ドキュメントの直接検索と複数検索クエリに対応できる秘密分散法を用いた秘匿検索

ムハンマド カマル アフマド アクマル アミヌディン[1,*]　岩村 恵市[1]

概要：複数の文字からなるドキュメントを暗号化して保存し，それを復元することなく一部の文字列からそのドキュメントを検索することを考える．この場合，共通鍵暗号方式と公開鍵暗号方式を用いる検索可能暗号は広く研究されている．しかし，これまで提案された手法では，すべてインデックスという概念を用いる．よって，登録されたインデックス以外の検索はできないという問題が発生する．また，検索可能暗号において実現される主な検索機能は，1 つの検索クエリを用いる方法と複数の検索クエリを用いる方法に分けられる．1 つの検索クエリを用いた検索可能暗号は広く研究されているが，複数検索クエリに対応できる検索可能暗号は少ない．よって，本論文では，軽い計算量を持つ秘密分散法を用いて，インデックスを用いないドキュメントからの直接検索を実現し，論理積演算及び論理和演算によって複数検索クエリに対する論理積検索と論理和検索を実現できる手法を提案する．また，提案手法が様々なコンテンツの秘匿検索に適用できることを示す．

キーワード：秘密分散，秘匿計算，秘匿検索，直接検索

Multiple Keywords Search of Document with Direct Searching using Searchable Encryption based on Secret Sharing Scheme

Ahmad Akmal Aminuddin Mohd Kamal[1,*]　Keiichi Iwamura[1]

Abstract: Suppose that a document that is made up of multiple words is being encrypted and stored, and we want to search for that document using some keywords without decrypting the original encrypted document. In this case, there has been a lot of studies focusing on searchable encryption that uses public key encryption and symmetric key encryption. However, most of the methods proposed so far utilized the concept of index searching. Therefore, there is a problem where keywords that are not registered in the index is not searchable. In addition, searching functions realized by searchable encryption can be divided into single query search and multiple queries search. Searchable encryption using single query has been widely studied; however, few methods of searchable encryption can realize multiple queries search. Therefore, in this paper, by using secret sharing scheme with lower computation cost, we realize direct searching without the use of an index. In addition, we also realize searching of multiple queries using the approach of logical AND and logical OR.

Keywords: Secret-Sharing Scheme, Secrecy Computation, Searchable Encryption, Direct Search

1. はじめに

近年，計算機の演算能力や通信技術が向上し，クラウドコンピューティングの普及が急速に進んでいる．クラウドコンピューティングでは，データを複数のサーバに保存するだけではなく，処理もサーバに行わせることが求められる．また，データのプライバシー及び秘匿性を確保するため，情報を暗号化して保管する必要がある．しかし，通常の暗号を用いた場合，暗号化データを復元すること無しにサーバ上で検索を行うことが出来ない. よって, 近年では，暗号化されたデータを復号することなく，正当なユーザまたは許可されたユーザだけがその情報を取り出せる検索可能暗号と呼ばれる手法が多数提案されている[2, 3, 4, 7, 8, 9].

既存の検索可能暗号の１つとして，共通鍵暗号方式を用いる手法があげられる[4, 8]．しかし，共通鍵暗号を用いた検索可能暗号は，検索者は暗号化の鍵を知っている必要があるため，検索者は暗号化した秘密鍵を知る登録者に限られるか, 復号を希望する者に秘密鍵を渡さなければならず，その者はクラウド上に保存されているデータをすべて復元することが可能となるという問題がある．

よって，近年では，公開鍵暗号を用いた検索可能暗号が注目されている．例えば，2004 年に Boneh らは[3]，データのオーナが公開鍵を用いてデータを暗号化してクラウドに保管し，秘密鍵を持つユーザが検索用のタグを生成し，クラウドに検索を実行させる公開鍵暗号を用いた検索可能暗号を提案した．しかし，公開鍵暗号を用いる検索可能暗号は計算量が膨大であることが知られている．

ここで，複数の文字からなるドキュメントを暗号化して保存し，それを復元することなく一部の文字列からそのドキュメントを検索することを考える．この場合，前述のよ

1 東京理科大学
Tokyo University of Science
* ahmad@sec.ee.kagu.tus.ac.jp

うに共通鍵暗号方式を用いる検索可能暗号では検索者が登録者に限定されるか，検索を希望する検索者に秘密鍵を渡さなければならないため，秘匿したドキュメントが漏洩する可能性があるという問題がある．また，公開鍵暗号を用いた検索可能暗号は計算量が膨大で，大量のドキュメントからドキュメントの検索をするには非効率と考えられる．よって，ドキュメント検索に適した検索可能暗号は今まで提案されていない．

それに対して本論文では，軽い計算量を持つ秘密分散法を用いるというアプローチをとる．ここで，秘密分散法とは，ある秘密情報を複数の異なる分散値に変換し，分散する手法である．(k,n)閾値秘密分散法と呼ばれる手法は，1つの秘密情報をn個の分散値に変換し，n台のサーバに分散する．(k,n)閾値秘密分散法の特徴は，分散したn個の分散値から，k個の分散値を集めれば，元の秘密情報を復元することができるが，k個未満の情報からは，秘密情報に関する情報を一切得ることができないということである．今まで秘密分散法を用いた秘匿検索法は提案されておらず，[12]において発表された手法が世界初と考えられる．ただし，この手法は Shamir によって提案された多項式を用いる(k,n)閾値秘密分散法[1]を用いるため高速化処理が考慮されておらず，検索時間などに実用的な問題が残った．

一方，辻下等によってパスワードが一致したときのみ効率的に秘密情報が取り出せるパスワード付き秘密分散法[11] (以降，PPSS)と呼ばれる手法が提案されているが，これはパスワードの一致を検出するための手法であり，ドキュメントへの適用は考えられていない．そこでまず本論文では，辻下等の PPSS を変形し，ドキュメント検索に適用できることを示す．この手法は一部を除いて Shamir による秘密分散法ではなく，栗原らによって提案された XOR によって処理を実現する XOR 法[5]と呼ぶ秘密分散法を用いているため，高速な検索が実現できる．

しかし，[11]を含め今まで提案されている PPSS は 1 つのパスワードの一致を検証できるのみであり，複数のパスワードの一致を検証できない．しかし，これでは複数文字列からなるドキュメント検索においては 1 文字のみの一致検索しかできない．この場合，1 文字単位でドキュメント中の文字の配置が漏洩する可能性がある．そこで，辻下等による PPSS を拡張し，複数文字列の完全一致検索を実現する手法を提案する．これによって，複数パスワードの一致検索も実現できるようになる．秘密分散法を用いた複数文字列または複数パスワードの完全一致検索は今まで提案されていない．さらに，提案手法が様々なコンテンツに対して適用できることを示す．

本論文の構成を以下に示す．第 2 章では検索可能暗号を含む従来研究について説明する．その後，第 3 章で PPSS を適用した 1 文字単位の検索手法を示す．また，第 4 章と第 5 章で複数文字列の AND 検索と AND・OR 検索手法を提案し，その安全性を示す．

2. 関連研究

2.1 (k,n)閾値秘密分散法

次の二つの条件を満たす秘密分散法を，(k,n)閾値秘密分散法という．

- $k-1$個以下の分散値からは，秘密情報sに関する情報は一切得ることはできない．
- 任意のk個以上の分散値から，元の秘密情報sを復元することができる．

代表的な(k,n)閾値秘密分散法として Shamir によって提案された Shamir の(k,n)閾値秘密分散法[1]（以降，(k,n)Shamir 法）と，栗原らによって提案された XOR を用いる(k,n)閾値秘密分散法[5]（以降，(k,n)XOR 法）がある．

