検索可能暗号を適用したデータベースに対する鍵失効方式と検索性能向上方式の研究

THESIS

Performance Improvements and Key Revocation Methods
for Public Key Searchable Encryption Database Systems
検索可能暗号を適用したデータベースに対する
鍵失効方式と検索性能向上方式の研究

２０２１年１２月

静　岡　大　学
大学院自然科学系教育部
情報科学専攻

松田 規

目 次

図 目 次

表 目 次

第1章　序論

1.1　本研究の背景と目的

近年，サービスの早期立ち上げのため，また企業の IT コスト低減のため，クラウドサービスを活用することが当たり前になりつつある．このような状況はさらに広がりを見せつつあり，これまでは一企業がクラウドを活用するケースが主流であったが，複数企業でクラウド上のデータを共同利用するケースへとニーズが拡大しつつある．例えば，企業間のオープンイノベーションが活発になりつつあるが，クラウドを活用して互いの開発成果を共有するケースも進みつつある．クラウドを活用することで，複数企業間での情報共有が円滑にできるだけでなく，利用者の役割に応じてアクセス権を付与する事で安全に情報共有できるメリットがある．他の例としては，機器管理を支援する IoT システムが考えられる．IoT システムでは，世界中に設置されたあらゆる機器から情報を収集するために，どこからでもアクセス可能なクラウドが活用されている．また，設計者や保守員など機器のライフサイクルに関わるあらゆるステークホルダーが，自身の役割の範囲でデータを閲覧できるようになり，各種サービス提供が円滑に進められるようになるというメリットがある．

このようなユースケースには，高安全なクラウドサービスの利用が必要となるが，その解決策の一つがデータ暗号化である．特に，データ暗号化の際に条件を指定でき，条件を満たすユーザのみが暗号化データを復号できる関数型暗号（Functional Encryption）や属性ベース暗号（Attribute-based Encryption）を用いると，暗号化データに復号権限を設定することができるため，機密度や公開範囲の異なる様々なデータを扱う企業がクラウドサービスを利用する際に都合が良く，これまでに様々な方式が提案されている [11]，[40]，[53]，[65]，[89]．また，多くの暗号化データが蓄積されると目的のデータがどれかが分からなくなるため，データを暗号化したまま検索ができるという機能を持つ検索可能暗号の活用も注目されている．この 2 つの技術を組み合わせることで実現されるのが，図 1.1 に示した検索可能暗号データベースである．

初めに属性ベース暗号について述べるが，これは ID ベース暗号，階層型 ID ベース暗号，内積述語暗号と発展してきたうえで完成した暗号方式である．ID ベース暗号は，暗号化時に指定した ID と，秘密鍵を発行した際に指定した ID が一致する場合，ただその時に限って，その秘密鍵で暗号化データを復号できる方式である (図 1.2)．階層型 ID ベース暗号は，ID 列を指定して復号制御ができる方式であり，内積述語暗号はベクトルの内積値が 0 になるときに限って復号できる方式である (図 1.2)．一方，属性ベース暗号はこれら暗号方式をさらに高機能にしたもので，暗号化データの復号条件を AND/OR などの論理式を使って指定できる暗号化方式である (図 1.3)．その復号条件の指定の仕方に応じて，鍵ポリシー型 (key policy) と暗号文ポリシー型 (ciphertext policy) に分けることができる．鍵ポリシー型 (KP-ABE) は，暗号化する際にデータに付与する属性集合 $\Gamma \in \Sigma$ を指定して暗号化を実施

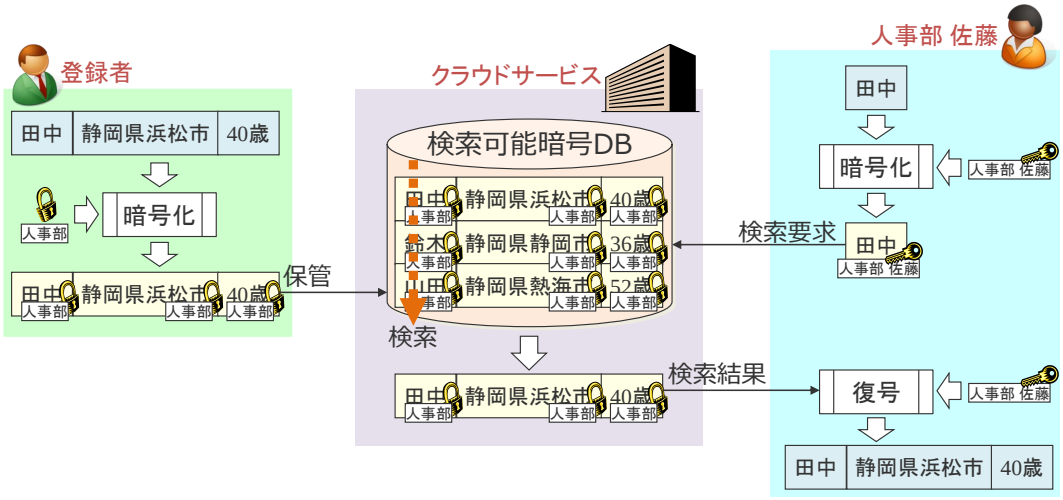

図 1.1: 検索可能暗号データベースの概要

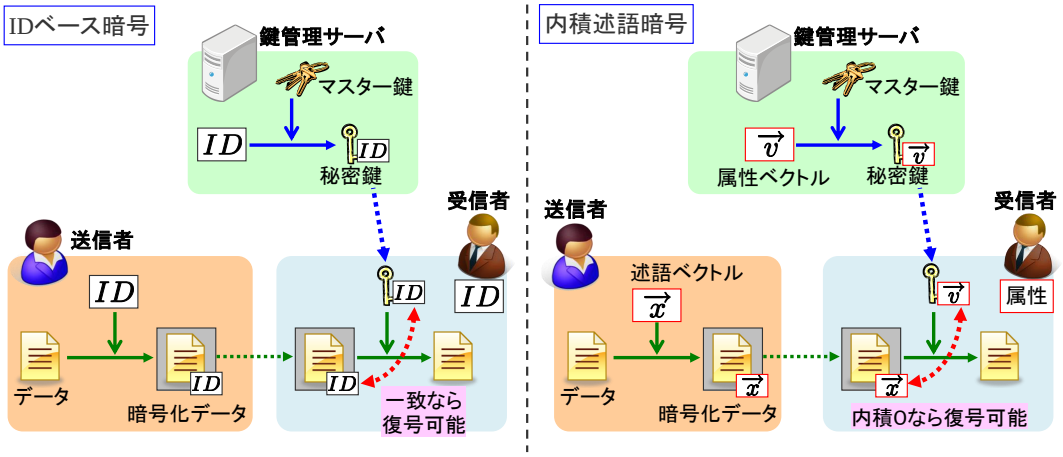

図 1.2: ID ベース暗号と内積述語暗号の概要

する．ユーザ秘密鍵は，復号可能な条件を属性を用いた条件式であるアクセス構造 $\mathbb{S} \in \mathcal{F}$ を指定して生成する．多くの属性ベース暗号では，AND/OR 条件を用いてアクセス構造を指定することができる．そして，属性集合 Γ がアクセス構造 $\mathbb{S}$ を満たす場合に限って，暗号化データの復号が可能となる．一方，暗号文ポリシー型 (CP-ABE) は，暗号化データ生成時にアクセス構造 $\mathbb{S} \in \mathcal{F}$ を指定し，ユーザ秘密鍵を生成する際に属性集合 $\Gamma \in \Sigma$ を指定する点が異なる．

最初の属性ベース暗号は，Sahai ら [71] によって提案された Fuzzy ID ベース暗号である．本方式は，秘密分散と ID ベース暗号を組合わせた方式であり，ユーザ秘密鍵にはユーザの属性集合と閾値からなるアクセス構造が埋め込まれる．一方，暗号化データには受信者が満たすべき属性集合を指定する．そして，暗号化データとユーザ秘密鍵の属性集合の積集合の大きさが閾値以上の場合に復号ができる方式である．本方式では通常の (k, n) 閾値法による秘密分散が利用されているため，AND/OR などの条件式を使うことができない．次に

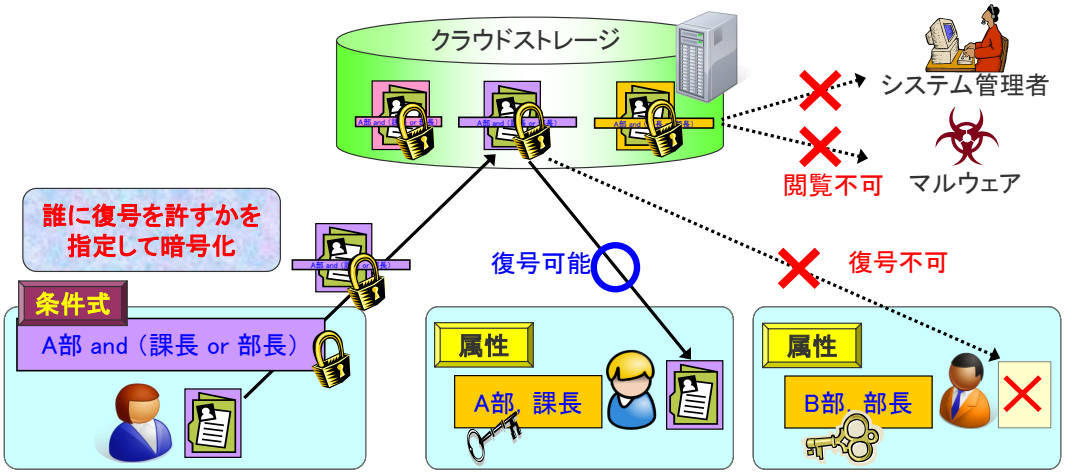

図 1.3: 属性ベース暗号 (暗号文ポリシー型) の概要

Goyal ら [41] によって提案された方式では，より柔軟なツリー構造による復元条件が指定できる秘密分散法を利用しているため，AND/OR 条件を使ったアクセス構造を指定できる KP-ABE が実現された．また，Ostrovsky ら [66] は，Broadcast Encryption で使われている失効のテクニックを KP-ABE に適用することで，AND/OR 条件に加えて NOT 条件もサポートできる属性ベース暗号を提案した．本論文では，KP-ABE での実現方式が示されているが，以下で述べる Bethencourt ら [11] の CP-ABE に対しても NOT 条件を拡張できることが示唆されている．Bethencourt ら [11] は，Goyal ら [41] の方式と同様に AND/OR 条件を使ったアクセス構造を指定できる CP-ABE 方式を提案した．秘密分散としては，Monotone Span Program(MSP) と呼ばれる方式を採用している．Waters[87] は，Linear Secret Sharing Scheme(LSSS) を使うことで，AND/OR 条件を使ったアクセス構造を指定でき，暗号化の効率を改善した CP-ABE 方式を提案した．Lewko ら [53] は，Waters[88] や Lewko ら [54] によって階層型 ID ベース暗号の安全性証明を adaptive-secure にするためのテクニックとして提案された Dual System Encryption を用い，adaptive-secure な属性ベース暗号の構成方法を提案した．本方式は，AND/OR 条件が利用でき，KP-ABE と CP-ABE の双方が実現されているが，合成数位数のペアリングを用いているため，素数位数のペアリングに比べて鍵長が大きくなってしまい，効率的ではないという課題があった．その後，高島ら [65] によって，彼らの提案した内積述語暗号 [53] と LSSS を組合わせた属性ベース暗号の一種である関数型暗号が提案された．本方式も Lewko ら [53] と同様に Dual System Encryption を用いて安全性証明が構成されており，DLIN 仮定に基づいて adaptive-secure な状況下で標準モデルでその安全性が示されている．更に，素数位数ペアリングを用いているため，Lewko らの方式に比べて計算効率も改善されている方式である．これらの研究によって，属性ベース暗号の基本的な構成は完成していった．

次に検索可能暗号について述べるが，一番最初に提案された検索可能暗号は，Song ら [79] による共通鍵暗号に基づく方式である．本方式は，共通鍵とキーワードの組によって一意に定まるランダム関数を生成し，そのランダム関数からマスク値を生成する．このマスク値で，暗号化キーワードを XOR して，それをタグとする．検索を行う場合は，暗号化キーワード

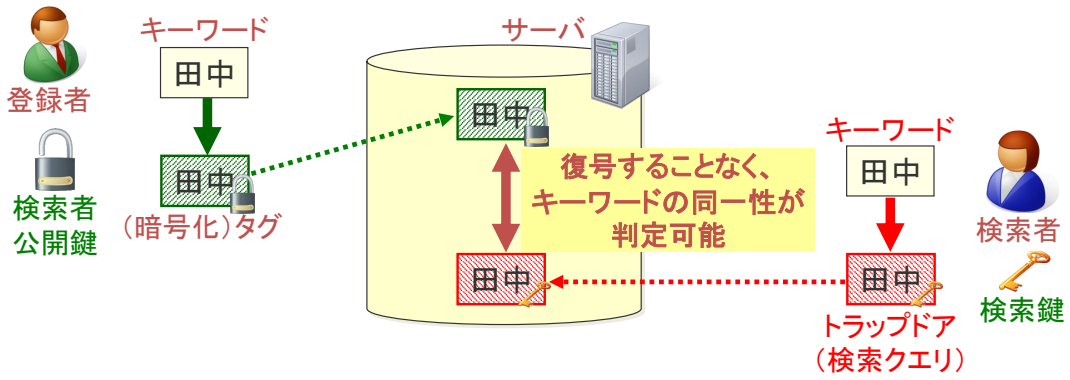

図 1.4: 公開鍵ベースの検索可能暗号の概要

と前記ランダム関数を生成し，これを検索クエリとする．サーバ側では，検索クエリの暗号化キーワードと，タグの XOR を取り，マスク値を復元したのち，検索クエリ内のランダム関数を用いてマスク値が正しい値かどうかをチェックする．チェックにパスすれば，キーワードが同一だと見なす方式である．本方式は非常にシンプルではあるが，キーワードを復元することなく検索ができるという点で非常に画期的な論文である．また，同じキーワードであっても毎回暗号化データが変わるという確率的アルゴリズムであるため，キーワードの同一性などの情報を隠すこともできる．ただし，共通鍵暗号やランダム関数などを用いているためキーワードが復元できないことを考察しているが，厳密な安全性証明は付与されていなかった．

本論文ののち，共通鍵ベースの方式と，公開鍵ベースの方式に分かれて発展を遂げていった．例えば，高速化や更なる安全性の向上などの研究がなされつつある [26][27][84]．また，大小比較を実現するため，安全性を IND-CPA から緩和させることで，順序保存暗号を実現する研究もなされている [3][12][13][36][44]．いずれの方式も，Song ら [79] の方式と同様に，データから取り出されたキーワードを検索可能暗号で暗号化してタグを生成し，データそのものは従来の暗号方式で暗号化したうえで，暗号化データとタグの組をサーバに保管する．そして，そのタグを暗号化したまま検索し，取り出した暗号化データを復号することで元データを得るものであり，基本的な考え方は同じである．

共通鍵ベースの方式とは，暗号化に用いる鍵と，検索クエリを作成する際に用いる鍵が同じ方式を指す．言い換えれば，データを暗号化できるユーザは，必ず検索を行うことができることを意味する．共通鍵ベースの方式は，共通鍵暗号やハッシュ関数などの高速演算が可能なプリミティブを用いているため，検索処理が非常に速いというメリットがある．しかし，暗号化と検索に用いる唯一の共通鍵を，システムを利用する全てのユーザで共有しなければならないという課題があった．この制約は，一企業がクラウドを活用するケースではさほど大きな問題とならないが，複数企業でデータを共有したいケースなどでは大きな制約となる．一方，公開鍵ベースの方式とは，暗号化に用いる鍵と，検索クエリを作成する際に用いる鍵が異なる方式で，暗号化は公開鍵を持っていれば誰でも実施できるが，検索クエリの生成は秘密鍵を持ったユーザのみが実施できるという特徴がある (図 1.4)．そのため，企業内や企業間でのデータ交換や，不特定多数のユーザから情報を収集するケースなど，誰が暗号化するか限定できないようなケースでは公開鍵ベースの方式が必須となる．ただし，公開

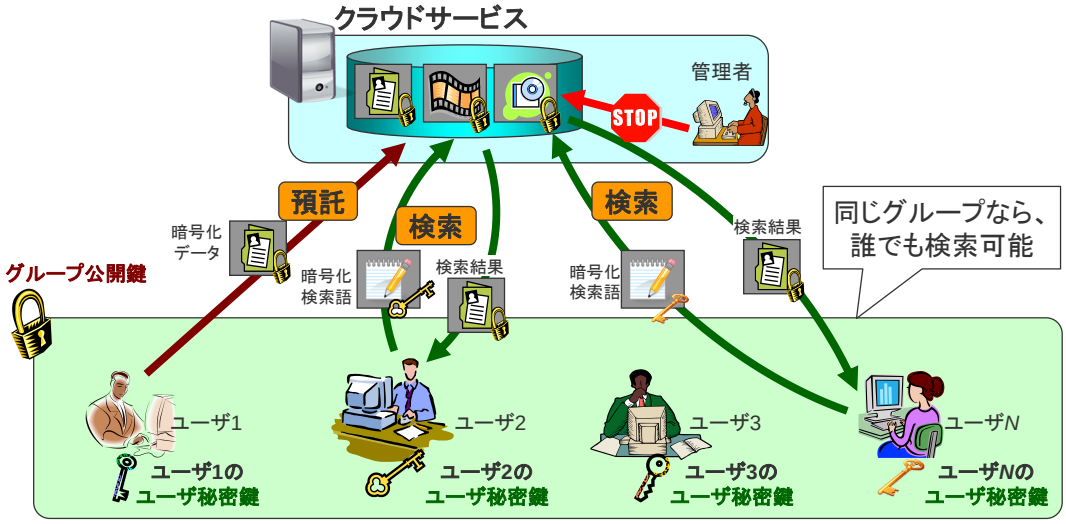

図 1.5: グループ共有に対応したマルチユーザ対応検索可能暗号の概要

鍵ベースの方式の多くは，ペアリング暗号による実現がほとんどであることから，検索処理に時間がかかるという課題もある．

企業間データ共有などの用途を想定した場合は，ユーザ ID だけでなく所属やプロジェクト名などで柔軟にアクセス制御が可能な，検索可能暗号の実用化が求められる (図 1.5)．これが実現できれば，適切な属性が割り当てられた検索鍵を持つユーザだけが，自身がアクセスしても良いデータのみを検索することができる様になる．しかし，共通鍵暗号ベースの検索可能暗号では，所属やプロジェクト名などを用いた柔軟なアクセス制御ができないという課題がある．そのため，共通鍵暗号ベースの検索可能暗号は，このようなシチュエーションには適さない．このような複数企業でデータ共有したいケースでは，公開鍵暗号に基づく検索可能暗号が適していると考えている．しかし，検索可能暗号に限らず通常の暗号においても公開鍵暗号は演算量が多く時間がかかること，更に，同じキーワードであっても暗号化のたびに異なる暗号化データになるという確率的暗号であるため，全ての暗号化データに対して一致判定処理を行わなければ検索結果が得られないという課題があることから，公開鍵ベース検索可能暗号の実用化に向けた高速化の研究がなされている [9][10]．この方式は，同じデータを暗号化すれば必ず同じ暗号化データになるという性質を持つ確定的暗号に基づく方式であるため，タグはキーワードを公開鍵で暗号化して生成し，トラップドアも同じようにキーワードを公開鍵で暗号化して生成する (図 1.6)．サーバ側では，タグとトラップドアが同じバイナリデータになるものを探すだけで済むため，一致判定処理が軽量化される．更に，一般的なデータベース製品では，バイナリ比較でデータを高速に検索できるように，インデックスと呼ばれる索引情報を生成する仕組みが用意されているため，その仕組みを使ってデータ件数 N に対して $O(N)$ にて検索を行うことができる．また，タグからキーワードに関する情報が漏れないことを定義するため，従来から使われていた indistinguishability(識別不可能性) では強すぎ，onewayness(一方向性) では不十分と考え，その中間的な安全性として新しく PRIV-security という安全性定義を導入してタグの安全性を証明している．ただし，検索クエリに相当するトラップドアも公開鍵から生成されるため，誰でも検索ができてしまうという欠点がある．言い換えれば，ランダムに選んだキーワードで検索して，タグの

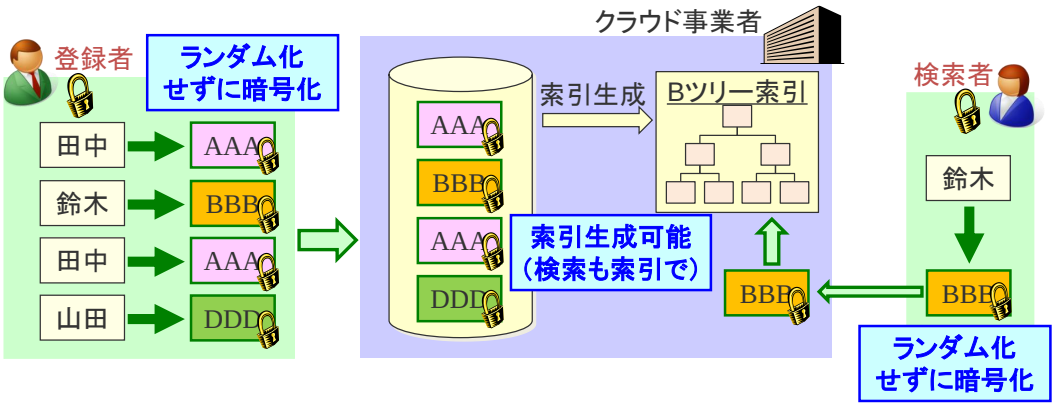

図 1.6: 確定的暗号による検索可能暗号の概要

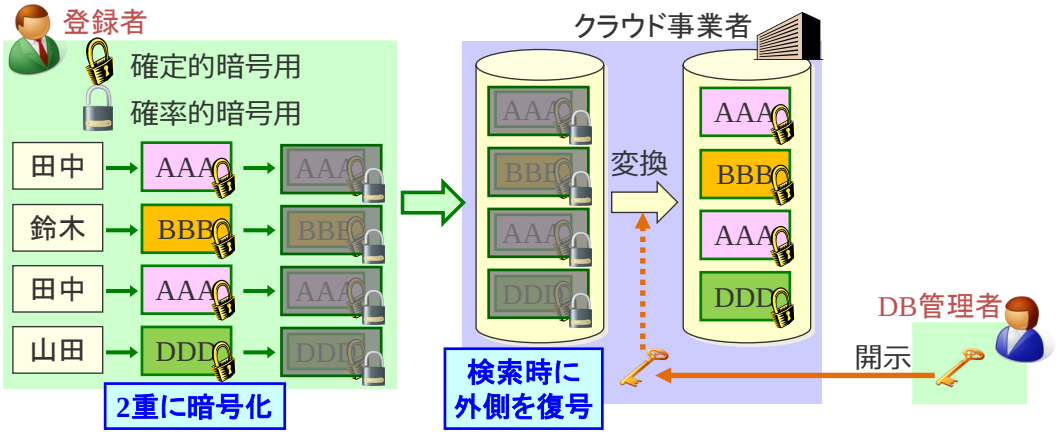

図 1.7: Popa らによるオニオン暗号化の概要

中のキーワードを推測するというブルートフォース攻撃が可能であることを意味しているため，本方式を安全に使うためには，ブルートフォース攻撃に対して安全になるようにキーワードのエントロピーがセキュリティパラメータと同程度に大きいことが求められる．一般的に，我々が使うキーワードは種類が大きいものの，エントロピーがセキュリティパラメータと同程度に大きくはないと予想されるため，理論的な面白さはあるものの実用的ではないと考えられる．

別の高速化手法が Popa ら [67] によって提案されている．この論文では，確定的暗号と確率的暗号で二重に暗号化するオニオン暗号化方式を提案し，検索の高速化と安全性の両立を図っている (図 1.7)．初期状態は確率的暗号で保護されているため，検索は実施できないが安全性が高い．もしキーワード検索を行いたい場合は，端末から DB サーバに対して一番外側の確率的暗号を確定的暗号に変換する鍵を渡すことで，安全性を確定的暗号レベルにまで落とす代わりに検索が実施できるようになる．一度確定的暗号に変換すれば，以降は高速に検索が行えるというアイデアである．これにより，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．しかし，確定的暗号に変換されてしまうと，Bellare らと同じ課題に直面する．それを回避するため，Popa らは共通鍵暗号ベースの検索

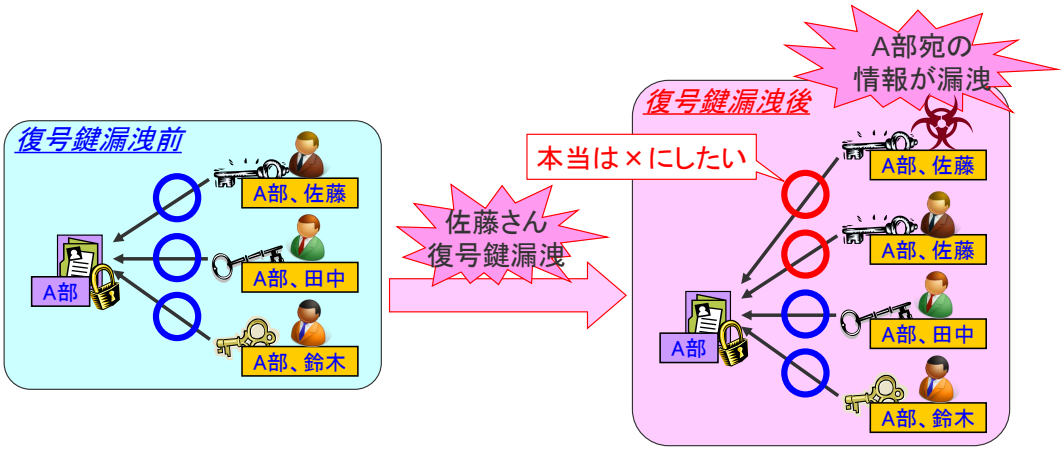

図 1.8: 属性ベース暗号における復号鍵失効の課題

可能暗号を用いて実装を行っていた．

以上のように，検索性能と安全性のバランスをとる方式の提案がなされつつあるものの，実用上に十分と考えられる方式は存在していない．また，企業間でのデータ共有に適したマルチユーザ化を図った方式も研究されている [43] が，検索性能と安全性のバランスを取りつつ，マルチユーザ化を図っていく必要がある．さらに，既存研究の多くはアルゴリズムレベルの改良にとどまっていたため，Web システム適用に向けたデータベースへの組込みについては研究は数少なく [81]，検索性能と安全性のバランスが取れ，マルチユーザ化を図った方式をデータベースに組み込む方式についての研究も必要である．

上記の検索性能の改善に加えて，企業での利用に着目すると，異動・退職に伴ってアクセス権が変化するケースや，復号鍵を紛失するケースなどが考えられることから，ユーザや復号鍵の失効処理，すなわち “それまで読めていたデータを読めなくする処理” も必要になると考えられる (図 1.8)．失効について，公開鍵と復号鍵が一対一に対応する RSA 暗号や ID ベース暗号では様々な方式が提案されている [46]，[75]，[80]，[83]，[91]．また，複数の受信者が存在する放送型暗号でも，暗号化する時に失効しているユーザを受信者から外す仕組みが研究されている [5]，[50]，[90]．しかし，関数型暗号や属性ベース暗号の場合，同じ属性を持っているユーザが複数存在しうるが，そのうちの特定ユーザだけを失効できなければならない．さらに，過去に復号できていた暗号化データも復号できないようにする必要があり，失効を困難なものとしている．例えば，暗号化データが A 部の社員であれば誰でも復号可能というポリシーで暗号化されている場合を考える．従来，A 部のメンバーが A 氏，B 氏，C 氏で 3 人ともに復号できていたのに，あるタイミングで C 氏が異動になった場合，C 氏の復号鍵だけ失効して，A 氏，B 氏は引き続き復号できるようにしなければならない．このように複数の復号鍵の一部だけを，任意の時点から無効化できることが必要である．

そこで，属性ベース暗号の失効方式についても様々な研究がなされている．そのアプローチは Direct Revocation と Indirect Revocation の 2 つに大別できる [6]．Direct Revocation は，暗号化時に失効されているユーザを指定することで，失効されたユーザが復号できないように暗号化する方式である．一方，Indirect Revocation は，暗号化時には誰が失効しているかを意識する必要が無く，暗号化データの変換や，失効していないユーザ秘密鍵の定期的

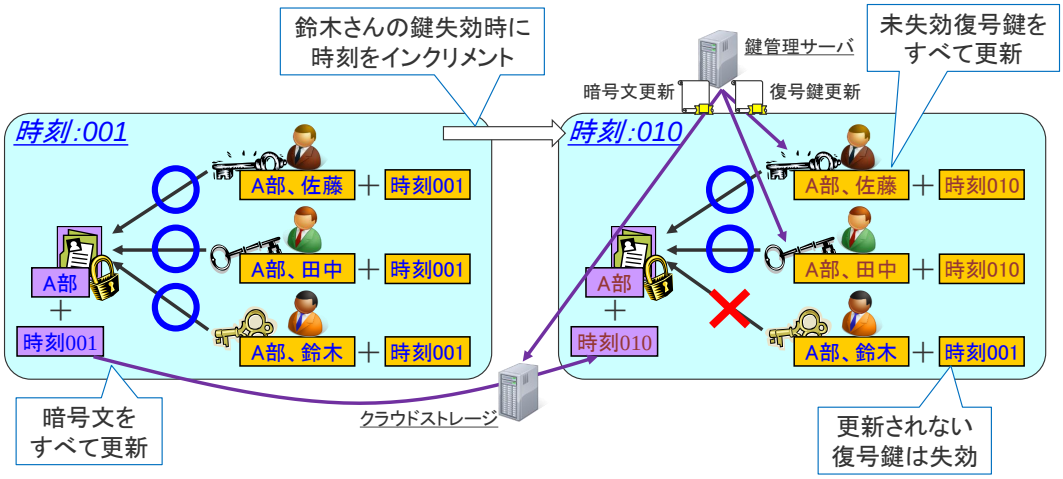

図 1.9: 属性ベース暗号の世代管理による失効方式

更新などにより，失効されたユーザが復号できないようにする方式である．どちらの方式が良いかはユースケース次第と考えているが，企業機密の保護・保管というユースケースを考慮すると，暗号化した後に復号鍵が漏洩する可能性にも対応できなければならず，Indirect Revocation のアプローチが適していると考えている．

この Indirect Revocation のアプローチを，暗号学的な手法で実現しようとした方式がいろいろと提案されている．いずれの方式も，任意のタイミングで秘密鍵の失効を行えるようにして，さらに過去に復号できていた暗号化データも復号できないようにすることを目指した方式である．実現方式は方式ごとに異なるが，大きくは暗号化データとユーザ秘密鍵に時刻 (世代番号) を付与して，両者の時刻が一致するときのみ復号ができる様に暗号化を行う (図 1.9)．ユーザの秘密鍵を失効させたい場合，時刻を +1 して，サーバに保管されたすべての暗号化データの時刻を +1 するように暗号文更新を行う．同様に，ユーザ秘密鍵のうち失効していないユーザ秘密鍵に関して時刻を +1 して秘密鍵更新を行い，失効させるユーザ秘密鍵に関しては時刻の更新を行わない．これにより，失効したユーザ秘密鍵では時刻が不一致となり復号できなくなるため，ユーザ秘密鍵を失効することができる．一方で，全ての暗号化データと失効していないユーザ秘密鍵を更新する必要が生じるため，暗号化データ数やユーザ数が多い場合，また更新頻度が高い場合などは，システムの運用に大きな負荷がかかるという問題点がある．

別のアプローチとして，上記において全ての暗号文や失効していない全てのユーザ秘密鍵の更新が必要という課題を解決すべく，プロキシーを活用することで効率化を図った方式が提案されている (図 1.10)．仕組みとしては，ユーザがクラウドストレージから暗号化データを取り出す際，プロキシーサーバを経由してアクセスするようにするが，このプロキシーサーバの力を借りることでユーザ秘密鍵の失効を行う方式である．具体的には，鍵管理サーバがユーザ秘密鍵を生成する際，同時にプロキシー鍵を生成してプロキシーサーバに配布する．暗号化データを復号するには，プロキシー鍵での部分復号，ユーザ秘密鍵での最終復号の 2 段階を経なければならないようになっている．ユーザがクラウドストレージから暗号化データを取り出す際，プロキシーサーバが部分復号を行うが，プロキシーサーバからプロキシー鍵を削除することでユーザ秘密鍵を使えなくする仕組みである．本アプローチは効率的

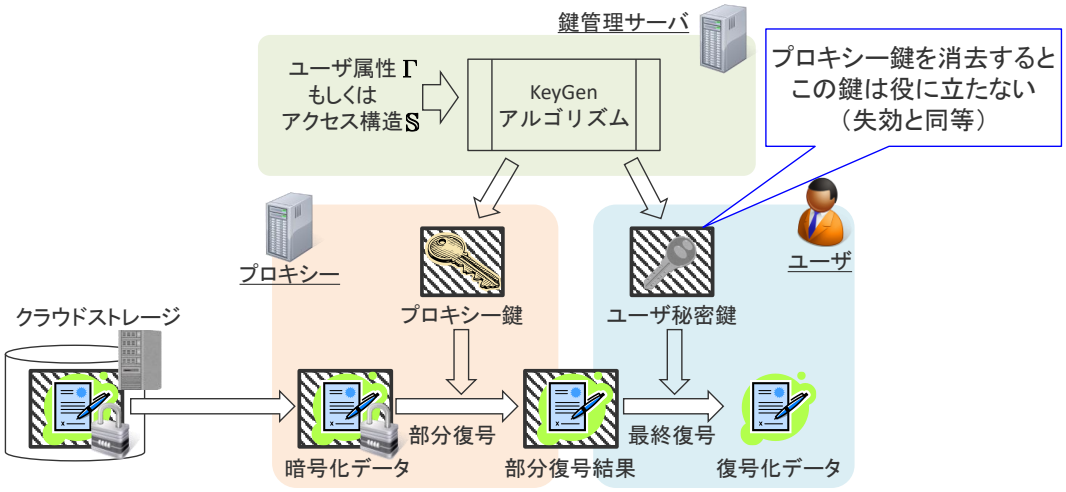

図 1.10: 属性ベース暗号のプロキシー支援による失効方式

には良い面が多くあるものの，KP-ABE と CP-ABE の双方について方式を示しており，かつ adaptive-secure で安全性証明が付与された方式が提案されていないという課題がある．

そこで本研究では，企業間での情報共有に適した検索可能暗号データベースの実現のため，上記にて示したユーザ秘密鍵失効と検索性能向上という課題を解決する方式を提案する．具体的には，下記に示す 2 つの方式を提案する．

1. ユーザ秘密鍵失効の実現
   所属などの属性が変わって新しいユーザ秘密鍵が発行されたユーザが，昔の所属で生成されたユーザ秘密鍵を使って暗号化データを復号できないようにするため，最もシステム適用時の運用負荷が少ないと考えられるプロキシー支援によるアプローチで，ユーザ秘密鍵の失効を実現する．高島ら [65] によって提案された関数型暗号をベースとして，元方式の復号鍵をプロキシー鍵とユーザ秘密鍵に分割することで，プロキシー鍵を持ったプロキシーサーバと，ユーザ秘密鍵を持ったユーザによる協調処理によって復号処理を実施する．ユーザ秘密鍵を失効させる場合は，プロキシーサーバが管理するプロキシー鍵を削除することで実現する．そして，既存研究よりもさらに高い adaptive-secure な状況下で安全性が証明できることを示す．

2. 検索性能の向上
   検索性能向上のためには，データベースの索引機能が有効となるような検索可能暗号の実現が必要である．そこで，k 匿名性のアイデアを検索可能暗号に融合することで，暗号化タグからユーザが指定したビット数の情報をサーバが取得できるようにすることで，サーバでの索引生成を可能とする．また，識別不可能性の定義を緩和し，ユーザが指定したビット数以上の情報が漏れないことを定式化し，そのモデルにおいて安全性を証明できることを示す．同時に，内積述語暗号を用いて階層型 ID を実現することで任意の ID にマッチするワイルドカードを実現することで，検索できるユーザが一人ではなく，検索権限を複数のユーザで共有できるグループ共有機能の実現も行っ

た．さらに，実現した検索可能暗号の索引生成の仕組みが，一般的なデータベースへと容易に組み込みができる事も示す．

1.2 本論文の構成

本論文の構成は次のとおりである．第 1 章では，本研究の背景と目的を述べた．第 2 章では，検索可能暗号データベースの実現に使われる関数型暗号（属性ベース暗号），および検索可能暗号に関するこれまでの研究状況や，検索可能暗号データベースの実現に向けた課題について整理する．その後，本研究が着目したユーザ秘密鍵失効，および検索性能向上の課題を具体的に述べ，本研究の位置づけを明確化する．第 3 章では，関数型暗号の復号鍵失効方式について KP-ABE および CP-ABE の両社に対して方式を提案し，adaptive-secure な安全性モデルで安全性証明が与えられることを示す．その後，第 4 章では，検索性能と安全性のバランスをとることができ，かつマルチユーザ化を図った索引生成対応検索可能暗号について提案方式を述べ，既存のデータベースへの適用方式についても考察する．最後に，第 5 章で本論文のまとめを述べる．

第2章　検索可能暗号データベースに関する既存研究と本研究の位置づけ

検索可能暗号データベースの実現においては，データを暗号化するための暗号化技術，暗号化データを検索するための検索可能暗号技術の双方が必要となる．ここでは，それぞれについて既存研究を紹介したのち，本研究の位置づけについて述べる．

2.1　暗号化技術

データ暗号化の歴史は古く，古代ギリシア時代のシーザー暗号 [101] が起源と言われている．その後，エニグマ暗号や米国標準暗号 DES(Data Encryption Standard)[62] や AES(Advanced Encryption Standard)[63] などが提案されているが，これらは全て暗号化と復号に用いる鍵が同じである共通鍵暗号と呼ばれる方式である．暗号化と復号で同じ共通鍵を用いるため，共通鍵は互いに秘密裏に共有・保管しておき，この秘密の共通鍵を持つユーザ同士でしか通信ができないという特徴がある．

その後，Diffie ら [30] によって公開鍵暗号の概念が提案され，暗号化技術にパラダイムシフトが起こった．公開鍵暗号は，暗号化に用いる公開鍵と，復号に用いる秘密鍵があり，それぞれ異なる値を取る．公開鍵と秘密鍵には数学的な関係性があるため，値としては異なる鍵となっているにも関わらず，公開鍵で暗号化された暗号化データは，秘密鍵で復号が可能となる．そして，公開鍵から秘密鍵を計算することができないという特性を持たせることで，公開鍵は秘密裏に管理する必要が無く，誰でも暗号化が行えるようになった．Diffie らの提案では，公開鍵暗号の概念を満たす提案方式が示されていなかったが，その後に RSA 暗号 [69] が提案され，Rabin 暗号 [68]，ElGamal 暗号 [33]，楕円曲線暗号 [49][60] と様々な方式が提案されていった．

公開鍵暗号は公開鍵を一般に配布することで誰でも暗号化できることであるが，一方で公開鍵はランダムに生成されたものであるため，公開鍵を見ただけでは誰の鍵か分からないという課題もあった．そこで，PKI(Public-Key Infrastructure)[99] という技術が発達していった．PKI では，CA(Certificate Authority) という信頼できる第三者機関を置き，CA が公開鍵の所有者を証明する証明書 [25] を発行することで，公開鍵と利用者の対応付に信頼を与える仕組みとなっている．また，万が一秘密鍵が漏洩したときには公開鍵を無効化する必要があることから，CRL(Certificate Revocation List) と呼ばれる失効した公開鍵のリストを発行したり，オンラインで失効を確認できる OCSP(Online Certificate Status Protocol) サーバ [73] などの仕組みも実現されている．そして，これらの仕組みを活用して，TLS(Transport Layer Security)[29] と呼ばれる暗号認証通信方式が実現され，今のインターネット上のセキュリティ機能として標準的に使われるようになっている．

一方，研究レベルでは PKI という重厚なインフラが無くても公開鍵の所有者を確認でき

る仕組みとして，ID ベース暗号が提案されている．ID ベース暗号は検索可能暗号とも深く関連しているため，以下の節で詳細に述べる．

2.1.1 ID ベース暗号

ID ベース暗号が最初に考案されたのは Shamir らの研究 [77] による．従来の RSA 暗号などの公開鍵暗号は，公開鍵がランダムに生成されるために誰の公開鍵かを証明するための公開鍵基盤（PKI）が必要となった．ID ベース暗号は，例えばメールアドレスの様なユーザにユニークに割り当てられている ID など，任意の文字列を公開鍵として利用できることが特徴である．メールアドレスは一般的に名刺交換やディレクトリサービスなどの別手段によって共有することができるため，公開鍵基盤がなくとも相手の公開鍵が容易に入手でき，誰の公開鍵かを容易に識別できることが特徴である．なお，公開鍵 (メールアドレスなど) で暗号化した暗号化データを復号するためには，対応する秘密鍵の生成が必要となる．この秘密鍵を生成するために鍵管理サーバ (PKG:private key generator) が存在し，公開鍵の正当性を確認して秘密鍵を発行する必要がある．例えばメールアドレスを公開鍵として利用する場合は，鍵管理サーバが対応する秘密鍵を生成してメールにてユーザに送付することで，メールアドレスの確認と鍵配送が同時に実施できるようになる．

実用に耐えうるだけの十分な安全性を備えた ID ベース暗号は，Boneh，Franklin らによる方式 [18] と考えている．本方式は，楕円曲線暗号に対する攻撃手法として提案された MOV reduction[59] にて使われていたペアリング写像を用いて構成した方式であるが，その安全性はランダムオラクルモデルで証明されている．ただし，安全性証明に不備があったため，その後に Galindo らによって証明の修正版が出されている．

安全性証明を改良した方式として，Boneh，Boyen らによる方式 [14] が提案されている．本論文では，decisional Bilinear Diffie-Hellman 仮定 (DBDH 仮定) に基づく方式 1 と，より強い安全性仮定である q-decisional BDH Inversion 仮定に基づく代わりに効率化が図られた方式 2 が提案されている．Boneh-Franklin らによる方式はランダムオラクルモデルで安全性が証明されていたが，本論文は両方式ともに selective-ID という制約は設けているが，標準モデルにて安全性が証明されている．

更に安全性証明を改良した方式として，同じく Boneh，Boyen らによる方式 [15] が提案されている．本論文で提案した ID ベース暗号は，効率的とは言えない方式だが，DBDH 仮定に基づいて初めて adaptive-secure な状況で安全性が証明できることを示しており，その研究の意義は大きい論文である．その後，Waters[86] によって，Boneh-Boyen らによる方式 [14] を改良して，効率的かつ adaptive-secure な ID ベース暗号が提案された．本方式は，DBDH 仮定に基づき，その安全性が証明されているが，帰着の効率が悪いという課題も残っていた．そこで，Gentry ら [37] によって，効率的かつ adaptive-secure な ID ベース暗号で，安全性仮定への帰着効率が改善された方式が提案された．ただし，本方式で利用している安全性仮定は q-ABDHE 仮定と言う強い仮定を用いていることから，暗号そのものの強度が高いかどうかは未知である．

2.1.2 階層型 ID ベース暗号

ID ベース暗号の発展形として，階層型 ID ベース暗号が発展してきた．ID ベース暗号は，鍵生成を行う鍵管理サーバが 1 台で，全てのユーザに対して秘密鍵を発行する必要があった．階層型 ID ベース暗号は，通常の PKI にて CA が階層構造をなすのと同様に，鍵管理サーバを階層的に設置できる点が特徴である．階層型 ID ベース暗号では，ID として $\{ID_1, \ldots, ID_n\}$ の様な n 階層の文字列を用いる．トップの鍵管理サーバは ID_1 に対応する秘密鍵を生成し，2 階層目の ID に対応する鍵管理サーバに秘密鍵を渡す．2 階層目の鍵管理サーバは $\{ID_1, ID_2\}$ に対応する秘密鍵を発行することができる．このようにすることで，複数企業にまたがった大規模な利用を行う場合でも，企業ごとに社員に対して秘密鍵を発行するなど，分散的な鍵発行・管理を行うことができる様になる．

階層型 ID ベース暗号は，Horwitz ら [45] や Gentry ら [39] によって方式が提案された．Horwitz らは，階層型 ID ベース暗号の概念を定義し，ID 階層を 2 階層に拡張した方式を提案した．そして，ランダムオラクルモデルにて，その安全性を証明した．しかし，2 層目のユーザの Collusion については，ある閾値を超えた場合には対応できないという課題もあった．Gentry らは，Boneh-Franklin の ID ベース暗号を拡張し，任意階層に対応可能な階層型 ID ベース暗号を提案した．本方式は，ランダムオラクルモデルにて安全性を証明し，更に Horwitz らの課題であった Collusion についても解決した方式である．

その後，Boneh ら [16] によって安全性証明を改良した階層型 ID ベース暗号が提案された．本方式は，ランダムオラクルモデルでは adaptive secure で証明ができるが，更に selective-ID という制約を付けることで標準モデルでも安全性を証明することができる方式である．Boyen ら [20] も selective-ID という制約の下で標準モデルで安全性を証明した方式を提案したが，さらに Bellare ら [8] によって提案された Key Privacy という性質を満たすことが特徴である．Chatterjee ら [22] は，Waters[86] によって提案された adaptive-secure な ID ベース暗号を拡張し，adaptive-secure な階層型 ID ベース暗号を提案した．そして，Genrtry ら [38] によって，多項式オーダーの階層数にて adaptive-secure な安全性証明が付けられる階層型 ID ベース暗号が提案された．

また，階層型 ID ベース暗号は ID 列が完全に一致するときのみ復号できるため，個人宛にデータを暗号化することしかできなかった．そこで，Abdalla ら [2] によって，ID としてワイルドカードを指定できる方式が提案されている．ある階層の ID_i としてワイルドカード '*' が指定されている場合，ワイルドカードはいかなる文字列ともマッチするようになっているため，複数のユーザ秘密鍵が ID 列に一致することとなり，グループ宛にデータを暗号化することが可能となった．

2.1.3 内積述語暗号

内積述語暗号は，ID ベースをさらに高機能に発展させたものである．ID ベース暗号は，暗号化時に指定した ID と，秘密鍵を発行したときに指定した ID が同一であるときに限って，暗号化データの復号が成功する．一方，内積述語暗号は，暗号化時に指定した属性ベクトル $\vec{x}$ と，秘密鍵を発行したときに指定した述語ベクトル $\vec{v}$ の内積値が 0 になるときに限って，暗号化データの復号が成功する．そのため，ID ベース暗号よりも柔軟な復号権限の制御が可能となる．また，ID ベース暗号が階層型 ID ベース暗号に拡張されたのと同様に，内積述語暗号を階層型内積述語暗号に拡張することができる．この場合，暗号化と秘密鍵生

成時に，それぞれ属性ベクトル列 $\{\overrightarrow{x_1}, \ldots, \overrightarrow{x_n}\}$ と述語ベクトル列 $\{\overrightarrow{v_1}, \ldots, \overrightarrow{v_n}\}$ を取り，それぞれの内積値が全て 0 になるときに限って復号に成功する．

Katz ら [48] は，新しく Predicate Encryption(PE) と言う概念を提案し，初めて内積述語暗号を提案した．PE は ID ベース暗号の概念を汎用化した概念である．データの暗号化は，属性集合 Σ から選んだ属性 $I \in \Sigma$ を指定して暗号化する．ユーザ秘密鍵は，述語集合 $\mathcal{F}$ から選んだ述語 f を指定して生成する．そして，$f(I) = 1$ を満たす場合に限って，述語 f を指定して生成したユーザ秘密鍵で，属性 I を指定して生成した暗号化データの復号に成功する暗号方式である．更に，payload-hiding と attribute-hiding という概念の定義も行った．payload-hiding は，暗号文 C から平文 M に関する情報が漏れないことを保証する．attribute-hiding は，key privacy[8] の概念を PE に拡張したものであり，暗号文 C から属性 I に関する情報が漏れないことを保証する．そして，Katz らは合成数位数のペアリングを利用して，内積述語暗号を実現した．ただ，合成数位数のペアリングは，素数位数のペアリングに比べて鍵長が大きくなってしまうため，効率的ではないという課題があった．

効率的な階層型内積述語暗号は，高島ら [64] によって初めて提案された．本方式は双対ペアリングベクトル空間という新しい実現方式を用いて，ベクトル空間でデータを扱うことを可能にすることで，内積述語暗号の実現を可能とした．その安全性については，RDSP(Decisional Subspace Problem with Relevant Dual Vector Tuples) 仮定と IDSP(Decisional Subspace Problem with Irrelevant Dual Vector Tuples) 仮定によって証明されているが，selective-attribute であるという制約がある．一方，key privacy と同様に暗号化に使った属性ベクトルの秘匿性を保証した attribute-hiding という性質も満たす．その後，高島ら [53] によって，adaptive-secure な階層型内積述語暗号へと拡張が行われた．

2.1.4 属性ベース暗号

属性ベース暗号 (ABE:Attribute-Based Encryption) は，PE の一種であり，内積述語暗号よりも汎用的な復号条件の指定が可能である．その復号条件の指定の仕方に応じて，key policy 型と ciphertext policy 型に分けることができる．key policy 型 (KP-ABE) は，暗号化する際にデータに付与する属性集合 $\Gamma \in \Sigma$ を指定して暗号化を実施する．ユーザ秘密鍵は，復号可能な条件を属性を用いた条件式であるアクセス構造 $\mathbb{S} \in \mathcal{F}$ を指定して生成する．多くの属性ベース暗号では，AND/OR 条件を用いてアクセス構造を指定することができる．そして，属性集合 Γ がアクセス構造 $\mathbb{S}$ を満たす場合に限って，暗号化データの復号が可能となる．一方，ciphertext policy 型 (CP-ABE) は，暗号化データ生成時にアクセス構造 $\mathbb{S} \in \mathcal{F}$ を指定し，ユーザ秘密鍵を生成する際に属性集合 $\Gamma \in \Sigma$ を指定する点が異なる．

最初の属性ベース暗号は，Sahai ら [71] によって提案された Fuzzy ID ベース暗号である．本方式は，秘密分散と ID ベース暗号を組合わせた方式であり，ユーザ秘密鍵にはユーザの属性集合と閾値からなるアクセス構造が埋め込まれる．一方，暗号化データには受信者が満たすべき属性集合を指定する．そして，暗号化データとユーザ秘密鍵の属性集合の積集合の大きさが閾値以上の場合に復号ができる方式である．本方式では通常の (k, n) 閾値法による秘密分散が利用されているため，AND/OR などの条件式を使うことができない．

Goyal ら [41] によって提案された方式では，より柔軟なツリー構造による復元条件が指定できる秘密分散法を利用しているため，AND/OR 条件を使ったアクセス構造を指定できる KP-ABE が実現された．本方式の安全性は，selective-attribute という制約が付くものの標

準モデルで DBDH 仮定に帰着できることが示されている．

Ostrovsky ら [66] は，Broadcast Encryption で使われている失効のテクニックを KP-ABE に適用することで，AND/OR 条件に加えて NOT 条件もサポートできる属性ベース暗号を提案した．本論文では，KP-ABE での実現方式が示されているが，以下で述べる Bethencourt ら [11] の CP-ABE に対しても NOT 条件を拡張できることが示唆されている．また，安全性は selective-attribute という制約の下で標準モデルで DBDH 仮定期に帰着できることが示されている．

Bethencourt ら [11] は，Goyal ら [41] の方式と同様に AND/OR 条件を使ったアクセス構造を指定できる CP-ABE 方式を提案した．秘密分散としては，Monotone Span Program(MSP) と呼ばれる方式を採用している．本方式の安全性は，Generic 群モデルという攻撃者に対して強い制約を課しているものの，adaptive-secure な状況下で安全性を証明している．

Waters[87] は，Linear Secret Sharing Scheme(LSSS) を使うことで，AND/OR 条件を使ったアクセス構造を指定でき，暗号化の効率を改善した CP-ABE 方式を提案した．さらに，selective-attribute の制約の下で，標準モデルで decisional Parallel BDHE 仮定に帰着できることが示されている．

Lewko ら [53] は，Waters[88] や Lewko ら [54] によって階層型 ID ベース暗号の安全性証明を adaptive-secure にするためのテクニックとして提案された Dual System Encryption を用い，adaptive-secure な属性ベース暗号の構成方法を提案した．本方式は，AND/OR 条件が利用でき，KP-ABE と CP-ABE の双方が実現されているが，合成数位数のペアリングを用いているため，素数位数のペアリングに比べて鍵長が大きくなってしまい，効率的ではないという課題があった．

その後，高島ら [65] によって，彼らの提案した内積述語暗号 [53] と LSSS を組合わせた属性ベース暗号が提案された．本方式も Lewko ら [53] と同様に Dual System Encryption を用いて安全性証明が構成されており，DLIN 仮定に基づいて adaptive-secure な状況下で標準モデルでその安全性が示されている．更に，素数位数ペアリングを用いているため，Lewko らの方式に比べて計算効率も改善されている方式である．

なお，いずれの方式も鍵管理サーバ (Key Authority) が全ユーザの秘密鍵を発行する方式であるため，鍵管理サーバは全ての暗号化データを復号する権限を持っている．そのため，実運用においては，信頼できる運用者によって，不正を行わないように運用する必要が生じる．そこで，鍵管理サーバを複数用意し，属性 1 の秘密鍵を鍵管理サーバ 1 が発行，属性 2 の秘密鍵を鍵管理サーバ 2 が発行，というように権限分散を図る Multi-authority な属性ベース暗号も提案されている [21]，[57]．これらの方式により，鍵管理サーバが全ての暗号化データを復号できないように権限を制約できるようになる．

2.1.5 秘密鍵失効

前節まで，ID ベース暗号から属性ベース暗号に至るまで，機能向上と安全性の向上が図られてきたことを述べた．しかし，いずれの方式も復号時の条件をきめ細かく制御できることにフォーカスしており，秘密鍵漏洩時の失効については考慮されていなかった．本節では，上記とは独立に発展してきた属性ベース暗号の秘密鍵失効技術について述べる．

初めに，既存の属性ベース暗号をそのまま活用し，システム的に失効を行う方式が考えられる．一つの方式として，クラウド側で秘密鍵を管理しておき，ユーザ認証を行った後に，

