Vol.2021-DPS-186 No.14
Vol.2021-CSEC-92 No.14
2021/3/15

確率的暗号ベースの検索可能暗号を用いた全文検索の高速化検討

森 郁海[†1, †2]　平野 貴人[†1]　中村 嘉隆[†2]　稲村 浩[†2]

概要：クラウドコンピューティングの普及に伴い，インターネット上に暗号化したデータを保管し，そのデータを検索する全文検索システムのニーズが増えている．全文検索において，暗号化データに紐づく平文のキーワードを検索に使用すると，キーワードから暗号化されたデータがどのようなものか推測されるおそれがある．そこで，キーワードを暗号化して検索を行う検索可能暗号の利用が考えられるが，検索速度が遅いという問題がある．その解決策として，キーワードのハッシュ値の一部（開示ビット）をクラウドに開示し，クラウドが開示ビットに基づいて検索空間を絞り込む高速化手法がある．この方法は，開示ビットの長さ（開示ビット長）が長いほど大きい高速化効果が得られるが，一方で平文のキーワードの頻度分布と開示ビットの頻度分布が近づきキーワードが特定される危険性があるため，適切な開示ビット長を決める必要がある．本稿では，平文のキーワードの頻度分布を隠しつつ，十分な高速化効果が得られる開示ビット長を，最小エントロピーと *k*-匿名性を用いて求める方法を提案する．さらに，データベースの分散化により，スケーラビリティを強化する．既存の検索可能暗号を用いた全文検索システムに提案方式を適用して評価した結果，文書ファイル数が 1,000 の場合で検索時間を最大 97.2%削減できることを確認した．

A study on Acceleration for Full-text Search using Probabilistic Searchable Encryption

IKUMI MORI[†1,†2]　TAKATO HIRANO[†1]　YOSHITAKA NAKAMURA[†2]
HIROSHI INAMURA[†2]

1. はじめに

チャットツールである Slack[1]やクラウドファイルストレージである Google ドライブ[2]のような情報共有ツールを利用する企業が増加している．情報共有ツールには多数の電子データが保管されており，それらの電子データの利活用を目的に，情報検索の需要が高まっている．情報検索システムをオンプレミスで運用すると RASIS[a)]の確保に多大なコストが生じるため，運用負荷低減とサービス導入の容易性からクラウドサービスを利用することが多い. 通常，クラウドにアップロードする社内の電子データやそのメタデータは，セキュリティ上，全て暗号化すべきであるが，暗号化を施すとデータの検索が困難となる．

性能面では，たとえば，ある開発プロジェクトの文書ファイル（以降，文書と表記）に対する情報検索をユースケースに想定すると，検索対象の文書数は最低でも 1,000 件規模で，主に検索処理の即時性を要求すると考えられる．

これらのことから，クラウド上で社内向けの情報検索を実現するためには，以下の要件が挙げられる．

要件(1) 暗号化したまま文書の検索できること
要件(2) 文書数 1,000 件での検索時間が十分短いこと

情報検索システムには，全文検索と呼ばれる複数の文書から特定の文字列を検索する技術が使用されていることが多い．全文検索の処理手順を以下に示す．図 1 は，処理手順を図示したものである．図中の番号は，手順番号と対応する．

文書登録

(1) クライアントは，文書をサーバに送信する．
(2) サーバは，文書を形態素解析し，登録キーワードを抽出する．登録キーワードと近い意味合いの検索キーワードでも文書がヒットするよう，キーワードの類義語や同義語を求める類義語展開を行う場合もある．
(3) サーバは，登録キーワードと文書の ID を関連付けた転置索引と呼ばれる表を作成する．

検索

(4) クライアントは，検索文をサーバに送信する．
(5) サーバは，検索文に対し形態素解析と必要に応じ類義語展開を行い，検索キーワードを抽出する．
(6) 転置索引からこの検索キーワードと一致するレコードを求め，文書 ID を特定する．
(7) 文書 ID から文書を特定し，文書をクライアントに送信する．

†1 三菱電機株式会社情報技術総合研究所
Information Technology R&D Center, Mitsubishi Electric Corporation, Kamakura, Kanagawa 247-8501, Japan
†2 公立はこだて未来大学
Future University Hakodate, Hakodate, Hokkaido 041-8655, Japan

a) Reliability（信頼性），Availability（可用性），Serviceability（保守性），Integrity（完全性），Security（機密性）

