Computer Security Symposium 2020
26 - 29 October 2020

誤り訂正符号を用いた効率的な秘匿部分一致検索

岩村 惠市 [1]　佐久田 尚 [1,*]

概要：検索可能暗号においては全ての検索文字が一致する完全一致検索と，ある範囲の不一致を許容する部分一致検索に分けられる．完全一致検索では検索する文字列が１つに特定されるが，部分一致検索では許容する不一致に応じて異なるパターンの検索文字列に対して複数回の検索を行わなければならない場合が多い．特に，インデックスと呼ばれる検索可能な文字列を事前に準備する手法では，１つの検索文字列に対して多くの異なるインデックスを準備する必要がある．また，それに応じて秘匿検索を行う回数も増大する．よって，効率的な部分一致検索を実現する手法はほとんどないと言える．そこで本論文では，誤り訂正符号を用いることによって，インデックスを用いず１回の検索だけで効率的に部分一致検索を実現する手法を提案する．この手法は秘匿された文章全体に対して１文字ずつずらしながら検索を行うことを１回と数えるが，設定する不一致の数に関係なく１回の検索によって秘匿部分一致検索が可能であり，かつ処理の軽い秘密分散法を基本とするため高速な検索が可能である．

キーワード：検索可能暗号，秘匿部分一致検索，リードソロモン符号，誤り訂正符号

Efficient Partial Concealment Search Using Error Correction Code

Keiichi Iwamura[1]　Hisashi Sakuta[1,*]

Abstract: In searchable cryptography, there are two types of search methods: exact match search, in which all search characters are matched, and partial match search, in which a set range of discrepancies is allowed. In the exact match search, we often have to perform multiple searches for different patterns of search strings depending on the discrepancies allowed. In particular, it is necessary to prepare many different indexes for the search string called index. The number of secret searches is correspondingly increased. Therefore, it can be said that there are few methods to realize efficient partial matching search. In this paper, we propose a method for efficient partial matching retrieval using an error correction code without an index. However, the proposed method is based on the secret decentralized method which is light in processing time and enables us to perform partial matching retrieval once, regardless of the number of discrepancies.

Keywords: Searchable Encryption, Reed-Solomon code, error correcting code

1. はじめに

クラウドコンピューティングでは，データのプライバシー及び秘匿性を確保するため，保存するデータを暗号化して保管する場合が多い．しかし一般に，秘匿したデータを検索する場合，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検索文字列を含むデータ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[1].

検索可能暗号においては全ての検索文字列が一致する完全一致検索と，ある範囲の不一致を許容する部分一致検索に分けられる．完全一致検索では検索する文字列が１つに特定されるが，部分一致検索では許容する不一致に応じて異なるパターンの検索文字列に対して複数回の検索を行わなければならない場合が多い．特に，インデックスと呼ばれる検索可能な文字列を事前に準備する手法では，１つの検索文字列に対する全ての部分一致系列を準備する場合，多くのインデックスを準備する必要がある．

ここで，部分一致検索が必要な場合について考える．検索したい文字列が特定できている場合，完全一致検索を用いればよいが，検索したい文字列が特定できず，その一部分しかわからない場合(検索文字列の一部を忘れた場合等)や広い範囲を検索したい場合等に有効と思われる.例えば，abc という検索文字列において b が思い出せない場合，axc（x は任意の文字）を検索できるようにしたいが，x に相当する部分を全アルファベットに対してインデックスとして準備するのは手間が大きい．また，a または c のみを検索する場合，膨大な数の文章が検索結果として挙げられると考えられ精度が劣化する．著者らが知る限り，効率的で精度の高い部分一致検索を実現する手法はほとんどない．

そこで本論文では，誤り訂正符号を用いることによって，効率的かつ精度が高い部分一致検索を実現する．特に，イ

1 東京理科大学　岩村研究室
Tokyo University of Science, Iwamura Lab

© 2020 Information Processing Society of Japan

ンデックスを準備する必要がなく，上記 x に相当する部分が複数文字を含む，または検索文字列中に複数存在する場合でも１回の検索だけで実行することができる．この手法は秘匿された文章全体に対して１文字ずつずらしながら検索を行うことを１回と数えるが，処理の軽い秘密分散法を基本とするため高速な検索が可能である．

一方，秘密分散を用いる秘匿検索としては，[2]に効率的な完全一致検索を行える手法が提案されている．本提案は完全一致検索を異なる部分が０の部分一致検索として扱えばこの手法の拡張となっている．

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．次に，第３章で[2]を含む本研究に関する関連研究を説明し，第４章で提案方式を示す．最後に，第５章において提案方式の評価・考察を行い，提案方式が検索可能暗号に比べ，効率的かつ高精度で部分一致検索が行えることを示す．高速に秘匿検索を実現できることを示す．

2. 従来研究

従来の秘匿部分一致検索は大きく４つに分類できる．

2.1 インデックスを用いる手法[3]