クラウド側で復号処理を行う仕組みが考えられる．この場合，秘密鍵は常にクラウド内に保管されているため，ユーザの属性が変更された場合に古い秘密鍵を消去することで，確実に秘密鍵を失効することができる．しかし，秘密鍵がクラウド上に保管され，復号処理もクラウド上で行われることになるため，データの機密保護の観点からは暗号化を用いているメリットが失われてしまう．

別の方式として，ユーザのログイン認証時に秘密鍵をオンライン配布し，端末側は秘密鍵をファイルに保管せずにアプリケーション内 (メモリ上) で復号に利用し，ログオフした際に秘密鍵を消去する方式が考えらえる．ログインのたびに，常に最新の属性に対応した秘密鍵が生成・ダウンロードされるため，属性の変更に追随することができる．しかし，万が一マルウェア感染などによってメモリ上の秘密鍵が漏洩した場合，その秘密鍵を失効することはできない．また，利用者が独自プログラムを開発して，ダウンロードした鍵をファイルに保存する事も可能であり，内部犯罪に対して脆弱であるという課題がある．

そのため，暗号アルゴリズムとして失効機能を実現する研究がなされている．既存の秘密鍵の失効技術については，大きく 2 つのアプローチがあることが知られている [6]．

- Direct Revocation データを暗号化する時点で失効しているユーザを指定して，そのユーザが復号できないように暗号化する．NOT 演算をサポートしている属性ベース暗号であれば，本アプローチにて失効処理ができるが，一方で誰が失効されているかという情報を全てのユーザが把握しておかねばならず，さらに暗号化した後に秘密鍵が失効した場合には対応できないという課題がある．

- Indirect Revocation データ暗号化時には失効ユーザを指定する必要が無く，それ以外の仕組みを使って秘密鍵を失効する．データを暗号化した後でも，秘密鍵を失効させて復号ができないようにすることができるが，秘密鍵失効のための仕組みが複雑化するケースが多い．

また，Indirect Revocation を提案した既存研究では，大きく下記のようなアプローチが提案されている．

1. 秘密鍵に有効期限を付与して自然失効

2. 秘密鍵や暗号文を更新して復号権限を削除

3. 復号処理にプロキシーを介在させる

上記 1. のアプローチは，Bethencourt ら [11] が属性ベース暗号の方式を提案している中で提案している (図 2.1)．具体的には，データを暗号化する際には，暗号化した時刻 Y を属性として指定して暗号化を行う．その際，時刻の大小比較が実施できるように，AND/OR ゲートを駆使して属性を含めるようにする．ユーザ秘密鍵には，その秘密鍵がいつまで有効かを示す時刻 X を指定する．そして，暗号化データの時刻 Y，ユーザ秘密鍵の時刻 X が，$X \geq Y$ を満たすときのみ，復号できるような仕組みとした．これにより，ユーザ秘密鍵に有効期限（時刻 X）を付与することができ，時刻 X より後に作成された暗号化データが復号できないようにすることができる．しかし，時刻 X より前に作成された暗号化データは依然として復号できるままであり，ユーザ秘密鍵を柔軟に失効させることができないという課題がある．

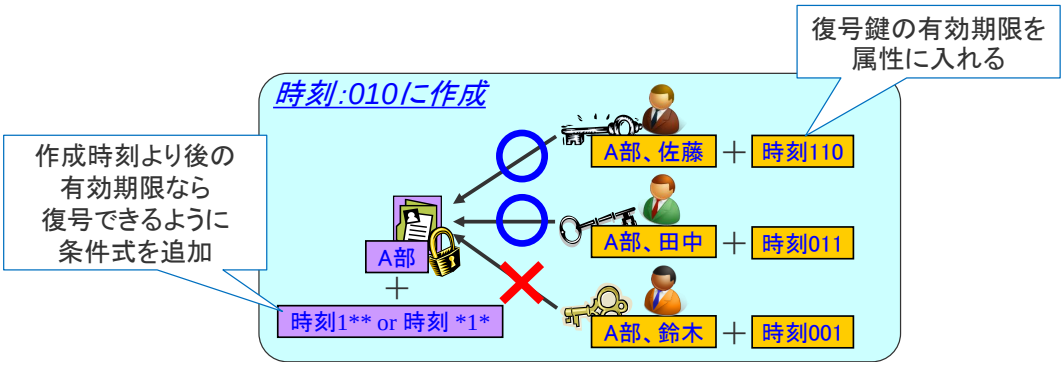

図 2.1: 属性ベース暗号の鍵有効期限を付与する失効方式

上記 2. のアプローチは，上記 1. のアプローチの欠点を改良すべく，任意のタイミングで秘密鍵の失効を行えるようにして，さらに過去に復号できていた暗号化データも復号できないようにすることを目指した方式である．実現方式は方式ごとに異なるが，大きくは暗号化データとユーザ秘密鍵に世代番号を付与して，両者の世代番号が一致するときのみ復号ができる様に暗号化を行う (図 1.9)．ユーザの秘密鍵を失効させたい場合，世代番号を +1 して，サーバに保管されたすべての暗号化データの世代番号を +1 するように暗号文更新を行う．同様に，ユーザ秘密鍵のうち失効していないユーザ秘密鍵に関して世代番号を +1 して秘密鍵更新を行い，失効させるユーザ秘密鍵に関しては世代番号の更新を行わない．これにより，失効したユーザ秘密鍵では世代番号が不一致となり復号できなくなるため，ユーザ秘密鍵を失効することができる．一方で，全ての暗号化データと失効していないユーザ秘密鍵を更新する必要が生じるため，暗号化データ数やユーザ数が多い場合，また更新頻度が高い場合などは，システムの運用に大きな負荷がかかるという問題点がある．

このアプローチの論文としては，Attrapadung ら [6] による方式が知られている．本方式は，KP-ABE において Direct Revocation/Indirect Revocation の両方をサポートした方式である．Indirect Revocation モードでは，誰か一人のユーザ秘密鍵が失効した場合，時刻を更新して失効していないユーザに対して鍵更新情報を送付する．ただし，暗号文の時刻を更新する仕組みは提供されていないため，過去に復号できていた暗号化データを復号できないようにする仕組みは提供されていない．また，安全性証明も selective-attribute の制約の下でしか付与されていない．その後，Sahai ら [70] によって過去に復号できていた暗号化データの復号もできないようにする仕組みが提案された．この方式では，ciphertext delegation という仕組みが提案されており，暗号化データに付与されたポリシーをより制約の強いポリシーに変換することができる．この仕組みにより，暗号化データに最初に暗号化された時刻情報が埋め込まれ，いずれかのユーザ秘密鍵が失効されるたびに時刻情報を更新していくことができる様になった．また，いずれかのユーザ秘密鍵が失効されるたびに，失効していないユーザ秘密鍵に埋め込まれた時刻情報も更新することで，失効したユーザ秘密鍵では過去に復号できていた暗号化データすら復号できない仕組みが実現できた．そして，予稿では KP-ABE について実現方式が示されているが，CP-ABE でも実現できることが示唆されている．また，安全性証明としては adaptive-secure で付与されているが，提案方式が Lewko ら [53] の方式をベースに作られているため，合成数位数のペアリングを用いており効率が悪いという課題は残る．

上記 3. のアプローチは，上記 2. において全ての暗号文や失効していない全てのユーザ秘密鍵の更新が必要という課題を解決すべく，プロキシーを活用することで効率化を図った方式である (図 1.10)．仕組みとしては，ユーザがクラウドストレージから暗号化データを取り出す際，プロキシーサーバを経由してアクセスするようにするが，このプロキシーサーバの力を借りることでユーザ秘密鍵の失効を行う方式である．具体的には，鍵管理サーバがユーザ秘密鍵を生成する際，同時にプロキシー鍵を生成してプロキシーサーバに配布する．暗号化データを復号するには，プロキシー鍵での部分復号，ユーザ秘密鍵での最終復号の 2 段階を経なければならないようになっている．ユーザがクラウドストレージから暗号化データを取り出す際，プロキシーサーバが部分復号を行うが，プロキシーサーバからプロキシー鍵を削除することでユーザ秘密鍵を使えなくする仕組みである．

このアプローチの論文としては，Yang らの方式が知られている [95]，[94]．この方式では，通常の属性ベース暗号の復号鍵をプロキシー鍵とユーザ秘密鍵に分割して 2 段階復号を実現した．しかし，Yang らの方式は CP-ABE でしか実現されておらず，KP-ABE での実現方式は示されていない．さらに安全性証明がジェネリック群モデルであるという制約がついている．Green ら [42] は，CP-ABE と KP-ABE の両方について実現方式を提案したが，安全性証明が selective-attribute という制約がある．また，Nomura ら [61] は，Multi-authority 環境の下で，プロキシー支援にて失効を実現する方式を提案している．しかし，Yang らの方式と同様に，CP-ABE についてのみ実現しており，KP-ABE での実現方式が示されていない点と，安全性証明が selective-attribute であるという制約がついている．Fan らが提案する方式 [35] も，Nomura らの方式と同様に CP-ABE のみ対応しており，安全性証明も selective-attribute であるという制約がついている．このように，上記 3. のアプローチは効率的には良い面が多くあるものの，KP-ABE と CP-ABE の双方について方式を示しており，かつ adaptive-secure で安全性証明が付与された方式は提案されていない．

2.2 検索可能暗号技術

検索可能暗号は，2000 年に Song ら [79] によって共通鍵ベースの方式が提案されたのが始まりである．本方式は，共通鍵とキーワードの組によって一意に定まるランダム関数を生成し，そのランダム関数からマスク値を生成する．このマスク値で，暗号化キーワードを XOR して，それをタグとする．検索を行う場合は，暗号化キーワードと前記ランダム関数を生成し，これを検索クエリとする．サーバ側では，検索クエリの暗号化キーワードと，タグの XOR を取り，マスク値を復元したのち，検索クエリ内のランダム関数を用いてマスク値が正しい値かどうかをチェックする．チェックにパスすれば，キーワードが同一だと見なす方式である．本方式は非常にシンプルではあるが，キーワードを復元することなく検索ができるという点で非常に画期的な論文である．ただし，共通鍵暗号やランダム関数などを用いているためキーワードが復元できないことを考察しているが，厳密な安全性証明は付与されていない．

本論文ののち，共通鍵ベースの方式と，公開鍵ベースの方式に分かれて発展を遂げていった．いずれの方式も，Song ら [79] の方式と同様に，データから取り出されたキーワードを検索可能暗号で暗号化してタグを生成し，データそのものは従来の暗号方式で暗号化したうえで，暗号化データとタグの組をサーバに保管する．そして，そのタグを暗号化したまま検索し，取り出した暗号化データを復号することで元データを得るものである．共通鍵ベース

の方式とは，暗号化に用いる鍵と，検索クエリを作成する際に用いる鍵が同じ方式を指す．言い換えれば，データを暗号化できるユーザは，必ず検索を行うことができることを意味する．共通鍵ベースの方式は，共通鍵暗号やハッシュ関数などの高速演算が可能なプリミティブを用いているため，検索処理が非常に速いというメリットがある．一方，公開鍵ベースの方式とは，暗号化に用いる鍵と，検索クエリを作成する際に用いる鍵が異なる方式で，暗号化は公開鍵を持っていれば誰でも実施できるが，検索クエリの生成は秘密鍵を持ったユーザのみが実施できるという特徴がある．そのため，企業内や企業間でのデータ交換や，不特定多数のユーザから情報を収集するケースなど，誰が暗号化するか限定できないようなケースでは公開鍵ベースの方式が必須となる．ただし，公開鍵ベースの方式の多くは，ペアリング暗号による実現がほとんどであることから，検索処理に時間がかかるという課題もある．本節では，企業間データ共有などの用途を想定した公開鍵ベースの方式に主眼を置いて説明する．

2.2.1 公開鍵ベースの検索可能暗号

公開鍵ベースの検索可能暗号は，Boneh ら [17] によって提案された．この方式では，検索を行うことができる検索ユーザがマスター公開鍵とマスター秘密鍵のペアを生成し，マスター公開鍵は一般に公開し，マスター秘密鍵は自分で秘密裏に保管する．データを暗号化するユーザは，検索ユーザからマスター公開鍵を受け取り，キーワードを暗号化して PEKS (Public-key encryption with Keyword Search) と呼ぶタグ (Tag) を生成し，暗号化データと共にサーバに保管する (図 1.10)．検索ユーザは，自身が管理するマスター秘密鍵と検索したいキーワードからトラップドア (Trapdoor) を生成し，サーバに検索クエリとして送付する．サーバは，保管しているタグと受け取ったトラップドアに対して一致判定処理を行い，タグを生成するときに指定したキーワードと，トラップドアを生成したときに指定したキーワードが同一かどうかを，キーワードを復号することなく判定することができる．

以上の様に，検索はマスター秘密鍵を持った検索ユーザしかできないが，誰でも暗号化できるという点が共通鍵ベースの方式とは大きく異なっている．また，Boneh らが提案した ID ベース暗号と同様にペアリングを用いて構成されており，その安全性は離散対数問題やペアリングの一方向性などに依存している．ただ本方式は，ランダムオラクルモデルに基づいて安全性が証明されている．

この論文の発表後，様々な検索可能暗号が提案されている [19][28][47][48][53]．例えば，Boneh ら [19] は，Hidden Vector Encryption(HVE) という方式を提案し，これにより柔軟な検索ができることを示している．HVE は，複数のキーワードをベクトル化し，AND 条件でベクトルが完全一致するかどうかを判定可能な検索可能暗号である．さらに，どのキーワードにも一致するワイルドカードを指定できるという特徴を持っている．これにより，複数キーワードを同時に検索する場合において，何個のキーワードが一致したかという部分情報を漏らすことなく，複数キーワードの同時検索を実現した．更に，この HVE の応用として，ベクトルの構成方法を工夫することで，例えば年齢のような有限範囲の整数値の大小比較や，集合の包含関係などを用いた検索ができることが示されている．

2.2.2 ID ベース暗号と検索可能暗号の関係性

Boneh らによって提案された PEKS と，ID ベース暗号には何らかの関係があると見られていたが，その関係性は Abdalla ら [1] によって解析がなされた．

最初に，ID ベース暗号をブラックボックスとして利用して，PEKS を構築できることが示された．ID ベース暗号のアルゴリズムが { Setup, KeyDer, Enc, Dec } であるとする．この時，PEKS(検索可能暗号) のアルゴリズムである鍵生成 KG，トラップドア生成 Td，タグ生成 PEKS，一致判定 Test の各アルゴリズムは次のように構成できることが示された．

- $KG(1^\lambda) \to (pk, sk)$
  $(pk, sk) \leftarrow Setup(1^\lambda)$,
  return (pk, sk)
- $Td(sk, w) \to Td_w$
  $Td_w \leftarrow KeyDer(sk, w)$,
  return Td_w
- $PEKS(pk, w) \to Tag_w$
  $r \xleftarrow{U} \{0,1\}^\lambda$,
  $C \leftarrow End(pk, w, r)$,
  return $Tag_w := (C, r)$
- $Test(Td_w, Tag_w) \to \{\text{true, false}\}$
  $r' \leftarrow Dec(pk, Td_w, C)$,
  return $(r' \stackrel{?}{=} r)$? true : false

直感的に言えば，保管するキーワードを ID(≒条件式) とし，乱数をデータとみなして，これを暗号化してタグを作る．タグには，暗号化した乱数と，暗号化データのペアを含める．そして，検索するキーワードを ID(≒属性) として秘密鍵を生成し，これをトラップドアとする．サーバは，タグ中の暗号化データをトラップドア (≒秘密鍵) で復号し，正しく乱数が復号できるかどうかを確認する．正しく乱数が復号できた場合，保管したキーワードと，検索するキーワードが同一であることが分かる．

さらに検索可能暗号として検討すべき性質としては，Security と Consistency の 2 つがあることが示された．Security は，検索用に作られたタグが与えられたときに，そのタグから元のキーワードについての情報が 1 ビットも得られないという安全性である．彼らは，これを PEKS-IND-CPA と定義した．そして，ID ベース暗号が key privacy を満たすなら，上記変換方式にて PEKS-IND-CPA が成立することを示した．一方 Consistency は，タグ生成時のキーワードと，トラップドア生成時のキーワードが一致する場合，その場合に限って一致判定で true と判定されるという一貫性に関する性質である．そして，ID ベース暗号が IND-CPA 攻撃者に対して安全であれば，上記変換方式にて構成した検索可能暗号が computationally consistent になることを示した．この構成により，確率的暗号である ID ベース暗号を用いて，確率的な検索可能暗号を構成することができる．そのため，同一のキーワードであっても，タグは毎回異なる値となり，さらにトラップドアも異なる値となるが，一致判定だけは実施できるという機能が実現できる．更に，ID ベース暗号だけでなく，階層型 ID ベース暗号，それらを高機能化した暗号を用いても，検索可能暗号が構成できることが分かった．

2.2.3 検索権限のグループ共有

検索権限のグループ共有とは，暗号化されたタグを検索できるユーザが一人でなく複数名に拡張された方式の事を指す．そのため，複数ユーザに拡張した multi-user 方式や，グループ名で暗号化できる group-oriented な方式などがある．筆者としては，multi-user 方式は複数ユーザ宛の暗号化であり，group-oriented 方式の方がグループに属するユーザが誰かを把握してなくても暗号化ができる点で柔軟性が高いと考えている．

multi-user に拡張した検索可能暗号として，Hwang ら [47] による方式が知られている．本方式は，タグを生成する際に検索ユーザを複数指定できる様に拡張することで，指定された検索ユーザなら誰でも検索できる方式である．single-user 版の方式ではタグが $\{A, B, C\}$ という 3 要素から構成されているが，multi-user に拡張した場合でも要素 B だけを複数ユーザ用に作成すれば良いため，タグサイズの効率化が図れる方式である．ただし，タグサイズが検索ユーザ数に比例するという点ではデータサイズの効率化はまだまだ課題がある．また，安全性証明がランダムオラクルモデルに基づいている点も改善の余地がある．

group-oriented な方式として，Hattori ら [43] による方式が提案されている．本方式は，Seo ら [76] によって提案された key privacy を満たす階層型 ID ベース暗号に対して，Abdalla ら [2] によって提案されたグループ宛に暗号可能な階層型 ID ベース暗号のテクニックを応用することで，group-oriented な検索可能暗号の構成を可能とした方式である．そのため，グループに属するユーザ数が何人いたとしてもタグサイズは一定であるというメリットがある．一方，合成数位数のペアリングを用いているため，素数位数のペアリングに比べて効率的ではないという課題があった．

2.2.4 検索性能の高速化

検索性能を高速化するにあたって，そのアプローチは以下の 2 つが考えられる．

1. 一致判定処理そのものを高速化するアプローチ

2. データ件数に比例した一致判定処理の回数削減のアプローチ

特に実利用においては，多くのデータが蓄積されるため，我々は後者のアプローチが必要と考えている．

検索性能の高速化の手法として，Shi ら [78] によって高次元範囲検索の手法が提案されている．この方式は，暗号化データの IND-CPA は保ちつつも，検索にヒットした場合に暗号化データの元データである数値情報が洩れる “match-revealing” と呼ばれる安全性の緩和を行うことで，一致判定処理が数値範囲 S の log オーダ $\mathrm{O}(\log|S|)$ で実施可能となった．しかし，一致判定処理そのものの高速化に着目した提案手法であるため，データ件数 N とした時に，データ件数に比例した検索処理時間 $\mathrm{O}(N \times \log|S|)$ がかかるという課題が残っている．

データ件数に比例した一致判定処理の回数削減のアプローチとしては，Bellare ら [9][10] によって提案された確定的暗号に基づく方式が知られている．確定的暗号は，同じデータを暗号化すれば必ず同じ暗号化データになるという方式であるため，タグはキーワードを公開鍵で暗号化して生成し，トラップドアも同じようにキーワードを公開鍵で暗号化して生成する (図 1.6)．サーバ側では，タグとトラップドアが同じバイナリデータになるものを探すだけで済むため，一致判定処理が軽量化される．更に，一般的なデータベース製品では，バイ

ナリ比較でデータを高速に検索できるように，インデックスと呼ばれる索引情報を生成する仕組みが用意されているため，その仕組みを使ってデータ件数 N に対して O(N) にて検索を行うことができる．また，タグからキーワードに関する情報が漏れないことを定義するため，従来から使われていた onewayness では不十分と考え，新しく PRIV-security という安全性定義を導入して，タグの安全性を証明している．ただし，検索クエリに相当するトラップドアも公開鍵から生成されるため，誰でも検索ができてしまうという欠点がある．言い換えれば，ランダムに選んだキーワードで検索して，タグの中のキーワードを推測するというブルートフォース攻撃が可能であることを意味しているため，本方式を安全に使うためには，ブルートフォース攻撃に対して安全になるようにキーワードのエントロピーがセキュリティパラメータと同程度に大きいことが求められる．一般的に，我々が使うキーワードは種類が大きいものの，エントロピーがセキュリティパラメータと同程度に大きくはないと予想されるため，理論的な面白さはあるものの実用的ではないと考えられる．

別の高速化手法が Popa ら [67] によって提案されている．この論文では，確定的暗号と確率的暗号で二重に暗号化するオニオン暗号化方式を提案し，検索の高速化と安全性の両立を図っている (図 1.7)．初期状態は確率的暗号で保護されているため，検索は実施できないが安全性が高い．もしキーワード検索を行いたい場合は，端末から DB サーバに対して一番外側の確率的暗号を確定的暗号に変換する鍵を渡すことで，安全性を落とす代わりに検索が実施できるようになる．一度確定的暗号に変換すれば，以降は高速に検索が行えるというアイデアである．これにより，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．

2.2.5 検索可能暗号のデータベースへの組込み

データの保管のためにデータベースを利用することが一般的と考えられるため，検索可能暗号で暗号化したデータも同様にデータベースに保管されると考えられる．データベースには文字列やバイナリデータを検索する機能はあるが，検索可能暗号には対応していないため，何らかの手段で検索可能暗号の一致判定処理を組み込むか，アプリケーション側で一致判定処理を実施しなければならない．

データベースへの検索可能暗号の組込みに関して，Popa ら [67] によって CryptDB と呼ぶデータベースが提案されている．検索可能暗号の一致判定処理については，DB サーバが用意しているユーザ定義関数（UDF：user-defined function) と呼ばれる仕組みを使って実装し，DB サーバが自動的に検索可能暗号の一致判定処理を呼び出せるように機能拡張している．また，データの暗号化については，CryptoDB proxy server と呼ばれる仕組みを実現し，アプリケーションサーバは暗号化を意識しなくても良い仕組みとしている．同様に，オニオン暗号化の皮をむくための鍵についても proxy server にて管理しており，クラウド側に DB サーバが，企業側に proxy server が置かれているような環境でなければ利用が難しい．一般的なクラウド活用では，アプリケーションサーバもクラウド側に置かれるため，そのようなケースではクラウド側でデータが平文で存在するタイミングがあったり，鍵が見えてしまうという課題がある．

また，鈴木ら [81] は検索可能暗号を Web アプリケーションに適用する一般的な構成方法を提案している．検索可能暗号をデータベースに組み込むために DB プラグインを用い，Web アプリケーションの改修を少なくするために DB Interface wrapper で検索可能暗号に対応

した特殊な SQL コマンドに変換し，利用者側の HTTP Proxy で鍵管理を行う，という構成を示し，共通鍵型の検索可能暗号にて実現方式を検証している．データの暗号化や検索などは，クライアントパソコンが設置される企業側のプロキシーにて実現する．本方式は公開鍵型の検索可能暗号にも適用できると考えられるが，共通鍵暗号型の検索可能暗号のみでしか検証は行われていない．

2.3 検索性能向上と鍵失効の課題と本研究の位置づけ

これまでに述べたように，検索可能暗号データベースの実現には，データを暗号化するための暗号化技術，暗号化データを検索するための検索可能暗号技術の双方が必要になる．これら技術を，実際のシステムに適用するまでには，ユーザ秘密鍵の失効，検索性能の向上，データのグループ共有，データベースへの組み込みなどの課題があることを述べた．本研究では，失効したユーザ秘密鍵を持つユーザが，データを復号できないようにする仕組みについて提案する．さらに，安全性と検索性能のバランスを利用者が調整することで検索性能を向上させつつ，データのグループ共有を実現する検索可能暗号技術を実現する．また，その技術をデータベースへ組み込むことで，システム適用を容易にする仕組みも実現する．以下，本研究の位置づけについて説明する．

2.3.1 ユーザ秘密鍵失効の課題

長期に保管されるデータベースにおいては，データベースに保管されている暗号化データを復号可能なユーザ秘密鍵が漏洩してしまい，機密が保護できなくなる可能性が生じる．そのため，暗号化時に失効チェックを行う既存のアプローチでは対応ができず，暗号化後にユーザ秘密鍵を失効できる仕組みが必要となる．

既存研究の一つのアプローチであるユーザ秘密鍵を更新して復号権限を削除するアプローチでは，任意のタイミングで特定のユーザ秘密鍵だけを失効させることができるものの，保管されている暗号化データをすべて変換する必要が生じたり，さらに失効していないすべてのユーザ秘密鍵も更新が必要という負荷がかかるため，理論的には安全性を担保しやすいものの，実用的ではないという課題があった．

一方，別のアプローチである復号処理にプロキシーを介在させる方式は，ユーザ秘密鍵の更新や暗号化データの更新が不要であることから実用的には十分であるが，その技術は発展途上であり，鍵ポリシー型属性ベース暗号（KP-ABE）と暗号文ポリシー型属性ベース暗号 (CP-ABE) の両方について，安全性証明を full-secure で達成した方式が提案されていないという課題があった．

2.3.2 検索性能向上の課題

現在提案されている検索可能暗号の多くは，暗号化されたタグから 1 ビットも情報が漏れないという識別不可能性を達成することを基本としており，さらに安全性をジェネリック群モデルやランダムオラクルモデルからスタートして，selective，adaptive-secure と改善を図るように発展してきた．暗号化データから一切の情報が漏れないことは暗号の安全性としては当たり前のことだが，裏を返せば一般的なデータベースが実施している索引生成が行え

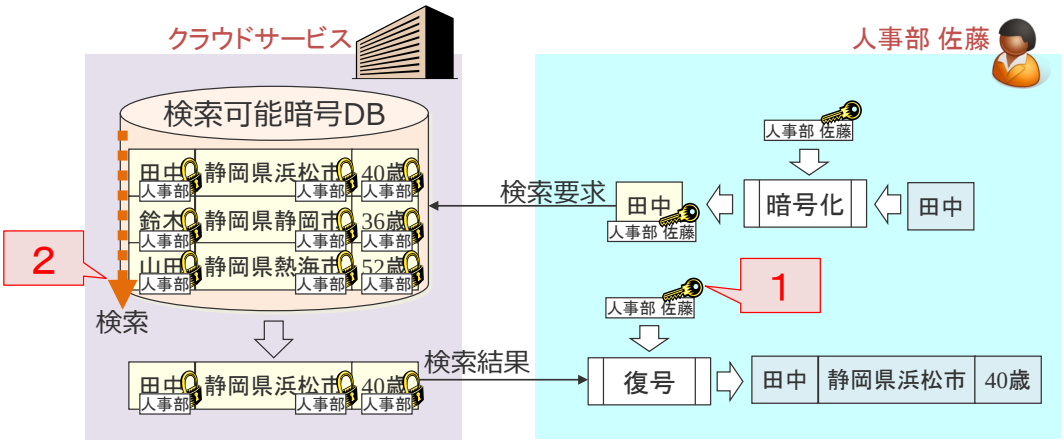

図 2.2: 検索可能暗号データベースの実現における本研究の位置づけ

ず，検索時にはすべてのデータをチェックしていかなければならないという課題が生じていた．そのため，保管されているデータ数に比例する検索処理量を改善するアプローチが必要であった．

この改善手法として，Bellare ら [9][10] の研究によって，識別不可能性の考え方を緩和させることで検索性能を向上できることが示されたが，平文空間のエントロピーがセキュリティパラメータと同程度に大きいことが前提となっていたため，理論的な面白さはあるものの実用的ではないという課題がある．そのため，実用的で，かつ検索性能の向上ができ，さらに必要以上の情報が漏れないことを証明できる仕組みの実現が課題であった．

2.3.3 本研究の位置づけ

本研究では，上記の 2 つの課題を解決した方式の実現を目指すものである (図 2.2)．

1. ユーザ秘密鍵失効の実現
   グループ共有機能を実現した検索可能暗号では，特定一人のユーザ秘密鍵を失効させることが困難である．そこで，もっともシステム適用時の運用負荷が少ないと考えられるプロキシー支援によるアプローチで，検索可能暗号データベースのユーザ秘密鍵の失効を実現することを目指す．最初に，暗号化されたデータを復号するためのユーザ秘密鍵を失効させる仕組みとして，既存技術よりも高い安全性を達成可能な方式の実現を目指す．

2. 検索性能の向上
   検索性能向上のためには，データベースの索引機能が有効となるような検索可能暗号の実現が必要である．そのため，Bellare らが識別不可能性を緩和させた PRIV-security という考え方を導入したのとは別のアプローチで，識別不可能性を緩和させた安全性モデルを定義し，さらに索引生成を可能とする検索可能暗号の実現を目指す．同時に，検索できるユーザが一人ではなく，検索権限を複数のユーザで共有できるグループ共有機能の実現も目指す．さらに，実現した検索可能暗号の索引生成の仕組みが，一般的なデータベースへと容易に組み込みができる方式の実現も目指す．

第3章　プロキシー支援による関数型暗号の復号鍵失効機能の実現

3.1　要件概要と想定ユースケース

3.1.1　要件概要

データ暗号化の際に条件を指定でき，条件を満たすユーザのみが暗号化データを復号できる関数型暗号（Functional Encryption）や属性ベース暗号（Attribute-based Encryption）を用いると，暗号化データに復号権限を設定することができるため，機密度や公開範囲の異なる様々なデータを扱う際に都合が良く，これまでに様々な方式が提案されている [11]，[40]，[53]，[65]，[89].

一方，企業では異動・退職に伴ってアクセス権が変化するケースや，復号鍵を紛失するケースなどが考えられるため，復号鍵の失効処理，すなわち “それまで読めていたデータを読めなくする処理” が必要となる．失効について，公開鍵と復号鍵が一対一に対応する RSA 暗号や ID ベース暗号では様々な方式が提案されている [46]，[75]，[80]，[83]，[91]．また，複数の受信者が存在する放送型暗号でも，暗号化する時に失効ユーザを受信者から外す仕組みが研究されている [5]，[50]，[90]．しかし，関数型暗号や属性ベース暗号の場合，データを復号できるユーザが複数存在しうるが，そのうち特定ユーザだけを失効できなければならない．さらに，過去に復号できていた暗号化データも復号できないようにする必要があり，失効を困難なものとしている．例えば，暗号化データが “A 部の社員であれば誰でも復号可能” という条件 (ポリシー) で暗号化されている場合を考える．従来，A 部メンバーの A 氏，B 氏，C 氏の 3 人ともに復号できたものが，C 氏が異動になった場合に C 氏の復号鍵だけ失効して，A 氏，B 氏は引き続き復号できるようにしなければならない．このように複数の復号鍵の一部だけを，任意の時点から無効化できることが必要である．

そこで，属性ベース暗号の失効方式についても様々な研究がなされている．そのアプローチは Direct Revocation と Indirect Revocation の 2 つに大別できる [6]．Direct Revocation は，暗号化時に失効ユーザを指定することで，失効ユーザが復号できないように暗号化する方式である．一方，Indirect Revocation は，暗号化時に誰が失効しているかを意識する必要が無く，暗号化データの変換や，失効していないユーザ秘密鍵の更新などにより，失効ユーザが復号できないようにする方式である．どちらの方式が良いかはユースケース次第だが，企業機密の保護・保管というユースケースを考慮すると，暗号化した後に復号鍵が漏洩する可能性にも対応できなければならず，Indirect Revocation のアプローチが適している．

そこで我々は，関数型暗号に対して，次に示す要件を満たす失効機能が必要と考えた．

要件 1:　暗号化時に失効されたユーザが誰かを意識しないで良いこと

要件 2:　復号できていた暗号化データが復号できなくなること

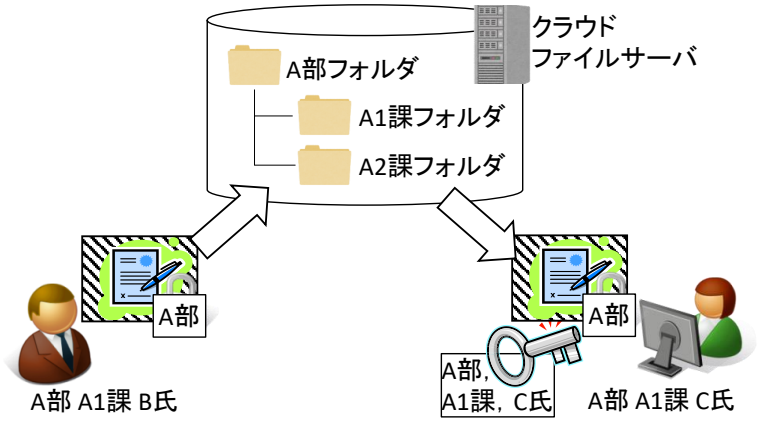

図 3.1: ファイルサーバのユースケース

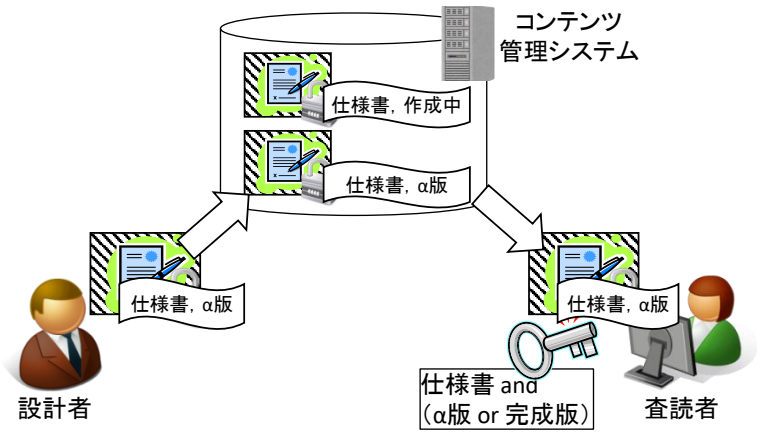

図 3.2: エンタープライズコンテンツ管理システムのユースケース

要件 3: 失効に伴い，クラウド側の処理が不要もしくは軽微であること

要件 4: 失効していないユーザの復号鍵は従来通り使えること

3.1.2 想定ユースケース

企業におけるクラウド活用による情報管理のユースケースについて例を交えて説明する．図 3.1 は，一般的なファイルサーバをクラウド上に移行したユースケースである．クラウドファイルサーバ上には，部や課の単位でフォルダが作成され，その部や課に所属するユーザにアクセス権限が与えられる．データを作成したユーザ（図中 “A 部 A1 課 B 氏”）は，そのデータを誰と共有するかを判断し，例えば “A 部” 内で共有したい場合は，“A 部” のフォルダにデータを置く．このようなケースでは，データを暗号化する際には例えば “A 部” の様に誰が閲覧できるかというアクセス権限を指定して暗号化を行い，ユーザには例えば “A 部, A1 課, C 氏” の様な属性を含む復号鍵を配布する暗号文ポリシー型関数型暗号の利用が適している．

別のユースケースとして，図 3.2 にエンタープライズコンテンツ管理システムのユースケースを示す．このユースケースでは，前述のファイルサーバと異なり，クラウド上のレポジトリにデータが蓄積される．この時，例えば “仕様書，α 版” の様に，データの属性を表すタグを付与して保管する．データを閲覧するユーザには，業務内容などを加味して例えば “仕様書 and （α 版 or 完成版）” の様にアクセスできるデータの条件がアクセス権限として付与されており，そのアクセス権限に合致する属性のデータを検索・閲覧することができる．このユースケースでは，データに付与するタグを属性として暗号化し，ユーザに配布する復号鍵にはアクセス権限を指定できる鍵ポリシー型関数型暗号の利用が適する．

以上の様に，データ管理方法に応じて，暗号文ポリシー型関数型暗号（CP-FE もしくは CP-ABE）と鍵ポリシー型関数型暗号（KP-FE もしくは KP-ABE）を使い分ける必要があることから，鍵ポリシー型と暗号文ポリシー型の両者で復号鍵の失効を実現できることが好ましい．

3.1.3 関連研究

関数型暗号の復号鍵失効に関して，様々な方式が提案されている．Bethencourt らは，CP-ABE を対象に，復号鍵に有効期限を付けて失効を行う方式を提案している [11]．本方式では，復号鍵に有効期限を入れ，暗号化データには暗号化時刻を入れることで，有効期限が切れた復号鍵では復号ができないようにする．しかし，復号鍵の有効期限が切れても，それまで復号できていた暗号化データは引き続き復号でき，さらに復号鍵を任意のタイミングで失効できないなどの課題があり，要件 2 を満たさない．また，復号鍵の有効期限と暗号化時刻の大小比較ができるようにアクセス構造を設定するとアクセス構造が複雑化しやすく暗号化や復号に時間がかかったり，復号鍵の定期的な更新が必要になるなどの課題もあり，要件 4 を満たさない．

Attrapadung らは，ユースケースに応じて Direct Revocation と Indirect Revocation を使い分けられるように，それらを統合した方式を提案している [6]．この方式では，暗号化する際に Direct/Indirect Revocation を指定でき，なおかつユーザは一つの秘密鍵でいずれの暗号化データも復号できるようになる．しかし，Direct モードでは，暗号化時に失効されているユーザが誰かを指定しなければならず要件 1 を満たさない．また，Indirect モードでは，暗号化データとユーザ秘密鍵に時刻情報を埋め込み，時刻情報が一致したときだけ復号ができる．失効されていないユーザは秘密鍵を更新して復号できる時刻を延長していくが，過去に復号できていた暗号化データを復号できなくなる仕組みはなく，要件 2 を満たさない．

Yu らは，復号鍵と暗号化データにバージョン情報を埋め込むことで失効を行う方式を提案している [97]．この方式では，失効していない復号鍵のバージョン番号をあげて，さらに暗号化データのバージョン番号を Proxy Re-encryption で更新することで，バージョン番号が更新されていない失効した復号鍵が使えないようにする．しかし，復号鍵の失効が生じるたびに暗号化データを変換する必要があることや，失効していないユーザの復号鍵をすべて更新する必要があることから，要件 3 や要件 4 を満たさない．Li ら [55] は，ユーザが失効するたびにグループ公開鍵を更新し，同時にユーザの復号鍵とクラウド上の暗号化データを更新することで失効を実現する仕組みを提案しているが，Yu らの方式と同様に暗号化データの変換が必要という課題がある．Wang ら [85] によって，階層型 ID ベース暗号と属性ベース暗号方式を組み合わせた方式も提案されているが，本方式も同様に失効時に暗号化データの変換が

必要である．Li らの方式 [56]，Yang らの方式 [93]，Yu らの方式 [96] では，失効される属性に関連する暗号化データに限定して再暗号化を行うように効率化を図ったものの，依然として暗号化データの変換が必要という課題は残ったままである．Sahai ら [70] は，公開情報だけを用いて暗号化データのアクセス権限に制限を加えて行く仕組みを提案し，失効ユーザが復号できないようにする Revocable-Storage Attribute-Based Encryption を実現した．しかし，秘密鍵を失効させるために，クラウド上の全ての暗号化データを確認し，失効した秘密鍵で復号可能な暗号文を変換していく必要があるため要件 3 を満たさない．Lee ら [52] は，暗号化データのアクセス権限に制限を加える変更が可能な Ciphertext Delegatable Encryption，それを用いて復号可能な時刻を制約可能な Self-Updatable Encryption という新しいプリミティブを定義した．これらを用いて，Revocable-Storage Attribute-Based Encryption を構築する方式を提案した．しかし，Sahai らの方式と同様に，失効した秘密鍵で復号可能な暗号文を全て変換していく必要があり要件 3 を満たさない．Yamada ら [92] は，属性ベース暗号に対して失効機能を付加する構成法として，Pair encoding framework に基づく方式と，ブール式に基づく属性ベース暗号を拡張する方式を提案した．しかし，いずれの方式も暗号化時に失効されたユーザのリストを指定する必要があり，要件 1 や要件 2 を満たさない．

Yang らは，プロキシー支援の下で効率的に失効を行う方式を提案している [95]，[94]．この方式では，通常の復号鍵をプロキシー鍵とユーザ秘密鍵に分割して 2 段階復号を行うようにして，プロキシー鍵を削除して対応するユーザ秘密鍵を使えなくすることで失効を実現する．本方式は，復号鍵を任意のタイミングで失効でき，クラウド上の暗号化データを再暗号化する必要もなく，失効されないユーザ秘密鍵を更新する必要がないう利点があり，要件 1，2，3，4 をすべて満たす．しかし，Yang らの方式は CP-ABE でしか実現されておらず，KP-ABE での実現方式は示されていない．さらに安全性証明がジェネリック群モデルであるという制約がついている．Green らは，CP-ABE と KP-ABE の両方について実現方式を提案しているが，安全性証明が selective という制約がある [42]．

また，Nomura らは，Multi-authority 環境の下で，プロキシー支援にて失効を実現する方式を提案している [61]．しかし，Yang らの方式と同様に，CP-ABE についてのみ実現しており，KP-ABE での実現方式が示されていない点と，安全性証明が selective であるという制約がついている．Fan らが提案する方式 [35] も，Nomura らの方式と同様に CP-ABE のみ対応しており，安全性証明も selective であるという制約がついている．

3.1.4 我々の貢献

本論文では，3.1.1 節で示した要件 1〜4 を満たしつつ，adaptive 状況下にて安全性証明が可能な方式を提案する．具体的には，企業内データを外部のクラウドサービスに保管する状況を想定し，人事部や情報システム部門によってユーザ秘密鍵が発行される Single-Authority モデルの関数型暗号が適すると考え，高島ら [65] が提案した鍵ポリシー型および暗号文ポリシー型の方式を採用する．これをベースとして，失効の即時性，クラウドサーバの負荷低減に優れ，要件 1〜4 を満たすことができるプロキシー支援型のアプローチに基づいて，失効機能付き鍵ポリシー型関数型暗号と失効機能付き暗号文ポリシー型関数型暗号を実現する．これは，関数型暗号の復号鍵を要素ごとに分解して，プロキシー鍵とユーザ秘密鍵に分割し，プロキシー鍵とユーザ秘密鍵のペアが無ければ暗号化データが復号できないようにすることで実現する．

また，あらゆる暗号化データを入手できるクラウドサーバ，全ユーザのプロキシー鍵を入手できるプロキシーサーバ，失効したユーザ秘密鍵を持つユーザの 3 者を攻撃者と想定し，このいずれの攻撃者であっても暗号化データの識別ができないことを示す安全性モデルを定義する．そして，本モデルに基づいて提案方式が adaptive な攻撃者に対して安全であることを示す．

著者の知る限り，プロキシー支援アプローチで復号鍵失効を行う方式で，鍵ポリシー型と暗号文ポリシー型の両方のアルゴリズムを実現し，さらに adaptive な攻撃者に対して安全性証明が付与された方式は無く，その点が本節の貢献である．

3.1.5 本章の構成

始めに 3.2 節にて，高島ら [65] らによって提案された関数型暗号の構成に必要な DPVS や安全性仮定などの定義について示す．次に 3.3 節では，想定するシステムモデルと，3.1.1 節で述べた要件の詳細について説明する．3.4 節にて提案方式である失効機能付き鍵ポリシー型関数型暗号，および失効機能付き暗号文ポリシー型関数型暗号のアルゴリズムを示し，3.5 節にて実システム適用時の鍵運用方法について示す．最後に，3.6 節にて本提案方式が要件を満たすことの考察を行う．

3.2 アルゴリズムの構成に用いる既存方式

高島ら [65] による DPVS や安全性仮定について示す．

3.2.1 表記方法

A を確率変数とした時，確率変数 A の分布に従ってランダムに要素 y を選ぶことを $y \xleftarrow{\mathsf{R}} A$ と書く．A が集合の時，一様分布にて要素 y を選ぶことを $y \xleftarrow{\mathsf{U}} A$ と書く．y を z によって定義するとき，$y := z$ と書く．値 a を何らかの定数としたとき，アルゴリズム A が入力 x によって値 a を出力することを $A(x) \to a$ と書く．任意の多項式 $f(x)$ およびセキュリティパラメータ λ に対して，確率 p が $1/f(\lambda)$ より小さいとき，確率が negligible であるという．

素数 q が与えられたとき，位数 q の有限体を $\mathbb{F}_q$ と記す．また，有限体 $\mathbb{F}_q$ から零元を取り除いたものを $\mathbb{F}_q^{\times} := \mathbb{F}_q \backslash \{0\}$ と記す．有限体 $\mathbb{F}_q$ 上で定義された n 次元ベクトル $(x_1, \ldots, x_n) \in \mathbb{F}_q^n$ を $\overrightarrow{x}$ と記す．特別に全ての要素が 0 からなる n 次元ベクトルを $\overrightarrow{0}$，全ての要素が 1 からなる n 次元ベクトルを $\overrightarrow{1}$ と記す．二つのベクトル $\overrightarrow{x} = (x_1, \ldots, x_n)$，$\overrightarrow{v} = (v_1, \ldots, v_n)$ の内積 $\sum_{i=1}^{n} x_i v_i$ を $\overrightarrow{x} \cdot \overrightarrow{v}$ と記す．行列 X の転置行列を X^T と記す．また，$\ell \times \ell$ の単位行列を I_ℓ と記す．ベクトル空間 $\mathbb{V}$ の要素を太字にて $\boldsymbol{x} \in \mathbb{V}$ と記す．ベクトル $\boldsymbol{b}_i \in \mathbb{V} (i = 1, \ldots, n)$ が与えられたとき，それらのベクトルで張られる部分空間の事を $\mathsf{span}\langle \boldsymbol{b}_1, \ldots, \boldsymbol{b}_n \rangle \subseteq \mathbb{V}$ と記す．また，ベクトル $\overrightarrow{x} = (x_1, \ldots, x_n)$ と基底 $\mathbb{B} := (\boldsymbol{b}_1, \ldots, \boldsymbol{b}_n)$ が与えられたとき，ベクトルの線形和 $\sum_{i=1}^{n} x_i \boldsymbol{b}_i$ を $(\overrightarrow{x})_{\mathbb{B}} = (x_1, \ldots, x_n)_{\mathbb{B}}$ と記す．属性フォーマット $\overrightarrow{n} := (d; n_1, \ldots, n_d)$ が与えられたとき，全ての $t = 1, \ldots, d$ に対して標準基底 $(\overbrace{0, \ldots, 0}^{j-1}, 1, \overbrace{0, \ldots, 0}^{n_t - j})$ を $\overrightarrow{e}_{t,j}$ と記す．

3.2.2 Dual Pairing Vector Spaces(DPVS)

関数型暗号の基礎となる DPVS について，その定義と性質を述べる．始めに，対称ペアリング群の定義を示す．

Definition 1. 対称ペアリング群 $(q, \mathbb{G}, \mathbb{G}_T, g, e)$ は，素数 q，位数 q の有限巡回乗法群 $\mathbb{G}$ と $\mathbb{G}_T$，群 $\mathbb{G}$ の生成元 g，多項式時間で計算可能な非退化ペアリング写像 $e : \mathbb{G} \times \mathbb{G} \to \mathbb{G}_T$ から構成される．ここで，$e(g^s, g^t) = e(g, g)^{st}$ かつ $e(g, g) \neq 1$ が成り立つ．アルゴリズム $\mathcal{G}_{\mathsf{bpg}}$ を，セキュリティパラメータ 1^λ を入力として受け取り，対称ペアリング群 $(q, \mathbb{G}, \mathbb{G}_T, g, e)$ を生成する関数として定義する．

次に，対称ペアリング群を用いた DPVS の定義を示す．

Definition 2. 対称ペアリング群 $(q, \mathbb{G}, \mathbb{G}_T, g, e)$ の直積で構成される *"Dual pairing vector spaces (DPVS)"* $(q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$ は，素数 q，$\mathbb{F}_q$ 上で定義される N 次元ベクトル空間 $\mathbb{V} := \overbrace{\mathbb{G} \times \cdots \times \mathbb{G}}^{N}$，位数 q の巡回群 $\mathbb{G}_T$，N 次元ベクトル空間 $\mathbb{V}$ の標準基底 $\mathbb{A} := (\boldsymbol{a}_1, \ldots, \boldsymbol{a}_N)$，ペアリング写像 $e : \mathbb{V} \times \mathbb{V} \to \mathbb{G}_T$ で構成される．ここで，$\boldsymbol{a}_i := (\overbrace{1, \ldots, 1}^{i-1}, g, \overbrace{1, \ldots, 1}^{N-i})$ である．

ペアリング写像は，2つのベクトル $\boldsymbol{x} := (g_1, \ldots, g_N) \in \mathbb{V}$ と $\boldsymbol{y} := (h_1, \ldots, h_N) \in \mathbb{V}$ があったとき，$e(\boldsymbol{x}, \boldsymbol{y}) := \prod_{i=1}^{N} e(g_i, h_i) \in \mathbb{G}_T$ と定義する．このペアリング写像は非退化写像であり，$e(s\boldsymbol{x}, t\boldsymbol{y}) = e(\boldsymbol{x}, \boldsymbol{y})^{st}$ となる．また，全ての $\boldsymbol{y} \in \mathbb{V}$ について $e(\boldsymbol{x}, \boldsymbol{y}) = 1$ が成り立つなら，$\boldsymbol{x} = \boldsymbol{0}$ となる．また，$g_T := e(g, g) \neq 1 \in \mathbb{G}_T$ であり，全ての i, j に対して $e(\boldsymbol{a}_i, \boldsymbol{a}_j) = g_T^{\delta_{i,j}}$ が成り立つ．ここで，$\delta_{i,j}$ はクロネッカーのデルタである．

DPVS 生成アルゴリズム $\mathcal{G}_{\mathsf{dpvs}}$ は，セキュリティパラメータ 1^λ，次元 $N \in \mathbb{N}$ を入力として受け取り，$(\mathbf{param}_{\mathbb{V}} := (q, \mathbb{V}, \mathbb{G}_T, \mathbb{A}, e)$ を生成する．

関数型暗号では，DPVS 上で定義されるランダム双対直行基底を用いるので，その生成アルゴリズムの定義を示す．

Definition 3. *"Random dual orthonormal bases generator"* $\mathcal{G}_{\mathsf{ob}}$ は，下記のように定義される．

$$
\begin{aligned}
&\mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n} := (d; n_1, \ldots, n_d)) : \\
&\quad \mathbf{param}_{\mathbb{G}} := (q, \mathbb{G}, \mathbb{G}_T, g, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bpg}}(1^\lambda), \\
&\quad \psi \xleftarrow{\mathsf{U}} \mathbb{F}_q^\times, N_0 := 5, N_t := 3n_t + 1 \ \textit{for}\ t = 1, \ldots, d, \\
&\quad \textit{for}\ t = 0, \ldots, d; \\
&\qquad \mathbf{param}_{\mathbb{V}_t} \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{dpvs}}(1^\lambda, N_t, \mathbf{param}_{\mathbb{G}}), \\
&\qquad X_t := \begin{pmatrix} \overrightarrow{\chi}_{t,1} \\ \vdots \\ \overrightarrow{\chi}_{t,N_t} \end{pmatrix} := (\chi_{t,i,j})_{i,j} \xleftarrow{\mathsf{U}} GL(N_t, \mathbb{F}_q),
\end{aligned}
$$

$$\begin{pmatrix} \overrightarrow{\vartheta}_{t,1} \\ \vdots \\ \overrightarrow{\vartheta}_{t,N_t} \end{pmatrix} := (\vartheta_{t,i,j})_{i,j} := \psi \cdot (X_t^T)^{-1},$$

$$\boldsymbol{b}_{t,i} := (\overrightarrow{\chi}_{t,i})_{\mathbb{A}_t} = \sum_{j=1}^{N_t} \chi_{t,i,j} \boldsymbol{a}_{t,j} \ \textit{for}\ i = 1, \ldots, N_t,$$

$$\boldsymbol{b}^*_{t,i} := (\overrightarrow{\vartheta}_{t,i})_{\mathbb{A}_t} = \sum_{j=1}^{N_t} \vartheta_{t,i,j} \boldsymbol{a}_{t,j} \ \textit{for}\ i = 1, \ldots, N_t,$$

$$\mathbb{B}_t := (\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,N_t}), \mathbb{B}^*_t := (\boldsymbol{b}^*_{t,1}, \ldots, \boldsymbol{b}^*_{t,N_t}),$$

$$g_T := e(g,g)^{\psi}, \mathbf{param}_{\overrightarrow{\pi}} := (\{\mathbf{param}_{\mathbb{V}_t}\}_{t=0,\ldots,d}, g_T),$$

$$\textit{return}\ (\mathbf{param}_{\overrightarrow{\pi}}, \{\mathbb{B}_t, \mathbb{B}^*_t\}_{t=0,\ldots,d}).$$

なお，全ての $t = 0, \ldots, d; i = 1, \ldots, N_t$ に対して，$e(\boldsymbol{b}_{t,i}, \boldsymbol{b}^*_{t,i}) = g_T$ となる．

3.2.3 スパンプログラムとアクセス構造

アクセス構造の表現のために non-monotone なスパンプログラムを用いる．その定義を下記に示す．

Definition 4. $\{p_1, \ldots, p_n\}$ を変数の集合とする．この時，有限体 $\mathbb{F}_q$ 上で定義されるスパンプログラムは，$\mathbb{F}_q$ 上の $(l \times r)$ 行列 M を用いて $\hat{M} := (M, \rho)$ と定義される．ここで，ρ はラベリングを行う写像であり，行列 M の行と，集合 $\{p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n\}$ の要素との対応を取る．

また，入力列 $\delta \in \{0,1\}^n$ に対して，行列 M の部分行列 M_δ を定める．これは，行列 M の各 j 行目の要素 M_j に対して，$\rho(j) = p_i$ かつ $\delta_i = 1$，もしくは $\rho(j) = \neg p_i$ かつ $\delta_i = 0$ となる場合を $\gamma(j) = 1$ と定義し，$M_\delta := (M_j)_{\gamma(j)=1}$ にて得ることができる．

