情報処理学会論文誌 Vol.63 No.2 684–693 (Feb. 2022)

誤り訂正符号を用いた秘密分散による効率的な秘匿部分一致検索

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

受付日 2021年5月10日, 採録日 2021年11月2日

概要：検索可能暗号においてはすべ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される文字列が 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，効率的な秘匿部分一致検索を実現する手法はほとんどないといえる．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ができるが，設定する不一致の数に関係なく文章全体の検索を 1 回行うだけで求める秘匿部分一致検索を高速に実現することができる．また，この手法はインデックスを用いる方式にも対応でき，より効率的な部分一致検索を実現する．

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 Matching Search Using Error Correction Code

Keiichi Iwamura[1,a)] Ahmad Akmal Aminuddin Mohd Kamal[1,b)]

Received: May 10, 2021, Accepted: November 2, 2021

Abstract: In searchable encryption, there are two types of search methods: exact matching search, in which all search characters matched, and partial matching search, in which a set range of difference is allowed. In the exact matching search, only one specified character string needs to be searched. In contrast, in the partial matching search, depending on the range of mismatches allowed, multiple search processes are required to be performed on every different pattern of the search query. In particular, in the method that realizes partial matching search using an index search, multiple pattern of indexes regarding the search query need to be prepared, therefore increasing the number of searches required. Therefore, it can be said that there are almost no methods that allow for an efficient partial matching search. In this paper, we propose a method of partial matching search using error correction code. We propose an efficient partial matching search where the searching is performed only once without the use of multiple indexes. Our method will also work with a method that uses index search, therefore realizing a more efficient partial matching search.

Keywords: searchable encryption, partial matching, Reed-Solomon code, error correction 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およ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なしにサーバ

[1] 東京理科大学
Tokyo University of Science, Katsushika, Tokyo 125–8585, Japan
[a)] iwamura@ee.kagu.tus.ac.jp
[b)] ahmad@sec.ee.kagu.tus.ac.jp

上で検索を行うことができない．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 [2], [3], [4], [5], [6], [7], [8], [9], [11], [12], [13], [14], [15], [16], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28], [29], [30]．

検索可能暗号は一般に，共通鍵暗号を用いる手法 [2], [4], [6], [7], [12], [14], [15], [16], [18], [21], [22], [26], [29], [30], 公開鍵暗号を用いる手法 [3], [5], [8], [9], [11], [13], [17], [20], [23], [27] と秘密分散を用いる手法 [24], [28] に分類することができる．共通鍵暗号を用いる検索可能暗号は，検索者は暗号化の鍵を知っている必要があるため，検索者は暗号化に使用した秘密鍵を知る登録者に限られる．それに対して，Boneh らによる手法 [3] をはじめとした公開鍵暗号を用いる検索可能暗号も多数提案されている．ただし，公開鍵暗号を用いる手法の多くは，完全準同型暗号などのような計算量が多い処理を利用しているため，膨大な計算量が必要となり，大量のデータから検索したいデータを取り出すには非効率と考えられる．よって，近年では，より軽い計算量を持つ秘密分散法を用いる手法 [24], [28] が注目されている．

また，検索可能暗号の検索機能として，すべ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 [15], [21], [22], [25] に分ける場合も多い．完全一致検索では検索される文字列は 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 [21] では，1 つの検索文字列に対してすべての部分一致系列を準備する場合，非常に手間がかかるという問題がある．

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ない場合（検索文字列の一部を忘れた場合など）や，より広い範囲を検索したい場合などに有効と考えられる．たとえば，abc という検索文字列において b が思い出せない場合，または a と c の間に任意の文字を含む広い範囲の検索をしたい場合，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あ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 に相当する部分が検索文字列中に複数存在する場合でも設定した訂正範囲内であれば 1 回の検索で効率的な秘匿部分一致検索を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ができるが，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．また，この手法はインデックスを用いる方式にも対応でき，より効率的な部分一致検索を実現できる．

一方，秘密分散を用いる秘匿検索としては，文献 [28] に完全一致検索を行える手法が提案されている．本提案はこの完全一致検索を拡張して設定された誤りまでの不一致に対応できる部分一致検索を実現する．

本論文の構成を以下に示す．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，3 章で文献 [28] を含む本研究に関する関連研究を説明し，4 章で提案方式を示す．最後に，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．

2. 従来研究

従来の秘匿部分一致検索として以下のようなものがある．

2.1 インデックスを用いる手法 [21]

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．たとえば，キーワードを $s = abc$ とする場合，インデックスとして $S_1(s) = \{a, ab, abc\}$，$S_2 = \{b, bc\}$,，$S_3 = \{c\}$ 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 [15]

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．たとえば，(“dog”) の検索式を $p = ((w[1] = \text{“}d\text{”}) \wedge (w[2] = \text{“}o\text{”}) \wedge (w[3] = \text{“}g\text{”}))$ 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 [22]

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，キーワードごと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

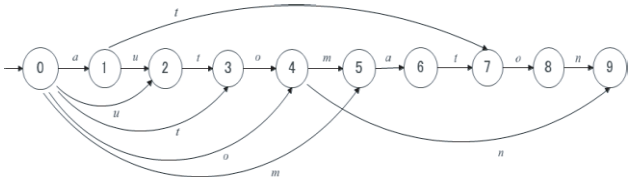

図 1 文字列 “*automaton*” に対する接尾辞オートマトン
Fig. 1 Suffix automaton for the character string “*automaton*”.

