数理解析研究所講究録
1093巻 1999年 156-161

分散RSA暗号における鍵生成と復号アルゴリズム
Key Generation and Decryption Algorithm in Distributed RSA Cryptosystem

宮崎 真悟　　櫻井 幸一
Shingo Miyazaki　　Kouichi Sakurai

九州大学大学院 システム情報科学研究科 情報工学専攻
〒812-8581 福岡市東区箱崎6-10-1
{shingo,sakurai}@csce.kyushu-u.ac.jp

概要： Bonehと Franklinは，複数の機関でRSAにおける鍵生成を行い，秘密鍵を(t,t)型で秘密共有する方式(D. Boneh and M. Franklin, CRYPTO '97 (1997))を示した．(2,2)型から(2,n)型へ変換する効率的な手法については同論文にて記されているが，(t,t)型から(t,n)型への変換は示されていない．そこで本稿では，Boneh-Franklin法を(t,n)型に拡張する手法について考察する．また，ユークリッドのアルゴリズムを用いたRSA暗号における分散復号アルゴリズムを示す．

キーワード： 閾値法，RSA，秘密分散，分散復号

1 はじめに

RSA暗号系に閾値法の概念を導入し，秘密鍵を(t,n)型で秘密分散させる方式 [FGMY97, ?] が提案されている．中でもFrankelら [FGMY97] は，秘密鍵そのものを算出することなく任意t個の機関で復号や署名を行える方式を示している．これは，合成数Nの異なる素因数知るディーラーが，素因数を知らなくても任意t個で復号や署名が行えるような秘密鍵dを作成することが特徴である．

一方，ディーラーの存在しない環境で，複数の機関で鍵生成を行う方式がBonehと Franklin [BF97] によって示された．この鍵生成を実行すると，秘密鍵dに対して(n,n)型の秘密分散が同時に行われることが特徴である．また保持する部分鍵を用いた全ての機関からの部分出力を合成することで，秘密鍵dを算出せずに暗号文を復号することができる．

この方法に閾値法の概念を導入して，同じ機能を実現させることが本稿の目的である．Frankel・MacKenzie・Yungは，Boneh-Franklin法に閾値法の概念を導入した方式 [FMY98] を提案している．彼らの方式では，Boneh-Franklin法に基づき得られる(n,n)型の秘密分散において，各機関P_iを部分鍵d_iのディーラーとみなす．それぞれの機関は全てのP_jからのd_jに対する共有情報を合成することで，最終的に秘密鍵dに対する秘密分散を行う．これにより，総数n個のうち任意t個の機関によるRSA秘密鍵dの鍵回復が可能である．

しかし，この方式では秘密鍵dそのものを算出せずに暗号文を復元することができない．これは，RSA公開鍵の素因数を誰も知らないため，巾部分での逆元を計算できないことが原因である．つまりはLagrangeの補間係数を計算できず，合成すれば平文を復元できるような部分出力を求めることが不可能となっている．

そこで本稿ではn個のうち任意t個による鍵回復に加え，分散復号を可能とするような手法を検討する．そのアプローチとして，以下二つを考える．一つは秘密分散において，dに関する(t,t)型の秘密分散を複数行い，組み合わせに応じた部分鍵を用いる方法である．これは文献 [BF97] で記されていた$(2,2)$型から$(2,n)$型の秘密分散を構築する手法を一般化する試みである．$(3,3)$型から$(3,3^2)$型を構成する手法は，Blackburn・Burmester・Desmedt・Wild [BBDW96] によって提案されている．しかし，同方法では$t=3$における$n>3^2$の場合と$t\geq 4$である場合には適用することができない．もう一つは文献 [FMY98] の手法で(t,n)型の秘密分散を行うが，公開鍵の選出を工夫し復号時にユークリッドアルゴリズムを応用する方法である．

前者については，機関の組み合わせによってはt個でも復号を行うことができないという問題がある．ただし，予め暗号文への入力を表として保管できるため，補間係数の計算を有するLagrange法よりも処理効率が良いという特徴がある．後者については，公開指数の選び方に制約がつくが，どんな組

み合わせでも確実に復号処理が行えるという特徴を持つ.

