「マルチメディア，分散，協調とモバイル
(DICOMO2013) シンポジウム」 平成25年7月

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

松田 規[1] 伊藤 隆[1] 柴田 秀哉[1] 服部 充洋[1] 平野 貴人[1]

概要：Boneh らによって公開鍵暗号に基づく検索可能暗号が提案されて以来，AND/OR 検索や範囲検索などの高機能化を図った成果が提案されている．しかし，検索時に，暗号化した文書に添付された検索用のタグを順次チェックする必要があるため，検索時間が文書数に比例するという欠点がある．一方，Bellare らによって提案された確定的暗号に基づく方式は，検索時間は文書数の log オーダに抑えられるが，キーワードのエントロピーが Brute Force 攻撃に対して安全であるほど大きくなければならないという制約があり実用的ではない．本論文では，はじめに安全性と高速性のバランスを調整可能な検索可能暗号方式を提案する．本方式は，タグから数ビットの情報漏れを許容することで，サーバ側で索引生成が可能となり，検索の高速化が可能となる．次に，提案方式を Web アプリケーションに適用する方式を提案する．本方式では，データベースのユーザ定義関数として検索可能暗号を実装し，またブラウザにて暗号・復号を行う構成をとることで，Web アプリケーションで復号のための秘密鍵を一切管理する必要がなくなり，利用者自身が機密情報のセキュリティをコントロールすることが可能となる．

Efficient Searchable Encryption and Its Application to Web Services

Nori Matsuda[1] Takashi Ito[1] Hideya Shibata[1] Mitsuhiro Hattori[1] Takato Hirano[1]

1. はじめに

近年では，企業の IT コスト低減のため，仮想化技術によって複数サーバを統合したり，データセンタやクラウドコンピューティングなどの第三者が提供するサービスを利用する動きが起こりつつある．しかし，第三者が提供するサービスを利用する場合，企業が保有するデータを社外サーバ上に保管する事になるため，企業機密をどのように保護するかという問題が生ずる．ここで，単にファイルを保管するという単純なサービスならデータを暗号化するだけで十分かもしれないが，多くのサービスは Web アプリケーションとして作られており，企業のデータを預かるとともに処理を行う事で付加価値を提供している．そのため，単純にデータを暗号化して機密保護を行うような手法は，多くのサービスには適用できないという課題がある．

その解決策の一つとして，Boneh ら [3] によって公開鍵暗号に基づく検索可能暗号が提案された．この方式は，利用者がキーワードを暗号化して検索用のタグを作成し，これを暗号化データに付加してサービスに保管する．公開鍵暗号に基づく方式なので，暗号化は誰でも実施する事ができる．検索は，秘密鍵を持つ利用者だけが実施可能で，検索キーワードを秘密鍵で暗号化してトラップドアを作成し，これをサービスに送付する．サービスは，暗号化データを復号することはできないが，タグとトラップドアを特殊な演算で比較することで検索ができる．本提案以降，AND 検索ができる方式，範囲検索ができる方式など，様々な検索可能暗号が提案されている [4], [6], [8]．しかし，これらの方式は確率的暗号に基いているため，タグからキーワードに関する情報が１ビットも漏れない事を保証する識別不可能性 (IND セキュリティ) と言う非常に高い安全性を満たすように設計されている．これは，登録時にタグから索引が作れないことを意味しており，検索時に全てのタグを順番にチェックする必要が生じる．そのため，検索時間が保管されているデータ数に比例するという問題があった．

一方，Bellare ら [1], [2] によって RSA-OAEP をベースとして，データ数の log オーダで検索可能な方式が提案さ

[1] 三菱電機（株） 情報技術総合研究所
Mitsubishi Electric Corp., 5-1-1, Ofuna, Kamakura 247-8501

れた．これは，OAEP パディングを行う際に，擬似乱数のシードとしてキーワードを用いることで，確定的に暗号化する方式である．その安全性評価のために，平文を知らない攻撃者が暗号文を見ただけでは，キーワードに関する部分情報を得る事ができないという新しい安全性モデル PRIV セキュリティを導入し，安全性を証明している．その後，本研究をベースとした改良案が色々と発表されているが，いずれの方式も確率的暗号とは異なりキーワードのエントロピーが極めて大きいことを仮定しており，現実のシステムに適用することが難しい．

本論文では，公開鍵暗号に基づく検索可能暗号において，安全性を IND セキュリティから若干弱めることによって，高速化を実現する方式について提案する．具体的には，通常はタグからは 1 ビットの情報も漏れないという IND セキュリティを満たすように構成するが，索引開示鍵という特別な鍵を利用者から開示してもらった場合に限って，キーワードから確定的に決まる数ビットの索引値をタグから取り出せるようにする．検索時，トラップドアの索引値と同一の索引値を持つタグだけを一致判定することで，一致判定を行うべきタグの数を減らして検索を高速化できる．また，索引開示鍵で開示する索引値のビット数を利用者がコントロールできるようにすることで，データ数が増加しても検索時間を一定以内に抑えられるような仕組みが実現できる．