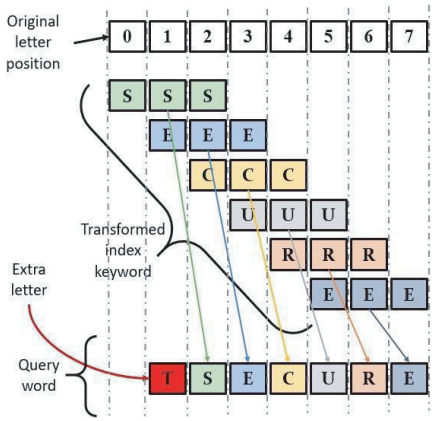

図 2 T_{shift} キーワード変換
Fig. 2 Keyword transformation, T_{shift}.

で状態遷移が可能であれば検索成功とする．よってたとえば，“*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 [25]

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以下のようになる．

{‘*c*1’, ‘*c*2’, ‘*o*2’, ‘*o*3’, ‘*o*4’, ‘*m*3’, ‘*m*4’, ‘*m*5’, ‘*p*4’,
‘*p*5’, ‘*p*6’, ‘*u*5’, ‘*u*6’, ‘*u*7’, ‘*t*5’, ‘*t*6’, ‘*t*7’, ‘*a*6’, ‘*a*7’,
‘*a*8’, ‘*t*7’, ‘*t*8’, ‘*t*9’, ‘*i*8’, ‘*i*9’, ‘*i*10’, ‘*o*9’, ‘*o*10’, ‘*o*11’,
‘*n*11’, ‘*n*12’}

この変換を T_{shift} と呼ぶ．これを用いて，**図 2** のように T_{shift} キーワード変換を用いることにより，1 文字ずつずれが生じる部分一致検索が可能になる．

3. 関連研究

3.1 (k, n) 閾値秘密分散法

次の 2 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 に関する情報をいっさい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 [1]（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 [10]（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 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 秘密分散法を用いる完全一致検索法

文献 [28] 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_x（$x = 1, \ldots, m$）を登録文書とし，g 個の文字からなる検索文字列 b の文字列 b_y（$y = 1, \ldots, g$）を検索クエリとする．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，文書 a 中の連続する g 個の文字列 a_v（$v = h+1, \ldots, h+g$）に対して，a_v と b_y の差 $(a_{h+1} - b_1), \ldots, (a_{h+g} - b_g)$ を $c_1, \ldots, c_g$ とする．その文字間の差 $c_1, \ldots, c_g$ がすべて 0 のときのみ 0 となる関数を $f(c_1, \ldots, c_g)$ として，$f(c_1, \ldots, c_g) = {c_1}^2 + \cdots + {c_g}^2$ を秘匿計算する．ただし，a_x，b_y のサイズを q とすると，法とする p を $gq^2 < p$ となる素数とする．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．

ここで，文書 a および検索クエリ b 中の文字 a_x，b_y は，たとえば英語であれば ASCII コード，日本語であれば JIS 漢字のように数値で表現できるとし，a_v と b_y の差とはその数値間の差となる．また，a_x，b_y のサイズとは a_x，b_y を表現するための空間の大きさを指し，ASCII コードであれば 1 文字 7 ビットで表されるため $q = 2^7$，JIS 漢字であれば 1 文字 16 ビットで表されるので $q = 2^{16}$ となる．よって，ASCII コードおよび JIS 漢字の各文字は $GF(q)$ の元と見なすことができる．また，${c_i}^2$（$i = 1, \ldots, g$）のサイズは最大 q の二乗であり，それを g 個加算するため，最大 gq^2 となる．よって，法とする p を gq^2 より大きくとれば，${c_i}^2$（$i = 1, \ldots, g$）がすべて 0 以外で加算結果が 0 となることはない．たとえば，ASCII コードに対して g を 16 程度とする場合，p を 19 ビット以上の素数とすればよい．

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 $[\varepsilon_x^{(u)}]_i = (\overline{[\varepsilon_x^{(u)}]}_i, \varepsilon_{x,j}^{(u)})$ を a の登録者が準備するが，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 $\varepsilon_{x,0}^{(u)}, \ldots, \varepsilon_{x,k-1}^{(u)}$ 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．

以上によって，$n > k$ であればサーバが故障しても k 台のサーバが無事であれば，無事なサーバが**図 3** の 3. の処

Protocol 3.2: 変換用乱数組生成処理

1. k 個の乱数 $\varepsilon_{x,0}^{(u)},\ldots,\varepsilon_{x,k-1}^{(u)}$（$x=1,\ldots,m$，$u=1,\ldots,$ 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. $\varepsilon_x^{(u)}$ と $\varepsilon_{x,0}^{(u)},\ldots,\varepsilon_{x,,k-1}^{(u)}$ を n 台のサーバ S_i（$i=0,\ldots,n-1$）に秘密分散する．サーバ S_i は以下を持つ．

$$(\overline{[\varepsilon_x^{(u)}]_i},\overline{[\varepsilon_{x,0}^{(u)}]_i},\ldots,\overline{[\varepsilon_{x,k-1}^{(u)}]_i})$$

3. 定められた k 台のサーバ S_j（$j=0,\ldots,k-1$）は $\overline{[\varepsilon_{x,j}^{(u)}]_i}$ を k 個集めて $\varepsilon_{x,j}^{(u)}$ を復元し，変換用乱数組として以下を保持する．

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

図 3 変換用乱数組生成処理のプロトコル
Fig. 3 Protocol for generating *conversion* random numbers.

