排他的論理和を用いた(k, n)しきい値秘密分散法

(k, n)-Threshold Secret Sharing Scheme Using Exclusive OR

松 尾 正 克　Masakatsu Matsuo
武 藤 浩 二　Kouji Mutou

要　旨

多項式を用いる(k, n)しきい値秘密分散法では高速処理が困難なため，排他的論理和を用いた高速処理可能な(k, n)しきい値秘密分散法が期待されている．しかし，これら排他的論理和を用いる従来の(k, n)しきい値秘密分散法は，特定のしきい値k，分散数nでしか実現できず，クラウドにおけるデータバックアップなど，適用困難な用途が存在した．そこで当社は，任意のk, nで実現できる排他的論理和を用いた高速処理可能な(k, n)しきい値秘密分散法を考案した．本稿において，その概念，およびアルゴリズムを解説する．

Abstract

It is difficult to quickly divide or reconstruct secret data by using the (k, n)-threshold secret sharing scheme based on the polynomial interpolation scheme. Thus, there are hopes that the (k, n)-threshold secret sharing scheme using exclusive OR will speed up processing. But since the conventional (k, n)-threshold secret sharing scheme using exclusive OR can only be used in specific threshold, k and the number of participants, n, it is hard to apply some solutions such as a data backup solution in the cloud. Hence we devised a new (k, n)-threshold secret sharing scheme that can use any k and n, using exclusive OR to make high-speed processing possible. We explain the concept and algorithm in this report.

1. はじめに

近年，業務効率化やBCP（Business Continuity Plan）対策で，クラウドサービスやスマートフォン・タブレットなどのモバイル端末ソリューションが注目されている．しかし，モバイル端末紛失やクラウド事業者による情報漏えいリスクが大きく，十分に活用されていない．

この解決手段として期待されているのが，秘密分散法である．秘密分散法は，データを複数の断片（分散データ）に分散する技術で，分散データからは情報漏えいが発生しないという特長をもつ．この技術を有効に用いれば，クラウドやモバイル端末からの情報漏えいリスクを大幅に低減できる．

2. (k, n)しきい値秘密分散法

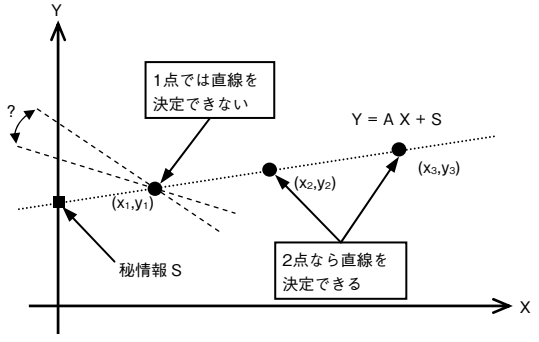
2.1 原理

秘密分散法の代表的な手法は，(k, n)しきい値秘密分散法である[1]．これは秘情報をn個に分散し，分散されたn個のうち，k個（2≦k≦n）を集めて秘情報を復元する技術で，kをしきい値，nを分散数と呼ぶ．

この原理を**第1図**に示す(2, 3)しきい値秘密分散で説明する．傾きAを物理乱数で決定後，秘情報Sを切片とする（1）式の直線から，任意の3点(x_1, y_1)，(x_2, y_2)，(x_3, y_3)を選択し，これを分散データとする．ただし，傾きAは安全のため，この直後に消去する．

$$Y = AX + S \quad \cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots\cdots (1)$$

これにより，どれか2つの分散データ（2点＝しきい値）で秘情報Sを復元でき，1つの分散データ（1点＝しきい値未満）では，秘情報を復元できない仕組みを実現できる．なお，k≧3の場合は，(k - 1)次曲線を利用する．