さらに，検索可能暗号を Web アプリケーション (サービス) へ適用するための仕組みについても提案する．Web アプリケーションの多くは，データをデータベースに保管するため，SQL を用いて検索可能暗号を利用できることが望ましい．また，検索や復号に利用する秘密鍵は，検索可能暗号の利点を損なわないように，利用者側で管理できることが望ましい．そこで，入力として表を受け取り，処理結果として表を出力できる表値型のユーザ定義関数を用いて，データベースに検索可能暗号を実装することで，SQL を用いて検索可能暗号を利用可能とする．また，利用者側は ActiveX [*1] を用いてブラウザで暗号・復号を行うようにし，生成した暗号化データ/タグやトラップドアはエンコードして Web アプリケーションに送付する．Web アプリケーションが，Web アプリケーション内でデータを加工・参照せずに，データベースにデータ/タグをそのまま渡すような作りで，さらに利用する SQL 文を設定ファイルで指定できるようになっている場合には，既存アプリケーションのコードを改修せずに検索可能暗号を利用可能となる．

2. 想定システム

企業内でデータを保管する場合，情報システム部門がサーバを管理するため，保管されているデータのセキュリティ管理もコントロールすることができる．一方，クラウドサービスとして提供されている Web アプリケーションの場合，クラウド事業者の管理者がサーバや Web アプリケーションを管理するため，従来のように企業の情報システム部門が全てのコントロール権を持って管理を行うことができない．もしクラウド事業者の管理者が悪意を持った場合，企業の機密情報を閲覧できてしまうリスクが考えられる．そのため，クラウドサービスを利用する時こそ，企業の利用者だけが機密情報を閲覧できるように暗号化を行いつつ，暗号化したまま Web アプリケーションで処理が行える暗号化技術が必要になると考えている．

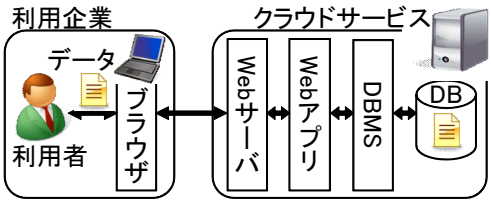

図 1 想定システム

そこで，我々は，クラウドサービスとして提供される Web アプリケーションを想定システムとした．そのシステム構成を，図 1 に示す．図のように，Web アプリケーションが置かれるサーバはクラウド事業者が管理する．Web アプリケーションは，Java[*2]Servlet や JSP[*3]などを用いてプログラミングされ，利用者との通信のために Web サーバが，データを保管するためにデータベースが利用される．また，クラウド事業者の管理者によって，サーバの利用状況のモニタリング，バックアップなどが実施される．一方，Web アプリケーションを利用する企業は，ブラウザを使って Web アプリケーションにアクセスを行う．保管するデータは，ブラウザから直接入力したり，文書ファイルをサーバにアップロードするなどの手法で，Web アプリケーションに登録する．

また我々は，Web アプリケーションとして複数の情報システムや入退機器で利用される ID 情報を管理する統合 ID 管理アプリケーションを想定した．これは，ID 情報には社員の個人情報として氏名，生年月日，性別などが含まれ，それらは企業の責任において厳格に管理すべき情報であることから，暗号化による機密保護が最も有用になると考えたためである．統合 ID 管理アプリケーションでは，利用者から入力された個人情報を預かるとともに，利用者の要求によって個人情報を検索したり，データ更新を受け付けたりする．また，条件にマッチする利用者の権限情報を抽出し，企業内の情報システムやビルの入退機器に ID 情報を配信することも可能である．

[*1] ActiveX は，米国 Microsoft Corp. の登録商標である

[*2] Java は，Oracle Corp. の登録商標である
[*3] JSP は，Oracle Corp. の登録商標である

3. 検索可能暗号

検索可能暗号とは，暗号化したままキーワード検索が可能な暗号化方式である．Boneh ら [3] によって提案された一致検索におけるアルゴリズムの定義を下記に示す．

- KeyGen は，セキュリティパラメータ 1^λ を入力とし，マスター公開鍵 pk とマスター秘密鍵 sk のペアを生成し，マスター公開鍵とマスター秘密鍵を出力
- PEKS は，マスター公開鍵 pk，キーワード空間 KW から選んだキーワード w を受け取り，キーワードを暗号化することでタグ S_w を生成し，タグを出力
- Trapdoor は，マスター公開鍵 pk，マスター秘密鍵 sk，検索キーワード w' を受け取り，検索キーワードを暗号化することでトラップドア $T_{w'}$ を生成し，トラップドアを出力
- Test は，マスター公開鍵 pk，タグ S_w，トラップドア $T_{w'}$ を受け取り，タグとトラップドアの一致判定演算を行うことで，タグを生成するのに用いたキーワードと，トラップドアを生成するのに用いた検索キーワードが同一かどうかを判定し，判定結果として $1(w = w'$ の時)，$0(w \neq w'$ の時) を出力

上記のように，検索可能暗号では，暗号化したキーワード (タグ) と，暗号化した検索キーワード (トラップドア) が同一かどうかを，復号することなく判定する事ができる．また，検索キーワードを暗号化するためにはマスター秘密鍵が必要なため，検索できる利用者を限定することができる．

なお，検索可能暗号はデータ自身を復号する機能は提供されないため，図 2 に示すように別途データを暗号化する必要がある．実際に利用する際は，従来の RSA 暗号や ID ベース暗号などを用いてデータ本体を暗号化して，検索のために上記アルゴリズムで生成したタグを関連付けて保管することになる．データを検索する際は，上記アルゴリズムでタグを検索し，ヒットしたタグに対応した暗号化データを検索者に返却する．検索者は自身の秘密鍵で暗号化データを復号すれば，検索キーワードに対応するデータを得ることができる．