理を行うことによって変換用乱数組を復元できる．

ただし，文献 [28] の完全一致検索の具体的アルゴリズムはサーバ間の通信が多く，効率的でない．そこで，文献 [28] を改良し高速化した完全一致検索のアルゴリズムを以下では用いる．

4. 提案方式

4.1 部分一致検索の概要

完全一致検索では ${c_i}^2$（$i=1,\ldots,g$）がすべて 0 であるかどうかを検証した．これは $C=[{c_1}^2,\ldots,{c_g}^2]$ をベクトルと見なしたとき，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．

$$[1,\ldots 1]C^T=0 \tag{1}$$

${c_i}^2$ 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，$[{c_1}^2,\ldots,{c_g}^2]$ を誤り訂正符号（ここではリードソロモン符号を想定する）における受信語と見なしたとき，$[0,\ldots,0]$ 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 1 カ所のとき以下を計算して，1 重誤り訂正をすれば C'^T を C^T に戻すことができる．C'^T は 1 カ所が 0 でない誤りを含むベクトルの転置とする．

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

式 (2) を t 重誤りのパリティ検査行列を用いたシンドローム計算処理とすれば t 個の誤り（不一致）に対応できる．

よって，本論文では以下のようなシステムを考える．まず，複数の文書を持つ登録者が後述する【ドキュメントの秘匿化処理】によって n 台のサーバに各文書を秘匿して登録しており，検索したいキーワードを持つ検索者が後述する【検索クエリ生成処理】によってそのキーワードを検索クエリとして秘匿して前記 n 台のサーバに入力する．n 台のサーバは検索クエリとして入力されたキーワードをそのまま含む文書があれば検索がヒットしたとする【完全一致検索処理】と，g 文字からなるキーワードのうち，あらかじめ決められた t 箇所が不一致であっても，$g-t$ カ所が一致していれば検索がヒットしたと見なす部分一致検索を【誤り訂正処理】によって実現し，ヒットした文章を検索者に返す．

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登録者が文章 a を秘匿して登録し，検索者が検索クエリ b を含む文章を検索する場合を示す．基本的に登録者と検索者は同一とするが，システムからの攻撃を考える．登録された文章中の文字 a_x（$x=1,\ldots,m$）と検索クエリ中の文字 b_y（$y=1,\ldots,g$）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】および【完全一致検索処理】で生成する乱数 $\alpha_{x,j}$，$\beta_{y,j}$，$\kappa_{h,j}$ 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）．p は完全一致検索の場合と同様に $gq^2<p$ である．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，各サーバは前述の変換用乱数組 $(\overline{[\varepsilon_x^{(u)}]_i},\varepsilon_{x,i}^{(u)})$ が登録者によって準備されているとする．

なお，以下では $x=1,\ldots,m$，$y=1,\ldots,g$，$i=0,\ldots,n-1$，$j=0,\ldots,k-1$ とする．また，簡単のために検索文字列の 1 文字が異なる場合の部分一致検索，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．

以下では簡単のため，$n=k$ とする．この場合，i と j は区別する必要はなく，3.2 節の変換用乱数生成処理において $\varepsilon_{x,i}^{(u)}$ は秘密分散されず，$[\varepsilon_x^{(u)}]_i$ として直接サーバ S_i に送られる．

【記号定義】

- a：m 個の文字 a_x（$x=1,\ldots,m$）からなる文章．
- $[a_x]_i$：サーバ S_i が保持する文字 a_x 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，$[a]_i=[a_1]_i,\ldots,[a_m]_i$．
- b：g 個の文字 b_y（$y=1,\ldots,g$）からなる検索クエリ．
- $[b_x]_i$：サーバ S_i が保持する文字 b_x 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，$[b]_i=[b_1]_i,\ldots,[b_g]_i$．

【ドキュメントの秘匿化処理】

- 入力：a_x（$x=1,\ldots,m$）
- 出力：$[a]_j=[a_1]_j,\ldots,[a_m]_j$（$j=0,\ldots,k-1$）

1. 登録者は前述の【変換用乱数組生成処理】を実行し，以下の変換用乱数組を事前に計算し，サーバ S_i に送

信する．

$$[\varepsilon_x^{(1)}]_i,\ [\varepsilon_x^{(2)}]_i,\ [\varepsilon_x^{(3)}]_i \quad (x=1,\ldots,m)$$

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

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

3. 登録者は計算した α_x と (a_x+1) をかけて $\alpha_x(a_x+1)$ を計算し，以下の $[a_x]_j$ をサーバ S_j に送る．

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

4. サーバ S_j は以下の情報を保持する．

$$[\varepsilon_x^{(1)}]_j=(\overline{[\varepsilon_x^{(1)}]_j},\varepsilon_{x,j}^{(1)}),$$
$$[\varepsilon_x^{(2)}]_j=(\overline{[\varepsilon_x^{(2)}]_j},\varepsilon_{x,j}^{(2)}),$$
$$[\varepsilon_x^{(3)}]_j=(\overline{[\varepsilon_x^{(3)}]_j},\varepsilon_{x,j}^{(3)}),$$
$$[a]_j=[a_1]_j,\ldots,[a_m]_j$$