スパンプログラム $\hat{M}$ が δ を受け入れるとは，$\overrightarrow{1} \in \mathsf{span}\langle M_\delta \rangle$ が成り立つとき，ただその時だけである．スパンプログラムを用いることで，入力 δ を受け入れるときのみ $f(\delta) = 1$ となるブール関数を構築することができる．なお，ラベリング写像 ρ が集合 $\{p_1, \ldots, p_n\}$ にしか写像しない場合，スパンプログラムは *monotone* であるという．一方，集合 $\{p_1, \ldots, p_n, \neg p_1, \ldots, \neg p_n\}$ に写像される場合，*non-monotone* であるという．

このスパンプログラムを用いて，属性がベクトル表現で表されるときのアクセス構造を下記のように定義する．本論文では，non-monotone なスパンプログラムを用いる．

Definition 5. アクセス構造 $\mathbb{S}$ はスパンプログラム $\hat{M} := (M, \rho)$ であり，変数 p は $t \in \{1, \ldots, d\}, \overrightarrow{v} \in \mathbb{F}_q^{n_t} \backslash \{\overrightarrow{0}\}$ のとき $(t, \overrightarrow{v})$ で表され，写像 ρ は $\rho : \{1, \ldots, \ell\} \to \{(t_1, \overrightarrow{v_1}), (t_2, \overrightarrow{v}_2), \ldots, \neg(t_1, \overrightarrow{v}_1), \neg(t_2, \overrightarrow{v}_2), \ldots\}$ で定義される．

属性集合 Γ は，集合 $\{1, \ldots, d\}$ の部分集合の要素 t を用いて，$\Gamma := \{(t, \overrightarrow{x}_t) | \overrightarrow{x}_t \in \mathbb{F}_q^{n_t} \backslash \{\overrightarrow{0}\}, 1 \leq t \leq d\}$ と表される．

属性集合 Γ とアクセス構造 $\mathbb{S}$ が与えられたとき，スパンプログラム $\hat{M} := (M, \rho)$ に対する写像 γ を次のように定義する．$i = 1, \ldots, \ell$ に対して，もし $[\rho(i) = (t, \overrightarrow{v}_i)] \wedge [(t, \overrightarrow{x}_t) \in$

$\Gamma] \wedge [\overrightarrow{v}_i \cdot \overrightarrow{x}_t = 0]$ または $[\rho(i) = \neg(t, \overrightarrow{v}_i)] \wedge [(t, \overrightarrow{x}_t) \in \Gamma] \wedge [\overrightarrow{v}_i \cdot \overrightarrow{x}_t \neq 0]$ が成り立つ場合，$\gamma(i) = 1$ とし，それ以外の場合は $\gamma(i) = 0$ とする．

アクセス構造 $\mathbb{S} := (M, \rho)$ が属性集合 Γ を受け入れるとは，$\overrightarrow{1} \in \mathsf{span}\left\langle (M_i)_{\gamma(i)=1} \right\rangle$ が成り立つ時を指す．

このスパンプログラムを用いて，秘密分散法を定義する．

Definition 6. スパンプログラム $\hat{M} := (M, \rho)$ に対する秘密分散法は下記のように構成される．

1. M を $l \times r$ 行列とする．また，$\overrightarrow{f}^T := (f_1, \ldots, f_r)^T \xleftarrow{\mathsf{U}} \mathbb{F}_q^r$ とする．$s_0 := \overrightarrow{1} \cdot \overrightarrow{f}^T = \sum_{k=1}^{r} f_k$ を分散したい秘密値としたとき，$\overrightarrow{s}^T := (s_1, \cdots, s_\ell)^T := M \cdot \overrightarrow{f}^T$ は ℓ 個の秘密分散値からなるベクトルである．また，各秘密分散値 s_i は $\rho(i)$ と対応する．
2. アクセス構造 $\mathbb{S}$ が属性集合 Γ を受け入れる場合，すなわち $\gamma : \{1, \ldots, \ell\} \to \{0, 1\}$ に対して $\overrightarrow{1} \in \mathsf{span}\left\langle (M_i)_{\gamma(i)=1} \right\rangle$ が成り立つ場合，$I \subseteq \{i \in \{1, \ldots, \ell\} | \gamma(i) = 1\}$ かつ $\sum_{i \in I} \alpha_i s_i = s_0$ となる定数 $\{\alpha_i \in \mathbb{F}_q | i \in I\}$ が存在する．

3.2.4 安全性仮定

始めに，安全性を帰着させる DLIN 仮定について述べる．

Definition 7. *Decisional Linear Assumption*（*DLIN*仮定）とは，$(\mathsf{param}_{\mathbb{G}}, g, g^{\xi}, g^{\kappa}, g^{\delta\xi}, g^{\sigma\kappa}, Y_\beta) \xleftarrow{\mathsf{R}} \mathcal{G}_\beta^{\mathsf{DLIN}}(1^\lambda)$ が与えられたとき，$\beta \in \{0, 1\}$ を推測する問題である．ここで，$\beta \xleftarrow{\mathsf{U}} \{0, 1\}$ に対して，

$$
\begin{aligned}
&\mathcal{G}_\beta^{\mathsf{DLIN}}(1^\lambda) : \\
&\quad \mathsf{param}_{\mathbb{G}} := (q, \mathbb{G}, \mathbb{G}_T, g, e) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{bpg}}(1^\lambda), \\
&\quad \kappa, \delta, \xi, \sigma \xleftarrow{\mathsf{U}} \mathbb{F}_q, Y_0 := g^{(\delta+\sigma)}, Y_1 \xleftarrow{\mathsf{U}} \mathbb{G}, \\
&\quad \textit{return } (\mathsf{param}_{\mathbb{G}}, g, g^{\xi}, g^{\kappa}, g^{\delta\xi}, g^{\sigma\kappa}, Y_\beta).
\end{aligned}
$$

確率的アルゴリズム $\mathcal{E}$ に対して，$\mathcal{E}$ のアドバンテージを下記のように定義する．

$$
\begin{aligned}
\mathsf{Adv}_{\mathcal{E}}^{\mathsf{DLIN}}(\lambda) \quad := \quad & \left| \mathsf{Pr}\left[\mathcal{E}(1^\lambda, \varrho) \to 1 \,\middle|\, \varrho \xleftarrow{\mathsf{R}} \mathcal{G}_0^{\mathsf{DLIN}}(1^\lambda) \right] \right. \\
& \left. - \mathsf{Pr}\left[\mathcal{E}(1^\lambda, \varrho) \to 1 \,\middle|\, \varrho \xleftarrow{\mathsf{R}} \mathcal{G}_1^{\mathsf{DLIN}}(1^\lambda) \right] \right|
\end{aligned}
$$

ここで，確率的アルゴリズム $\mathcal{E}$ が多項式時間アルゴリズムの場合，アドバンテージ $\mathsf{Adv}_{\mathcal{E}}^{\mathsf{DLIN}}(\lambda)$ は *negligible* となる．

失効機能付き関数型暗号では，Desicional Liniear Assumption に帰着可能な下記 2 つの安全性仮定をもちいて安全性を証明する．その定義を示す．

Definition 8. *Problem 1* は，$(\mathbf{param}_{\overrightarrow{n}}, \mathbb{B}_0, \hat{\mathbb{B}^*_0}, \boldsymbol{e}_{\beta,0}, \{\mathbb{B}_t, \hat{\mathbb{B}^*_t}, \boldsymbol{e}_{\beta,t,1}, \boldsymbol{e}_{t,i}\}_{t=1,\ldots,d;i=2,\ldots,n_t}) \xleftarrow{\mathsf{R}} \mathcal{G}^{\mathsf{P1}}_\beta(1^\lambda, \overrightarrow{n})$ が与えられたときに，$\beta \in \{0,1\}$ を推測する問題である．ここで，$\beta \xleftarrow{\mathsf{U}} \{0,1\}$ に対して，

$$
\begin{aligned}
&\mathcal{G}^{\mathsf{P1}}_\beta(1^\lambda, \overrightarrow{n}) : \\
&\quad (\mathbf{param}_{\overrightarrow{n}}, \{\mathbb{B}_t, \mathbb{B}^*_t\}_{t=0,\ldots,d}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n}), \\
&\quad \hat{\mathbb{B}^*_0} := (\boldsymbol{b}^*_{0,1}, \boldsymbol{b}^*_{0,3}, \ldots, \boldsymbol{b}^*_{0,5}), \\
&\quad \hat{\mathbb{B}^*_t} := (\boldsymbol{b}^*_{t,1}, \ldots, \boldsymbol{b}^*_{t,n_t}, \boldsymbol{b}^*_{t,2n_t+1}, \ldots, \boldsymbol{b}^*_{t,3n_t+1}) \\
&\qquad\qquad \textit{for } t = 1, \ldots, d, \\
&\quad \omega, z_0, \gamma_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \\
&\quad \boldsymbol{e}_{0,0} := (\omega, 0, 0, 0, \gamma_0)_{\mathbb{B}_0}, \boldsymbol{e}_{1,0} := (\omega, z_0, 0, 0, \gamma_0)_{\mathbb{B}_0}, \\
&\quad \textit{for } t = 1, \ldots, d; \\
&\qquad \overrightarrow{e}_{t,1} := (1, 0^{n_t - 1}) \in \mathbb{F}_q^{n_t}, \overrightarrow{z}_t \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n_t}, \gamma_t \xleftarrow{\mathsf{U}} \mathbb{F}_q, \\
&\qquad \boldsymbol{e}_{0,t,1} := (\omega \overrightarrow{e}_{t,1}, 0^{n_t}, 0^{n_t}, \gamma_t)_{\mathbb{B}_t}, \\
&\qquad \boldsymbol{e}_{1,t,1} := (\omega \overrightarrow{e}_{t,1}, \overrightarrow{z}_t, 0^{n_t}, \gamma_t)_{\mathbb{B}_t}, \\
&\qquad \boldsymbol{e}_{t,i} := \omega \boldsymbol{b}_{t,i} \textit{ for } i = 2, \ldots, n_t, \\
&\quad \textit{return } (\mathbf{param}_{\overrightarrow{n}}, \mathbb{B}_0, \hat{\mathbb{B}^*_0}, \boldsymbol{e}_{\beta,0}, \\
&\qquad\qquad \{\mathbb{B}_t, \hat{\mathbb{B}^*_t}, \boldsymbol{e}_{\beta,t,1}, \boldsymbol{e}_{t,i}\}_{t=1,\ldots,d;i=2,\ldots,n_t}).
\end{aligned}
$$

確率的アルゴリズム $\mathcal{B}$ に対して，$\mathcal{B}$ のアドバンテージを下記のように定義する．

$$
\begin{aligned}
\mathsf{Adv}^{\mathsf{P1}}_{\mathcal{B}}(\lambda) \;:=\; &\left| \mathsf{Pr}\left[\mathcal{B}(1^\lambda, \varrho) \to 1 \,\middle|\, \varrho \xleftarrow{\mathsf{R}} \mathcal{G}^{\mathsf{P1}}_0(1^\lambda, \overrightarrow{n}) \right] \right. \\
&\left. - \mathsf{Pr}\left[\mathcal{B}(1^\lambda, \varrho) \to 1 \,\middle|\, \varrho \xleftarrow{\mathsf{R}} \mathcal{G}^{\mathsf{P1}}_1(1^\lambda, \overrightarrow{n}) \right] \right|
\end{aligned}
$$

Lemma 1. *Problem1* に対する確率的多項式時間攻撃者 $\mathcal{B}$ が存在するならば，*DLIN* 仮定に対する確率的多項式時間アルゴリズム $\mathcal{E}$ が存在する．攻撃者 $\mathcal{B}$ のアドバンテージは，任意のセキュリティパラメータ λ に対して $\mathsf{Adv}^{\mathsf{P1}}_{\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{E}}(\lambda) + (d+6)/q$ となる．

Definition 9. *Problem 2* は，$(\mathbf{param}_{\overrightarrow{n}}, \hat{\mathbb{B}_0}, \mathbb{B}^*_0, \boldsymbol{h}^*_{\beta,0}, \boldsymbol{e}_0, \{\hat{\mathbb{B}_t}, \mathbb{B}^*_t, \boldsymbol{h}_{\beta,t,i}, \boldsymbol{e}_{t,i}\}_{t=1,\ldots,d;i=1,\ldots,n_t}) \xleftarrow{\mathsf{R}} \mathcal{G}^{\mathsf{P2}}_\beta(1^\lambda, \overrightarrow{n})$ が与えられたときに，$\beta \in \{0,1\}$ を推測する問題である．ここで，$\beta \xleftarrow{\mathsf{U}} \{0,1\}$ に対して，

$$
\begin{aligned}
&\mathcal{G}^{\mathsf{P2}}_\beta(1^\lambda, \overrightarrow{n}) : \\
&\quad (\mathbf{param}_{\overrightarrow{n}}, \{\mathbb{B}_t, \mathbb{B}^*_t\}_{t=0,\ldots,d}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n}), \\
&\quad \hat{\mathbb{B}_0} := (\boldsymbol{b}_{0,1}, \boldsymbol{b}_{0,3}, \ldots, \boldsymbol{b}_{0,5}), \\
&\quad \hat{\mathbb{B}_t} := (\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,n_t}, \boldsymbol{b}_{t,2n_t+1}, \ldots, \boldsymbol{b}_{t,3n_t+1}) \\
&\qquad\qquad \textit{for } t = 1, \ldots, d, \\
&\quad \delta, \delta_0, \omega \xleftarrow{\mathsf{U}} \mathbb{F}_q, \tau, u_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q^\times, z_0 := u_0^{-1}, \\
&\qquad \begin{pmatrix} \overrightarrow{z}_{t,1} \\ \vdots \\ \overrightarrow{z}_{t,n_t} \end{pmatrix} := Z_t \xleftarrow{\mathsf{U}} GL(n_t, \mathbb{F}_q) \textit{ for } t = 1, \ldots, d,
\end{aligned}
$$

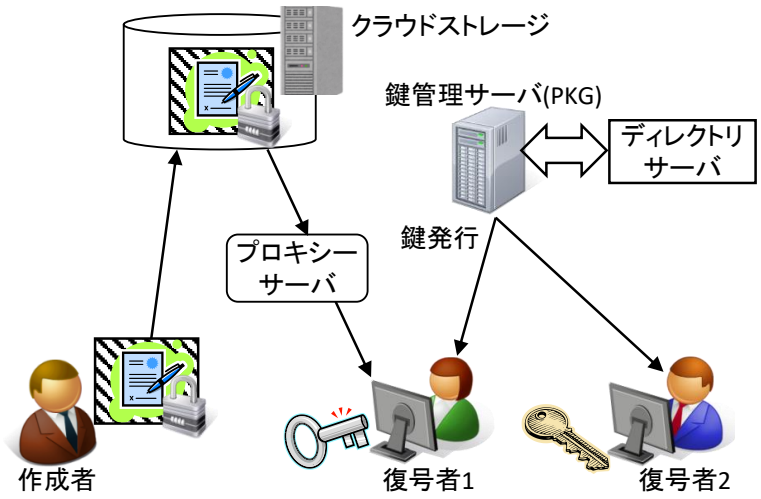

図 3.3: システムの登場人物

$$\begin{pmatrix} \overrightarrow{u}_{t,1} \\ \vdots \\ \overrightarrow{u}_{t,n_t} \end{pmatrix} := (Z_t^{-1})^T \ \textit{for}\ t = 1, \ldots, d,$$

$$\boldsymbol{h}_{0,0} := (\delta, 0, 0, \delta_0, 0)_{\mathbb{B}_0^*}, \boldsymbol{h}_{1,0} := (\delta, u_0, 0, \delta_0, 0)_{\mathbb{B}_0^*},$$

$$\boldsymbol{e}_0 := (\omega, \tau z_0, 0, 0, 0)_{\mathbb{B}_0},$$

$$\textit{for}\ t = 1, \ldots, d; i = 1, \ldots, n_t;$$

$$\overrightarrow{e}_{t,i} := (0^{i-1}, 1, 0^{n_t - i}) \in \mathbb{F}_q^{n_t},\ \overrightarrow{\delta}_{t,i} \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n_t},$$

$$\boldsymbol{h}^*_{0,t,i} := (\delta \overrightarrow{e}_{t,i}, 0^{n_t}, \delta_{t,i}, 0)_{\mathbb{B}_t^*},$$

$$\boldsymbol{h}^*_{1,t,i} := (\delta \overrightarrow{e}_{t,i}, \overrightarrow{u}_{t,i}, \delta_{t,i}, 0)_{\mathbb{B}_t^*},$$

$$\boldsymbol{e}_{t,i} := (\omega \overrightarrow{e}_{t,i}, \tau \overrightarrow{z}_{t,i}, 0^{n_t}, 0)_{\mathbb{B}_t},$$

$$\textit{return}\ (\mathbf{param}_{\overrightarrow{n}}, \hat{\mathbb{B}}_0, \mathbb{B}_0^*, \boldsymbol{h}^*_{\beta,0}, \boldsymbol{e}_0,$$

$$\{\hat{\mathbb{B}}_t, \mathbb{B}_t^*, \boldsymbol{h}_{\beta,t,i}, \boldsymbol{e}_{t,i}\}_{t=1,\ldots,d;i=1,\ldots,n_t}).$$

確率的アルゴリズム $\mathcal{B}$ に対して，$\mathcal{B}$ のアドバンテージを下記のように定義する．

$$\begin{aligned} \mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}}(\lambda) \ := \ & \Big| \mathsf{Pr}\left[\mathcal{B}(1^\lambda, \varrho) \to 1 \,\middle|\, \varrho \xleftarrow{\mathsf{R}} \mathcal{G}_0^{\mathsf{P2}}(1^\lambda, \overrightarrow{n}) \right] \\ & - \mathsf{Pr}\left[\mathcal{B}(1^\lambda, \varrho) \to 1 \,\middle|\, \varrho \xleftarrow{\mathsf{R}} \mathcal{G}_1^{\mathsf{P2}}(1^\lambda, \overrightarrow{n}) \right] \Big| \end{aligned}$$

Lemma 2. *Problem2* に対する確率的多項式時間攻撃者 $\mathcal{B}$ が存在するならば，*DLIN* 仮定に対する確率的多項式時間アルゴリズム $\mathcal{E}$ が存在する．攻撃者 $\mathcal{B}$ のアドバンテージは，任意のセキュリティパラメータ λ に対して $\mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}}(\lambda) \leq \mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{E}}(\lambda) + 5/q$ となる．

3.3 システムモデルと機能要件

本提案方式では，3.1.4 節で述べたように，プロキシー支援の下で失効を実現する．そのシステム構成は図 3.3 に示す構成となり，下記に示すエンティティにて構成される．

- **クラウドストレージ**：暗号化データが保管されるストレージ．クラウドベンダによって運用・管理され，企業がサービス利用契約を締結して利用する．他社によって運営されることから，クラウドストレージは semi-honest であると仮定する．すなわち，正しくデータは保管されるが，クラウドストレージの管理者によってデータ閲覧されるリスクが存在する．

- **鍵管理サーバ**：ユーザの属性や役割に基づき，プロキシー鍵とユーザ秘密鍵を発行するサーバ．鍵生成や失効管理などセキュリティの要となることから，信頼できるエンティティでなければならない．そのため，鍵管理サーバの運営は，理想的には企業側にて実施することが好ましい．更に，企業側管理者による悪意を防ぐため，権限分散によってマスター秘密鍵を保管したり，監視によって不正が行われていないことをチェックするなどの運用体制を整えることが必要となる．ただ，現実としては鍵管理サーバを企業側で運営することは難しいケースもあるため，クラウド管理者にシステムの運営を任せるケースも考えられる．この場合，鍵発行に用いるマスター秘密鍵だけは企業側管理者によって安全に管理する事が必要であり，企業側管理者パスワードでマスター秘密鍵を暗号化しておくことや，IC カードなどの物理デバイスを用いて保護するなどの対策を施す事が好ましい．

- **プロキシーサーバ**: プロキシー鍵を管理し，ユーザの要求に基づき暗号化データの部分復号を行うサーバ．鍵管理サーバからの指示により，プロキシー鍵の削除も行う．企業内に設置されるケースや，クラウドストレージ側に設置されるケースが考えられるため，クラウドストレージと同様に semi-honest と仮定する．すなわち，システムによって自動処理される仕組みになっているため，ユーザの要求通りに正しく部分復号を行い，鍵管理サーバからの指示によってプロキシー鍵を確実に管理・消去すると想定する．また，プロキシー鍵を安全に管理するため，例えば，プロキシー鍵は暗号化して保管し，利用時に復号してから利用することも想定する．さらに，例えば Confidential Computing などの技術を用いて，復号したプロキシー鍵が漏洩しないような対策を施されていることを想定する．これらの対策に加え，適切な運用者教育によって，ユーザと結託して暗号化データを復号するような能動的な攻撃を行うリスクは極めて低いと想定する．

- **作成者**：データの作成者であり，作成したデータを関数型暗号にて暗号化して，クラウドストレージに保管するユーザ．次に示す復号者も兼ねることがある．

- **復号者**：ユーザ秘密鍵を保有し，暗号化データを閲覧するユーザ．鍵管理サーバにてアクセス権限や属性が管理されている．

上記システムにおいて，秘匿性を保証することに加え，失効機能として以下の要件を満たす必要がある．

要件 1: 暗号化時に失効されたユーザが誰かを意識しないで良いこと．すなわち，暗号化時は従来方式通りにアクセス可能者の条件のみを指定すればよく，誰が失効されているかを意識する必要がないこと．

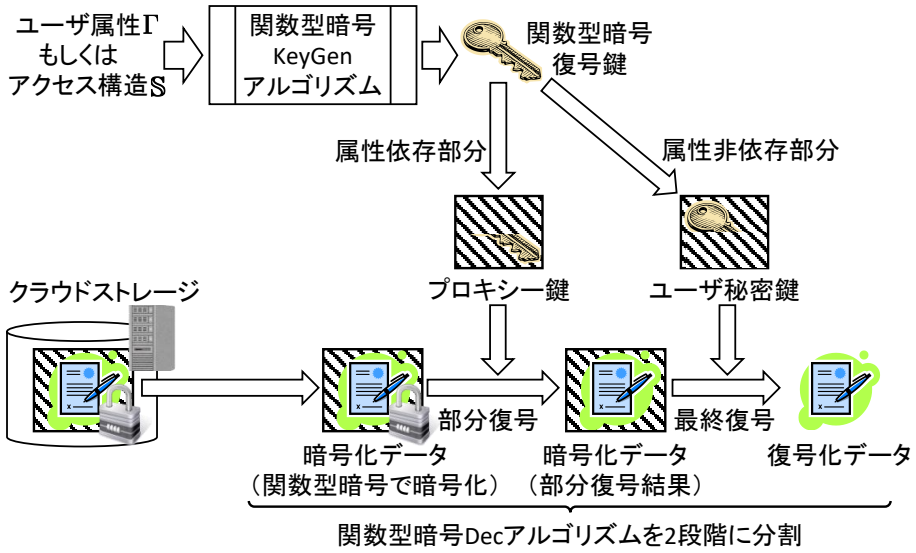

図 3.4: 失効機能付き関数型暗号の実現アイデア

要件 2: 復号できていた暗号化データが復号できなくなること．すなわち，人事異動などで属性やアクセス権限が変わった際に，またユーザ秘密鍵が漏洩した際に，失効させたユーザ秘密鍵で暗号化データが復号できてはならない．

要件 3: 失効処理に伴い，クラウド側の処理が不要もしくは軽微であること．すなわち，失効を行うために，クラウドストレージに保管された暗号化データの変換などが不要であること．

要件 4: 失効していないユーザ秘密鍵は従来通り使えること．すなわち，失効していないユーザに対して，ユーザ秘密鍵の更新などが不要であること．

3.4 提案方式

失効機能付き鍵ポリシー型関数型暗号，および失効機能付き暗号文ポリシー型関数型暗号の実現方式を提案する．

3.4.1 実現のためのアイデア

関数型暗号に失効機能を付加するためのアイデアを図 3.4 を用いて説明する．関数型暗号では，鍵ポリシー型の場合はアクセス構造 $\mathbb{S}$ を，暗号文ポリシー型の場合はユーザの属性集合 Γ を指定して，復号鍵を生成する．この復号鍵を持つユーザは，暗号化データの復号が可能となる．我々の提案方式では，関数型暗号の復号鍵をプロキシーサーバ用のプロキシー鍵と，ユーザ用のユーザ秘密鍵の 2 個に分割する．この構成より，暗号化データを復号する際は，最初にプロキシー鍵で暗号化データを部分復号するが，復号処理が途中で止まった状態となる．その部分復号結果をユーザ秘密鍵を用いて最終復号することで，復号結果である

データが得られる．そのため，プロキシー鍵を削除することで，対応するユーザ秘密鍵だけでは暗号化データが復号できなくなり，ユーザ秘密鍵の失効が実現できる．

また，暗号化データの秘匿性に加えて失効機能の有効性を示すためには，暗号化データを管理するクラウドサーバ，全ユーザのプロキシー鍵を管理するプロキシーサーバ，失効したユーザ秘密鍵を持つユーザの 3 者を攻撃者と想定し，いずれの攻撃者も暗号化データから一切の情報が得られないことを示す必要がある．そこで安全性モデルとして，チャレンジ暗号文を復号できない場合はプロキシー鍵とユーザ秘密鍵の双方をクエリーできるのに加え，復号できる場合でもプロキシー鍵かユーザ秘密鍵の一方をクエリーすることを許容した状況で，攻撃者がチャレンジ暗号文の識別ができないことをモデル化した．そして，本モデルに基づいて提案方式が adaptive な攻撃者に対して安全であることを示す．

ここで課題となるのは，Dual system encryption に基づいて安全性証明がなされた方式では，チャレンジ暗号文を復号できる属性やアクセス構造を用いて復号鍵をクエリすると，semi-functional な復号鍵が semi-functional チャレンジ暗号文と correlation を起こす点にある [34]，[51]．関数型暗号では，自明な攻撃を防ぐためにチャレンジ暗号文を復号できる復号鍵をクエリーできないという制約を持たせており，この correlation は問題にならなかった．しかし，本提案方式では，チャレンジ暗号文を復号できる属性やアクセス構造であっても，プロキシー鍵かユーザ秘密鍵の一方であればクエリーを許すため，この点が問題となる．そこで我々は，復号鍵を要素ごとに分解し，correlation を起こす要素をプロキシー鍵とユーザ秘密鍵に分けて配置することで，チャレンジ暗号文を復号できる属性やアクセス構造を指定された場合でも，いずれか一方しか入手できない攻撃者のビューからは correlation が見えないような仕組みを実現した．これにより，Dual system encryption の枠組みのままで安全性証明を可能とした．

3.4.2 失効機能付き鍵ポリシー型関数型暗号

鍵ポリシー型関数型暗号に対して失効機能を付加したアルゴリズム，およびその安全性について示す．

3.4.2.1 アルゴリズム定義とセキュリティ定義

失効機能付き鍵ポリシー型関数型暗号（RKP-FE）のアルゴリズム定義を下記に示す．

Definition 10. 失効機能付き鍵ポリシー型関数型暗号（*RKP-FE*）は，下記のアルゴリズムによって構成される．

Setup セキュリティパラメータ 1^λ と属性フォーマット $\overrightarrow{n} = (d; n_1, \ldots, n_d)$ を入力とし，公開パラメータ mpk とマスター秘密鍵 msk を出力．

KeyGen 公開パラメータ mpk，マスター秘密鍵 msk，アクセス構造 $\mathbb{S} := (M, \rho)$ を入力とし，プロキシー鍵 $\mathsf{pxk}_{\mathbb{S}}$ とユーザ秘密鍵 $\mathsf{sk}_{\mathbb{S}}$ を出力．

Enc 公開パラメータ mpk，属性集合 $\Gamma := \{(t, \overrightarrow{x}_t) | \overrightarrow{x}_t \in \mathbb{F}_q^{n_t} \backslash \{\overrightarrow{0}\}, 1 \leq t \leq d\}$，メッセージ空間 msg から選んだ平文 m を入力とし，暗号文 ct_Γ を出力．

Trans 公開パラメータ mpk，暗号文 ct_Γ，プロキシー鍵 $\mathsf{pxk}_\mathbb{S}$ を入力とし，属性集合 Γ がアクセス構造 $\mathbb{S}$ を満たすなら部分復号データ ct'_Γ を出力し，満たさないなら $\perp$ を出力．

Dec 公開パラメータ mpk，ユーザ秘密鍵 $\mathsf{sk}_\mathbb{S}$，部分復号データ ct'_Γ を入力とし，平文 m か $\perp$ を出力．

上記アルゴリズムにおいて，Correctness property は下記のように定義される．

Definition 11. 全ての $(\mathsf{mpk}, \mathsf{msk}) \xleftarrow{\mathsf{R}} \mathbf{Setup}(1^\lambda, \overrightarrow{\pi})$，全てのアクセス構造 $\mathbb{S}$，全てのプロキシー鍵とユーザ秘密鍵ペア $(\mathsf{pxk}_\mathbb{S}, \mathsf{sk}_\mathbb{S}) \xleftarrow{\mathsf{R}} \mathbf{KeyGen}(\mathsf{mpk}, \mathsf{msk}, \mathbb{S})$，全てのメッセージ m，アクセス構造 $\mathbb{S}$ が受理する全ての属性集合 Γ，全ての暗号文 $\mathsf{ct}_\Gamma \xleftarrow{\mathsf{R}} \mathbf{Enc}(\mathsf{mpk}, m, \Gamma)$ に対して，$m = \mathbf{Dec}(\mathsf{mpk}, \mathsf{sk}_\mathbb{S}, \mathbf{Trans}(\mathsf{mpk}, \mathsf{pxk}_\mathbb{S}, \mathsf{ct}_\Gamma))$ が成り立つ．

また，上記の失効機能付き鍵ポリシー型関数型暗号の IND-CPA 攻撃者に対する安全性は，下記のゲームによって定義される．

Definition 12. 次のゲームにおいて，多項式時間攻撃者 $\mathcal{A}$ のアドバンテージが *negligible* であるなら，その失効機能付き鍵ポリシー型関数型暗号方式は *IND-CPA* 安全であるという．

Setup 挑戦者 $\mathcal{C}$ は，**Setup** を実行して公開パラメータ mpk，マスター秘密鍵 msk を生成し，公開パラメータ mpk を攻撃者 $\mathcal{A}$ に与える．

Phase1 攻撃者 $\mathcal{A}$ は，適応的に多項式回だけアクセス構造 $\mathbb{S}_\iota := (M_\iota, \rho_\iota)$ に対して以下のクエリを実施できる．

- $\mathrm{Create}(\mathbb{S}_\iota)$: 挑戦者 $\mathcal{C}$ は，アクセス構造 $\mathbb{S}_\iota$ からプロキシー鍵 $\mathsf{pxk}_{\mathbb{S}_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\mathbb{S}_\iota}$ を生成し，手元に保管する．
- $\mathrm{CorruptProxyKey}(\iota)$*:* 挑戦者 $\mathcal{C}$ は，ι 番目に生成したプロキシー鍵 $\mathsf{pxk}_{\mathbb{S}_\iota}$ を攻撃者 $\mathcal{A}$ に対して送付する．
- $\mathrm{CorruptSecretKey}(\iota)$: 挑戦者 $\mathcal{C}$ は，ι 番目に生成したユーザ秘密鍵 $\mathsf{sk}_{\mathbb{S}_\iota}$ を攻撃者 $\mathcal{A}$ に送付する．

Challenge 攻撃者 $\mathcal{A}$ は，チャレンジする平文 $m^{(0)}, m^{(1)}$ と属性集合 Γ^* を選び，挑戦者 $\mathcal{C}$ に送付する．挑戦者 $\mathcal{C}$ は，一様にビット $b \in \{0, 1\}$ を選び，平文 $m^{(b)}$ を暗号化して，暗号文 $\mathsf{ct}^{(b)}_{\Gamma^*}$ を攻撃者 $\mathcal{A}$ に送付する．ただし，自明な攻撃を防ぐため，プロキシー鍵 $\mathsf{pxk}_{\mathbb{S}_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\mathbb{S}_\iota}$ のペアを取得したアクセス構造 $\mathbb{S}_\iota$ を満たすような属性集合 Γ^* を選んではならない．

Phase2 攻撃者 $\mathcal{A}$ は，*Phase1* と同様にクエリを実行する．ただし，属性集合 Γ^* を満たすようなプロキシー鍵 $\mathsf{pxk}_{\mathbb{S}_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\mathbb{S}_\iota}$ のペアをクエリしてはならない．

Guess 攻撃者 $\mathcal{A}$ 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 $\mathcal{A}$ のアドバンテージは，セキュリティパラメータ λ に対して $\mathsf{Adv}^{\mathsf{RKP-FE}}_{\mathcal{A}}(\lambda) := |\mathsf{Pr}[b' = b] - 1/2|$ にて定義される．

3.4.2.2 実現方式

3.4.1 節で述べたアイデアにより構成した失効機能付き鍵ポリシー型関数型暗号のアルゴリズムを下記に示す.

Setup $(1^\lambda, \overrightarrow{n} = (d; n_1, \ldots, n_d))$:

$(\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{B}_t, \mathbb{B}_t^*\}_{t=0,\ldots,d}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n})$,
$\hat{\mathbb{B}}_0 := (\boldsymbol{b}_{0,1}, \boldsymbol{b}_{0,3}, \boldsymbol{b}_{0,5})$,
$\hat{\mathbb{B}}_t := (\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,n_t}, \boldsymbol{b}_{t,3n_t+1})$ for $t = 1, \ldots, d$,
$\hat{\mathbb{B}}_0^* := (\boldsymbol{b}_{0,1}^*, \boldsymbol{b}_{0,3}^*, \boldsymbol{b}_{0,4}^*)$,
$\hat{\mathbb{B}}_t^* := (\boldsymbol{b}_{t,1}^*, \ldots, \boldsymbol{b}_{t,n_t}^*, \boldsymbol{b}_{t,2n_t+1}^*, \ldots, \boldsymbol{b}_{t,3n_t}^*)$ for $t = 1, \ldots, d$,
$\mathsf{mpk} := (1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\hat{\mathbb{B}}_t\}_{t=0,\ldots,d})$,
$\mathsf{msk} := \{\hat{\mathbb{B}}_t^*\}_{t=0,\ldots,d}$,
return $\mathsf{mpk}, \mathsf{msk}$.

KeyGen $(\mathsf{mpk}, \mathsf{msk}, \mathbb{S})$:

$\overrightarrow{f} \xleftarrow{\mathsf{U}} \mathbb{F}_q^r$, $\overrightarrow{s}^T := (s_1, \ldots, s_\ell)^T := M \cdot \overrightarrow{f}^T$,
$s_0 := \overrightarrow{1} \cdot \overrightarrow{f}^T, \eta_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q$,
$\boldsymbol{k}_0^* := (-s_0, 0, 1, \eta_0, 0)_{\mathbb{B}_0^*}$,
for $i = 1, \ldots, \ell$;
  if $\rho(i) = (t, \overrightarrow{v}_i)$,
    $\theta_i \xleftarrow{\mathsf{U}} \mathbb{F}_q, \overrightarrow{\eta}_i \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n_t}$,
    $\boldsymbol{k}_i^* := (s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i, 0^{n_t}, \overrightarrow{\eta_i}, 0)_{\mathbb{B}_t^*}$,
  if $\rho(i) = \neg(t, \overrightarrow{v}_i)$,
    $\overrightarrow{\eta}_i \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n_t}$,
    $\boldsymbol{k}_i^* := (s_i \overrightarrow{v}_i, 0^{n_t}, \overrightarrow{\eta_i}, 0)_{\mathbb{B}_t^*}$,
$\mathsf{pxk}_{\mathbb{S}} := (\mathbb{S}, \boldsymbol{k}_1^*, \ldots, \boldsymbol{k}_\ell^*)$,
$\mathsf{sk}_{\mathbb{S}} := (\boldsymbol{k}_0^*)$,
return $\mathsf{pxk}_{\mathbb{S}}, \mathsf{sk}_{\mathbb{S}}$.

Enc $(\mathsf{mpk}, \mathrm{m}, \Gamma)$:

$\omega, \varphi_0, \varphi_t, \zeta \xleftarrow{\mathsf{U}} \mathbb{F}_q$ for $(t, \overrightarrow{x}_t) \in \Gamma$,
$\boldsymbol{c}_0 := (\omega, 0, \zeta, 0, \varphi_0)_{\mathbb{B}_0}$,
$\boldsymbol{c}_t := (\omega \overrightarrow{x}_t, 0^{n_t}, 0^{n_t}, \varphi_t)_{\mathbb{B}_t}$ for $(t, \overrightarrow{x}_t) \in \Gamma$,
$c_{d+1} := g_T^\zeta m$,
return $\mathsf{ct}_\Gamma := (\Gamma, \boldsymbol{c}_0, \{\boldsymbol{c}_t\}_{(t,\overrightarrow{x}_t)\in\Gamma}, c_{d+1})$.

Trans $(\mathsf{mpk}, \mathsf{pxk}_{\mathbb{S}}, \mathsf{ct}_\Gamma)$:

もし $\mathbb{S}$ が Γ を受け入れるなら，下記に示すような I と $\{\alpha_i\}_{i\in I}$ を計算する:
  $\overrightarrow{1} = \sum_{i\in I} \alpha_i M_i$,
  $I \subseteq \{i \in \{1, \ldots, \ell\} | [\rho(i) = (t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \in \Gamma \wedge \overrightarrow{v}_i \cdot \overrightarrow{x}_t = 0]$
    $\vee [\rho(i) = \neg(t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \in \Gamma \wedge \overrightarrow{v}_i \cdot \overrightarrow{x}_t \neq 0]\}$,
$K_2 := \prod_{i\in I \wedge \rho(i)=(t,\overrightarrow{v}_i)} e(\boldsymbol{c}_t, \boldsymbol{k}_i^*)^{\alpha_i}$

$\cdot \prod_{i \in I \wedge \rho(i) = \neg(t, \overrightarrow{v}_i)} e(\boldsymbol{c}_t, \boldsymbol{k}_i^*)^{\alpha_i / (\overrightarrow{v}_i \cdot \overrightarrow{x}_t)}$,
return $\mathsf{ct}'_{\mathbb{S}} := (\boldsymbol{c}_0, c_{d+1}, K_2)$.

Dec $(\mathsf{mpk}, \mathsf{sk}_{\mathbb{S}}, \mathsf{ct}'_{\Gamma})$:
$K_1 := e(\boldsymbol{c}_0, \boldsymbol{k}_0^*)$,
$K = K_1 \cdot K_2$,
return $m' = c_{d+1}/K$.

なお，上記アルゴリズムにおいて，ベクトル $\overrightarrow{x}_t := (x_{t,1}, \ldots, x_{t,n_t})$ は，$x_{t,1} = 1$ になるように正規化されていると仮定する．

[Correctness property]
上記アルゴリズムにおいて，属性集合 Γ がアクセス構造 $\mathbb{S}$ を満たすなら，下記が成り立つ．

$$
\begin{aligned}
K_1 &= e(\boldsymbol{c}_0, \boldsymbol{k}_0^*) \\
&= g_T^{-\omega s_0 + \zeta} \\
K_2 &= \prod_{i \in I \wedge \rho(i) = (t, \overrightarrow{v}_i)} e(\boldsymbol{c}_t, \boldsymbol{k}_i^*)^{\alpha_i} \\
&\quad \cdot \prod_{i \in I \wedge \rho(i) = \neg(t, \overrightarrow{v}_i)} e(\boldsymbol{c}_t, \boldsymbol{k}_i^*)^{\alpha_i / (\overrightarrow{v}_i \cdot \overrightarrow{x}_t)} \\
&= \prod_{i \in I \wedge \rho(i) = (t, \overrightarrow{v}_i)} g_T^{\omega \alpha_i (s_i + \theta_i \overrightarrow{v}_i \cdot \overrightarrow{x}_t)} \\
&\quad \cdot \prod_{i \in I \wedge \rho(i) = \neg(t, \overrightarrow{v}_i)} g_T^{\omega \alpha_i s_i (\overrightarrow{v}_i \cdot \overrightarrow{x}_t) / (\overrightarrow{v}_i \cdot \overrightarrow{x}_t)} \\
&= g_T^{\omega (\sum_{i \in I} \alpha_i s_i)} = g_T^{\omega s_0} \\
K &= K_1 \cdot K_2 \\
&= g_T^{-\omega s_0 + \zeta} \cdot g_T^{\omega s_0} \\
&= g_T^{-\omega s_0 + \zeta} \cdot g_T^{\omega s_0} \\
&= g_T^{\zeta}
\end{aligned}
$$

それゆえ，

$$c_{d+1}/K = g_T^{\zeta} m / g_T^{\zeta} = m$$

よって，Correctness property が成り立つ．

3.4.2.3 安全性

上記提案方式の安全性は，下記の様に示すことができる．

Theorem 1. 失効機能付き鍵ポリシー型関数型暗号に対して選択平文攻撃を行う確率的多項式時間攻撃者 $\mathcal{A}$ を仮定したとき，攻撃者 $\mathcal{A}$ のアドバンテージ $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RKP-FE}}(\lambda)$ は下記に示すように *DLIN* 仮定のもとで *negligible* であり，提案方式は *IND-CPA* 安全である．

$$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RKP-FE}}(\lambda) \leq \mathsf{Adv}_{\mathcal{E}_1}^{\mathsf{DLIN}}(\lambda)$$

$$+\sum_{h=0}^{\nu-1}\left(\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{E}^{+}_{2,h}}(\lambda)+\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{E}_{2,h+1}}(\lambda)\right)+\epsilon$$

ここで，ν は攻撃者 $\mathcal{A}$ の鍵クエリの回数であり，$\epsilon=(2d\nu+16\nu+d+7)/q$ である．

本定理の証明は，3.7 節に示す．

3.4.3 失効機能付き暗号文ポリシー型関数型暗号

暗号文ポリシー型関数型暗号に対して失効機能を付加したアルゴリズム，およびその安全性について示す．

3.4.3.1 アルゴリズム定義とセキュリティ定義

失効機能付き暗号文ポリシー型関数型暗号（RCP-FE）のアルゴリズム定義を下記に示す．

Definition 13. 失効機能付き暗号文ポリシー型関数型暗号（*RCP-FE*）は，下記のアルゴリズムによって構成される．

Setup セキュリティパラメータ 1^{λ} と属性フォーマット $\overrightarrow{n}=(d;n_1,\ldots,n_d)$ を入力とし，公開パラメータ mpk とマスター秘密鍵 msk を出力．

KeyGen 公開パラメータ mpk，マスター秘密鍵 msk，属性集合 $\Gamma:=\{(t,\overrightarrow{x}_t)|\overrightarrow{x}_t\in\mathbb{F}_q^{n_t},1\leq t\leq d\}$ を入力とし，プロキシー鍵 pxk_Γ とユーザ秘密鍵 sk_Γ を出力．

Enc 公開パラメータ mpk，アクセス構造 $\mathbb{S}:=(M,\rho)$，メッセージ空間 msg から選んだ平文 m を入力とし，暗号文 $\mathsf{ct}_{\mathbb{S}}$ を出力．

Trans 公開パラメータ mpk，暗号文 $\mathsf{ct}_{\mathbb{S}}$，プロキシー鍵 pxk_Γ を入力とし，属性集合 Γ がアクセス構造 $\mathbb{S}$ を満たすなら部分復号データ $\mathsf{ct}'_{\mathbb{S}}$ を出力し，満たさないなら $\perp$ を出力．

Dec 公開パラメータ mpk，ユーザ秘密鍵 sk_Γ，部分復号データ $\mathsf{ct}'_{\mathbb{S}}$ を入力とし，平文 m か $\perp$ を出力．

上記アルゴリズムにおいて，Correctness property は下記のように定義される．

Definition 14. 全ての $(\mathsf{mpk},\mathsf{msk})\xleftarrow{\mathsf{R}}\mathbf{Setup}(1^{\lambda},\overrightarrow{n})$，全ての属性集合 Γ，全てのプロキシー鍵と秘密鍵ペア $(\mathsf{pxk}_\Gamma,\mathsf{sk}_\Gamma)\xleftarrow{\mathsf{R}}\mathbf{KeyGen}(\mathsf{mpk},\mathsf{msk},\Gamma)$，全てのメッセージ m，属性集合 Γ を受理する全てのアクセス構造 $\mathbb{S}$，全ての暗号文 $\mathsf{ct}_{\mathbb{S}}\xleftarrow{\mathsf{R}}\mathbf{Enc}(\mathsf{mpk},m,\mathbb{S})$ に対して，$m=\mathbf{Dec}(\mathsf{mpk},\mathsf{sk}_\Gamma,\mathbf{Trans}(\mathsf{mpk},\mathsf{pxk}_\Gamma,\mathsf{ct}'_{\mathbb{S}}))$ が成り立つ．

また，上記の失効機能付き暗号文ポリシー型関数型暗号の IND-CPA 攻撃者に対する安全性は，下記のゲームによって定義される．

Definition 15. 次のゲームにおいて，多項式時間攻撃者 $\mathcal{A}$ のアドバンテージが *negligible* であるなら，その失効機能付き暗号文ポリシー型関数型暗号方式は *IND-CPA* 安全であるという．

Setup 挑戦者 $\mathcal{C}$ は，**Setup** を実行して公開パラメータ mpk，マスター秘密鍵 msk を生成し，公開パラメータ mpk を攻撃者 $\mathcal{A}$ に与える．

Phase1 攻撃者 $\mathcal{A}$ は，適応的に多項式回だけ属性集合 Γ に対応する以下のクエリを実施できる．

- $\mathrm{Create}(\Gamma_\iota)$: 挑戦者 $\mathcal{C}$ は，属性集合 Γ_ι に対応するプロキシー鍵 $\mathsf{pxk}_{\Gamma_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\Gamma_\iota}$ を生成し，手元に保管する．
- $\mathrm{CorruptProxyKey}(\iota)$*:* 挑戦者 $\mathcal{C}$ は，ι 番目に生成したプロキシー鍵 $\mathsf{pxk}_{\Gamma_\iota}$ を攻撃者 $\mathcal{A}$ に対して送付する．
- $\mathrm{CorruptSecretKey}(\iota)$: 挑戦者 $\mathcal{C}$ は，ι 番目に生成したユーザ秘密鍵 $\mathsf{sk}_{\Gamma_\iota}$ を攻撃者 $\mathcal{A}$ に送付する．

Challenge 攻撃者 $\mathcal{A}$ は，チャレンジする平文 $m^{(0)}, m^{(1)}$ とアクセス構造 $\mathbb{S}^*$ を選び，挑戦者 $\mathcal{C}$ に送付する．挑戦者 $\mathcal{C}$ は，一様にビット $b \in \{0,1\}$ を選び，平文 $m^{(b)}$ を暗号化して，暗号文 $\mathsf{ct}_{\mathbb{S}^*}^{(b)}$ を攻撃者 $\mathcal{A}$ に送付する．ただし，自明な攻撃を防ぐため，プロキシー鍵 $\mathsf{pxk}_{\Gamma_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\Gamma_\iota}$ のペアを取得した属性集合 Γ_ι が満たすようなアクセス構造 $\mathbb{S}^*$ を選んではならない．

Phase2 攻撃者 $\mathcal{A}$ は，*Phase1* と同様にクエリを実行する．ただし，アクセス構造 $\mathbb{S}^*$ を満たすような属性集合 Γ_ι に対して，プロキシー鍵 $\mathsf{pxk}_{\Gamma_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\Gamma_\iota}$ のペアをクエリしてはならない．

Guess 攻撃者 $\mathcal{A}$ 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 $\mathcal{A}$ のアドバンテージは，セキュリティパラメータ λ に対して $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RCP-FE}}(\lambda) := |\mathsf{Pr}[b' = b] - 1/2|$ にて定義される．

3.4.3.2 実現方式

3.4.1 節で述べたアイデアにより構成した失効機能付き暗号文ポリシー型関数型暗号のアルゴリズムを下記に示す．

Setup $(1^\lambda, \overrightarrow{n} = (d; n_1, \ldots, n_d))$:
$(\mathsf{param}_{\overrightarrow{n}}, \{\mathbb{B}_t, \mathbb{B}_t^*\}_{t=0,\ldots,d}) \xleftarrow{\mathsf{R}} \mathcal{G}_{\mathsf{ob}}(1^\lambda, \overrightarrow{n})$,
$\hat{\mathbb{B}}_0 := (\boldsymbol{b}_{0,1}, \boldsymbol{b}_{0,3}, \boldsymbol{b}_{0,5})$,
$\hat{\mathbb{B}}_t := (\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,n_t}, \boldsymbol{b}_{t,3n_t+1})$ for $t = 1, \ldots, d$,
$\hat{\mathbb{B}}_0^* := (\boldsymbol{b}_{0,1}^*, \boldsymbol{b}_{0,3}^*, \boldsymbol{b}_{0,4}^*)$,
$\hat{\mathbb{B}}_t^* := (\boldsymbol{b}_{t,1}^*, \ldots, \boldsymbol{b}_{t,n_t}^*, \boldsymbol{b}_{t,2n_t+1}^*, \ldots, \boldsymbol{b}_{t,3n_t}^*)$ for $t = 1, \ldots, d$,
$\mathsf{mpk} := (1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\hat{\mathbb{B}}_t\}_{t=0,\ldots,d})$,
$\mathsf{msk} := \{\hat{\mathbb{B}}_t^*\}_{t=0,\ldots,d}$,
return $\mathsf{mpk}, \mathsf{msk}$.

KeyGen (mpk,msk,Γ):

$\delta, \varphi_0 \xleftarrow{\cup} \mathbb{F}_q$, $\overrightarrow{\varphi}_t \xleftarrow{\cup} \mathbb{F}_q^{n_t}$ s.t. $(t, \overrightarrow{x}_t) \in \Gamma$,

$\boldsymbol{k}_0^* := (\delta, 0, 1, \varphi_0, 0)_{\mathbb{B}_0^*}$,

$\boldsymbol{k}_t^* := (\delta \overrightarrow{x}_t, 0^{n_t}, \overrightarrow{\varphi}_t, 0)_{\mathbb{B}_t^*}$ for $(t, \overrightarrow{x}_t) \in \Gamma$,

$\mathsf{pxk}_\Gamma := (\Gamma, \{\boldsymbol{k}_t^*\}_{(t,\overrightarrow{x}_t)\in\Gamma})$,

$\mathsf{sk}_\Gamma := (\boldsymbol{k}_0^*)$,

return $\mathsf{pxk}_\Gamma, \mathsf{sk}_\Gamma$.

Enc (mpk, m, $\mathbb{S}$):

$\overrightarrow{f} \xleftarrow{\cup} \mathbb{F}_q^r$, $\overrightarrow{s}^T := (s_1, \ldots, s_\ell)^T := M \cdot \overrightarrow{f}^T$,

$s_0 := \overrightarrow{1} \cdot \overrightarrow{f}^T, \eta_0, \eta_i, \theta_i, \zeta \xleftarrow{\cup} \mathbb{F}_q (i = 1, \ldots, \ell)$,

$\boldsymbol{c}_0 := (-s_0, 0, \zeta, 0, \eta_0)_{\mathbb{B}_0}$,

for $i = 1, \ldots, \ell$;

  if $\rho(i) = (t, \overrightarrow{v}_i)$,

    $\boldsymbol{c}_i := (s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i, 0^{n_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}$,

  if $\rho(i) = \neg(t, \overrightarrow{v}_i)$,

    $\boldsymbol{c}_i := (s_i \overrightarrow{v}_i, 0^{n_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}$,

$c_{d+1} := g_T^\zeta m$,

return $\mathsf{ct}_\mathbb{S} := (\mathbb{S}, \boldsymbol{c}_0, \boldsymbol{c}_1, \ldots, \boldsymbol{c}_\ell, c_{d+1})$.

Trans (mpk, pxk_Γ, $\mathsf{ct}_\mathbb{S}$):

もし $\mathbb{S}$ が Γ を受け入れるなら，下記に示すような I と $\{\alpha_i\}_{i\in I}$ を計算する:

  $\overrightarrow{1} = \sum_{i\in I} \alpha_i M_i$,

  $I \subseteq \{i \in \{1, \ldots, \ell\} | [\rho(i) = (t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \in \Gamma \wedge \overrightarrow{v}_i \cdot \overrightarrow{x}_t = 0]$

    $\vee [\rho(i) = \neg(t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \in \Gamma \wedge \overrightarrow{v}_i \cdot \overrightarrow{x}_t \neq 0]\}$,

$K_2 = \prod_{i\in I\wedge\rho(i)=(t,\overrightarrow{v}_i)} e(\boldsymbol{c}_i, \boldsymbol{k}_t^*)^{\alpha_i}$

  $\cdot \prod_{i\in I\wedge\rho(i)=\neg(t,\overrightarrow{v}_i)} e(\boldsymbol{c}_i, \boldsymbol{k}_t^*)^{\alpha_i/(\overrightarrow{v}_i\cdot\overrightarrow{x}_t)}$,

return $\mathsf{ct}'_\mathbb{S} := (\boldsymbol{c}_0, c_{d+1}, K_2)$.

Dec (mpk, sk_Γ, $\mathsf{ct}'_\mathbb{S}$):

$K_1 := e(\boldsymbol{c}_0, \boldsymbol{k}_0^*)$,

$K = K_1 \cdot K_2$,

return $m' = c_{d+1}/K$.