2.2 共通鍵暗号を用いる検索可能暗号

共通鍵暗号を用いる検索可能暗号は 2000 年に Song ら[2]によって提案された．それをもとに，Curtmola ら[4]はいくつかの改良を試みている．ただし，従来の共通鍵暗号による暗号化では，暗号化したデータ同士で比較を行うために，検索者はデータのオーナに限定されるか，検索を希望する検索者に秘密鍵を渡す必要が生じてしまう．よって，検索者はクラウド上に保存されているデータをすべて復元することが可能となるという問題がある．また，近年では，それを解決できる手法が研究されているが，それを実現するための演算処理は非常に複雑になる．

2.3 公開鍵暗号を用いる検索可能暗号

2004 年に Boneh ら[3]は，データのオーナが公開鍵を用いてデータを暗号化してクラウドに保管し，秘密鍵を持つ検索者が検索用のタグを生成し，クラウドに検索を実行させる公開鍵暗号検索可能暗号を提案した．それをもとに，いくつかの改良が試みられている．2012 年に，縫田ら[7]によって，暗号文同士での加法演算ができる加法準同型暗号による検索可能暗号が提案されている．加法準同型暗号を適用することにより，計算速度の高速化を実現できるが，この手法は加減算の処理に限定され，乗除算が必要な一致判定がある秘匿検索への適用ができないという欠点がある．

よって，この問題を解決するために，暗号文同士の乗算もできる乗法準同型性を持つ完全準同型暗号による検索可能暗号が研究されている．代表的な完全準同型暗号として，Gentry によって 2009 年に提案されたイデアル格子を応用した暗号法[6]があげられる．しかし，完全準同型暗号には，計算量が非常多いことで知られている．そのため，近年では，並列処理やデータベースの分割によるマスタ・ワーカ型の分散処理などを適用し，従来の準同型暗号の処理を高速化するための研究が盛んに行われている．ただし，計算の高速化ができても，元々準同型暗号が持つ膨大な計算量が必要になるという事実はまで解決できない．

2.4 秘密分散法を用いる検索可能暗号

秘密分散法を用いた画像に対する秘匿検索は，[12]において発表した手法が世界初であると考えられる．この手法は，著者等が提案した秘密分散法を用いた秘匿計算法を適用することによって実現される．しかし，この手法は全ての秘密分散に Shamir 法を用いるため高速化処理が考慮されておらず，検索時間などに実用的な問題が残った．それに対して，辻下等による PPSS[11]は一部を除き，Shamir 法ではなく XOR によって処理が可能な XOR 法を用いているため高速処理が可能である．しかし，辻下等による PPSS はパスワードの一致を検出するための手法であり，ドキュメント検索への適用は考えられていない．

3. 提案方式１：１文字ごとの検索

3.1 提案方式１のアルゴリズム

辻下らの PPSS は，Pを登録されているパスワード，P'を検索用のパスワード，rを乱数，sを秘密情報としたとき，$r(P-P')+s$を有限体上で秘匿計算することによって実現される．すなわち，P,P'が一致したときだけ$P-P'=0$となるため，検索者は秘密情報sを取り出せ，一致しないときは乱数rによってパスワード及び秘密情報が秘匿され，情報を漏洩させない．

本提案方式では，辻下らによる PPSS を拡張し，パスワードを文字に変更し，検索処理を検索者ではなくサーバ側で行い，サーバ側で一致検証を行う．よって，秘密情報は文字列が一致したときだけサーバから提供される．

以下に提案方式１の詳細なプロトコルを示す．なお，$a_j(j=1,\dots,m)$を登録されたドキュメントの文字列，$b_i\ (i=1,\dots,g)$を検索クエリの文字列とし，文字間の差を$a_i-b_i=c_i$とする．また，通常，文字を表す文字コードでは，0 の値を「NULL 文字」を表すコードとなり，保存するテキストファイルや pdf などに用いられない場合が多い．よって，簡単のために，本提案法では，登録したドキュメントや検索クエリに「NULL 文字」を含まないとする．すなわち，入力情報に 0 を含まないとする．また，演算は法pの上で行い，【秘匿検索処理】において生成する乱数$\rho_{v,i}$は$\rho_{v,i}\in GF(p)$であり，0 を含まないとする．

【記号定義】

- a：複数の文字$a_i(i=1,\dots,m)$からなるドキュメント．ここで，$a=a_1,\dots,a_m$とする．
- $\overline{[\alpha]}_i$：サーバS_iが保持する乱数αに関する分散値．
- $[a_j]_i^{(1)}$：サーバS_iが保持するaの各文字の分散値集合．例えば，文字a_1に対する分散値集合は$[a_1]_i^{(1)}=\alpha_j a_j,\overline{[\alpha_0]}_i,\dots,\overline{[\alpha_{k-1}]}_i$となる．
- $[a]_i$：サーバS_iが保持するドキュメント$a=a_1,\dots,a_m$の分散値集合．すなわち，$[a]_i=[a_1]_i,\dots,[a_m]_i$とする．
- $[1]_i^{(\varepsilon_j^u)}$：サーバ$S_i$が保持する乱数からなる分散値集合．$\varepsilon_j^u$は任意に選択でき，1 の分散値集合と呼ぶ．例えば，

$$[1]_i^{(\varepsilon_j^u)}=\overline{[\varepsilon_j^u]}_i,\overline{[\varepsilon_{j,0}^u]}_i,\dots,\overline{[\varepsilon_{j,k-1}^u]}_i.$$

上記の 1 の分散値集合は例えば，以下のように容易に生成できる．なお，以下のアルゴリズムで生成する乱数$\varepsilon_{j,0}^u,\dots,\varepsilon_{j,k-1}^u$は$\varepsilon_{j,0}^u,\dots,\varepsilon_{j,k-1}^u\in GF(p)$であり，かつ，生成する乱数には 0 を含まないとする．また，すべての秘匿演算も同様にpを法として行われる．ここでは，下記処理 1，2 の実行は登録者によって行われる．また，サーバS_iは十分な数の 1 の分散値集合$\left([1]_i^{(\varepsilon_j^u)}=\overline{[\varepsilon_j^u]}_i,\overline{[\varepsilon_{j,l}^u]}_i\right)$を持つとする．

【1 に対する分散値の生成】

1. k個の乱数$\varepsilon_{j,0}^u,\dots,\varepsilon_{j,k-1}^u\ (j=1,\dots,m)$を生成し，乱数$\varepsilon_j^u=\prod_{l=0}^{k-1}\varepsilon_{j,l}^u$を計算する．
2. ε_j^uを(k,n)Shamir 法，$\varepsilon_{j,0}^u,\dots,\varepsilon_{j,k-1}^u$を$(k,n)$XOR 法で$n$台のサーバ$S_i\ (i=0,\dots,n-1)$に分散する．
3. サーバS_iは 1 に関する分散値集合として以下を保持する．なお，$i=0,\dots,n-1,j=1,\dots,m,l=0,\dots,k-1$とする．

$$[1]_i^{\left(\varepsilon_j^u\right)}=\left(\overline{[\varepsilon_j^u]}_i,\overline{[\varepsilon_{j,0}^u]}_i,\dots,\overline{[\varepsilon_{j,k-1}^u]}_i\right)$$

【ドキュメントの秘匿化処理】

入力：$a_j(j=1,\dots,m)$

出力：$[a]_i=[a_1]_i^{(1)},\dots,[a_m]_i^{(1)}\ (i=0,\dots,k-1)$

1. 登録者は秘密情報$a_j(j=1,\dots,m)$に対して，乱数$\alpha_{j,i}$，$(i=0,\dots,k-1)$を生成し，以下を計算する．

$$\alpha_j=\prod_{i=0}^{k-1}\alpha_{j,i}$$

2. 登録者は計算した乱数α_jを秘密情報a_jにかけて，$\alpha_j a_j$を計算して公開し，$\alpha_{j,i}$を(k,n)XOR 法でn個の分散値を生成しn台のサーバに分散する．
3. サーバS_iは秘密情報a_jに対する分散値集合として，以下を保持する．

$$[a_j]_i^{(1)}=\alpha_j a_j,\overline{[\alpha_{j,0}]}_i,\dots,\overline{[\alpha_{j,k-1}]}_i$$