【検索クエリ生成処理】

- 入力：b_y（$y=1,\ldots,g$）
- 出力：$[b]_j=[b_1]_j,\ldots,[b_g]_j$（$j=0,\ldots,k-1$）

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

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

2. 検索者は計算した β_y と (b_y+1) をかけて，以下の $[b_y]_j$ をサーバ S_j に送る．

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

検索処理 1：【完全一致検索処理】

- 入力：$[b]_j$，$[a]_j$（$j=0,\ldots,k-1$）
- 出力：a, or $\perp$

1. k 台のサーバ S_j は $h=0$ 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，$y=1,\ldots,g$，$v=h+y$ とする．

(1) サーバ S_j は乱数 $\kappa_{h,j}$ を生成し，以下を計算して k 台中の 1 台のサーバに送る（ここではサーバ S_0 とする）．ただし，$h=0,\ldots,m-g$．

$$\frac{\kappa_{h,j}}{(\alpha_{v,j})^2\varepsilon_{v,j}^{(1)}},\quad \frac{\kappa_{h,j}}{(\beta_{y,j})^2\varepsilon_{v,j}^{(2)}},\quad \frac{\kappa_{h,j}}{\alpha_{v,j}\beta_{y,j}\varepsilon_{v,j}^{(3)}}$$

(2) サーバ S_0 は以下を計算し，全サーバに送信する．

$$\frac{\kappa_h}{{\alpha_v}^2\varepsilon_v^{(1)}}=\prod_{j=0}^{k-1}\frac{\kappa_{h,j}}{(\alpha_{v,j})^2\varepsilon_{v,j}^{(1)}},$$
$$\frac{\kappa_h}{{\beta_y}^2\varepsilon_v^{(2)}}=\prod_{j=0}^{k-1}\frac{\kappa_{h,j}}{(\beta_{y,j})^2\varepsilon_{v,j}^{(2)}},$$
$$\frac{\kappa_h}{\alpha_v\beta_y\varepsilon_v^{(3)}}=\prod_{j=0}^{k-1}\frac{\kappa_{h,j}}{\alpha_{v,j}\beta_{y,j}\varepsilon_{v,j}^{(3)}}$$

(3) サーバ S_i は以下を計算し，サーバ S_0 に送る．ただし，$h=0,\ldots,m-g$．

$$\sum_{y=1}^{g}\overline{[\kappa_h(a_v-b_y)^2]}_i$$
$$=\sum_{y=1}^{g}\Bigg[(\alpha_v(a_v+1))^2\times\frac{\kappa_h}{{\alpha_v}^2\varepsilon_v^{(1)}}\times\overline{[\varepsilon_v^{(1)}]}_i$$
$$+(\beta_y(b_y+1))^2\times\frac{\kappa_h}{{\beta_y}^2\varepsilon_v^{(2)}}\times\overline{[\varepsilon_v^{(2)}]}_i$$
$$-2\times\alpha_v(a_v+1)\times\beta_y(b_y+1)$$
$$\times\frac{\kappa_h}{\alpha_v\beta_y\varepsilon_v^{(3)}}\times\overline{[\varepsilon_v^{(3)}]}_i\Bigg]$$

(4) サーバ S_0 は送られた秘匿計算結果の分散値 $\sum_{y=1}^{g}\overline{[\kappa_h(a_v-b_y)^2]}_i$ から $\sum_{y=1}^{g}\kappa_h(a_v-b_y)^2$ を復元する．

2. サーバ S_j は $\sum_{y=1}^{g}\kappa_h(a_v-b_y)^2=0$ であれば一致として，$[a]_j$ 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．

検索処理 2：【誤り訂正処理】

1. サーバ S_i は完全一致検索で用いた変数を用いて以下を計算し，サーバ S_0 に送る．ただし，$h=0,\ldots,m-g$．

$$\sum_{y=1}^{g}y\overline{[\kappa_h(a_v-b_y)^2]}_i$$
$$=\sum_{y=1}^{g}y\Bigg[(\alpha_v(a_v+1))^2\times\frac{\kappa_h}{{\alpha_v}^2\varepsilon_v^{(1)}}\times\overline{[\varepsilon_v^{(1)}]}_i$$
$$+(\beta_y(b_y+1))^2\times\frac{\kappa_h}{{\beta_y}^2\varepsilon_v^{(2)}}\times\overline{[\varepsilon_v^{(2)}]}_i$$
$$-2\times\alpha_v(a_v+1)\times\beta_y(b_y+1)$$
$$\times\frac{\kappa_h}{\alpha_v\beta_y\varepsilon_v^{(3)}}\times\overline{[\varepsilon_v^{(3)}]}_i\Bigg]$$

2. サーバ S_0 は以下を復元して全サーバに送る．

$$s1=\sum_{y=1}^{g}\kappa_h(a_v-b_y)^2$$
$$s2=y\sum_{y=1}^{g}\kappa_h(a_v-b_y)^2$$