Vol.2021-DPS-186 No.14
Vol.2021-CSEC-92 No.14
2021/3/15

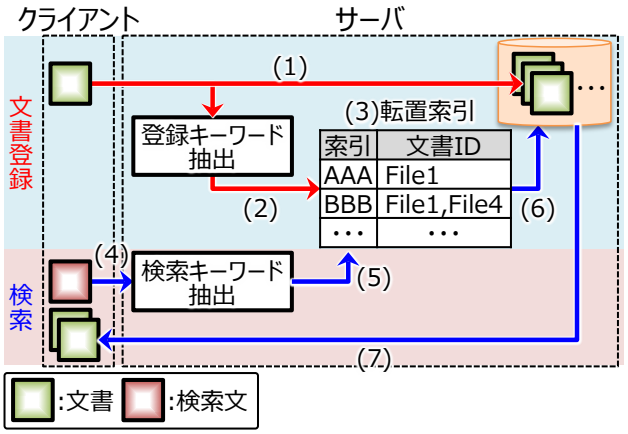

図 1 全文検索の処理手順

以上のように，全文検索では転置索引を作成するため，平文のキーワードがサーバに保存される．その結果，悪意のあるクラウド管理者やクラウド内に侵入したマルウェアによってキーワードが取得され，機密情報が流出するおそれがある．したがって，要件(1)に示したように，悪意のあるクラウド管理者から文書を保護するために，キーワードを暗号化した上で検索することが求められる．

データを暗号化したまま検索を行う技術として，検索可能暗号がある[3][4][5]．検索可能暗号の処理手順を以下に示す．図 2 は，処理手順を図示したものである．図中の番号は，処理手順番号と対応する．

文書登録

(1) クライアントは，文書から登録キーワードを抽出し，登録キーワードを暗号化鍵で暗号化して検索用のタグ（以降，タグと表記）を作成する．並行して，文書を暗号化鍵で暗号化する．暗号化した文書とタグをサーバに送信する．なお，キーワードの暗号化鍵と文書の暗号化鍵は異なっていてもよい．

(2) サーバは，暗号化済みの文書とタグを関連付けてデータベース（以降，DB と表記）に保管する．

検索

(3) クライアントは，検索文から検索キーワードを抽出し，検索キーワードを暗号化鍵で暗号化して検索語（以降，トラップドアと表記）を作成する．このトラップドアをサーバに送信する．

(4) サーバは，トラップドアと DB 内のタグを特殊な演算を用いて網羅的に比較する．この演算は，トラップドアとタグを入力すると，1（：＝一致）または 1 以外（：＝不一致）を出力するため，トラップドアとタグに対応する暗号化前のキーワードが等しいかどうかを判定できる．

(5) サーバは，トラップドアと一致したタグに関連付く暗号化済みの文書をクライアントに返却する．

(6) クライアントは，復号鍵で文書を復号する．

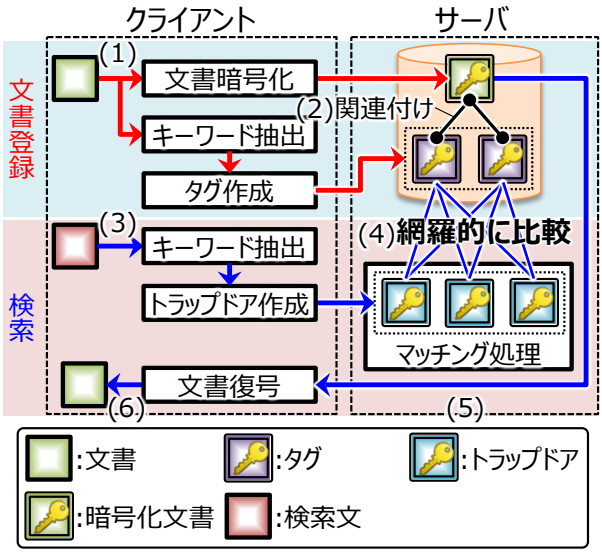

図 2 検索可能暗号の処理手順

表 1 主な検索可能暗号の方式と利点・欠点

	確定的暗号ベース	確率的暗号ベース	ハイブリッド
代表例	Bellare[3]ら	Song ら[4], Boneh ら[5]	平野ら[6]
利点	処理が高速	頻度分析攻撃に強い	処理が高速，頻度分析攻撃に強い
欠点	頻度分析攻撃に弱い	処理が低速	パラメータ調整が必要