【検索クエリ生成処理】

入力：$b_t(t=1,\dots,g)$

出力：$[b]_i=[b_1]_i^{(1)},\dots,[b_g]_i^{(1)}$

1. 検索者は検索したい文字列$b_t(t=1,\dots,g)$に対して，乱数$\beta_{t,i}$を生成し，以下を計算する．

$$\beta_t=\prod_{i=0}^{k-1}\beta_{t,i}$$

2. 検索者は計算した乱数β_tをb_tにかけて，$\beta_t b_t$を計算して公開し，$\beta_{t,i}$を(k,n)XOR 法でn個の分散値を生成しn台のサーバに分散する．
3. サーバS_iは検索クエリb_tに対する分散値集合として，以下を保持する．

$$[b_t]_i^{(1)}=\beta_t b_t,\overline{[\beta_{t,0}]}_i,\dots,\overline{[\beta_{t,k-1}]}_i$$

【秘匿検索処理】
入力：$[T_b]_i, [a]_i$
出力：a，または⊥

1. サーバS_iは検索クエリb_tと一致した文字列を見つけるまでに，サーバに保存されてある各文字$[a_1]_i, \dots, [a_m]_i$に対して以下の 2.~5.を行う．
2. サーバS_iは先頭位置を$h(h = 0, \dots, m-g)$とするa中の連続するg個の文字列に対して乱数$\rho_{v,i}(v = h+1, \dots, h+g)$を生成し，$\alpha_{v,i}, \beta_{v,i}$と$\varepsilon^1_{v,i}, \varepsilon^2_{v,i}$を$(k,n)$XOR 法で復元し，以下を計算して$k$台中の 1 台のサーバに送る（ここでは$S_0$とする）．
$$\frac{\rho_{v,i}}{\alpha_{v,i}\varepsilon^1_{v,i}}, \frac{\rho_{v,i}}{\beta_{v,i}\varepsilon^2_{v,i}},$$
3. サーバS_0は以下を計算し，k台のサーバS_iに送信する．
$$\frac{\rho_v}{\alpha_v\varepsilon^1_v} = \prod_{i=0}^{k-1} \frac{\rho_{v,i}}{\alpha_{v,i}\varepsilon^1_{v,i}}, \frac{\rho_v}{\beta_v\varepsilon^2_v} = \prod_{i=0}^{k-1} \frac{\rho_{v,i}}{\beta_{v,i}\varepsilon^2_{v,i}}$$
4. サーバS_iは以下を計算し，サーバS_0に送る．
$$[\rho_v(a_v - b_v)]_i = \alpha_v a_v \times \frac{\rho_v}{\alpha_v\varepsilon^1_v} \times \overline{[\varepsilon^1_v]_i} - \beta_v b_v \times \frac{\rho_v}{\beta_v\varepsilon^2_v} \times \overline{[\varepsilon^2_v]_i}$$
5. サーバS_0は$\rho_v(a_v - b_v)$を(k,n)Shamir 法で復元し，全て 0 であれば一致とする．
6. 一致しなければ，サーバS_iは$v = v+1$として 2.~5.を繰り返す．
7. サーバS_iは検索クエリ$b_1, \dots, b_g$と$a_v, \dots, a_{v+g-1}$が一致すれば，それを含むドキュメントaの分散値集合$[a]_i$を検索者に送信する．一致したドキュメントがなければ処理を停止する．
8. 検索者はサーバから得られたk個の分散値集合より，ドキュメントaを復元する．

3.2 提案方式 1 の問題点

提案方式 1 は，パスワードを文字に対応させ，その一致検索を行ったものであり，その安全性は[11]に依存する．しかし，パスワードを文字列の検索に変更した場合，パスワードの場合に生じなかった以下の問題が発生する．

例えば，各文字を ASCII コードで表すとすると，文字の値の候補は 0～127 の間であり，攻撃者は$k-1$台までのサーバの処理をのぞき見できるため，毎回異なる値の文字をもつ 1 文字を含んで検索処理を実行すれば，最大 128 回の検索でドキュメント中の全文字の位置が漏洩する．すなわち，128 回の検索で全数探索が可能であるという問題がある．それに対して，パスワード検索では全数探索に耐える長いパスワードを使うことを前提としているため，この問題は発生しない．

ドキュメント検索においても，例えば 18 文字列単位で検索を行う等という規定をかければ$(2^7)^{18} = 2^{126}$となり，2^{126}回検索をしなければ全数探索できないためこの問題に対応できるが，1 文字単位でずらしながら検索を行うためには少なくとも 1 文字当り$18 \times n$種類の分散値を保存する必要がある．また，検索対象を 18 文字列単位ではなく，20 文字列や 25 文字列単位で検索したい場合には対応できない．

よって，ドキュメントの秘匿検索においては複数文字列を文字単位で同時に検索できる必要がある．すなわち，1 文字単位で検索しながら，その検索結果は 1 文字単位ではなく複数文字で一括してわかるようにする必要がある．これは PPSS では複数パスワードの完全一致検証の実現を意味する．すなわち，安全性向上のために複数パスワードの一致を検証する場合，その結果がパスワード単位でわかれば攻撃者は正しいパスワードと間違ったパスワードを識別できる．それに対してパスワード毎に検証するが，全て一致しなければ検証成功とならない場合，攻撃の回数を著しく増加させ，安全性の向上を実現する．しかし，今まで秘密分散法による複数パスワードの一致を同時に検証できる PPSS は提案されていない．

よって，提案方式 2 では複数文字の完全一致検証を実現する．これが実現できれば，前述の複数パスワードの完全一致検証も実現できる．

4. 提案方式 2：AND 検索

4.1 提案方式 2 のアルゴリズム

提案方式 2 では，1 文字単位で検索しながら，その検索結果が複数文字で一括してわかる複数文字の完全一致検証を実現する．$a_j(j = 1, \dots, m)$を登録されたドキュメントの文字列，$b_i\ (i = 1, \dots, g)$を検索クエリの文字列とし，文字間の差を$a_i - b_i = c_i$とする．このとき，$c_1, \dots, c_g$が全て0でないときのみ，$f(c_1, \dots, c_g) \neq 0$となる演算$f$は AND 演算であり，$c_1 \dots c_g$となる．一方，AND の負論理では，$c_1, \dots, c_g$が全て0のときのみ，$f(c_1, \dots, c_g) = 0$となり，演算$f$は，$\overline{c_1} \dots \overline{c_g}$となる．ド・モルガンの定理によると，これは，$\overline{\overline{c_1} \dots \overline{c_g}} = c_1 + \dots + c_g$となる．しかし，負の値がないようにするため，複数文字列の完全一致検証を行う際に，以下を計算すればよい．

$$\sum (a_i - b_i)^2 = \sum c_i$$

ここで，$a_j, b_i < q$とするときの法pを$gq^2 < p$とし，0 を含まないとする．また，登録されたドキュメントの文字列a_jと検索クエリb_iは$a_j, b_i \in GF(p)$であり，【ドキュメントの秘匿化処理】，【検索クエリ生成処理】及び【秘匿検索処理】で生成する乱数$\alpha_{j,i}$, $\gamma_{j,i}$, c_j, $\beta_{j,i}$, $\delta_{t,i}$, d_t, $\rho_{v,i}$, $\varphi_{v,i}$, $\kappa_{v,i}$は$\alpha_{j,i}$, $\gamma_{j,i}$, c_j, $\beta_{j,i}$, $\delta_{t,i}$, d_t, $\rho_{v,i}$, $\varphi_{v,i}$, $\kappa_{v,i} \in GF(p)$である（ただし，生成する乱数は 0 を除く）．