ドキュメント中のキーワードに対するインデックスを複数生成し，共通鍵暗号で暗号化して登録する．例えば，キーワードを $s = abc$ とする場合，インデックスとして $S_1(s) = \{a, ab, abc\}$，$S_2(s) = \{b, bc\}$，$S_3(s) = \{c\}$ を登録する．検索文字列が $s' = bc$ の場合，それを同様に暗号化して同じインデックスが存在するかどうかを調べる．存在すれば，そのドキュメントはその部分系列を含むとして検索成功とする．

2.2 ブルームフィルタを用いる手法[4]

キーワードごとにブルームフィルタを使用して登録を行う．また，疑似ランダム関数を複数使用して，ブルームフィルタに追加する．文書は任意の暗号化方式で暗号化して保存する．例えば，(″dog″)の検索式を$p = ((w[1] = "d") \wedge (w[2] = "o") \wedge (w[3] = "g")$とする．この検索式に対して，２つの疑似ランダム関数を準備してその結果をブルームフィルタに登録する．検索文字列として″og″を検索する場合，その検索式に対する検索結果は(″dog″)を登録したブルームフィルタに含まれるため，検索成功となる．

2.3 接尾辞オートマトンを用いる手法[5]

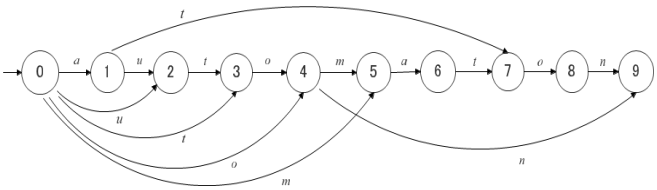

図１　文字列 *automaton* に対する接尾辞オートマトン

接尾辞オートマトンとは文字列の状態遷移を表したものである．図１は文字列”automaton”に対する接尾辞オートマトンであり，キーワード毎に図のような接尾辞オートマトンを準備する．検索文字列に対して，最後の文字まで状態遷移が可能であれば検索成功とする．よって例えば，”maton”を検索文字列とすると最後の文字まで状態遷移が可能であるので検索成功となる．

2.4 T_{Shift} キーワード変換を用いる手法[6]

各キーワードについて，以下に示すモノグラムセットを生成する．単語の各文字は３つの異なるモノグラムで表示され，１つは文字の正確な位置の後に続く文字，２つ目は文字の前の位置の後に続く文字，そして３つ目は文字の次の位置の後に続く文字である．したがって，キーワード’computation’のための変換されたキーワードまたはモノグラムセットは，{′c1′,′c2′,′o2′,′o3′,′o4′,′m3′,′m4′,′m5′,′p4′,′p5′,′p6′,′u5′,′u6′,′u7′,′t5′,′t6′,′t7′,′a6′,′a7′,′a8′,′t7′,′t8′,′t9′,′i8′,′i9′,′i10′,′o9′,′o10′,′o11′,′n10′,′n11′,′n12′}となる．この変換を T_{shift} と呼ぶ．これを用いて，以下のように T_{shift} キーワード変換を用いることにより，一文字ずつずれが生じる部分一致検索が可能になる．

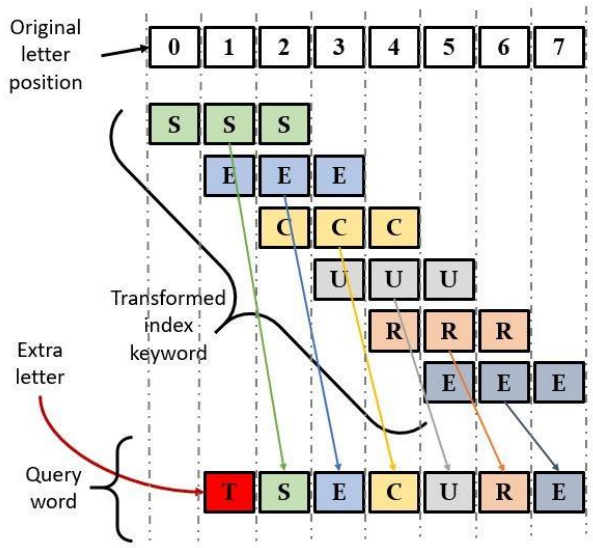

図２　T_{Shift} キーワード変換

3. 関連研究

3.1 (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多項式を用いる手法[7]（以降，Shamir 法）と，栗原らによって提案された XOR によって秘密情報の分散・復元を実現する手法[8]（以降，XOR 法），加法的秘密分散法などがある．以下では特に断らない限り$(k,\ n)$Shamir 法を用い，法とする素数をpとする．また，秘密情報sを分散した分散値を$\overline{[s]}_i$ で表す．