なお，上記アルゴリズムにおいて，ベクトル $\overrightarrow{x}_t := (x_{t,1}, \ldots, x_{t,n_t})$ は，$x_{t,1} = 1$ になるように正規化されていると仮定する．また，ベクトル $\overrightarrow{v}_i := (v_{i,1}, \ldots, v_{i,n_t})$ は，$v_{i,n_t} \neq 0$ であると仮定する．

[Correctness property]

上記アルゴリズムにおいて，属性集合 Γ がアクセス構造 $\mathbb{S}$ を満たすなら，下記が成り立つ．

$$
\begin{aligned}
K_1 &= e(\boldsymbol{c}_0, \boldsymbol{k}_0^*) \\
&= g_T^{-\delta s_0 + \zeta} \\
K_2 &= \prod_{i\in I\wedge\rho(i)=(t,\overrightarrow{v}_i)} e(\boldsymbol{c}_i, \boldsymbol{k}_t^*)^{\alpha_i}
\end{aligned}
$$

$$
\begin{aligned}
& \cdot \prod_{i\in I\wedge\rho(i)=\neg(t,\overrightarrow{v}_i)} e(\boldsymbol{c}_i,\boldsymbol{k}_t^*)^{\alpha_i/(\overrightarrow{v}_i\cdot\overrightarrow{x}_t)} \\
= & \prod_{i\in I\wedge\rho(i)=(t,\overrightarrow{v}_i)} g_T^{\delta\alpha_i(s_i+\theta_i\overrightarrow{v}_i\cdot\overrightarrow{x}_t)} \\
& \cdot \prod_{i\in I\wedge\rho(i)=\neg(t,\overrightarrow{v}_i)} g_T^{\delta\alpha_i s_i(\overrightarrow{v}_i\cdot\overrightarrow{x}_t)/(\overrightarrow{v}_i\cdot\overrightarrow{x}_t)} \\
= & g_T^{\delta\sum_{i\in I}\alpha_i s_i} = g_T^{\delta s_0} \\
K = & K_1\cdot K_2 \\
= & g_T^{-\delta s_0+\zeta}\cdot g_T^{\delta s_0} \\
= & g_T^{\zeta}
\end{aligned}
$$

それゆえ，

$$c_{d+1}/K = g_T^{\zeta} m / g_T^{\zeta} = m$$

よって，Correctness property が成り立つ．

3.4.3.3 安全性

上記提案方式の安全性は，下記の様に示すことができる．

Theorem 2. 失効機能付き暗号文ポリシー型関数型暗号に対して選択平文攻撃を行う確率的多項式時間攻撃者 $\mathcal{A}$ を仮定したとき，攻撃者 $\mathcal{A}$ のアドバンテージ $\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RCP-FE}}(\lambda)$ は下記に示すように *DLIN* 仮定のもとで *negligible* であり，提案方式は *IND-CPA* 安全である．

$$
\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RCP-FE}}(\lambda) \leq\ & \mathsf{Adv}_{\mathcal{E}_1}^{\mathsf{DLIN}}(\lambda) \\
& + \sum_{h=0}^{\nu-1}\left(\mathsf{Adv}_{\mathcal{E}_{2,h}^{+}}^{\mathsf{DLIN}}(\lambda) + \mathsf{Adv}_{\mathcal{E}_{2,h+1}}^{\mathsf{DLIN}}(\lambda)\right) + \epsilon
\end{aligned}
$$

ここで，ν は攻撃者 $\mathcal{A}$ の鍵クエリの回数であり，$\epsilon = (2d\nu + 16\nu + 2d + 8)/q$ である．

本定理の証明は，3.8 節に示す．

3.5 鍵管理方式

システム構成がシンプルで分かりやすい失効機能付き暗号文ポリシー型関数型暗号を例に，提案方式を実システムに適用する場合の鍵管理方式について述べる．基本的な流れは鍵ポリシー型関数型暗号でも同一である．

3.5.1 システムセットアップ手順

システムのセットアップとして，提案方式の鍵生成を行う手順を図 3.5 に示す．始めに，ユーザの属性情報が管理されているディレクトリサーバから，どの属性情報を用いるかを抽出し，属性番号表を作成する．この属性番号表では，属性情報と，属性値をベクトル表現

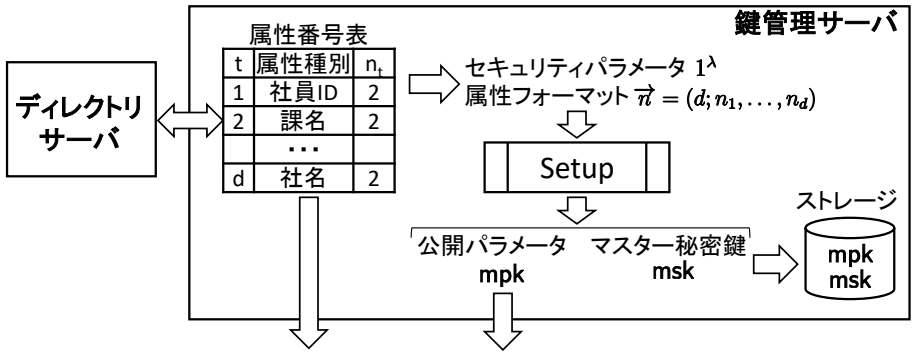

図 3.5: 鍵生成などシステムセットアップの流れ

$(t, \vec{x}_t)$ する際の属性番号 t との対応付けを行う．また，本提案方式では属性値の一致をベクトル内積値で判定する必要があるため，$\vec{x}_t = (1, '\text{属性値}')$，$\vec{v}_t = ('\text{属性値}', -1)$ とベクトル化するルールとし，$n_t = 2$ $(1 \leq t \leq d)$ とする．

この属性番号表より，セキュリティパラメータ 1^λ と属性フォーマット $\vec{n}$ を決定し，**Setup** アルゴリズムを実行して，公開パラメータ mpk とマスター秘密鍵 msk を生成する．この両者をストレージで安全に保管するとともに，公開パラメータと属性番号表は暗号化・復号処理に必要なためユーザに対して公開する．

3.5.2 ユーザ追加手順

ユーザに対して新たにユーザ秘密鍵を発行する手順を図 3.6 に示す．鍵管理サーバは，システム管理者から社員 ID を受領したら，ディレクトリサービスから該当ユーザの属性を取得する．本例では，社員 ID“$ID001$” のユーザが指定されたと仮定する．そして，セットアップ時に定めた属性番号表を参照して属性集合 Γ_{ID001} を生成したのちに，ストレージに保管された公開パラメータ mpk とマスター秘密鍵 msk を取り出して **KeyGen** アルゴリズムを実行し，プロキシー鍵 $\mathsf{pxk}_{\Gamma_{ID001}}$ とユーザ秘密鍵 $\mathsf{sk}_{\Gamma_{ID001}}$ を生成する．プロキシー鍵 $\mathsf{pxk}_{\Gamma_{ID001}}$ はプロキシーサーバへ登録を要求し，ユーザ秘密鍵 $\mathsf{sk}_{\Gamma_{ID001}}$ はユーザへ配布する．

プロキシー鍵 $\mathsf{pxk}_{\Gamma_{ID001}}$ を受け取ったプロキシーサーバは，自身が管理するプロキシー鍵リストに鍵 ID・社員 ID と関連付けて保管を行う．また，ユーザは，受領したユーザ秘密鍵 $\mathsf{sk}_{\Gamma_{ID001}}$ を鍵 ID と関連付けて自身が管理するストレージに安全に保管する．

3.5.3 暗号化データ復号手順

ユーザが暗号化データにアクセスする際の手順を図 3.7 に示す．始めに社員 ID が“$ID001$” であるユーザは，自身が復号権限を持つ暗号化データ $\mathsf{ct}_{\mathbb{S}}$ を指定して，プロキシーサーバに取得を依頼する．要求を受けたプロキシーサーバは，クラウドストレージから暗号化データ $\mathsf{ct}_{\mathbb{S}}$ をダウンロードし，管理するプロキシー鍵リストから社員 ID=“$ID001$”，かつ属性集合 Γ_{ID001} が暗号化データのアクセス構造 $\mathbb{S}$ を満たすプロキシー鍵 $\mathsf{pxk}_{\Gamma_{ID001}}$ を取り出す．そして，**Trans** アルゴリズムを実行し，部分復号データ $\mathsf{ct}'_{\mathbb{S}}$ を得る．この部分復号データを，利

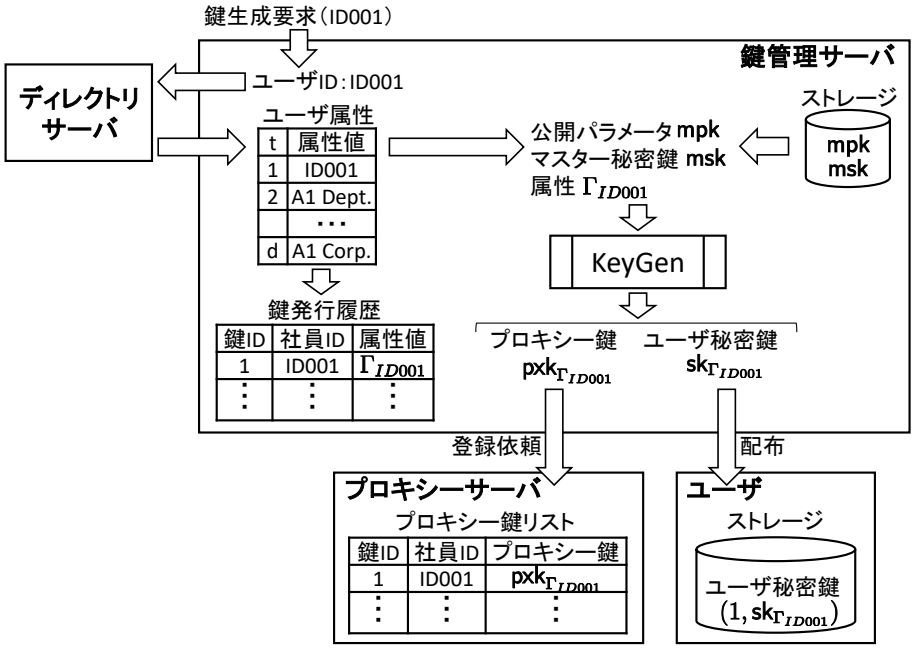

図 3.6: プロキシー鍵とユーザ秘密鍵生成の流れ

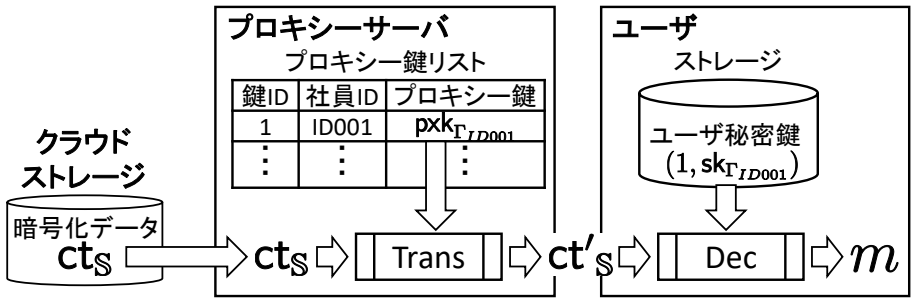

図 3.7: 暗号化データの取得・復号の流れ

用したプロキシー鍵に関連付けた鍵 ID と共にユーザに送付する．最後にユーザは，鍵 ID に対応するユーザ秘密鍵 $\mathsf{sk}_{\Gamma_{ID001}}$ を用いて **Dec** アルゴリズムを実行し，復号結果 m を得る．

3.5.4 ユーザ秘密鍵の失効手順

ユーザ秘密鍵を失効させる手順を図 3.8 に示す．3.4.1 節で述べたように，ユーザ秘密鍵を失効させるには対応するプロキシー鍵を消去するだけで十分である．そこで，鍵管理サーバはシステム管理者から失効させたいユーザ秘密鍵として，社員 ID と鍵 ID ペア $(ID001, 1)$ を受け取り，自身の管理する鍵発行履歴を参照して鍵 ID=1 のユーザ秘密鍵が有効な状態にあることを確認したのちに，履歴を無効状態に変更するとともに，プロキシーサーバに対してプロキシー鍵 $\mathsf{pxk}_{\Gamma_{ID001}}$ の削除を依頼する．削除依頼を受けたプロキシーサーバは，自身の管理するプロキシー鍵リストから鍵 ID=1 のプロキシー鍵 $\mathsf{pxk}_{\Gamma_{ID001}}$ を消去する．3.5.3 節

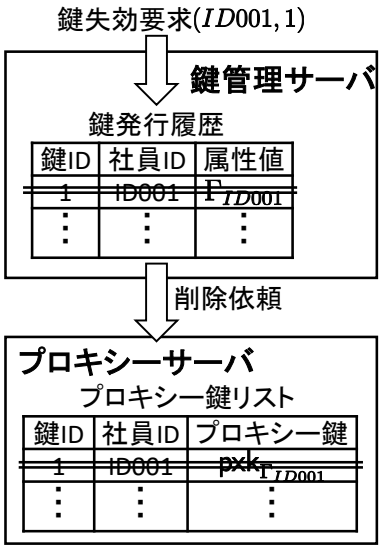

図 3.8: ユーザ秘密鍵の失効の流れ

で述べたように，ユーザがクラウドストレージに保管された暗号化データを復号するには，プロキシーサーバが管理するプロキシー鍵 $\mathsf{pxk}_{\Gamma_{ID001}}$ が必要であることから，これを削除することでユーザは復号ができなくなる．すなわち，ユーザ秘密鍵の失効が完了する．

3.6 考察

3.6.1 要件への適合性

3.3 節にて述べた要件を満たすことを考察する．

[要件 1:暗号化時に失効されたユーザが誰かを意識しないで良いこと]
失効処理は鍵管理サーバとプロキシーサーバが連携して実施するため，ユーザは失効処理について意識する必要がないし，誰が失効しているかを意識する必要もない．そのため，暗号化においては復号できるユーザの条件を指定するだけで良く，誰が失効しているかを意識する必要がない．

[要件 2:復号できていた暗号化データが復号できなくなること]
暗号化データの復号には，プロキシー鍵とユーザ秘密鍵のペアが必要であることから，プロキシー鍵を削除することで，その瞬間から対応するユーザ秘密鍵では復号が行えなくなる．すなわち，ユーザ秘密鍵を失効して，復号権限を無効化できる．プロキシーサーバは，企業や委託先にて管理されているサーバであることから，ユーザ秘密鍵を削除するより容易に実施できる．なお，ユーザ秘密鍵が漏洩した際は，従来と同一の属性集合もしくはアクセス構造にてユーザ秘密鍵を再発行することになるが，この場合も新しく生成した乱数からユーザ秘密鍵とプロキシー鍵のペアを生成することから，従来のユーザ秘密鍵は無効にしたまま，新しいユーザ秘密鍵を再発行することができる．

[要件 3:失効処理に伴い，クラウド側の処理が不要もしくは軽微であること]
本方式では，プロキシー鍵を削除するだけで失効ができる．そのため，ユーザ秘密鍵失効を行う際に，公開鍵の更新や，クラウドストレージ上の暗号化データの再暗号化などは一切不要である．

ただ，提案方式の仕組みにより，従来は端末側で実施していた復号処理の一部がプロキシーサーバにて実施が必要であることから，プロキシーサーバをクラウド側に置いた時にクラウド側の処理が増加することとなる．失効機能の導入に伴ってシステム全体の計算量が増えたわけではないが，クラウド側の処理量がアクセス数に応じて増加することとなる．そのため，プロキシーサーバはアクセス数に応じて自動的にスケールアウトするような実装が好ましい．システム全体の計算量が増加しないことは，3.6.2 節にて考察する．

[要件 4:失効していないユーザ秘密鍵は従来通り使えること]
本方式では，失効したいユーザ秘密鍵に対応するプロキシー鍵を削除するだけで失効ができる．そのため，他ユーザのユーザ秘密鍵は一切更新不要であり，従来通り利用することができる．

3.6.2 計算量

本方式では，関数型暗号の復号鍵を要素ごとに分解してプロキシー鍵とユーザ秘密鍵に分割し，復号処理時はプロキシー鍵での部分復号と，ユーザ秘密鍵による最終復号の 2 段階を踏むように変更しただけであるため，計算量はオリジナルの関数型暗号と同一である．ここでは，鍵ポリシー型を例に，その計算量の変化について考察する．

[**KeyGen** アルゴリズムの計算量]
3.4.2.2 節で述べたように，**KeyGen** アルゴリズムでは，関数型暗号の復号鍵を生成し，それを分割してプロキシー鍵とユーザ秘密鍵にする．そのため，計算量はオリジナルの関数型暗号と同一である．失効機能を付け加えることによる新たな演算はなく，計算量の増加はない．

[**Trans** アルゴリズムの計算量]
オリジナルの関数型暗号 **Dec** アルゴリズムの処理は，本提案方式の **Trans** アルゴリズムと **Dec** アルゴリズムに分割されているが，トータルの計算量は同じである．**Trans** アルゴリズムは，関数型暗号 **Dec** アルゴリズムとほぼ同一であるが，ベクトル空間 $\mathbb{V}$ におけるペアリング演算 1 回（K_1 の計算)，群 G_T 上での K_1 と K_2 の積，および c_{d+1}/K の除算が **Dec** アルゴリズムに移行しているので，その分だけ計算量は少なくなる．失効機能を付け加えることによる新たな演算はなく，計算量の増加はない．

[**Dec** アルゴリズムの計算量]
上記 **Trans** アルゴリズムにて述べたが，**Dec** アルゴリズムではベクトル空間 $\mathbb{V}$ におけるペアリング演算 1 回（K_1 の計算)，群 G_T 上での K_1 と K_2 の積，および c_{d+1}/K の除算だけを行う必要がある．失効機能を付け加えることによる新たな演算はなく，計算量の増加はない．

3.7 Theorem1 の証明

本提案方式は，高島ら [65] が提案した関数型暗号の復号鍵に対してのみ修正を行っているので，関数型暗号の証明をベースとして安全性証明を実施する．本証明は Dual System Encryption に基づいており，Game0 が Definition12 に示したオリジナルのゲームであり，チャレンジ暗号文を semi-functional に変化させた Game1 に変化させる．その後，プロキシー鍵もしくはユーザ秘密鍵を 1 個づつ semi-functional に変えていくが，最初に 1 回目にクエリーされたプロキシー鍵もしくはユーザ秘密鍵を pre-semi-functional に変えた Game2-0^+，次に semi-functional に変えた Game2-1 に変化させる．これをすべてのクエリに対して実施し，Game2-ν まで変化させる．最後に，チャレンジ暗号文を random に変化させた Game3 まで到達する．

ここで課題となるのは，オリジナルの関数型暗号の証明では，Game2$-h^+$ においてチャレンジ暗号文を復号できるアクセス構造を指定して復号鍵をクエリーしたときにのみ，チャレンジ暗号文の要素 $\boldsymbol{c}_0$ の第 2 成分 r_0，復号鍵の要素 $\boldsymbol{k}_0^*$ の第 2 成分 w_0，同じく復号鍵の要素 $\boldsymbol{k}_i^*$ の第 (n_t+1)〜$2n_t$ 成分に含まれる $\{a_i\}_{i=1,\cdots,\ell}$ から導出される要素 a_0 の 3 点が，$w_0 = a_0/r_0$ という correlation を起こしてしまい，正しく semi-functional key がシミュレートできない点である．言い換えれば，3 個の変数のうち 2 個が独立変数で，1 個が従属変数になるという correlation が存在する．我々の実現方式では，これらの correlation を起こす要素をチャレンジ暗号文，プロキシー鍵，ユーザ秘密鍵にぞれぞれ 1 個づつ分散配置するアルゴリズム構成としたことで，チャレンジ暗号文を復号できるアクセス構造を指定した場合においても，自明な攻撃を防ぐために攻撃者はプロキシー鍵かユーザ秘密鍵のいずれか一方しかクエリーできないという制約から，上記で述べた correlation を起こす 3 変数のうち 2 変数しか入手することができず，攻撃者のビューからは correlation が見えなくなる．そのため，Dual system encryption の枠組みを用いて正しくゲームがシミュレートでき，安全性が証明できる．

Proof. Theorem1 を証明するためのゲーム列を下記に示す．

- **Game0:** オリジナルのゲーム．チャレンジ暗号文は，正規の手順で生成した基底を用いて，下記のように正規の暗号文を生成する．

  $$\boldsymbol{c}_0 := (\omega, 0, \zeta, 0, \varphi_0)_{\mathbb{B}_0},$$
  $$\boldsymbol{c}_t := (\omega \overrightarrow{x}_t, 0^{n_t}, 0^{n_t}, \varphi_t)_{\mathbb{B}_t} \text{ for } (t, \overrightarrow{x}_t) \in \Gamma,$$
  $$c_{d+1} := g_T^{\zeta} m^{(b)},$$

  同様に，プロキシー鍵とユーザ秘密鍵も正規の手順で生成した基底を用いて正規の手順で生成する．

  $$\boldsymbol{k}_0^* := (-s_0, 0, 1, \eta_0, 0)_{\mathbb{B}_0^*},$$
  for $i = 1, \ldots, \ell$;
  　if $\rho(i) = (t, \overrightarrow{v}_i)$, $\boldsymbol{k}_i^* := (s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i, 0^{n_t}, \overrightarrow{\eta_i}, 0)_{\mathbb{B}_t^*}$,
  　if $\rho(i) = \neg(t, \overrightarrow{v}_i)$, $\boldsymbol{k}_i^* := (s_i \overrightarrow{v}_i, 0^{n_t}, \overrightarrow{\eta_i}, 0)_{\mathbb{B}_t^*}$,
  $$\mathsf{pxk}_{\mathbb{S}} := (\mathbb{S}, \boldsymbol{k}_1^*, \ldots, \boldsymbol{k}_\ell^*),$$
  $$\mathsf{sk}_{\mathbb{S}} := (\boldsymbol{k}_0^*),$$

- **Game1:** チャレンジ暗号文を下記の様に生成する点だけが Game0 と異なる．

$$
\begin{aligned}
&r_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \overrightarrow{r}_t \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n_t},\\
&\boldsymbol{c}_0 := (\omega, \boxed{r_0}, \zeta, 0, \varphi_0)_{\mathbb{B}_0},\\
&\boldsymbol{c}_t := (\omega \overrightarrow{x}_t, \boxed{\overrightarrow{r}_t}, 0^{n_t}, \varphi_t)_{\mathbb{B}_t} \text{ for } (t, \overrightarrow{x}_t) \in \Gamma,
\end{aligned}
$$

- **Game2-h^+ $(h = 0, \cdots, \nu - 1)$:** プロキシー鍵，ユーザ秘密鍵，チャレンジ暗号文の下記の要素の生成方法だけが Game2-h と異なる．なお，Game2-0 は Game1 と同一である．

$$
\begin{aligned}
&w_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \overrightarrow{g} \xleftarrow{\mathsf{U}} \mathbb{F}_q^r, \overrightarrow{a}^T := (a_1, \ldots, a_\ell)^T := M \cdot \overrightarrow{g}^T, \pi_i \xleftarrow{\mathsf{U}} \mathbb{F}_q (i = 1, \ldots, \ell),\\
&Z_t \xleftarrow{\mathsf{U}} GL(n_t, \mathbb{F}_q), U_t := (Z_t^{-1})^T \text{ for } t = 1, \ldots, d,\\
&\boldsymbol{k}_0^* := (-s_0, \boxed{w_0}, 1, \eta_0, 0)_{\mathbb{B}_0^*},\\
&\text{for } i = 1, \ldots, \ell;\\
&\quad \text{if } \rho(i) = (t, \overrightarrow{v}_i),\ \boldsymbol{k}_i^* := (s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i, \boxed{(a_i \overrightarrow{e}_{t,1} + \pi_i \overrightarrow{v}_i) \cdot Z_t}, \overrightarrow{\eta_i}, 0)_{\mathbb{B}_t^*},\\
&\quad \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i),\ \boldsymbol{k}_i^* := (s_i \overrightarrow{v}_i, \boxed{a_i \overrightarrow{v}_i \cdot Z_t}, \overrightarrow{\eta_i}, 0)_{\mathbb{B}_t^*},\\
&\boldsymbol{c}_t := (\omega \overrightarrow{x}_t, \boxed{\overrightarrow{x}_t \cdot U_t}, 0^{n_t}, \varphi_t)_{\mathbb{B}_t} \text{ for } (t, \overrightarrow{x}_t) \in \Gamma,
\end{aligned}
$$

- **Game2-$(h+1)$ $(h = 0, \cdots, \nu - 1)$:** プロキシー鍵とチャレンジ暗号文の下記の要素の生成方法だけが Game2-h^+ と異なる．

$$
\begin{aligned}
&\overrightarrow{r}_t \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n_t},\\
&\boldsymbol{k}_0^* := (-s_0, w_0, 1, \eta_0, 0)_{\mathbb{B}_0^*},\\
&\text{for } i = 1, \ldots, \ell;\\
&\quad \text{if } \rho(i) = (t, \overrightarrow{v}_i),\ \boldsymbol{k}_i^* := (s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i, \boxed{0^{n_t}}, \overrightarrow{\eta_i}, 0)_{\mathbb{B}_t^*},\\
&\quad \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i),\ \boldsymbol{k}_i^* := (s_i \overrightarrow{v}_i, \boxed{0^{n_t}}, \overrightarrow{\eta_i}, 0)_{\mathbb{B}_t^*},\\
&\boldsymbol{c}_t := (\omega \overrightarrow{x}_t, \boxed{\overrightarrow{r}_t}, 0^{n_t}, \varphi_t)_{\mathbb{B}_t} \text{ for } (t, \overrightarrow{x}_t) \in \Gamma,,
\end{aligned}
$$

- **Game3:** チャレンジ暗号文の下記の要素の生成方法だけが Game2-ν と異なる．

$$
\begin{aligned}
&\zeta' \xleftarrow{\mathsf{U}} \mathbb{F}_q,\\
&\boldsymbol{c}_0 := (\omega, r_0, \boxed{\zeta'}, 0, \varphi_0)_{\mathbb{B}_0},\\
&c_{d+1} := g_T^{\zeta} m^{(b)},
\end{aligned}
$$

Game0 が Definition12 に示したオリジナルのゲームであり，この要素を徐々に変えていき，Game3 まで変化させる．前のゲームから変化する点は，四角枠で囲った部分である．Game0，1，2-h，2-h^+，3 の攻撃者 $\mathcal{A}$ のアドバンテージを $\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(2-h)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)$ とする．このとき，それぞれのアドバンテージの違いは後で示す Lemma3, 4, 6, 7, 8 の様になるため，$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RKP-FE}}(\lambda)$ は下記の計算できる．

$$
\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RKP-FE}}(\lambda) &= \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda)\\
&\leq \left| \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda) \right|
\end{aligned}
$$

$$
\begin{aligned}
&+\sum_{h=0}^{\nu-1}\left|\mathsf{Adv}_{\mathcal{A}}^{(2-h)}(\lambda)-\mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda)\right| \\
&+\sum_{h=0}^{\nu-1}\left|\mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda)-\mathsf{Adv}_{\mathcal{A}}^{(2-(h+1))}(\lambda)\right| \\
&+\left|\mathsf{Adv}_{\mathcal{A}}^{(2-\nu)}(\lambda)-\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)\right| \\
&+\left|\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)\right| \\
\leq\ & \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{P1}}(\lambda)+\sum_{h=0}^{\nu-1}\mathsf{Adv}_{\mathcal{B}_{2,h}^+}^{\mathsf{P2}}(\lambda) \\
&+\sum_{h=0}^{\nu-1}\mathsf{Adv}_{\mathcal{B}_{2,h+1}}^{\mathsf{P2}}(\lambda)+(2d\nu+6\nu+1)/q \\
\leq\ & \mathsf{Adv}_{\mathcal{E}_1}^{\mathsf{DLIN}}(\lambda) \\
&+\sum_{h=0}^{\nu-1}\left(\mathsf{Adv}_{\mathcal{E}_{2,h}^+}^{\mathsf{DLIN}}(\lambda)+\mathsf{Adv}_{\mathcal{E}_{2,h+1}}^{\mathsf{DLIN}}(\lambda)\right) \\
&+(2d\nu+16\nu+d+7)/q
\end{aligned}
$$

以上により，Theorem1 が成り立つ． □

次に，上記証明で引用した Lemma を下記に示す．

Lemma 3. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$|\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda)-\mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{P1}}(\lambda)$ となる計算量仮定 *Problem1* に対する確率的多項式時間攻撃者 $\mathcal{B}_1$ が存在する．

Proof. 攻撃者 $\mathcal{A}$ を利用して，Problem1 を解く確率的アルゴリズム $\mathcal{B}_1$ を下記のように構成する．

1. $\mathcal{B}_1$ は Problem1 インスタンス $(\mathbf{param}_{\overrightarrow{n}}, \mathbb{B}_0, \hat{\mathbb{B}_0^*}, \boldsymbol{e}_{\beta,0}, \{\mathbb{B}_t, \hat{\mathbb{B}_t^*}, \boldsymbol{e}_{\beta,t,1}, \boldsymbol{e}_{t,j}\}_{t=1,\ldots,d;j=2,\ldots,n_t})$ を受け取る．

2. $\mathcal{B}_1$ は，Problem1 インスタンスから公開パラメータ $\mathsf{mpk} := (1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\hat{\mathbb{B}}_t\}_{t=0,\ldots,d})$ を計算し，攻撃者 $\mathcal{A}$ に与える．ここで，$\hat{\mathbb{B}}_0 := (\boldsymbol{b}_{0,1}, \boldsymbol{b}_{0,3}, \boldsymbol{b}_{0,5})$, $\hat{\mathbb{B}}_t := (\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,n_t}, \boldsymbol{b}_{t,3n_t+1})$ である．

3. 攻撃者 $\mathcal{A}$ からアクセス構造 $\mathbb{S}$ に対する鍵生成クエリを受け取ったら，Problem1 インスタンスに含まれる基底 $\{\hat{\mathbb{B}_t^*}\}_{t=0,\ldots,d}$ を用いて Game0 に記載の手順でプロキシー鍵 $\mathsf{pxk}_{\mathbb{S}_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\mathbb{S}_\iota}$ を生成する．CorruptProxyKey クエリではプロキシー鍵 $\mathsf{pxk}_{\mathbb{S}_\iota}$ を，CorruptSecretKey クエリではユーザ秘密鍵 $\mathsf{sk}_{\mathbb{S}_\iota}$ を攻撃者 $\mathcal{A}$ に与える．

4. 攻撃者 $\mathcal{A}$ からチャレンジメッセージ $(m^{(0)}, m^{(1)})$，属性集合 $\Gamma := \{(t, \overrightarrow{x}_t) | 1 \leq t \leq d\}$ を受け取ったら，下記の様にチャレンジ暗号文を生成し，攻撃者 $\mathcal{A}$ に送付する．
   $\zeta \xleftarrow{\mathsf{U}} \mathbb{F}_q, b \xleftarrow{\mathsf{U}} \{0,1\}, \boldsymbol{c}_0 := \boldsymbol{e}_{\beta,0} + \zeta \boldsymbol{b}_{0,3}$,
   $\boldsymbol{c}_t := x_{t,1}\boldsymbol{e}_{\beta,t,1} + \sum_{j=2}^{n_t} x_{t,j}\boldsymbol{e}_{t,j}, c_{d+1} := g_T^{\zeta} m^{(b)}$,

5. 攻撃者 $\mathcal{A}$ からクエリーを受け取ったら，ステップ (3) と同様に応答する．

6. 攻撃者 $\mathcal{A}$ がビット b' を出力したら，$\mathcal{B}_1$ は $b = b'$ の場合は 1 を，そうでなければ 0 を出力する．

上記シミュレーションで生成される暗号文について確認していく．
まず $\beta = 0$ の時，暗号文は下記の様に計算される．

$$\begin{aligned}
\boldsymbol{c}_0 &:= \boldsymbol{e}_{\beta,0} + \zeta \boldsymbol{b}_{0,3} \\
&= (\omega, 0, 0, 0, \gamma_0)_{\mathbb{B}_0} + (0, 0, \zeta, 0, 0)_{\mathbb{B}_0} \\
&= (\omega, 0, \zeta, 0, \gamma_0)_{\mathbb{B}_0} \\
\boldsymbol{c}_t &:= x_{t,1} \boldsymbol{e}_{\beta,t,1} + \sum_{j=2}^{n_t} x_{t,j} \boldsymbol{e}_{t,j} \\
&= x_{t,1} (\omega \overrightarrow{e}_{t,1}, 0^{n_t}, 0^{n_t}, \gamma_t)_{\mathbb{B}_t} + \sum_{j=2}^{n_t} x_{t,j} (\omega \overrightarrow{e}_{t,j}, 0^{n_t}, 0^{n_t}, 0)_{\mathbb{B}_t} \\
&= (\omega \overrightarrow{x}_t, 0^{n_t}, 0^{n_t}, \gamma_t)_{\mathbb{B}_t} \\
c_{d+1} &:= g_T^{\zeta} m^{(b)}
\end{aligned}$$

まず，暗号文の形式としては，Game0 にて定義した通りの暗号文になっている．その分布については，$\omega, \gamma_0, \gamma_t,$ は Problem1 にて独立一様分布に選ばれており，ζ も独立に選ばれている．そのため，Game0 の暗号文を正しくシミュレートしている．

次に $\beta = 1$ の時の暗号文について，下記の様に計算される．

$$\begin{aligned}
\boldsymbol{c}_0 &:= \boldsymbol{e}_{\beta,0} + \zeta \boldsymbol{b}_{0,3} \\
&= (\omega, z_0, 0, 0, \gamma_0)_{\mathbb{B}_0} + (0, 0, \zeta, 0, 0)_{\mathbb{B}_0} \\
&= (\omega, z_0, \zeta, 0, \gamma_0)_{\mathbb{B}_0} \\
\boldsymbol{c}_t &:= x_{t,1} \boldsymbol{e}_{\beta,t,1} + \sum_{j=2}^{n_t} x_{t,j} \boldsymbol{e}_{t,j} \\
&= x_{t,1} (\omega \overrightarrow{e}_{t,1}, \overrightarrow{z}_t, 0^{n_t}, \gamma_t)_{\mathbb{B}_t} + \sum_{j=2}^{n_t} x_{t,j} (\omega \overrightarrow{e}_{t,j}, 0^{n_t}, 0^{n_t}, 0)_{\mathbb{B}_t} \\
&= (\omega \overrightarrow{x}_t, \overrightarrow{z}_t, 0^{n_t}, \gamma_t)_{\mathbb{B}_t} \\
c_{d+1} &:= g_T^{\zeta} m^{(b)}
\end{aligned}$$

まず，暗号文の形式としては，Game1 にて定義した通りの semi-functional な暗号文になっている．その分布については，$\omega, \gamma_0, \gamma_t, z_0, \overrightarrow{z}_t$ は Problem1 にて独立一様分布に選ばれており，ζ も独立に選ばれている．そのため，Game1 の暗号文を正しくシミュレートしている．

よって，2 つのゲームを識別することは Problem1 インスタンスを識別することと同等である． □

Lemma 4. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$|\mathsf{Adv}_{\mathcal{A}}^{(2-h)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}_{2,h}^+}^{\mathsf{P2}}(\lambda) + (d+3)/q$ となる計算量仮定 *Problem2* に対する確率的多項式時間攻撃者 $\mathcal{B}_2^+$ が存在する．

Proof. 攻撃者 $\mathcal{A}$ を利用して，Problem2 を解く確率的アルゴリズム $\mathcal{B}_2^+$ を下記のように構成する．

1. $\mathcal{B}_2^+$ はProblem2インスタンス $(\mathbf{param}_{\overrightarrow{n}}, \hat{\mathbb{B}}_0, \mathbb{B}_0^*, \boldsymbol{h}_{\beta,0}^*, \boldsymbol{e}_0, \{\hat{\mathbb{B}}_t, \mathbb{B}_t^*, \boldsymbol{h}_{\beta,t,j}, \boldsymbol{e}_{t,j}\}_{t=1,\ldots,d;j=2,\ldots,n_t})$ を受け取る．

2. $\mathcal{B}_2^+$ は，Problem2 インスタンスから公開パラメータ $\mathsf{mpk} := (1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\hat{\mathbb{B}}'_t\}_{t=0,\ldots,d})$ を計算し，攻撃者 $\mathcal{A}$ に与える．ここで，$\hat{\mathbb{B}}'_0 := (\boldsymbol{b}_{0,1}, \boldsymbol{b}_{0,3}, \boldsymbol{b}_{0,5}), \hat{\mathbb{B}}'_t := (\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,n_t}, \boldsymbol{b}_{t,3n_t+1})$ である．

3. 攻撃者 $\mathcal{A}$ からアクセス構造 $\mathbb{S}$ に対する ι 番目の鍵生成クエリを受け取ったら，下記のようにプロキシー鍵 $\mathsf{pxk}_{\mathbb{S}_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\mathbb{S}_\iota}$ を生成する．

   (a) $1 \leq \iota \leq h$ の場合，Problem2 インスタンスに含まれる $\{\mathbb{B}_t^*\}_{t=0,\ldots,d}$ を用いて，Game2-$(h+1)$ に示した手順で生成する．

   (b) $\iota = h+1$ の場合，Problem2 インスタンスを用いて下記の様に生成する．ここで，$M = (M_{i,k})_{i=1,\ldots,\ell;k=1,\ldots,r}$ とする．

$$
\begin{array}{l}
\pi_t, \mu_t, g_k, \tilde{\mu}_k \xleftarrow{\cup} \mathbb{F}_q \text{ for } t = 1, \ldots, d; k = 1, \ldots, r, \\
\tilde{\boldsymbol{p}}_{\beta,0}^* := \sum_{k=1}^r \left(g_k \boldsymbol{h}_{\beta,0}^* + \tilde{\mu}_k \boldsymbol{b}_{0,1}^* \right), \\
\text{for } t = 1, \ldots, d; k = 1, \ldots, r; j = 1, \ldots, n_t; \\
\quad \boldsymbol{p}_{\beta,t,j}^* := \pi_t \boldsymbol{h}_{\beta,t,j}^* + \mu_t \boldsymbol{b}_{t,j}^*, \\
\quad \tilde{\boldsymbol{p}}_{\beta,t,k,j}^* := g_k \boldsymbol{h}_{\beta,t,j}^* + \tilde{\mu}_k \boldsymbol{b}_{t,j}^*, \\
\boldsymbol{k}_0^* := -\tilde{\boldsymbol{p}}_{\beta,0}^* + \boldsymbol{b}_{0,3}^*, \\
\text{for } i = 1, \ldots, \ell; \\
\quad \text{if } \rho(i) = (t, \overrightarrow{v}_i), \\
\quad\quad \boldsymbol{k}_i^* := \sum_{j=1}^{n_t} v_{i,j} \boldsymbol{p}_{\beta,t,j}^* + \sum_{k=1}^r M_{i,k} \tilde{\boldsymbol{p}}_{\beta,t,k,1}^*, \\
\quad \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i), \\
\quad\quad \boldsymbol{k}_i^* := \sum_{j=1}^{n_t} v_{i,j} \left(\sum_{k=1}^r M_{i,k} \tilde{\boldsymbol{p}}_{\beta,t,k,j}^* \right),
\end{array}
$$

   (c) $\iota \geq h+2$ の場合，Problem2 インスタンスの $\{\mathbb{B}_t^*\}_{t=0,\ldots,d}$ を用いて，Game1 に示した手順で生成する．

   CorruptProxyKey クエリではプロキシー鍵 $\mathsf{pxk}_{\mathbb{S}_\iota}$ を，CorruptSecretKey クエリではユーザ秘密鍵 $\mathsf{sk}_{\mathbb{S}_\iota}$ を攻撃者 $\mathcal{A}$ に与える．

4. 攻撃者 $\mathcal{A}$ からチャレンジメッセージ $(m^{(0)}, m^{(1)})$，属性集合 $\Gamma := \{(t, \overrightarrow{x}_t) | 1 \leq t \leq d\}$ を受け取ったら，下記の様にチャレンジ暗号文を生成し，攻撃者 $\mathcal{A}$ に送付する．
   $\zeta \xleftarrow{\cup} \mathbb{F}_q, b \xleftarrow{\cup} \{0,1\}, \boldsymbol{q}_0 \xleftarrow{\cup} \mathsf{span}\langle \boldsymbol{b}_{0,5} \rangle,$
   $\boldsymbol{q}_t \xleftarrow{\cup} \mathsf{span}\langle \boldsymbol{b}_{t,3n_t+1} \rangle, \boldsymbol{c}_0 := \boldsymbol{e}_0 + \zeta \boldsymbol{b}_{0,3} + \boldsymbol{q}_0,$
   $\boldsymbol{c}_t := \sum_{j=1}^{n_t} x_{t,j} \boldsymbol{e}_{t,j} + \boldsymbol{q}_t, c_{d+1} := g_T^\zeta m^{(b)}.$

5. 攻撃者 $\mathcal{A}$ からクエリーを受け取ったら，ステップ (3) と同様に応答する．

6. 攻撃者 $\mathcal{A}$ がビット b' を出力したら，$\mathcal{B}_2^+$ は $b = b'$ の場合は 1 を，そうでなければ 0 を出力する．

なお，ステップ (3)-(b) の $\tilde{\boldsymbol{p}}^*_{\beta,0}, \boldsymbol{p}^*_{\beta,t,j}, \tilde{\boldsymbol{p}}^*_{\beta,t,k,j}$ は，$\theta_t := \pi_t\delta + \mu_t, f_k := g_k\delta + \tilde{\mu}_k, s_0 := \sum_{k=1}^r f_k, a_0 := \sum_{k=1}^r g_k$ としたとき，下記の様に計算できる．

$$
\begin{aligned}
&\tilde{\boldsymbol{p}}^*_{0,0} = (s_0, 0, 0, a_0\delta_0, 0)_{\mathbb{B}^*_0}, \tilde{\boldsymbol{p}}^*_{1,0} = (s_0, a_0u_0, 0, a_0\delta_0, 0)_{\mathbb{B}^*_0},\\
&\boldsymbol{p}^*_{0,t,j} := (\theta_t \overrightarrow{e}_{t,j}, 0^{n_t}, \pi_t \overrightarrow{\delta}_{t,j}, 0)_{\mathbb{B}^*_t},\\
&\tilde{\boldsymbol{p}}^*_{0,t,k,j} := (f_k \overrightarrow{e}_{t,j}, 0^{n_t}, g_k \overrightarrow{\delta}_{t,j}, 0)_{\mathbb{B}^*_t},\\
&\boldsymbol{p}^*_{1,t,j} := (\theta_t \overrightarrow{e}_{t,j}, \pi_t \overrightarrow{u}_{t,j}, \pi_t \overrightarrow{\delta}_{t,j}, 0)_{\mathbb{B}^*_t},\\
&\tilde{\boldsymbol{p}}^*_{1,t,k,j} := (f_k \overrightarrow{e}_{t,j}, g_k \overrightarrow{u}_{t,j}, g_k \overrightarrow{\delta}_{t,j}, 0)_{\mathbb{B}^*_t}
\end{aligned}
$$

ここで，$\delta, \delta_0, \overrightarrow{e}_{t,j}, \overrightarrow{u}_{t,j}, \overrightarrow{\delta}_{t,j}$ は Problem2 にて定義された値であり，$\delta, \delta_0, \overrightarrow{\delta}_{t,j}$ は独立して一様分布する．さらに，$\pi_t, \mu_t, g_k, \tilde{\mu}_k$ は $\mathbb{F}_q$ から独立して一様分布に選んでおり，$\{\theta_t, \pi_t\}_{t=1,\ldots,d}$, $\{f_k, g_k\}_{k=1,\ldots,r}$ もそれぞれ独立に一様分布となる．

始めに，暗号文について確認する．Problem2 の要素を用いて，暗号文は下記のように構成されている．なお，式の途中まで Problem2 に合わせて $\overrightarrow{a}_{t,i}$ と記載しているが，最後はゲーム列に合わせて U_t としている．

$$
\begin{aligned}
\boldsymbol{c}_0 &:= \boldsymbol{e}_0 + \zeta\boldsymbol{b}_{0,3} + \boldsymbol{q}_0\\
&= (\omega, \tau z_0, 0, 0, 0)_{\mathbb{B}_0} + (0, 0, \zeta, 0, 0)_{\mathbb{B}_0} + (0, 0, 0, 0, \varphi_0)_{\mathbb{B}_0}\\
&= (\omega, \tau z_0, \zeta, 0, \varphi_0)_{\mathbb{B}_0}\\
\boldsymbol{c}_t &:= \sum_{j=1}^{n_t} x_{t,j}\boldsymbol{e}_{t,j} + \boldsymbol{q}_t\\
&= \sum_{j=1}^{n_t} x_{t,j}(\omega \overrightarrow{e}_{t,i}, \tau \overrightarrow{z}_{t,i}, 0^{n_t}, 0)_{\mathbb{B}_t} + (0, 0, 0, 0, \varphi_t)_{\mathbb{B}_t}\\
&= (\omega \overrightarrow{x}_t, \tau \overrightarrow{x}_t U_t, 0^{n_t}, 0)_{\mathbb{B}_t} + (0, 0, 0, 0, \varphi_t)_{\mathbb{B}_t}\\
&= (\omega \overrightarrow{x}_t, \tau \overrightarrow{x}_t U_t, 0^{n_t}, \varphi_t)_{\mathbb{B}_t}\\
c_{d+1} &:= g_T^{\zeta} m^{(b)}
\end{aligned}
$$

暗号文の構成としては，Game1 で定義された形式と同一である．次に分布について確認するが，$\zeta, \varphi_0, \varphi_t$ は独立一様分布に選んでおり，ω, τ は Problem2 で一様分布に取られている値である．また，U_t も各 t 毎に $GL(n_t, \mathbb{F}_q)$ から一様分布で選んでいる．そのため，上記暗号文は Game1 にて規定した semi-functional な暗号文と分布が同一となる．ただし，$z_0 = u_0^{-1}$ であるため，この点だけ別途プロキシー鍵やユーザ秘密鍵側との依存性の検証が必要となる．

$\beta = 0$ の場合，プロキシー鍵とユーザ秘密鍵は下記のように計算される．

$$
\begin{aligned}
&\boldsymbol{k}^*_0 = (-s_0, 0, 1, -a_0\delta_0, 0)_{\mathbb{B}^*_0},\\
&\text{for } i = 1, \ldots, \ell;\\
&\quad \text{if } \rho(i) = (t, \overrightarrow{v}_i),\\
&\qquad \boldsymbol{k}^*_i = (s_i \overrightarrow{e}_{t,1} + \theta_t \overrightarrow{v}_i, 0^{n_t}, \overrightarrow{\eta}_i, 0)_{\mathbb{B}^*_t},\\
&\quad \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i),\\
&\qquad \boldsymbol{k}^*_i = (s_i \overrightarrow{v}_i, 0^{n_t}, \overrightarrow{\eta}_i, 0)_{\mathbb{B}^*_t}
\end{aligned}
$$

ここで，$\overrightarrow{\eta}_i := \pi_t \sum_{j=1}^{n_t} v_{i,j} \overrightarrow{\delta}_{t,j} + (\sum_{k=1}^r g_k M_{i,k}) \overrightarrow{\delta}_{t,1}$，$\overrightarrow{\eta'}_i = (\sum_{k=1}^r M_{i,k} g_k)(\sum_{j=1}^{n_t} v_{i,j} \overrightarrow{\delta}_{t,j})$ であるが，$\overrightarrow{\delta}_{t,j}$ は $\mathbb{F}_q^{n_t}$ 上を独立して一様分布するため，$\overrightarrow{\eta}_i, \overrightarrow{\eta'}_i$ もそれぞれ $\mathbb{F}_q^{n_t}$ 上を独立し

て一様分布する．また，δ_0, θ_t も $\mathbb{F}_q$ 上を独立して一様分布することから，各変数の分布も含めて正しく Game 2-h の秘密鍵をシミュレートしている．そのため，高島らが示したのと同様に確率 $(d+2)/q$ の例外を除いて，正しく Game2 $-$ h のチャレンジ暗号文，プロキシー鍵，ユーザ秘密鍵がシミュレートできる．

$\beta = 1$ の場合，プロキシー鍵とユーザ秘密鍵は下記のように計算される．

$$
\begin{aligned}
&\boldsymbol{k}_0^* = (-s_0, -w_0, 1, -a_0\delta_0, 0)_{\mathbb{B}_0^*},\\
&\text{for } i = 1, \ldots, \ell;\\
&\quad \text{if } \rho(i) = (t, \overrightarrow{v}_i),\\
&\qquad \boldsymbol{k}_i^* = (s_i \overrightarrow{e}_{t,1} + \theta_t \overrightarrow{v}_i, (\pi_t \overrightarrow{v}_i + a_i \overrightarrow{e}_{t,1}) \cdot Z_t, \overrightarrow{\eta}_i, 0)_{\mathbb{B}_t^*},\\
&\quad \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i),\\
&\qquad \boldsymbol{k}_i^* = (s_i \overrightarrow{v}_i, a_i \overrightarrow{v}_i \cdot Z_t, \overrightarrow{\eta}_i, 0)_{\mathbb{B}_t^*}
\end{aligned}
$$

ここで，$\beta = 0$ のケースとの違いは，各 $\boldsymbol{k}_i^*$ の第 2 成分であり，それ以外の要素については $\beta = 0$ のケースと同じであることから正しくシミュレートできている．高島らは，暗号文の要素 $\boldsymbol{c}_0$ の第 2 要素 r_0 と，秘密鍵の要素 $\boldsymbol{k}_0^*$ の第 2 要素 w_0 に $w_0 := a_0/r_0$ という関係性があるため，a_0 の独立性を確認するため，$a_0 = \overrightarrow{1} \cdot \overrightarrow{g}^T$，$(a_1, \ldots, a_\ell)^T = M \cdot \overrightarrow{g}^T$，$U_t = (Z_t^{-1})^T$ という関係があることから，$\overrightarrow{w}_i := (\pi_t \overrightarrow{v}_i + a_i \overrightarrow{e}_{t,1}) \cdot Z_t$，$\overrightarrow{w}_i := a_i \overrightarrow{v}_i \cdot Z_t$，$\overrightarrow{r}_t := \overrightarrow{x}_t \cdot U_t$ の独立性の検証が必要となると指摘していた．そこで下記の 5 パターンについて確認する．

1. $\gamma(i) = 1$ で $[\rho(i) = (t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \in \Gamma \wedge \overrightarrow{v}_i \cdot \overrightarrow{x}_t = 0]$
   $\overrightarrow{w}_i \cdot \overrightarrow{r}_t = (\pi_t \overrightarrow{v}_i + a_i \overrightarrow{e}_{t,1}) \cdot Z_t \cdot \overrightarrow{x}_t \cdot U_t = a_i$ であることから，$(\overrightarrow{w}_i, \overrightarrow{r}_t)$ は C_{a_i} 上を一様分布する．

2. $\gamma(i) = 1$ で $[\rho(i) = \neg(t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \in \Gamma \wedge \overrightarrow{v}_i \cdot \overrightarrow{x}_t \neq 0]$
   $\overrightarrow{w}_i \cdot \overrightarrow{r}_t = a_i(\overrightarrow{v}_i \cdot \overrightarrow{x}_t)$ であることから，$(\overrightarrow{w}_i, \overrightarrow{r}_t)$ は $C_{a_i(\overrightarrow{v}_i \cdot \overrightarrow{x}_t)}$ 上を一様分布する．

3. $\gamma(i) = 0$ で $[\rho(i) = (t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \in \Gamma]$
   $(\overrightarrow{w}_i, \overrightarrow{r}_t)$ は $C_{(\overrightarrow{v}_i \cdot \overrightarrow{x}_t)\pi_i + a_i}$ 上を一様分布する．ここで，π_i は $\mathbb{F}_q$ 上を独立一様分布する変数であるため，$(\overrightarrow{w}_i, \overrightarrow{r}_t)$ は $\mathbb{F}_q^{n_t}$ 上を独立一様分布する．

4. $\gamma(i) = 0$ で $[\rho(i) = \neg(t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \in \Gamma]$
   $(\overrightarrow{w}_i, \overrightarrow{r}_t)$ は C_0 上を一様分布する．

5. $[\rho(i) = (t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \notin \Gamma]$ もしくは $[\rho(i) = \neg(t, \overrightarrow{v}_i) \wedge (t, \overrightarrow{x}_t) \notin \Gamma]$
   $\overrightarrow{v}_i$ に対応する属性ベクトル $\overrightarrow{x}_t$ が存在しないので，$\overrightarrow{v}_i$ は $\mathbb{F}_q^{n_t}$ 上を独立一様分布する．

高島らが示したように，チャレンジ暗号文を復号できない場合は上記のケース 3〜5 の場合に該当し，$\overrightarrow{w}_i, \overrightarrow{w}_i, \overrightarrow{r}_t$ は a_0 とは独立に一様分布となる．そのため，Problem2 の δ が 0 になる例外を除いて，チャレンジ暗号文，プロキシー鍵，ユーザ秘密鍵は正しくシミュレートできる．

一方，チャレンジ暗号文が復号できるアクセス構造が指定された場合，上記のケース (1), (2) について考慮してプロキシー鍵とユーザ秘密鍵の分布を評価する必要がある．自明な攻撃を防ぐために，攻撃者はプロキシー鍵かユーザ秘密鍵のいずれか一方のみが取得できるように制約しているので，それぞれのケースについて考察する．

[プロキシー鍵のみを取得する場合]
攻撃者がプロキシー鍵のみ取得でき，ユーザ秘密鍵が取得できないケースについて考察する．プロキシー鍵には，$\overrightarrow{w}_i, \overrightarrow{\overrightarrow{w}}_i$ が含まれるため，その分布について前述のケース (1) と (2) についての評価が必要となる．ケース (1) については，$\overrightarrow{w}_i \cdot \overrightarrow{r}_t = (\pi_t \overrightarrow{v}_i + a_i \overrightarrow{e}_{t,1}) \cdot Z_t \cdot \overrightarrow{x}_t \cdot U_t = a_i$ であることから，$(\overrightarrow{w}_i, \overrightarrow{r}_t)$ は C_{a_i} 上を独立一様分布する．同様にケース (2) についても，$\overrightarrow{w}_i \cdot \overrightarrow{r}_t = a_i(\overrightarrow{v}_i \cdot \overrightarrow{x}_t)$ であることから，$(\overrightarrow{w}_i, \overrightarrow{r}_t)$ は $C_{a_i(\overrightarrow{v}_i \cdot \overrightarrow{x}_t)}$ 上を一様分布する．これより，攻撃者はプロキシー鍵の $\overrightarrow{w}_i, \overrightarrow{\overrightarrow{w}}_i$ だけから $(a_1, \ldots, a_\ell)$ に関する情報を得ることはできないが，チャレンジ暗号文の $\overrightarrow{r}_t$ と内積値を取ることで $(a_1, \ldots, a_\ell)$ に関する情報が得られる．しかし，$(a_1, \ldots, a_\ell)$ に関する情報が得られても，攻撃者はチャレンジ暗号文とプロキシー鍵の correlation を生み出すユーザ秘密鍵の要素 $w_0 := a_0/r_0$ は入手できないため，これを入手できない攻撃者にとっては，チャレンジ暗号文とプロキシー鍵は独立一様分布に見える．

[ユーザ秘密鍵のみを取得する場合]
攻撃者がユーザ秘密鍵のみ取得でき，プロキシー鍵が取得できないケースについて考察する．この場合，$\boldsymbol{k}_0^*$ に含まれる $w_0 := a_0/r_0$ を得ることができるが，$a_0 = \sum g_k$ は独立一様分布な乱数であること，およびプロキシー鍵に含まれる $(a_1, \ldots, a_\ell)$ が得られないことから a_0 に関する追加の情報が得られず，攻撃者にとってはユーザ秘密鍵もチャレンジ暗号文とは独立一様分布に見えることが分かる．