以下に提案したアルゴリズムを示す．なお，以下のアルゴリズムに示す秘密分散の処理も含めてすべての秘匿演算はpを法として行われる．また，提案方式 2 においてc_v, d_vは$c_v + d_v = 0$とならない乱数であり，【秘匿検索処理】5.において復元したとき 0 であれば，登録者と検索者は【ドキュメントの秘匿化処理】と【検索クエリ生成処理】におけ

る 1 ～3.を$c_v + d_v \neq 0$となるまで行う．提案方式 2 の【秘匿検索処理】では【検索クエリ生成処理】で入力された任意の文字数gに応じた完全一致検索が実現できる．なお，【秘匿検索処理】においてサーバS_iは同じでi規定される乱数（$\gamma_{v,i}$や$\varepsilon_{v,i}^1$など）を扱うとする．また，サーバS_iは十分な数の 1 の分散値集合$\left([1]_i^{(\varepsilon_j^u)} = \overline{[\varepsilon_j^u]}_i, \overline{[\varepsilon_{j,l}^u]}_i\right)$を持つとする．

【ドキュメントの秘匿化処理】

入力：$a_j (j = 1, \dots, m)$

出力：$[a]_i = [a_1]_i, \dots, [a_m]_i \ (i = 0, \dots, k-1)$

1. 登録者は秘密情報$a_j (j = 1, \dots, m)$に対して，乱数$\alpha_{j,i}, \gamma_{j,i}, c_j \ (i = 0, \dots, k-1)$を生成し，以下を計算する．

$$\alpha_j = \prod_{i=0}^{k-1} \alpha_{j,i}, \gamma_j = \prod_{i=0}^{k-1} \gamma_{j,i}$$

2. 登録者は計算した乱数α_j, γ_jをそれぞれ秘密情報a_jと乱数c_jにかけて，$\alpha_j a_j, \gamma_j c_j$を計算して公開し，$\alpha_{j,i}, \gamma_{j,i}$を$(k, n)$XOR 法で$n$個の分散値を生成し$n$台のサーバに分散する．

3. サーバS_iは秘密情報a_j及び乱数c_jに対する分散値集合として，以下を保持する．

$$[a_j]_i^{(1)} = \alpha_j a_j, \overline{[\alpha_{j,0}]}_i, \dots, \overline{[\alpha_{j,k-1}]}_i$$

$$[c_j]_i^{(1)} = \gamma_j c_j, \overline{[\gamma_{j,0}]}_i, \dots, \overline{[\gamma_{j,k-1}]}_i$$

4. $[a_j]_i \coloneqq \left([a_j]_i^{(1)}, [c_j]_i^{(1)}\right)$を秘密情報$a_j$の分散情報と定義する．よって，サーバ$S_i$は以下の情報を保持する．

$$[a]_i = [a_1]_i, \dots, [a_m]_i$$

【検索クエリ生成処理】

入力：$b_t (t = 1, \dots, g)$

出力：$[T_b]_i = \left[T_{b_1}\right]_i, \dots, \left[T_{b_g}\right]_i$

1. 検索者は検索したい文字列$b_t (t = 1, \dots, g)$に対して，乱数$\beta_{t,i}, \delta_{t,i}, d_t$を生成し，以下を計算する．

$$\beta_t = \prod_{i=0}^{k-1} \beta_{t,i}, \delta_t = \prod_{i=0}^{k-1} \delta_{t,i}$$

2. 検索者は計算した乱数β_t, δ_tをそれぞれb_tと乱数d_tにかけて，$\beta_t b_t, \delta_t d_t$を計算して公開し，$\beta_{t,i}, \delta_{t,i}$を$(k, n)$XOR 法で$n$個の分散値を生成し$n$台のサーバに分散する．

3. サーバS_iは検索クエリb_t及び乱数d_tに対する分散値集合として，以下を保持する．

$$[b_t]_i^{(1)} = \beta_t b_t, \overline{[\beta_{t,0}]}_i, \dots, \overline{[\beta_{t,k-1}]}_i$$

$$[d_t]_i^{(1)} = \delta_t d_t, \overline{[\delta_{t,0}]}_i, \dots, \overline{[\delta_{t,k-1}]}_i$$

4. $\left[T_{b_t}\right]_i \coloneqq \left([b_t]_i^{(1)}, [d_t]_i^{(1)}\right)$を検索クエリ$b_t$の分散情報と定義する．よって，サーバ$S_i$は以下の情報を保持する．

$$[T_b]_i = \left[T_{b_1}\right]_i, \dots, \left[T_{b_g}\right]_i$$

【秘匿検索処理】

入力：$[T_b]_i, [a]_i$

出力：a，または$\perp$

1. サーバS_iは検索クエリb_tと一致した文字列を見つけるまでに，サーバに保存されてある各文字$[a_1]_i, \dots, [a_m]_i$に対して以下の 2.~11.を行う．

2. サーバS_iは先頭位置を$h (h = 0, \dots, m-g)$とするa中の連続するg個の文字列に対して乱数$\rho_{v,i}, \varphi_{v,i} (v = h+1, \dots, h+g)$を生成し，$\gamma_{v,i}, \delta_{v,i}, \alpha_{v,i}, \beta_{v,i}$と$\varepsilon_{v,i}^1, \varepsilon_{v,i}^2, \varepsilon_{v,i}^3, \varepsilon_{v,i}^4, \varepsilon_{v,i}^5$を$(k, n)$XOR 法で復元し，以下を計算して$k$台中の 1 台のサーバに送る（ここでは$S_0$とする）．

$$\frac{\rho_{v,i}}{\gamma_{v,i}\varepsilon_{v,i}^1}, \frac{\rho_{v,i}}{\delta_{v,i}\varepsilon_{v,i}^2}, \frac{\varphi_{v,i}}{\alpha_{v,i}\varepsilon_{v,i}^3}, \frac{\varphi_{v,i}}{\beta_{v,i}\varepsilon_{v,i}^4}, \frac{\varphi_{v,i}}{\rho_{v,i}\varepsilon_{v,i}^5}$$

3. サーバS_0は以下を計算し，k台のサーバS_iに送信する．

$$\frac{\rho_v}{\gamma_v\varepsilon_v^1} = \prod_{i=0}^{k-1} \frac{\rho_{v,i}}{\gamma_{v,i}\varepsilon_{v,i}^1}, \frac{\rho_v}{\delta_v\varepsilon_v^2} = \prod_{i=0}^{k-1} \frac{\rho_{v,i}}{\delta_{v,i}\varepsilon_{v,i}^2}$$

$$\frac{\varphi_v}{\alpha_v\varepsilon_v^3} = \prod_{i=0}^{k-1} \frac{\varphi_{v,i}}{\alpha_{v,i}\varepsilon_{v,i}^3}, \frac{\varphi_v}{\beta_v\varepsilon_v^4} = \prod_{i=0}^{k-1} \frac{\varphi_{v,i}}{\beta_{v,i}\varepsilon_{v,i}^4}$$

$$\frac{\varphi_v}{\rho_v\varepsilon_v^5} = \prod_{i=0}^{k-1} \frac{\varphi_{v,i}}{\rho_{v,i}\varepsilon_{v,i}^5}$$

4. サーバS_iは以下を計算し，サーバS_0に送る．

$$[\rho_v(c_v + d_v)]_i = \gamma_v c_v \times \frac{\rho_v}{\gamma_v\varepsilon_v^1} \times \overline{[\varepsilon_v^1]}_i + \delta_v d_v \times \frac{\rho_v}{\delta_v\varepsilon_v^2} \times \overline{[\varepsilon_v^2]}_i$$