検索可能暗号は，確定的暗号ベースの方式[3]，確率的暗号ベースの方式[4][5]，ハイブリッド方式[6]がある（表 1）．

- **確定的暗号ベースの方式**：図 3 上段に示すように，同一の入力からは同一の出力が得られる暗号方式を用いて，タグとトラップドアを作成する．同一のキーワードから常に同一のタグとトラップドアが生成されるため，検索時のマッチング処理はタグとトラップドアのバイナリ一致判定を行うだけでよい．サーバの DB テーブルのレコード数nに対して$O(\log n)$で検索できる．
- **確率的暗号ベースの方式**：図 3 下段に示すように，同一の入力でも異なる出力が得られる暗号方式を用いて，タグとトラップドアを作成する．検索時のマッチング処理ではタグとトラップドアの間で特殊な演算を行う必要があり，サーバの DB テーブルのレコード数nに対して$O(n)$の検索時間を必要とする．
- **ハイブリッド方式**：確率的暗号ベースの方式を基本に，キーワードのハッシュ値の一部（以降，開示ビットと表記）をサーバに開示し，サーバが開示ビットに基づいて検索空間を絞り込むことで高速化を図る．サーバの DB テーブルのレコード数nに対して$O(n/2^c)$の検索時間である（cは，開示ビットのビット長）．

Vol.2021-DPS-186 No.14
Vol.2021-CSEC-92 No.14
2021/3/15

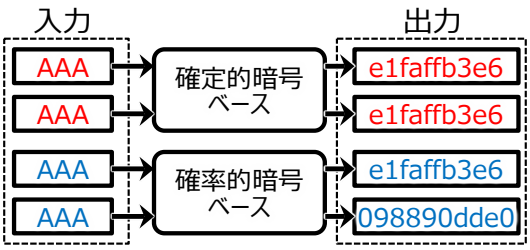

図 3 確率的出力と確定的出力

確定的暗号ベースの方式は，登録キーワードとタグ，および，検索キーワードとトラップドアが 1:1 で対応するため，DB テーブル中のタグの頻度分布と平文のキーワードの頻度分布が等しくなり，頻度分析攻撃によりキーワードが特定されやすくなってしまう[7]．一方，確率的暗号ベースの方式の場合，DB テーブル中のタグの頻度分布からは，キーワードの頻度分布が漏れないため，確定的暗号ベースの方式よりは安全性が高いとされる．しかし，検索時間が$O(n)$のため，検索を対話的に行うことが困難となるおそれがある．ハイブリッド方式は，開示ビットの長さ（以降，開示ビット長と表記）が長いほど検索空間が小さくなり，大きい高速化効果を得られるが，一方で開示ビットの頻度分布とキーワードの頻度分布が近づき頻度分析攻撃へのリスクが上がる．

本稿では，キーワードの頻度分布を隠しつつ，要件(2)を満たすような高速化効果が得られる開示ビット長を，最小エントロピーとk-匿名性[8]を用いて求める方法を提案する．さらに，データベースの分散化により，スケーラビリティを強化する．また，検索可能暗号を用いた全文検索システムとして，Li らの方式[9]と尾形らの方式[10]と同等なシステムを実装し，それらに提案方式を適用して文書数 1,000 件で性能評価した結果，キーワードの頻度分布を隠しつつ，検索時間を最大 97.2%削減できることを確認した．

2. 関連研究

2.1 検索可能暗号を用いた全文検索

2.1.1 タグにあいまい性を持たせる方式

Li らの論文[9]では，検索キーワードのタイプミスや表現の揺らぎを許容するために，ファジーキーワード検索技術を適用した方式を提案している．

文書登録時に，登録キーワードw_iに対して，編集距離[11]を基準としたファジーキーワード集合$\{w_i'\}$を網羅的に求め，$\{\mathrm{w}_i, \{w_i'\}\}$のタグを生成し，サーバに保管しておく．編集距離は，2 つの文字列がどの程度異なっているかを示す距離の一種であり，1 文字の挿入・削除・置換によって，一方の文字列をもう一方の文字列に変形するのに必要な手順の最小回数として定義される．たとえば，キーワードが CASTLE の場合，1 文字目に対する置換操作は{AASTLE, BASTLE, DASTLE, ･･･, YASTLE, ZASTLE}のようになる．しかし，このように単純に列挙すると$\{\mathrm{w}_i, \{w_i'\}\}$のサイズが著しく増加する．そこで，ワイルドカードを用いて{CASTLE, *CASTLE, *ASTLE, C*ASTLE, C*STLE, ･･･, CASTL*E, CASTL*, CASTLE*}のように表現し，$\{\mathrm{w}_i, \{w_i'\}\}$のサイズを抑える．検索キーワード$w$に対してはファジーキーワード集合を作成せず，通常通りトラップドアを作成し，タグとのマッチング処理を行う．