また，検索可能暗号アルゴリズムの IND-CPA 攻撃者に対する安全性は下記のようなゲームで定義される．

(1) 挑戦者 $\mathcal{C}$ は KeyGen を実行してマスター公開鍵 pk，マスター秘密鍵 sk を生成し，マスター公開鍵 pk を攻撃者 $\mathcal{A}$ に与える
(2) $\mathcal{A}$ は, 適応的に多項式回だけ検索キーワード $w \in \{0,1\}^*$ に対応するトラップドア T_w を取得する
(3) $\mathcal{A}$ は，challenge するキーワード w_0, w_1 を選択し，$\mathcal{C}$ に送付する．ただし，すでに取得したトラップドア T_w で区別ができてしまうようなキーワードを選ぶことはできない．$\mathcal{C}$ は，ランダムに $b \in \{0,1\}$ を選んでタグ S_{w_b} を生成したのち，タグ S_{w_b} のみを $\mathcal{A}$ に送付する．
(4) $\mathcal{A}$ は，ステップ 3 の制約のもと，ステップ 2 と同様にトラップドアを取得する
(5) $\mathcal{A}$ は，ビット b の推測値 b' を出力する

上記ゲームにおいて，攻撃者 $\mathcal{A}$ がビット b を正しく推定する確率が，1/2 と negligible な差しかない場合，検索可能暗号は IND-CPA 攻撃者に対して安全である．

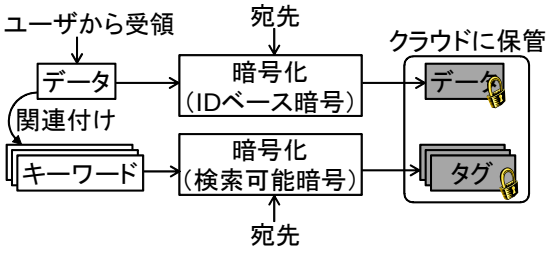

図 2 利用方法

4. 課題

Web アプリケーションに検索可能暗号を適用するにあたり，最も問題となるのは検索性能である．単純に検索可能暗号を適用すると，データ件数に比例した検索時間を要するという課題がある．一般的なデータベースでは，保管するデータからハッシュテーブルや B-Tree 等を用いた索引を生成することで，検索時の応答時間を大きく短縮している．しかし，1 ビットの情報漏れも起こらない IND-CPA を満たすことが保証されている検索可能暗号では，キーワードの同一性という情報すら隠してしまい，結果として索引を生成することができない．そのため，検索時に全てのタグをチェックする必要が生じてしまい，データ件数に比例した検索時間を要する．さらに，公開鍵暗号系の検索可能暗号は，そのアルゴリズムの実現にペアリングという演算を用いるが，この演算は 1 回あたり 0.41ms 程を要する [12]．我々が用いた岡本，高島によって開発された内積述語暗号 [8] では，このペアリング演算を 10 回程度用いるため，タグ 1 個のチェックでも 4.1ms を要してしまう．例えば，社員の ID 情報を検索するようなシステムを想定する場合，大企業ではグループ社員として 10 万から 30 万人ほどの社員をかかえているため，検索時間が数分から数十分オーダになってしまい，実用に耐えられなくなってしまう．

また，既存研究は理論的なアルゴリズム開発がほとんどであり，実際のアプリケーションに適用するための技術が確立されていないという欠点もある．例えば，データが保管されているデータベースには，検索可能暗号に対応した検索機能は実装されていないため，Web アプリケーションに検索可能暗号の検索機能を組み込む必要がある．さらに，データベースのある列を検索するために生成したトラッ

プドアで，他の列のデータが検索できてしまうとセキュリティ上の問題となるため，列ごとに暗号化の鍵を変えるなどの対策を考えなければならない．また，複数の利用者がWeb アプリケーションを使う環境では，どのようにデータを共有すれば良いかを検討しなければならない．

5. 関連研究

先に述べたように，検索可能暗号の高速化方式としては，Bellare ら [1], [2] によって RSA-OAEP をベースとして，データ数の log オーダで検索可能な方式が提案されている．これは，タグを生成するときに使用するキーワードのエントロピーが極めて大きいことを仮定した場合に，PRIV セキュリティという安全性を満たすことが保証された方式である．しかし，例えば日本人の姓は約 20 万種類しかなく，エントロピーが極めて大きいとはいえないため，本方式では十分な安全性を保証することができなくなるという問題がある．

また，内積述語暗号を用いた検索可能暗号の高速化方式が，吉田ら [13] によって提案されている．この方式は，検索にヒットするデータが極めて少ないという状況であるなら，データ数の log オーダで検索が可能である．しかし，キーワード空間のサイズに比例してタグやトラップドアのサイズが大きくなってしまうと言う問題がある．そのため，氏名などの種類が多い個人情報を暗号化するのには適していない．