5. サーバS_0は$\rho_v(c_v + d_v)$を(k, n)Shamir 法で復元し，サーバS_iに送る．

6. サーバS_iは以下を計算し，サーバS_0に送る．

$$\begin{aligned}[\varphi_v(a_v - b_v + c_v + d_v)]_i &= \alpha_v a_v \times \frac{\varphi_v}{\alpha_v\varepsilon_v^3} \times \overline{[\varepsilon_v^3]}_i \\ &- \beta_v b_v \times \frac{\varphi_v}{\beta_v\varepsilon_v^4} \times \overline{[\varepsilon_v^4]}_i \\ &+ \rho_v(c_v + d_v) \times \frac{\varphi_v}{\rho_v\varepsilon_v^5} \times \overline{[\varepsilon_v^5]}_i\end{aligned}$$

7. サーバS_0は$\varphi_v(a_v - b_v + c_v + d_v)$を$(k, n)$Shamir 法で復元し，サーバ$S_i$に送る．

8. サーバS_iは乱数$\kappa_{v,i}$を生成し，$\varphi_{v,i}, \rho_{v,i}, \varepsilon_{v,i}^6, \varepsilon_{v,i}^7, \varepsilon_{v,i}^8$を$(k, n)$XOR 法で復元し，以下を計算し，サーバ$S_0$に送る．

$$\frac{\kappa_{v,i}}{(\varphi_{v,i})^2\varepsilon_{v,i}^6}, \frac{\kappa_{v,i}}{\varphi_{v,i}\rho_{v,i}\varepsilon_{v,i}^7}, \frac{\kappa_{v,i}}{(\rho_{v,i})^2\varepsilon_{v,i}^8}$$

9. サーバS_0は以下を計算し，サーバS_iに送信する．

$$\frac{\kappa_v}{(\varphi_v)^2\varepsilon_v^6} = \prod_{i=0}^{k-1} \frac{\kappa_{v,i}}{(\varphi_{v,i})^2\varepsilon_{v,i}^6}$$

$$\frac{\kappa_v}{\varphi_v\rho_v\varepsilon_v^7} = \prod_{i=0}^{k-1} \frac{\kappa_{v,i}}{\varphi_{v,i}\rho_{v,i}\varepsilon_{v,i}^7}$$

$$\frac{\kappa_v}{(\rho_v)^2\varepsilon_v^8} = \prod_{i=0}^{k-1} \frac{\kappa_{v,i}}{(\rho_{v,i})^2\varepsilon_{v,i}^8}$$

10. サーバS_iは以下を計算する．

$$\sum_{v=h}^{h+g}\overline{[\kappa_v(a_v-b_v)^2]}_i=\sum_{v=h}^{h+g}\Bigg\{\Bigg[(\varphi_v)^2(a_v-b_v+c_v+d_v)^2 \times\frac{\kappa_v}{(\varphi_v)^2\varepsilon_v^6}\times\overline{[\varepsilon_v^6]}_i\Bigg] -\Bigg[2\times\varphi_v(a_v-b_v+c_v+d_v) \times\rho_v(c_v+d_v)\times\frac{\kappa_v}{\varphi_v\rho_v\varepsilon_v^7}\times\overline{[\varepsilon_v^7]}_i\Bigg] +\Bigg[(\rho_v)^2(c_v+d_v)^2\times\frac{\kappa_v}{(\rho_v)^2\varepsilon_v^8}\times\overline{[\varepsilon_v^8]}_i\Bigg]\Bigg\}$$

11. サーバS_iは協力して$\sum_{v=h}^{h+g}\kappa_v(a_v-b_v)^2$を$(k,n)$Shamir 法で復元し，0 であれば一致とする．
12. 一致しなければ，サーバS_iは$v=v+1$として 2. ~11.を繰り返す．
13. サーバS_iは検索クエリ$b_1,\dots,b_g$と$a_v,\dots,a_{v+g-1}$が一致すれば，それを含むドキュメントaの分散値集合$[a]_i$からc_jに関する分散値集合を除いたものを検索者に送信する．一致したドキュメントがなければ処理を停止する．
14. 検索者はサーバから得られたk個の分散値集合より，ドキュメントaを復元する．

4.2 提案方式 2 の安全性

攻撃者は登録者の情報を知らず，$t\ (t<k)$台のサーバと結託した検索者とし，攻撃者が登録者の保存したドキュメントまたはキーワードを識別できた時, 攻撃成功とみなす．なお, 攻撃者はプロトコルに従う passive adversary とする．

提案方式 2 においてc_v,d_vは$c_v+d_v=0$とならない乱数であり，$\varphi_v(a_v-b_v+c_v+d_v)$を復元したときに 0 となっても，$a_v-b_v$が 0 でない$-(c_v+d_v)$と等しいことは分かるが，$c_v+d_v$が$\rho_v$で秘匿されているため，1 文字単位の差分である$a_v-b_v$はわからない．よって，復元された$\rho_v(c_v+d_v)$と$\varphi_v(a_v-b_v+c_v+d_v)$から文字単位での情報は漏洩しない．

また，提案方式 2 により，最終的な一致検証は【秘匿検索処理】の 11.において行われるが，$\sum_{v=h}^{h+g}\kappa_v(a_v-b_v)^2$は$g$個の文字列の差分の和であり，法$p$は$gq^2<p$であるため，$\sum_{v=h}^{h+g}\kappa_v(a_v-b_v)^2=0$であれば全ての$a_v-b_v=0$であることが言える．これは, 検索対象となる$g$文字列全ての同時検証，すなわち完全一致検証ができていることを意味する．

また，全数探索に対しては全文字が一致しなければ$\sum_{v=h}^{h+g}\kappa_v(a_v-b_v)^2=0$とならないため，$g=18$の場合，全数探索を行うには$2^{126}$回の探索が必要であり，十分安全であると言える．

5. 提案方式 3：AND・OR の組合せの検索

5.1 提案方式 3 のアルゴリズム

ここではより一般的に利用される場合を想定し, AND 検索と OR 検索を組合せた関数型秘匿検索を示す．例えば，簡単のため AND 検索の文字数をgとするx種類の異なる検索クエリで OR 検索を行う場合，$a_j(j=1,\dots,m)$を登録されたドキュメントの文字列，$b_{l,i}\ (l=1,\dots,x,i=1,\dots,g)$をそれぞれ$x$個の検索クエリの文字列とする．ここで，1 つの検索クエリに対する完全一致検索を行うには 4 章に示したように，$\sum(a_i-b_{l,i})^2=\sum c_{l,i}\ (l=1,..,x)$を秘匿計算すればよい．

よって, x個の検索クエリに対してそれぞれAND検索を行い，x個の検索クエリのうち 1 つでも一致していれば0を出力するようにする，すなわち OR 検索を実現する．$c_1,\dots,c_g$のうち 1 つでも0でないときのみ，$f(c_1,\dots,c_g)\neq 0$となる演算fは OR 演算であり，$c_1+\cdots+c_g$となる．よって，OR の負論理では, $c_1,\dots,c_g$のうち, 1 つでも0のとき, $f(c_1,\dots,c_g)=0$となり，演算fは，$\overline{c_1+\cdots+c_g}$となる．ここでは，ド・モルガンの定理により，$\overline{c_1+\cdots+c_g}=c_1\dots c_g$であるので，複数検索クエリに対する OR 検索には，以下を計算する．

$$\prod\left(\sum(a_i-b_{l,i})^2\right)=\prod\left(\sum c_{l,i}\right)$$

提案方式 3 において，法pは$a_i,b_i<q$であればgq^2以上の素数とする．以下に提案方式 3 の詳細なアルゴリズムを示すが,【ドキュメントの秘匿化処理】と【検索クエリ生成処理】は提案方式 2 と同様であるので省略し， AND・OR の【秘匿検索処理】のみを示す．なお，簡単のために，以下に検索クエリの数を$x=2$とした場合について示す（検索クエリ 1 を$b_{1,t}(t=1,\dots,g)$, 検索クエリ 2 を$b_{2,t}(t=1,\dots,g)$とする）．よって，下記秘匿検索処理における 1.~4.は一つ目の検索クエリ$b_{1,t}$に対する AND 検索となっており，5.~8.は 2 つ目の検索クエリ$b_{2,t}$に対する AND 検索となっている．9.以降が 2 つの検索クエリ$b_{1,t},b_{2,t}$に対する OR 検索となる．ただし，$\sum_{v=h}^{h+g}\left(\kappa_{1,v}\left((a_v-b_{1,v})^2+c_{1,v}+d_{1,v}\right)\right)$または$\sum_{v=h}^{h+g}\left(\kappa_{2,v}\left((a_v-b_{2,v})^2+c_{2,v}+d_{2,v}\right)\right)$の復元結果が 0 になる場合は, 乱数$c_{1,v},d_{1,v}$または$c_{2,v},d_{2,v}$を変えてやり直す. また，提案方式 3 においてもサーバS_iは同じでi規定される乱数を扱うとする．