第1図　(2, 3)しきい値秘密分散
Fig. 1　(2, 3)-threshold secret sharing scheme

2.2 利点

下記に利点を記載する．

① (n - k)個の分散データが消失しても秘情報は復元可能である．したがって，RAID（Redundant Arrays of Inexpensive Disks）技術と同様に，記憶装置故障時のリカバリー対策や稼動率向上に利用できる．

② （1）式から明らかなように，分散データは，傾きA（物理乱数）を鍵データとして，秘情報を暗号化したものである．これは一種のバーナム暗号なので，通常の暗号とは異なり，計算量的安全性ではなく，情報理論的安全性をもつ．そのため，一部の分散データが流出しても，秘情報に関する手がかりは一切漏えいしない．これに対し，秘情報を暗号化しRAID技術などで分散する手法は，計算量的安全性であり，必ずしも安全とは言えない．

③ 暗号において鍵データの安全な保管は悩ましい課題である．しかし，(k, n)しきい値秘密分散法では，鍵データ（傾きA）は，（2）式～（4）式に示すように，分散データ(x_1, y_1)，(x_2, y_2)，(x_3, y_3)間の相対的位置情報（多項式計算）から復元できるので，鍵データの保管は不要である．

$$A = (y_3 - y_2) / (x_3 - x_2) \quad \cdots\cdots (2)$$
$$= (y_3 - y_1) / (x_3 - x_1) \quad \cdots\cdots (3)$$
$$= (y_2 - y_1) / (x_2 - x_1) \quad \cdots\cdots (4)$$

2.3 欠点

しきい値kの増加に伴い多項式計算の計算量が増加するため，通常の共通鍵暗号よりも遅くなる．

3. 先行研究

ここで，(k, n)しきい値秘密分散法の代表的な高速化手法を簡単に紹介する．

3.1 ガロア体を用いた高速化手法

ガロア体で多項式計算を行う高速化手法である[2]．しかし，しきい値kが非常に大きくなると，通常の共通鍵暗号よりも遅くなる．

3.2 分散データに鍵データを分配する高速化手法

鍵データを，分散データ間の相対的位置情報ではなく，分散データに付加して記憶する高速化手法である[3]．具体的には，秘情報と鍵データを排他的論理和（XOR：eXclusive OR）処理して分散データを生成し，この鍵データを他の分散データに記憶する．復元時は，各分散データから鍵データを集結し，これと分散データとの排他的論理和を計算し秘情報を復元する．分散・復元とも排他的論理和処理なので，通常の共通鍵暗号よりも高速である．

しかしながら，この手法は特定のしきい値k，分散数nでしか実現できず，用途が制限される．

4. 当社の高速化手法

当社は以下に解説する任意のk, nで実現可能な排他的論理和を用いた(k, n)しきい値秘密分散法を考案した．

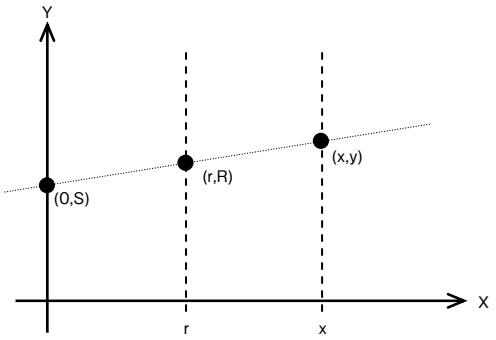
4.1 概念

当社手法では，鍵データは分散データと秘情報の相対的位置情報に記憶する．（2）式～（4）式と対比させれば，この手法は，（5）式に示すように，秘情報(0, S)と分散データ(x, y)から傾きA（鍵データ）を復元するものである．

$$A = (y - S) / x \quad \cdots\cdots (5)$$

実際は傾きAの代わりに点(r, R)を鍵データとし，分散データと秘情報からこの鍵データを復元するものとする．

この仕組みは，しきい値2では(2, 2)しきい値秘密分散で実現できる．**第2図**のように，(2, 2)しきい値秘密分散で秘情報を鍵データと分散データに分散すれば，鍵データは，分散データと秘情報から一意に求められるものになる．