3. サーバは $y' = s2/s1$ を計算し，y' が検索クエリの範囲（$1,\ldots,g$）に入っていなければ誤訂正として 5. に移る．入っていれば 4. に行く．
4. サーバは y' の位置にあたる文字を除いて $s3 = \sum_{y=1(y\neq y')}^{g} \overline{[\kappa_h(a_v - b_y)^2]}_j$ を計算して S_0 に集めて復元し，0 であれば $[a]_j$ を検索者に送信する．0 でなければ誤訂正として 5. に移る．
5. 一致した文章がなければ一致なしとする．

5. 考察および評価

5.1 提案方式の安全性

(1) 完全一致検索の安全性

以降では，共通鍵暗号を用いた検索可能暗号との比較を考える．一般に，共通鍵暗号を用いた検索可能暗号では秘密情報の登録者と検索者は同一人物であるので，検索者に対する安全性は考慮されず，システムのみを攻撃者としてその安全性を評価する．

秘密分散法におけるシステムからの攻撃としては n 台のサーバ中，r（$< k$）台のサーバが攻撃者に乗っ取られるなどして，その情報を攻撃者が利用できる状況を考え，攻撃者が登録された文章 a の各文字列 a_x，または検索クエリ b の文字列 b_y（$y = 1,\ldots,g$）を知ることができたとき，攻撃成功と見なす．ただし，本論文では semi-honest な攻撃者を仮定する．

以下に，完全一致検索に対する安全性を示す．

攻撃者は r 台のサーバ情報を知ることができるので，以下の情報を知ることができる．

$$[\varepsilon_x^{(1)}]_j,\ [\varepsilon_x^{(2)}]_j,\ [\varepsilon_x^{(3)}]_j,\ [b]_j,\ [a]_j,\ \kappa_{h,j} \quad (j = 0,\ldots,r-1)$$

また，【完全一致検索処理】の (1) より，以下も知る．

$$\frac{\kappa_{h,j}}{(\alpha_{v,j})^2\varepsilon_{v,j}^{(1)}},\ \frac{\kappa_{h,j}}{(\beta_{y,j})^2\varepsilon_{v,j}^{(2)}},\ \frac{\kappa_{h,j}}{\alpha_{v,j}\beta_{y,j}\varepsilon_{v,j}^{(3)}} \quad (j = 0,\ldots,k-1)$$

しかし，攻撃者は $\kappa_{h,j}/(\alpha_{v,j})^2\varepsilon_{v,j}^{(1)}$，$\kappa_{h,j}/(\beta_{y,j})^2\varepsilon_{v,j}^{(2)}$，$\kappa_{h,j}/\alpha_{v,j}\beta_{y,j}\varepsilon_{v,j}^{(3)}$（$j = r,\ldots,k-1$）を分解することができないので，$\alpha_x$，$\beta_y$，$\varepsilon_x^{(1)}$，$\varepsilon_x^{(2)}$，$\varepsilon_x^{(3)}$，$\kappa_{h,j}$（$j = r,\ldots,k-1$）を知ることができない．よって，攻撃者は秘密情報 a_x と検索クエリ b_y に関する情報を知ることができない．

また，攻撃者は (4) から $\sum_{y=1}^{g}\kappa_h(a_v - b_y)^2$ が 0 かそうでないかを知る．しかし，攻撃者は κ_h を知らず，$\sum_{y=1}^{g}\kappa_h(a_v - b_y)^2$ を分解できないため，$\sum_{y=1}^{g}\kappa_h(a_v - b_y)^2$ を知っても秘密情報 a_x と検索クエリ b_y に関する情報を得ることはできない．

以上の攻撃者が知る情報を L とすると，以下が成り立つ．よって，完全一致検索は情報理論的安全性を持つことがいえる．

$$H(a_x) = H(a_x|L),$$
$$H(b_y) = H(b_y|L)$$

また，攻撃者が検索者に成りすまして攻撃することも考えられるが，登録者はあらかじめパスワードを秘密分散して登録し，検索者はパスワードと検索クエリを入力して，パスワードが完全一致した場合に，4.2 節に示す検索を行えばよい．ただし，パスワードは必要に応じて更新されるとする．これによって，パスワードを知らない攻撃者は偶然の一致を除き，何度検索しても検索文字列や文書 a を得ることはできない．

(2) 部分一致検索の安全性

部分一致検索は完全一致検索で用いた値を用いて計算されるため，$s2$ 以外に新たに得られる情報はない．$s1$ は誤りパターンの秘匿値（乱数 κ_h によって a_v と b_y の差分は秘匿される）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. および 4. によって誤訂正は防止される．また，t 誤り訂正の場合 $2t$ 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，$t+1$ 個以上の誤り位置の検出は誤りパターンなどの $2t+2$ 個以上の未知数を含むため特定不能）．誤りが t 個以下の場合，$2t$ 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い，誤り数値の特定は行わない．これは誤り数値を特定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分からないためである．ただし，設定範囲内の誤りであった場合，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】では誤りの有無と，設定した範囲の誤り位置は知られるが，秘密情報や検索文字列に関する情報は漏洩しない．

(3) 共通鍵暗号を用いる検索可能暗号との比較

共通鍵暗号を用いた検索可能暗号では安全性を検討する際にドキュメントの追加および検索を行った際に必ず流出する情報を定義し，それ以上の情報が流出しない方式を安全としている場合が多い．本論文では，共通鍵暗号を用いた検索可能暗号と提案方式の流出情報を比較するために，Curtmola ら [7] によって定義された共通鍵暗号を用いた検索可能暗号で流出することが多い以下の情報について検討する．