以上の考察より，たとえ暗号化データを復号できる条件の下であっても，プロキシー鍵もしくはユーザ秘密鍵のいずれか一方しか入手できない攻撃者にとっては，Problem2 の δ が 0 になる例外を除いて，正しくプロキシー鍵やユーザ秘密鍵をシミュレートできていることが分かる．よって，2 つのゲームを識別することは Problem2 インスタンスを識別することと同等である． □

なお，上記証明においては，高島らが示した下記の Lemma を利用した．

Lemma 5. V を $\mathbb{F}_q^n$ 上の n 次元ベクトル空間，V^* をその双対空間とする．$p \in \mathbb{F}_q$ に対して，$C_p := \{(\overrightarrow{x}, \overrightarrow{v}) | \overrightarrow{x} \cdot \overrightarrow{v} = p\} \subset V \times V^*$ とする．また，$Z \xleftarrow{\mathsf{U}} GL(n, \mathbb{F}_q), U := (Z^{-1})^T$ とする．このとき，あらゆる $(\overrightarrow{x}, \overrightarrow{v}), (\overrightarrow{r}, \overrightarrow{w}) \in C_p$ に対して，$Pr[\overrightarrow{x}U = \overrightarrow{r} \wedge \overrightarrow{v}Z = \overrightarrow{w}] = Pr[\overrightarrow{x}Z = \overrightarrow{r} \wedge \overrightarrow{v}U = \overrightarrow{w}] = 1/\sharp C_p$ となる．

Lemma 6. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$|\mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2-(h+1))}(\lambda)| \leq \mathsf{Adv}_{\mathcal{B}_{2,h+1}}^{\mathsf{P2}}(\lambda) + (d+3)/q$ となる計算量仮定 *Problem2* に対する確率的多項式時間攻撃者 $\mathcal{B}_2$ が存在する．

Proof. 攻撃者 $\mathcal{A}$ を利用して，Problem2 を解く確率的アルゴリズム $\mathcal{B}_2$ を構成する．アルゴリズム $\mathcal{B}_2$ の動作はアルゴリズム $\mathcal{B}_2^+$ と以下の点を除いて同一である．

1. ステップ (3) のケース (b) において，$\boldsymbol{k}_0^*$ の計算方法が下記のようになる．
   $r_0' \xleftarrow{\mathsf{U}} \mathbb{F}_q, \boldsymbol{k}_0^* := -\tilde{\boldsymbol{p}}_{\beta,0}^* + r_0' \boldsymbol{b}_{0,2}^* + \boldsymbol{b}_{0,3}^*$
2. 最後のステップで，攻撃者 $\mathcal{A}$ がビット b' を出力したら，$\mathcal{B}_2$ は $b = b'$ の場合は 0 を，そうでなければ 1 を出力する．

分布に関しては，Lemma4 と同様に確認することができる．

$\beta = 0$ の場合は，$\delta = 0$ の場合，すなわち確率 $1/q$ を除いて，Game 2-(h+1) と分布が同一であることがわかる．

$\beta = 1$ の場合は，$\delta = 0$ の場合，および $\overrightarrow{r}_t = \overrightarrow{0}$ となる場合を除いて，Game 2-h^+ と分布が同一であることがわかる．その例外が生じる確率は $(d+2)/q$ となる．

よって，2 つのゲームを識別することは Problem2 インスタンスを識別することと同等である． □

Lemma 7. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) \leq \mathsf{Adv}^{(2-\nu)}_{\mathcal{A}}(\lambda) + 1/q$ である．

Proof. 新しい基底 $\mathbb{D}_0$ および $\mathbb{D}^*_0$ を定義する．

$$\begin{aligned}
&\theta \xleftarrow{\mathsf{U}} \mathbb{F}_q, \boldsymbol{d}_{0,2} := \boldsymbol{b}_{0,2} - \theta \boldsymbol{b}_{0,3}, \boldsymbol{d}^*_{0,3} := \boldsymbol{b}^*_{0,3} + \theta \boldsymbol{b}^*_{0,2},\\
&\mathbb{D}_0 := (\boldsymbol{b}_{0,1}, \boldsymbol{d}_{0,2}, \boldsymbol{b}_{0,3}, \boldsymbol{b}_{0,4}, \boldsymbol{b}_{0,5}),\\
&\mathbb{D}^*_0 := (\boldsymbol{b}^*_{0,1}, \boldsymbol{b}^*_{0,2}, \boldsymbol{d}^*_{0,3}, \boldsymbol{b}^*_{0,4}, \boldsymbol{b}^*_{0,5}),
\end{aligned}$$

この時，j 番目に生成したプロキシー鍵 $\mathsf{pxk}^{(j)}_{\mathrm{S}}$ の要素 $\boldsymbol{k}^{(j)*}_0$ は下記の様に表せる．

$$\begin{aligned}
\boldsymbol{k}^{(j)*}_0 &= (-s^{(j)}_0, w^{(j)}_0, 1, \eta^{(j)}_0, 0)_{\mathbb{B}^*_0}\\
&= (-s^{(j)}_0, w^{(j)}_0 + \theta, 1, \eta^{(j)}_0, 0)_{\mathbb{D}^*_0}\\
&= (-s^{(j)}_0, \vartheta^{(j)}_0, 1, \eta^{(j)}_0, 0)_{\mathbb{D}^*_0}\\
\boldsymbol{c}_0 &= (\omega, r_0, \zeta, 0, \phi_0)_{\mathbb{B}_0}\\
&= (\omega, r_0, \zeta + r_0\theta, 0, \phi_0)_{\mathbb{D}_0} = (\omega, r_0, \zeta', 0, \phi_0)_{\mathbb{D}_0}
\end{aligned}$$

ただし，$\vartheta^{(j)}_0 := w^{(j)}_0 + \theta, \zeta' := \zeta + r_0\theta$ である．

攻撃者のビューからすれば，基底 $\mathbb{B}_0$ も基底 $\mathbb{D}_0$ も公開パラメータ $\mathsf{mpk} := (1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\hat{\mathbb{B}}_t\}_{t=0,\ldots,d})$ と合致するため，どちらの基底で作られた暗号文でも区別はつかない．そのため，Game2$-\nu$ と Game3 は，乱数 $r_0 = 0$ の場合を除いて conceptual change である． □

Lemma 8. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) = 0$ である．

Proof. ζ および ζ' が独立した値であることから，攻撃者はメッセージに関する情報は得られない．ゆえに，$\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) = 0$ が成り立つ． □

3.8 Theorem2 の証明

高島ら [65] が示した証明に対して，3.7 節で示した修正と同様の修正を行うことで証明が可能である．ただし，暗号文ポリシー型はもう一工夫が必要である．Lemma10 の証明にて，correlation を起こす 3 要素のうち w_0 および $\{a_i\}_{i=1,\cdots,\ell}$ の 2 要素がチャレンジ暗号文に埋め込まれ，残りの一要素 r_0 がユーザ秘密鍵に埋め込まれるため，攻撃者がチャレンジ暗号文を復号可能な属性集合を指定してユーザ秘密鍵をクエリすると，correlation に気が付く可能性がある．しかし実際は，チャレンジ暗号文に埋め込まれた $\{a_i\}_{i=1,\cdots,\ell}$ は，$\overrightarrow{w}_i := (a_i \overrightarrow{e}_{t,1} + \pi_i \overrightarrow{v}_i) \cdot Z_t$ もしくは $\overrightarrow{w}_i := a_i \overrightarrow{v}_i \cdot Z_t$ という形で埋め込まれており，行列 Z_t が $GL(n_t, \mathbb{F}_q)$ から独立一様分布で選ばれているため，チャレンジ暗号文中の $\overrightarrow{w}_i, \overrightarrow{w}_i$ は a_i

に依存せずに $\mathbb{F}_q^{n_t}$ 上を一様分布する．これは，$\overrightarrow{w}_i, \overrightarrow{\overrightarrow{w}}_i$ から a_i に関する情報を得るためには，対になる $\overrightarrow{x}_t U_t$ が必要になることを意味する．そこで，対になる要素 $\overrightarrow{x}_t U_t$ がプロキシー鍵に埋め込まれているため，前述の correlation が攻撃者のビューでは見えなくなることを利用して安全性証明を行う．

Theorem2 の証明を下記に示す．

Proof. Theorem2 を証明するためのゲーム列を下記に示す．

- **Game0:** オリジナルのゲーム．チャレンジ暗号文は，正規の手順で生成した基底を用いて，下記のように正規の暗号文を生成する．

$$
\begin{aligned}
&\boldsymbol{c}_0 := (-s_0, 0, \zeta, 0, \eta_0)_{\mathbb{B}_0},\\
&\text{for } i = 1, \ldots, \ell;\\
&\quad \text{if } \rho(i) = (t, \overrightarrow{v}_i),\ \boldsymbol{c}_i := (s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i, 0^{n_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t},\\
&\quad \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i),\ \boldsymbol{c}_i := (s_i \overrightarrow{v}_i, 0^{n_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t},\\
&c_{d+1} := g_T^{\zeta} m,
\end{aligned}
$$

  同様に，プロキシー鍵とユーザ秘密鍵も正規の手順で生成した基底を用いて正規の手順で生成する．

$$
\begin{aligned}
&\boldsymbol{k}_0^* := (\delta, 0, 1, \varphi_0, 0)_{\mathbb{B}_0^*},\\
&\boldsymbol{k}_t^* := (\delta \overrightarrow{x}_t, 0^{n_t}, \overrightarrow{\varphi}_t, 0)_{\mathbb{B}_t^*} \text{ for } (t, \overrightarrow{x}_t) \in \Gamma,\\
&\mathsf{pxk}_\Gamma := (\Gamma, \{\boldsymbol{k}_t^*\}_{(t,\overrightarrow{x}_t)\in\Gamma}),\\
&\mathsf{sk}_\Gamma := (\boldsymbol{k}_0^*),
\end{aligned}
$$

- **Game1:** チャレンジ暗号文を下記の様に生成する点だけが Game0 と異なる．

$$
\begin{aligned}
&w_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \overrightarrow{w}_i, \overrightarrow{\overrightarrow{w}}_i \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n_t} \text{ for } i = 1, \ldots, \ell,\\
&\boldsymbol{c}_0 := (-s_0, \boxed{w_0}, \zeta, 0, \eta_0)_{\mathbb{B}_0},\\
&\text{for } i = 1, \ldots, \ell;\\
&\quad \text{if } \rho(i) = (t, \overrightarrow{v}_i),\ \boldsymbol{c}_i := (s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i, \boxed{\overrightarrow{w}_i}, 0^{n_t}, \eta_i)_{\mathbb{B}_t},\\
&\quad \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i),\ \boldsymbol{c}_i := (s_i \overrightarrow{v}_i, \boxed{\overrightarrow{\overrightarrow{w}}_i}, 0^{n_t}, \eta_i)_{\mathbb{B}_t},
\end{aligned}
$$

- **Game2-h^+ $(h = 0, \cdots, \nu - 1)$:** プロキシー鍵，ユーザ秘密鍵，チャレンジ暗号文の下記の要素の生成方法だけが Game2-h と異なる．なお，Game2-0 は Game1 と同一である．

$$
\begin{aligned}
&r_0 \xleftarrow{\mathsf{U}} \mathbb{F}_q, \overrightarrow{g} \xleftarrow{\mathsf{U}} \mathbb{F}_q^r, \overrightarrow{a}^T := (a_1, \ldots, a_\ell)^T := M \cdot \overrightarrow{g}^T, \pi_i \xleftarrow{\mathsf{U}} \mathbb{F}_q (i = 1, \ldots, \ell),\\
&Z_t \xleftarrow{\mathsf{U}} GL(n_t, \mathbb{F}_q), U_t := (Z_t^{-1})^T \text{ for } t = 1, \ldots, d,\\
&\boldsymbol{k}_0^* := (\delta, \boxed{r_0}, 1, \varphi_0, 0)_{\mathbb{B}_0^*},\\
&\boldsymbol{k}_t^* := (\delta \overrightarrow{x}_t, \boxed{\overrightarrow{x}_t \cdot U_t}, \overrightarrow{\varphi}_t, 0)_{\mathbb{B}_t^*} \text{ for } (t, \overrightarrow{x}_t) \in \Gamma,
\end{aligned}
$$

for $i = 1, \ldots, \ell$;
if $\rho(i) = (t, \overrightarrow{v}_i)$, $\boldsymbol{c}_i := (s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i, \boxed{(a_i \overrightarrow{e}_{t,1} + \pi_i \overrightarrow{v}_i) \cdot Z_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}$,
if $\rho(i) = \neg(t, \overrightarrow{v}_i)$, $\boldsymbol{c}_i := (s_i \overrightarrow{v}_i, \boxed{a_i \overrightarrow{v}_i \cdot Z_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}$,

- **Game2-$(h+1)$ $(h = 0, \cdots, \nu - 1)$:** プロキシー鍵とチャレンジ暗号文の下記の要素の生成方法だけが Game2-h^+ と異なる．

$\overrightarrow{w}_i, \overrightarrow{w}_i \xleftarrow{\mathsf{U}} \mathbb{F}_q^{n_t}$ for $i = 1, \ldots, \ell$,
$\boldsymbol{k}_t^* := (\delta \overrightarrow{x}_t, \boxed{0^{n_t}}, \overrightarrow{\varphi}_t, 0)_{\mathbb{B}_t^*}$ for $(t, \overrightarrow{x}_t) \in \Gamma$,
for $i = 1, \ldots, \ell$;
if $\rho(i) = (t, \overrightarrow{v}_i)$, $\boldsymbol{c}_i := (s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i, \boxed{\overrightarrow{w}_i}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}$,
if $\rho(i) = \neg(t, \overrightarrow{v}_i)$, $\boldsymbol{c}_i := (s_i \overrightarrow{v}_i, \boxed{\overrightarrow{w}_i}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}$,

- **Game3:** チャレンジ暗号文の下記の要素の生成方法だけが Game2-ν と異なる．

$\zeta' \xleftarrow{\mathsf{U}} \mathbb{F}_q$,
$\boldsymbol{c}_0 := (-s_0, \omega_0, \boxed{\zeta'}, 0, \eta_0)_{\mathbb{B}_0}$,
$c_{d+1} := g_T^{\zeta} m^{(b)}$,

Game0 が Definition15 に示したオリジナルのゲームであり，この要素を徐々に変えていき，Game3 まで変化させる．前のゲームから変化する点は，四角枠で囲った部分である．Game0，1，2-h，2-h^+，3 の攻撃者のアドバンテージを $\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(2-h)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda), \mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)$ とする．このとき，それぞれのアドバンテージの違いは後続の Lemma9, 10, 11, 12, 13 の様に計算することができるため，$\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RCP-FE}}(\lambda)$ は下記の様に計算できる．

$$
\begin{aligned}
\mathsf{Adv}_{\mathcal{A}}^{\mathsf{RCP-FE}}(\lambda) &= \mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) \\
&\leq \left|\mathsf{Adv}_{\mathcal{A}}^{(0)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(1)}(\lambda)\right| \\
&\quad + \sum_{h=0}^{\nu-1} \left|\mathsf{Adv}_{\mathcal{A}}^{(2-h)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda)\right| \\
&\quad + \sum_{h=0}^{\nu-1} \left|\mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2-(h+1))}(\lambda)\right| \\
&\quad + \left|\mathsf{Adv}_{\mathcal{A}}^{(2-\nu)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)\right| \\
&\quad + \left|\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda)\right| \\
&\leq \mathsf{Adv}_{\mathcal{B}_1}^{\mathsf{P1}}(\lambda) + \sum_{h=0}^{\nu-1} \mathsf{Adv}_{\mathcal{B}_{2,h}^+}^{\mathsf{P2}}(\lambda) \\
&\quad + \sum_{h=0}^{\nu-1} \mathsf{Adv}_{\mathcal{B}_{2,h+1}}^{\mathsf{P2}}(\lambda) + (2d\nu + 6\nu + d + 2)/q \\
&\leq \mathsf{Adv}_{\mathcal{E}_1}^{\mathsf{DLIN}}(\lambda)
\end{aligned}
$$

$$+\sum_{h=0}^{\nu-1}\left(\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{E}^{+}_{2,h}}(\lambda)+\mathsf{Adv}^{\mathsf{DLIN}}_{\mathcal{E}_{2,h+1}}(\lambda)\right)$$
$$+(2d\nu+16\nu+2d+8)/q$$

以上により，Theorem2 が成り立つ． □

次に，上記証明で利用した Lemma を下記に示す．3.7 節で示した Lemma と同様に，高島ら [65] が示した Lemma の証明を修正することで証明することができる．

Lemma 9. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$|\mathsf{Adv}^{(0)}_{\mathcal{A}}(\lambda)-\mathsf{Adv}^{(1)}_{\mathcal{A}}(\lambda)| \leq \mathsf{Adv}^{\mathsf{P1}}_{\mathcal{B}_1}(\lambda)+(d+1)/q$ となる計算量仮定 *Problem1* に対する確率的多項式時間攻撃者 $\mathcal{B}_1$ が存在する．

Proof. 攻撃者 $\mathcal{A}$ を利用して，Problem1 を解く確率的アルゴリズム $\mathcal{B}_1$ を下記のように構成する．

1. $\mathcal{B}_1$ は Problem1 インスタンス $(\mathbf{param}_{\overrightarrow{n}}, \mathbb{B}_0, \hat{\mathbb{B}^*_0}, \boldsymbol{e}_{\beta,0}, \{\mathbb{B}_t, \hat{\mathbb{B}^*_t}, \boldsymbol{e}_{\beta,t,1}, \boldsymbol{e}_{t,j}\}_{t=1,\ldots,d;j=2,\ldots,n_t})$ を受け取る．

2. $\mathcal{B}_1$ は，下記の様に $\hat{\mathbb{D}}_t, \hat{\mathbb{D}}^*_t$ が定義されたとみなす．

$$\begin{aligned}
&\mathbb{D}_0 := \mathbb{B}_0, \mathbb{D}^*_0 := \mathbb{B}^*_0, \hat{\mathbb{D}}_0 := (\boldsymbol{b}_{0,1}, \boldsymbol{b}_{0,3}, \boldsymbol{b}_{0,5}), \hat{\mathbb{D}}^*_0,\\
&\mathbb{D}_t := (\boldsymbol{d}_{t,j})_{j=1,\ldots,3n_t+1} := (\boldsymbol{b}_{t,2},\ldots,\boldsymbol{b}_{t,n_t}, \boldsymbol{b}_{t,1}, \boldsymbol{b}_{t,n_t+1},\ldots,\boldsymbol{b}_{t,3n_t+1}),\\
&\mathbb{D}^*_t := (\boldsymbol{d}^*_{t,j})_{j=1,\ldots,3n_t+1} := (\boldsymbol{b}^*_{t,2},\ldots,\boldsymbol{b}^*_{t,n_t}, \boldsymbol{b}^*_{t,1}, \boldsymbol{b}^*_{t,n_t+1},\ldots,\boldsymbol{b}^*_{t,3n_t+1}),\\
&\hat{\mathbb{D}}_t := (\boldsymbol{d}_{t,1},\ldots,\boldsymbol{d}_{t,n_t}, \boldsymbol{d}_{t,3n_t+1}), \hat{\mathbb{D}}^*_t := (\boldsymbol{d}^*_{t,1},\ldots,\boldsymbol{d}^*_{t,n_t}, \boldsymbol{d}_{t,2n_t+1},\ldots,\boldsymbol{d}_{t,3n_t+1}),
\end{aligned}$$

そして，Problem1 から $\hat{\mathbb{D}}_t, \hat{\mathbb{D}}^*_t$ を計算し，公開パラメータ $\mathsf{mpk} := (1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\hat{\mathbb{D}}_t\}_{t=0,\ldots,d})$ を計算し，攻撃者 $\mathcal{A}$ に与える．

3. 攻撃者 $\mathcal{A}$ から属性集合 Γ に対する鍵生成クエリを受け取ったら，Problem1 インスタンスから生成した基底 $\{\hat{\mathbb{D}}^*_t\}_{t=0,\ldots,d}$ を用いて Game0 に記載の手順でプロキシー鍵 $\mathsf{pxk}_{\Gamma_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\Gamma_\iota}$ を生成する．CorruptProxyKey ではプロキシー鍵 $\mathsf{pxk}_{\Gamma_\iota}$ を，CorruptSecretKey クエリではユーザ秘密鍵 $\mathsf{sk}_{\Gamma_\iota}$ を攻撃者 $\mathcal{A}$ に与える．

4. 攻撃者 $\mathcal{A}$ からチャレンジメッセージ $(m^{(0)}, m^{(1)})$，アクセス構造 $\mathbb{S} := (M, \rho)$ を受け取ったら，下記の様にチャレンジ暗号文を生成し，攻撃者 $\mathcal{A}$ に送付する．

$$\begin{aligned}
&b \xleftarrow{\mathsf{U}} \{0,1\}, \overrightarrow{f} \xleftarrow{\mathsf{U}} \mathbb{F}_q^r, \overrightarrow{s}^T := (s_1,\ldots,s_\ell)^T := M\cdot\overrightarrow{f}, s_0 := \overrightarrow{1}\cdot\overrightarrow{f}, \theta_i, \zeta \xleftarrow{\mathsf{U}} \mathbb{F}_q,\\
&\text{if } \rho(i) = (t, \overrightarrow{v}_i), \overrightarrow{c}_i := s_i\overrightarrow{e}_{t,1} + \theta_i\overrightarrow{v}_i,\\
&\text{if } \rho(i) = \neg(t, \overrightarrow{v}_i), \overrightarrow{c}_i := s_i\overrightarrow{v}_i,\\
&\boldsymbol{c}_0 := -s_0\boldsymbol{e}_{\beta,0} + \zeta\boldsymbol{b}_{0,3}, \boldsymbol{c}_i := \textstyle\sum_{j=1}^{n_t-1} c_{i,j}\boldsymbol{e}_{t,j+1} + c_{i,n_t}\boldsymbol{e}_{\beta,t,1}, c_{d+1} := g_T^{\zeta} m^{(b)}
\end{aligned}$$

5. 攻撃者 $\mathcal{A}$ からクエリーを受け取ったら，ステップ (3) と同様に応答する．

6. 攻撃者 $\mathcal{A}$ がビット b' を出力したら，$\mathcal{B}_1$ は $b = b'$ の場合は 1 を，そうでなければ 0 を出力する．

上記シミュレーションで生成される暗号文について確認していく．
まず $\beta = 0$ の時，暗号文は下記の様に計算される．

$$
\begin{aligned}
\boldsymbol{c}_0 &:= -s_0 \boldsymbol{e}_{\beta,0} + \zeta \boldsymbol{b}_{0,3} \\
&= (-\omega s_0, 0, 0, 0, -s_0\gamma_0)_{\mathbb{B}_0} + (0, 0, \zeta, 0, 0)_{\mathbb{B}_0} \\
&= (-\omega s_0, 0, \zeta, 0, -s_0\gamma_0)_{\mathbb{B}_0} \\
\boldsymbol{c}_i &:= \sum_{j=1}^{n_t-1} c_{i,j} \boldsymbol{e}_{t,j+1} + c_{i,n_t} \boldsymbol{e}_{\beta,t,1} \\
&= \sum_{j=1}^{n_t-1} c_{i,j} (\omega \overrightarrow{e}_{t,j+1}, 0^{n_t}, 0^{n_t}, 0)_{\mathbb{B}_t} + c_{i,n_t} (\omega \overrightarrow{e}_{t,1}, 0^{n_t}, 0^{n_t}, \gamma_t)_{\mathbb{B}_t} \\
&= (\omega \overrightarrow{c}_i, 0^{n_t}, 0^{n_t}, c_{i,n_t}\gamma_t)_{\hat{\mathbb{D}}_t} \\
&= \begin{cases} (\omega(s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i), 0^{n_t}, 0^{n_t}, c_{i,n_t}\gamma_t)_{\hat{\mathbb{D}}_t} & \text{if } \rho(i) = (t, \overrightarrow{v}_i) \\ (\omega s_i \overrightarrow{v}_i, 0^{n_t}, 0^{n_t}, c_{i,n_t}\gamma_t)_{\hat{\mathbb{D}}_t} & \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i) \end{cases} \\
c_{d+1} &:= g_T^{\zeta} m^{(b)}
\end{aligned}
$$

まず，暗号文の形式としては，全ての s_i が ω 倍されているものの，Game0 にて定義した通りの暗号文になっている．その分布については，$\gamma_0, \gamma_t,$ は Problem1 にて独立一様分布に選ばれており，ζ も独立に選ばれている．そのため，Game0 の暗号文を正しくシミュレートしている．

次に $\beta = 1$ の時の暗号文について，下記の様に計算される．

$$
\begin{aligned}
\boldsymbol{c}_0 &:= -s_0 \boldsymbol{e}_{\beta,0} + \zeta \boldsymbol{b}_{0,3} \\
&= (-\omega s_0, -s_0 z_0, 0, 0, -s_0\gamma_0)_{\mathbb{B}_0} + (0, 0, \zeta, 0, 0)_{\mathbb{B}_0} \\
&= (-\omega s_0, -s_0 z_0, \zeta, 0, -s_0\gamma_0)_{\mathbb{B}_0} \\
\boldsymbol{c}_i &:= \sum_{j=1}^{n_t-1} c_{i,j} \boldsymbol{e}_{t,j+1} + c_{i,n_t} \boldsymbol{e}_{\beta,t,1} \\
&= \sum_{j=1}^{n_t-1} c_{i,j} (\omega \overrightarrow{e}_{t,j+1}, 0^{n_t}, 0^{n_t}, 0)_{\mathbb{B}_t} + c_{i,n_t} (\omega \overrightarrow{e}_{t,1}, \overrightarrow{z}_t, 0^{n_t}, \gamma_t)_{\mathbb{B}_t} \\
&= (\omega \overrightarrow{c}_i, c_{i,n_t} \overrightarrow{z}_t, 0^{n_t}, c_{i,n_t}\gamma_t)_{\hat{\mathbb{D}}_t} \\
&= \begin{cases} (\omega(s_i \overrightarrow{e}_{t,1} + \theta_i \overrightarrow{v}_i), c_{i,n_t} \overrightarrow{z}_t, 0^{n_t}, c_{i,n_t}\gamma_t)_{\hat{\mathbb{D}}_t} & \text{if } \rho(i) = (t, \overrightarrow{v}_i) \\ (\omega s_i \overrightarrow{v}_i, 0^{n_t}, c_{i,n_t} \overrightarrow{z}_t, c_{i,n_t}\gamma_t)_{\hat{\mathbb{D}}_t} & \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i) \end{cases} \\
c_{d+1} &:= g_T^{\zeta} m^{(b)}
\end{aligned}
$$

まず，暗号文の形式としては，Game1 にて定義した通りの semi-functional な暗号文になっている．その分布については，$\gamma_0, \gamma_t, z_0, \overrightarrow{z}_t$ は Problem1 にて独立一様分布に選ばれており，ζ も独立に選ばれている．ただし，$s_0 = 0$ もしくは $c_{i,n_t} = 0$ となる場合は，正しく Game1 がシミュレートできない．そのため，確率 $(d+1)/q$ の例外を除いて，Game1 の暗号文を正しくシミュレートしている．

よって，2 つのゲームを識別することは Problem1 インスタンスを識別することと同等である． □

Lemma 10. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$|\mathsf{Adv}_{\mathcal{A}}^{(2-h)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda)| \leq \mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}_{2,h}^+}(\lambda) + (d+3)/q$ となる計算量仮定 *Problem2* に対する確率的多項式時間攻撃者 $\mathcal{B}_2^+$ が存在する．

Proof. 攻撃者 $\mathcal{A}$ を利用して，Problem2 を解く確率的アルゴリズム $\mathcal{B}_2^+$ を下記のように構成する．

1. $\mathcal{B}_2^+$ は Problem2 インスタンス $(\mathbf{param}_{\overrightarrow{n}}, \hat{\mathbb{B}}_0, \mathbb{B}_0^*, \boldsymbol{h}_{\beta,0}^*, \boldsymbol{e}_0, \{\hat{\mathbb{B}}_t, \mathbb{B}_t^*, \boldsymbol{h}_{\beta,t,j}, \boldsymbol{e}_{t,j}\}_{t=1,\ldots,d;j=2,\ldots,n_t})$ を受け取る．

2. $\mathcal{B}_2^+$ は，Problem2 インスタンスから公開パラメータ $\mathsf{mpk} := (1^\lambda, \mathsf{param}_{\overrightarrow{n}}, \{\hat{\mathbb{B}'}_t\}_{t=0,\ldots,d})$ を計算し，攻撃者 $\mathcal{A}$ に与える．ここで，$\hat{\mathbb{B}'}_0 := (\boldsymbol{b}_{0,1}, \boldsymbol{b}_{0,3}, \boldsymbol{b}_{0,5})$, $\hat{\mathbb{B}'}_t := (\boldsymbol{b}_{t,1}, \ldots, \boldsymbol{b}_{t,n_t}, \boldsymbol{b}_{t,3n_t+1})$ である．

3. 攻撃者 $\mathcal{A}$ から属性集合 $\Gamma = \{(t, \overrightarrow{x}_t)\}$ に対する ι 番目の鍵生成クエリを受け取ったら，下記のようにプロキシー鍵 $\mathsf{pxk}_{\Gamma_\iota}$ とユーザ秘密鍵 $\mathsf{sk}_{\Gamma_\iota}$ を生成する．

   (a) $1 \leq \iota \leq h$ の場合，Problem2 インスタンスに含まれる $\{\mathbb{B}_t^*\}_{t=0,\ldots,d}$ を用いて，Game2-$(h+1)$ に示した手順で生成する．

   (b) $\iota = h+1$ の場合，Problem2 インスタンスを用いて下記の様に生成する．

$$\begin{aligned}
&\boldsymbol{k}_0^* := \boldsymbol{h}_{\beta,0}^* + \boldsymbol{b}_{0,3}^*, \\
&\boldsymbol{k}_t^* := \textstyle\sum_{j=1}^{n_t} x_{t,j} \boldsymbol{h}_{\beta,t,j} \text{ for } (t, \overrightarrow{x}_t) \in \Gamma
\end{aligned}$$

   (c) $\iota \geq h+2$ の場合，Problem2 インスタンスの $\{\mathbb{B}_t^*\}_{t=0,\ldots,d}$ を用いて，Game1 に示した手順で生成する．

   CorruptProxyKey ではプロキシー鍵 $\mathsf{pxk}_{\Gamma_\iota}$ を，CorruptSecretKey クエリではユーザ秘密鍵 $\mathsf{sk}_{\Gamma_\iota}$ を攻撃者 $\mathcal{A}$ に与える．

4. 攻撃者 $\mathcal{A}$ からチャレンジメッセージ $(m^{(0)}, m^{(1)})$，アクセス構造 $\mathbb{S} := (M, \rho)$ を受け取ったら，下記の様にチャレンジ暗号文を生成し，攻撃者 $\mathcal{A}$ に送付する．

$$\begin{aligned}
&\pi'_t, \mu_t, g'_k, \tilde{\mu}_k \xleftarrow{\mathsf{U}} \mathbb{F}_q \text{ for } t = 1, \ldots, d; k = 1, \ldots, r, \\
&\tilde{\boldsymbol{f}}_0 := \textstyle\sum_{k=1}^{r} \left(g'_k \boldsymbol{e}_0 + \tilde{\mu}_k \boldsymbol{b}_{0,1}\right), \\
&\text{for } t = 1, \ldots, d; k = 1, \ldots, r; j = 1, \ldots, n_t; \\
&\quad \boldsymbol{f}_{t,j} := \pi'_t \boldsymbol{e}_{t,j} + \mu_t \boldsymbol{b}_{t,j}, \\
&\quad \tilde{\boldsymbol{f}}_{t,k,j} := g'_k \boldsymbol{e}_{t,j}^* + \tilde{\mu}_k \boldsymbol{b}_{t,j}^*, \\
&\zeta \xleftarrow{\mathsf{U}} \mathbb{F}_q, \boldsymbol{q}_0 \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}_{0,5} \rangle, \boldsymbol{q}_i \xleftarrow{\mathsf{U}} \mathsf{span}\langle \boldsymbol{b}_{i,3n_t+1} \rangle, \\
&\boldsymbol{c}_0 := -\tilde{\boldsymbol{f}}_0 + \zeta \boldsymbol{b}_{0,3}^* + \boldsymbol{q}_0, \\
&\text{for } i = 1, \ldots, \ell; \\
&\quad \text{if } \rho(i) = (t, \overrightarrow{v}_i), \\
&\qquad \boldsymbol{c}_i := \textstyle\sum_{j=1}^{n_t} v_{i,j} \boldsymbol{f}_{t,j} + \sum_{k=1}^{r} M_{i,k} \tilde{\boldsymbol{f}}_{t,k,1} + \boldsymbol{q}_i, \\
&\quad \text{if } \rho(i) = \neg(t, \overrightarrow{v}_i), \\
&\qquad \boldsymbol{c}_i := \textstyle\sum_{j=1}^{n_t} v_{i,j} \left(\sum_{k=1}^{r} M_{i,k} \tilde{\boldsymbol{f}}_{t,k,j}\right) + \boldsymbol{q}_i, \\
&c_{d+1} := g_T^{\zeta} m^{(b)}
\end{aligned}$$

   ただし，$(M_{i,k})_{i=1,\ldots,\ell;k=1,\ldots,r} := M$ である．

5. 攻撃者 $\mathcal{A}$ からクエリーを受け取ったら，ステップ (3) と同様に応答する．

6. 攻撃者 $\mathcal{A}$ がビット b' を出力したら，$\mathcal{B}_2^+$ は $b = b'$ の場合は 1 を，そうでなければ 0 を出力する．

なお，ステップ (4) の $\tilde{\boldsymbol{f}}_0, \boldsymbol{f}_{t,j}, \tilde{\boldsymbol{f}}_{t,k,j}$ は，$\pi_t := \tau\pi'_t, \theta_t := \pi_t\omega + \mu_t, g_k := \tau g'_k, f_k := g_k\omega + \tilde{\mu}_k, s_0 := \sum_{k=1}^{r} f_k, a_0 := \sum_{k=1}^{r} g_k, \omega_0 := a_0 z_0 = a_0/u_0$ としたとき，下記の様に計算できる．

$$
\begin{aligned}
&\tilde{\boldsymbol{f}}_0 = (s_0, \omega_0, 0, 0, 0)_{\mathbb{B}_0},\\
&\boldsymbol{f}_{t,j} := (\theta_t \overrightarrow{e}_{t,j}, \pi_t \overrightarrow{z}_{t,j}, 0^{n_t}, 0)_{\mathbb{B}_t},\\
&\tilde{\boldsymbol{f}}_{t,k,j} := (f_k \overrightarrow{e}_{t,j}, g_k \overrightarrow{z}_{t,j}, 0^{n_t}, 0)_{\mathbb{B}_t}
\end{aligned}
$$

ここで，$\tau, \omega, u_0, \overrightarrow{e}_{t,j}, \overrightarrow{z}_{t,j}$ は Problem2 にて定義された値であり，$\tau, \omega, \overrightarrow{z}_{t,j}$ は独立して一様分布する．さらに，$\pi'_t, \mu_t, g'_k, \tilde{\mu}_k$ は $\mathbb{F}_q$ から独立して一様分布に選んでおり，$\{\theta_t, \pi_t\}_{t=1,\ldots,d}$, $\{f_k, g_k\}_{k=1,\ldots,r}$ もそれぞれ独立に一様分布となる．

始めに，暗号文について確認する．Problem2 の要素を用いて，暗号文は下記のように構成されている．

$$
\begin{aligned}
\boldsymbol{c}_0 &:= -\tilde{\boldsymbol{f}}_0 + \zeta \boldsymbol{b}_{0,3} + \boldsymbol{q}_0\\
&= -(s_0, w_0, 0, 0, 0)_{\mathbb{B}_0} + (0, 0, \zeta, 0, 0)_{\mathbb{B}_0} + (0, 0, 0, 0, \eta_0)_{\mathbb{B}_0}\\
&= (-s_0, -w_0, \zeta, 0, \eta_0)_{\mathbb{B}_0}
\end{aligned}
$$

if $\rho(i) = (t, \overrightarrow{v}_i)$:

$$
\begin{aligned}
\boldsymbol{c}_i &:= \sum_{j=1}^{n_t} v_{i,j} \boldsymbol{f}_{t,j} + \sum_{k=1}^{r} M_{i,k} \tilde{\boldsymbol{f}}_{t,k,1} + \boldsymbol{q}_i\\
&= \sum_{j=1}^{n_t} v_{i,j} (\theta_t \overrightarrow{e}_{t,j}, \pi_t \overrightarrow{z}_{t,j}, 0^{n_t}, 0)_{\mathbb{B}_t} + \sum_{k=1}^{r} M_{i,k} (f_k \overrightarrow{e}_{t,1}, g_k \overrightarrow{z}_{t,1}, 0^{n_t}, 0)_{\mathbb{B}_t}\\
&\quad + (0^{n_t}, 0^{n_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}\\
&= (\theta_t \overrightarrow{v}_i, \pi_t \overrightarrow{v}_i \cdot Z_t, 0^{n_t}, 0)_{\mathbb{B}_t} + (s_i \overrightarrow{e}_{t,1}, a_i \overrightarrow{e}_{t,1} \cdot Z_t, 0^{n_t}, 0)_{\mathbb{B}_t}\\
&\quad + (0^{n_t}, 0^{n_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}\\
&= (s_i \overrightarrow{e}_{t,1} + \theta_t \overrightarrow{v}_i, (a_i \overrightarrow{e}_{t,1} + \pi_t \overrightarrow{v}_i) \cdot Z_t, 0^{n_t}, \eta_i)_{\mathbb{B}_t}
\end{aligned}
$$

if $\rho(i) = \neg(t, \overrightarrow{v}_i)$:

$$
\begin{aligned}
\boldsymbol{c}_i &:= \sum_{j=1}^{n_t} v_{i,j} \left(\sum_{k=1}^{r} M_{i,k} \tilde{\boldsymbol{f}}_{t,k,j} \right) + \boldsymbol{q}_i\\
&= \sum_{j=1}^{n_t} v_{i,j} \left(\sum_{k=1}^{r} M_{i,k} (f_k \overrightarrow{e}_{t,j}, g_k \overrightarrow{z}_{t,j}, 0^{n_t}, 0)_{\mathbb{B}_t} \right) + (0^{n_t}, 0^{n_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}\\
&= \sum_{j=1}^{n_t} v_{i,j} (s_i \overrightarrow{e}_{t,j}, a_i \overrightarrow{z}_{t,j}, 0^{n_t}, 0)_{\mathbb{B}_t} + (0^{n_t}, 0^{n_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}\\
&= (s_i \overrightarrow{v}_i, a_i \overrightarrow{v}_i \cdot Z_t, 0^{n_t}, 0)_{\mathbb{B}_t} + (0^{n_t}, 0^{n_t}, 0^{n_t}, \eta_i)_{\mathbb{B}_t}\\
&= (s_i \overrightarrow{v}_i, a_i \overrightarrow{v}_i \cdot Z_t, 0^{n_t}, \eta_i)_{\mathbb{B}_t}\\
c_{d+1} &:= g_T^{\zeta} m^{(b)}
\end{aligned}
$$

暗号文の構成としては，Game1 で定義された形式と同一である．次に分布について確認するが，ζ, η_0, η_t は独立一様分布に選んでおり，Z_t も各 t 毎に $GL(n_t, \mathbb{F}_q)$ から一様分布で選んでいる．また，f_k, g_k が独立一様分布であることから，各 s_i, a_i も独立一様分布となる．また，$w_0 = a_0/u_0 = a_0 z_0$ であり，u_0 が一様分布するのであれば，a_0 とは独立に w_0 は独立一様分布となる．そのため，上記暗号文は $w_0 = 0, (a_i \overrightarrow{e}_{t,1} + \pi_t \overrightarrow{v}_i) \cdot Z_t = \overrightarrow{0}, a_i \overrightarrow{v}_i \cdot Z_t = \overrightarrow{0}$ の例外を除いて，Game1 にて規定した semi-functional な暗号文と分布が同一となる．ただし，$z_0 = u_0^{-1}$ であるため，別途プロキシー鍵側との依存性の検証が必要となる．

$\beta = 0$ の場合，プロキシー鍵は下記のようになる．

$$\begin{aligned}
\boldsymbol{k}_0^* &= \left(\boldsymbol{h}_{\beta,0}^* + \boldsymbol{b}_{0,3}^*\right) \\
&= (\delta, 0, 0, \delta_0, 0)_{\mathbb{B}_0^*} + (0, 0, 1, 0, 0)_{\mathbb{B}_0^*} \\
&= (\delta, 0, 1, \delta_0, 0)_{\mathbb{B}_0^*} \\
\boldsymbol{k}_t^* &:= \sum_{j=1}^{n_t} x_{t,j} \boldsymbol{h}_{\beta,t,j} \\
&= \sum_{j=1}^{n_t} x_{t,j} (\delta \overrightarrow{e}_{t,j}, 0^{n_t}, \delta_{t,j}, 0)_{\mathbb{B}_t^*} \\
&= (\delta \overrightarrow{x}_t, 0^{n_t}, \overrightarrow{\varphi_t}, 0)_{\mathbb{B}_t^*}
\end{aligned}$$

ここで，$\overrightarrow{\varphi_t} = \sum_{j=1}^{n_t} x_{t,j} \delta_{t,j}$ である．なお，プロキシー鍵の形としては，Game 1 の秘密鍵と同一である．次に分布について考察する．$\delta, \delta_0, \delta_{t,j}$ は Problem2 にて独立一様分布に選ばれている乱数のため，$\overrightarrow{\varphi_t}$ は $\mathbb{F}_q^{n_t}$ 上を独立して一様分布する．そのため，$\delta = 0$ の場合（確率 $1/q$），および暗号文が semi-functional にならない場合（確率 $(d+1)/q$）を除いて，正しく暗号文，プロキシー鍵，ユーザ秘密鍵をシミュレートしている．

$\beta = 1$ の場合，プロキシー鍵は下記のようになる．

$$\begin{aligned}
\boldsymbol{k}_0^* &= \boldsymbol{h}_{\beta,0}^* + \boldsymbol{b}_{0,3}^* \\
&= (\delta, u_0, 0, \delta_0, 0)_{\mathbb{B}_0^*} + (0, 0, 1, 0, 0)_{\mathbb{B}_0^*} \\
&= (\delta, u_0, 1, \delta_0, 0)_{\mathbb{B}_0^*} \\
\boldsymbol{k}_t^* &:= \sum_{j=1}^{n_t} x_{t,j} \boldsymbol{h}_{\beta,t,j} \\
&= \sum_{j=1}^{n_t} x_{t,j} (\delta \overrightarrow{e}_{t,j}, \overrightarrow{u}_{t,j}, \delta_{t,j}, 0)_{\mathbb{B}_t^*} \\
&= (\delta \overrightarrow{x}_t, \overrightarrow{x}_t U_t, \overrightarrow{\varphi_t}, 0)_{\mathbb{B}_t^*}
\end{aligned}$$

ここで，$\beta = 0$ のケースとの違いは，各 $\boldsymbol{k}_i^*$ の第 2 成分であり，それ以外の要素については $\beta = 0$ のケースと同じであることから正しくシミュレートできている．高島らが示した元の方式では，暗号文の要素 $\boldsymbol{c}_0$ の第 2 要素 w_0 と，秘密鍵の要素 $\boldsymbol{k}_0^*$ の第 2 要素 r_0 に $w_0 := a_0/r_0$ という関係性があった．そこで a_0 の独立性を確認するため高島らは，$a_0 = \overrightarrow{1} \cdot \overrightarrow{g}^T$，$(a_1, \ldots, a_\ell)^T = M \cdot \overrightarrow{g}^T$，$U_t = (Z_t^{-1})^T$ という関係があることから，$\overrightarrow{w}_i := (\pi_t \overrightarrow{v}_i + a_i \overrightarrow{e}_{t,1}) \cdot Z_t$，$\overrightarrow{w}_i := a_i \overrightarrow{v}_i \cdot Z_t$，$\overrightarrow{r}_t := \overrightarrow{x}_t \cdot U_t$ の独立性の検証が必要となると指摘していた．そこで下記の 5 パターンについて確認する．

1. $\gamma(i)=1$ で $[\rho(i)=(t,\overrightarrow{v}_i)\wedge(t,\overrightarrow{x}_t)\in\Gamma\wedge\overrightarrow{v}_i\cdot\overrightarrow{x}_t=0]$
   $\overrightarrow{w}_i\cdot\overrightarrow{r}_t=(\pi_t\overrightarrow{v}_i+a_i\overrightarrow{e}_{t,1})\cdot Z_t\cdot\overrightarrow{x}_t\cdot U_t=a_i$ であることから，$(\overrightarrow{w}_i,\overrightarrow{r}_t)$ は C_{a_i} 上を一様分布する．

2. $\gamma(i)=1$ で $[\rho(i)=\neg(t,\overrightarrow{v}_i)\wedge(t,\overrightarrow{x}_t)\in\Gamma\wedge\overrightarrow{v}_i\cdot\overrightarrow{x}_t\neq 0]$
   $\overrightarrow{w}_i\cdot\overrightarrow{r}_t=a_i(\overrightarrow{v}_i\cdot\overrightarrow{x}_t)$ であることから，$(\overrightarrow{w}_i,\overrightarrow{r}_t)$ は $C_{a_i(\overrightarrow{v}_i\cdot\overrightarrow{x}_t)}$ 上を一様分布する．

3. $\gamma(i)=0$ で $[\rho(i)=(t,\overrightarrow{v}_i)\wedge(t,\overrightarrow{x}_t)\in\Gamma]$
   $(\overrightarrow{w}_i,\overrightarrow{r}_t)$ は $C_{(\overrightarrow{v}_i\cdot\overrightarrow{x}_t)\pi_i+a_i}$ 上を一様分布する．ここで，π_i は $\mathbb{F}_q$ 上を独立一様分布する変数であるため，$(\overrightarrow{w}_i,\overrightarrow{r}_t)$ は $\mathbb{F}_q^{n_t}$ 上を独立一様分布する．

4. $\gamma(i)=0$ で $[\rho(i)=\neg(t,\overrightarrow{v}_i)\wedge(t,\overrightarrow{x}_t)\in\Gamma]$
   $(\overrightarrow{w}_i,\overrightarrow{r}_t)$ は C_0 上を一様分布する．

5. $[\rho(i)=(t,\overrightarrow{v}_i)\wedge(t,\overrightarrow{x}_t)\notin\Gamma]$ もしくは $[\rho(i)=\neg(t,\overrightarrow{v}_i)\wedge(t,\overrightarrow{x}_t)\notin\Gamma]$
   $\overrightarrow{v}_i$ に対応する属性ベクトル $\overrightarrow{x}_t$ が存在しないので，$\overrightarrow{v}_i$ は $\mathbb{F}_q^{n_t}$ 上を独立一様分布する．

上記のうち，ケース 3〜5 の場合は高島らが示したように a_0 とは独立に分布することが分かる．チャレンジ暗号文を復号できない属性を指定してプロキシー鍵とユーザ秘密鍵を入手した場合，$\overrightarrow{1}\notin\mathsf{span}\langle(M_i)_{\gamma(i)=1}\rangle$ であることから，$\overrightarrow{g}$ が一様分布に選ばれたという条件の下で，$a_0:=\overrightarrow{1}\cdot\overrightarrow{g}$ と $\{a_i:=M_i\cdot\overrightarrow{g}^T|\gamma(i)=1\}$ は独立一様分布となる．そのため，$\overrightarrow{w}_i,\overrightarrow{w}_i,\overrightarrow{r}_t$ は独立一様分布となり，Problem2 の δ が 0 になる例外を除いて，チャレンジ暗号文，プロキシー鍵，ユーザ秘密鍵は正しくシミュレートできる．

一方，チャレンジ暗号文が復号できる属性が指定された場合，上記のケース (1)，(2) について考慮してプロキシー鍵とユーザ秘密鍵の分布を評価する必要がある．自明な攻撃を防ぐために，攻撃者はプロキシー鍵かユーザ秘密鍵のいずれか一方のみが取得できるように制約しているので，それぞれのケースについて考察する．

[プロキシー鍵のみを取得する場合]
攻撃者がプロキシー鍵のみ取得でき，ユーザ秘密鍵が取得できないケースについて考察する．プロキシー鍵には $\overrightarrow{r}_t$ が含まれるため，その分布について前述のケース (1) と (2) についての評価が必要となる．ケース (1) については，$\overrightarrow{w}_i\cdot\overrightarrow{r}_t=(\pi_t\overrightarrow{v}_i+a_i\overrightarrow{e}_{t,1})\cdot Z_t\cdot\overrightarrow{x}_t\cdot U_t=a_i$ であることから，$(\overrightarrow{w}_i,\overrightarrow{r}_t)$ は C_{a_i} 上を独立一様分布する．同様にケース (2) についても，$\overrightarrow{w}_i\cdot\overrightarrow{r}_t=a_i(\overrightarrow{v}_i\cdot\overrightarrow{x}_t)$ であることから，$(\overrightarrow{w}_i,\overrightarrow{r}_t)$ は $C_{a_i(\overrightarrow{v}_i\cdot\overrightarrow{x}_t)}$ 上を一様分布する．これより，攻撃者はチャレンジ暗号文の $\overrightarrow{w}_i,\overrightarrow{w}_i$ だけから $(a_1,\ldots,a_\ell)$ に関する情報を得ることはできないが，プロキシー鍵の $\overrightarrow{r}_t$ と内積値を取ることで $(a_1,\ldots,a_\ell)$ に関する情報が得られる．しかし，$(a_1,\ldots,a_\ell)$ に関する情報が得られても，攻撃者はチャレンジ暗号文とプロキシー鍵の correlation を生み出すユーザ秘密鍵の要素 r_0 は入手できず，これを入手できない攻撃者にとっては，チャレンジ暗号文とプロキシー鍵は独立一様分布に見える．

[ユーザ秘密鍵のみを取得する場合]
攻撃者がユーザ秘密鍵のみ取得でき，プロキシー鍵が取得できないケースについて考察する．この場合，$\boldsymbol{k}_0^*$ に含まれる r_0 を得ることができるが，$r_0=u_0$ は Problem2 にて独立一様分布に選ばれた乱数であること，およびプロキシー鍵に含まれる $(a_1,\ldots,a_\ell)$ が得られない

ことから a_0 に関する追加の情報が得られず，これを入手できない攻撃者にとっては，ユーザ秘密鍵の r_0 とチャレンジ暗号文の $w_0 := a_0/r_0$ は独立に一様分布に見える．

以上の考察より，たとえ暗号化データを復号できる条件の下であっても，プロキシー鍵もしくはユーザ秘密鍵のいずれか一方しか入手できない攻撃者にとっては，Problem2 の δ が 0 になる例外を除いて，正しくプロキシー鍵やユーザ秘密鍵をシミュレートできていることが分かる．よって，2 つのゲームを識別することは Problem2 インスタンスを識別することと同等である． □

Lemma 11. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$|\mathsf{Adv}_{\mathcal{A}}^{(2-h^+)}(\lambda) - \mathsf{Adv}_{\mathcal{A}}^{(2-(h+1))}(\lambda)| \leq \mathsf{Adv}^{\mathsf{P2}}_{\mathcal{B}_2,h+1}(\lambda) + (d+3)/q$ となる計算量仮定 *Problem2* に対する確率的多項式時間攻撃者 $\mathcal{B}_2$ が存在する．

Proof. 攻撃者 $\mathcal{A}$ を利用して，Problem2 を解く確率的アルゴリズム $\mathcal{B}_2$ を構成する．アルゴリズム $\mathcal{B}_2$ の動作はアルゴリズム $\mathcal{B}_2^+$ と以下の点を除いて同一である．

1. ステップ (3) のケース (b) において，$\boldsymbol{k}_0^*$ の計算方法が下記のようになる．
   $r_0' \xleftarrow{\mathsf{U}} \mathbb{F}_q, \boldsymbol{k}_0^* := \boldsymbol{h}_{\beta,0}^* + r_0' \boldsymbol{b}_{0,2}^* + \boldsymbol{b}_{0,3}^*$
2. 最後のステップで，攻撃者 $\mathcal{A}$ がビット b' を出力したら，$\mathcal{B}_2$ は $b = b'$ の場合は 0 を，そうでなければ 1 を出力する．

分布に関しては，Lemma4 と同様に確認することができる．

$\beta = 0$ の場合は，$\delta = 0$ の場合，すなわち確率 $1/q$ を除いて，Game 2-(h+1) と分布が同一であることがわかる．

$\beta = 1$ の場合は，$\delta = 0$ の場合，および $\overrightarrow{r}_t = \overrightarrow{0}$ となる場合を除いて，Game 2-h^+ と分布が同一であることがわかる．その例外が生じる確率は $(d+2)/q$ となる．

よって，2 つのゲームを識別することは Problem2 インスタンスを識別することと同等である． □

Lemma 12. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$\mathsf{Adv}_{\mathcal{A}}^{(3)}(\lambda) \leq \mathsf{Adv}_{\mathcal{A}}^{(2-\nu)}(\lambda) + 1/q$ である．

Proof. 新しい基底 $\mathbb{D}_0$ および $\mathbb{D}_0^*$ を定義する．

$$
\begin{aligned}
&\theta \xleftarrow{\mathsf{U}} \mathbb{F}_q, \boldsymbol{d}_{0,2} := \boldsymbol{b}_{0,2} - \theta \boldsymbol{b}_{0,3}, \boldsymbol{d}_{0,3}^* := \boldsymbol{b}_{0,3}^* + \theta \boldsymbol{b}_{0,2}^*,\\
&\mathbb{D}_0 := (\boldsymbol{b}_{0,1}, \boldsymbol{d}_{0,2}, \boldsymbol{b}_{0,3}, \boldsymbol{b}_{0,4}, \boldsymbol{b}_{0,5}),\\
&\mathbb{D}_0^* := (\boldsymbol{b}_{0,1}^*, \boldsymbol{b}_{0,2}^*, \boldsymbol{d}_{0,3}^*, \boldsymbol{b}_{0,4}^*, \boldsymbol{b}_{0,5}^*),
\end{aligned}
$$