2 分散復号における問題点

文献 [FMY98] では, Boneh-Franklin 法 [BF97] に基づく (t,n) 型の鍵生成・共有方式を提案している. 彼らの方式では, まず (n,n) 型の鍵生成を行い, 秘密鍵 d に対する和の多項式を生成する. それから, 各機関 P_j が部分鍵 d_j のディーラーとなって, Sum-to-Poly の変換を行い, 結果, 秘密鍵 d の (t,n) 型の秘密共有を行う. これにより, 任意の t 個の機関は秘密鍵 d を算出することができる.

しかし, 秘密鍵 d そのものを算出せずに, 秘密鍵 d を用いた署名や復号を行う場合において, 同方式では問題が生じる. 今, 公開鍵 (e,N) で暗号化された暗号文 $C = M^e \pmod N$ がある. この暗号文 C を任意の t 個の機関[1]で復号する場合, 各機関 P_j は以下のような Lagrange の補間係数 $\lambda_{j,\Lambda}$ を計算する必要がある.

$$\lambda_{j,\Lambda} = \prod_{l \in \Lambda \setminus \{j\}} \frac{l}{l-j} \pmod{\phi(N)}$$

ところが, どの機関も合成数 N の素因数を知らないので, $\phi(N)$ を法とした $l-j$ の乗法逆元を計算することができない. つまり, Lagrange の補間係数を計算できないため, 以下のような分散復号を行うことできない.

$$\prod_{j \in \Lambda} C^{s_j \cdot \lambda_{j,\Lambda}} = C^d = M \pmod N$$

そこで, 本稿では (1) Lagrange の補間多項式を用いない (t,n) 型鍵共有方式と, (2) 公開鍵に制約をつけた Lagrange 補間法に基づく分散復号アルゴリズムを示す.

3 (2,n) 型閾値法 [BF97]

Boneh と Franklin [BF97] は, $(2,2)$ 型の秘密分散から $(2,n)$ 型の秘密分散 ($n \geq 3$ を効率的に構築する手法を示している (pp.237, footnote 参照). そのアルゴリズムを以下に示す.

ここでは簡単のため, 秘密情報 d そのものを知るユーザがいて, このユーザが d を $(2,n)$ 型で秘密分散することを考える. ここで, $r = \lceil \log n \rceil$ とする. まず, ユーザは $d = d_{0,0} + d_{0,1} = d_{1,0} + d_{1,1} = \cdots = d_{r,0} + d_{r,1}$ を満たす $(2,2)$ 型の秘密分散多項式を独立に $r+1$ 個作成する. $z \in [0,n]$ とし, z の 2 進表示 $z(2) = \beta_r \beta_{r-1} \ldots \beta_0$ とする. ユーザは z 番目の機関に, $r+1$ 個の部分情報: $\{d_{r,\beta_r}, d_{r-1,\beta_{r-1}}, \ldots, d_{0,\beta_0}\}$ を送る.

機関の番号を唯一に設定することで, 任意の二つの機関 $i,j (i \neq j)$ はその異なるビットでの $(2,2)$ 型秘密分散から d を復元することができる.

4 (t,n) 型閾値法への拡張

4.1 (t,n) 型秘密分散

3節で示した Boneh-Franklin 法を一般的 (t,n) 型に拡張する手法を示す. ここでは, まず文献 [BF97] の手法を用いて, n 個の機関で RSA 暗号の秘密鍵 d・公開鍵 (e,N) を分散生成する. 各機関 j は d の部分情報 d_j を保持し, d_j のディーラーとなって (t,n) 型の秘密分散を行う. この操作を全ての機関が行うことで, 秘密鍵 d に対する (t,n) 型の秘密共有を実現する.

以下に具体的プロトコルを示す.