この方式は，ベースとなる検索可能暗号にワイルドカードを含むタグとトラップドアとの部分一致検索機能を要求する．また，タグ数は，キーワードとワイルドカードの組み合わせの場合の数に比例して増加するため，文書数が多くなると高速化が必須となる．

2.1.2 トラップドアにあいまい性を持たせる方式

尾形らの論文[10]では，文書登録時は手を加えず，検索時に検索キーワードwと意味的に近いキーワード集合$S(w)$を求め，$\{w, S(w)\}$に対しトラップドアを生成し，各トラップドアでの検索結果を OR 演算することであいまい検索を実現する方法を提案している．

この方式は，Li らの方式と異なりベースとなる検索可能暗号には，完全一致検索の機能があればよい．一方，文書数に対するスケーラビリティの問題は依然として残る．

2.2 確率的暗号ベースの検索可能暗号に対する高速化

平野らの論文[6]では，開示ビットと呼ぶ，キーワードから確定的に決まる数ビットの値をサーバへ開示し，この開示ビットを使用して DB テーブルの検索空間を絞り込むことで高速化を図る方式を提案している．さらに，提案した方式に対して安全性証明をつけている．

開示ビットは，キーワードのハッシュ値を予め定められた長さでトリミングして求める（図 4）．この開示ビットを使用した検索空間の絞り込み手順を以降で説明する．

文書登録

(1) クライアントは，登録キーワードからタグを作成する際に，登録キーワードから開示ビットも計算し，サーバに送信する．

(2) サーバは，開示ビットをタグと紐づけて DB テーブルに保存する（表 2）．

検索

(3) クライアントは，検索キーワードからトラップドアを作成する際に，検索キーワードから開示ビットも計算し，サーバに送信する．

(4) サーバは，DB テーブルの開示ビットと，検索時に送信された開示ビットが一致するレコードを絞り込み，絞り込んだレコードに対してタグとトラップドアのマッチング処理を行う．

(5) 以降の処理は，通常の検索可能暗号の処理手順と同じ．

この動作により，開示ビットによってマッチング処理時の検索空間を減らすことができる．ただし，開示ビット長

Vol.2021-DPS-186 No.14
Vol.2021-CSEC-92 No.14
2021/3/15

図 4 開示ビットの計算方法

表 2 開示ビットを導入した転置索引例

開示ビット	タグ	文書 ID
e1fa	d96b4f8a9d	File1
e1fa	a0663e3e41	File2
defb	17b4539cd4	File2
…	…	…

が長すぎると，絞り込み後のレコード集合の大きさが 1 になり，キーワードの頻度分布と一致してしまう．しかし，平野らは開示ビット長の決定方法に関する詳細な議論をしていない．

3. 提案方式

開示ビットへの頻度分析攻撃の対策を行いつつ，1 章で示した要件(2)を満たすような高速化効果が得られる開示ビット長を，最小エントロピーとk-匿名性を用いて求める方法を提案する．さらに，データベースの分散化により，文書数 1,000 規模の全文検索へのスケーラビリティを確保する．

この章では，以下の手順で提案方式の詳細を説明する．

(1) 頻度分析攻撃への対策アプローチ
(2) k-匿名性が成り立つための十分条件
(3) 開示ビット長の決定方法の詳細

3.1 頻度分析攻撃への対策アプローチ

提案方式は，開示ビット長をできるだけ短く設定することで，開示ビットの頻度分布を隠すアプローチを採る．

伊藤らは，キーワードの出現数または生起確率（以降，これらを頻度と表記）とタグの頻度をそれぞれ降順に並べ，頻度の高い順にキーワードとタグを対応付けることで，最尤推定によるキーワードの特定が可能と述べている[7]．確率的暗号ベースの方式では，サーバ側で検索結果を継続的に監視し，タグの頻度の統計を取る必要があり，攻撃成功までには長い時間を要する．一方，確定的暗号ベースの方式では，サーバが保管する DB 中のタグの種類とその出現数を数え上げるだけで頻度情報が得られるため，攻撃が成功しやすい．