この時，j 番目に生成したプロキシー鍵 $\mathsf{pxk}_\Gamma^{(j)}$ の要素 $\boldsymbol{k}_0^{(j)*}$ は下記の様に表せる．

$$
\begin{aligned}
\boldsymbol{k}_0^{(j)*} &= (\delta^{(j)}, r_0^{(j)}, 1, \varphi_0^{(j)}, 0)_{\mathbb{B}_0^*}\\
&= (\delta^{(j)}, r_0^{(j)} + \theta, 1, \varphi_0^{(j)}, 0)_{\mathbb{D}_0^*}\\
&= (\delta^{(j)}, \vartheta_0^{(j)}, 1, \varphi_0^{(j)}, 0)_{\mathbb{D}_0^*}\\
\boldsymbol{c}_0 &= (-s_0, w_0, \zeta, 0, \eta_0)_{\mathbb{B}_0}
\end{aligned}
$$

$$
\begin{aligned}
&= (-s_0, w_0, \zeta + r_0\theta, 0, \eta_0)_{\mathbb{D}_0} \\
&= (-s_0, w_0, \zeta', 0, \eta_0)_{\mathbb{D}_0}
\end{aligned}
$$

ただし，$\vartheta_0^{(j)} := r_0^{(j)} + \theta, \zeta' := \zeta + r_0\theta$ である．

攻撃者のビューからすれば，基底 $\mathbb{B}_0$ も基底 $\mathbb{D}_0$ も公開パラメータ $\mathsf{mpk} := (1^\lambda, \mathsf{param}_{\vec{n}}, \{\hat{\mathbb{B}}_t\}_{t=0,\ldots,d})$ と合致するため，どちらの基底で作られた暗号文でも区別はつかない．そのため，Game2$-\nu$ と Game3 は，乱数 $r_0 = 0$ の場合を除いて conceptual change である． □

Lemma 13. 任意の確率的多項式時間攻撃者 $\mathcal{A}$ に対して，$\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) = 0$ である．

Proof. ζ および ζ' が独立した値であることから，攻撃者はメッセージに関する情報は得られない．ゆえに，$\mathsf{Adv}^{(3)}_{\mathcal{A}}(\lambda) = 0$ が成り立つ． □

3.9 本章のまとめ

本章では，高島ら [65] が提案した関数型暗号において，復号鍵を失効させるための仕組みについて提案した．具体的には，プロキシー支援型のアプローチを採用し，プロキシーサーバとユーザが協調して復号処理を行うようにすることで，プロキシーサーバが管理するプロキシー鍵を削除するだけでユーザ秘密鍵を失効できる仕組みを実現した．これにより，暗号化する際は失効しているユーザが誰かを意識しなくてよいため要件 1 を満たし，さらに失効処理を行った時点で過去に復号できていた暗号化データをすぐに復号できないようにする失効が実現できるため要件 2 を満たす．また，クラウド側の処理が一切不要であることから要件 3 を満たし，失効していないユーザには全く影響が出ないことから要件 4 も満たす．また，関数型暗号の復号鍵を要素ごとに分解して 2 分割するというシンプルな手法で，プロキシー鍵とユーザ秘密鍵を生成するようにしたことで計算量の増加は全くない．さらに，鍵ポリシー型と暗号文ポリシー型の双方の関数型暗号に対して失効機能を付与し，かつ adaptive-secure で安全性証明が付けられることも示した．

第4章　暗号化タグの索引付による高速化を可能としたグループ共有型検索可能暗号

4.1　検索性能向上の要件と課題

4.1.1　要件

企業間情報共有に適した検索可能暗号データベースの実現のためには，これまでに述べたように公開鍵暗号に基づく検索可能暗号が適していると考えている．しかし，検索可能暗号に限らず通常の暗号においても公開鍵暗号は演算量が多く時間がかかるため，蓄積された大量のデータを処理しなければならない検索可能暗号データベースでは，この演算量の多さが大きな課題となっている．また，企業間でのデータ共有に適したマルチユーザ化を図ることや，データベースへの組み込みについても課題となる．

そこで我々は，公開鍵ベースの検索可能暗号の実用化に着目し，下記要件を満たすようなアルゴリズムレベルの改善を図ることとした．

要件 A: 実用的な検索速度
暗号化データの検索にかかる処理量のオーダーが，データの件数に比例するのではなく，それ以下に抑えられること

要件 B: 複数利用者でのデータ共有
ユーザ ID だけでなく，所属やプロジェクト名などのグループ ID でデータ共有ができること

また，クラウドサービスに適用するためには検索可能暗号自体の改良に加え，Web アプリケーションからの利用を容易にするために下記要件も満たす必要があると考えた．

要件 C: データベースへの検索可能暗号の組み込み
Web アプリケーションの改修量を削減するため，特殊な API を呼ばなくても SQL 文にて検索可能暗号化利用できること

なお，ユースケースによってはデータベースのテーブル名やカラム名も秘匿したいというケースが考えられるが，本研究ではテーブル構成そのものを秘匿することはスコープ外とし，中に保管されているデータを秘匿する一般的なユースケースを対象として考えた．一般的な SQL 文では，テーブル名や列名などを一意に識別する必要があるため，ここに検索可能暗号を適用するにはリレーショナルデータベース管理システム (RDBMS) の改修が必要と考えるが，データの秘匿だけであれば本研究で提案したように RDBMS の改修は行わずとも実現することが可能となる．

表 4.1: 既存の検索可能暗号と要件の充足性

	要件 A	要件 B	要件 C
Shi et al.			
Bellare et al.	✓		
Popa et al.	✓		✓
Hattori et al.		✓	
Suzuki et al.			✓
提案方式	✓	✓	✓

4.1.2 既存の公開鍵暗号型検索可能暗号の課題

一番最初に提案された公開鍵ベースの検索可能暗号は，Boneh ら [17] によって提案されたアルゴリズムである．本方式のメリットは，公開鍵を持つ利用者であれば誰でもキーワードの暗号化が可能で，公開鍵に対応する秘密鍵を持つユーザだけが暗号化データを検索できるという点である．これ以降に提案した方式も含め，保管したいデータから後で検索されるであろうキーワードを抽出し，キーワードを暗号化してタグを生成する．そして，暗号化データとともにタグを関連付けて保管する．ユーザがキーワード検索をしたい場合は，検索キーワードからトラップドアを生成し，これを検索要求としてサーバに送付する．

この方式の提案以降，AND 検索ができる方式，範囲検索ができる方式など，公開鍵ベース検索可能暗号で高度な検索機能を持った方式が提案されている [19][28][47][48][53]．いずれの方式も，暗号化データ数に比例した検索時間がかかるため低速であり，さらに複数利用者間でのデータ共有はできない．多くの方式は，Hidden Vector Encryption (HVE) と呼ばれる方式から構成できる．しかし，いずれも IND-CPA セキュアと呼ばれる高い安全性を持っているため，検索性能に課題を抱えている．これは，暗号化データから 1 ビットの情報も漏らさない高い安全性を持つため索引が作れず，データ件数に比例した計算量が必要となってしまう（要件 A を満たさない）．さらに，複数ユーザでの暗号化データの共有，言い換えれば検索権限の共有を行うことができない（要件 B を満たさない）．さらに，鈴木ら [81] や Popa ら [67] の提案方式を除いて，他方式はアルゴリズムの提案だけにとどまっており，データベースとの連携については検討されていない（要件 C を満たさない）．

要件 A を満たすため，公開鍵暗号ベースの検索可能暗号を高速化するアプローチとして 2 つのアプローチが考えられる．一つ目は (a) 一致判定処理そのものを高速化するアプローチ，もう一つは (b) データ件数に比例した一致判定処理の回数削減のアプローチ，例えば索引を用いた検索処理回数の削減などが考えられる．特に実利用においては，多くのデータが蓄積されるため，我々は (b) のアプローチが要件 A の達成には必要と考えている．

要件 A を満たすことを目指した性能改善の手法として，Shi ら [78] によって高次元範囲検索の手法が提案されている．この方式は，暗号化データの IND-CPA は保ちつつも，検索にヒットした場合に暗号化データの元データである数値情報が洩れる “match-revealing” と呼ばれる安全性の緩和を行うことで，一致判定処理が数値範囲 S の log オーダ $\mathrm{O}(\log|S|)$ で実施可能となった．しかし，一致判定処理そのものの高速化に着目した提案手法であるため，HVE と同様に，データ件数 N とした時に，データ件数に比例した検索処理時間 $\mathrm{O}(N \times \log|S|)$

がかかるという課題が残っている．

検索可能暗号を高速化するため，Bellare ら [9][10] によって RSA-OAEP をベースとした確定的暗号方式によって，データ数の log オーダで検索可能な方式が提案された．この方式は検索時間は改善されているものの，複数利用者間でのデータ共有ができないという課題がある．

また，検索可能暗号の高速化，及びデータベースへの検索可能暗号の組込みに関して，Popa ら [67] によって CryptDB と呼ぶデータベースが提案されている．この論文でも，Bellare らの方式と同様に確定的暗号を用いることで検索の高速化を図っているが，複数利用者間でのデータ共有は同じ鍵を共有することによって実現しており，安全なデータ共有が実現できないという課題がある．

また，要件 B に関して，服部ら [43] によって，検索可能暗号でのデータ共有を実現する方式について提案されている．本方式は複数ユーザ間でのデータ共有を実現しているが，Boneh らの方式 [17] と同様に検索処理がデータ数に比例するという点が改善されておらず，実用的な性能を出すことが難しい．

要件 C に関して，著者ら [58] も公開鍵ベースの検索可能暗号を市販データベースに組み込むための手法について提案している．また，その後，鈴木ら [81] によって，検索可能暗号を RDBMS に組込む一般的な構成方法へと拡張がなされている．しかし，本方式は共通鍵ベースの検索可能暗号では適用性が評価されているものの，公開鍵ベースの検索可能暗号に適用できるかは検証されておらず，プロトタイプ実装などもなされていない．

これまでの結果を表 4.1 にまとめるが，それぞれの要件を満たす方式は検討されているものの，全ての要件を同時に満たすように検討されている研究はなされていない．

4.1.3 我々のアプローチ

本論文では，*“索引生成可能な検索可能暗号”(Public-key Searchable Encryption with Index Generation)* を提案する．最初のステップとして，要件 A を満たす方式を提案し，その拡張として要件 A および要件 B を同時に満たす方式を提案する．その後，提案方式をデータベースに組み込むための実装方式についても提案し，要件 C を満たすようにする．

要件 A の実現では，データ匿名化 [72] の際に利用される汎化という考え方を応用してキーワードのグループ化を行うことで索引生成を可能とし，更に索引生成レベルの調整も可能とすることで検索性能とセキュリティレベルのバランスを調整できる基本方式を実現する．具体的には，キーワードを汎化してグループ化を行い，そのグループにグループ ID を付与する．グループ ID からはキーワードを一意に特定することができないようにしておく．そして，キーワードを暗号化する際に，確率的な公開鍵型検索可能暗号で暗号化を行うとともに，グループ ID を索引情報として付加する．検索時，検索クエリとともにグループ ID を指定することで，最初にグループ ID で検索対象を絞り込み，絞り込んだ暗号化データのみを検索可能暗号で検索することで，検索可能暗号の処理回数を削減することができるため，検索処理を高速化できる．検索時，このグループ ID を索引情報として用いることで，検索の高速化を図る．

索引生成の実現に加えて，グループ ID はあらかじめ暗号化しておき，サーバには分からないようにして保管する．グループ ID は，索引開示鍵をサーバに開示することで，部分的に開示することができる．グループ ID の開示範囲を動的に変更する仕組みを提供すること

で，データ量が多い場合や少ない場合でも，安全性と検索性能のトレードオフを利用者自身がコントロールできる仕組みも実現する．具体的には，前述のグループ ID をビット単位で暗号化し，最初はグループ ID がサーバに分からない状態で保管する．ユーザが索引開示鍵を作成してサーバに開示することで，サーバではグループ ID を 1 ビットずつ知ることができる．利用者は，索引開示鍵の個数を調整することで，暗号化データ数の増加の度合いを見ながら，安全性と検索性能のトレードオフが調整できる．

更に，IND-CPA セキュリティと匿名性を組み合わせた新しいセキュリティ定義も提案する．このセキュリティ定義では，開示された範囲で索引値が同じメッセージが 2 つあったとき，暗号化データを見てもいずれのメッセージを暗号化したものかが分からないことを定式化した．通常の IND-CPA セキュリティでは，暗号化データを見ても 2 つのメッセージのいずれを暗号化したかが分からないことを定義するので，ここに索引の考え方を取り込んだ自然な拡張である．

要件 B の実現では，階層型 ID を用いて検索可能なユーザを指定することで，暗号化データをマルチユーザで共有する拡張方式を実現する．階層型 ID を用いることで，ユーザ ID だけでなく，ユーザが属する企業やプロジェクト名も表現することができる．しかし，階層型 ID を導入するだけでは不十分である．これは，階層型 ID にはユーザ ID を指定するフィールドも含まれるため，階層型 ID を使って暗号化しても，グループ宛の暗号文ではなく個人宛の暗号文となってしまうためである．そこで，階層型 ID で検索可能なユーザを指定する際，ID としてワイルドカード “$*$” を指定できるようにすることで，特定の ID 階層以下に所属する複数ユーザが検索できる仕組みを実現する．ワイルドカードによって，ユーザ ID は誰であってもよい，という表現を行うことをサポートできるようになるため，所属やプロジェクトグループ宛の暗号化が実現できるようになる．この階層型 ID の表現，及びワイルドカードを実現するため，内積述語暗号 [53] を活用した．そして，ユーザ情報やマスター秘密鍵は幹事会社が運営する key administrator が管理し，ユーザに対して検索鍵を発行する仕組みとした．また，セキュリティ定義についても，マルチユーザ化を考慮した定義へと拡張を行った．

要件 C の実現では，検索可能暗号を Web アプリケーション (サービス) へ適用するために，本提案方式をデータベースに組み込む実装方式を示す．具体的には，データベースに対して暗号化データとタグを保管する際に，索引情報もテーブルに保管する．検索クエリを受け取った際は，検索クエリからも索引情報を生成する．この索引情報の一致・不一致の判定は，データベースのインデクシングの機能を用いて高速に判断することができるため，検索処理を高速化することができる．また，索引開示鍵を受け取った際に索引情報をアップデートすることもできる．これら機能を，ユーザ定義関数やユーザ定義型という SQL が持つ仕組みを使って実現することで，SQL を使って検索可能暗号が利用できるように実現を図った．

4.1.4 本章の構成

はじめに，4.2 節では既存の検索可能暗号のアルゴリズムの定義を示す．4.3 節， 4.4 節，4.5 節では，我々が提案する検索可能暗号のアルゴリズムについて，ステップ・バイ・ステップで示す．始めに要件 A だけを満たす基本方式，次に要件 A と要件 B の両方を満たす拡張方式の順で示す．4.5 節では，要件 C を満たすデータベース組込み方法について示すとともに，索引生成関数や性能評価などの実装について示す．

4.2 既存の検索可能暗号

4.2.1 表記法

集合 A から要素 y を一様分布で選ぶ場合，$y \xleftarrow{U} A$ と書く．値 y を z によって定義する場合，$y := z$ と書く．入力 x によってアルゴリズム A を実行して出力 a を得る場合，$A(x) \to a$ と書く．自然数の集合を $\mathbb{N}$，素数 $q \in \mathbb{N}$ によって定義される位数 q の有限体を $\mathbb{F}_q$ と書く．有限体 $\mathbb{F}_q$ 上で定義されるベクトルを，$\overrightarrow{x}$ と書く．この時，$\overrightarrow{x} := (x_1, \ldots, x_n) \in \mathbb{F}_q^n$ である．内積述語暗号 (HIPE) のメッセージ空間を M_{HIPE}，検索可能暗号のキーワード空間を $W \subseteq \mathbb{F}_q \setminus \{0, q-1\}$ と書く．階層型 ID のそれぞれの階層 $(1 \leq i \leq n)$ における ID を $identity_i$ とし，これを $\mathbb{F}_q \setminus \{0, q-1\}$ の要素としたとき，階層型 ID を $\{identity_1, \cdots, identity_n\}$ と書く．シングルクォートで囲まれた文字列'*string*' は，文字列 (string) に対応する $\mathbb{F}_q \setminus \{0, q-1\}$ の要素とする．文字列や ID を結合する場合，演算子 | を用いる．確率が neligible であるとは，あらゆる正多項式 p と任意のセキュリティパラメータ λ を用いて，確率が $1/p(\lambda)$ より小さいことを意味する．確率が overwhelming であるとは，あらゆる正多項式 p と任意のセキュリティパラメータ λ を用いて，確率が少なくとも $1 - 1/p(\lambda)$ であることを意味する．

4.2.2 検索可能暗号（Searchable Encryption）

検索可能暗号とは，暗号化データを復号することなく，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．実現方式としては公開鍵暗号に基づく方式，共通鍵暗号に基づく方式の大きく 2 つに大別されるが，我々は公開鍵暗号に基づく方式に着目する．そこで，公開鍵暗号に基づく方式について，Boneh ら [17] によって提案された PEKS と呼ばれるアルゴリズムの定義を下記に示す．

Definition 16. 非対話型の検索可能暗号は下記の多項式時間アルゴリズムによって構成される．

KeyGen: 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 mpk とマスター秘密鍵 msk のペアを生成し，マスター公開鍵とマスター秘密鍵のペア (mpk, msk) を出力

GenTag: マスター公開鍵 mpk，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

GenTrapdoor: マスター公開鍵 mpk，マスター秘密鍵 msk，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: マスター公開鍵 mpk，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

上記検索可能暗号の Correctness property は，任意のセキュリティパラメータ 1^λ，任意のキーワード w に対して，$\mathbf{Test}(mpk, Tag_w, Td_w) = TRUE$ が成り立つことである．ここで，

$\mathbf{KeyGen}(1^\lambda) \to (mpk, msk)$，$\mathbf{GenTag}(mpk, w) \to Tag_w$，$\mathbf{GenTrapdoor}(mpk, msk, w) \to Td_w$ である．

さらに，Consistency property は，$\mathbf{Test}(mpk, Tag_w, Td_{w'}) = FALSE$ が overwhelming な確率で成り立つことである．ここで，$w \neq w'$，$\mathbf{GenTag}(mpk, w) \to Tag_w$，$\mathbf{GenTrapdoor}(mpk, msk, w') \to Td_{w'}$ である．

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．

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．

Definition 17. セキュリティパラメータを λ とする．以下のゲームによって攻撃者 $\mathcal{A}$ のアドバンテージ $\mathbf{Adv}^{SE}_{\mathcal{A}}(\lambda)$ を定義し，そのアドバンテージが *negligible* であるとき，検索可能暗号は選択キーワード攻撃に対して識別不可能性を持つという．すなわち，*IND-CPA* セキュアである．

1. 挑戦者 $\mathcal{C}$ は $\mathbf{KeyGen}(1^\lambda)$ アルゴリズムを実行してマスター公開鍵 mpk，マスター秘密鍵 msk を生成し，マスター公開鍵 mpk を攻撃者 $\mathcal{A}$ に与える
2. 攻撃者 $\mathcal{A}$ 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. 攻撃者 $\mathcal{A}$ は，チャレンジする 2 つのキーワード w_0^* と w_1^* を選択し，挑戦者 $\mathcal{C}$ 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．挑戦者 $\mathcal{C}$ は，ランダムに $b \in \{0,1\}$ を選んでチャレンジタグ $S_{w_b} := \mathbf{GenTag}(mpk, w_b^*)$ を生成したのち，チャレンジタグ S_{w_b} のみを攻撃者 $\mathcal{A}$ に送付する．
4. 攻撃者 $\mathcal{A}$ は，ステップ 2 と同様に任意の検索キーワード w に対するトラップドアを取得する．ただし，自明な攻撃を防ぐため，$w \neq w_0^*$ かつ $w \neq w_1^*$ でなければならない
5. 最後に，攻撃者 $\mathcal{A}$ は，ビット b の推測値 $b' \in \{0,1\}$ を出力する．もし $b' = b$ であれば攻撃者の勝ちである

ここで，攻撃者 $\mathcal{A}$ のアドバンテージを下記式にて定義する．

$$\mathbf{Adv}^{SE}_{\mathcal{A}}(\lambda) = |\Pr[b' = b] - 1/2|$$

4.2.3 階層型内積述語暗号（Hierarchical Inner-product Predicate Encryption）

本提案手法の実現のために，我々は暗号化に用いられたベクトルの秘匿性である Attribute-hiding，およびデータの秘匿性を表す Payload-hiding の両者のセキュリティを満たす階層的内積述語暗号 (HIPE) を用いる．例えば，岡本・高島ら [53] によって提案された手法が知られており，これは Attribute-hiding と Payload-hiding の両方を満たしている．そのアルゴリズムの定義を下記に引用する．

Definition 18. 次元 d の属性空間の階層フォーマットを $\overrightarrow{\mu} := (n, d; \mu_1, \ldots, \mu_d)$ とする．ここで，$0 < \mu_1 < \mu_2 < \cdots < \mu_d = n$ である．階層型述語暗号 *(Hierarchical Predicate Encryption)* の一つである階層型内積述語暗号 *(HIPE: Hierarchical Inner-Product Encryption)* は，階層型属性 Σ 上の内積述語 F に対して，確率的多項式時間アルゴリズム **Setup**，**KeyGen**，**Enc**，**Dec**，**Delegate**$_\ell(\ell = 1, \ldots, d-1)$ で構成される．その定義は次のように与えられる．

- **Setup** は，セキュリティパラメータ 1^λ，階層フォーマット $\overrightarrow{\mu}$ を入力として受け取り，マスター公開鍵 mpk とマスター秘密鍵 msk を出力する．
- **KeyGen** は，マスター公開鍵 mpk，マスター秘密鍵 msk，述語ベクトル $(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_\ell)$ を入力として受け取り，対応する秘密鍵 $sk_{(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_\ell)}$ を出力する．
- **Enc** は，マスター公開鍵 mpk，階層 $1 \leq h \leq d$ である属性ベクトル $(\overrightarrow{x}_1, \cdots, \overrightarrow{x}_h)$，平文空間 **msg** に含まれる平文 m を受け取り，暗号文 c を出力する．
- **Dec** は，マスター公開鍵 mpk，階層 $1 \leq \ell \leq d$ である秘密鍵 $sk_{(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_\ell)}$，暗号文 c を入力として受け取り，平文 m もしくはエラー $\perp$ を出力する．
- **Delegate**$_l$ は，マスター公開鍵 mpk，階層 ℓ である秘密鍵 $sk_{(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_\ell)}$，階層 $(\ell+1)$ の述語ベクトル $\overrightarrow{v}_{\ell+1}$ を入力として受け取り，階層 $\ell+1$ である $sk_{(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_{\ell+1})}$ を出力する．

階層型内積述語暗号の Correctoness property は，任意のセキュリティパラメータ 1^λ，任意の述語ベクトル $(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_\ell)$，述語ベクトルとの内積値が 0 となる任意の属性ベクトル $(\overrightarrow{x}_1, \cdots, \overrightarrow{x}_\ell)$ に対して，下記が成り立つことである．

$$\begin{aligned}
&\mathbf{Dec}(mpk, sk_{(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_\ell)}, c) \to m, \\
&\quad \text{s.t. } \overrightarrow{v}_i \cdot \overrightarrow{x}_i = 0 \ (1 \leq i \leq \ell), \\
&\quad \text{where } \mathbf{Setup}(1^\lambda, \overrightarrow{\mu}) \to (mpk, msk), \\
&\quad \mathbf{Enc}(mpk, (\overrightarrow{x}_1, \cdots, \overrightarrow{x}_\ell), m) \to c, \\
&\quad \mathbf{KeyGen}(mpk, msk, (\overrightarrow{v}_1, \cdots, \overrightarrow{v}_k)) \to sk_{(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_k)}, \\
&\quad \text{and } \mathbf{Delegate}_j(mpk, sk_{(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_j)}) \to sk_{(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_{j+1})}.
\end{aligned}$$

加えて，Consistency property は，少なくとも一つの i に対して $\overrightarrow{v}_i \cdot \overrightarrow{x}_i \neq 0$ であるとき，overwhelming な確率にて下記が成り立つことである．

$$\mathbf{Dec}(mpk, sk_{(\overrightarrow{v}_1, \cdots, \overrightarrow{v}_\ell)}, c) \to \perp$$

内積述語暗号の安全性としては，*payload-hiding* と *attribute-hiding* という 2 つの重要な考え方がある．定義はほぼ同じであるが，チャレンジフェーズに若干の違いがある．payload hiding では攻撃者 $\mathcal{A}$ が 2 つのチャレンジメッセージ m_0^* と m_1^* を選ぶのに対して，attribute hiding の場合は攻撃者 $\mathcal{A}$ は 2 つのチャレンジ属性 $\overrightarrow{x}_0^*$ と $\overrightarrow{x}_1^*$ を選ぶ．

Definition 19. 階層型内積述語暗号が *CPA* 攻撃者に対して *payload hiding (PH)* であるとは，任意の多項式時間攻撃者 $\mathcal{A}$ に対して下記のゲームを考えたとき，攻撃者のアドバンテージがセキュリティパラメータに対して *negligible* であることである．

1. チャレンジャーは **Setup** を実行し，マスター公開鍵 mpk とマスター秘密鍵 msk のペアを生成する．マスター公開鍵 mpk は攻撃者 $\mathcal{A}$ に与える．

2. 攻撃者 $\mathcal{A}$ は適用的に多項式回だけ下記に示すクエリーを実行できる．

   - *Create key:* 攻撃者 $\mathcal{A}$ は述語 $f \in F$ に対する秘密鍵の生成を要求する．チャレンジャーは述語 f に対する秘密鍵を生成するが，生成した時点では攻撃者 $\mathcal{A}$ に対して秘密鍵は開示しない．
   - *Create delegated key:* 攻撃者 $\mathcal{A}$ は既に秘密鍵を生成した述語 f を指定し，述語 $f' \leq f$ にたいして委譲 *(delegate)* の実施を要求する．*Create key* と同様に，チャレンジャーは委譲した秘密鍵を生成した時点では，秘密鍵は攻撃者 $\mathcal{A}$ に対して開示しない．
   - *Reveal key:* 攻撃者 $\mathcal{A}$ は既に秘密鍵を生成した述語 f を指定し，秘密鍵の開示を要求する．チャレンジャーは，この要求を受け取った時点で述語 f に対応する秘密鍵を攻撃者 $\mathcal{A}$ に開示する．

3. 攻撃者 $\mathcal{A}$ はチャレンジ属性 $\overrightarrow{x} := (\overrightarrow{x}_1, \ldots, \overrightarrow{x}_h)$，およびチャレンジメッセージ m_0^* と m_1^* を出力する．なお制約として，既に *Reveal key* で取得した秘密鍵を使ってチャレンジ暗号文が識別できる様なチャレンジ属性，チャレンジメッセージを指定することはできない．

4. チャレンジャーはランダムにビット b を選び，攻撃者 $\mathcal{A}$ に対してチャレンジ暗号文 $c^{(b)} := \mathbf{Enc}(mpk, m_b^*, \overrightarrow{x})$ を与える．

5. 攻撃者 $\mathcal{A}$ は，ステップ *2* と同様に繰り返しクエリーを実行することができる．なお，チャレンジ暗号文が識別できる様な述語を指定して *Reveal key* クエリを行うことはできない．

6. 攻撃者 $\mathcal{A}$ はビット b' を出力する．もし $b' = b$ であれば攻撃者の勝ちである．

上記ゲームにおいて，攻撃者 $\mathcal{A}$ のアドバンテージは次のように定義される．

$$\mathbf{Adv}_{\mathcal{A}}^{HIPE,PH}(\lambda) = |\Pr[b' = b] - 1/2|$$

Definition 20. 階層型内積述語暗号が *CPA* 攻撃者に対して *attribute-hiding (AH)* であるとは，任意の多項式時間攻撃者 $\mathcal{A}$ に対して下記のゲームを考えたとき，攻撃者のアドバンテージがセキュリティパラメータに対して *negligible* であることである．

1. チャレンジャーは **Setup** を実行し，マスター公開鍵 mpk とマスター秘密鍵 msk のペアを生成する．マスター公開鍵 mpk は攻撃者 $\mathcal{A}$ に与える．

2. 攻撃者 $\mathcal{A}$ は適用的に多項式回だけ下記に示すクエリーを実行できる．

   - *Create key:* 攻撃者 $\mathcal{A}$ は述語 $f \in F$ に対する秘密鍵の生成を要求する．チャレンジャーは述語 f に対する秘密鍵を生成するが，生成した時点では攻撃者 $\mathcal{A}$ に対して秘密鍵は開示しない．

- *Create delegated key:* 攻撃者 $\mathcal{A}$ は既に秘密鍵を生成した述語 f を指定し，述語 $f' \leq f$ にたいして委譲 *(delegate)* の実施を要求する．*Create key* と同様に，チャレンジャーは委譲した秘密鍵を生成した時点では，秘密鍵は攻撃者 $\mathcal{A}$ に対して開示しない．
- *Reveal key:* 攻撃者 $\mathcal{A}$ は既に秘密鍵を生成した述語 f を指定し，秘密鍵の開示を要求する．チャレンジャーは，この要求を受け取った時点で述語 f に対応する秘密鍵を攻撃者 $\mathcal{A}$ に開示する．

3. 攻撃者 $\mathcal{A}$ はチャレンジ属性 $\overrightarrow{x}_0^* := (\overrightarrow{x}_1^{(0)}, \ldots, \overrightarrow{x}_{h_0}^{(0)})$ と $\overrightarrow{x}_1^* := (\overrightarrow{x}_1^{(1)}, \ldots, \overrightarrow{x}_{h_1}^{(1)})$，およびチャレンジメッセージ m を出力する．なお制約として，既に *Reveal key* で取得した秘密鍵を使ってチャレンジ属性が識別できる様なチャレンジ属性，チャレンジメッセージを指定することはできない．すなわち，既に *Reveal key* で取得した秘密鍵を取得した全ての f に対して，$f(\overrightarrow{x}_0^*) = f(\overrightarrow{x}_1^*)$ でなければならない．

4. チャレンジャーはランダムにビット b を選び，攻撃者 $\mathcal{A}$ に対してチャレンジ暗号文 $c^{(b)} := \mathbf{Enc}(mpk, m, \overrightarrow{x}_b^*)$ を与える．

5. 攻撃者 $\mathcal{A}$ は，ステップ *2* と同様に繰り返しクエリーを実行することができる．なお，チャレンジ暗号文が識別できる様な述語を指定して *Reveal key* クエリを行うことはできない．すなわち，$f(\overrightarrow{x}_0^*) = f(\overrightarrow{x}_1^*)$ でなければならない．

6. 攻撃者 $\mathcal{A}$ はビット b' を出力する．もし $b' = b$ であれば攻撃者の勝ちである．

上記ゲームにおいて，攻撃者 $\mathcal{A}$ のアドバンテージは次のように定義される．

$$\mathbf{Adv}_{\mathcal{A}}^{HIPE,AH}(\lambda) = |\Pr[b' = b] - 1/2|$$

高島ら [53] によって提案された階層型内積述語暗号は，上記の payload-hiding と attribute-hiding の両方の性質を満たすことが証明されている．そこで，我々は高島らの内積述語暗号を我々のスキームの実装で用いる．

4.2.4 HIPE による検索可能暗号の一般的構成法

Abdalla ら [1] によって提案された ID ベース暗号から検索可能暗号を構成する一般的構成法を用いて，内積述語暗号からシングルユーザ版検索可能暗号を構成することができる．以下にその構成方法を示す．

- $\mathbf{Setup}(1^\lambda) \to (mpk, msk)$

  $\overrightarrow{\mu} := (n = 2, d = 1; \mu_1 = 2)$

  return $(mpk, msk) := \mathbf{Setup}_{HIPE}(1^\lambda, \overrightarrow{\mu})$

- $\mathbf{GenTag}(mpk, w) \to Tag_w$

  $r \xleftarrow{U} M$, $c := \mathbf{Enc}_{HIPE}(mpk, \overrightarrow{x} := (w, 1), r)$

  return $Tag_w := (r, c)$

- **GenTrapdoor**$(mpk, msk, w') \rightarrow Td_{w'}$

  return $Td_{w'} :=$ **KeyGen**$(mpk, msk, \overrightarrow{v} := (1, -w'))$

- **Test**$(mpk, Tag_w, Td_{w'}) \rightarrow \{0, 1\}$

  $m' :=$ **Dec**$(mpk, Td_{w'}, c)$

  If $m' = r$, $result :=$ TRUE, otherwise $result :=$ FALSE

  return $result$

概略を述べると，ランダムに生成した乱数 r を，キーワード w から生成した属性ベクトルで暗号化することで，タグを生成する．タグには，乱数 r もそのまま含める．トラップドアは，検索キーワード w' から生成した述語ベクトルを用いて鍵生成することで，それをトラップドアとする．そのため，属性ベクトル側のキーワード w と述語ベクトル側のキーワード w' が一致した場合，すなわち暗号化データ内のキーワード w と検索要求側のキーワード w' が一致した場合，**Test** アルゴリズム内で実行する HIPE 復号処理にて，**GenTag** アルゴリズム側で生成してタグに含めていた乱数 r が正しく復号されるため，キーワードが一致していることが分かる．ここで，HIPE が attribute-hiding を満たしているなら，暗号化データから暗号化に用いた属性ベクトルが漏れないことが保証されているため，タグからキーワード w に関する情報を得ることができない．この構成によって，HIPE からシングルユーザ版検索可能暗号が構成できることが分かるが，我々はこの方式をマルチユーザ版に拡張できることを示す (4.4 節).

4.3 索引生成対応検索可能暗号

本節では，要件 A を満たす基本方式について示す.

4.3.1 索引生成による高速化のアイデア

高速化を実現するための索引生成に関するアイデアについて説明する．従来の検索可能暗号では，暗号化タグからは一切の情報が得られないため，1 回の検索にて全ての暗号化タグに対してトラップドアとの比較を行うための一致判定処理 **Test** を実施する必要が生じる．そのため，暗号化タグの総数 N に対して，検索の処理量は $O(N)$ となっていた.

この検索性能を改善させるため，我々は検索可能暗号に索引生成機能の導入を目指した．我々のアイデアは，キーワードからタグを生成する際に，キーワードから確定的アルゴリズムで導出した索引値をタグと関連付けて保管する．同様に，トラップドア生成においてもキーワードから索引値を生成し，検索要求としてトラップドアと索引値をサーバに送付する．説明の都合上，両者を混同する可能性のある場合は，それぞれタグ索引値，トラップドア索引値と呼ぶ．そして，タグの一致判定処理を行う前に，サーバはタグ索引値とトラップドア索引値を比較し，両者が一致する暗号化タグだけ検索結果候補として抽出する．その検索結果候補になったタグに対してだけ，一致判定処理 **Test** を実施する．そのため，全てのタグに対して判定処理をするのではなく，検索結果候補となった一部のタグに対してのみ一致判定処理を行うことで，一回の検索で処理するタグの数を減らすことを目指す.

このアイデアは，匿名化と呼ばれる研究分野で使われている汎化 (generalization) という考え方に相当する．住所を例にして，我々のアイデアを説明する．例えばデータベースに住所というカラムがある場合，住所が漏洩すると個人が特定されるリスクがあるため，住所は個人情報の一つと考えられるため，暗号化して保護しておきたい．しかし，住所がどの都道府県（グループ）に属するかという情報であれば，個人の特定につながるリスクは極めて小さくなるため，都道府県の情報は暗号化しなくてもよいかもしれない．このようなケースでは，住所という暗号化データに対して，都道府県によるグループを索引情報として使うことが可能と考えられる．日本で一番人口が多い東京都であっても，その人口は日本の全人口の 1 割程度であることから，索引として「東京都」が指定されていた場合でも検索対象のタグを 1 割に削減することができ，一定の匿名性を確保したまま 10 倍の高速化を図ることができる．

我々は合わせて索引の開示レベルをコントロールする仕組みも提案する．言い換えれば，汎化のレベルを調整できることを意味する．これは，システムを長期間にわたって運用する場合，最初は保管したデータ件数が少なく，徐々にデータ件数が増えていく．例えば，当初想定よりもデータ件数が増えないケース，想定よりもデータ件数が増加したケース，など様々に起こりうる．いかなるケースにおいても一定の検索性能を確保するには，データ件数が最も多いケースに合わせ，さらにマージンを見込んで匿名化レベルを決定する必要が生じるため，匿名化レベルを事前に決定するのが難しい．そのため，データのサーズに応じて汎化のレベル，言い換えれば索引の開示レベルをコントロールできる仕組みは非常に有用と考えられる．これを実現するため，複数の匿名化レベルで索引値を生成し，それぞれ暗号化してタグに添付することとした．もし Key Administrator が性能改善が必要だと判断した場合，開示する索引の量に応じて索引開示鍵を生成し，データベースに送付する．これによりデータベースは，それまでより多くの索引をタグから導出できるようになり，より詳細に索引生成が可能となる．このように，利用状況を見ながら必要なレベルの索引値を開示するようにすることで，その調整を容易にする仕組みを実現した．

例えば，データの件数が少ない場合は，全ての暗号化タグを判定処理しても十分に実用な時間で検索処理が終了すると考えられる．この場合，住所がどの県に属するかという情報は一切開示する必要はない．しかし，データが徐々に蓄積されてくると検索時間を要するようになるため，このタイミングで住所が東日本・西日本のどちらのグループに属するかを開示することで，検索時間を短縮することができる．更にデータが蓄積されて処理時間を要するようになったら，今度は関東地方や近畿地方などの情報を索引として開示することで高速化する．同様にデータが蓄積されて処理時間を要するようになったら，最終的には県の情報を索引として開示することで高速化を図る．このように，データ件数が少ない場合は索引を開示せずに秘匿性を最大限に確保しつつ，データ件数の増加に応じて索引情報を提示することで秘匿性と高速化とのバランスを利用者自身が調整できるようにする．

上記は住所を例として説明したが，この考え方を，住所を含めて様々なデータに対して適用可能にしたものが本提案手法である．**GenTag** アルゴリズムや **GenTrapdoor** アルゴリズムで用いる索引関数を $\mathbf{Index} : \{0,1\}^* \times \{1,\ldots,L_{index}\} \to \{0,1\}^j$ とする．ここで，j は索引値のサイズ，L_{index} は最大索引階層数であり，$\mathbf{Index}(,)$ アルゴリズムは多項式時間で計算できるものである．これは，キーワード w と索引階層数 $\ell (1 \leq \ell \leq L_{index})$ が与えられたとき，索引値 $idx_\ell \in \{0,1\}^j$ が生成されることを意味する．ここで，キーワード w に対して，$\mathbf{Index}(w,\ell) = \mathbf{Index}(w',\ell)$ となるキーワード $w' (\neq w)$ が存在すると仮定する．i レベ

ルの索引値 idx_i を開示した場合，キーワード空間 W を k_i 個のグループに分割できる．先ほどの住所の例でいえば，$|W| = 47$ であり，最大索引階層数 $L_{index} = 3$，各レベルの索引において $k_1 = 2$，$k_2 = 8$，$k_3 = 47$ となる．言い換えれば，$idx_1 \in \{0, 1\}$，$idx_2 \in \{0, \ldots, 7\}$，$idx_3 \in \{0, \ldots, 46\}$ となる．もし，索引関数がキーワード空間 W を誤差 $\pm\delta \in \mathbb{N}$ の範囲で概ね同一サイズに分割するなら，例えば $\ell = 1$ の場合は，k_1 個のグループのそれぞれは概ね $|W|/k_1 - \delta$ 個のキーワードを含む．そのため，$|\{w \in W \mid idx_1 = 0\}| \geq |W|/k_1 - \delta$ かつ $|\{w \in W \mid idx_1 = 1\}| \geq |W|/k_1 - \delta$ となる．すなわち，索引値 ind_1 が開示されたとしても，索引値からは暗号化データの元となったキーワードの候補は $|KW|/k_1 - \delta$ 個までしか絞り込むことができず，$|KW|/k_1 - \delta$-匿名性を確保することができる．上記はタグだけでなくトラップドアについても同様の事が言える．この索引値を用いることで，データベースの検索処理（**Test** の処理回数）は $O(N)$ から $O(N/k_1)$ へと削減できるため，検索性能の向上が見込める．

また，実システムへ適用する場合は，きめ細かく安全性と検索性能のバランスを調整できる仕組みが好ましいと考えている．そのため，本研究では各階層の索引値は $\{0, 1\}$ の 1 ビットになるように，2 分木の考え方を用いてキーワード空間をグループ化することで，Index 関数を構成することを想定した．これにより，索引値を 1 ビット単位で開示コントロールができるようになり，タグから漏れる情報量が 1 ビット増えることで安全性は低下する一方で，1 ビットの開示に伴って検索性能が 2 倍になる．

4.3.2 アルゴリズムの定義

我々の索引生成対応検索可能暗号のアルゴリズム定義を示す．

Definition 21. シングルユーザ版の索引生成対応公開鍵検索可能暗号は，以下に示す確率的多項式時間アルゴリズム **Setup**, **GenTag**，**GenTrapdoor**，**Test**，**GenIndexKey**，**ExtractTagIndex**，**ExtractTrapdoorIndex** から構成される．その定義は次のように与えられる．

Setup: セキュリティパラメータ 1^λ，最大索引ビット数 L_{index} を入力とし，マスター公開鍵 mpk とマスター秘密鍵 msk のペアを出力

GenTag: マスター公開鍵 mpk，キーワード空間 W から選ばれたキーワード w を受け取り，タグ Tag_w を出力

GenTrapdoor: マスター公開鍵 mpk，マスター秘密鍵 msk，検索キーワード w を受け取り，トラップドア Td_w を出力

Test: マスター公開鍵 mpk，タグ Tag_w，トラップドア Td_w を受け取り，タグとトラップドアの一致判定結果（*1:*一致，*0:*不一致）を出力．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の場合，判定結果として 1 が出力され，それ以外の場合は 0 が返る

GenIndexKey: マスター公開鍵 mpk，マスター秘密鍵 msk，生成を許可する索引ビット数 ℓ を受け取り，索引開示鍵 $IKey_\ell$ を出力

ExtractTagIndex: マスター公開鍵 mpk，タグ Tag_w，索引開示鍵 $IKey_\ell$ を受け取り，索引値 $\{0,1\}^*$ を出力（もしくはエラーを返す）

ExtractTrapdoorIndex: マスター公開鍵 mpk，トラップドア Td_w，索引開示鍵 $IKey_\ell$ を受け取り，索引値 $\{0,1\}^*$ を出力（もしくはエラーを返す）

本スキームが正しく動作するためには，同一のキーワードから作られたタグとトラップドアは一致判定処理 **Test** で True が返ることが必要である．同様に，2 つのキーワード w と w' が階層 ℓ において同じ索引値を持つならば，キーワード w と w' から生成したタグやトラップドアから階層 ℓ の索引値を導出した場合，その索引値は同一でなければならない．そのため，下記に示す Correctness condition を満たす必要がある．

Definition 22 (Correctness conditions)**.** シングルユーザ版の索引生成対応公開鍵検索可能暗号の *Correctness conditions* は，任意のセキュリティパラメータ 1^λ，任意の索引階層数 $\ell \leq L_{index}$，ℓ 層目までの索引値が同一の値となる任意の異なるキーワード w および w' に対して，以下が成り立つことである．

1. $\mathbf{Test}(mpk, Tag_w, Td_w) = TRUE$
2. $\mathbf{ExtractTagIndex}(mpk, Tag_w, IKey_\ell)$
   $= \mathbf{ExtractTrapdoorIndex}(mpk, Td_w, IKey_\ell)$
3. $\mathbf{ExtractTagIndex}(mpk, Tag_w, IKey_\ell)$
   $= \mathbf{ExtractTagIndex}(mpk, Tag_{w'}, IKey_\ell)$

where $\mathbf{KeyGen}(1^\lambda) \to (mpk, msk)$, $\mathbf{GenTag}(mpk, w) \to Tag_w$,
$\mathbf{GenTag}(mpk, w') \to Tag_{w'}$, $\mathbf{GenTrapdoor}(mpk, msk, w) \to Td_w$,
$\mathbf{GenIndexKey}(mpk, msk, \ell) \to IKey_\ell$.

また，不要なデータが検索されないようにするためには，異なるキーワードから作られたタグとトラップドアであれば，一致判定処理 **Test** が False を返す必要がある．同様に，2 つのキーワード w と w' が階層 ℓ において異なる索引値を持つならば，キーワード w と w' から生成したタグやトラップドアから階層 ℓ の索引値を導出した場合，その索引値は異なっていなければならない．そのため，下記に示す Cosistency condition を満たす必要がある．

Definition 23 (Consistency conditions)**.** シングルユーザ版の索引生成対応公開鍵検索可能暗号の *Consistency conditions* は，任意のセキュリティパラメータ 1^λ，任意の索引階層数 $\ell \leq L_{index}$，ℓ 層目までの索引値が異なる値を取る任意の異なるキーワード w および w' に対して，以下が成り立つことである．

1. $\mathbf{Test}(mpk, Tag_w, Td_{w'}) = FALSE$ *with overwhelming probability*
2. $\mathbf{ExtractTagIndex}(mpk, Tag_w, IKey_\ell)$
   $\neq \mathbf{ExtractTrapdoorIndex}(mpk, Td_{w'}, IKey_\ell)$
3. $\mathbf{ExtractTagIndex}(mpk, Tag_w, IKey_\ell)$
   $\neq \mathbf{ExtractTagIndex}(mpk, Tag_{w'}, IKey_\ell)$

where $\mathbf{KeyGen}(1^\lambda) \to (mpk, msk)$, $\mathbf{GenTag}(mpk, w) \to Tag_w$, $\mathbf{GenTag}(mpk, w') \to Tag_{w'}$, $\mathbf{GenTrapdoor}(mpk, msk, w') \to Td_{w'}$, $\mathbf{GenIndexKey}(mpk, msk, \ell) \to IKey_\ell$.

4.3.3 セキュリティの定義

4.3.1 節では，k-anonimity の考え方を使って索引情報を構成できるという安全性について考察した．つぎに本節では，索引生成対応検索可能暗号について，その安全性を定義する．提案方式では，索引値は最初は保護されており，性能改善のために部分的に開示していく．そこで，索引を開示していない場合は IND-CPA と同じセキュリティが保たれているが，索引値を開示した場合は，開示した索引情報に対応するグループにキーワードが属することは分かるが，そのグループに対応するキーワードのうち，どのキーワードかを特定することはできない，という状況を索引生成対応検索可能暗号のセキュリティとして定式化する．以下に，4.3.2 節にて示した索引生成対応検索可能暗号の安全性定義を示す．

Definition 24. 最大索引階層数 L_{index} は任意の定数とする．最大索引階層数 L_{index} であるシングルユーザ版の索引生成対応公開鍵検索可能暗号が，*CPA* 攻撃者に対して *adaptively secure* であるとは，以下のゲームを実施する全ての確率的多項式時間攻撃者 $\mathcal{A}$ のアドバンテージがセキュリティパラメータ 1^λ に対して *negligible* であることである．

1. チャレンジャーは **Setup** を実行し，マスター公開鍵 mpk とマスター秘密鍵 msk のペアを生成する．マスター公開鍵 mpk は攻撃者 $\mathcal{A}$ に与える．

2. 攻撃者 $\mathcal{A}$ は適用的に多項式回だけ下記に示すクエリーを実行できる．

   - *Trapdoor query:* 攻撃者 $\mathcal{A}$ は任意のキーワード $w \in W$ に対するトラップドアの生成を要求する．チャレンジャーはキーワード w に対応するトラップドア Td_w を生成し，これを攻撃者 $\mathcal{A}$ に送付する．
   - *Index key query:* 攻撃者 $\mathcal{A}$ は任意の索引階層数 $\ell \leq L_{index}$ に対する索引開示鍵 $IKey_\ell$ の生成を要求する．チャレンジャーは索引階層数 ℓ に対応する索引開示鍵 $IKey_\ell$ を生成し，これを攻撃者 $\mathcal{A}$ に送付する．

3. 攻撃者 $\mathcal{A}$ はチャレンジキーワード $w_0^*, w_1^* \in W$ を選び，チャレンジャーに送付する．なお制約として，*Trapdoor query* で取得したトラップドアや，*Index key query* で取得した索引開示鍵を使ってチャレンジキーワードが識別できる様なチャレンジキーワードを指定することはできない．

4. チャレンジャーはランダムにビット b を選び，攻撃者 $\mathcal{A}$ に対してチャレンジ暗号文 $Tag_{w_b^*} := \mathbf{GenTag}(mpk, w_b^*)$ を与える．

5. 攻撃者 $\mathcal{A}$ は，ステップ *2* と同様に繰り返しクエリーを実行することができる．なお，チャレンジキーワード文が識別できる様なキーワード w を指定して *Trapdoor query* を行うことはできない．同様に，チャレンジキーワード文が識別できる様な索引階層数 ℓ を指定して *Index key query* を行うことはできない．

6. 攻撃者 $\mathcal{A}$ はビット b' を出力する．もし $b' = b$ であれば攻撃者の勝ちである．

上記ゲームにおいて，攻撃者 $\mathcal{A}$ のアドバンテージは次のように定義される．

$$\mathbf{Adv}_{\mathcal{A}}^{SE}(\lambda) = |\Pr[b' = b] - 1/2|$$

本ゲームでは，一般的な検索可能暗号の IND-CPA ゲームと同様に，攻撃者が自明な攻撃をできないようにするため，クエリーに対して幾つか制約を設けている．それに加えて，本定義では L_{index} をセキュリティパラメータとは独立した小さな定数という仮定を置いている．

その理由は，多項式時間攻撃者 $\mathcal{A}$ が，索引値が同一となるチャレンジキーワードを選べることを保証するためである．グループ $S_{Idx} = \{w \in W | \mathbf{Index}(w) = Idx\}(Idx \in Image(\mathbf{Index}))$ の数は，最大索引階層数 L_{index} の増加に伴い指数的に増えていく．上記ゲームでは，多項式時間攻撃者は同じ索引値を持つ 2 つのキーワード $\{w, w' | \mathbf{Index}(w) = \mathbf{Index}(w')\}$ を選ぶ必要がある．もし L_{index}(より正確には ℓ) が，例えば 10 や 20 程度の小さな定数であれば，多項式時間攻撃者 $\mathcal{A}$ であっても，索引値が同一となる 2 つのキーワードを選択できるので，上記ゲームを成り立たせることができる．一方，L_{index} が 100 や 200 など大きくなると，多項式時間攻撃者 $\mathcal{A}$ は索引値が同一となる 2 つのキーワードを選ぶことができなくなり，ゲームが成立しない．そのため，L_{index} がセキュリティパラメータ 1^λ とは独立の小さな定数という仮定が必須となる．

実際のケースでは，この制約は問題にはならないと考えている．例えば日本人の苗字であれば，その種類はおよそ 10 万種類程度と言われている [24]．このようなケースでは，最大索引階層数 $L_{index} = 5$ とした場合に概ね 3000-anonymity となり，検索性能はおよそ $2^5 = 32$ 倍高速化される．また，最大索引階層数 $L_{index} = 10$ とした場合に概ね 100-anonymity となり，検索性能はおよそ 1024 倍高速化される．このように，最大索引階層数をある定数以下と制約を加えても，実用の範囲では十分と考えている．

4.3.4 構成方法

シングルユーザセッティングのケースにおけるアルゴリズムを下記に示す．

$\mathbf{Setup}(1^\lambda, L_{index}) \rightarrow (mpk, msk)$

$\overrightarrow{\mu}_{HIPE} := (n = 2d, d = 2; \mu_1 = 2, \mu_2 = 4)$,
$(mpk_{HIPE}, msk_{HIPE}) := \mathbf{Setup}_{HIPE}(1^\lambda, \overrightarrow{\mu}_{HIPE})$
choose suitable index function, $\mathbf{Index}(w \in W, \ell \in \{1, \ldots, L_{index}\}) \rightarrow \{0,1\}^*$,
　which returns an ℓ-th level of index value.
$mpk := (mpk_{HIPE}, L_{index}, \mathbf{Index}(,))$, $msk := (msk_{HIPE})$
return (mpk, msk)

$\mathbf{GenTag}(mpk, w) \rightarrow Tag_w$

$r \xleftarrow{U} M_{HIPE}$,
$c_{SE} := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'SE'}, w\}), r)$,
$Tag_{SE} := (r, c_{SE})$
For all $i \in \{1, \ldots, L_{index}\}$,
　$EncIdxTag_i := \mathbf{Enc}_{HIPE}(mpk_{HIPE},$
　　$\mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'IdxTag'}, i\}), \mathbf{Index}(w, i))$

return $Tag_w := (Tag_{SE}, \{EncIdxTag_i\}_{L_{index}})$

GenTrapdoor$(mpk, msk, w) \rightarrow Td_w$

$Td_{SE} :=$ **KeyGen**$(mpk_{HIPE}, msk,$ **EncodePredicate**$(mpk_{HIPE}, \{\text{'SE'}, w\}))$

For all $i \in \{1, \ldots, L_{index}\}$,
$EncIdxTd_i :=$ **Enc**$_{HIPE}(mpk_{HIPE},$
EncodeAttribute$(mpk_{HIPE}, \{\text{'IdxTd'}, i\}),$ **Index**$(w, i))$

return $Td_w := (Td_{SE}, \{EncIdxTd_i\}_{L_{index}})$

Test$(mpk, Tag_w, Td_w) \rightarrow \{TRUE, FALSE\}$

$m' :=$ **Dec**$(mpk_{HIPE}, Td_{SE}, C_{SE})$

If $m' = r$, $result =$ TRUE, otherwise $result =$ FALSE

return $result$

GenIndexKey$(mpk, msk, \ell) \rightarrow IKey_\ell$

For all $i \in \{1, \ldots, \ell\}$,
$IKTag_i :=$ **KeyGen**$(mpk_{HIPE}, msk_{HIPE},$
EncodePredicate$(mpk_{HIPE}, \{\text{'IdxTag'}, i\}))$,
$IKTd_i :=$ **KeyGen**$(mpk_{HIPE}, msk_{HIPE},$
EncodePredicate$(mpk_{HIPE}, \{\text{'IdxTd'}, i\}))$

return $IKey_\ell := (\{IKTag_i\}_\ell, \{IKTd_i\}_\ell)$

ExtractTagIndex$(mpk, Tag_w, IKey_\ell) \rightarrow \{0,1\}^*$

For all $i \in \{1, \ldots, \ell\}$,
$IdxTag_i :=$ **Dec**$(mpk_{HIPE}, IKTag_i, EncIdxTag_i)$

return $result := IdxTag_1 | \ldots | IdxTag_\ell$

ExtractTrapdoorIndex$(mpk, Td_w, IKey_\ell) \rightarrow \{0,1\}^*$

For all $i \in \{1, \ldots, \ell\}$,
$IdxTd_i :=$ **Dec**$(mpk_{HIPE}, IKTd_i, EncIdxTd_i)$

return $result := IdxTd_1 | \ldots | IdxTd_\ell$

EncodeAttribute$(pk, attr) \rightarrow (\overrightarrow{x}_1, \overrightarrow{x}_2)$