- **情報 1**：ドキュメント a の長さ m

ドキュメント a の秘匿情報 $[a]_j$ をサーバに登録すると，サーバは $[a]_j = [a_1]_j,\ldots,[a_m]_j$ から登録されたドキュメントの長さを知ることができる．ただし，システムは 5.1 節に示すように文字 a_x，b_y を区別することができないため，

ドキュメントや検索クエリの内容は漏洩しない．

- **情報 2**：ある検索クエリ $[b]_i$ で一致判定を行った際に出力される検索結果

検索クエリ $[b]_i$ で一致判定を行うと，システムはその判定結果を知る．また，提案方式ではある検索クエリに対して 0 が復元されると，システムはそれに対応するドキュメントを検索者に送信する．よって，システムはある検索クエリに対する検索結果とそれを含むドキュメントを知る．

- **情報 3**：検索者が i 回目に検索した文字列と j 回目に検索した文字列が同じか否か

共通鍵暗号を用いる検索可能暗号では，検索クエリが確定的になる場合が多く，検索クエリが同じであれば検索文字列も同じと分かる場合が多い．提案方式では，同じ検索文字列でも，【検索クエリ生成処理】の際に異なる乱数を用いて検索クエリを生成すれば，異なる検索クエリが生成されるため，検索文字列が同じかどうかは漏洩しない．

- **情報 4**：Forward privacy を満たすかどうか

Forward privacy とは，ユーザがキーワード t に関する検索を行ったことがあるという条件の下で，キーワード t を含むドキュメントを追加したときに，システムが過去に使用された検索クエリなどを用いて，追加されたドキュメントがキーワード t を含むか判別できるかどうかという問題である [18], [19], [31]．文献 [31] では，システムがキーワード t を含むか判別できないとき，その方式は forward privacy を満たすとしている．提案方式では文章中の同じ単語でも，【ドキュメントの秘匿化処理】において異なる乱数を用いて秘匿すれば異なる表現になる．よって，システムは追加された文章に過去に検索されたキーワードを含むか判定できない．よって，提案方式は forward privacy を満たすといえる．

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 g 個まで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（$1 \leq t \leq g$）．

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

式 (3) の行列はパリティ検査行列である．t や g が大きくなると，式 (3) に必要な計算量は t や g に比例して増える．誤り訂正処理に対しては CD や DVD などに用いられている効率的な多重誤り訂正手法などを用いれば対応できる．

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 はその短縮化符号と見なせる．ただし，C の前には符号長から g を引いた長さの 0 の情報系列があると見なす．また，手順 3. において検出された誤り位置が長さ g の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．

また，提案方式では誤り訂正符号化は行わないが，文字間の差分がすべて 0 である場合のみ正しいので，それを符号語と見なす．また一般に，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．または，まず 1 重誤り訂正を設定して部分一致検索を行い，ヒットしない場合 2 重誤り訂正を試みるなどすることもできる．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．したがって，提案方式は設定した任意の数の誤りに対応できる．

また，4.2 節では分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，同時に行う方が効率的である．すなわち，【完全一致検索処理】の手順 (3) で $\sum_{y=1}^{g} \overline{[\kappa_h(a_v - b_y)^2]}_j$ と $\sum_{y=1}^{g} y\overline{[\kappa_h(a_v - b_y)^2]}_j$ を計算し，次に【誤り訂正処理】の手順 2. を行う．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分かる．そうでない場合，【誤り訂正処理】の手順 3.，4. を行い 1 重誤り訂正を行う．これは t 重誤り訂正の場合にも拡張できる．

5.3 インデックス方式への対応

提案方式はインデックスを用いず秘匿文書からの直接検索が可能であるが，インデックス方式にも対応できる．たとえば，1 重誤り訂正を設定し，abc をキーワードとする場合，abc を【ドキュメントの秘匿化処理】によって秘密分散したものをインデックスとして登録する．【検索クエリ生成処理】によって xbc，axc，abx（x は任意の文字）などの検索用クエリを入力しても誤り訂正によって abc が検索できる．以上より，提案方式は 1 つのインデックスを登録するだけでよく，2 章に示した従来方式よりも効率的に部分一致検索できることがいえる．

また，文章中の代表的なキーワードのみをインデックスとし，インデックス検索をした後にキーワードが一致しない場合，4 章に示した秘匿文章からの直接検索を行うハイブリッド的な検索も行うことができる．この場合，インデックスの数を少なくして代表的なキーワードのみとし，インデックスで検索できた場合，必要な計算量を大きく削減できる．また，インデックスが一致せず，直接検索まで行っても，最初から直接検索を行った場合と比べてインデックス検索分の計算量が少量増えるだけとなる．また，出現頻度の高いキーワードはインデックス検索し，低出現頻度のキーワードは直接検索するという組合せも考えられる．