開示ビットの導入によって，確定的暗号ベースの方式の場合と同様に，開示ビットの頻度分布に対して最尤推定が行われキーワードが特定される可能性がある．前述の最尤推定に対しては，伊藤らのように複数の DB にタグを確率的に振り分けてタグの頻度を攪拌するか，タグ（開示ビット）の頻度分布を隠す方法が有効である．

ハイブリッド方式の検索時間が$O(n/2^c)$であることから，開示ビット長cを増やすと，検索空間の削減量が増えるので検索性能は向上する．一方で，検索空間が削減されるということは，ある開示ビット値を持つキーワードの数が減るということであり，開示ビットの頻度分布とキーワードの頻度分布が近づき，最尤推定によるキーワード特定の攻撃を受け易くなる．

したがって，検索性能と最尤推定による攻撃に対する安全性のトレードオフをとるために，1 つの開示ビットに対応付けられるキーワードがk個以上存在するように開示ビット長cの最大値を検索対象のデータから求め，この最大値を上限として，必要な検索性能を得るための開示ビット長を設定する．

このように，「データセット中のどのようなカラムを参照してもk以上のレコードが該当する」状態をk-匿名性を持つという[8]．本稿では，どのような開示ビットで検索したとしても，絞り込み後の DB テーブルレコードが最低でも 2-匿名性を持てば統計加工された場合と同様な状態となり，攻撃者に対しキーワードの出現数や頻度分布を隠蔽することができると考え，$k = 2$をkの最小値とした．

ただし，本稿で想定する攻撃者は悪意のあるクラウド管理者で，DB テーブルのレコード内容を全て把握できるが，自身がクライアントとなり検索クエリを発行できないものとする．よって，確率的暗号ベースの方式に対する頻度分析攻撃は，攻撃成功までに長時間を要するため想定しない．

3.2 k-匿名性が成り立つための十分条件

k-匿名性を満たすための十分条件を，暗号理論においてよく利用される尺度である最小エントロピーを用いて表現する[12]. 確率変数Xの最小エントロピーは，式(1)のように定義され，Xの取り得る値の中で最も生起しやすい値の曖昧さを表すものである．

$$H_\infty(X) := -\log_2 \max_{x \in X} P(X) \quad (1)$$

最小エントロピーが 0 に近いほど曖昧さがないため，本稿の場合ではキーワードが特定されやすくなることを意味する．この定義に従い，k-匿名性が成立するための条件を求める．Wをキーワードの確率変数，B_cを開示ビット長がcの開示ビットの確率変数，$P(W|B_c = b)$をある開示ビット$b \in B_c$が与えられた時のキーワードWの条件付き生起確率とすると，k-匿名性が成立するための条件は，条件付き最小エントロピー$H_\infty(W|B_c)$が式(2)の不等式を満たすことである．

$$H_\infty(W|B_c) := -\log_2 \max_{w \in W, b \in B_c} P(W|B_c = b) \geq \log_2 k \quad (2)$$

Vol.2021-DPS-186 No.14
Vol.2021-CSEC-92 No.14
2021/3/15

ただし，$c = 0$の場合は開示ビットで絞り込まれないため，式(3)のように読み替える．$P(W)$は，キーワードの生起確率である．

$$H_\infty(W) := -\log_2 \max_{w \in W} P(W) \quad (3)$$

式(2)の意味について説明する．$H_\infty(W|B_c)$が$\log_2 k$以上とは，式(2)の真数部分が式(4)を満たす場合である．

$$\max_{w \in W, b \in B_c} P(W|B_c = b) \leq \frac{1}{k} \quad (4)$$

式(4)は，どのような開示ビット値bに対しても，キーワードが少なくともk種類存在しないと成立しない．よって，検索対象のデータが式(4)を満たすことを確認できれば，最低でもk-匿名性を確保することができる．

3.3 開示ビット長の決定方法

開示ビット長は，以下の手順で決定する．

開示ビット長の決定手順

(1) 本評価環境で，開示ビット長を変化させながら，開示ビットが与えられた時のキーワードの条件付き最小エントロピーを計算する．

(2) 手順(1) で算出した最小エントロピーが式(2)を満たす最大の開示ビット長を求める．

(3) 手順(2) で求めた最大の開示ビット長以下で，ユースケースが求める性能を満たす検索空間削減率（$= 1 - 1/2^c$，cは開示ビット長）を達成する最小の開示ビット長を決定する．