また，検索可能暗号の高速化，及び Web アプリケーションへの適用に関して，Popa ら [7] によって CryptDB と呼ぶデータベースが提案されている．この論文では，オニオン暗号化方式と呼ぶ手法を使うことで，検索の高速化を図っている．例えば，キーワードを確定的暗号，確率的暗号の順に 2 重に暗号化してデータベースに保管しておく．通常は，確率的暗号で暗号化された状態のため，キーワードに関しては 1 ビットの情報も漏れない．検索を行う場合，外側の確率的暗号を復号するための秘密鍵を端末からデータベースに送付し，データベースにて確率的暗号の復号を行うことで，キーワードは確定的暗号で暗号化された状態となる．そして，端末にて検索キーワードを確定的暗号で暗号化して，それを検索クエリとしてデータベースに送付する．確定的暗号はキーワードの同一性などの情報が漏洩するが，保管されている暗号化データと検索クエリをバイナリ比較するだけでキーワードの一致判定ができるため，検索が高速に実施できる．そのため，通常のデータベースの検索性能から約 26% 程度の性能劣化で済むと報告されている．また，SQL クライアント S/W にてキーワードの暗号化・復号化を実施するようにして，Web アプリケーションへの適用を可能としている．しかし，一度検索を行うことでキーワードが確定的暗号で暗号化された状態になるため，安全性が IND セキュリティより劣る OW セキュリティ (onewayness) に低下するという問題がある．また，Web アプリケーションがクラウドサービスとして提供される場合，SQL クライアント S/W はクラウド側で動作するため，データを復号する秘密鍵は暗号化データとともにクラウド側に保管されることになる．そのため，データの機密性が十分に保証できないという問題がある．

6. 提案方式

我々は，確率的暗号に基づく検索可能暗号をベースとして，利用者が許可した場合にのみ，サーバ側で索引を生成できるような仕組みを実現することで，検索性能の向上を図る．また，データベースのユーザ定義関数として検索可能暗号を実現することで，SQL 文を使って検索可能暗号を利用可能とする．さらに，ActiveX としてブラウザ上で動作する暗号化・復号の仕組みを実現することで，秘密鍵を利用者側で秘密に保持したまま，検索可能暗号を Web アプリケーションへ適用できる仕組みも実現する．本節では，それらの実現方式について述べる．

6.1 索引生成可能な検索可能暗号

通常の検索可能暗号では，データベースに保管したタグからはキーワードに関する情報が一切漏れないため，検索時にすべてのタグとトラップドアを比較する必要が生ずる．これが，文書数に比例した検索時間がかかる原因である．

そこで我々は，キーワードの部分情報を 1 ビットずつ開示可能な索引開示鍵という新しい鍵を定義する．この索引開示鍵を利用者がデータベースに開示すると，データベースにてタグから索引値という情報を得ることができるようになり，それを元に索引生成を行うことで検索の高速化ができる．索引開示鍵をデータベースに開示しない状態では，タグから一切の情報が漏れない IND-CPA セキュリティを満たすことが保証される．代わりに，検索時間は保管されているタグ数に比例する．一方，索引開示鍵を開示した場合，データベースでは開示を受けた l ビット分の索引値が生成可能となる．この場合，データベースでは保管されているタグを事前に 2^l 個に分類しておけるため，検索性能を 2^l 倍にすることができる．更に，索引値が同一になるキーワード同士では，キーワードの識別不可能性が保証できるという特徴がある．

まず初めに，我々のアルゴリズムの定義を示す．我々の定義では，前記の索引を生成する仕組みに加え，階層型 ID ベース暗号 (HIBE) を用いて複数ユーザへの対応も実現できるように拡張する [10].

- Setup は，セキュリティパラメータ 1^λ，ID 階層数 L_{ID}，最大索引ビット数 L_{IDX} を入力とし，マスター公開鍵 pk とマスター秘密鍵 sk をのペアを生成し，マスター公開鍵とマスター秘密鍵を出力
- GenUserKey は，マスター公開鍵 pk，マスター秘密鍵

sk，ユーザ IDID_U を受け取り，検索キーワードの暗号化に用いるユーザ秘密鍵 sk_U を生成しユーザ秘密鍵を出力

- GenTag は，マスター公開鍵 pk，宛先 IDID_R，キーワード w を受け取り，キーワードを暗号化することでタグ S_w を生成し，タグを出力
- GenTrapdoor は，マスター公開鍵 pk，ユーザ秘密鍵 sk_U，検索キーワード w' を受け取り，検索キーワードを暗号化することでトラップドア $T_{w'}$ を生成し，トラップドアを出力
- Test は，マスター公開鍵 pk，タグ S_w，トラップドア $T_{w'}$ を受け取り，タグとトラップドアの一致判定演算を行うことで，タグを生成するのに用いたキーワードと，トラップドアを生成するのに用いた検索キーワードが同一かどうかを，タグやトラップドアを復号することなく一致判定を行い，判定結果（1:一致，0:不一致）を出力
- GenIndexKey は，マスター公開鍵 pk，マスター秘密鍵 sk，索引生成を許可する宛先 IDID_R，生成を許可する索引ビット数 l を受け取り，タグやトラップドアから検索の高速化を可能とする索引値を導出するための索引開示鍵 ik_l を生成し，索引開示鍵を出力
- RevealTagIndex は，マスター公開鍵 pk，タグ S_w，索引開示鍵 ik_l を受け取り，キーワードによって確定的に定まる索引値 $\{0,1\}^l$ をタグから計算し，索引値を出力
- RevealTdIndex は，マスター公開鍵 pk，トラップドア $T_{w'}$，索引開示鍵 ik_l を受け取り，検索キーワードによって確定的に定まる索引値 $\{0,1\}^l$ をタグから計算し，索引値 $\{0,1\}^l$ を出力