第2図 (2,2)しきい値秘密分散
Fig. 2 (2,2)-threshold secret sharing

4.2 分散方法の原理

次に，この分散方法の原理を，**第3図**に示す秘情報$S = \{S_1, S_2, S_3\}$を(2, 3)しきい値秘密分散する事例で説明する．なお，以下のr1, f, g, hは既知の固定値とする．

① 物理乱数の鍵データ$R1 = \{R1_1, R1_2, R1_3, R1_4\}$を用意する．

② 2点$(0, S_1)$，$(r1, R1_1)$を通る直線と，X = fとの交点のY値をF_1とする．これは，点$(0, S_1)$を2点$(r1, R1_1)$，(f, F_1)に分散する(2, 2)しきい値秘密分散である．同様の操作を繰り返して，分散データ$F = \{F_1, F_2, F_3\}$を生成する．

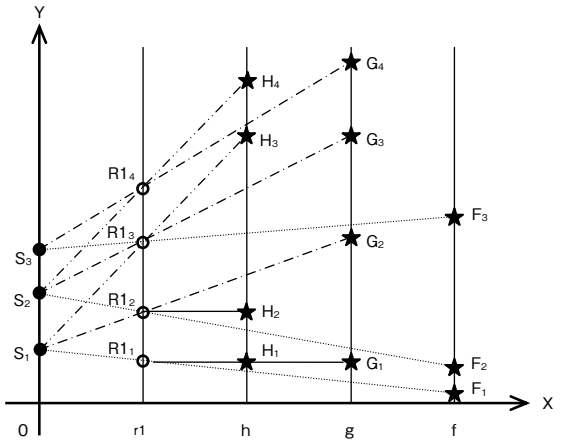
③ $G_1 = R1_1$とし，Sと$R1$の組合せが②とは異なる2点$(0, S_1)$，$(r, R1_2)$を通る直線と，X = gとの交点のY値をG_2とする．同様の操作を繰り返して，分散データ$G = \{G_1, G_2, G_3, G_4\}$を生成する．

④ $H_1 = R1_1$，$H_2 = R1_2$とし，Sと$R1$の組合せが②，③とは

異なる2点(0, S_1)，(r, $R1_3$)を通る直線と，X = hとの交点のY値をH_3とする．同様の操作を繰り返して，分散データH = {H_1, H_2, H_3, H_4}を生成する．

⑤ 安全のため，R1は消去する．

このように②，③，④と，(2, 2)しきい値秘密分散を3回実行することで，3個の分散データを生成する．



第3図　分散復元の原理
Fig. 3　Principle of sharing and recovery

4.3　復元方法の原理

次に，この復元方法の原理を，分散データF, Gを用いる事例で説明する．

① G_1より，$R1_1$を復元する．

② 次に，2点(r1, $R1_1$)，(f, F_1)を通る直線とX = 0の交点からS_1を復元する．

③ 次に，2点(0, S_1)，(g, G_2)を通る直線とX = r1との交点から$R1_2$を復元する．

④ 次に，2点(r1, $R1_2$)，(f, F_2)を通る直線とX = 0の交点からS_2を復元する．

⑤ 次に，2点(0, S_2)，(g, G_3)を通る直線とX = r1との交点から$R1_3$を復元する．

⑥ 次に，2点(r1, $R1_3$)，(f, F_3)を通る直線とX = 0の交点からS_3を復元する．

このように，片方の分散データと秘情報から鍵データの復元を行い，もう片方の分散データと鍵データから秘情報の復元を行うというように，玉突きで復元を行う．各分散データの生成でSとR1の組合せを変えたのは，この玉突きの復元を実現するためである．

4.4　排他的論理和による高速処理

次に，分散・復元を排他的論理和で行う方法を説明する．これまでは，(2, 2)しきい値秘密分散を実現するため，次の①～⑧の3点は同一直線上に配置した．

① 点(0, S_1)，点(r1, $R1_1$)，点(f, F_1)