Step.1: まず n つの機関は [BF97] 方式を用いて, RSA 公開鍵 (e,N) と秘密鍵 d を生成する. ここで今, n 個の機関は N の異なる二つ素因数と秘密鍵 d を (n,n) 型で秘密共有している. 生成される秘密鍵 d は

$$d = d_1 + d_2 + \cdots d_n$$

の関係式を満たし, 各機関 $i (1 \leq i \leq n)$ は d_i を保管している. 以下, 各機関 j は d_i のディーラーとして機能する.

Step.2: 各ディーラー i は乱数 $S_{j,k} (0 \leq j \leq r, 0 \leq l \leq t-2)$ を生成する. ここで, $r = \lceil \log_t n \rceil$ とする. また,

$$S^{(i)}_{j,t-1} = d_i - \sum_{l=0}^{t-2} S_{j,l}$$

とする.

Step.3: 各ディーラー i は以下のようにして機関 $T_z (1 \leq z \leq n)$ に情報を送る.

[1] この集合を Λ とする

(a) z を t 進数表示する.

$$z = \beta_{r,z}t^r + \beta_{r-1,z}t^{r-1} + \cdots + \beta_{0,z}$$

簡単のため，以下のように表記する.

$$z(t) = \beta_{r,z}\beta_{r-1,z}\ldots\beta_{0,z}\ (\beta_{j,z} \in \{0,\ldots,t-1\})$$

(b) 機関 z には，$(S^{(i)}_{r,\beta_{r,z}}, S^{(i)}_{r-1,\beta_{r-1,z}}, \cdots, S^{(i)}_{0,\beta_{0,z}})$ を送信する．$1 \le z \le n$ である.

Step.4: 各機関 $T_z(1 \le z \le n)$ は，ディーラー i から受信した $(S^{(i)}_{r,\beta_{r,z}}, S^{(i)}_{r,\beta_{r-1,z}}, \ldots, S^{(i)}_{0,\beta_{0,z}})$ から，以下のようにして $d_{j,k}$ を計算する．ただし，$k \in \{\beta_{r,z}, \beta_{r-1,z}, \ldots, \beta_{0,z}\}, 0 \le j \le r$ である.

$$d_{j,k} = \sum_{i=1}^{t} S^{(i)}_{j,k} = S^{(1)}_{j,k} + S^{(2)}_{j,k} + \cdots + S^{(t)}_{j,k}$$

各機関 T_z は，$r+1$ 個の部分秘密情報 $d_{j,k}$ を保管する.

秘密鍵 d と部分秘密鍵 $d_{j,k}$ との関係は以下のようになっている.

$$\begin{aligned} d &= d_{0,0} + d_{0,1} + \cdots + d_{0,t-1} \quad (t^0\text{桁目}) \\ &= d_{1,0} + d_{1,1} + \cdots + d_{1,t-1} \quad (t^1\text{桁目}) \\ &= \qquad\qquad \vdots \\ &= d_{r,0} + d_{r,1} + \cdots + d_{r,t-1} \quad (t^r\text{桁目}) \end{aligned}$$

例えば，(3,27) 型秘密分散における機関 $z = 20$ の保管情報は以下の通りである.

$$\begin{aligned} r &= \log_3 27 = 3 \\ 20 &= 2 \times 3^3 \times 0 + 3^2 + 0 \times 3^1 + 2 \times 3^0 \\ 20(3) &= 0202\ (\text{3進表示}) \end{aligned}$$

よって，機関 20 は

$$\begin{aligned} d &= \underline{d_{3,0}} + d_{3,1} + d_{3,2} \quad (3^3\text{桁}) \\ &= d_{2,0} + d_{2,1} + \underline{d_{2,2}} \quad (3^2\text{桁}) \\ &= \underline{d_{1,0}} + d_{1,1} + d_{1,2} \quad (3^1\text{桁}) \\ &= d_{0,0} + d_{0,1} + \underline{d_{0,2}} \quad (3^0\text{桁}) \end{aligned}$$

$(d_{3,0}, d_{2,2}, d_{1,0}, d_{0,2})$ を d の部分鍵として保管する.

4.2 分散復号化

n の機関のうち，任意の t つの機関 T_z（この集合を Λ とする）はグループの公開鍵で暗号化されたデータ $C = M^e \pmod N$ を以下のようにして分散復号化する.