さらに，提案方式は AND・OR 検索も可能であるので，

複数のインデックスを 1 度の検索処理で処理することも可能である．たとえば，簡単のため論理積（AND）検索の文字数を g とする w 種類の異なる検索クエリで論理和（OR）検索を行う場合，a_x（$x = 1, \ldots, m$）を登録されたドキュメントの文字列，$b_{l,y}$（$l = 1, \ldots, w$，$y = 1, \ldots, g$）をそれぞれ w 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，$\sum(a_y - b_{l,}y)^2 = \sum {c_{l,y}}^2$ を秘匿計算すればよい．さらに，w 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には論理和（OR）検索を実現すればよい．よって，$c_{1,y}, \ldots, c_{w,y}$ のうち 1 つでも 0 のとき，$f(c_{1,y}, \ldots, c_{w,y}) = 0$ となる演算 f は，$\overline{\overline{c_{1,y}} + \cdots + \overline{c_{w,y}}}$ となる．ここでは，$\overline{\overline{c_{1,y}} + \cdots + \overline{c_{w,y}}} = c_{1,y} \cdots c_{w,y}$ であるので，複数検索クエリに対する論理和（OR）検索には，$\prod\{\sum(a_y - b_{l,y})^2\} = \prod\left(\sum c_{l,y}^2\right)$ を秘匿計算すればよい．

5.4 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

文章の秘匿以外に何もしなくても設定範囲以内の不一致に対応できる性質を柔軟性と呼ぶ．

従来方式は事前にインデックス [21]，ブルームフィルタ [15]，接尾辞オートマトン [22]，モノグラムセット [25] などを秘匿した文章の他に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，提案方式は直接検索の場合，秘密分散法によって文章の秘匿を行うだけで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 つでも含めば，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いえる．

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．

たとえば，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，従来方式では登録していない文字を含むと検索できないため，“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いえる．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場合が多く，“a” と “c” を同時に含む文章がすべて検索され，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 文字を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．設定する誤りの数は 5.2 節に示すように事前に準備をすることなく，任意に変更可能である．また，ある検索文字列について検索したとき，多くの文章が検索された場合，提案方式は前述のように 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擬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いえる．

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，インデックス方式において柔軟性を高くしたい場合，文章中のすべ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で文書から直接検索を行う場合，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．また，インデックスを登録する場合，インデックスを別に登録する必要があるが，登録するインデックスは完全一致検索用のインデックスのみでよく，キーワードを派生させた複数のインデックスを登録する必要はなく，計算量は少ないといえる．

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，誤り訂正能力は 1 重訂正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．秘匿検索は秘密分散を用いた秘匿計算によって実現されるが，秘密分散による秘匿計算を高速に行う場合，サーバ間をつなぐ高速通信網が必要とされる．提案方式は $n = k = 2$ を選択できるので，2 台のサーバが高速な通信網でつながれていれば実現可能である．ただし，具体的な実装による処理時間の比較については今後の課題である．

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いえる．

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．たとえば，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] Shamir, A.: How to share a secret, *Commun. ACM*, Vol.22, No.11, pp.612–613 (1979).

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

[3] Boneh, D., Crescenzo, G.D., Ostrovsky, R. and Persiano, G.: Public key encryption with keyword search, *Advances in Cryptology–EUROCRYPT 2004*, *LNCS*, Vol.3027, pp.506–522 (2004).

[4] Golle, P., Staddon, J. and Waters, B.: Secure conjunctive keyword search over encrypted data, *Applied Cryptography and Network Security–ACNS 2004*, *LNCS*, Vol.3089, pp.31–45 (2004).

[5] Park, D.J., Kim, K. and Lee, P.J.: Public key encryption with conjunctive field keyword search, *Information Security Applications–WISA 2004*, *LNCS*, Vol.3325, pp.73–86 (2004).

[6] Chang, Y.C. and Mitzenmacher, M.: Privacy preserving keyword searches on remote encrypted data, *Applied Cryptography and Network Security–ACNS 2005*, *LNCS*, Vol.3531, pp.442–455 (2005).

[7] Curtmola, R., Garay, J., Kamara, S. and Ostrovsky, R.: Searchable symmetric encryption: Improved definitions and efficient constructions, *Proc. 13th ACM Conference on Computer and Communications Security*, pp.79–88 (2006).

[8] Byun, J.W., Lee, D.H. and Lim, J.: Efficient conjunctive keyword search on encrypted data storage system, *Public Key Infrastructure–EuroPKI 2006*, *LNCS*, Vol.4043, pp.184–196 (2006).

[9] Wang, P., Wang, H. and Pieprzyk, J.: Keyword field-free conjunctive keyword searches on encrypted data and extension for dynamic groups, *Cryptology and Network Security–CANS 2008*, *LNCS*, Vol.5339, pp.178–195 (2008).

[10] Kurihara, J., Kiyomoto, S., Fukushima, K. and Tanaka, T.: A new (k,n)-threshold secret sharing scheme and its extension, *Information Security–ISC 2008*, *LNCS*, Vol.5222, pp.455–470 (2008).

[11] Zhang, B. and Zhang, F.: An efficient public key encryption with conjunctive subset keywords search, *Journal of Network and Computer Applications*, Vol.34, No.1, pp.262–267 (2011).

[12] Kurosawa, K. and Ohtaki, Y.: UC-secure searchable symmetric encryption, *Financial Cryptography and Data Security–FC 2012*, *LNCS*, Vol.7397, pp.285–298 (2012).

