加えた場所を 1 として Step 2 へもどる。



Step 3': もし区間が同じ値であったら、 $m_1/2$  の場所を 1 とし、 $m_1$  は動かさず Step 2 へ戻る。ただし最初の位置と  $m_1$  とが 2 以下なら停止する。

Step 4: 最後の 2 個を比較する。

## 処理量

まったく同じ内容であれば、1000万のコンテンツで、24回 x 2回分のハッシュ値の参照で終了する。SHA-1などのハッシュ関数は高速であり、具体例としてLinux 2.4、Althron 1GHz、Mem 1GB上で100万鍵分の鍵IDである8Mバイト分データのハッシュ値を計算すると実測値で約0.12秒ほどである。この値を元に100万鍵分の処理にかかる時間の近似計算をすると $0.12 \text{ 秒} * 2 \text{ 倍} * 2 \text{ 回} = 0.48 \text{ 秒}$ という値になる(先に紹介した単純なアルゴリズムでは約69秒であった)。実際には前後の処理も入るので単純にはこの値にはならないが、極めて効率的であるといえる<sup>1</sup>。

---

<sup>1</sup> 本ドキュメント作成にはテストプログラムを作成し正しく動作するかの検証まで進んでいない

## ACKNOWLEDGEMENT

アルゴリズム等は、大阪市立大学学術情報総合センター中野秀男氏、大西克実氏、石橋勇人氏とメーリングリスト上でディスカッションをベースにしている。

各 PGP Public Keyserver の情報を調べるに当たり各サイトの管理者である Peter N. Wan <Peter.Wan@cc.gatech.edu>、Francisco Jesus Monserrat Coll <francisco.monserrat@rediris.es>、Teun Nijssen <Teun.Nijssen@kub.nl>の協力があった。

## 関連資料

- [1] URL <http://pgp.nic.ad.jp/docs/2001.txt>
- [2] 2002/1/27 Teun Nijssen <Teun.Nijssen@kub.nl>との私的メール交換
- [3] 2002/1/29 Francisco Jesus Monserrat Coll <francisco.monserrat@rediris.es>との私的メール交換
- [4] 2002/2/3 Peter N. Wan <Peter.Wan@cc.gatech.edu>との私的メール交換
- [5] The Art of Computer Programming, Donald, E. Knuth, 2nd ed, vol. 3, pp1, 1997.