Step 1: t つの参加機関 T_z 全ての識別番号 j を t 進表示する $(z(t) = \beta_{r,z}\beta_{r-1,z}\cdots\beta_{0,z})$.

Step 2: $0 \le i \le r$ に対し，全ての $a, b \in \Lambda (a \ne b)$ に対し $\beta_{i,a} \ne \beta_{i,b}$ であるような i を見つける．これを満たす i がない場合，この集合による分散復号はできないものとし多少メンバーを入れ換えて Step 1 からやり直す.

Step 3: Step 2 で条件を満たす i に対し，各 T_z は t^i 桁に対応する部分鍵 $d_{i,k}$ $(k = \beta_{i,z})$ を用いて部分出力 $X_z = C^{d_{i,k}} \pmod N$ を計算する.

Step 4: 各機関からの部分出力 $X_z (z \in \Lambda)$ を合成することでメッセージを復元できる.

$$\begin{aligned} \prod_{z\in\Lambda} &= (M^e)^{d_{i,1}+d_{i,2}} + \cdots + d_{i,t} \\ &= (M^e)^d \\ &= M \pmod N \end{aligned}$$

5 拡張方式の問題点と比較

5.1 分散復号の処理不能性

文献 [Ped91b] で提案されている方式では，n 個のうち任意の t 個の機関はグループの鍵で暗号化された暗号文を復号することができる．しかし，4.1節に示した (t,n) 型秘密分散への拡張法では，選ばれた t つの機関が分散復号できない場合がある.

(3,27) 型の秘密分散を例にこれを示す．今，秘密鍵 d について，以下の $r+1$ 個の関係式が成り立っている.

$$\begin{aligned} d &= d_{3,0} + d_{3,1} + d_{3,2} \quad (3^3\text{桁}) \\ &= d_{2,0} + d_{2,1} + d_{2,2} \quad (3^2\text{桁}) \\ &= d_{1,0} + d_{1,1} + d_{1,2} \quad (3^1\text{桁}) \\ &= d_{0,0} + d_{0,1} + d_{0,2} \quad (3^0\text{桁}) \end{aligned}$$

まず分散復号が可能な場合を示す．27つの機関のうち，機関 P_{20}, P_{23}, P_{26} がプロトコルに介入する時，復号操作 $C^d = M^{ed} = M$ は成功する．今，各機関の3進表示は

$$P_{20} = 0202 \quad , \quad P_{23} = 0212 \quad , \quad P_{26} = 0222$$

であり，それぞれの保管情報は以下の通りである．

	P_{20}	P_{23}	P_{26}
3^3 桁目	$d_{3,0}$	$d_{3,0}$	$d_{3,0}$
3^2 桁目	$d_{2,2}$	$d_{2,2}$	$d_{2,2}$
3^1 桁目	$d_{1,0}$	$d_{1,1}$	$d_{1,2}$
3^0 桁目	$d_{0,2}$	$d_{0,2}$	$d_{0,2}$

この場合，3^1 桁に注目すると，

$$d = d_{1,0} + d_{1,1} + d_{1,2}$$

から，d による復号化処理が可能となる．

$$C^{d_{1,0}} C^{d_{1,1}} C^{d_{1,2}} = M^{ed} = M \pmod N$$

ところが，P_{20}, P_{23}, P_{11} の三つの機関では分散復号を行うことができない．各機関の3進表示は

$$P_{20} = 0202 \quad , \quad P_{23} = 0212 \quad , \quad P_{11} = 0102$$

であり，各機関の保管情報は以下のようになる．

	P_{20}	P_{23}	P_{11}
3^3 桁目	$d_{3,0}$	$d_{3,0}$	$d_{3,0}$
3^2 桁目	$d_{2,2}$	$d_{2,2}$	$d_{2,1}$
3^1 桁目	$d_{1,0}$	$d_{1,1}$	$d_{1,0}$
3^0 桁目	$d_{0,2}$	$d_{0,2}$	$d_{0,2}$