3.2 秘密分散を用いる完全一致検索法

[2]に示される秘密分散を用いた完全一致検索法を説明する．m個の文字からなる文書aの文字列$(a_1,\dots,a_m)$を登録された文書とし，g個の文字からなる検索文字列bの文字列$(b_1,\dots,b_g)$を検索クエリとする．ここで，$x=1,\dots,m,\ y=1,\dots,g$として，登録された文書$a$の文字列を$a_x\ (x=1,\dots,m)$，検索クエリ$b$の文字列を$b_y\ (y=1,\dots,g)$と書く．$g$文字からなる検索クエリ$b$の完全一致検索を実現するには，検索文字列の先頭位置を$h+1$として，文章a中の連続するg個の文字列$a_v\ (v=h+1,\dots,h+g)$に対して，a_xとb_yの差$(a_{h+1}-b_1),\dots,(a_{h+g}-b_g)$を$c_1,\dots,c_g$とすると，その文字間の差$c_1,\dots,c_g$が全て0のときのみ，$f(c_1,\dots,c_g)=0$となる$(a_{h+1}-b_1)^2+\cdots+(a_{h+g}-b_g)^2={c_1}^2+\cdots+{c_g}^2$を秘匿計算すればよい．

ただし，a_x,b_yのサイズをqとすると，法とするpを$gq^2<p$とする．これは上記演算結果が0以外で0となることを防ぐためである．また，gは全数探索を防げる範囲で任意の値を選択できる（後述するパスワードを用いればgは任意の長さを選択可能）．

この手法では変換用乱数組と呼ばれる攻撃者が知らない乱数の積$\varepsilon_x^{(u)}$の分散値とそれを構成する乱数$\varepsilon_{x,j}^{(u)}$の組$\left[\varepsilon_x^{(u)}\right]_i=\left(\overline{\left[\varepsilon_x^{(u)}\right]}_i,\varepsilon_{x,j}^{(u)}\right)$を必要とするが，変換用乱数組は以下のようにして生成される．なお，以下のアルゴリズムで生成する乱数$\varepsilon_{x,0}^{(u)},\dots,\varepsilon_{x,k-1}^{(u)}$は$GF(p)$の要素であり，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．

Protocol 3: ***【変換用乱数組生成処理】***

1. k 個の乱数 $\varepsilon_{x,0}^{(u)},\dots,\varepsilon_{x,k-1}^{(u)}\ (x=1,\dots,m,\ u=1,\dots,$ *必要個数*$)$を生成し，次式で乱数$\varepsilon_x^{(u)}$を計算する．

$$\varepsilon_x^{(u)}=\prod_{j=0}^{k-1}\varepsilon_{x,j}^{(u)}$$

2. $\varepsilon_x^{(u)}$ を Shamir 法で n 台のサーバ $S_i\ (i=0,\dots,n-1)$に分散し，$\varepsilon_{x,j}^{(u)}$をサーバS_iに送る．
3. 全サーバS_iは変換用乱数組として以下を保持する．

$$\left[\varepsilon_x^{(u)}\right]_i=\left(\overline{\left[\varepsilon_x^{(u)}\right]}_i,\varepsilon_{x,j}^{(u)}\right)$$

[2]の完全一致検索のアルゴリズムの説明はページ数の都合で省略するが，これによって semi-honest な攻撃者に対して情報理論的安全性をもつ秘匿検索が実現できる．

4. 提案方式

4.1 提案方式の概要

完全一致検索では${c_i}^2(i=1,\dots,g)$が全て 0 であるかどうかを検証した．これは$C=[{c_1}^2,\dots,{c_g}^2]$をベクトルとみなした時，下記を計算して検証しているのと等しい．ただし，C^TはCの転置を表す．

$$[1,\dots,1]C^T=0 \qquad (1)$$

${c_i}^2$の中に不一致箇所がある場合，その要素は 0 とはならない．これは，$[{c_1}^2,\dots,{c_g}^2]$を誤り訂正符号における受信語とみなしたとき，[0,⋯,0]が正しい符号語であり，不一致箇所は誤りに相当する．よって，不一致箇所が一か所のとき以下を計算して，誤り訂正をすればC'^TをC^Tに戻すことができる．C'^Tは一か所が 0 でない誤りを含むベクトルの転置とする．

$$\begin{bmatrix}1 & 1 & \cdots & 1\\ 1 & 2 & \cdots & g\end{bmatrix}C'^T=\begin{bmatrix}s1\\ s2\end{bmatrix} \qquad (2)$$

よって，提案方式は式(1)に加えて，式(2)を検査する処理を加えたものであり，【誤り訂正処理】として以下に示す．

4.2 秘密分散を用いる部分一致検索法

以下に提案方式の具体的アルゴリズムを示す．このアルゴリズムは文章aを秘匿して登録する登録者と，検索クエリbを含む文章を検索する検索者からなる．登録された文章の文字列a_xと検索クエリb_yのサイズはqであるが$GF(p)$の要素であり，【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$\alpha_{x,j}$，$\gamma_{x,j}$，c_x，$\beta_{y,j}$，$\delta_{y,j}$，d_y，$\rho_{h,j}$，$\varphi_{h,j}$，$\kappa_{h,j}$は$GF(p)$の要素である（ただし，生成する乱数は 0 の値を除く）．また，以下に示す秘