② 点(0, S_2)，点(r1, $R1_2$)，点(f, F_2)

③ 点(0, S_3)，点(r1, $R1_3$)，点(f, F_3)

④ 点(0, S_1)，点(r1, $R1_2$)，点(g, G_2)

⑤ 点(0, S_2)，点(r1, $R1_3$)，点(g, G_3)

⑥ 点(0, S_3)，点(r1, $R1_4$)，点(g, G_4)

⑦ 点(0, S_1)，点(r1, $R1_3$)，点(h, H_3)

⑧ 点(0, S_2)，点(r1, $R1_4$)，点(h, H_4)

しかし，(2, 2)しきい値秘密分散は排他的論理和でも実現できる．例えば，①のS_1，$R1_1$，F_1は（6）式～（8）式に示すとおり，排他的論理和でも相互に算出でき，この方法でも分散データと秘情報から鍵データを復元できる．したがって，①～⑧はすべて排他的論理和による(2, 2)しきい値秘密分散に変換可能である．

$$F_1 = R1_1 \oplus S_1 \quad \cdots\cdots (6)$$

$$R1_1 = S_1 \oplus F_1 \quad \cdots\cdots (7)$$

$$S_1 = F_1 \oplus R1_1 \quad \cdots\cdots (8)$$

⊕は排他的論理和を示す

4.2節の分散処理をすべて排他的論理和で行えば，**第1表**に示すバーナム暗号になり，通常の共通鍵暗号よりも高速になる．

第1表　XORを利用した分散法
Table 1　Sharing scheme by using XOR

鍵データ	$R1_1$	$R1_2$	$R1_3$	
秘情報	S_1	S_2	S_3	
分散データ	$F_1=R1_1\oplus S_1$	$F_2=R1_2\oplus S_2$	$F_3=R1_3\oplus S_3$	

鍵データ	$R1_1$	$R1_2$	$R1_3$	$R1_4$
秘情報		S_1	S_2	S_3
分散データ	$G_1=R1_1$	$G_2=R1_2\oplus S_1$	$G_3=R1_3\oplus S_2$	$G_4=R1_4\oplus S_3$

鍵データ	$R1_1$	$R1_2$	$R1_3$	$R1_4$
秘情報			S_1	S_2
分散データ	$H_1=R1_1$	$H_2=R1_2$	$H_3=R1_3\oplus S_1$	$H_4=R1_4\oplus S_2$

同時に4.3節の復元も以下に示す排他的論理和だけの処理になり，これも通常の共通鍵暗号よりも高速になる．

① $R1_1 = G_1$

② $S_1 = R1_1 \oplus F_1$

③ $R1_2 = S_1 \oplus G_2$

④ $S_2 = R1_2 \oplus F_2$

⑤ $R1_3 = S_2 \oplus G_3$

⑥ $S_3 = R1_3 \oplus F_3$

5.　当社の(k, n)しきい値秘密分散法

5.1　分散アルゴリズム

次に，排他的論理和を用いた(k, n)しきい値秘密分散法の分散アルゴリズムを，秘情報S = {S_1, S_2,…, S_{12}}を，

(5, 6)しきい値秘密分散する事例で説明する．

① 物理乱数のk-1 = 4個の鍵データR1 = {$R1_1$, $R1_2$,⋯, $R1_{12}$}，R2 = {$R2_1$, $R2_2$,⋯, $R2_{12}$}，R3 = {$R3_1$, $R3_2$,⋯, $R3_{12}$}，R4 = {$R4_1$, $R4_2$,⋯, $R4_{12}$}を用意する．

② **第4図**に示すように，k = 5行の配列表を用意し，分散データ1の中央の行に秘情報Sを，1, 2, 4, 5行目にそれぞれ鍵データR1, R2, R3, R4を配置する．なお, 各行の先頭データをどの列に配置するかは，本質的な問題ではないので，説明は割愛する．