この場合，d を構成する3つの成分 $(d_{j,0}, d_{j,1}, d_{j,2})$ がどの桁を見ても揃わない．このように保管情報の衝突により，復号操作の出来ない場合が考えられる．

このように衝突が生じる場合，その t つの機関では分散復号処理を行えないため，別の t つの機関を選び直す必要がある．Pedersen方式 [Ped91b] のように任意の t つの機関で分散復号・鍵復元が行うには，最悪，秘密鍵 d に関する ${}_nC_t$ 個の独立な関係式を作れば良い．つまり，ある組み合わせ（Λ とする）の時は該当する関係式：

$$d = d_{1,\Lambda} + d_{2,\Lambda} + \cdots + d_{t,\Lambda}$$

により分散復号を行うという形である．しかし，n が大きくなるほど必要な関係式の数は膨大になる．

n が大きく，閾値 t 以上の機関を比較的自由に選択できるのであれば，4.1節の (t,n) 型の秘密分散を行うことが考えられる．そこで復号不能状態の場合は異なる t つの機関を再編成する．再編成というよりも復元可能なグループで復号プロトコルを開始する形である．

逆に n の数が小さく機関の選択余地がない場合は，全ての組み合わせに応じた関係式を作成する方が適している．どちらの手法を取るかは，使用する環境にもよりそうである．復号プロセスに加わることが大きな意味を持っており，募ったメンバーによる絶対の処理成功が要求されている環境には 4.1節の (t,n)-sharing は向かない．逆に復号メンバーの編成が比較的容易く，様々な機関による代理が期待できそうな環境なら，あえて ${}_nC_t$ つの独立関係式を用意する必要はないであろう．メンバーの数や処理の効率，使用される環境に応じた使い分けを行うことができる．

5.2 補間法に基づく方式との比較

Lagrange の補間法に基づく閾値法を用いた分散復号化では，秘密鍵が各機関の持つ部分鍵の単純和ではない．そこで各機関で選ばれた集合に対する Lagrange の補間係数を計算し合成する必要がある．これを事前に計算して表として保管する場合，一つの部分鍵情報と協力する機関の全ての組み合わせに応じた $\binom{n}{t}$ 個の補間係数が必要である．

一方，4.1節での (t,n) 型秘密分散法は，$\log_t n + 1$ 個から機関の組み合わせに対応する部分鍵を選ぶだけである．よって保存情報としては $\log_t n + 1$ 個である．

復号処理不能が許されない場では，補間法に基づく閾値法が使用を余儀無くされるだろう．ただし，t 個の機関を比較的自由に選べ，また効率の良い処理が要求される場では4.1の方式が有効である．

6 ユークリッドアルゴリズムを用いた分散復号化

ユークリッドアルゴリズムを用いた分散復号化法を提案する．今，Non-Dealer モデルにおける Frankel・MacKenzie・Yung の秘密分散共有

法 [FMY98] を用いて，n 個の機関 $\mathcal{P}_j$ が RSA 暗号化鍵 (e,N) を生成かつ秘密鍵 d を (t,n) 型の秘密分散共有を行っている．ここで，$t-1$ 個の乱数 $a_1,\ldots,a_{t-1} \in Z$ に対し，以下の多項式を定める．

$$f(x) = d + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1}$$

各機関 $\mathcal{P}_j$ は，$N = pq = (p_1 + p_2 + \cdots + p_n)(q_1 + q_2 + \cdots + q_n)$ を満たす (p_j, q_j) と $s_j = f(j)$ を分散秘密情報として保管している．

今，ユーザ $\mathcal{U}$ は n 個の機関で成る組織の鍵で暗号化された暗号文 $C = M^e \pmod N$ を持っている．$\mathcal{U}$ は以下のようにして，任意 t 個の機関（この集合を Λ）に暗号文 C を分散復号してもらう．ここで，$L = n!$ かつ $\gcd(e, L^2) = 1$ とする．