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 1 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】の手順(4)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】における手順 2, 3. を$c_v + d_v \neq 0$となるまで行う，または手順 2 において異なる 2 個のc_v, d_vを送っておき，手順(3)で組合せを変えて計算する．また，各サーバは前述の変換用乱数組$\left(\overline{\left[\varepsilon_x^{(u)}\right]_i}, \varepsilon_{x,i}^{(u)}\right)$を持つとする．なお，以下では説明を簡単にするために$n = k$として説明するが，$x = 1, \dots, m$, $y = 1, \dots, g$, $i = 0, \dots, n-1$, $j = 0, \dots, k-1$とする．また，簡単のために検索文字列の 1 文字が異なる，すなわち 1 重誤り訂正の場合を【誤り訂正処理】として示す．異なる部分が t 文字存在する場合は t 重誤り訂正を行うことによって対応できる．よって，【誤り訂正処理】以外は前述の秘密分散を用いた完全一致と同様である．

Notations:

- a：m個の文字$a_x (x = 1, \dots, m)$からなる文章a.
- $[a_x]_i'$：サーバ$S_i (i = 0, \dots, n-1)$が保持する文字a_xに関する秘匿値の集合.
- $[a_x]_i$：サーバS_iが保持する文字a_xと乱数c_vに関する秘匿値の集合.
- $[a]_i$：サーバS_iが保持する文章aに関する秘匿値の集合．すなわち，$[a]_i = [a_1]_i, \dots, [a_m]_i$.
- b：g個の文字$b_y (y = 1, \dots, g)$からなる検索クエリb.
- $[b_x]_i'$：サーバ$S_i (i = 0, \dots, n-1)$が保持する文字b_xに関する秘匿値の集合.
- $[b_x]_i$：サーバS_iが保持する文字b_xと乱数d_vに関する秘匿値の集合.
- $[b]_i$：サーバS_iが保持する検索文字列bに関する秘匿値の集合．すなわち，$[b]_i = [b_1]_i, \dots, \left[b_g\right]_i$.

Protocol 4.1:【ドキュメントの秘匿化処理】

- Input: a_x $(x = 1, \dots, m)$
- Output: $[a]_i = [a_1]_i, \dots, [a_m]_i$ $(i = 0, \dots, n-1)$

1. 登録者は【変換用乱数組生成処理】を実行し，以下の変換用乱数組$\left[\varepsilon_x^{(2)}\right]_i, \left[\varepsilon_x^{(4)}\right]_i, \left[\varepsilon_x^{(5)}\right]_i, \left[\varepsilon_x^{(6)}\right]_i, \left[\varepsilon_x^{(7)}\right]_i, \left[\varepsilon_x^{(8)}\right]_i (x = 1, \dots, m)$を事前に計算し，サーバ$S_i$に送信しておく．サーバ$S_i$は以下の情報を保持する．

$$\left[\varepsilon_x^{(2)}\right]_i = \left(\overline{\left[\varepsilon_x^{(2)}\right]_i}, \varepsilon_{x,i}^{(2)}\right), \quad \left[\varepsilon_x^{(4)}\right]_i = \left(\overline{\left[\varepsilon_x^{(4)}\right]_i}, \varepsilon_{x,i}^{(4)}\right),$$

$$\left[\varepsilon_x^{(5)}\right]_i = \left(\overline{\left[\varepsilon_x^{(5)}\right]_i}, \varepsilon_{x,i}^{(5)}\right), \ \left[\varepsilon_x^{(6)}\right]_i = \left(\overline{\left[\varepsilon_x^{(6)}\right]_i}, \varepsilon_{x,i}^{(6)}\right),$$

$$\left[\varepsilon_x^{(7)}\right]_i = \left(\overline{\left[\varepsilon_x^{(7)}\right]_i}, \varepsilon_{x,i}^{(7)}\right), \ \left[\varepsilon_x^{(8)}\right]_i = \left(\overline{\left[\varepsilon_x^{(8)}\right]_i}, \varepsilon_{x,i}^{(8)}\right)$$

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

$$\alpha_x = \prod_{j=0}^{k-1} \alpha_{x,j}, \qquad \gamma_x = \prod_{j=0}^{k-1} \gamma_{x,j}$$

3. 登録者は計算した乱数α_x, γ_xをそれぞれ秘密情報$(a_x + 1)$と乱数c_xにかけて，$\alpha_x(a_x + 1)$, $\gamma_x c_x$を計算して，以下の$[a_x]_j', [c_x]_j'$を秘匿計算に参加するサーバS_jに送る．

$$[a_x]_j' = \left(\alpha_x(a_x + 1), \alpha_{x,j}\right)$$

$$[c_x]_j' = \left(\gamma_x c_x, \gamma_{x,j}\right)$$

4. $[a_x]_j \coloneqq \left([a_x]_j', [c_x]_j'\right)$として，サーバ$S_j$は以下の情報を保持する．

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

Protocol 4.2:【検索クエリ生成処理】

- Input: $b_y (y = 1, \dots, g)$
- Output: $[b]_i = [b_1]_i, \dots, \left[b_g\right]_i$ $(i = 0, \dots, n-1)$