For all $i \in \{1, 2\}$,
$\sigma_i \xleftarrow{U} \mathbb{F}_q \setminus \{0\}$, $\overrightarrow{x}_i = \sigma_i(attr_i, 1)$

return $(\overrightarrow{x}_1, \overrightarrow{x}_2)$

EncodePredicate$(pk, pred) \rightarrow (\overrightarrow{v}_1, \overrightarrow{v}_2)$

For all $i \in \{1, 2\}$,
$\rho_i \xleftarrow{U} \mathbb{F}_q \setminus \{0\}$, $\overrightarrow{v}_i = \rho_i(1, -pred_i)$

return $(\overrightarrow{v}_1, \overrightarrow{v}_2)$

GenTag はタグを生成するアルゴリズムである．タグは 2 つの要素からなり，一つは通常の検索可能暗号のタグに相当する Tag_{SE} (SE パート) である，もう一つが暗号化索引 $\{EncIdxTag_i\}_{L_{index}}$ である．SE パートは，4.2.4 節で述べたような一般的な検索可能暗号の手法により生成する．暗号化索引 $\{EncIdxTag_i\}_{L_{index}}$ は，**Index** 関数を用いて生成した索引情報を索引階層ごとに暗号化することで生成する．**GenTrapdoor** アルゴリズムも同様で，通常の検索可能暗号のトラップドア (SE パート) と，タグと同様に索引情報を暗号化したもので構成される．**Test** アルゴリズムは，タグの SE パートと，トラップドアの SE パートを一致判定することで，データを復号することなくキーワードの一致不一致を判定する．**GenIndexKey** アルゴリズムは，Key Administrator から開示が許可された索引階層数に応じて索引開示鍵を生成する．ここで生成した索引開示鍵を用いると，**ExtractTagIndex** アルゴリズムと **ExtractTrapdoorIndex** アルゴリズムを用いて，タグとトラップドアのそれぞれから索引情報を導出することができる．これを索引情報として用いることで，**Test** アルゴリズムを実行する前段階で，キーワードが一致する可能性があるタグを絞り込むことができ，検索時の **Test** アルゴリズムの実行回数を削減することができる．

ただし，検索が高速化される一方で，次のようなトレードオフが生じる．

1. 暗号化した索引情報が付加されるため，タグやトラップドアのサイズは最大索引階層数 L_{index} に比例して増加する．その増加分は，HIPE 暗号文約 L_{index} 個分であり，高速化される代わりにストレージを必要とする．

2. 索引開示鍵 $IKey_{\ell-1}$ が開示されている状況で新たに索引開示鍵索引開示鍵 $IKey_{\ell}$ を開示することで，保管されているすべてのタグに対して索引導出処理 **ExtractTagIndex** を実行する必要が生じる．この時の計算量は，保管されているタグ数 $\#TAG$ と $\mathbf{Dec_{HIPE}}$ アルゴリズムの計算量 $O(Dec_{HIPE})$ を用いて $\#TAG \times O(Dec_{HIPE})$ と表される．更に，データ保管時にタグから索引情報を導出する処理も追加となる．増加する処理量は，開示されている索引開示鍵の個数を ℓ としたとき，HIPE の復号が ℓ 回分である．

3. 検索を実施する場合，受領したトラップドアから索引情報を導出する処理 **ExtractTrapdoorIndex** が追加で必要となる．タグをデータベースに追加するときと同様に，増加する処理量は，開示されている索引開示鍵の個数を ℓ としたとき，HIPE の復号が ℓ 回分である．

そのため，システム適用時には最大データ量の見積もりを行い，索引階層 L_{index} をむやみに大きくしないことが必要となる．なお，**Index** 関数の選び方については，4.5.3 にて再考する．

また，**EncodeAttribute** アルゴリズムは 2 要素の属性 $(attr_1, attr_2)$ を 4 次元ベクトル $(\sigma_1 attr_1, \sigma_1, \sigma_2 attr_2, \sigma_2)$ に，**EncodePredicate** アルゴリズムは 2 要素の述語 $(pred_1, pred_2)$ を 4 次元ベクトル $(\rho_1, -\rho_1 pred_1, \rho_2, -\rho_2 pred_2)$ にベクトル化する関数である．内積述語暗号の復号が成功するのは両ベクトルの内積値が 0 となる場合だけ，すわなち $\sigma_1\rho_1(attr_1 - pred_1) + \sigma_2\rho_2(attr_2 - pred_2) = 0$ が成立するときである．この式が成り立つのは，$attr_1 = pred_1, attr_2 = pred_2$ の時であり，属性と述語が同一の時である．

Theorem 3. シングルユーザ版の基本方式は，*Definition 22* を満たす．

Proof. 上記を元に，Correctness condition 1 について述べる．**Test** アルゴリズムは，**GenTag** アルゴリズムにて属性 $('SE', w)$ にて HIPE で暗号化された乱数 r を，**GenTrapdoor** アルゴリズム内で述語 $('SE', w)$ で生成された秘密鍵にて復号するため，その復号結果は乱数 r となる．この復号結果をタグ内に暗号化されずに添付された乱数 r と同一かどうかを比較し，結果として $True(=1)$ を返す．そのため，Correctness condition 1 が成り立つ．

次に Correctness condition 2 について述べる．**GenTag** アルゴリズムと **GenTrapdoor** アルゴリズムの両方とも，**Index** 関数にて生成したキーワード w に関する索引情報を，それぞれ属性 $('IdxTag', i)$ および $('IdxTd', i)$ で暗号化している．**GenIndexKey** アルゴリズムでは，述語 $('IdxTag', i)$ および $('IdxTd', i)$ を用いて秘密鍵を生成する．**ExtractTagIndex** アルゴリズムおよび **ExtractTrapdoorIndex** アルゴリズムは，それぞれタグおよびトラップドア内に含まれる暗号化された索引情報を秘密鍵にて復号するアルゴリズムであり，同じ復号結果 (**Index** アルゴリズムの出力) を返す．そのため，Correctness condition 2 も成り立つ．

次に Correctness condition 3 について述べる．キーワード w, w' の索引値をそれぞれ $Idx_w, Idx_{w'}$ とする．仮定より，キーワード w, w' は ℓ-th level まで同一の索引値 Idx となる．Correctness condition2 で示した様に，本構成方法では索引値を固定の属性 $\{'IdxTag', i\}$ で暗号化し，固定の述語 $\{'IdxTag', i\}$ で復号するだけである．そのため，HIPE が Correctness condition を満たすなら，**ExtractTagIndex** アルゴリズムの出力は両者ともに同一の索引値 Idx となる．すなわち，Correctness condition 3 も成り立つ． □

Theorem 4. シングルユーザ版の基本方式は，*Definition 23* を満たす．

Proof. 始めに，consistency condition1 について述べる．本方式は Abdalla ら [1] の変換方式を利用して，IND-CPA を満たす HIPE から PEKS を構築している．そのため，タグとトラップドアの一致判定処理については，Abdalla らが示したように computationally consistent となる．証明方法は Abdalla らの論文と同様であるため，そのアウトラインのみ述べる．本方式の consistency に対する多項式時間攻撃者 $\mathcal{U}$ が存在すると仮定し，HIPE の IND-CPA 多項式時間攻撃者 $\mathcal{A}$ を次のように構成する．まず，攻撃者 $\mathcal{A}$ は，攻撃者 $\mathcal{U}$ を呼び出し，2 つのキーワード w, w' を取得する．そして，ランダムに選んだメッセージ R_0, R_1 とともに，キーワード w をベクトル化した $\{'SE', 1, w, 1\}$ を属性ベクトルとして Challenger に投げ，チャレンジ暗号文 C_b を得る．このチャレンジ暗号文は，R_b を属性ベクトル $\{'SE', 1, w, 1\}$ で暗号化したものである．それと同時に，チャレンジャーは，キーワード w' をベクトル化した $\{1, -'SE', 1, -w'\}$ を述語ベクトルとして，秘密鍵生成オラクルから秘密鍵 $sk_{\{1,-'SE',1,-w'\}}$ を取得する．もし $\mathbf{Dec}_{HIPE}(sk_{\{1,-'SE',1,-w'\}}, C_b) = R_1$ なら，チャレンジャーは 1 を返し，そうでなければ 0 を返す．攻撃者 $\mathcal{U}$ が本方式の consistency を破っていた場合，チャレンジ暗号文は正しく復号されるので，その結果から b が推定できる．これは，HIPE が IND-CPA を満たすことに矛盾するため，本方式は computationally consistent である．

次に Consistency condition 2 について述べる．キーワード w, w' の索引値をそれぞれ $Idx_w, Idx_{w'}$ とする．仮定より，キーワード w, w' は ℓ 階層目で異なる索引値を持つため，$Idx_w \neq Idx_{w'}$ である．Correctness condition2 で示した様に，本構成方法では索引値を固定の属性 $\{'IdxTag', i\}$ で暗号化し，固定の述語 $\{'IdxTag', i\}$ で復号するだけである．そのため，HIPE が Correctness condition を満たすなら，**ExtractTagIndex** と **ExtractTrapdoorIndex** の出力は，それぞれ $Idx_w, Idx_{w'}$ である．すなわち，Consistency condition 2 が成り立つ．

最後に Consistency condition3 について述べる．考え方は Consistency condition2 と同様であるが，キーワード w, w' の索引値をそれぞれ $Idx_w, Idx_{w'}$ とする．キーワード w, w' は ℓ 階層目で異なる索引値を持つため，**ExtractTagIndex** の出力は $Idx_w \neq Idx_{w'}$ である．Correctness condition3 で示した方法と同様に，本構成方法は両索引値を同じ属性で暗号化し，同じ述語で復号するだけである．そのため，そのため，HIPE が Correctness condition を満たすなら，Consistency condition3 を満たす． □

4.3.5 安全性証明

前節で示した安全性定義に基づき，4.3.4 節で示した方式が開示した索引情報以上の情報を攻撃者に与えないことを示す．

Theorem 5. 最大索引階層数 L_{index} は任意の定数とする．提案方式であるシングルユーザ版の索引生成対応公開鍵検索可能暗号は，*attribute-hiding* および *payload-hiding* を満たす内積述語暗号 *(HIPE)* が存在するなら，*CPA* 攻撃者に対して *adaptively secure* である．あらゆる確率的多項式時間攻撃者 $\mathcal{A}$ に対して，攻撃者 $\mathcal{A}$ のアドバンテージは下記の式にて表され，*negligible* となる．

$$\mathbf{Adv}^{SE}_{\mathcal{A}}(\lambda) \leq L_{index}\mathbf{Adv}^{HIPE,PH}_{\mathcal{B}}(\lambda) + \mathbf{Adv}^{HIPE,AH}_{\mathcal{B}}(\lambda)$$

Proof. 本定理を証明するにあたって，下記に示すゲーム列を考える．

Game0: オリジナルのゲーム．チャレンジメッセージは下記のように作られる．

$r \xleftarrow{U} M_{HIPE}$,
$c_{SE} := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'SE'}, w_b\}), r)$,
$Tag_{SE} := (r, c_{SE})$
For all $i \in \{1, \ldots, L_{index}\}$,
$EncIdxTag_i := \mathbf{Enc}_{HIPE}(mpk_{HIPE},$
$\mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'IdxTag'}, i\}),$
$\mathbf{Index}(w_b, i))$

Game1-1: Game0 と比べて L_{index} 層目の暗号化索引の生成方法のみ異なる．

$r \xleftarrow{U} M_{HIPE}$,
$c_{SE} := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'SE'}, w_b\}), r)$,
$Tag_{SE} := (r, c_{SE})$
For all $i \in \{1, \ldots, L_{index} - 1\}$,
$EncIdxTag_i := \mathbf{Enc}_{HIPE}(mpk_{HIPE},$
$\mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'IdxTag'}, i\}),$
$\mathbf{Index}(w_b, i))$
$\boxed{R_{L_{index}} \xleftarrow{U} M_{HIPE}}$,

$$EncIdxTag_{L_{index}} := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \\ \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'IdxTag'}, L_{index}\}), \\ \boxed{R_{L_{index}}})$$

Game1-k ($k = 1, \cdots, L_{index} - \ell$)**:** Game1-(k-1) と比べて ($L_{index} - (k-1)$) 層目の暗号化索引の生成方法のみ異なる．

$r \xleftarrow{U} M_{HIPE}$,

$c_{SE} := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'SE'}, w_b\}), r)$,

$Tag_{SE} := (r, c_{SE})$

For all $i \in \{1, \ldots, L_{index} - k\}$,
$EncIdxTag_i := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \\ \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'IdxTag'}, i\}), \\ \mathbf{Index}(w_b, i))$

For all $\boxed{i \in \{L_{index} - (k-1), \ldots, L_{index}\}}$,
$\boxed{R_i \xleftarrow{U} M_{HIPE}}$,
$EncIdxTag_i := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \\ \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'IdxTag'}, i\}), \boxed{R_i})$

Game2: Game1-($L_{index} - \ell$) と比べて SE パートの属性ベクトルの作成方法のみ異なる．

$r \xleftarrow{U} M_{HIPE}$,

$\boxed{R_w \xleftarrow{U} \mathbb{F}_q}$,

$c_{SE} := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'SE'}, \boxed{R_w}\}), r)$,

$Tag_{SE} := (r, c_{SE})$

For all $i \in \{1, \ldots, L_{index} - \ell\}$,
$EncIdxTag_i := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \\ \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'IdxTag'}, i\}), \\ \mathbf{Index}(w_b, i))$

For all $i \in \{L_{index} - (\ell+1), L_{index}\}$,
$R_i \xleftarrow{U} M_{HIPE}$,
$EncIdxTag_i := \mathbf{Enc}_{HIPE}(mpk_{HIPE}, \\ \mathbf{EncodeAttribute}(mpk_{HIPE}, \{\text{'IdxTag'}, i\}), R_i)$

上記ゲームにおいて，ℓ は攻撃者 $\mathcal{A}$ が適応的に呼び出しを実施した索引開示鍵の開示階層数である．また，$\mathbf{Adv}^{(0)}_{\mathcal{A}}(\lambda)$，$\mathbf{Adv}^{(1-k)}_{\mathcal{A}}(\lambda)$，$\mathbf{Adv}^{(2)}_{\mathcal{A}}(\lambda)$ は，Game0，Game1 $- k$，Game2 における攻撃者 $\mathcal{A}$ のアドバンテージ $\mathbf{Adv}^{SE}_{\mathcal{A}}(\lambda)$ である．Game2 においては，攻撃者 $\mathcal{A}$ がキーワードを識別するための情報が含まれていないため，明らかに $\mathbf{Adv}^{(2)}_{\mathcal{A}}(\lambda) = 0$ である．

最初に，Game0 と Game1 $-$ 1 におけるアドバンテージの違いについて説明する．ゲーム列からわかるように，ゲームの違いは L_{index} 層目の暗号化索引 $EncIdxTag_{L_{index}}$ である．HIPE が payload-hiding を満たすなら，攻撃者は Game0 と Game1 $-$ 1 の暗号化索引

$EncIdxTag_{L_{index}}$ の分布の違いを識別することはできない．そのため，2 つのゲームのギャップは下記の様に評価できる．

$$|\mathbf{Adv}_{\mathcal{A}}^{(0)}(\lambda) - \mathbf{Adv}_{\mathcal{A}}^{(1-1)}(\lambda)| \leq \mathbf{Adv}_{\mathcal{B}}^{HIPE,PH}(\lambda)$$

同様にして，Game1 $-\ k$ と Game1 $-\ (k+1)$ のアドバンテージの違いも下記の様に評価できる．

$$|\mathbf{Adv}_{\mathcal{A}}^{(1-k)}(\lambda) - \mathbf{Adv}_{\mathcal{A}}^{(1-(k+1))}(\lambda)| \leq \mathbf{Adv}_{\mathcal{B}}^{HIPE,PH}(\lambda)$$

最後の Game1 $-\ (L_{index} - \ell)$ と Game2 の違いについてもこれまでの議論と同様であるが，ゲームとしての違いは c_{SE} を作成する際の属性が異なっている点である．HIPE が attribute-hiding を満たすなら，攻撃者は Game1 $-\ (L_{index} - \ell)$ と Game2 における c_{SE} の分布の違いを識別することはできない．そのため，2 つのゲームのギャップは下記の様に評価できる．

$$|\mathbf{Adv}_{\mathcal{A}}^{(1-(L_{index}-\ell))}(\lambda) - \mathbf{Adv}_{\mathcal{A}}^{(2)}(\lambda)| \leq \mathbf{Adv}_{\mathcal{B}}^{HIPE,AH}(\lambda)$$

以上により下記の結果を得る．

$$\begin{aligned}
\mathbf{Adv}_{\mathcal{A}}^{SE}(\lambda) \quad \leq \quad & |\mathbf{Adv}_{\mathcal{A}}^{(0)}(\lambda) - \mathbf{Adv}_{\mathcal{A}}^{(1-1)}(\lambda)| \\
& + |\mathbf{Adv}_{\mathcal{A}}^{(1-1)}(\lambda) - \mathbf{Adv}_{\mathcal{A}}^{(1-2)}(\lambda)| \\
& \cdots \\
& + |\mathbf{Adv}_{\mathcal{A}}^{(1-(L_{index}-(\ell-1)))}(\lambda) - \mathbf{Adv}_{\mathcal{A}}^{(1-(L_{index}-\ell))}(\lambda)| \\
& + |\mathbf{Adv}_{\mathcal{A}}^{(1-(L_{index}-\ell))}(\lambda) - \mathbf{Adv}_{\mathcal{A}}^{(2)}(\lambda)| \\
& + |\mathbf{Adv}_{\mathcal{A}}^{(2)}(\lambda)| \\
\leq \quad & L_{index}\mathbf{Adv}_{\mathcal{B}}^{HIPE,PH}(\lambda) + \mathbf{Adv}_{\mathcal{B}}^{HIPE,AH}(\lambda).
\end{aligned}$$

以上により定理が証明できる． □

4.4 マルチユーザへの拡張

次に，シングルユーザ版の基本方式を，マルチユーザ版に対応した拡張方式にすることで，要件 A と要件 B を満たすようにする．

4.4.1 マルチユーザへの拡張のアイデア

マルチユーザ化を図る基本的なアイデアは，ワイルドカードに対応した階層型 ID を我々の提案方式に組み込むことである．この時，ワイルドカード対応の階層型 ID を内積述語暗号化扱うベクトルに変換する手法を検討する必要がある．我々のアイデアは，暗号化時には階層型 ID$\{ID_1, \ldots, ID_{L_{ID}}\}$ を HIPE のベクトル $(ID_1, 1, \ldots, ID_{L_{ID}}, 1)$ に変換する．また，トラップドアや鍵生成時には，階層型 ID$\{ID_1, \ldots, ID_{L_{ID}}\}$ を HIPE のベクトル $(1, -ID_1, \ldots, 1, -ID_{L_{ID}})$ に変換する．この内積が 0 になるのは両者の階層型 ID が同一である時のみであり，階層型 ID によるアクセス制御を実現できる．また，ID としてワイルドカードをサポートするために，ワイルドカードが指定された時にはベクトル $(0, 0)$ に変換するようにした．ゼロベクトルは

いかなるベクトルとも内積値が 0 となるため，いかなる ID ともマッチするワイルドカードの機能を実現することができる．例えば，$('Adept.', *)$ という階層型 ID は，$('Adept.','UserA')$ や $('Adept.','UserB')$ という階層型 ID と同一視できることとなり，'A Dept.' に属するユーザであれば誰でも検索ができる様になり，マルチユーザへの拡張が実現できる．

4.4.2 アルゴリズムの定義

アルゴリズムの定義を，ワイルドカードに対応した階層型 ID をサポートできるように拡張する．

Definition 25. ユーザ ID は，$\{ID_1, \ldots, ID_{L_{ID}}\} \in (\mathbb{F}_q \setminus \{0, q-1\})^{L_{ID}}$ にて表現する．マルチユーザ版の索引生成対応公開鍵検索可能暗号は，以下に示す確率的多項式時間アルゴリズム **Setup**，**GenUserKey**，**GenTag**，**GenTrapdoor**，**Test**，**GenIndexKey**，**ExtractTagIndex**，**ExtractTrapdoorIndex** から構成される．その定義は次のように与えられる．

Setup: セキュリティパラメータ 1^λ，最大 ID 階層数 L_{ID}，最大索引ビット数 L_{index} を入力とし，マスター公開鍵 mpk とマスター秘密鍵 msk のペアを出力

GenUserKey: マスター公開鍵 mpk，マスター秘密鍵 msk，ユーザ $ID : ID_U$ を受け取り，ユーザ秘密鍵 sk_U を出力

GenTag: マスター公開鍵 mpk，受信者ユーザ $ID : ID_R$，キーワード空間 W に含まれるキーワード w を受け取り，タグ Tag_w を出力

GenTrapdoor: マスター公開鍵 mpk，ユーザ秘密鍵 sk_U，検索キーワード w を受け取り，トラップドア Td_w を出力

Test: マスター公開鍵 mpk，タグ Tag_w，トラップドア Td_w を受け取り，タグとトラップドアの一致判定結果（*1:*一致，*0:*不一致）を出力．なお，一致と判定されるのはタグ生成時に用いられたキーワードと，トラプドア生成時に用いられたキーワードが一致していることを意味する．

GenIndexKey: マスター公開鍵 mpk，マスター秘密鍵 msk，受信者ユーザ $ID : ID_R$，生成を許可する索引ビット数 ℓ を受け取り，索引開示鍵 $IKey_{R,\ell}$ を出力

ExtractTagIndex: マスター公開鍵 mpk，タグ Tag_w，索引開示鍵 $IKey_{R,\ell}$ を受け取り，索引値 $\{0,1\}^\ell$ を出力（もしくはエラーを返す）

ExtractTrapdoorIndex: マスター公開鍵 mpk，トラップドア Td_w，索引開示鍵 $IKey_{R,\ell}$ を受け取り，索引値 $\{0,1\}^l$ を出力（もしくはエラーを返す）

アルゴリズムの定義は，Definition 21 似て示したシングルユーザ版とほぼ同じであり，ユーザ ID として階層型 ID をサポートしている点だけが異なる．これを実現するため，各ユーザに対して検索鍵を生成するための鍵生成アルゴリズム **GenUserKey** を新しく定義している．また，暗号化の際は，誰が復号できるかを示す受信者ユーザ ID を引数として追加している．Correctness conditions や Consistency conditions もシングルユーザ版と同じ考えであり，階層型 ID をサポートしている点が追加されている．

4.4.3 セキュリティの定義

4.4.2 節で示したマルチユーザにおける安全性定義は，下記のようにシングルユーザ版の定義を拡張して定義できる．

Definition 26. 最大索引階層数 L_{index} は任意の定数とする．最大索引階層数 L_{index} である索引生成対応公開鍵検索可能暗号が，*CPA* 攻撃者に対して *adaptively secure* であるとは，以下のゲームを実施する全ての確率的多項式時間攻撃者 $\mathcal{A}$ のアドバンテージがセキュリティパラメータ 1^λ に対して *negligible* であることである．

1. チャレンジャーは **Setup** を実行し，マスター公開鍵 mpk とマスター秘密鍵 msk のペアを生成する．マスター公開鍵 mpk は攻撃者 $\mathcal{A}$ に与える．
2. 攻撃者 $\mathcal{A}$ は適用的に多項式回だけ下記に示すクエリーを実行できる．
   - *User key query:* 攻撃者 $\mathcal{A}$ は任意のユーザ $ID : ID_U$ に対する検索鍵の生成を要求する．チャレンジャーはユーザ $ID : ID_U$ に対応する検索鍵 sk_U を生成し，これを攻撃者 $\mathcal{A}$ に送付する．
   - *Trapdoor query:* 攻撃者 $\mathcal{A}$ は任意のユーザ $ID : ID_U$ と任意のキーワード $w \in W$ に対するトラップドアの生成を要求する．チャレンジャーはキーワード w に対応するトラップドア Td_w を生成し，これを攻撃者 $\mathcal{A}$ に送付する．
   - *Index key query:* 攻撃者 $\mathcal{A}$ は任意の受信者ユーザ $ID : ID_R$ と任意の索引階層数 $\ell \leq L_{index}$ に対する索引開示鍵 $IKey_{R,\ell}$ の生成を要求する．チャレンジャーは索引階層数 ℓ に対応する索引開示鍵 $IKey_{R,\ell}$ を生成し，これを攻撃者 $\mathcal{A}$ に送付する．
3. 攻撃者 $\mathcal{A}$ はチャレンジキーワード $w_0^*, w_1^* \in W$ と受信者ユーザ $ID : ID_R^*$ を選び，チャレンジャーに送付する．なお制約として，*User key query* で取得した検索鍵，*Trapdoor query* で取得したトラップドア，*Index key query* で取得した索引開示鍵を使ってチャレンジキーワードが識別できる様なチャレンジキーワードを指定することはできない．
4. チャレンジャーはランダムにビット b を選び，攻撃者 $\mathcal{A}$ に対してチャレンジ暗号文 $Tag_{w_b^*} := \mathbf{GenTag}(mpk, ID_R^*, w_b^*)$ を与える．
5. 攻撃者 $\mathcal{A}$ は，ステップ *2* と同様に繰り返しクエリーを実行することができる．なお，チャレンジキーワード文が識別できる様な受信者ユーザ $ID : ID_R$ やキーワード w を指定して *User key query* や *Trapdoor query* を行うことはできない．同様に，チャレンジキーワード文が識別できる様な受信者ユーザ $ID : ID_R$ や索引階層数 ℓ を指定して *Index key query* を行うことはできない．
6. 攻撃者 $\mathcal{A}$ はビット b' を出力する．もし $b' = b$ であれば攻撃者の勝ちである．

上記ゲームにおいて，攻撃者 $\mathcal{A}$ のアドバンテージは次のように定義される．

$$\mathbf{Adv}_{\mathcal{A}}^{SE}(\lambda) = |\Pr[b' = b] - 1/2|$$

本方式では，階層型 ID をサポートしたため，攻撃者がチャレンジキーワード w_0^*, w_1^* を指定する際にチャレンジ IDID_R^* も指定できるように拡張している．また索引開示鍵クエリーを実行する際も，同様に受信者 ID を指定するように拡張した．更に，攻撃者は任意の ID を指定して検索鍵をクエリーできる様にした．ただし，自明な攻撃を防ぐため，チャレンジキーワードを区別できるようになる ID を指定して検索鍵をクエリーできないという制約を付けている．

4.4.4 構成方法

マルチユーザへ拡張したアルゴリズムを，下記に示す．

Setup$(1^\lambda, L_{ID}, L_{index}) \rightarrow (mpk, msk)$

$\overrightarrow{\mu}_{HIPE} := (n = 2d, d = L_{ID} + 2; \mu_1 = 2, \ldots, \mu_d = n)$,
$(mpk_{HIPE}, msk_{HIPE}) := \mathbf{Setup}_{HIPE}(1^\lambda, \overrightarrow{\mu}_{HIPE})$
choose suitable index generation function,
　Index$(w \in W, \ell \in \{1, \ldots, L_{index}\}) \rightarrow \{0,1\}^*$,
　which returns an ℓ-th index value.
$mpk := (mpk_{HIPE}, L_{ID}, L_{index}, \mathbf{Index}(,))$, $msk := (msk_{HIPE})$
return (mpk, msk)

GenUserKey$(mpk, msk, ID_U) \rightarrow sk_U$

$sk_{SE} := \mathbf{KeyGen}_{HIPE}(mpk_{HIPE}, msk_{HIPE},$
　　$\mathbf{EncodePredicate}(mpk_{HIPE}, 1, L_{ID}+1, \text{'SE'}|ID_U))$
return $sk_U := (sk_{SE}, ID_U)$

GenTag$(mpk, ID_R, w) \rightarrow Tag_w$

$r \xleftarrow{U} M_{HIPE}$,
$c_{SE} := \mathbf{Enc}_{HIPE}(mpk_{HIPE},$
　　$\mathbf{EncodeAttribute}(mpk_{HIPE}, 1, L_{ID}+2, \text{'SE'}|ID_R|w), r)$,
$Tag_{SE} := (r, c_{SE})$
For all $i \in \{1, \ldots, L_{index}\}$,
　$EncIdxTag_i :=$
　　$\mathbf{Enc}_{HIPE}(mpk_{HIPE},$
　　$\mathbf{EncodeAttribute}(mpk_{HIPE}, 1, L_{ID}+2, \{\text{'IdxTag'}, ID_R, i\})$,
　　$\mathbf{Index}(w, i))$
return $Tag_w := (Tag_{SE}, \{EncIdxTag_i\}_{L_{index}})$

GenTrapdoor$(mpk, sk_U, w) \rightarrow Td_w$

$Td_{SE} := \mathbf{Delegate}(mpk_{HIPE}, sk_{SE}, \mathbf{EncodePredicate}(mpk_{HIPE}, L_{ID}+2, 1, w))$,

For all $i \in \{1, \ldots, L_{index}\}$,
$EncIdxTd_i :=$
$\mathbf{Enc}_{HIPE}(mpk_{HIPE},$
$\mathbf{EncodeAttribute}(mpk_{HIPE}, 1, L_{ID}+2, \{\text{'IdxTd'}, ID_U, i\}),$
$\mathbf{Index}(w, i))$
return $Td_w := (Td_{SE}, \{EncIdxTd_i\}_{L_{index}})$

$\mathbf{Test}(mpk, Tag_w, Td_w) \to \{TRUE, FALSE\}$

$m' := \mathbf{Dec}(mpk_{HIPE}, Td_{SE}, c_{SE})$
If $m' = r$, $result =$ TRUE, otherwise $result =$ FALSE
return $result$

$\mathbf{GenIndexKey}(mpk, msk, ID_R, \ell) \to IKey_{R,\ell}$

For all $i \in \{1, \ldots, \ell\}$,
$IKTag_{R,i} :=$
$\mathbf{KeyGen}(mpk_{HIPE}, msk_{HIPE},$
$\mathbf{EncodePredicate}(mpk_{HIPE}, 1, L_{ID}+2, \{\text{'IdxTag'}, ID_R, i\})),$
$IKTd_{R,i} :=$
$\mathbf{KeyGen}(mpk_{HIPE}, msk_{HIPE},$
$\mathbf{EncodePredicate}(mpk_{HIPE}, 1, L_{ID}+2, \{\text{'IdxTd'}, ID_R, i\}))$
return $IKey_{R,\ell} := (\{IKTag_{R,i}\}_\ell, \{IKTd_{R,i}\}_\ell)$

$\mathbf{ExtractTagIndex}(mpk, Tag_w, IKey_{R,\ell}) \to \{0,1\}^*$

For all $i \in \{1, \ldots, \ell\}$,
$IdxTag_{R,i} := \mathbf{Dec}(mpk_{HIPE}, IKTag_{R,i}, EncIdxTag_i)$
return $result := IdxTag_1 | \ldots | IdxTag_\ell$

$\mathbf{ExtractTrapdoorIndex}(mpk, Td_w, IKey_{R,\ell}) \to \{0,1\}^*$

For all $i \in \{1, \ldots, \ell\}$,
$IdxTd_i := \mathbf{Dec}(mpk_{HIPE}, IKTd_{R,i}, EncIdxTd_i)$
return $result := IdxTd_1 | \ldots | IdxTd_\ell$

$\mathbf{EncodeAttribute}(pk, pos, len, attr) \to (\overrightarrow{x}_{pos}, \ldots, \overrightarrow{x}_{pos+(len-1)})$

For all $i \in \{pos, \ldots, pos + len - 1\}$,
$\sigma_i \xleftarrow{U} \mathbb{F}_q \setminus \{0\}$,
$\overrightarrow{x}_i = (0, 0)$ $(attr_i = *)$,
$\overrightarrow{x}_i = \sigma_i(q-1, 1)$ $(attr_i = *$, for EncIdxTag),
$\overrightarrow{x}_i = \sigma_i(attr_i, 1)$ (others)
return $(\overrightarrow{x}_{pos}, \ldots, \overrightarrow{x}_{pos+(len-1)})$

$\mathbf{EncodePredicate}(pk, pos, len, attr) \to (\overrightarrow{v}_{pos}, \ldots, \overrightarrow{v}_{pos+(len-1)})$

For all $i \in \{pos, \ldots, pos + len - 1\}$, $\quad \rho_i \xleftarrow{U} \mathbb{F}_q \setminus \{0\}$,
$\overrightarrow{v}_i = (0, 0)$ $(pred_i = *)$,
$\overrightarrow{v}_i = \sigma_i(1, -(q-1))$ $(pred_i = *$, for IKTag),
$\overrightarrow{v}_i = \rho_i(1, -pred_i)$ (others)
return $(\overrightarrow{v}_{pos}, \ldots, \overrightarrow{v}_{pos+(len-1)})$

詳細は割愛するが，1 つ目のポイントは，それぞれの関数に対して階層型 ID を指定できるようにしたことである．それぞれの関数では，**EncodeAttribute** 関数と **EncodePredicate** 関数を用いて階層型 ID を属性や述語に埋め込むことで，階層型 ID が一致している場合にのみ検索ができる様になる．これに伴い，ユーザに対して秘密鍵を発行する **GenUserKey** 関数も用意した．2 つ目のポイントは，階層型 ID の ID としてワイルドカード（$*$）を使えるようにしたことである．ワイルドカードが指定された際は，$(0, 0)$ というベクトルにエンコードするが，これはどのようなベクトルに対しても内積値 0 を与えるため，いかなる ID にもマッチすることとなる．例えば，$('Adept.', *)$ という階層型 ID は，$('Adept.', 'UserA')$ や $('Adept.', 'UserB')$ という階層型 ID と同一視できることとなり，'A Dept.' に属するユーザであれば誰でも検索ができる様になり，マルチユーザへの拡張が実現できる．Correctness に関しては多少複雑化するが，シングルユーザの場合と同様に検証することができる．

4.4.5 安全性証明

シングルユーザ版の基本方式と同様に，安全性について下記定理が成り立つ．

Theorem 6. 最大索引階層数 L_{index} は任意の定数とする．提案方式であるマルチユーザ版の索引生成対応公開鍵検索可能暗号は，*attribute-hiding* および *payload-hiding* を満たす内積述語暗号 *(HIPE)* が存在するなら，*CPA* 攻撃者に対して *adaptively secure* である．あらゆる確率的多項式時間攻撃者 $\mathcal{A}$ に対して，攻撃者 $\mathcal{A}$ のアドバンテージは下記の式にて表され，*negligible* となる．

$$\mathbf{Adv}^{SE}_{\mathcal{A}}(\lambda) \leq L_{index}\mathbf{Adv}^{HIPE,PH}_{\mathcal{B}}(\lambda) + \mathbf{Adv}^{HIPE,AH}_{\mathcal{B}}(\lambda)$$

証明については階層型 ID の概念が入るが，基本的にはシングルユーザ版と同一なため省略する．

4.5 実装および評価と考察

本節では，要件 C を満たすための実現方式について述べる．これまでに提案した拡張方式を適用することで，要件 A・要件 B・要件 C の全てを満たすことができる．また，提案方式の性能評価についても述べる．

4.5.1 データベースへの組込み

要件 A，B を満たす索引生成機能付き検索可能暗号を，要件 C も満たすようにデータベースに組込むための方式について述べる．

[Database schema] 検索可能暗号で暗号化したデータをデータベースに保管するため，データベースのスキーマ変更が必要となる．本提案方式では，タグおよび索引情報を保管することが必要であり，さらに暗号化データ自身も保管する必要がある．そこで，図 4.1 に示すように，データ 1 種類当たり 4 列のカラムを用意する．暗号化データ列は，マルチユーザサポートがなされた例えば HIPE などの通常の暗号化方式で暗号化されたデータを保管する列であり，検索された時の結果となる．暗号化タグ列は，検索可能暗号で暗号化されたデータが格納され，検索時に利用される．暗号化データ列，タグ列の両方とも，バイナリデータが保管されるため，バイナリデータ型となる．索引値列は，開示された索引情報を保管するための列であり文字列型とする．文字列型の列とすることで，データベースが持つ索引機能も働くことができ，高速に検索が可能となる．宛先 ID 列は，暗号化データを復号できるグループを示す情報であり，文字列型とする．なお，本実装では，データを通常の暗号と検索可能暗号とでそれぞれ暗号化し，それぞれ DB に保管している．そのため，この両者を結び付けているのは同じ行に保管されているかどうかだけで，暗号学的な関連付け話されていない．この両者を組み合わせて使用する場合に安全性を確保する仕組みも研究されており [23][74][82][98]，本提案方式と組合わせることで，より安全性を高めることができると考えられる．

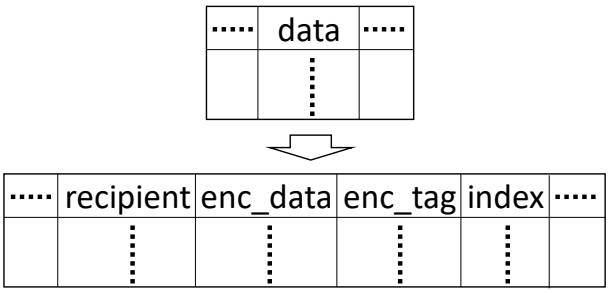

図 4.1: 暗号化データテーブルの構成

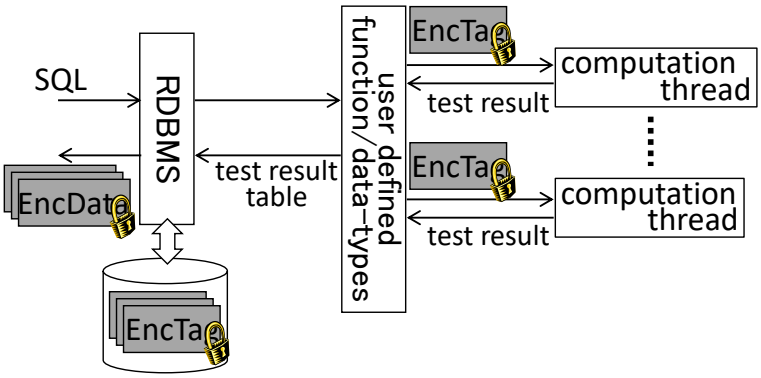

図 4.2: RDBMS への組込み方法と構成

なお，実際のテーブルには複数のカラムがあり，1 行が複数のデータによって構成されることが一般的であるが，本研究では 1 行をまとめて暗号化するのではなく，データごとに暗号化する仕組みとしている．これは，一般的な RDBMS ではカラム毎に索引を生成するた

め，カラム毎にデータを暗号化する方が親和性が高いこと，また検索可能暗号が完全一致しか対応できないためカラム毎に暗号化する必要があることなどに起因する．

また，実際のデータベースにはいろいろな情報が保管されるため，全てのデータを暗号化する必要がないケースも考えられる．そのため，テーブルに保管されたデータが漏洩した時のリスクを検討し，暗号化する列，平文のままにする列を使い分けることが好ましい．そうすることで，次の [検索手順] で述べるように性能劣化を押させることができる．また，例えばテーブルの Join などを実施したい場合は，その時のキーに使うデータは暗号化することは好ましくない．これは，暗号化すると常に異なる値となることから，テーブルの Join などは実現できなくなることによる．そのため，このようなデータについても平文のままとすることが好ましい．

[検索手順] 次に，データベースが検索されるときの処理について説明する．まずデータベースに含まれる非暗号化列を優先して検索処理し，可能な限り検索可能暗号の処理対象となる行を削減する．次に，トラップドアとともに検索者の ID 情報を受け取り，宛先列で検索者が検索できる行のみに絞り込む．その後，そのトラップドアで検索できる宛先のうち，最も索引情報が開示されている宛先の索引開示鍵を取り出し，トラップドアから索引情報を計算する．これにより，トラップドアから可能な限りの索引情報を取得することができる．このトラップドアから取り出した索引情報を用いて，データベースに保管された暗号化タグの索引値列を検索することで，検索対象となる行を更に絞り込む．なお，宛先ごとに開示されている索引情報のレベルが異なることから，索引情報は完全一致で検索するのではなく，短い方の索引情報に合わせて前方一致にて比較する．例えば，トラップドアの索引情報が”0011” の場合，索引値が”0011” だけでなく，”001” の行も検索対象として絞り込む．そして，絞り込んだ行に含まれる暗号化タグに対して，トラップドアを用いて検索可能暗号の判定処理 **Test** を行うことで，検索キーワードを含む行を特定することができるため，該当する暗号化データを検索結果として返す．これにより，索引情報を用いて処理対象データを削減しつつ，検索可能暗号でデータベース検索を行うことができる．

[索引導出手順] 次に，ユーザから索引開示鍵を受領した場合の処理について説明する．索引開示鍵を受領した場合は，同時にどの宛先に向けた索引開示鍵かも受領し，宛先列で該当する行に絞り込む．その後，それぞれの暗号化タグから索引情報を引き出し，該当行の索引値列に書き込む．索引値を更新している最中に検索が行われるケースもあるが，検索時の処理フローで述べたように，索引値は前方一致にて絞り込むことことから，更新中に検索されても対応可能である．

[ユーザ定義関数の実装] また，これらの処理をアプリケーション側にて実施するのでは，アプリケーション開発の負荷が高くなる．そこで，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とした．実現システムの構成を図 4.2 に示す．暗号化データやタグをデータベースに格納し，時間がかかる一致判定処理 (**Test** 関数) は複数スレッドもしくは複数計算ノードを用いて並行処理する．また，SQL にて検索可能暗号の処理に対応させるため，ユーザ定義関数やユーザ定義データ型の仕組みを用いて一致判定処理を実装する．これにより，暗号化した列を検索する場合に RDBMS から検索可能暗号が呼ばれるように SQL 文を記述でき，検索可能暗号独特の一致判定処理 (**Test** 関数) を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，自動的に複数スレッドを用いた並行処理を行うため，検索の高速化も可能となる．また，同様にして，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実装する．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，トラップドアから索引値を計算するために索引開示鍵などが必要となる．そこで，暗号アルゴリズム，公開パラメータ，索引開示鍵など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．

これら実装により，全ての要件 (A)，要件 (B)，要件 (C) を満たす公開鍵ベースの検索可能暗号が完成する．

4.5.2 鍵管理

本方式を安全に運用するためには，鍵管理が重要となる．本方式では，公開パラメータを全ユーザに配布したり，ユーザに対して秘密鍵を発行するために，企業アライアンスの幹事会社などが Key Administrator になる事を想定している．Key Administrator は，本方式のマスター秘密鍵を管理するとともに，マスター公開鍵を暗号化用の鍵としてアライアンス内に配布する．また，アライアンスに参加する企業，そのユーザの属性情報を管理し，ユーザに対して所属や参加するプロジェクト名などの情報を埋め込んだ秘密鍵を発行する．ユーザは，受け取ったマスター公開鍵で暗号化が行えるとともに，発行して貰った秘密鍵を用いて検索が可能となる．また，Key Administrator は索引開示鍵を発行する役割も担う．Key Administrator は保管されたデータの件数を定期的に監視，もしくはユーザからの申請によって，特定の宛先 ID_R に対する暗号化データの検索処理を高速化する必要があると判断した場合，索引開示鍵を発行し，データベースサーバに索引変更の指示を出す．これにより，key administrator がガバナンスを効かせて，ユーザの管理や索引レベルのコントロールを行う事ができる．

4.5.3 Index 関数の実装方式

本方式を安全に利用するためには，**Index** 関数の構成方法が重要となる．基本的なアイデアは，攻撃者が索引情報を見ても，暗号化タグに含まれるキーワードの候補が一定数未満に絞り込めない状況を作り出すことで，データの秘匿性を保証することである．これは，k-anonimity の考え方と類似している．k-anonimity は，プライバシー情報が含まれる表から，個人を特定する識別子を削除したり，複数項目を組み合わせることで個人特定につながる準識別子を汎化・削除するなどして，表のある行を見ても，対応する個人の候補を k 人以下に絞り込めない状況を作り出す．これによって，個人特定ができないように安全性を保ちつつ，データの有用性を生み出している．我々の方式も，**Index** で生成される索引情報を用いて，平文空間を複数グループに分割する．全てのグループのサイズが少なくとも k 個以上含まれれば，暗号化タグや索引値を見てもキーワードの候補が k 個以下に絞り込むことがで

きず，k-anonimity と同様の安全性を満たす．同時に，グループ分けを行う索引情報を活用することで，検索の高速化を達成することができる．

Index 関数の構成方法として，匿名化テーブルを作成する方法がある．これは，キーワードが市区町村のデータや郵便番号データであるなど，事前にキーワードの空間が定まっている場合に有用である．例えば，市区町村のデータを 4 ビットの索引値を使って保管したいと仮定すると，市区町村名空間を $2^4 = 16$ 個のグループに分け，それぞれのグループには同一個数の市区町村名を割り当てる．この時，4.3.1 節では分かりやすさのために東日本・西日本などエリアごとに区切る例を示したが，住んでいる地域が分かることによるリスクを避けるため，地域がばらけるようにグループに割り当てる．この 4 ビットのグループ ID と，グループに含まれる市区町村名を匿名化テーブルとして利用者間で共有する．**Index** 関数は，この匿名化テーブルを見て，市区町村名に対応する 4 ビットのグループ ID を 1 ビットずつ索引情報として出力するように構成する．市区町村名は 2015 時点で 1897 個ある [32] ため，索引として 1 ビットを開示したとしても，その候補は 948 個以下に絞り込むことができない (948-anonimity)．また，2 ビットの索引を開示したとしても，候補を 474 個以下に絞り込むことができない (474-anonimity)．これにより，万が一攻撃者が暗号化タグと索引情報を見て，暗号化データに対応する市区町村名を特定しようとしても，その候補は少なくとも k 個は存在するため k-匿名化が達成されており，市区町村名の特定を防ぐことができる．

匿名化テーブルはキーワード空間が事前に定まっている場合には有用であるが，必ずしもそうでない場合もある．その時の **Index** 関数の構成方法として，例えばハッシュ関数を用いてキーワードのハッシュを計算し，そのハッシュ値の一部ビットを索引として利用する方法が考えられる．ハッシュ関数のビットは一般的にランダムだと考えられていることから，キーワード空間を概ね等分に分割してくれると予想される．そこで，ハッシュアルゴリズム SHA-256 の先頭 4 ビットを用いて，市区町村名をグループに分割した結果を表 4.2 示す．

この実験から，ハッシュ関数を用いた索引生成にて，3 ビットを使って 8 グループに分割した場合はグループのサイズも概ね同一であり，グループに含まれる市区町村名の数は平均から 5.5% 以内に収まっている事が分かる．しかし，4 ビットを使って 16 グループに分割した場合は，平均から最大で 18.19% もグループに含まれる市区町村名数がばらつくことが分かった．ハッシュ関数はランダムな出力を生成するため 0 と 1 の出現確率は同等と考えられるが，各グループのキーワード数が多い場合は索引ビットで 0 と 1 の出現確率が 1/2 に収束していくと考えられるが，今回のケースでは各グループのキーワード数が 100 程度しかないため 1/2 に収束しきらなかったと考えられる．ただ，本実証の結果では，グループ要素数が 200 から 300 程度まではハッシュ値でグループ分けしても，平均からの誤差が数%程度でグループサイズが同一となることが分かった．一般的に，キーワード空間が事前い定義できないようなケースでは，キーワード空間のサイズは十分に大きいと予想される．例えば，広辞苑に掲載された見出し語数はおおよそ 25 万語であり，日本人の苗字であればおよそ 10 万種類 [24] と言われている．このようなケースでは，索引ビット数として 5 ビットから 10 ビットの範囲で利用できるものと考えられる．

キーワード空間は概ね均等に分割される可能性が分かったが，キーワードの出現頻度が同一とは限らない．例えば，市区町村ごとの人口のばらつきは非常に大きく，一番人口が多い地区では 903346 人，一番少ない地区では 0 人である．平均人口は 67377 人，中央値は 29638 人，標準偏差 98252 とバラつきが多いことが分かる．そこで，キーワードの出現頻度も考慮した場合に，索引情報毎の出現頻度のばらつきについても評価する．人口比率に応じてデー

表 4.2: 同一の索引値を持つキーワードの数

bit1	bit2	bit3	bit4
956 (bit1=0)	459 (bit2=0)	224 (bit3=0)	120 (bit4=0)
			104 (bit4=1)
		235 (bit3=1)	97 (bit4=0)
			138 (bit4=1)
	497 (bit2=1)	248 (bit3=0)	126 (bit4=0)
			122 (bit4=1)
		249 (bit3=1)	136 (bit4=0)
			113 (bit4=1)
941 (bit1=1)	470 (bit2=0)	230 (bit3=0)	101 (bit4=0)
			129 (bit4=1)
		240 (bit3=1)	131 (bit4=0)
			109 (bit4=1)
	471 (bit2=1)	237 (bit3=0)	105 (bit4=0)
			132 (bit4=1)
		234 (bit3=1)	120 (bit4=0)
			114 (bit4=1)

タベースにレコードが挿入された場合，各索引値ごとにデータの出現確率を示したのが表 4.3 である．

表を見ると，人口分布は非常に大きなばらつきがあったにも関わらず，各索引値の出現確率は平均化されていることが分かる．3 ビットの索引を使って 8 グループに分割した場合，各索引値の出現確率は 11.0%から 14.1%の範囲に平準化されていることが分かる．各市区町村が一様分布で現れると仮定した理想的な場合の出現確率が 12.5%であることから，今回の様に分布に大きな偏りがあるケースであっても，その出現確率は平準化され，索引情報の分布を見てもキーワードの類推が難しいと考えられる．また，索引値の出現確率が平準化されることから，開示した索引情報のビット数 n に応じて，検索性能が概ね 2^n に高速化できることが分かる．

4.5.4 性能評価

本提案手法の性能について評価した結果を述べる．提案方式を実現するにあたって，内積述語暗号として高島ら [53] によって提案された方式を用い，ペアリングとしては BN カーブ [7] を用いた．そのパラメータや計算アルゴリズムは文献 [4] によって提案された方式を用いた．提案手法は，C プログラムによって実装し，Amazon Linux AMI release 2018.03 上にて性能評価を行った．CPU リソースは T2.micro Xeon ファミリー（最大 3.3GHz, 64bit, single core），メモリ 1GB，データベースとして PostgreSQL 9.6.10 を利用した．データセットとしては郵便番号を用い，グループ名とユーザ名の 2 階層の ID を用いて暗号化した暗号化タグをデータベースに登録した．索引生成関数 **Index** としては，ハッシュ関数 SHA-256

表 4.3: 索引開示ビットに応じた索引出現頻度の測定結果

bit1	bit2	bit3	bit4
50.7% (bit1=0)	23.6% (bit2=0)	11.2% (bit3=0)	7.0% (bit4=0)
			4.2% (bit4=1)
		12.4% (bit3=1)	4.6% (bit4=0)
			7.9% (bit4=1)
	27.1% (bit2=1)	14.1% (bit3=0)	7.2% (bit4=0)
			6.9% (bit4=1)
		12.9% (bit3=1)	6.9% (bit4=0)
			6.1% (bit4=1)
49.3% (bit1=1)	23.4% (bit2=0)	11.0% (bit3=0)	4.8% (bit4=0)
			6.2% (bit4=1)
		12.4% (bit3=1)	6.3% (bit4=0)
			6.1% (bit4=1)
	25.9% (bit2=1)	13.1% (bit3=0)	6.0% (bit4=0)
			7.1% (bit4=1)
		12.8% (bit3=1)	6.3% (bit4=0)
			6.4% (bit4=1)

を用いた．登録された暗号化タグを，索引情報にて一時絞り込みを実施し，絞り込んだすべての行に対して検索可能暗号による一致判定を用いて，その時の SQL の応答時間を測定した．検索可能暗号の処理は 3 並列にて実行している．また，測定は 10 回実施し，その平均値を測定結果とした．性能測定結果を表 4.4 に示す．

測定結果の様に，索引開示無しのデータを見ると，登録されているデータ数が増えると，件数に比例して検索時間も増加する．例えば，5 千件の暗号化タグが登録されている場合は 11.71 秒の検索時間，1 万件の場合は 23.29 秒，5 万件の場合は 114.70 秒の検索時間となる．ここで索引情報 1 ビット開示することで，検索時間は約半分の 63.27 秒に高速化される．同様に，索引情報を 3 ビット，5 ビットと増加させることで，検索性能も 12.41 秒，2.31 秒と高速化される事が分かる．例えば，応答時間を数秒程度に抑えたいと考えた場合，データ件

表 4.4: 提案方式の検索性能測定結果

DB 上の	開示した索引ビット数				
データ件数	開示なし	1 bit	3 bit	5 bit	7bit
5,000	11.71[s]	5.31[s]	1.33[s]	0.73[s]	0.27[s]
10,000	23.29[s]	15.12[s]	2.74[s]	0.67[s]	0.36[s]
50,000	114.70[s]	63.27[s]	12.41[s]	2.31[s]	0.70[s]
100,000	232.73[s]	127.06[s]	30.16[s]	4.25[s]	0.73[s]

数が 1000 件や 2000 件の場合は索引情報を開示せず，データ件数が 5000 件程度に増えたら 1 ビットの索引情報を開示して高速化を図り，さらにデータ件数が 1 万件に増えたら 3 ビット開示に変更し，さらにデータ件数が 5 万件に増えたら 5 ビットを開示に変更する，という運用を行うことで，性能要求を満たす運用が可能となる．このようにセキュリティと検索性能のバランスを取ることができる．

4.6 本章のまとめ

本章では，公開鍵暗号に基づく検索可能暗号に対して，実用に向けた課題として (要件 A) データ件数に応じて検索性能を高速化する仕組み，(要件 B) 複数利用者でデータを共有する仕組み，(要件 C) データベースへの検索可能暗号の組み込み，の 3 点が重要であることを指摘した．そして，要件 A については，k-anonimity の考え方を応用した新しい安全性により，検索の高速化と安全性の確保を両立する仕組みについて提案した．そして，内積述語暗号を用いて実現するためのアルゴリズムを提案した．同様に要件 B についても，階層型 ID ベース暗号の考え方をベースに階層型 ID としてワイルドカードを用いる仕組み，および内積述語暗号のゼロベクトルを用いた実現アルゴリズムについて提案した．要件 C に関しては，データベースのユーザ定義関数の仕組みを用い，SQL にて検索可能暗号を利用できる様にする仕組みについて提案を行った．