分散復号化プロトコル：

Step 1: $\mathcal{U}$ は C を各機関 $\mathcal{P}_j (\in \Lambda)$ に送る．

Step 2: 各機関 $\mathcal{P}_j$ は，部分鍵 s_j を用いて部分出力 Z_j を計算し，$\mathcal{U}$ に送る．

$$\begin{aligned} \lambda_{j,\Lambda} &= \prod_{l \in \Lambda \setminus \{j\}} \frac{l}{l-j} \\ \sigma_j &= s_j \cdot L^2 \cdot \lambda_{j,\lambda} \\ Z_j &= C^{\sigma_j} \pmod N \end{aligned}$$

Step 3: $\mathcal{U}$ は t 個の $\mathcal{P}_j$ からの部分出力を合成し，$M^{L^2} \pmod N$ を求める．

$$\prod_{j \in \Lambda} Z_j = (M^e)^{L^2 \cdot \sum_{j \in \Lambda} (s_j \cdot \lambda_{j,\Lambda})} = M^{L^2} \pmod N$$

Step 4: $\mathcal{U}$ は異なる二つの暗号文 $(C_1, C_2) = (M^{L^2}, M^e)$ から，以下のようにして M を算出する．

[4a]: $a_1 = (L^2)^{-1} \pmod e$

[4b]: $a_2 = (a_1 L^2 - 1)/e$

[4c]: $M = C_1^{a_1} (C_2^{a_2})^{-1} \pmod N$

ここで，プロトコルの健全性を以下に示す．

$$C_1^{a_1} (C_2^{a_2})^{-1} = M^{L^2 a_1 - e a_2} = M \pmod N$$

7 他方式との比較

7.1 Frankel-Gemmell-MacKenzie-Yung 方式

t を閾値として n 個の機関に秘密を分散預託する際，まず Dealer は次のようにして変形 RSA 公開鍵 (e,N) と秘密鍵 d を生成する（$L = n!$ とする）．$H = \gcd(e, L^2)$ として，$eP + \frac{L^2}{H}\tilde{s} = 1$ を満たす $(P, \tilde{s})$ を計算する．それから，$d \equiv P + L^2 k \pmod{\phi(N)}$ を満たす k を計算する．ただし，$k \equiv d\tilde{s}H^{-1} \pmod{\phi(N)}$ である．ここで $f(0) = L^2 k$ であるような関数 $f(x)$ を以下のように定める．

$$f(x) = f_0 + f_1 x + f_2 x^2 + \ldots + f_{t-1} x^{t-1}$$

ただし，$f_j \in_R \{0, L, \ldots, 2L^3 n^{2+\epsilon} t\} (1 \le j \le t-1)$ とする．ディーラーは $s_j = f(j)$ を計算し，P と s_j を各機関 j に送信する．

ディーラーによる分散共有が完了した状況で，複数の機関による分散復号プロトコルを考える．今，依頼人 X は，ディーラーの公開鍵 (e,N) で暗号化されたメッセージ：

$$C = M^e \pmod N$$

を持っている．X は複数の機関に暗号文 C を入力として与え，メッセージ M を入手したい．ここで任意の t 個の機関（この集合を Λ）が処理に参加するものとする．

Step.1: X は C を処理に関係する各機関 j に送信する．

Step.2: 各機関 j は s_j を用いて $\sigma_j = s_j \lambda_{j,\Lambda}$ を求め，以下を計算する．

$$Z_j = C^{\sigma_j} \pmod N \ , \ \lambda_{j,\Lambda} = \prod_{l \in \Lambda \setminus \{j\}} \frac{l}{l-j}$$

機関 j は Z_j を秘密吏に X に送信する．いずれかの機関は $C^P \pmod N$ を X に送る（誰でも $(P, \tilde{s})$ を計算することができるので，必ずしも C^P を送る必要はない）．

Step.3: X は，Z_j と $M^P \pmod N$ から，

$$\begin{aligned} C^P \prod Z_j &= C^{P + \sum_j s_j \lambda_{j,\Lambda}} \\ &= M^{ed} \\ &= M \pmod N \end{aligned}$$