1. 検索者は【変換用乱数組生成処理】を実行し，変換用乱数組$\left[\varepsilon_x^{(1)}\right]_i, \left[\varepsilon_x^{(3)}\right]_i (x = 1, \dots, m)$を事前に計算し，サーバ$S_i$に送信しておく．サーバ$S_i$は以下の情報を保持する．

$$\left[\varepsilon_x^{(1)}\right]_i = \left(\overline{\left[\varepsilon_x^{(1)}\right]_i}, \varepsilon_{x,i}^{(1)}\right), \qquad \left[\varepsilon_x^{(3)}\right]_i = \left(\overline{\left[\varepsilon_x^{(3)}\right]_i}, \varepsilon_{x,i}^{(3)}\right)$$

2. 検索者は検索したい文字列$b_y (y = 1, \dots, g)$に対して，乱数$\beta_{y,j}, \delta_{y,j}, d_y$ $(j = 0, \dots, k-1)$を生成し，以下を計算する．

$$\beta_y = \prod_{j=0}^{k-1} \beta_{y,j}, \quad \delta_y = \prod_{j=0}^{k-1} \delta_{y,j}$$

3. 検索者は計算した乱数β_y, δ_yをそれぞれ$(b_y + 1)$と乱数d_yにかけて，$\beta_y(b_y + 1), \delta_y d_y$を計算して，以下の$\left[b_y\right]_j', \left[d_y\right]_j'$をサーバ$S_j$に送る．.

$$\left[b_y\right]_j' = \left(\beta_y(b_y + 1), \beta_{y,j}\right)$$

$$[d_y]_j{}' = (\delta_y d_y, \delta_{y,j})$$

4. $[b_y]_j \coloneqq \left([b_y]_j{}', [d_y]_j{}'\right)$として，サーバ$S_j$は以下の情報を保持する．

$$[b]_j = [b_1]_j, \dots, [b_g]_j$$

Protocol 4.3: Searching Process1【完全一致検索処理】

- Input：$[b]_j, [a]_j$ $(j = 0, \dots, k-1)$
- Output: a, or $\perp$

1. k台のサーバS_jはまず$h = 0$としてa中の連続するg個の文字列と検索文字列bに対して以下を行う．以下では，$y = 1, \dots, g, v = h + y$とする．

(1) サーバS_jは乱数$\rho_{h.j}, \varphi_{h,j}$を生成し，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）．

$$\frac{\rho_{h,j}}{\gamma_{v,j}\varepsilon_{v,j}^{(1)}}, \frac{\rho_{h,j}}{\delta_{y,j}\varepsilon_{v,j}^{(2)}}, \frac{\varphi_{h,j}}{\alpha_{v,j}\varepsilon_{v,j}^{(3)}}, \frac{\varphi_{h,j}}{\beta_{y,j}\varepsilon_{v,j}^{(4)}}, \frac{\varphi_{h,j}}{\rho_{h,j}\varepsilon_{v,j}^{(5)}}$$

(2) サーバS_0は以下を計算し，サーバS_jに送信する．

$$\frac{\rho_h}{\gamma_v\varepsilon_v^{(1)}} = \prod_{j=0}^{k-1}\frac{\rho_{h.j}}{\gamma_{v,j}\varepsilon_{v,j}^{(1)}}, \quad \frac{\rho_h}{\delta_y\varepsilon_v^{(2)}} = \prod_{j=0}^{k-1}\frac{\rho_{h.j}}{\delta_{y,j}\varepsilon_{v,j}^{(2)}}$$

$$\frac{\varphi_h}{\alpha_v\varepsilon_v^{(3)}} = \prod_{j=0}^{k-1}\frac{\varphi_{h,j}}{\alpha_{v,j}\varepsilon_{v,j}^{(3)}}, \quad \frac{\varphi_h}{\beta_y\varepsilon_v^{(4)}} = \prod_{j=0}^{k-1}\frac{\varphi_{h,j}}{\beta_{y,j}\varepsilon_{v,j}^{(4)}}$$

$$\frac{\varphi_h}{\rho_h\varepsilon_v^{(5)}} = \prod_{j=0}^{k-1}\frac{\varphi_{h,j}}{\rho_{h,j}\varepsilon_{v,j}^{(5)}}$$

(3) サーバS_jは以下を計算し，サーバS_0に送る．

$$[\rho_h(c_v + d_y)]_j = \gamma_v c_v \times \frac{\rho_h}{\gamma_v\varepsilon_v^{(1)}} \times \overline{[\varepsilon_v^{(1)}]}_j + \delta_y d_y \times \frac{\rho_h}{\delta_y\varepsilon_v^{(2)}} \times \overline{[\varepsilon_v^{(2)}]}_j$$