次に，その実現アルゴリズムについて述べる．SCIS'12 における我々の提案 [11] では，内積述語暗号から HVE(Hidden Vector Encryption) を構成し，キーワードのハッシュ値を $L_{IDX}+1$ 次元ベクトルとみなして，HVE を使ってタグを生成していた．キーワードの一致判定を行う場合は，検索キーワードのハッシュ値からトラップドアを生成し，タグとトラップドアに含まれるハッシュ値が同一かどうかを検証する．また，索引開示鍵を生成する際は，ハッシュ値のある 1 ビットだけに値をセットして，残りの値を*(ワイルドカード) にした索引開示用トラップドアと索引開示用タグを生成し，これを索引開示鍵としていた．これを使えば，タグやトラップドアから 1 ビットの索引値を取り出すことができた．しかし，キーワードのハッシュ値を使って一致判定を行うため，一致判定 Test の処理時間が $(L_{IDX}+1)$ 倍に増加してしまい，索引で一致判定すべきタグの数を絞り込んだ効果が薄れてしまうという問題点があった．

そこで，キーワードは我々が提案 [10] した階層型 ID に対応した検索可能暗号 (HPEKS) を用いて暗号化しつつ，索引として開示するハッシュ値は，HVE の代わりに HIBE(Hierarchical Identity-Based Encryption) を使ってキーワードとは別に暗号化する方式を提案する．本方式では，一致判定 Test の処理時間が増加することがないため，索引として開示したビット数に応じた高速化が可能となる．下記に，その具体的なアルゴリズムを示す．

- $\mathsf{Setup}(1^\lambda, L_{ID}, L_{IDX}) \rightarrow (\mathsf{pk}, \mathsf{sk})$
  - $(\mathsf{pk}_{\mathsf{HPEKS}}, \mathsf{sk}_{\mathsf{HPEKS}}) := \mathsf{Setup}_{\mathsf{HPEKS}}(1^\lambda, L_{ID}+1)$
  - $(\mathsf{pk}_{\mathsf{tagidx}}, \mathsf{sk}_{\mathsf{tagidx}}) := \mathsf{Setup}_{\mathsf{HIBE}}(1^\lambda, 1)$
  - $(\mathsf{pk}_{\mathsf{tdidx}}, \mathsf{sk}_{\mathsf{tdidx}}) := \mathsf{Setup}_{\mathsf{HIBE}}(1^\lambda, L_{ID}+1)$
  - $\mathsf{pk} := \{\mathsf{pk}_{\mathsf{HPEKS}}, \mathsf{pk}_{\mathsf{tagidx}}, \mathsf{pk}_{\mathsf{tdidx}}, L_{ID}, L_{IDX}\}$
  - $\mathsf{sk} := \{\mathsf{sk}_{\mathsf{HPEKS}}, \mathsf{sk}_{\mathsf{tagidx}}, \mathsf{sk}_{\mathsf{tdidx}}\}$
  - return pk, sk
- $\mathsf{GenUserKey}(\mathsf{pk}, \mathsf{sk}, ID_U) \rightarrow sk_U$
  - $sk_U := \mathsf{GenKey}_{\mathsf{HPEKS}}(\mathsf{pk}_{\mathsf{HPEKS}}, \mathsf{sk}_{\mathsf{HPEKS}}, ID_U)$
  - return sk_U
- $\mathsf{GenTag}(\mathsf{pk}, ID_R, w) \rightarrow S_w$
  - $S_{HPEKS} := \mathsf{GenTag}_{\mathsf{HPEKS}}(\mathsf{pk}_{\mathsf{HPEKS}}, ID_R, w)$
  - $h := \mathsf{Hash}(w)$
  - $ID_{R'} := \mathsf{String}(ID_R)$
  - $EIDX_i := \mathsf{Encrypt}_{\mathsf{HIBE}}(\mathsf{pk}_{\mathsf{tagidx}}, ID_{R'}|\mathsf{I2S}(i), h_i)$, $i = 1, \ldots, L_{IDX}$
  - $S_w := (S_{HPEKS}, \{EIDX_i\})$
  - return S_w
- $\mathsf{GenTrapdoor}(\mathsf{pk}, sk_U, w') \rightarrow T_{w'}$
  - $T_{HPEKS} := \mathsf{GenTrapdoor}_{\mathsf{HPEKS}}(\mathsf{pk}_{\mathsf{HPEKS}}, sk_U, w')$
  - $h := \mathsf{Hash}(w)$
  - $EIDX'_i := \mathsf{Encrypt}_{\mathsf{HIBE}}(\mathsf{pk}_{\mathsf{tdidx}}, ID_R||\mathsf{I2S}(i), h_i)$, $i = 1, \ldots, L_{IDX}$
  - $T_{w'} := (T_{HPEKS}, \{EIDX'_i\})$
  - return $T_{w'}$
- $\mathsf{Test}(\mathsf{pk}, S_w, T_{w'}) \rightarrow \{0,1\}$
  - $result := \mathsf{Test}_{\mathsf{HPEKS}}(\mathsf{pk}_{\mathsf{HPEKS}}, S_{HPEKS}, T_{HPEKS})$
  - return $result$
- $\mathsf{GenIndexKey}(\mathsf{pk}, \mathsf{sk}, ID_R, l) \rightarrow ik_l$
  - $ID_{R'} := \mathsf{String}(ID_R)$
  - $tagkey_i := \mathsf{KeyGen}_{\mathsf{HIBE}}(\mathsf{pk}_{\mathsf{tagidx}}, ID_{R'}|\mathsf{I2S}(i))$, $i = 1, \ldots, l$
  - $tdkey_i := \mathsf{KeyGen}_{\mathsf{HIBE}}(\mathsf{pk}_{\mathsf{tdidx}}, ID_R||\mathsf{I2S}(i))$, $i = 1, \ldots, l$
  - $ik_l := (\{tagkey_i\}_{i=1,\ldots,l}, \{tdkey_i\}_{i=1,\ldots,l})$
  - return ik_l
- $\mathsf{RevealTagIndex}(\mathsf{pk}, S_w, ik_l) \rightarrow \{0,1\}^l$
  - $PIDX_i := \mathsf{Decrypt}_{\mathsf{HIBE}}(\mathsf{pk}_{\mathsf{tagidx}}, tagkey_i, EIDX_i)$
  - $IDX_l := PIDX_1|PIDX_2|\ldots|PIDX_l$
  - return IDX_l
- $\mathsf{RevealTdIndex}(\mathsf{pk}, S_w, ik_l) \rightarrow \{0,1\}^l$
  - $PIDX_i := \mathsf{Decrypt}_{\mathsf{HIBE}}(\mathsf{pk}_{\mathsf{tdidx}}, tdkey_i, EIDX'_i)$