なお，以下に示す【秘匿検索処理】において生成する乱数は$\kappa_{1,v,i}$, $\kappa_{2,v,i}$, $\phi_i\in GF(p)$であり， 0 を含まないとする．また，以下に示すアルゴリズムは，秘密分散の処理も含めてすべての秘匿演算はpを法として行われる．

【秘匿検索処理】

入力：$[T_{b1}]_i,[T_{b2}]_i,[a]_i$

出力：a，または⊥

1. サーバS_iは検索クエリ$b_{1,t}(t=1,\dots,g)$に対して AND 検索の 1.～7.を実行し，乱数$\kappa_{1,v,i}$を生成し，$\varphi_{1,v,i}$, $\rho_{1,v,i}$, $\varepsilon_{1,v,i}^6$, $\varepsilon_{1,v,i}^7$, $\varepsilon_{1,v,i}^8$, $\varepsilon_{1,v,i}^9$を(k,n)XOR 法で復元し，以下を計算し，サーバS_0に送る．

$$\frac{\kappa_{1,v,i}}{(\varphi_{1,v,i})^2\varepsilon_{1,v,i}^6},\frac{\kappa_{1,v,i}}{\varphi_{1,v,i}\rho_{1,v,i}\varepsilon_{1,v,i}^7},\frac{\kappa_{1,v,i}}{(\rho_{1,v,i})^2\varepsilon_{1,v,i}^8},\frac{\kappa_{1,v,i}}{\rho_{1,v,i}\varepsilon_{1,v,i}^9}$$

2. サーバS_0は以下を計算し，サーバS_iに送信する．

$$\frac{\kappa_{1,v}}{(\varphi_{1,v})^2\varepsilon_{1,v}^6}=\prod_{i=0}^{k-1}\frac{\kappa_{1,v,i}}{(\varphi_{1,v,i})^2\varepsilon_{1,v,i}^6},\frac{\kappa_{1,v}}{\varphi_{1,v}\rho_{1,v}\varepsilon_{1,v}^7}=\prod_{i=0}^{k-1}\frac{\kappa_{1,v,i}}{\varphi_{1,v,i}\rho_{1,v,i}\varepsilon_{1,v,i}^7}$$

$$\frac{\kappa_{1,v}}{(\rho_{1,v})^2\varepsilon_{1,v}^8}=\prod_{i=0}^{k-1}\frac{\kappa_{1,v,i}}{(\rho_{1,v,i})^2\varepsilon_{1,v,i}^8},\frac{\kappa_{1,v}}{\rho_{1,v}\varepsilon_{1,v}^9}=\prod_{i=0}^{k-1}\frac{\kappa_{1,v,i}}{\rho_{1,v,i}\varepsilon_{1,v,i}^9}$$

3. サーバS_iは以下を計算し，サーバS_0に送る．

$$\sum_{v=h}^{h+g}\overline{\left[\kappa_{1,v}\left(\left(a_v-b_{1,v}\right)^2+c_{1,v}+d_{1,v}\right)\right]_i}$$

$$=\sum_{v=h}^{h+g}\left\{\left[\left(\varphi_{1,v}\right)^2\left(a_v-b_{1,v}+c_{1,v}+d_{1,v}\right)^2\times\frac{\kappa_{1,v}}{\left(\varphi_{1,v}\right)^2\varepsilon_{1,v}^6}\times\overline{\left[\varepsilon_{1,v}^6\right]_i}\right]\right.$$
$$-\left[2\times\varphi_{1,v}\left(a_v-b_{1,v}+c_{1,v}+d_{1,v}\right)\right.$$
$$\left.\times\rho_{1,v}\left(c_{1,v}+d_{1,v}\right)\times\frac{\kappa_{1,v}}{\varphi_{1,v}\rho_{1,v}\varepsilon_{1,v}^7}\times\overline{\left[\varepsilon_{1,v}^7\right]_i}\right]$$
$$+\left[\left(\rho_{1,v}\right)^2\left(c_{1,v}+d_{1,v}\right)^2\times\frac{\kappa_{1,v}}{\left(\rho_{1,v}\right)^2\varepsilon_{1,v}^8}\times\overline{\left[\varepsilon_{1,v}^8\right]_i}\right]$$
$$\left.+\left[\rho_{1,v}\left(c_{1,v}+d_{1,v}\right)\times\frac{\kappa_{1,v}}{\rho_{1,v}\varepsilon_{1,v}^9}\times\overline{\left[\varepsilon_{1,v}^9\right]_i}\right]\right\}$$

4. サーバS_0は$\sum_{v=h}^{h+g}\left(\kappa_{1,v}\left(\left(a_v-b_{1,v}\right)^2+c_{1,v}+d_{1,v}\right)\right)$を$(k,n)$Shamir 法で復元し，サーバ$S_i$に送る．

5. サーバS_iは検索クエリ$b_{2,t}(t=1,\dots,g)$に対して AND 検索の 1.～7.を実行し，乱数$\kappa_{2,v,i}$を生成し，$\varphi_{2,v,i}$，$\rho_{2,v,i}$，$\varepsilon_{2,v,i}^6$，$\varepsilon_{2,v,i}^7$，$\varepsilon_{2,v,i}^8$，$\varepsilon_{2,v,i}^9$を(k,n)XOR 法で復元し，以下を計算し，サーバS_0に送る．

$$\frac{\kappa_{2,v,i}}{\left(\varphi_{2,v,i}\right)^2\varepsilon_{2,v,i}^6},\frac{\kappa_{2,v,i}}{\varphi_{2,v,i}\rho_{2,v,i}\varepsilon_{2,v,i}^7},\frac{\kappa_{2,v,i}}{\left(\rho_{2,v,i}\right)^2\varepsilon_{2,v,i}^8},\frac{\kappa_{2,v,i}}{\rho_{2,v,i}\varepsilon_{2,v,i}^9}$$

6. サーバS_0は以下を計算し，サーバS_iに送信する．

$$\frac{\kappa_{2,v}}{\left(\varphi_{2,v}\right)^2\varepsilon_{2,v}^6}=\prod_{i=0}^{k-1}\frac{\kappa_{2,v,i}}{\left(\varphi_{2,v,i}\right)^2\varepsilon_{2,v,i}^6}$$
$$\frac{\kappa_{2,v}}{\varphi_{2,v}\rho_{2,v}\varepsilon_{2,v}^7}=\prod_{i=0}^{k-1}\frac{\kappa_{2,v,i}}{\varphi_{2,v,i}\rho_{2,v,i}\varepsilon_{2,v,i}^7}$$
$$\frac{\kappa_{2,v}}{\left(\rho_{2,v}\right)^2\varepsilon_{2,v}^8}=\prod_{i=0}^{k-1}\frac{\kappa_{2,v,i}}{\left(\rho_{2,v,i}\right)^2\varepsilon_{2,v,i}^8}$$
$$\frac{\kappa_{2,v}}{\rho_{2,v}\varepsilon_{2,v}^9}=\prod_{i=0}^{k-1}\frac{\kappa_{2,v,i}}{\rho_{2,v,i}\varepsilon_{2,v,i}^9}$$