[13] 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．マルチメディア，分散協調とモバイルシンポジウム 2013 論文集，Vol.2013, pp.2067–2074 (2013).

[14] Cash, D., Jarecki, S., Jutla, C., Krawczyk, H., Rosu, M.C. and Steiner, M.: Highly-scalable searchable symmetric encryption with support for boolean queries, *Advances in Cryptology–CRYPTO 2013*, *LNCS*, Vol.8042, pp.353–373 (2013).

[15] Suga, T., Nishide, T. and Sakurai, K.: Character-based symmetric searchable encryption and its implementation and experiment on mobile devices, *Security and Communication Networks*, Vol.9, No.12, pp.1717–1725 (2013).

[16] Kurosawa, K.: Garbled searchable symmetric encryption, *Financial Cryptography and Data Security–FC 2014*, *LNCS*, Vol.8437, pp.234–251 (2014).

[17] Song, C., Liu, X. and Yan, Y.: Efficient public key encryption with field-free conjunctive keywords search, *Trusted Systems–INTRUST 2014*, *LNCS*, Vol.9473, pp.394–406 (2014).

[18] Yang, Y., Li, H., Liu, W., Yao, H. and Wen, M.: Secure dynamic searchable symmetric encryption with constant document update cost, *Proc. 2014 IEEE Global Communications Conference*, pp.775–780 (2014).

[19] 佐藤尋時：Forward privacy を考慮した動的な検索可能暗号，2015 年暗号と情報セキュリティシンポジウム (2015).

[20] 堀 史明，岸本 渡：委任検索可能公開鍵暗号のマルチユーザ拡張，2016 年暗号と情報セキュリティシンポジウム (2016).

[21] 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 年暗号と情報セキュリティシンポジウム (2016).

[22] 山本博章，宮崎 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 年暗号と情報セキュリティシンポジウム (2016).

[23] Yang, Y. and Ma, M.: Conjunctive keyword search with designated tester and timing enabled proxy re-encryption function for e-health clouds, *IEEE Trans. Information Forensics and Security*, Vol.11, No.4, pp.746–759 (2016).

[24] Mohd Kamal, A.A.A., Iwamura, K. and Kang, H.: Searchable encryption of image based on secret sharing scheme, *Proc. 2017 Asia-Pacific Signal and Information Processing Association Annual Summit and Conference* (*APSIPA ASC*), pp.1495–1503 (2017).

[25] Manazir Ahsan, M.A., Chowdhury, F.Z., Sabilah, M., Bin Abdul Wahab, A.W. and Idris, M.Y.I.B.: An efficient fuzzy keyword matching technique for searching through encrypted cloud data, *Proc. 2017 International Conference on Research and Innovation in Information Systems* (*ICRIIS*), pp.1–5 (2017).

[26] Sun, S., Yuan, X., Liu, J.K., Steinfeld, R., Sakzad, A., Vo, V. and Nepal, S.: Practical backward-secure searchable encryption from symmetric puncturable encryption, *Proc. 2018 ACM SIGSAC Conference on Computer and Communications Security* (*CCS 2018*), pp.763–780 (2018).

[27] Xu, P., Tang, S., Xu, P., Wu, Q., Hu, H. and Susilo, W.: Practical multi-keyword and Boolean search over encrypted e-mail in cloud server, *IEEE Trans. Services Computing*, Vol.14, No.6, pp.1877–1889 (2019).

[28] Mohd Kamal, A.A.A. and Iwamura, K.: Searchable encryption using secret sharing scheme that realizes direct search of encrypted documents and disjunctive search of multiple keywords, *Journal of Information Security and Applications*, Vol.59, 102824 (June 2021) .

[29] Li, J., Huang, Y., Wei, Y., Lv, S., Liu, Z., Dong, C. and Lou, W: Searchable symmetric encryption with forward search privacy, *IEEE Trans. Dependable and Secure Computing*, Vol.18, No.1, pp.460–474 (2021).

[30] Liu, X., Yang, G., Mu, Y. and Deng, R.H.: Multi-user verifiable searchable symmetric encryption for cloud storage, *IEEE Trans. Dependable and Secure Computing*, Vol.17, No.6, pp.1322–1332 (2020).

[31] Bost, R.: $\Sigma o \varphi o \varsigma$: Forward secure searchable encryption, *Proc. 2016 ACM SIGSAC Conference on Computer and Communications Security* (*CCS 2016*), pp.1143–1154 (2016).

岩村 恵市 （正会員）

1958 年生．1980 年九州大学工学部情報工学科卒業．1982 年同大学大学院情報工学研究科修士課程修了．同年キヤノン（株）入社．1994 年東京大学博士（工学）．現在，東京理科大学工学部電気工学科教授．主に符号理論，並列処理，情報セキュリティ，電子透かしの研究に従事．IEEE，電子情報通信学会，電気学会各会員．エンリッチド・マルチメディア（EMM）研究会元委員長，情報ハイディングおよびその評価基準（IHC）研究会元委員長，電子情報通信学会フェロー，本会フェロー．

ムハンマド カマル アフマド アクマル アミヌディン
（学生会員）

2017 年東京理科大学工学部電気工学科卒業．2019 年同大学大学院工学研究科電気工学専攻修士課程修了．同年同大学院工学研究科電気工学専攻博士後期課程に入学．現在，情報セキュリティに関する研究に従事．