– $IDX_l := PIDX_1|PIDX_2|\ldots|PIDX_l$
– return IDX_l

なお，上記アルゴリズム中の関数 String は ID 列を文字列に変換する関数，関数 I2S は数値を文字列に変換する関数，記号 | は文字列を結合する演算子，記号 || は ID 階層を 1 階層追加する演算子である．

6.2 データベースへの組込み

Web アプリケーションは，SQL を用いてデータベースにアクセスするが，検索可能暗号の一致判定処理 Test は通常のデータベースでは実装されていない．そのため，Web アプリケーション側でタグを取り出し，一致判定処理を独自に実施する必要が生ずる．これは，既存の Web アプリケーションにとってプログラムの改修を行う必要があり，適用時のコスト上昇を招く．そこで我々 [9] は，データベースに対して検索可能暗号の一致判定処理を組込むことで，Web アプリケーションの改修量を低減可能な仕組みを実現する．また，一致判定を複数スレッド/PC で並行処理する事で，検索処理の高速化も図る．

実現システムの概念を図 3 に示す．タグの一致判定処理は時間がかかるため複数スレッド/PC で並行処理することが好ましいが，暗号化データやタグの管理は，単一データベースで管理する仕組みにする．これは，(1) 複数データベースを用いたタグの分散管理を行うと，複数データベース間でのデータ整合性を保証する仕組みが必要となることから管理に手間がかかること，(2) 複数データベースを用いた分散処理を行いたい場合，本検索可能暗号に対応したデータベースを基本単位として，これを複数並べた構成に拡張すれば良いと考えたこと，が理由である．

次に，Web アプリケーションが SQL を用いて検索可能暗号が利用できるように，ユーザ定義関数の仕組みを用いて一致判定処理を実装する．その際，DBMS が管理する多くのタグを並行処理することで高速化が図れるように，通常使われるスカラ値型のユーザ定義関数ではなく，レコード集合を出力可能な表値型のユーザ定義関数を用いて，一致判定処理を実現する．本関数は，入力として DB から取り出した複数のタグ（を含むレコード集合）を入力として受け取る．受け取ったタグは複数の計算ノードに配分して，各計算ノードで一致判定処理を行い，判定結果一覧 (一致/不一致) を追加したレコード集合として出力する．これにより，暗号化した列を検索する場合にユーザ定義関数が呼ばれるように SQL 文を作成することで，検索可能暗号独特の一致判定処理を Web アプリケーションで実施することなく，検索可能暗号を利用することが可能となる．同時に，ユーザ定義関数内で自動的に複数の計算ノードを用いた分散処理を行うため，検索の高速化も可能となる．

また，我々が開発した検索可能暗号は，タグから導出した索引値と，トラップドアから導出した索引値を比較して，索引値が同一となるタグのみ一致判定することで検索の高速化が可能な仕組みである．そこで，タグから索引値を生成するユーザ定義関数，トラップドアから索引値を生成するユーザ定義関数も開発する．

なお，検索可能暗号では，一致判定処理のために公開パラメータが必要となる．そこで，暗号アルゴリズムや公開パラメータを管理するための管理テーブルを設ける．この管理テーブルでは，テーブル名・列名毎に，アルゴリズム ID・パラメータ ID・公開パラメータ・索引開示鍵を管理することができる．前述のユーザ定義関数では，この管理テーブルから必要な公開パラメータや索引開示鍵を取得して処理を行う．