7. サーバS_iは以下を計算し，サーバS_0に送信する．

$$\sum_{v=h}^{h+g}\overline{\left[\kappa_{2,v}\left(\left(a_v-b_{2,v}\right)^2+c_{2,v}+d_{2,v}\right)\right]_i}$$

$$=\sum_{v=h}^{h+g}\left\{\left[\left(\varphi_{2,v}\right)^2\left(a_v-b_{2,v}+c_{2,v}+d_{2,v}\right)^2\times\frac{\kappa_{2,v}}{\left(\varphi_{2,v}\right)^2\varepsilon_{2,v}^6}\times\overline{\left[\varepsilon_{2,v}^6\right]_i}\right]\right.$$
$$-\left[2\times\varphi_{2,v}\left(a_v-b_{2,v}+c_{2,v}+d_{2,v}\right)\right.$$
$$\left.\times\rho_{2,v}\left(c_{2,v}+d_{2,v}\right)\times\frac{\kappa_{2,v}}{\varphi_{2,v}\rho_{2,v}\varepsilon_{2,v}^7}\times\overline{\left[\varepsilon_{2,v}^7\right]_i}\right]$$
$$+\left[\left(\rho_{2,v}\right)^2\left(c_{2,v}+d_{2,v}\right)^2\times\frac{\kappa_{2,v}}{\left(\rho_{2,v}\right)^2\varepsilon_{2,v}^8}\right.$$
$$\left.\times\overline{\left[\varepsilon_{2,v}^8\right]_i}\right]$$
$$\left.+\left[\rho_{2,v}\left(c_{2,v}+d_{2,v}\right)\times\frac{\kappa_{2,v}}{\rho_{2,v}\varepsilon_{2,v}^9}\times\overline{\left[\varepsilon_{2,v}^9\right]_i}\right]\right\}$$

8. サーバS_0は$\sum_{v=h}^{h+g}\left(\kappa_{2,v}\left(\left(a_v-b_{2,v}\right)^2+c_{2,v}+d_{2,v}\right)\right)$を$(k,n)$Shamir 法で復元し，サーバ$S_i$に送る．

9. サーバS_iは乱数ϕ_iを生成し，$\kappa_{1,v,i}$，$\kappa_{2,v,i}$，$\rho_{1,v,i}$，$\rho_{2,v,i}$，ε_i^{10}，ε_i^{11}，ε_i^{12}，ε_i^{13}を(k,n)XOR 法で復元し，以下を計算し，サーバS_0に送る．

$$\frac{\phi_i}{\sum_{v=h}^{h+g}\left(\kappa_{1,v,i}\right)\sum_{v=h}^{h+g}\left(\kappa_{2,v,i}\right)\varepsilon_i^{10}}$$
$$\frac{\phi_i}{\sum_{v=h}^{h+g}\left(\rho_{1,v,i}\right)\sum_{v=h}^{h+g}\left(\kappa_{2,v,i}\right)\varepsilon_i^{11}}$$
$$\frac{\phi_i}{\sum_{v=h}^{h+g}\left(\rho_{2,v,i}\right)\sum_{v=h}^{h+g}\left(\kappa_{1,v,i}\right)\varepsilon_i^{12}}$$
$$\frac{\phi_i}{\sum_{v=h}^{h+g}\left(\rho_{1,v,i}\right)\sum_{v=h}^{h+g}\left(\rho_{2,v,i}\right)\varepsilon_i^{13}}$$

10. サーバS_0は以下を計算し，サーバS_iに送信する

$$\frac{\phi}{\sum_{v=h}^{h+g}\left(\kappa_{1,v}\right)\sum_{v=h}^{h+g}\left(\kappa_{2,v}\right)\varepsilon^{10}}=\prod_{i=0}^{k-1}\frac{\phi_i}{\sum_{v=h}^{h+g}\left(\kappa_{1,v,i}\right)\sum_{v=h}^{h+g}\left(\kappa_{2,v,i}\right)\varepsilon_i^{10}}$$
$$\frac{\phi}{\sum_{v=h}^{h+g}\left(\rho_{1,v}\right)\sum_{v=h}^{h+g}\left(\kappa_{2,v}\right)\varepsilon^{11}}=\prod_{i=0}^{k-1}\frac{\phi_i}{\sum_{v=h}^{h+g}\left(\rho_{1,v,i}\right)\sum_{v=h}^{h+g}\left(\kappa_{2,v,i}\right)\varepsilon_i^{11}}$$
$$\frac{\phi}{\sum_{v=h}^{h+g}\left(\rho_{2,v}\right)\sum_{v=h}^{h+g}\left(\kappa_{1,v}\right)\varepsilon^{12}}=\prod_{i=0}^{k-1}\frac{\phi_i}{\sum_{v=h}^{h+g}\left(\rho_{2,v,i}\right)\sum_{v=h}^{h+g}\left(\kappa_{1,v,i}\right)\varepsilon_i^{12}}$$
$$\frac{\phi}{\sum_{v=h}^{h+g}\left(\rho_{1,v}\right)\sum_{v=h}^{h+g}\left(\rho_{2,v}\right)\varepsilon^{13}}=\prod_{i=0}^{k-1}\frac{\phi_i}{\sum_{v=h}^{h+g}\left(\rho_{1,v,i}\right)\sum_{v=h}^{h+g}\left(\rho_{2,v,i}\right)\varepsilon_i^{13}}$$

11. サーバS_iは以下を計算する．

$$\prod_{l=1}^{2}\overline{\left[\phi\left(\sum_{v=h}^{h+g}\left(a_v-b_{l,v}\right)^2\right)\right]_i}$$

$$=\left[\left\{\sum_{v=h}^{h+g}\kappa_{1,v}\left(\left(a_v-b_{1,v}\right)^2+c_{1,v}\right.\right.\right.$$
$$\left.\left.+d_{1,v}\right)\sum_{v=h}^{h+g}\kappa_{2,v}\left(\left(a_v-b_{2,v}\right)^2+c_{2,v}\right.\right.$$
$$\left.\left.+d_{2,v}\right)\right\}\times\frac{\phi}{\sum_{v=h}^{h+g}\left(\kappa_{1,v}\right)\sum_{v=h}^{h+g}\left(\kappa_{2,v}\right)\varepsilon^{10}}$$
$$\left.\times\overline{\left[\varepsilon^{10}\right]_i}\right]$$
$$-\left[\sum_{v=h}^{h+g}\rho_{1,v}\left(c_{1,v}+d_{1,v}\right)\sum_{v=h}^{h+g}\kappa_{2,v}\left(\left(a_v-b_{2,v}\right)^2+c_{2,v}+d_{2,v}\right)\right.$$
$$\left.\times\frac{\phi}{\sum_{v=h}^{h+g}\left(\rho_{1,v}\right)\sum_{v=h}^{h+g}\left(\kappa_{2,v}\right)\varepsilon^{11}}\times\overline{\left[\varepsilon^{11}\right]_i}\right]$$
$$-\left[\sum_{v=h}^{h+g}\rho_{2,v}\left(c_{2,v}+d_{2,v}\right)\sum_{v=h}^{h+g}\kappa_{1,v}\left(\left(a_v-b_{1,v}\right)^2+c_{1,v}+d_{1,v}\right)\right.$$
$$\left.\times\frac{\phi}{\sum_{v=h}^{h+g}\left(\rho_{2,v}\right)\sum_{v=h}^{h+g}\left(\kappa_{1,v}\right)\varepsilon^{12}}\times\overline{\left[\varepsilon^{12}\right]_i}\right]$$
$$+\left[\sum_{v=h}^{h+g}\rho_{1,v}\left(c_{1,v}+d_{1,v}\right)\sum_{v=h}^{h+g}\rho_{2,v}\left(c_{2,v}+d_{2,v}\right)\right.$$
$$\left.\times\frac{\phi}{\sum_{v=h}^{h+g}\left(\rho_{1,v}\right)\sum_{v=h}^{h+g}\left(\rho_{2,v}\right)\varepsilon^{13}}\times\overline{\left[\varepsilon^{13}\right]_i}\right]$$