(4) サーバS_0は$\rho_h(c_v + d_y)$をShamir法で復元し，サーバS_jに送る．

(5) サーバS_jは以下を計算し，サーバS_0に送る．

$$[\varphi_h(a_v - b_y + c_v + d_y)]_j = \alpha_v(a_v + 1) \times \frac{\varphi_h}{\alpha_v\varepsilon_v^{(3)}} \times \overline{[\varepsilon_v^{(3)}]}_j - \beta_y(b_y + 1) \times \frac{\varphi_h}{\beta_y\varepsilon_v^{(4)}} \times \overline{[\varepsilon_v^{(4)}]}_j + \rho_h(c_v + d_y) \times \frac{\varphi_h}{\rho_h\varepsilon_v^{(5)}} \times \overline{[\varepsilon_v^{(5)}]}_j$$

(6) サーバS_0は$\varphi_h(a_v - b_y + c_v + d_y)$をShamir法で復元し，サーバ$S_j$に送る．

(7) サーバS_jは乱数$\kappa_{h,j}$を生成し，以下を計算し，サーバS_0に送る．

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

(8) サーバS_0は以下を計算し，サーバS_jに送信する．

$$\frac{\kappa_h}{(\varphi_h)^2\varepsilon_v^{(6)}} = \prod_{j=0}^{k-1}\frac{\kappa_{h,j}}{(\varphi_{h,j})^2\varepsilon_{v,j}^{(6)}}$$

$$\frac{\kappa_h}{\varphi_h\rho_h\varepsilon_v^{(7)}} = \prod_{j=0}^{k-1}\frac{\kappa_{h,j}}{\varphi_{h,j}\rho_{h,j}\varepsilon_{v,j}^{(7)}}$$

$$\frac{\kappa_h}{(\rho_h)^2\varepsilon_v^{(8)}} = \prod_{j=0}^{k-1}\frac{\kappa_{h,j}}{(\rho_{h,j})^2\varepsilon_{v,j}^{(8)}}$$

(9) サーバS_jは以下を計算する．

$$\sum_{y=1}^{g}\overline{\left[\kappa_h(a_v - b_y)^2\right]}_j = \sum_{v=h+1}^{h+g}\left\{\left[(\varphi_h)^2(a_v - b_y + c_v + d_y)^2 \times \frac{\kappa_h}{(\varphi_h)^2\varepsilon_v^{(6)}} \times \overline{[\varepsilon_v^{(6)}]}_j\right] - \left[2 \times \varphi_h(a_v - b_y + c_v + d_y) \times \rho_h(c_v + d_y) \times \frac{\kappa_h}{\varphi_h\rho_h\varepsilon_v^{(7)}} \times \overline{[\varepsilon_v^{(7)}]}_j\right] + \left[(\rho_h)^2(c_v + d_y)^2 \times \frac{\kappa_h}{(\rho_h)^2\varepsilon_v^{(8)}} \times \overline{[\varepsilon_v^{(8)}]}_j\right]\right\}$$

(10) サーバS_jは協力して$\sum_{y=1}^{g}\kappa_h(a_v-b_y)^2$の結果をShamir法で復元し，0であれば一致とし，手順2.に移る．

(11) 一致しなければ，サーバ'S_jは$h=h+1$として手順(1)~(11)を$h=m-g$まで繰り返す．

2. サーバS_j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，文章aのk個の$[a]_i'$を検索者に送信する．一致した文章がなければ誤り訂正処理に移る．

Protocol 4.4: Searching Process2【誤り訂正処理】

1. サーバS_jは完全一致検索で用いた変数を用いて以下を計算する．

$$\sum_{y=1}^{g} y\overline{\left[\kappa_h(a_v-b_y)^2\right]_j}$$

$$=\sum_{v=h+1}^{h+g} y\left\{\left[(\varphi_h)^2(a_v-b_y+c_v+d_y)^2\times\frac{\kappa_h}{(\varphi_h)^2\varepsilon_v^{(6)}}\times\overline{\left[\varepsilon_v^{(6)}\right]_j}\right]-\left[2\times\varphi_h(a_v-b_y+c_v+d_y)\times\rho_h(c_v+d_y)\times\frac{\kappa_h}{\varphi_h\rho_h\varepsilon_v^{(7)}}\times\overline{\left[\varepsilon_v^{(7)}\right]_j}\right]+\left[(\rho_h)^2(c_v+d_y)^2\times\frac{\kappa_h}{(\rho_h)^2\varepsilon_v^{(8)}}\times\overline{\left[\varepsilon_v^{(8)}\right]_j}\right]\right\}$$