③ 分散データ1の1行目を右に2列，2行目を右に1列，4行目を左に1列，5行目を左に2列移動させたデータ配置を分散データ2とし，分散データ2を同手順で移動させたデータ配置を分散データ3とする. 以下同様に, 分散データ4, 5, 6のデータ配置を決める．ここで各分散データのデータ配置を変えるのは，玉突きの復元を実現するためである．

④ データ部の同列のデータは排他的論理和処理する．

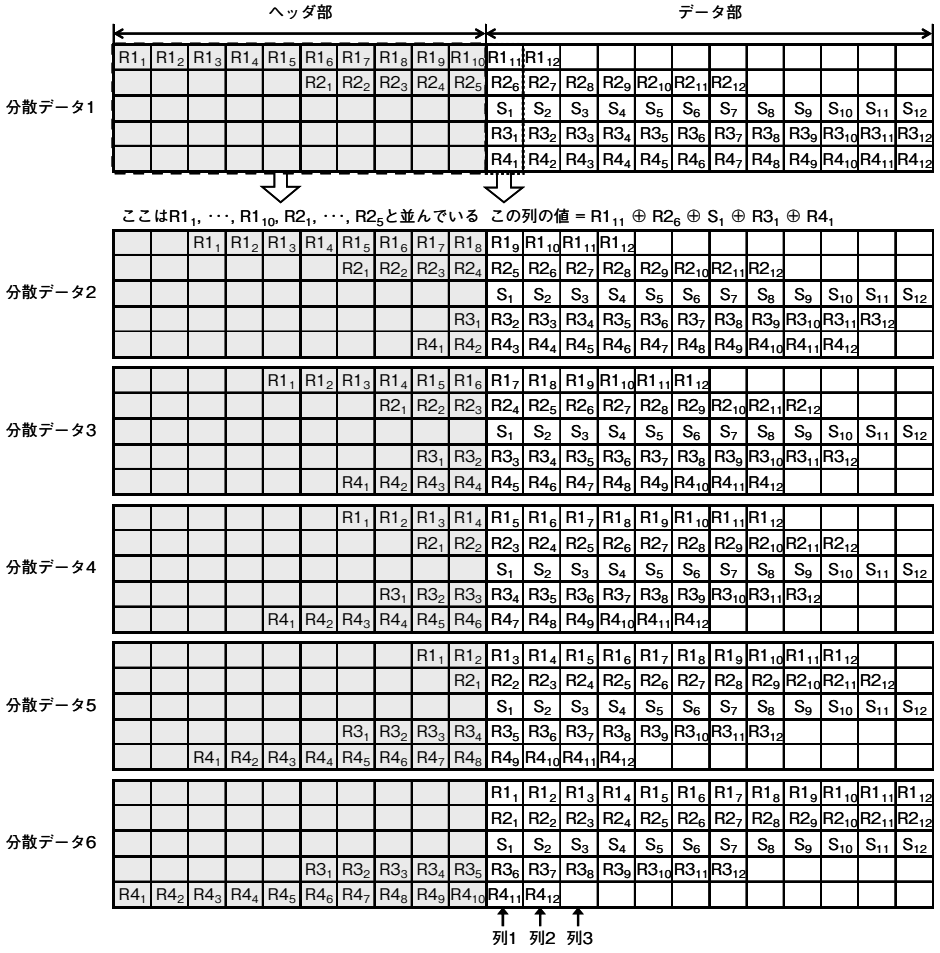
⑤ 安全のため，鍵データR1, R2, R3, R4は消去する．

⑥ ヘッダ部のデータはその分散データのヘッダ情報とする．なお，②の先頭データの配列方法を工夫すれば，このヘッダ情報はより小さくできる．

5.2 復元アルゴリズム

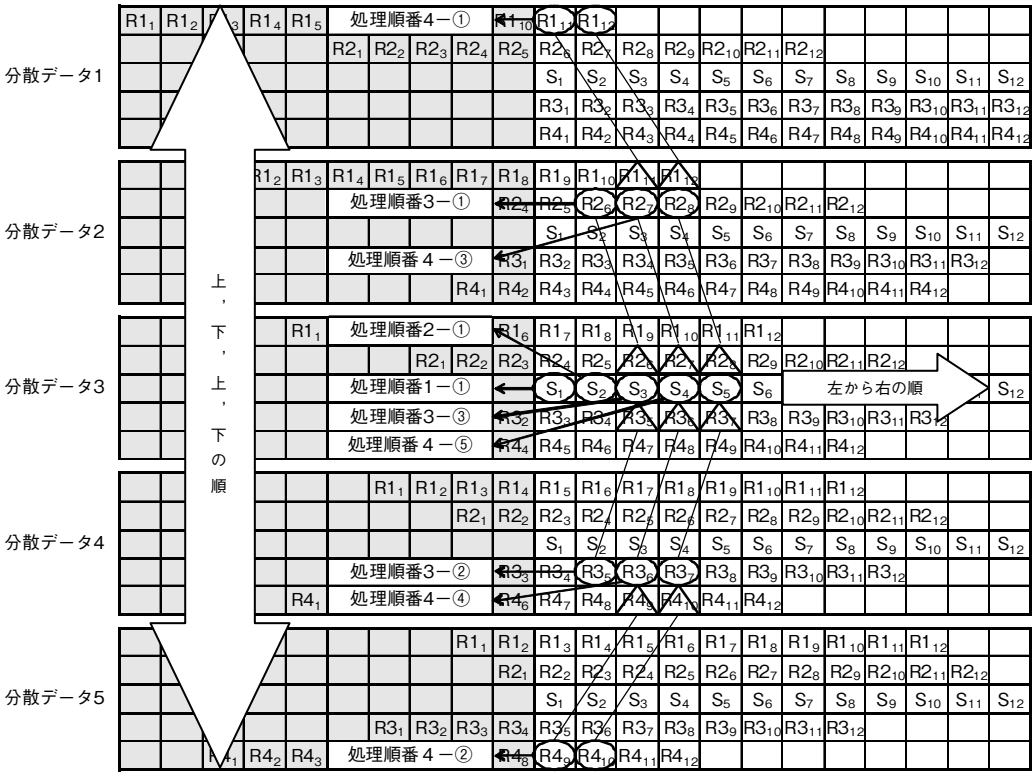
次に，排他的論理和を用いた(k, n)しきい値秘密分散法の復元アルゴリズムを，分散データ1～5を用いる事例で説明する．

① **第5図**に示すように，1順目は分散データ3のS_1(3行,1列)，2順目は分散データ3のS_2(3行,2列)，3順目は分散データ2⇒4⇒3の順に$R2_6$(2行,2列)，$R3_5$(4行,2列)，S_3(3行,3列)，4順目は分散データ1⇒5⇒2⇒4⇒3の順に，$R1_{11}$(1行,1列)，$R4_9$(5行,1列)，$R2_7$(2行,3列)，$R3_6$(4行,3列)，S_4(3行,4列)のデータを復元する．5順目以降は4順目と同様の手順で次列のデータを復元する．このように分散データは上・下・上・下の順に復元を行う．また，各分散データでは各行を左から右へ復元していく．