6.3 Web アプリケーション適用

Web アプリケーションに検索可能暗号を適用するため，データベースのスキーマ変更が必要となる．前述のように，検索可能暗号を利用する場合は，検索のタグを生成するのとは別に，データを従来の暗号で暗号化して，その両者を保管する必要が生ずる．また，我々の提案した検索可能暗号では，タグから索引値を生成することができ，それを使った検索の高速化が可能なため，索引値もあわせて保管する必要がある．そこで，図 4 に示したように，元々のテーブルでデータが保管されている列があった場合，暗号化データ，タグ，索引値の 3 列に増やして保管した．暗号化データとタグはバイナリデータ型，索引値は文字列型とする．また，暗号化したデータを検索する場合，最初にトラップドアから索引値を算出するユーザ定義関数を用いてトラップドアから索引値を取得し，検索対象の表を索引値が一致するという条件で絞り込む．次に，絞り込んだレコードを一致判定を行うユーザ定義関数に渡してタグの一

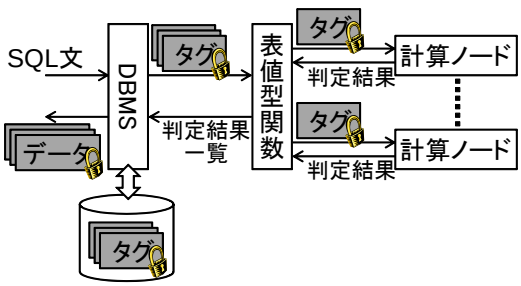

図 3 データベースへの組込み方式

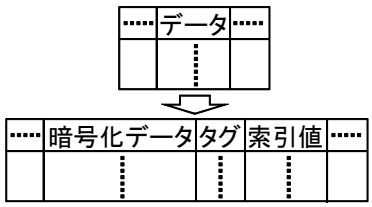

図 4 データベースのスキーマ変更

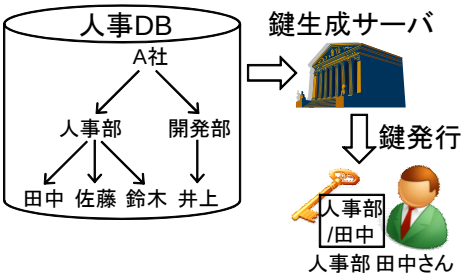

図 5 秘密鍵への属性情報埋め込み方法

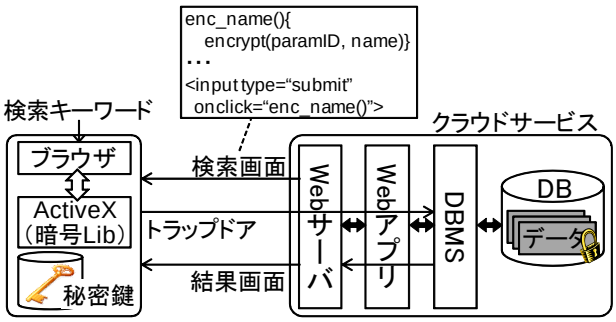

図 6 Web アプリへの適用方法

致判定を行い，対応する暗号化データを取り出す，という SQL 文を記述すれば良い．

次に，鍵管理方法について述べる．クラウドサービスに検索可能暗号を適用する場合，データのセキュリティを企業がコントロールできることが好ましい．これを実現するためには，企業内で鍵の管理を行う必要がある．そこで，鍵管理は企業内に設置された鍵生成サーバにて実施する．また，一般的な Web アプリケーションは，企業内の複数人の利用者が利用し，データを共有するために使われる．例えば，人事部員 A が暗号化したデータは，人事部に属する部員全員が読めるようにしたい，等のニーズが考えられる．我々が開発した検索可能暗号はこのようなケースに対応可能で，グループで検索権限を共有することが可能となる．そこで，企業内に設置された人事 DB やディレクトリ等と連携して，鍵生成サーバが社員毎に所属情報を埋め込んだ秘密鍵を配布する事で，グループ共有が可能な仕組みを実現する．例えば図 5 は，鍵生成サーバが人事部 田中さんに対して秘密鍵を発行するケースである．まず，人事 DB では全社員の所属情報が管理されており，鍵生成サーバから参照可能であると仮定する．田中さんに秘密鍵を発行する場合，鍵生成サーバは人事 DB から田中さんの所属情報を取り出し，属性”人事部/田中”を埋め込んだ秘密鍵を発行し，田中さんに送付する．同様に，佐藤さんには属性”人事部/佐藤”が埋め込まれた秘密鍵を発行する．タグが，宛先”人事部/*”にて作成されているなら，前記の田中さんの秘密鍵，佐藤さんの秘密鍵のいずれで作成したトラップドアであっても，正しく一致判定が行われる．もし，属性”開発部/井上”を持つ秘密鍵で検索を行った場合，属性と宛先が不一致となるため，キーワードの一致・不一致に関わらず常に一致判定の結果は不一致となり，キーワードに関する情報が漏れることはない．