上記手順を，本評価環境下で Li らの方式に提案方式を適用した場合について説明する．

まず，手順(1) に従い実際に最小エントロピーを求めると，図 5 赤線のようになる．次に，手順(2) に従い最大の開示ビット長を求めると，$k = 2$の場合，最小エントロピーが$\log_2 2 = 1$（図 5 赤破線）以上で最も長い開示ビット長は 7 であることが分かる．最後に，手順(3) に従い開示ビット長を決定する．仮に本ユースケースで 10 倍の高速化を得ようとする場合，検索空間を 90%ほど削減でき，かつ，匿名性が上がるようにできるだけkの値が大きくなるようにcを設定すればよい．図 5 青線が検索空間削減率で，図 5 青破線が 90%削減ラインである．この場合，$c = 4, k = 22$で達成可能である（$\because H_\infty(W|B_4) \fallingdotseq 4.48 \geq \log_2 22 \fallingdotseq 4.46$）．よって，設定すべき開示ビット長は 4 と決定する．一方，尾形らの方式に提案方式を適用した場合も同様に計算すると，設定すべき開示ビット長は 4 であった．

最後に，開示ビットでの絞り込み後のキーワードの種類の数の最小値が 22 よりも大きいことを確認する．図 6 は，Li らの方式における開示ビットでの絞り込み後のキーワードの種類の数の最小値の実測値である．開示ビット長が 4 の場合，最低でも開示ビット 1 つあたり 2,598 種類のキーワードを含み，22-匿名性以上の匿名性を持つことを確認した．

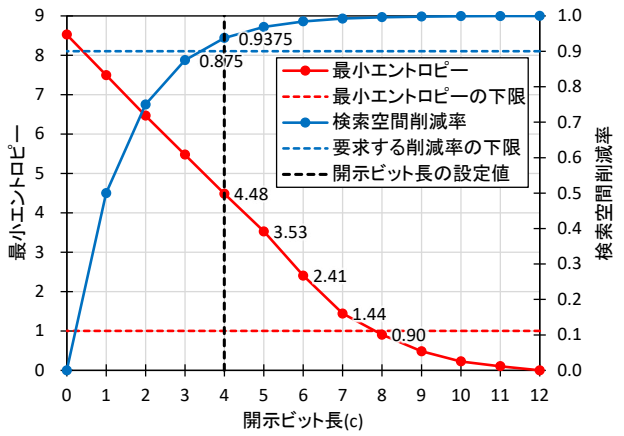

図 5 最小エントロピーと検索空間削減率

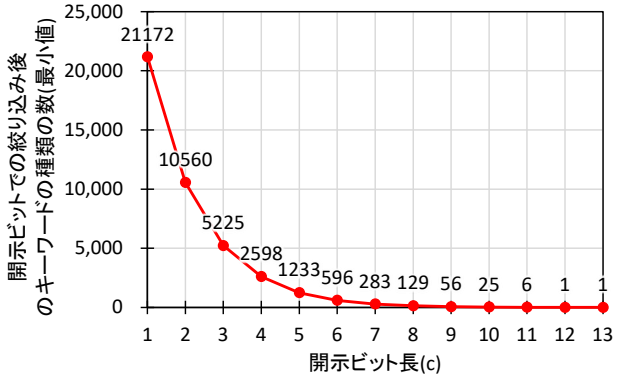

図 6 開示ビットでの絞り込み後のキーワードの種類の数（最小値）の実測値

4. 評価

1,000 件の文書に対して Li ら，尾形らの方式と提案方式を実装したものを用いて性能評価し，キーワードの頻度分布を隠しつつ，検索時間を短縮可能であることを示す．

4.1 評価条件

表 3 に評価条件の一覧を示す．提案方式の適用対象である Li らの方式で用いるファジーキーワード集合$\{w_i'\}$と，尾形らの方式で用いる検索キーワードと意味的に近いキーワード集合$S(w)$を，キーワードの類義語の集合に変更する．このように変更しても，タグあるいはトラップドアにあいまい性を持たせ，全文検索を実現するという本質的な動作は変わらない．また，提案方式を適用しない Li らの方式と尾形らの方式は，単一の DB を使用し，開示ビットを使用しない．

評価に使用する文書は，1 章で述べた社内文書検索を想

定すると設計書や仕様書が理想だが，研究目的での利用ができないため，青空文庫[13]の書籍を代替とした．

評価項目は検索時間で，10 回の試行における標本平均と，95%信頼区間を算出する．

4.2 評価環境

図 7 にプログラムスタックを示す．サーバは，Citus Docker（v7.0.3）[14][15]をベースイメージとして用い，平野らの方式を C 言語で実装した．クライアントは，Python ベースであるが，C 言語で実装した平野らの方式を呼び出している．Oracle VM ホスト環境を表 4 に，ゲスト環境を表 5 に示す．