12. サーバS_iは協力して$\prod_{l=1}^{2}\phi\left(\sum_{v=h}^{h+g}\left(a_v-b_{l,v}\right)^2\right)$を$(k,n)$Shamir 法で復元し，0 であれば一致とする．

13. 一致しなければ，サーバS_iは$v=v+1$として 1. ~12.を繰り返す．

14. 検索クエリ$b_{1,i}$または$b_{2,i}$と一致した$a_v,\dots,a_{v+g-1}$の登録された秘密情報を見つければ，一致した秘密情報を含む分散値集合$[a]_i$を検索者に送信する．一致した秘密情報がなければ処理を停止する．

15. 検索者はサーバから得られたk個の分散値集合より，元の秘密情報を復元する．

5.2 提案方式 3 の安全性

提案方式 3 により，最終的な一致検証は【秘匿検索処理】の 12.において行われるが，$\prod_{l=1}^{2}\phi\left(\sum_{v=h}^{h+g}\left(a_v-b_{l,v}\right)^2\right)$は$x$個の

検索クエリと登録された文字列の差分の和を掛け合わせた結果であり，$\prod_{l=1}^{2}\phi\left(\sum_{v=h}^{h+g}(a_v-b_{l,v})^2\right)=0$であれば入力した$x$個の検索クエリのうち、1 つ以上が一致していることを意味する．また，提案方式 3 の【秘匿検索処理】の 1.~8.は 1 つ目と 2 つ目の検索クエリに対する AND 検索となっており，その安全性は 4.2 と同じであると言える．よって，攻撃者は 4.及び 8.により，$\sum_{v=h}^{h+g}\left(\kappa_{1,v}\left((a_v-b_{1,v})^2+c_{1,v}+d_{1,v}\right)\right)$及び$\sum_{v=h}^{h+g}\left(\kappa_{2,v}\left((a_v-b_{2,v})^2+c_{2,v}+d_{2,v}\right)\right)$を知るが，それらの情報より秘密情報$a_v$に関する情報を得ることができない．

また，最後に，攻撃者は 12.より$\prod_{l=1}^{2}\phi\left(\sum_{v=h}^{h+g}(a_v-b_{l,v})^2\right)$を知ることができる．ここで，復元された結果が 0 でなければ，攻撃者は秘密情報を知るには，乱数ϕを知る必要がある．しかし，攻撃者は乱数ϕを知ることができないため，秘密情報a_vは漏洩しないことが言える．よって、攻撃者は秘密情報a_vを個々に知ることはできないことが言える．

6. 考察

提案方式 1，提案方式 2 及び提案方式 3 はそれぞれ 1 文字ごとの検索，複数文字列の AND 検索と AND・OR 検索を実現するが，ドキュメントだけでなく様々な応用が考えられる．

例えば，提案方式 2 における文字列$a_j(j=1,\dots,m)$を登録した複数のパスワードとし，b_jを入力するパスワードとすれば，全てのパスワードが一致しなければ認証成功とならないパスワードの完全一致検証が実現できる．

また，動画などの検索ではその動画の特徴を表すキーワードをインデックスとして登録し，その一致検索を行う場合が多いが，$a_j(j=1,\dots,m)$を登録されたインデックス，$b_i(i=1,\dots,g)$を検索したいインデックスとすれば，インデックスの秘匿検索にも適用できる．

一方，秘密分散を用いた秘匿検索は誰でもデータを登録でき，誰でも秘匿検索できる．それに対して，共通鍵暗号を用いた検索可能暗号は一般に登録者と検索者は同一である．これと同じ制限を設けたい場合，$[a_j]_i$の 1 つをパスワードとすれば，文字の情報に加えて，パスワードを知らない人には一致検索ができないようにできる．ただし，法とする p はサイズの大きい登録値に統一される．

また，公開鍵暗号を用いた検索可能暗号と同様の機能を実現したい場合，検索者はサーバに予めパスワードを分散して登録しておき，登録者は制限なくドキュメントを登録するが，検索時には検索者が登録したパスワードを含めてドキュメント検索すれば，検索者しか検索できないシステムとできる．

7. まとめ

本論文の提案方式 1 では，辻下らによる PPSS 法[11]を変形し，ドキュメントの検索に適用した．これによって，より高速な検索が実現できることを示した．また，提案方式 2 では，PPSS を拡張し，複数の文字列の AND 検索を実現する手法を提案した．これによって，複数パスワードの一致検索を含む様々な応用に適用できるようになることを明らかにした．最後に，提案方式 3 では，複数の異なる文字列による AND・OR 検索が実現できることを示した．

今後の課題として，より広い分野で適用できるようにするため，提案した手法における前提条件である入力情報に 0 を含まないという条件を緩和し，さらに，MATLAB などを用いて実装を行い，提案した手法が効率的であることを明確にしたいと考えている．

参考文献

[1] A. Shamir: “How to share a secret.” Communications of the ACM, 22, (11), pp. 612-613. (1979).

[2] D. X. Song, D. Wagner, A. Perrig: “Practical techniques for searches on encrypted data.” In Proceeding 2000 IEEE Symposium on Security and Privacy. S&P 2000, pp. 44-55. (2000).

[3] D. Boneh, G. D. Crescenzo, R. Ostrovsky: “Public key encryption with keyword search.” In EUROCRYPT 2004. LCNS, Vol. 3027, Springer, Berlin, Heidelberg. (2004).

[4] R. Curtmola, J. Garay, S. Kamara, R. Ostrovsky: “Searchable symmetric encryption: improved definitions and efficient constructions.” In Proceedings of the 13th ACM Conference on Computer and communications Security, pp. 79-88, Alexandria, Virginia, USA. (2006).

[5] J. Kurihara, S. Kiyomoto, K. Fukushima, T. Tanaka: “A new (k,n)-threshold secret sharing scheme and its extension.” In ISC 2008, pp. 455-470, Springer, Berlin, Heidelberg. (2008).

[6] C. Gentry: “Fully homomorphic encryption using ideal lattices.” In Proceedings of the 41st annual ACM symposium on Theory of Computing (STOC 09), pp. 169-178, Bethesda, MD, USA. (2009).

[7] 縫田光司，清水佳奈，荒井ひろみ，浜田道昭，津田宏治，広川貴次，花岡悟一郎，佐久間淳，浅井潔：“加法準同型暗号を用いた化合物データベースの秘匿検索プロトコル．”コンピュータセキュリティシンポジウム 2012 論文集，pp. 382-389. (2012).

[8] K. Kurosawa: “Garbled searchable symmetric encryption.” In International Conference on Financial Cryptography and Data Security, pp. 234-251, Springer, Heidelberg. (2014).

[9] A. A. A. Mohd Kamal, K. Iwamura, H. Kang: "Searchable encryption of image based on secret sharing scheme." 2017 APSIPA ASC. IEEE, pp. 1495-1503, Kuala Lumpur, Malaysia. (2017).

[10] 鴇田恭平，岩村惠市: “高速かつ n<2k-1 において秘密情報に 0 を含んでも実行可能な秘密分散法による秘匿計算法.” 電気学会論文誌C，第 138 巻, 12 号, pp.1634-1645. (2018).

[11] K. Tsujishita, K. Iwamurra: "Password-protected secret sharing scheme with the same threshold in distribution and restoration." In 2018 MobiSecServ. IEEE. (2018).

[12] ムハンマド　カマル　アフマド　アクマル　アミヌディン，岩村惠市: “秘密分散法を用いた四則演算の組み合わせに対して安全な次数変化のない秘匿計算.” 情報処理学会論文誌，第 59 巻，第 9 号，pp. 1581-1595. (2018).