次に，データを暗号化・復号する仕組みについて述べる．一般的にクラウドサービスでは，端末からブラウザを使ってサービスにアクセスする．そこで，端末側で実施する検索可能暗号の処理は，全てブラウザ上で実施できる必要がある．そこで，図 6 に示すように，ActiveX を用いてブラウザで暗号化・復号を行う仕組みとする．ブラウザでクラウドサービスにアクセスすると，Web サーバから検索画面が送付される．この検索画面には submit(検索) ボタンが付いており，submit ボタンが押下された時に呼ばれる JavaScript [*4]を埋め込む．ここで，暗号化データに対応した秘密鍵を使って検索用のトラップドアを生成しなければ正しい検索ができなくなるため，どの公開パラメータを使って暗号化が行われているかを示すパラメータ ID なども埋め込む．そして，利用者が検索キーワードを指定して submit ボタンを押下すると，JavaScript が ActiveX を呼びだして検索キーワードからトラップドアを作成し，Web サーバへ送付する．送付されたトラップドアは，Web アプリケーションによって SQL 文のプレースホルダにセットされると，そのままデータベースに受け渡されるため，データベースにて検索可能暗号の一致判定処理を行う．最後に，検索結果が Web サーバによって結果画面としてブラウザに送付されるので，同様に JavaScript と ActiveX を使って暗号化データを復号して，画面に表示を行う．

このようにすることで，Web アプリケーションに保管された機密情報を暗号化によって保護するだけでなく，その秘密鍵をクラウドサービス側ではなく企業側で管理することで，企業がセキュリティをコントロールできるという検索可能暗号のメリットを発揮することができる．

7. まとめ

本論文では，クラウドサービスとして提供される Web アプリケーションに対して検索可能暗号を適用することで，利用者の機密情報を保護するための手法について提案した．提案方式では，確率的暗号に基づく検索可能暗号をベースとして，利用者が許可した場合にのみ，クラウドサービス側で索引を生成できるような仕組みを実現することで，検索性能の向上を図った．また，表値型のユーザ定義関数として検索可能暗号を実現することで，SQL 文を使って検索可能暗号を利用可能としつつ，並行処理によって高速化が図れる仕組みを実現した．さらに，ActiveX としてブラウザ上で動作する暗号化・復号の仕組みを実現することで，秘密鍵を利用者側で秘密に保持したまま，検索可能暗号を Web アプリケーションへ適用した．

[*4] JavaScript は，Oracle Corp. の登録商標である

今後は，実際の Web アプリケーションへ適用し，その実現性や性能評価を実施する予定である．また，実際の運用においては，運用中に公開パラメータを変更するなどの鍵管理が必要となるため，その実現方式についても検討を行う予定である．

参考文献

[1] Bellare, M., Boldyreva, A. and O ' Neill, A. : Deterministic and Efficiently Searchable Encryption, *CRYPTO 2007*, LNCS, vol.4622, pp.535-552 (2007).

[2] Bellare, M., Fischlin, M., O ' Neill, A. and Ristenpart, T.: Deterministic Encryption: Definitional Equivalences and Constructions without Random Oracles, *CRYPTO 2008*, LNCS, vol.5157, pp.360-378 (2008).

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

[4] Boneh, D. and Waters, B.: Conjunctive, subset, and range queries on encrypted data, *TCC 2007*, LNCS, vol.4392, pp.535-554 (2007).

[5] Brakerski, Z. and Segev, G.: Better Security for Deterministic Public-Key Encryption: The Auxiliary-Input Setting, *CRYPTO 2011*, LNCS, vol.6841, pp.543-560 (2011).

[6] Katz, J., Sahai, A. and Waters, B.: Predicate encryption supporting disjunctions, polynomial equations, and inner products, *EUROCRYPT 2008*, LNCS, vol.4965, pp.146-162 (2008).

[7] Popa, R. A., Redfield, C. M. S., Zeldovich, N. and Balakrishnan, H.: CryptDB: Protecting Confidentiality with Encrypted Query Processing, *Proc. of ACM, SOSP 2011*, pp. 85-100 (2011).

[8] Lewko, A., Okamoto, T., Sahai, A., Takashima, K. and Waters, B.: Fully secure functional encryption: Attribte-based encryption and (hierarchical) inner product encryption, *EUROCRYPTO 2010*, LNCS, vol.6110, pp.62-91 (2010).

[9] 柴田秀哉，田村孝之: RDBMS へアドオン可能な高 CPU 負荷演算の分散並列処理手法，第 5 回データ工学と情報マネジメントに関するフォーラム (DEIM2013)，F10-4 (2013).

[10] 松田規，服部充洋，伊藤隆，米田 健: 階層的内積述語暗号を用いたデータセンタシステムの検討，2010 年 暗号と情報セキュリティシンポジウム (SCIS2010)，4F2-3 (2010).

[11] 松田規，服部充洋，平野貴人，森拓海，伊藤隆，川合豊，坂井祐介，太田和夫: 安全性と高速性の両立を目指した検索可能暗号 (1)，2012 年 暗号と情報セキュリティシンポジウム (SCIS2012)，1A3-1 (2012).

[12] 光成滋生，照屋唯紀，岡本栄司: BN 曲線上の Optimal Ate ペアリングのソフトウェア実装，2012 年 暗号と情報セキュリティシンポジウム (SCIS2012)，1B1-4 (2012).

[13] 吉田麗生，永井彰，小林鉄太郎，冨士仁: 内積述語検索可能暗号のためのバッチ検索アルゴリズム，2011 年 暗号と情報セキュリティシンポジウム (SCIS2011)，4C2-4 (2011).