第4図 分散アルゴリズム

Fig. 4 Sharing algorithm



第5図　復元アルゴリズム
Fig. 5　Rrecovery algorithm

② 4順目までの実際の復元処理を**第6図**に示す．計算に必要なはじめのいくつかのデータは，ヘッダ情報から取得する．

なお，この分散・復元は任意のk, nで実現可能である．

1順目	①	S_1 = <<分散データ3の列1>>⊕$R1_7$ ⊕$R2_4$ ⊕$R3_3$ ⊕$R4_5$
2順目	①	S_2 = <<分散データ3の列2>>⊕$R1_8$ ⊕$R2_5$ ⊕$R3_4$ ⊕$R4_6$
3順目	①	$R2_6$ = <<分散データ2の列2>>⊕$R1_{10}$ ⊕S_2 ⊕$R3_3$ ⊕$R4_4$
	②	$R3_5$ = <<分散データ4の列2>>⊕$R1_6$ ⊕$R2_4$ ⊕S_2 ⊕$R4_8$
	③	S_3 = <<分散データ3の列3>>⊕$R1_9$ ⊕$R2_6$ ⊕$R3_5$ ⊕$R4_7$
4順目	①	$R1_{11}$ = <<分散データ1の列1>>⊕$R2_6$ ⊕S_1 ⊕$R3_1$ ⊕$R4_1$
	②	$R4_9$ = <<分散データ5の列1>>⊕$R1_3$ ⊕$R2_2$ ⊕S_1 ⊕$R3_5$
	③	$R2_7$ = <<分散データ2の列3>>⊕$R1_{11}$ ⊕S_3 ⊕$R3_4$ ⊕$R4_5$
	④	$R3_6$ = <<分散データ4の列3>>⊕$R1_7$ ⊕$R2_5$ ⊕S_3 ⊕$R4_9$
	⑤	S_4 = <<分散データ3の列4>>⊕$R1_{10}$ ⊕$R2_7$ ⊕$R3_6$ ⊕$R4_8$

第6図　復元処理
Fig. 6　Recovery processing

5.3　(k, L, n)ランプ型しきい値秘密分散

(k, n)しきい値秘密分散法は，**第5図**から明らかなように，各分散データのサイズは秘情報以上でメモリ効率が悪い．これを解決する手段に(k, L, n)ランプ型しきい値秘密分散法（2≦L≦k）がある[4]．(k, n)しきい値秘密分散法は完全秘密分散とも呼ばれ，しきい値未満（k-1個）の分散データを入手しても，情報理論的安全性があるので，秘情報に関する手がかりは一切漏えいしない．(k, L, n)ランプ型しきい値秘密分散法は，安全性を多少落とす（k-L個までが情報理論的安全性，k-L+1～k-1個は計算量的安全性）代わりに，分散データのサイズを秘情報の1/Lに圧縮する．

この技術の代表的な適用分野はクラウドである．クラウドのデータ保存においては，安全性や可用性（稼動率）だけでなく，経済性も重要な観点であり，メモリ効率向上は大きなポイントである．

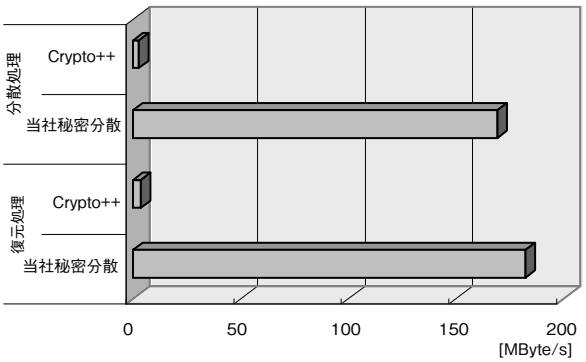
当社の(k, n)しきい値秘密分散法は，この(k, L, n)ランプ型しきい値秘密分散も容易に実現できる．例えば，第5図の鍵データの一部を秘情報に置き換え，秘情報を2行に分けて配置すれば，分散データサイズを約1/2に圧縮できる．同様に，秘情報を3, 4, 5行に分けて配置すれば，分散データサイズをそれぞれ約1/3, 1/4, 1/5に圧縮できる．