を計算することで，メッセージ M を復号する．

RSA 暗号では署名と復号化操作が同じであるので，上記プロトコルにおいて依頼人が入力として文書 M を与えると，出力として署名 M^d が得られる．

7.2 比較

Frankel-Gemmell-MacKenzie-Yung 方式も本提案方式も，α, β を係数として以下のユークリッド公式を考えている．

$$e\alpha + L^2\beta = 1, \quad \text{where } \gcd(e, L^2) = 1$$

ここで，Frankel らの方式では $(\alpha, \beta) = (P, \tilde{s}/H)$，本提案方式では $(\alpha, \beta) = (-a_2, a_1)$ としたものである．Frankel らの方式では，ディーラーが鍵分散時にユークリッド公式を作成するので，復号時には各機関からの部分出力を合成するだけでよい．本提案方式では復号時に部分出力を合成し，その出力からユークリッドアルゴリズムにより処理を行う分，効率が悪い．しかし Frankel らの方式は $\phi(N)$ を知っているディーラーの存在が前提である．一方，本提案方式ではディーラーの存在有無に関わらず，秘密を分散共有している n 個の機関うち，t 個の任意の機関で RSA 署名・復号が可能である．

8 おわりに

本稿では，ディーラーのいない環境下での (t, n) 型分散復号について検討した．Lagrange 補間法を用いない手法では，分散復号時に補間係数を計算する手間が省ける分だけ効率が良い．ただし，保持情報の衝突による復元不能の問題がある．最悪 $\binom{n}{t}$ の独立多項式を用意すれば衝突の可能性はないが，独立多項式を最小限に抑える手法を検討する必要があろう．もはやディーラーの有無は関係なく，n 羽のうち任意 t 羽の鳥を異なる巣箱に入れるような関数群を見つける問題でもある（[KS96]）．

参考文献

[BBDW96] S. R. Blackburn, M. Burmester, Y. Desmedt and P. R. Wild, "*Efficient multiplicative sharing schemes*," Advances in Cryptology – EUROCRYPT '96, pp. 107-118, 1996.

[BF97] D. Boneh and M. Franklin, "*Efficient generation of shared RSA keys*," Advances in Cryptology – CRYPTO '97, LNCS 1294, pp. 425-439, 1997.

[Des97] Y. Desmedt, "*Some recent reserch aspects of threshold cryptography*," Information Security, LNCS 1396, pp. 158-173, 1997.

[DF89] Y. Desmedt and Y. Frankel, "*Threshold cryptosystems*," Advances in Cryptology – CRYPTO '89, LNCS 435, pp. 307-315, 1989.

[FGMY97] Y. Frankel, P. Gemmell, P. D. MacKenzie and M. Yung, "*Optimal-resilience proactive public-key cryptosystems*," 38th Annual Symposium on Foundations of Computer Science, pp. 384-393, 1997.

[FMY98] Y. Frankel, P. D. MacKenzie and M. Yung, "*Robust efficient distributed RSA-key generation*," Proceedings of the thirtieth annual ACM symposium on theory of computing, pp. 663-672, 1998.

[FY98] Y. Frankel and M. Yung, "*Distributed public key cryptosystems*," (Invited Paper) Public Key Cryptography, LNCS 1431, pp. 1-13, 1998.

[KS96] K. Kurosawa and D. Stinson, Personal communication, June 1996 (Referred in Desmedt's paper [Des97]).

[Ped91a] T. P. Pedersen, "*Distributed provers with applications to undeniable signatures*," Advances in Cryptology – Eurocrypt '91, LNCS 547, pp. 221-238, 1991.

[Ped91b] T. P. Pedersen, "A threshold cryptosystem without a trusted party," Advances in Cryptology – Eurocrypt '91, LNCS 547, pp. 522-526, 1991.

[Sha79] A. Shamir, "*How to share a secret*," Communication ACM, 22, pp. 612-613, 1979.

[Sim83] G. J. Simmons, "A 'weak' privacy protocol using the RSA cryptoalgorithm," Cryptologia, vol. 7, pp. 180-182, 1983.