第5章　まとめと今後の展望・課題

5.1　全体のまとめ

本論文では，企業内の広範囲な部署での情報共有，もしくは企業間をまたがっての情報共有を行う場合を想定し，データを暗号化で保護したまま情報共有を可能とする検索可能暗号データベースの実現に着目した．この検索可能暗号データベースでは，暗号化したまま検索を可能とする検索可能暗号と，暗号化したデータの復号権限の制御を利用者の属性を用いて実現する関数型暗号を組み合わせて利用する必要があるが，その際に下記の課題があることに着目した．

1. ユーザ秘密鍵失効の実現
   グループ共有機能を実現した検索可能暗号では，検索や復号が行えるユーザが複数人存在してしまうため，特定一人のユーザ秘密鍵だけを失効させることが困難である．そこで，暗号化されたデータを復号するためのユーザ秘密鍵を失効させる仕組みが課題である．

2. 検索性能の向上
   既存の検索可能暗号では，暗号化データから1ビットも情報が漏れないことを意味する識別不可能性という性質のおかげで，安全性は極めて高いもののデータベースの索引機能が全く機能させることができないため，データ件数に比例した検索処理が必要となる課題がある．そのため，検索性能を向上させるためには，データベースの索引機能が有効となるような検索可能暗号の実現が必要である．同時に，検索できるユーザが一人ではなく，検索権限を複数のユーザで共有できるグループ共有機能の実現や，実現した検索可能暗号の索引生成の仕組みが，一般的なデータベースへと容易に組み込みができる方式の実現も課題である．

これら2つの課題を解決するため，関数型暗号のユーザ秘密鍵を失効させる仕組み，および検索可能暗号の検索性能の向上について研究を行った．

まず3章では，関数型暗号の復号鍵の失効方式として，下記の要件を満たすことが重要であることを指摘した．

要件 1:　暗号化時に失効されたユーザが誰かを意識しないで良いこと

要件 2:　復号できていた暗号化データが復号できなくなること

要件 3:　失効に伴い，クラウド側の処理が不要もしくは軽微であること

要件 4:　失効していないユーザの復号鍵は従来通り使えること

そして，高島ら [65] が提案した関数型暗号において，復号鍵を失効させるための仕組みについて提案した．具体的には，プロキシー支援型のアプローチを採用し，プロキシーサーバとユーザが協調して復号処理を行うようにすることで，プロキシーサーバが管理するプロキシー鍵を削除するだけでユーザ秘密鍵を失効できる仕組みを実現した．これにより，暗号化する際は失効しているユーザが誰かを意識しなくてよいため要件 1 を満たし，さらに失効処理を行った時点で過去に復号できていた暗号化データをすぐに復号できないようにする失効が実現できるため要件 2 を満たす．また，クラウド側の処理が一切不要であることから要件 3 を満たし，失効していないユーザには全く影響が出ないことから要件 4 も満たす．また，関数型暗号の復号鍵に乱数を掛け合わせるというシンプルな手法で，プロキシー鍵とユーザ秘密鍵を生成するようにしたことで計算量の増加は極めて少ない．さらに，鍵ポリシー型と暗号文ポリシー型の双方の関数型暗号に対して失効機能を付与し，かつ adaptive-secure で安全性証明が付けられることも示した．

次に 4 章では，公開鍵暗号に基づいた検索可能暗号に対して，実用に向けて下記の 3 点が重要であることを指摘した．

要件 A: データ件数に応じて検索性能を高速化する仕組み

要件 B: 複数利用者でデータを共有する仕組み

要件 C: データベースへの検索可能暗号の組み込み

そして，要件 A については，k-anonimity の考え方を応用した新しい安全性により，検索の高速化と安全性の確保を両立するアイデアについて提案し，これを内積述語暗号を用いて実現するためのアルゴリズムを提案した．同様に要件 B についても，階層型 ID ベース暗号の考え方をベースに階層型 ID としてワイルドカードを用いる仕組み，および内積述語暗号のゼロベクトルを用いた実現アルゴリズムについて提案した．要件 C に関しては，データベースのユーザ定義関数の仕組みを用い，SQL にて検索可能暗号を利用できる様にする仕組みについて提案を行った．

以上が本研究の貢献である．

5.2 今後の展望と課題

従来の暗号技術は，データの機密を守ることに注力してきたが，様々な機器やセンサーで情報が生成され，それらを活用していくことを考えると，今後は単に機密を守るだけの暗号技術では不十分であり，様々な機能を持った暗号技術が必要になると考えている．その中で注目されている技術の一つが検索可能暗号や関数型暗号や属性ベース暗号などの高機能暗号である．本論文でも検索性能の向上やユーザ秘密鍵の失効に向けた方式を提案したが，これからも高機能暗号の実用化に向けた応用研究が盛んに実施され，実用化に向けて進んでいくと思われる．例えば，下記のような技術について引き続き研究が必要であると考えている．

- 検索鍵の失効方式
  本研究では検索結果として取得した暗号化データを復号するためのユーザ秘密鍵を失効させる仕組みを提案したが，可能であれば検索そのものが行えないようにするために，検索鍵を失効させる仕組みの実現が求められる．

現時点で安全性証明が付与可能な実現方式の目途はついていないが，本研究で検索可能暗号を実現するために利用した階層型内積述語暗号 [53] の安全性証明でも，$\overrightarrow{x} \cdot \overrightarrow{v} = 0$ になる場合，すなわちチャレンジ暗号文を復号可能な条件で秘密鍵がクエリーされた場合，正しく秘密鍵がシミュレートできないことが示されている．この条件が秘密鍵のシミュレーションを困難にしている理由を検証することで，本研究と同様に秘密鍵を分割するアプローチで解決できるかどうかを検討していきたい．

- キーワード空間のサイズ，求める安全性，索引値ビット数のバランスのとり方
  索引値を開示することで，暗号化データに含まれるキーワードの候補が絞り込まれることになる．すなわち，索引値の開示ビット数を増やせば検索性能が向上する一方で，その開示量に応じて安全性が低下することを意味する．この索引値として開示できる最大ビット数が，キーワード空間のサイズ，蓄積されるデータ件数，求められる安全性からどのように求められるかについての研究が必要となる．提案方式は，索引を開示した時の暗号化データの安全性として k-anonymity の考え方を導入したので，索引開示ビット数 ℓ に対して $k \approx |W|/(2^{\ell})$ であり，索引を開示していない場合にキーワードを類推される確率 $\Pr[w = w'] = 1/|W|$ に対して，ℓ ビットの索引値を開示した際にキーワードを類推される確率は $\Pr[w = w'] = 1/k \approx (2^{\ell})/|W|$ に向上する．この確率の上昇が，扱うデータの重要性に比べて許容できるかどうかで決定する方法が考えられる．しかし，匿名化 (プライバシー保護) の研究分野においても，この k の選び方に汎用的な手法が確立していない事が高橋ら [100] の講演でも指摘されている．更に，安全性と有用性のバランスを考慮すると，$k = 2, 3$ や $k = 5, 10$ で良いという説があると紹介されているが，これは主にデータ分析の結果に影響する (有用性が保てない) という理由により k が小さな値でも良いとされたものと推測しており，安全性の観点から k がいくつであるべきかは明確に判断ができない．

  別のアプローチとして，Dwork らによって差分プライバシーのアイデアが提案されている [31]．これは，値を開示したくないデータベースから統計値を計算・開示するケースにおいて，データベースの中身が追加・削除されたときに同じ統計値を計算・開示することで，追加・削除されたデータに関する情報が漏洩する可能性に着目し，統計値の差が十分に小さくなることを安全性の尺度とした方式である．k-anonymity とは異なる安全性の考え方であり，さらに統計値に注目しているため，我々の提案手法へ単純に適用することは難しいと考えているが，これらの分野で行われた議論についても調査を行い，索引値ビット数の選定に使える考え方がないかを検討していきたい．

  加えて，テーブルに複数の列が存在する時，それぞれの列の索引開示量の依存関係についての考察も必要である．これは例えば，住所，年齢，職業の 3 列があったとき，各列のデータで個人が特定できなかったとしても，3 列のデータを組み合わせて個人を特定できる可能性が高まることが k-anonymity の分野でも議論されている．そのため，索引の開示ビット数を考えるときは，列単位で開示ビット数を考えるだけでなく，組み合わせで考える必要も生じる可能性がある．これらの依存関係についても検討を行っていきたい．

- ユーザ秘密鍵や検索鍵の管理
  ユーザ秘密鍵や検索鍵は，端末側で安全に管理する必要が生じるが，人事異動などによって属性が変わるたびに再配布が必要という運用上の負荷もかかる．そのため，ディ

レクトリサービスと鍵生成機能の連携や，オンライン鍵配布などの運用上の負荷が少ない鍵管理方法の実現も求められる．

- ブラウザなどの既存アプリケーションでの実現方式
  最近では専用のアプリではなく，Web ブラウザを用いて様々なアプリが実現されることが多い．そのため，検索可能暗号や関数型暗号も Web ブラウザ上で実現するニーズが増えてくると予想する．ブラウザなどの既存アプリケーション上での実現方式が求められる．

参考文献

[1] Michel Abdalla, Mihir Bellare, Dario Catalano, Eike Kiltz, Tadayoshi Kohno, Tanja Lange, John Malone-Lee, Gregory Neven, Pascal Paillier, and Haixia Shi. Searchable encryption revisited: Consistency properties, relation to anonymous ibe and extensions. In Victor Shoup, editor, *CRYPTO 2005*, Vol. 3621 of *LNCS*, pp. 205–222. Springer, Heidelberg, 2005.

[2] Michel Abdalla, Dario Catalano, Alex Dent, Jorh Malone-Lee, Gregory Neven, and Nigel Smart. Identity-based encryption gone wild. In Michele Bugliesi, Bart Preneel, Vladimiro Sassone, and Ingo Wegener, editors, *ICALP 2006*, Vol. 4052 of *LNCS*, pp. 300–311. Springer, Heidelberg, 2006.

[3] Rakesh Agrawal, Jerry Kiernan, Ramakrishnan Srikant, and Yirong Xu. Order preserving encryption for numeric data. In *Proceedings of the 2004 ACM SIGMOD international conference on Management of data*, SIGMOD '04, pp. 563–574. ACM, 2004.

[4] Diego F. Aranha, Koray Karabina, Patrick Longa, Catherine H. Gebotys, and Julio López Hernandez. Faster explicit formulas for computing pairings over ordinary curves. In *EUROCRYPT 2011*, Vol. 6632 of *LNCS*, pp. 48–68. Springer, 2011.

[5] Nuttapong Attrapadung, Jun Furukawa, and Hideki Imai. Forward-secure and searchable broadcast encryption with short ciphertexts and private keys. In *ASIACRYPT 2006*, Vol. 4284 of *LNCS*, pp. 161–17. Springer, Heidelberg, 2006.

[6] Nuttapong Attrapadung and Hideki Imai. Attribute-based encryption supporting direct/indirect revocation modes. In Matthew G. Parker, editor, *Cryptography and Coding*, pp. 278–300, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[7] Paulo S. L. M. Barreto and Michael Naehrig. Pairing-friendly elliptic curves of prime order. In Bart Preneel and Stafford Tavares, editors, *Selected Areas in Cryptography*, pp. 319–331, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[8] Mihir Bellare, Alexandra Boldyreva, Anand Desai, and David Pointcheval. Key-privacy in public-key encryption. In Colin Boyd, editor, *Advances in Cryptology — ASIACRYPT 2001*, pp. 566–582, Berlin, Heidelberg, 2001. Springer Berlin Heidelberg.

[9] Mihir Bellare, Alexandra Boldyreva, and Adam O'Neill. Deterministic and efficiently searchable encryption. In Alfred Menezes, editor, *CRYPTO 2007*, Vol. 4622 of *LNCS*, pp. 535–552. Springer, Heidelberg, 2007.

[10] Mihir Bellare, Marc Fischlin, Adam O'Neill, and Thomas Ristenpart. Deterministic encryption: Definitional equivalences and constructions without random oracles. In David Wagner, editor, *Advances in Cryptology – CRYPTO 2008*, pp. 360–378, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[11] John Bethencourt, Amit Sahai, and Brent Waters. Ciphertext-policy attribute-based encryption. In *2007 IEEE Symposium on Security and Privacy (SP '07)*, pp. 321–334, 2007.

[12] Alexandra Boldyreva, Nathan Chenette, and O'Neill Adam. Order-preserving encryption revisited: Improved security analysis and alternative solutions. In Phillip Rogaway, editor, *Advances in Cryptology – CRYPTO 2011*, Vol. 6841 of *Lecture Notes in Computer Science*, pp. 578–595. Springer, 2011.

[13] Alexandra Boldyreva, Nathan Chenette, Younho Lee, and O'Neill Adam. Order-Preserving Symmetric Encryption. In Antoine Joux, editor, *Advances in Cryptology - EUROCRYPT 2009*, Vol. 5479 of *Lecture Notes in Computer Science*, pp. 224–241. Springer, 2009.

[14] Dan Boneh and Xavier Boyen. Efficient selective-id secure identity-based encryption without random oracles. In Christian Cachin and Jan L. Camenisch, editors, *EUROCRYPT 2004*, Vol. 3027 of *LNCS*, pp. 223–238. Springer, Heidelberg, 2004.

[15] Dan Boneh and Xavier Boyen. Secure identity based encryption without random oracles. In Matt Franklin, editor, *Advances in Cryptology – CRYPTO 2004*, pp. 443–459, Berlin, Heidelberg, 2004. Springer Berlin Heidelberg.

[16] Dan Boneh, Xavier Boyen, and Eu-Jin Goh. Hierarchical identity based encryption with constant size cipher. In Ronald Cramer, editor, *EUROCRYPT 2005*, Vol. 3494 of *LNCS*, pp. 440–456. Springer, Heidelberg, 2005.

[17] Dan Boneh, Giovanni Di Crescenzo, Rafail Ostrovsky, and Giuseppe Persiano. Public key encryption with keyword search. In Christian Cachin and Jan Camenisch, editors, *EUROCRYPT 2004*, Vol. 3027 of *LNCS*, pp. 506–522. Springer, Heidelberg, 2004.

[18] Dan Boneh and Matt Franklin. Identity-based encryption from the Weil pairing. In Joe Kilian, editor, *CRYPTO 2001*, Vol. 2139 of *LNCS*, pp. 213–229. Springer, Heidelberg, 2001.

[19] Dan Boneh and Brent Waters. Conjunctive, subset, and range queries on encrypted data. In Salil P. Vadhan, editor, *TCC 2007*, Vol. 4392 of *LNCS*, pp. 535–554. Springer, Heidelberg, 2007.

[20] Xavier Boyen and Brent Waters. Anonymous hierarchical identity-based encryption (without random oracles). In Cynthia Dwork, editor, *CRYPTO 2006*, Vol. 4117 of *LNCS*, pp. 290–307. Springer, Heidelberg, 2006.

[21] Melissa Chase and Sherman S.M. Chow. Improving privacy and security in multi-authority attribute-based encryption. In *Proceedings of the 16th ACM Conference on Computer and Communications Security*, CCS '09, p. 121–130, New York, NY, USA, 2009. Association for Computing Machinery.

[22] Sanjit Chatterjee and Palash Sarkar. Hibe with short public parameters without random oracle. In Xuejia Lai and Kefei Chen, editors, *Advances in Cryptology – ASIACRYPT 2006*, pp. 145–160, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[23] Yu Chen, Jiang Zhang, Dongdai Lin, and Zhenfeng Zhang. Generic constructions of integrated pke and peks. Vol. 78, pp. 493–526, 2016.

[24] Satoshi Chida and Shigeru Mase. Statistical analysis of japanese surnames. *Journal of the Japan Statistical Society*, Vol. 35, No. 1, pp. 55–70, sep 2005.

[25] D. Cooper, S. Santesson, S. Farrel, S. Boeyen, R. Housley, and W. Polk. Internet x.509 public key infrastructure certificate and certificate revocation list (crl) profile. RFC5280, 2008.

[26] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *13th ACM CCS*, pp. 79–88. ACM Press, New York, 2006.

[27] Reza Curtmola, Juan Garay, Seny Kamara, and Rafail Ostrovsky. Searchable symmetric encryption: Improved definitions and efficient constructions. In *In Journal of Computer Security 19(5)*, pp. 895–934. IOS Press, 2011.

[28] Angelo De Caro, Vincenzo Iovino, and Giuseppe Persiano. Hidden vector encryption fully secure against unrestricted queries. Cryptology ePrint Archive, Report 2011/546, 2011.

[29] T. Dierks and E. Rescorla. The transport layer security (tls) protocol version 1.3. RFC8446, 2018.

[30] W. Diffie and M. Hellman. New directions in cryptography. *IEEE Transactions on Information Theory*, Vol. 22, No. 6, pp. 644–654, 1976.

[31] Cynthia Dwork, Frank McSherry, Kobbi Nissim, and Adam Smith. Calibrating noise to sensitivity in private data analysis. In Shai Halevi and Tal Rabin, editors, *Theory of Cryptography*, pp. 265–284, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[32] e-Stat. `https://www.e-stat.go.jp/regional-statistics/ssdsview/municipality`.

[33] Taher ElGamal. A public key cryptosystem and a signature scheme based on discrete logarithms. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pp. 10–18, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[34] Keita Emura, Atsushi Takayasu, and Yohei Watanabe. Efficient identity-based encryption with hierarchical key-insulation from hibe. Vol. 89 of *Designs, Codes and Cryptography*, pp. 2397–2431, Berlin, Heidelberg, 2021. Springer Berlin Heidelberg.

[35] Kai Fan, Junxiong Wang, Xin Wang, and Yintang Yang. Proxy-assisted access control scheme of cloud data for smart cities. In *Personal and Ubiquitous Computing*, pp. 937–947. Springer Berlin Heidelberg, 2017.

[36] Jun Furukawa. Request-based comparable encryption. In Jason Crampton, Sushil Jajodia, and Keith Mayes, editors, *Computer Security – ESORICS 2013*, pp. 129–146, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[37] Craig Gentry. Practical identity-based encryption without random oracles. In Serge Vaudenay, editor, *Advances in Cryptology - EUROCRYPT 2006*, pp. 445–464, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[38] Craig Gentry and Shai Halevi. Hierarchical identity based encryption with polynomially many levels. In Omer Reingold, editor, *Theory of Cryptography*, pp. 437–456, Berlin, Heidelberg, 2009. Springer Berlin Heidelberg.

[39] Craig Gentry and Alice Silverberg. Hierarchical id-based cryptography. In Yuliang Zheng, editor, *ASIACRYPT 2002*, Vol. 2501 of *LNCS*, pp. 548–566. Springer, Heidelberg, 2002.

[40] Vipul Goyal, Abishek Jain, Omkant Pandey, and Amit Sahai. Bounded ciphertext policy attribute-based encryption. In *ICALP 2008*, Vol. 5126 of *LNCS*, pp. 579–591. Springer, Heidelberg, 2008.

[41] Vipul Goyal, Omkant Pandey, Amit Sahai, and Brent Waters. Attribute-based encryption for fine-grained access control of encrypted data. In *Proceedings of the 13th ACM Conference on Computer and Communications Security*, CCS '06, pp. 89—-98. Association for Computing Machinery, 2006.

[42] Matthew Green, Susan Hohenberger, and Brent Waters. Outsourcing the decryption of abe ciphertexts. *USENIX 2011*, 2011.

[43] Mitsuhiro Hattori, Takato Hirano, Takashi Ito, Nori Matsuda, Takumi Mori, Yusuke Sakai, and Kazuo Ohta. Ciphertext-policy delegatable hidden vector encryption and its application to searchable encryption in multi-user setting. In Liqun Chen, editor, *IMACC 2011*, Vol. 7089 of *LNCS*, pp. 190–209. Springer, Heidelberg, 2011.

[44] Caleb Horst, Ryo Kikuchi, and Keita Xagawa. Cryptanalysis of comparable encryption in sigmod ' 16. In *Proceedings of the 2017 ACM International Conference on*

Management of Data, SIGMOD '17, p. 1069–1084, New York, NY, USA, 2017. Association for Computing Machinery.

[45] Jeremy Horwitz and Ben Lynn. Towards hierarchical identity-based encryption. In Lars R. Knudsen, editor, *EUROCRYPT 2002*, Vol. 2332 of *LNCS*, pp. 466–481. Springer, Heidelberg, 2002.

[46] Ziyuan Hu, Shengli Liu, Kefei Chen, and Joseph K. Liu. Revocable identity-based encryption from the computational diffie-hellman problem. In *ACISP 2018*, Vol. 10946 of *LNCS*, pp. 265–283. Springer, Heidelberg, 2018.

[47] Yong Ho Hwang and Pil Joong Lee. Public key encryption with conjunctive keyword search and its extension to a multi-user system. In Tsuyoshi Takagi, Tatsuaki Okamoto, Eiji Okamoto, and Takeshi Okamoto, editors, *Pairing 2007*, Vol. 4575 of *LNCS*, pp. 2–22. Springer, Heidelberg, 2007.

[48] Jonathan Katz, Amit Sahai, and Brent Waters. Predicate encryption supporting disjunctions, polynomial equations, and inner products. In Nigel Smart, editor, *EUROCRYPT 2008*, Vol. 4965 of *LNCS*, pp. 146–162. Springer, Heidelberg, 2008.

[49] Neal Koblitz. Elliptic curve cryptosystems. In *Mathematics of Computation*, Vol. 48, pp. 203–209, 1987.

[50] Jianchang Lai, Yi Mu, Fuchun Guo, Willy Susilo, and Rongmao Chen. Anonymous identity-based broadcast encryption with revocation for file sharing. In *ACISP 2016*, Vol. 9723 of *LNCS*, pp. 223–239. Springer, Heidelberg, 2016.

[51] Kwangsu Lee. Revocable hierarchical identity-based encryption with adaptive security. Cryptology ePrint Archive, Report 2016/749, 2016.

[52] Kwangsu Lee, Seung Geol Choi, Dong Hoon Lee, Jong Hwan Park, and Moti Yung. Self-updatable encryption: Time constrained access control with hidden attributes and better efficiency. In Kazue Sako and Palash Sarkar, editors, *Advances in Cryptology - ASIACRYPT 2013*, pp. 235–254, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[53] Allison Lewko, Tatsuaki Okamoto, Amit Sahai, Katsuyuki Takashima, and Brent Waters. Fully secure functional encryption: Attribte-based encryption and (hierarchical) inner product encryption. In *EUROCRYPT 2010*, Vol. 6110 of *LNCS*, pp. 62–91. Springer, Heidelberg, 2010.

[54] Allison Lewko and Brent Waters. New techniques for dual system encryption and fully secure hibe with short ciphertexts. In Daniele Micciancio, editor, *Theory of Cryptography*, pp. 455–479, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[55] J. Li, W. Yao, Y. Zhang, H. Qian, and J. Han. Flexible and fine-grained attribute-based data storage in cloud computing. *IEEE Transactions on Services Computing*, Vol. 10, No. 5, pp. 785–796, 2017.

[56] X. Li, S. Tang, L. Xu, H. Wang, and J. Chen. Two-factor data access control with efficient revocation for multi-authority cloud storage systems. *IEEE Access*, Vol. 5, pp. 393–405, 2017.

[57] Zhen Liu, Zhenfu Cao, Qiong Huang, Duncan S. Wong, and Tsz Hon Yuen. Fully secure multi-authority ciphertext-policy attribute-based encryption without random oracles. In Vijay Atluri and Claudia Diaz, editors, *Computer Security – ESORICS 2011*, pp. 278–297, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg.

[58] Nori Matsuda, Takashi Ito, Hideya Shibata, Mitsuhiro Hattori, and Takato Hirano. Efficient searchable encryption and its application to web services. In *DICOMO'2013*, Vol. 2013, pp. 2067–2074, jul 2013.

[59] A.J. Menezes, T. Okamoto, and S.A. Vanstone. Reducing elliptic curve logarithms to logarithms in a finite field. *IEEE Transactions on Information Theory*, Vol. 39, No. 5, pp. 1639–1646, 1993.

[60] Victor S. Miller. Use of elliptic curves in cryptography. In Hugh C. Williams, editor, *Advances in Cryptology — CRYPTO '85 Proceedings*, pp. 417–426, Berlin, Heidelberg, 1986. Springer Berlin Heidelberg.

[61] Kenta Nomura, Masami Mohri, Yoshiaki Shiraishi, and Masakatu Morii. Attribute revocable multi-authority attribute-based encryption with forward secrecy for cloud storage. *IEICE TRANSACTIONS on Information and Systems*, Vol. E100-D, No. 10, pp. 2420–2431, 2017.

[62] National Institute of Standards and Technology. Data encryption standard (des). FIPS 46, 1977.

[63] National Institute of Standards and Technology. Advanced encryption standard (aes). FIPS 197, 2001.

[64] Tatsuaki Okamoto and Katsuyuki Takashima. Hierarchical predicate encryption for inner-products. In Mitsuru Matsui, editor, *ASIACRYPT 2009*, Vol. 5912 of *LNCS*, pp. 214–231. Springer, Heidelberg, 2009.

[65] Tatsuaki Okamoto and Katsuyuki Takashima. Fully secure functional encryption with general relations from the decisional linear assumption. In Tal Rabin, editor, *CRYPTO 2010*, Vol. 6223 of *LNCS*, pp. 191–208. Springer, Heidelberg, 2010.

[66] Rafail Ostrovsky, Amit Sahai, and Brent Waters. Attribute-based encryption with non-monotonic access structures. In *Proceedings of the 14th ACM Conference on Computer and Communications Security*, CCS '07, pp. 195—-203. Association for Computing Machinery, 2007.

[67] Raluca Ada Popa, Catherine M. S. Redfield, Nickolai Zeldovich, and Hari Balakrishnan. Cryptdb: Protecting confidentiality with encrypted query processing. In

Proceedings of the Twenty-Third ACM Symposium on Operating Systems Principles, SOSP '11, pp. 85–100, New York, NY, USA, 2011. ACM.

[68] Michael O. Rabin. Digitalized signatures and public-key functions as intractable as factorization. MIT Laboratory for Computer Science, 1979.

[69] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, Vol. 21, No. 2, p. 120–126, February 1978.

[70] Amit Sahai, Hakan Seyalioglu, and Brent Waters. Dynamic credentials and ciphertext delegation for attribute-based encryption. In Reihaneh Safavi-Naini and Ran Canetti, editors, *Advances in Cryptology – CRYPTO 2012*, pp. 199–217, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[71] Amit Sahai and Brent Waters. Fuzzy identity-based encryption. In Ronald Cramer, editor, *EUROCRYPT 2005*, Vol. 3494 of *LNCS*, pp. 457–473. Springer, Heidelberg, 2005.

[72] Latanya Samarati, Pierangela; Sweeney. Protecting privacy when disclosing information: k-anonymity and its enforcement through generalization and suppression, 1998.

[73] S. Santesson, M. Myers, R. Ankney, A. Malpani, S. Galperin, and C. Adams. X.509 internet public key infrastructure online certificate status protocol - ocsp. RFC6960, 2013.

[74] Vishal Saraswat and Rajeev Anand Sahu. Short integrated pke+peks in standard model. In Sk Subidh Ali, Jean-Luc Danger, and Thomas Eisenbarth, editors, *Security, Privacy, and Applied Cryptography Engineering*, pp. 226–246, Cham, 2017. Springer International Publishing.

[75] Jae Hong Seo and Keita Emura. Adaptive-id secure revocable hierarchical identity-based encryption. In *IWSEC 2015*, Vol. 9241 of *LNCS*, pp. 21–38. Springer, Heidelberg, 2015.

[76] Jae Hong Seo, Tetsutaro Kobayashi, Miyako Ohkubo, and Koutaro Suzuki. Anonymous hierarchical identity-based encryption with constant size ciphertexts. In Stanislaw Jarecki and Gene Tsudik, editors, *PKC 2009*, Vol. 5443 of *LNCS*, pp. 215–234. Springer, Heidelberg, 2009.

[77] Adi Shamir. Identity-based cryptosystems and signature schemes. In George Robert Blakley and David Chaum, editors, *Advances in Cryptology*, pp. 47–53, Berlin, Heidelberg, 1985. Springer Berlin Heidelberg.

[78] Elaine Shi, John Bethencourt, T-H. Hubert Chan, Dawn Song, and Adrian Perrig. Multi-dimensional range query over encrypted data. In *IEEE SP 2007*, pp. 350–364. IEEE Press, Los Alamitos, 2007.

[79] Dawn Xiaodong Song, David Wagner, and Adrian Perrig. Practical techniques for searches on encrypted data. In *2000 IEEE Symposium on Security and Privacy*, pp. 44–55. IEEE Press, Los Alamitos, 2000.

[80] Yinxia Sun, Futai Zhang, and Anmin Fu. Revocable certificateless encryption with ciphertext evolution. In *ACISP 2018*, Vol. 10946 of *LNCS*, pp. 741–749. Springer, Heidelberg, 2018.

[81] Takayuki Suzuki, Go Kojima, Keisei Fujiwara, and Masayuki Yoshino. Proposal and implementation of secure architecture for web application. *IEICE Journal*, Vol. J101-D, No. 1, pp. 68–78, jan 2018.

[82] Tatsuya Suzuki, Keita Emura, and Toshihiro Ohigashi. A generic construction of integrated secure-channel free peks and pke. In Chunhua Su and Hiroaki Kikuchi, editors, *Information Security Practice and Experience*, pp. 69–86, Cham, 2018. Springer International Publishing.

[83] Atsushi Takayasu and Yohei Watanabe. Lattice-based revocable identity-based encryption with bounded decryption key exposure resistance. In *ACISP 2017*, Vol. 10342 of *LNCS*, pp. 184–204. Springer, Heidelberg, 2017.

[84] Peter van Liesdonk, Saeed Sedghi, Jeroen Doumen, Pieter Hartel, and Willem Jonker. Computationally efficient searchable symmetric encryption. In Willem Jonker and Milan Petkovic, editors, *Secure Data Management*, Vol. 6358 of *Lecture Notes in Computer Science*, pp. 87–100. Springer Berlin / Heidelberg, 2010.

[85] Guojun Wang, Qin Liu, and Jie Wu. Hierarchical attribute-based encryption for fine-grained access control in cloud storage services. In *ACM CCS'10*, pp. 735—-737, 2010.

[86] Brent Waters. Efficient identity-based encryption without random oracles. In Ronald Cramer, editor, *Advances in Cryptology – EUROCRYPT 2005*, pp. 114–127, Berlin, Heidelberg, 2005. Springer Berlin Heidelberg.

[87] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. Cryptology ePrint Archive, Report 2008/290, 2008. https://ia.cr/2008/290.

[88] Brent Waters. Dual system encryption: Realizing fully secure IBE and HIBE under simple assumptions. In Shai Halevi, editor, *Advances in Cryptology - CRYPTO 2009*, Vol. 5677 of *LNCS*, pp. 619–636, Berlin, Heidelberg, 2009. Springer, Heidelberg.

[89] Brent Waters. Ciphertext-policy attribute-based encryption: An expressive, efficient, and provably secure realization. In *PKC 2011*, Vol. 6571 of *LNCS*, pp. 53–70. Springer, Heidelberg, 2011.

[90] Qianhong Wu, Bo Qin, Lei Zhang, Josep Domingo-Ferrer, and Oriol Farràs. Bridging broadcast encryption and group key agreement. In *ASIACRYPT 2011*, pp. 143–160. Springer Berlin Heidelberg, 2011.

[91] Shengmin Xu, Guomin Yang, Yi Mu, and Willy Susilo. Mergeable and revocable identity-based encryption. In *ACISP 2017*, Vol. 10342 of *LNCS*, pp. 147–167. Springer, Heidelberg, 2017.

[92] Kotoko Yamada, Nuttapong Attrapadung, Keita Emura, Goichiro Hanaoka, and Keisuke Tanaka. Generic constructions for fully secure revocable attribute-based encryption. In Simon N. Foley, Dieter Gollmann, and Einar Snekkenes, editors, *Computer Security – ESORICS 2017*, pp. 532–551, Cham, 2017. Springer International Publishing.

[93] K. Yang and X. Jia. Expressive, efficient, and revocable data access control for multi-authority cloud storage. *IEEE Transactions on Parallel and Distributed Systems*, Vol. 25, No. 7, pp. 1735–1744, 2014.

[94] Yanjiang Yang, Joseph K. Liu, Kaitai Liang, Kim-Kwang Raymond Choo, and Jianying Zhou. Extended proxy-assisted approach: Achieving revocable fine-grained encryption of cloud data. In *ESORICS 2015*, pp. 146–166. Springer International Publishing, 2015.

[95] Yanjiang Yang, Joseph Liu, Zhuo Wei, and Xinyi Huang. Towards revocable fine-grained encryption of cloud data: Reducing trust upon cloud. In *Information Security and Privacy*, pp. 127–144, Cham, 2017. Springer International Publishing.

[96] Ping Yu, Qiaoyan Wen, Wei Ni, Wenmin Li, Caijun Sun, Hua Zhang, and Zhengping Jin. Decentralized, revocable and verifiable attribute-based encryption in hybrid cloud system. In *Wireless Personal Communications volume 106*, pp. 719–738. Springer Berlin Heidelberg, 2019.

[97] S. Yu, C. Wang, K. Ren, and W. Lou. Achieving secure, scalable, and fine-grained data access control in cloud computing. In *IEEE INFOCOM 2010*, pp. 1–9, 2010.

[98] Rui Zhang and Hideki Imai. Combining public key encryption with keyword search and public key encryption. *IEICE Journal*, Vol. E92-D, No. 5, pp. 888–896, May 2009.

[99] カーライルアダムズ, スティーブロイド. PKI–公開鍵インフラストラクチャの概念、標準、展開. ピアソンエデュケーション, 2000.

[100] 高橋克巳. パネルディスカッション＜匿名加工情報の利活用に向けて＞k-匿名性に関する疑問に答える, 11 月 2017 年.

[101] Douglas R. Stinson(桜井幸一監訳). 暗号理論の基礎. 共立出版株式会社, 1996.

謝辞

本研究を進めるにあたり，指導教員として丁寧にかつきめ細やかなご指導を賜りました静岡大学 西垣正勝教授に心より御礼申し上げます．特に大学院入学前にアルゴリズム開発やプロト開発まで完了していたにも関わらず，暗号として重要な安全性証明が完了していない状況でしたが，色々と議論していただき安全性証明を完成することができました．また，研究成果を情報処理学会論文誌に投稿する際に，そして博士論文をまとめる際に，研究の背景・課題が読者に分かりやすくかかれているか，そしてアイデアのポイントが万人に分かりやすく書けているか等，研究者としての基本的なスキルからご指導をいただきましたことを感謝しております．

博士論文審査委員として，学位論文の内容について様々な視点からご助言をいただきました静岡大学 大木哲史准教授，岩田太教授，宮崎佳典教授に御礼申し上げます．私の専門である公開鍵暗号技術の視点から学位論文を執筆いたしましたが，計算アルゴリズムやプライバシー保護など幅広い視点でご意見・ご指摘をいただき，より広い研究分野の中から自分の研究の位置づけを見直し，さらに今後の研究の方向性についても再考するきっかけとなりました．

筆者が情報セキュリティの研究を始めたきっかけは，大学 4 年の卒業研究にて標数 2 の拡大体上にて楕円曲線の構成方法を研究したことであり，その際に色々と公開鍵暗号の基礎から楕円曲線暗号に至るまで，色々とご指導いただきました中央大学 趙晋輝教授，中央大学研究開発機構 辻井重男教授に御礼申し上げます．この時に学んだ楕円曲線暗号の知識が基礎となり，本研究を実施することができました．また，ID ベース暗号や検索可能暗号の安全性を証明する技術について色々とご指導いただきました電気通信大学 太田和夫教授に御礼申し上げます．この時にご教授いただいた安全性証明の知見が，本研究で提案した方式の安全性を証明するにあたって非常に役立ちました．

本研究は筆者が社会人になってから実施した研究であり，会社の業務の一環として実施いたしました．筆者が検索可能暗号の研究に取り組みたいとお願いした際に快諾いただき，開発プロジェクトの立ち上げや予算申請，社内で研究を継続するための社内 PR，大学との共同研究の立ち上げなど，筆者が研究を行うにあたって色々とお世話になりました三菱電機 開発本部 松井充フェロー，同社 情報セキュリティ統括室 米田健部長に御礼申し上げます．また，開発プロジェクトのメンバーとして，新しく取り組む検索可能暗号の論文勉強会や技術ディスカッションなどを実施し，またプロトタイプ開発なども一緒になって実施いただきました三菱電機 情報技術総合研究所情報セキュリティ技術部のみなさま，特に平野貴人氏，川合豊氏，山中忠和氏，伊藤隆氏，服部充洋氏（現鎌倉製作所），酒井康行氏（現本社電子システム事業本部），高島克幸氏（現早稲田大学 数学教育専攻 教授），三菱電機インフォメーションシステムズの開発者の皆様に御礼申し上げます．また，データベースへの組込みに関しては，三菱電機 情報技術総合研究所 データマネージメント技術部のみなさまに色々とご協力いただき，プロトタイプ開発まで実施することができました．特に同部 田村孝之部長，柴田秀哉氏に御礼申し上げます．Oracle 社データベースや PostgreSQL に検索可能暗

号を組み込むには，筆者が持つデータベースを利用する知識だけでは大きく不足しておりました．データベースの専門家として，検索可能暗号を組み込むための実現方法を調査し，プロト開発まで実施して頂きました．データベース組込はお二方含めた皆様のご協力があったからであり，感謝しております．

本研究を完成させるにあたって，静岡大学へお誘いいただいた愛知工業大学 情報科学部 水野忠則教授，そして業務をしながらの大学院通学を勧めて頂きました三菱電機 名古屋製作所 大谷治之部長，同社 情報技術総合研究所 情報セキュリティ技術部 大松史生部長に御礼申し上げます．業務が多忙な中で，社会人学生として研究・論文執筆や学位取得を行うには非常に大きな不安がありましたが，業務と両立できるように計らっていただき，まだ入学後も色々とご支援いただきましたことを感謝しております．

研究を進める上での事務手続きなど，西垣研究室 安藤敦子秘書にご対応いただき，とてもお世話になりました．筆者は社会人学生で遠方に居住していたこと，在学中に新型コロナの影響で移動の自粛が求められるようになり，紙書類などでやり取りする際はすべて安藤秘書にお取次ぎいただきまして，感謝しております．

最後に，社会人学生として研究を進めるためには，会社の休日に研究を進める必要があり，多くの週末を研究に費やすこととなりました．筆者が無事に研究活動を行うことができたのも，家族の協力があってのものと考えています．協力いただいた妻，子供たち，両親に感謝いたします．

論文目録

A. 学位論文申請資格に関わる論文

1) N.Matsuda, T.Hirano, Y.Kawai, T.Ito, M.Hattori, T.Yamanaka, M.Nishigaki : Public-key Searchable Encryption with Index Generation for Shared Database, Journal of Information Processing (JIP), Vol.28, pp.520–536 (2020)

B. 学位論文内容に関わる論文（未発表論文も含む）
なし

C. その他の論文

1) 松田 規, 米田 健 : 状況依存アクセス制御方式によるセキュア車載 Java プラットフォームの実現, 情報処理学会論文誌, Vol.46, No.12, pp.2983–2996 (2005).
2) M.Fujita, Y.Iida, M.Hattori, T.Yamanaka, N.Matsuda, S.Ito, H.Kikuchi : Proposal and Development of Anonymization Dictionary Using Public Information Disclosed by Anonymously Processed Information Handling Business Operators, In: Barolli L., Yim K., Chen HC. (eds), Innovative Mobile and Internet Services in Ubiquitous Computing (IMIS 2021), Lecture Notes in Networks and Systems, vol 279. Springer, Cham, pp.30-39 (2021).
3) M.Hattori, T.Hirano, N.Matsuda, F.Omatsu, R.Shimizu, Y.Wang : Privacy-Utility Tradeoff for Applications Using Energy Disaggregation of Smart-Meter Data, Journal of Information Processing (JIP), Vol.26, pp.648–661 (2018).
4) Y.Wang, N.Raval, P.Ishwar, M.Hattori, T.Hirano, N.Matsuda, R.Shimizu : On methods for privacy-preserving energy disaggregation, 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2017), pp. 6404-6408 (2017).
5) T.Hirano, T.Ito, Y.Kawai, N.Matsuda, T.Yamamoto, T.Munaka : A practical attack to AINA2014's countermeasure for cancelable biometric authentication protocols, 2016 International Symposium on Information Theory and Its Applications (ISITA), pp. 315-319 (2016).
6) T.Hirano, M.Hattori, Y.Kawai, N.Matsuda, M.Iwamoto, K.Ohta, Y.Sakai, T.Munaka : Simple, Secure, and Efficient Searchable Symmetric Encryption with Multiple Encrypted Indexes, In: Ogawa K., Yoshioka K. (eds), Advances in Information and Computer Security (IWSEC 2016), Lecture Notes in Computer Science, vol 9836. Springer, Cham, pp.91-110 (2016).
7) M.Hattori, T.Hirano, T.Ito, N.Matsuda, T.Mori, Y.Sakai, K.Ohta : Ciphertext-Policy Delegatable Hidden Vector Encryption and Its Application, IEICE TRANSACTIONS on Fundamentals of Electronics, Communications and Computer Sciences, Vol.E96-A, No.1, pp.53-67 (2013).

8) T.Hirano, M.Hattori, T.Ito, N.Matsuda : Cryptographically-Secure and Efficient Remote Cancelable Biometrics Based on Public-Key Homomorphic Encryption, In: Sakiyama K., Terada M. (eds) Advances in Information and Computer Security, International Workshop on Security (IWSEC 2013), Lecture Notes in Computer Science, vol 8231. Springer, Berlin, Heidelberg, pp.183–200 (2013).

9) M.Hattori, N.Matsuda, T.Ito, Y.Shibata, K.Takashima, T.Yoneda : Provably-Secure Cancelable Biometrics Using 2-DNF Evaluation, Journal of Information Processing (JIP), Vol.20, Issue 2, pp.496–507 (2012).

10) T.Hirano, M.Hattori, T.Ito, N.Matsuda, T.Mori : Homomorphic encryption based cancelable biometrics secure against replay and its related attack, 2012 International Symposium on Information Theory and its Applications, pp.421–425 (2012).

11) M.Hattori, T.Hirano, T.Ito, N.Matsuda, T.Mori, Y.Sakai, K.Ohta : Ciphertext-Policy Delegatable Hidden Vector Encryption and Its Application to Searchable Encryption in Multi-user Setting, In: Chen L. (eds) Cryptography and Coding, IMA International Conference on Cryptography and Coding (IMACC 2011), Lecture Notes in Computer Science, vol 7089. Springer, Berlin, Heidelberg, pp.190–209 (2011).

12) 伊藤 隆, 太田 英憲, 松田 規, 米田 健 : 散布型センサネットワークにおける散布の確率分布を利用した鍵格納方式, 電子情報通信学会論文誌 A, Vol.J89-A, No.12, pp.1034–1043 (2006).

13) T.Ito, H.Ohta, N.Matsuda, T.Yoneda : A key pre-distribution scheme for secure sensor networks using probability density function of node deployment, Proceedings of the 3rd ACM workshop on Security of ad hoc and sensor networks (SASN '05), pp.69—75 (2005).

14) 楠 和浩, 今井 功, 祢宜 知孝, 松田 規, 牛島 和夫 : インターネットを利用した遠隔制御機器アクセス方式の提案と評価, 電子情報通信学会論文誌 B, Vol.J83-B, No.9, pp.1283–1292 (2000).

15) J.Chao, N.Matsuda, S.Tsujii : Efficient construction of secure hyperelliptic discrete logarithm problems, In: Han Y., Okamoto T., Qing S. (eds), Information and Communications Security (ICICS 1997), Lecture Notes in Computer Science, vol 1334. Springer, Berlin, Heidelberg, pp.291–301 (1997).

D. 口頭発表など

1) 森 拓海, 藤田 真浩, 柴田 陽一, 松田 規 : エラー状態や競合発生を防止するユーザ認可方式の提案と実装, コンピュータセキュリティシンポジウム (CSS 2021) 論文集, pp.199–206 (2021).

2) 藤田 真浩, 山中 忠和, 松田 規, 金岡 晃 : 日本における認証サイトパスワード構成ポリシーの大規模実態調査, コンピュータセキュリティシンポジウム (CSS 2021) 論文集, pp.425–432 (2021).

3) 山中 忠和, 松田 規, 宮崎 一哉 : 閾値署名を適用したリモート署名システムの安全性評価, コンピュータセキュリティシンポジウム (CSS 2021) 論文集, pp.514–521 (2021).
4) 藤田 真浩, 山中 忠和, 松田 規, 金岡 晃 : パスワード構成ポリシー自動取得技術の開発 : Web 上認証サイトのパスワード構成ポリシー表示方法に関する大規模調査, 情報処理学会研究報告コンピュータセキュリティ, Vol.2021-CSEC-94, No.19, pp.1-8 (2021).
5) 藤田 真浩, 山中 忠和, 松田 規 : パスワード認証システムとユーザー間におけるパスワード構成ポリシーのギャップを緩和するパスワード構成ポリシー変換システムの提案, 暗号と情報セキュリティシンポジウム (SCIS 2021), 2B4-1 (2021).
6) 藤田 真浩, 飯田 泰興, 服部 充洋, 山中 忠和, 松田 規, 伊藤 聡志, 菊池 浩明 : 匿名加工情報取扱事業者の公表情報の記載内容に関する初期検討 : 匿名加工カタログの実装から得られた知見の報告, コンピュータセキュリティシンポジウム (CSS 2020) 論文集, pp.665–672 (2020).
7) 藤田 真浩, 飯田 泰興, 服部 充洋, 山中 忠和, 松田 規, 伊藤 聡志, 菊池 浩明 : 匿名加工情報取扱事業者による公表情報を利用した匿名加工カタログの提案と実装, コンピュータセキュリティシンポジウム (CSS 2020) 論文集, pp.1214–1221 (2020).
8) 藤田 真浩, 飯田 泰興, 服部 充洋, 山中 忠和, 松田 規, 菊池 浩明 : 匿名加工情報取扱事業者による公表情報を利用した匿名加工カタログの提案と実装, 暗号と情報セキュリティシンポジウム (SCIS 2020), 3C4-4 (2020).
9) M.Hattori, T.Ito, N.Matsuda, Y.Shibata : A Privacy-Preserving Anomaly Detection Outsourcing Scheme Using Predicate Encryption, IWSEC 2019 Poster Session (2019).
10) M.Hattori, T.Hirano, N.Matsuda, R.Shimizu, Y.Wang : Privacy-Utility Trade-off for the Appliance Usage Analysis of Smart-Meter Data, Computer Security Symposium (CSS 2016), pp.1215–1222 (2016).
11) 森 拓海, 松田 規 : クラウド ID 管理と関数型暗号を用いた機密ファイル共有システム, マルチメディア，分散協調とモバイルシンポジウム (DICOMO 2016) 論文集, pp.1251–1258 (2016).
12) 祢宜 知孝, 森 拓海, 平野 貴人, 小関 義博, 松田 規, 河内 清人, 米田 健 : セキュア SIM を搭載したスマートフォンを利用したトランザクション署名手法の提案, 情報処理学会研究報告コンピュータセキュリティ, Vol.2015-CSEC-71, No.11, pp.1-8 (2015).
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 (DICOMO 2013) 論文集, pp.2067–2074 (2013).
14) 市川 幸宏, 松田 規, 坂上 勉 : 関数型暗号アプリケーションにおける適切な述語付与方式の検討, 情報処理学会研究報告コンピュータセキュリティ, Vol.2013-CSEC-60, No.5, pp.1–6 (2013).

15) 伊藤 隆, 平野 貴人, 森 拓海, 服部 充洋, 松田 規 : 秘匿検索の実システム適用に関する考察, 電子情報通信学会 ISEC 研究会 (ISEC2012-124), Vol.112, No.461, pp.279–283 (2013).

16) 松田 規, 服部 充洋, 市川 幸宏, 平野 貴人, 伊藤 隆, 酒井 康行 : 関数型暗号における IC カード上の復号演算の効率化に関する考察, 暗号と情報セキュリティシンポジウム (SCIS 2013), 3F3-5 (2013).

17) 森 拓海, 平野 貴人, 服部 充洋, 伊藤 隆, 松田 規 : 検索可能暗号におけるハフマン符号を利用した索引手法, 情報処理学会第 74 回全国大会講演論文集, Vol.2012, No.1, pp.589–590 (2012).

18) 伊藤 隆, 平野 貴人, 森 拓海, 服部 充洋, 松田 規 : 統計量ごとに開示・非開示を設定可能な秘匿集計方式, 電子情報通信学会 ISEC 研究会 (ISEC2011-103), Vol.111, No.455, pp.189–193 (2012).

19) 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), 暗号と情報セキュリティシンポジウム (SCIS 2012), 1A3-1 (2012).

20) 伊藤 隆, 服部 充洋, 松田 規, 坂井 祐介, 太田 和夫 : 頻度分析耐性を持つ高速秘匿検索方式, 電子情報通信学会 ISEC 研究会 (ISEC2010-72), Vol.110, No.443, pp.1–6 (2011).

21) 平野 貴人, 森 拓海, 服部 充洋, 伊藤 隆, 松田 規 : 秘匿生体認証における準同型性を利用した能動的攻撃への対策, 電子情報通信学会 ISEC 研究会 (ISEC2010-73), Vol.110, No.443, pp.7–14 (2011).

22) M.Hattori, N.Matsuda, T.Ito, Y.Shibata, K.Takashima, T.Yoneda : Efficient Biometric Authentication Protocols Using 2-DNF Evaluation, マルチメディア, 分散協調とモバイルシンポジウム (DICOMO 2010) 論文集, DS-5 (2010).

23) 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, 暗号と情報セキュリティシンポジウム (SCIS 2010), 4F2-3 (2010).

24) 服部 充洋, 柴田 陽一, 伊藤 隆, 松田 規, 高島 克幸, 米田 健 : 2-DNF 準同型暗号を用いた秘匿生体認証, 電子情報通信学会 ISEC 研究会 (ISEC2009-68), Vol.109, No.271, pp.113–120 (2009).

25) 鶴川 達也, 河野 篤, 子安 健彦, 松田 規 : 著作権を考慮したデジタルデータのバックアップ・リストア-車買換え時のカーナビ蓄積データ移動サービスの提案-, 情報処理学会第 69 回全国大会講演論文集, Vol.2007, No.1, pp.413–414 (2007).

26) 齋藤 和美, 太田 英憲, 松田 規, 伊藤 隆, 辻 宏郷, 米田 健 : 携帯電話スマートキーを活用した車操作権限の貸与方式の提案, 情報処理学会研究報告コンピュータセキュリティ, Vol.2004-CSEC-028, No.33, pp.223–228 (2005).

27) 松田 規, 米田 健, 野村 立, 中川路 哲男 : JAVA による MISTY ソケット通信ライブラリの実現, 情報処理学会第 64 回全国大会講演論文集, Vol.2002, No.1, pp.397–398 (2002).

28) 松田 規, 中野 初美, 中川路 哲男：携帯電話応用システムにおけるセキュリティ機能に関する考察, 情報処理学会第 62 回全国大会講演論文集, Vol.2001, No.1, pp.467–468 (2002).

29) 松田 規, 中野 初美, 中川路 哲男：暗号通信における監査機能の実現方式に関する考察, 情報処理学会第 59 回全国大会講演論文集, Vol.1999, pp.397–398 (1999).

30) 中野 初美, 竹原 明, 松田 規, 中川路 哲男：キーリカバリシステムの試作と商用システムへの応用に関する検討, 情報処理学会研究報告コンピュータセキュリティ, Vol.1998-CSEC-003, No.108, pp.1–6 (1998).

31) 松田 規, 竹原 明, 中野 初美, 中川路 哲男：キーリカバリシステムの試作, 情報処理学会第 57 回全国大会講演論文集, Vol.1998, pp.494–495 (1999).

32) 佐藤 淳一, 松田 規, 趙 晋輝, 辻井 重男：素体上における大種数超楕円曲線の構成方法の考察, 電子情報通信学会総合大会, A-7-9 (1998).

33) 中村 理, 松田 規, 趙 晋輝, 辻井 重男：CM を有するアーベル多様体を用いた暗号系に関する考察, 電子情報通信学会 ISEC 研究会 (ISEC96-81), Vol.96, No.590, pp.171–176 (1997).

34) 佐藤 淳一, 松田 規, 趙 晋輝, 辻井 重男：素体上における大種数超楕円曲線の構成方法の考察, 電子情報通信学会 ISEC 研究会 (ISEC96-80), Vol.96, No.588, pp.165–170 (1997).

35) 中村 理, 松田 規, 趙 晋輝, 辻井 重男：CM アーベル多様体に基づく暗号系に関する考察, 電子情報通信学会ソサエティ大会, A-7-16 (1997).

36) 松田 規, 趙 晋輝, 辻井 重男：超楕円曲線上における安全な離散対数問題の効率的な構成法に関する考察, 電子情報通信学会 ISEC 研究会 (ISEC96-18), Vol.96, No.167, pp.117–126 (1996).

37) 松田 規, 趙 晋輝, 辻井 重男：超楕円曲線上における安全な離散対数問題の効率的な構成法に関する考察, 電子情報通信学会ソサエティ大会, A-163 (1996).

38) 松田 規, 趙 晋輝, 辻井 重男：On discrete logarithm problems over hyperelliptic curves, 暗号と情報セキュリティシンポジウム (SCIS 1996), (1996).

39) 趙 晋輝, 松田 規, 原田 幸治, 辻井 重男：拡大体上の暗号学的に安全な楕円曲線の効率的な構成法に関する考察, 電子情報通信学会ソサエティ大会, A-119 (1995).

E. 表彰

1) DICOMO2013 優秀論文賞受賞
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 (DICOMO 2013) 論文集, pp.2067–2074 (2013). に対して

2) DICOMO2010 最優秀論文賞受賞
M.Hattori, N.Matsuda, T.Ito, Y.Shibata, K.Takashima, T.Yoneda： Efficient Biometric Authentication Protocols Using 2-DNF Evaluation, マルチメディア，

分散協調とモバイルシンポジウム (DICOMO 2010) 論文集, DS-5 (2010). に対して

3) SCIS'96 論文賞受賞
松田 規, 趙 晋輝, 辻井 重男：On discrete logarithm problems over hyperelliptic curves, 暗号と情報セキュリティシンポジウム (SCIS 1996), (1996). に対して