これは安全性の確認（k-L個までが情報理論的安全性）においてもメリットがある．一般の(k, L, n)ランプ型しきい値秘密分散法が専門的でユーザーが安全性を確認しづらいのに対し，当社方式では，秘情報が配置されている行数がLになるので，これを見ることで，安全性を容易に確認できる．

5.4　処理速度

次に，当社の秘密分散法とオープンソースの暗号ライ

ブラリCrypto++ (注) [5]による(5, 6)しきい値秘密分散のソフトウェア処理速度の比較を**第7図**に示す．**第4図，第5図**からわかるように，分散・復元ともデータのポインタアップと排他的論理和処理で実現されているため，当社の秘密分散法は，しきい値が大きくても，またソフトウェア処理であっても非常に高速に処理できる．



OS：Ubuntu 12.04 32 bit
CPU：Intel CORE i7 2.93 GHz
Crypto++ Version：5.6.2

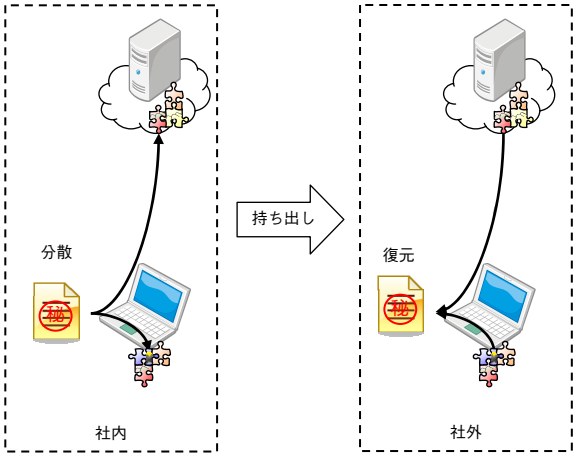
第7図　処理速度比較
Fig. 7　Processing speed comparison

6. 応用

高速処理という特長を生かし，当社はこの秘密分散技術をモバイル端末紛失時の情報漏えい防止ソリューションに活用している．

第8図に一例を示すが，文章や映像などの秘情報から生成した2つの分散データを，モバイル端末とクラウドにそれぞれ保存し，利用時にだけ秘情報をモバイル端末のRAM（Random Access Memory）上で復元するようにした．モバイル端末を紛失しても，クラウド上の分散データさえ消去すれば秘情報は復元不可能になるので，リモートワイプに比べ，確実に情報漏えいを防止することが可能である．

また5.3節で解説したように，当社の秘密分散法はクラウドへの親和性が高いので，パブリッククラウドを利用した安全・安価なデータバックアップソリューションやBCPソリューションにも応用可能である．



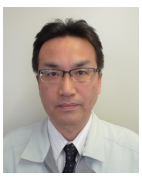
第8図　秘密分散法の応用
Fig. 8　Application of secret sharing scheme

参考文献

[1] Adi Shamir, “How to share a secret,” Communications of the ACM, vol. 22, no. 11, pp. 612-613, 1979.
[2] Kevin M. Greenan et al., “Analysis and construction of galois fields for efficient storage reliability,” Technical Report UCSC-SSRC-07-09, Aug. 2007.
[3] 保坂範和 他, “秘密分散法とその応用，” 東芝レビュー, vol. 62, no. 7, pp. 23-26, 2007.
[4] 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, no. 9, pp. 945-952, 1985.
[5] Wei Dai, “Crypto++ library 5.6.2,” Crypto++ library 5.6.2 - a free C++ class library of cryptographic schemes, http://www.cryptopp.com, 参照 Oct. 21. 2013

執筆者紹介

松尾　正克　Masakatsu Matsuo
パナソニック システムネットワークス（株）
先行技術開発センター
Advanced Technology Development Center,
Panasonic System Networks Co., Ltd.



武藤　浩二　Kouji Mutou
パナソニック システムネットワークス（株）
先行技術開発センター
Advanced Technology Development Center,
Panasonic System Networks Co., Ltd.

（注）Wei Dai 氏の米国における登録商標