4.3 評価結果

Li らの方式および尾形らの方式における，提案方式の適用有無による平均検索時間と信頼区間の評価結果を表 6 に示す．表 6 の信頼区間は，平均検索時間を起点とした範囲である．

提案方式を適用しない Li らの方式は，平均検索時間が 155.80 秒であり，本稿で想定する社内情報検索のユースケースの応答性能を満たしているとは言えない．

一方，提案方式を適用した場合は，4.39 秒に短縮でき，提案方式を適用しない場合と比較し 97.2%削減できることを確認した．この値であれば，ユーザが対話的に社内情報検索を使用しても問題ない範囲である．

同様に，提案方式を適用しない尾形らの方式は，平均検索時間が 227.75 秒であったが，提案方式を適用した場合は，19.89 秒に短縮され，提案方式を適用しない場合と比較し 91.3%削減できることを確認した．

表 3 評価条件一覧

	値
提案方式の適用対象	Li らの方式 尾形らの方式
文書	青空文庫の宮本百合子の書籍 1,000 冊
形態素解析エンジン	Janome[16]（固有名詞，一般名詞のみ抽出，MeCab IPA 辞書[17]使用）
類義語展開エンジン	word2vec[18] （日本語学習済みモデル[19]を使用）
類義語展開数	1 語につき 3 つの類義語に展開
検索文	愛（ヒット率：210 冊/1,000 冊）
開示ビット長	4
分散 DB	PostgreSQL 9.6 + Citus Extension
分散数	8
試行回数	10（標本平均，95%信頼区間を算出）

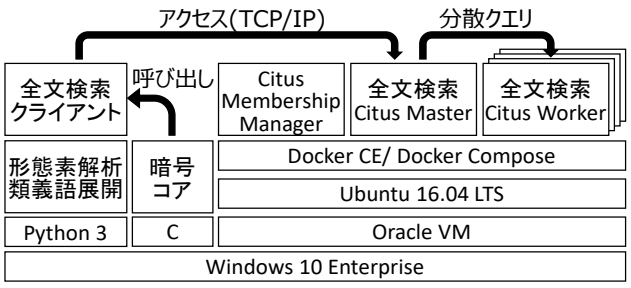

図 7 プログラムスタック

5. 考察

評価の結果から，Li ら，尾形らの方式に対して提案方式を適用すると，文書数 1,000 件の場合において検索時間が 4.39 秒～19.89 秒に短縮可能（91.3%～97.2%削減）であることを確認できた．よって，提案方式の使用で，1 章で述べた要件(1)，要件(2)を満たすことができる．

また，提案方式は，開示ビットによる絞り込み後の DB テーブルのレコード集合が最低でも 2-匿名性を持つように設計しているため，キーワードの頻度分布が漏洩する可能性も少ない．

さらに，分散 DB を使用することにより 1 つあたりの DB に格納されるタグ数が減るので，スケーラビリティの強化と同時に，確率的暗号ベースの方式での頻度分析攻撃の成立に必要なタグの頻度の観測を困難にすることができる（DB どうしが結託しない場合）．

なお，平野らの方式と伊藤らの方式は，組み合わせることができる．この場合も開示ビットとキーワードが 1:1 に対応するケースを完全に排除するために，本稿で示した開示ビット長の設定が必要である．

6. 結論

平野らの開示ビットに基づく検索可能暗号の高速化を使用する際に必要な開示ビット長の設定を，最小エントロピーと k-匿名性を用いて決定する方法を提案した．また，1,000 件の文書に対して Li ら，尾形らの方式と提案方式を

表 4 Oracle VM ホスト環境

	値
CPU	Intel Core i5-8500 CPU@3GHz (6 コア 6 スレッド)
メモリ	64GB
ディスク	Samsung SSD 860 EVO 1TB × 2 （Read 550 MB/s ， Write 520 MB/s）

表 5 Oracle VM ゲスト環境

	値
CPU	Intel Core i5-8500 CPU@3GHz (4 コア 4 スレッド)
メモリ	16GB
ディスク	100GB（シンプロビジョニング）

表 6 平均検索時間（カッコ内が信頼区間）

	提案方式適用前	提案方式適用後
Li らの方式	155.80 秒 (±2.95 秒)	4.39 秒 (±0.55 秒)
尾形らの方式	227.75 秒 (±1.33 秒)	19.89 秒 (±0.33 秒)