2. サーバは協力して以下を復元する．

$$s1=\sum_{y=1}^{g}\kappa_h(a_v-b_y)^2$$

$$\mathrm{s2}=\mathrm{y}\sum_{y=1}^{g}\kappa_h(a_v-b_y)^2$$

3. サーバは$\mathrm{y}'=\mathrm{s2}/\mathrm{s1}$を計算し，それが検索クエリの範囲に入っていなければ誤訂正として処理を停止し，そうでなければ4. に行く．
4. サーバはy'の位置に当たる文字を除いて$\mathrm{s3}=\sum_{y=1,y\neq y'}^{g}\overline{\left[\kappa_h(a_v-b_y)^2\right]_j}$を計算し復元する．
5. 上記を$h=0$から$h=m-g$まで繰り返し，$\mathrm{s3}=0$となれば，1文字を除いて部分一致したと判断し，文章aのk個の$[a]_i'$を検索者に送信する．一致した文章がなければ検索処理を停止する．

5. 考察及び評価

5.1 提案方式の安全性

4章に示す方式では鍵またはパスワードが用いられていないが，[2]と同様にパスワードを秘密分散して完全一致検索を行えばパスワードを知らない攻撃者は何度検索しても秘密情報であるキーワードは漏洩しない．すなわち，パスワードに対して【完全一致検索処理】を行い，手順2. において一致しなければ，処理を終了する．完全一致した場合は，キーワードに対する【誤り訂正処理】を含む部分一致検索を行う．完全一致検索に関する安全性は[2]に示されており，情報理論的安全性を持つことが証明されている．

また，【誤り訂正処理】では$\kappa_h(a_v-b_y)^2$を$c_i{}^2(i=1,\dots,g)$として誤り訂正を行うが，秘匿計算により完全一致検索時に知られる$s1$に加えて，式(2)の計算結果である$s2$が知られる．$s1$は誤りパターンを示しており，$s1$と$s2$の比は誤り位置を示している．ただし，誤りが２以上ある場合，手順3. 及び5. によって誤訂正は防止される．また，そのような場合$2t$個のシンドロームから$t+1$個以上の誤りに関する位置やパターンに関する情報は得られない（$2t$までの誤り検出は可能であるが，これは$2t$まで誤検出がないことを意味し，誤り位置の検出は誤りパターンなどの未知の値を含むため特定不能）．誤りがt個の場合，$2t$個のシンドロームから，誤り訂正可能であ，る．ただし，【誤り訂正処理】では，誤り位置検出のみを行う．これは誤り訂正しても異なる文字間の差分は乱数κ_hによって秘匿されているので文字間の差分はわからないためである．ただし，正しく誤りが検出された場合，文章aが送られるので，検索者はそれを復元することによって文字間の違いを知ることができる．

よって，【誤り訂正処理】では正しく誤りが検出された場合，設定した範囲の誤り位置のみが知られるが，検索文字列に関する情報は漏洩しない．

5.2 提案方式の誤り検出能力

４章では誤りの個数が１個の場合を示したが，提案方式は任意の数の誤りに対応できる．すなわち，【誤り訂正処理】において以下から得られるs_1～s_{2t}をシンドロームとしてt重誤り訂正を行えばよい（$1\le t\le g$）．

$$\begin{bmatrix}1 & 1 & \dots & 1\\ 1 & 2 & \dots & 2^g\\ \vdots & \vdots & \dots & \vdots\\ 1 & 2t & \dots & (2t)^g\end{bmatrix}C'^T=\begin{bmatrix}s_1\\ s_2\\ \vdots\\ s_{2t}\end{bmatrix}\qquad(3)$$

提案方式では用いるpを$gq^2 < p$とするため非常に長い符号長を設定できる．よって，Cはそれをパリティを含んで短縮符号化した符号語とみなせる．パリティを含んだ短縮符号化とは，例えばg重誤り訂正を行う場合，Cをg個のパリティ，その前に全０のg個のパリティがあり，さらにその前に全０の情報シンボルがあるとしたものである．従って，手順 3.において誤り位置が検索クエリの範囲に入っていなければ誤訂正として検出される．

ただし一般に，部分一致検索では検索クエリ内の多くの誤りに対応することは少なく，通常 1 文字または 2 文字程度の違いのみに対応する場合が多い．よって，提案方式では許容する誤りの範囲をあらかじめ定めて必要以上のシンドロームの復元は行わないようにする必要がある．提案方式は semi-honest な攻撃者を想定するため，必要以上のシンドロームは復元されない．従って，提案方式は設定した任意の数の誤りに対応できる．

また，提案方式では表現をわかりやすくするため【完全一致検索処理】と【誤り訂正処理】を分けて説明したが，【完全一致検索処理】の手順(9)において

$\sum_{y=1}^{g}\overline{\left[\kappa_h\left(a_v - b_y\right)^2\right]}_j$と$\sum_{y=1}^{g} y\overline{\left[\kappa_h\left(a_v - b_y\right)^2\right]}_j$を計算し，

【誤り訂正処理】の手順 2. を行ってもよい．この場合，誤りがなければ$s1 = s2 = 0$となり，完全一致していることがわかる．そうでない場合，【誤り訂正処理】の手順 3. を行い 1 重誤り訂正を行う．これによって，1 回の検索で完全一致検索と部分一致検索を同時に実現できる．