実装したものを用いて性能評価した結果，キーワードの頻度分布を隠しつつ，それぞれ 97.2%と 91.3%の検索時間の短縮が可能であることを示した．

今後は，以下の課題に取り組む予定である．

- 検索対象のデータセットとキーワードが追加/削除されていく場合に，どのように開示ビット長cを調整するか．
- キーワードの発生確率に非常に大きな偏りがある場合に，k-匿名性だけで対処できるか．

さらに，今回の評価のように事前にキーワードの頻度分布が把握できない場合もあるため，主要なキーワードの頻度分布をいくつか挙げ，高速化効果と安全性の関係を整理することで，適用可能なユースケースを増やしていく．

参考文献

[1] Slack Technologies : Slack, Slack (オンライン), 入手先〈https://slack.com/intl/ja-jp/〉(参照 2020-09-02).
[2] Google : Google ドライブ, Google (オンライン), 入手先〈https://www.google.com/intl/ja_ALL/drive/〉(参照 2020-09-02).
[3] Bellare, M., Boldyreva, A. and O'Neill, A.: Deterministic and Efficiently Searchable Encryption, Proc. Advances in Cryptology - CRYPTO 2007, LNCS 4622, pp.535-552 (2007).
[4] Song, D.X., Wagner, D. and Perrig, A.: Practical Techniques for Searches on Encrypted Data, Proc. IEEE Symposium on Security and Privacy, IEEE Computer Society, pp.44-55 (2000).
[5] Boneh, D., Di Crescenzo, G., Ostrovsky, R. and Persiano, G.: Public Key Encryption with Keyword Search, Proc. Advances in Cryptology - EUROCRYPT 2004, LNCS 3027, pp.506-522 (2004).
[6] 平野貴人，川合豊，小関義博：確定値を部分的に使った高速な共通鍵暗号ベース秘匿検索，コンピュータセキュリティシンポジウム 2017 論文集，vol. 2017，No.2，pp.749-756(2017).
[7] 伊藤隆，平野貴人，森拓海，服部充洋：秘匿検索の頻度分析対策としての複数 DB 活用について，情報処理学会論文誌，Vol.60, No.1,pp.240-249 (2019).
[8] Samarati, P. and Sweeney, L. : Protecting Privacy when Disclosing Information: k-Anonymity and its Enforcement through Generalization and Suppression, SRI International (1998).
[9] Li, J., Wang, Q., Wang, C., Cao, N., Ren, Kui., and Lou, W. : Fuzzy Keyword Search over Encrypted Data in Cloud Computing, Proc. IEEE Conference on Computer Communications, IEEE, pp. 441–445 (2010).
[10] 尾形わかは，金岡晃，松尾真一郎：実用的な多機能検索可能暗号方式～身も蓋もない方式を考えてみた～，暗号と情報セキュリティシンポジウム（SCIS2015）予稿集，3F1-3 (2015).
[11] Levenshtein, V. : Binary codes capable of correcting spurious insertions and deletions of ones, Problems of Information Transmission, vol.1, no.1, pp.8–17(1965).
[12] 四方順司，渡邉洋平：情報理論的暗号技術について，情報処理，Vol.55, No.3, pp. 260 ～ 267（2014）.
[13] 青空文庫，入手先〈https://www.aozora.gr.jp/〉(参照 2020-01-20).
[14] Docker : citusdata/citus, dockerhub，入手先〈https://hub.docker.com/r/citusdata/citus〉(参照 2020-01-06).
[15] Citus Data : Citus，入手先〈https://github.com/citusdata/docker/tree/v7.0.3〉(参照 2020-01-06).
[16] 打田 智子：Janome v0.3 documentation(ja)，入手先〈https://mocobeta.github.io/janome/〉(参照 2020-01-20).
[17] MeCab: Yet Another Part-of-Speech and Morphological Analyzer，入手先〈http://taku910.github.io/mecab/〉(参照 2020-01-20).
[18] Gensim : models.word2vec – Word2vec embeddings, Radim Řehůřek，入手先〈https://radimrehurek.com/gensim/models/word2vec.html〉(参照 2020-01-20).
[19] Kyubyong / wordvectors : Pre-trained models , github，入手先〈https://github.com/Kyubyong/wordvectors〉(参照 2020-01-20).