5.3 性能評価

提案方式と従来方式を柔軟性，精度，計算量の観点から評価する．

a. 柔軟性

従来方式は事前にインデックス[3], ブルームフィルタ[4], 接尾辞オートマトン[5], モノグラムセット[6]などを準備する必要があり，その事前に準備された範囲の検索のみが可能である．それに対して，提案方式は事前の準備無しに文章に含まれる任意の言葉を検索可能であり，柔軟性が非常に高い．

また，従来方式は検索文字列に記憶違いやミスなどによって登録していない文字を 1 つでも含んで入力すれば，検索できない．それに対して提案方式は設定範囲内の間違いであればその入力に近いキーワードを含む文章を検索可能であり，この観点からも柔軟性が高いと言える．

b. 精度

検索文字列の一部が不明でも意図する文章を絞り込んで検索できる精度について考える．

例えば, キーワード”abc”に対して”axc”（x は不明の文字）を検索したい場合，従来方式では”a”または”c”を検索するしかないが，非常に多くの文章が検索され，その精度は非常に悪いと言える．また，”a”と”c”の AND 検索が実行できたとしても”a”と”c”の間の文字数や順番を指定できないため”a”と”c”を同時に含む文章が全て検索され，やはり精度は悪いと考えられる. それに対して, 提案方式は”azc”（z は任意の文字）を検索文字列として 1 重誤り訂正を設定して検索すれば，”a”と”c”の間に 1 文字のみ含む部分一致検索が可能である. また, 2 重誤り訂正を設定した場合, ”a”と”c”の間に 2 文字含む部分一致検索が可能であり，意図する文章が得られる精度が高い．

また，あるキーワードについて検索したとき，多くの文章が検索された場合，提案方式は[2]を基本とするため AND/OR 検索が可能である．よって，さらなる絞り込み検索にも対応できる．

一方，ブルームフィルタは多くの単語を登録すると，登録していないキーワードを検出する偽陽性の検出を行う場合があり，従来方式の中でも精度が悪いと言える．

c. 計算量

計算量については，従来方式ではキーワードを登録するための計算量が大きいと考えられる．特に，柔軟性を高くしたい場合，文章中の全ての表現を細かな単位でキーワードとして登録する必要があるため膨大な計算量が必要と考えられる．それに対して提案方式の登録処理は文章の 1 文字ごとに秘密分散を行うだけであり，大きな計算量は必要としない．

ただし，検索処理については従来方式の方が高速と考えられる．それは検索文字列を 1 度暗号化して比較・検索を行うだけでよいためである．提案方式は比較による検索が成功しない場合（完全一致でない場合）, 誤り訂正処理を行う必要があるため計算量が大きいと思われる．しかし，誤り訂正所は 1 重または 2 重程度を想定するため，比較的処理を軽くすることができる．ただし，この部分については今後，実装などによって検証していく必要がある．

6. おわりに

提案方式は従来法に比べて非常に高い精度で意図する文章を部分一致検索できると言える．

残された課題として実装による実用性の検証と，文字飛びへの対応が考えられる．例えば，キーワード”abcde”に対して,b を忘れて検索クエリを”acde”とした場合，誤った個所は 1 つであるが結果的に連続して 3 文字異なるため 1,2 重誤り訂正程度では対応できない．”afbcde”の場合も同様である．実装による検証が終わった後は文字飛びにも対応できるように拡張していく予定である．

参考文献

[1]松田規，伊藤隆，柴田秀哉，服部充洋，平野貴人, “検索可能暗号の高速化と Web アプリケーションへの適用方式に関する提案”，分散協調とモバイルシンポジウム 2013 論文集, vol 2013, p2067-2074

[2]Ahmad Akmal Aminuddin Mohd Kamal, Keiichi Iwamura, “Searchable Encryption Using Secret Sharing Scheme that Realizes Direct Search of Encrypted Documents and Disjunctive Search of Multiple Keywords”, 2019 IEEE Intl Conf on Dependable

[3]平野貴人，川合豊，太田和夫，岩本貢, “共通鍵暗号型の秘匿部分一致検索(その 1)”, 2016 Symposium on Cryptography and Information Security Kumamoto, Japan, Jan. 19 - 22, 2016

[4]Takanori Suga, Takashi Nishide and Kouichi Sakurai,”Character-based symmetric searchable encryption and its implementation and experiment on mobile devices” Security and Communication Networks 2016, vol. 9 p. 1717-1725

[5]山本博章，宮嵜敬 “暗号データに対する部分文字列検索可能な安全な検索法” 2016 Symposium on Cryptography and Information Security Kumamoto, Japan, Jan. 19 - 22, 2016

[6]M A Manazir Ahsan, Musarat Sabilah, Ainuddin Wahid Bin, "An Efficient Fuzzy Keyword Matching Technique for Searching Through Encrypted Cloud Data" ,Abdul Wahab2017 International Conference on Research and Innovation in Information Systems (ICRIIS)

[7]A.Shamir . “How to Share a Secret.”, Communications of the ACM,vol,22,no.11,pp.612-613,1979

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