共通鍵暗号による秘匿検索暗号のセキュリティ

太田和夫（おおたかずお）

要　旨

秘匿検索暗号は、事前に設定しておいたキーワード等によって、ファイル等を暗号化したままキーワード検索を実行することを可能にする技術である。近年、パブリック・クラウドの登場により、金融分野をはじめ、さまざまなデータやその処理を外部事業者にアウトソース化することが可能となってきているが、そうしたクラウド活用を検討するうえで、外部事業者に対してデータの機密性を確保しつつキーワード検索を可能とする秘匿検索暗号は、重要な技術の1つと考えられる。最近では、秘匿検索暗号にかかる研究が活発化しているほか、既存のクラウド上で秘匿検索暗号によるファイルの暗号化やキーワード検索を実現する製品も登場するなど、実用化に向けた動きが活発化しているといえる。本稿では、こうした秘匿検索暗号のうち、高速での暗号化等が可能な共通鍵暗号ベースの方式に着目し、それらがどのようなセキュリティを達成することができるかについて分析するとともに、今後の技術的な課題を考察する。

キーワード：　共通鍵暗号、キーワード検索、高機能暗号、秘匿検索暗号、パブリック・クラウド

本稿は、日本銀行金融研究所からの委託研究論文である。本稿の作成に当たっては、筑波大学の西出隆志准教授、芦原聡介氏から有益なコメントを頂戴した。ここに記して感謝したい。ただし、本稿に示されている意見は、筆者個人に属し、日本銀行の公式見解を示すものではない。また、ありうべき誤りはすべて筆者個人に属する。

太田和夫　電気通信大学大学院情報理工学研究科（E-mail: kazuo.ohta@uec.ac.jp）

1. はじめに

近年、さまざまな分野において、情報技術をより一段と活用して業務の効率化や新しいビジネスの立上げを推進する機運が高まっている。そうした動きの 1 つとして、パブリック・クラウド（以下、クラウド）による業務のアウトソース化が挙げられる。金融分野においても、こうしたアウトソース化が進展しているとみられるが、金融機関が取り扱うデータには機密性が高いデータが少なからず存在しており、そうしたデータの処理をクラウドによって実現する場合、当該データの機密性をどのように確保するかが課題といわれている（金融情報システムセンター［2015］）。

こうした課題への対応に資する技術として注目を集めているのが、高機能暗号である（清藤・四方［2014］、小暮・下山・安田［2015］）。高機能暗号は、暗号化したままデータを演算できるなど、通常のデータの暗号化・復号に加えて、より高度な機能を実現する暗号の総称である。高機能暗号が利用できるようになれば、クラウド上でデータを処理しつつ、暗号化によってそれらの機密性を確保することが可能になると期待される。これまでにさまざまなタイプの高機能暗号が提案されており、その機能や実現形態は多岐にわたっているが、クラウド活用を展望した場合、特に注目されるのは、秘匿検索暗号（データを暗号化したままキーワード検索等が可能）、準同型暗号（データを暗号化したまま四則演算等が可能）、属性ベース暗号（暗号化したデータの復号権限をエンティティの属性に応じて柔軟に設定可能）である[1]。これらの暗号を今後活用していくうえで、高機能暗号がどのようなセキュリティ（安全性）を達成できるかについて理解しておくことが有用である。

本稿では、高機能暗号のうち、秘匿検索暗号に焦点を当てる。秘匿検索暗号が実現すれば、例えば、暗号化した大量のファイルをクラウドに預託したうえで、当該ファイルに対してキーワード検索をクラウド上で実施し、対応する（複数の）ファイルを暗号化したままクラウドから受信する（受信後に復号する）といった外部のサービスを利用することができるようになる[2]。金融機関が有する大量のデータを安全かつ安価に保管するとともに、キーワード検索によって必要なデータを抽出するといった用途も想定され、金融機関の業務の効率化につながる可能性がある。

秘匿検索暗号の具体的な実現方式は、これまでに数多く提案されている。それらは、共通鍵暗号に基づく方式と公開鍵暗号に基づく方式に大別されるが、本稿で

1 秘匿検索暗号は、検索可能暗号とも呼ばれている。

2 最近では、秘匿検索の機能をクラウドで実現するための技術やツールの提案、それらを用いたサービスの実証実験等が、複数のベンダーにおいて活発に進められている。例えば、日本電気［2013］、日立ソリューションズ［2016］、富士通研究所［2014］、三菱電機［2016］、NTT テクノクロス［2013］が挙げられる。

は、相対的に高速で暗号化等が可能といわれている共通鍵暗号に基づく方式に焦点を当てる。そのうえで、さまざまな提案方式のなかから代表的な方式を選び、それらが達成しうるセキュリティを明らかにするとともに、それを証明する手法を説明する。また、実用化を展望した際に、技術的な課題としてどのようなものが残されているかを考察する。

2 節では、共通鍵暗号ベースの秘匿検索暗号の基本的なモデルや各種の検索機能等を説明する。3、4 節では、既存の提案方式のなかから代表的なものとして、カートモラ（Curtmola）らの方式（Curtmola *et al*. [2006]）とアシャロフ（Asharov）らの方式（Asharov *et al*. [2016]）をそれぞれ取り上げ、方式の概要やセキュリティの考え方について説明する。5 節では、今後の秘匿検索暗号にかかる主な技術的な課題を考察する。

2. 秘匿検索暗号の機能と主な実現方式

(1) モデルとエンティティ

秘匿検索暗号の代表的な活用形態として、金融機関が内部で生成したデータ（機密性が高いものも含まれる）をクラウドに預託し、当該金融機関やその関係者が必要に応じてキーワード検索を実施して該当するデータをクラウドから抽出するというものを考える。本稿では、ユーザ（金融機関に相当する）とサーバ（クラウドに相当する）からなるモデルを想定する[3]。

ユーザは、サーバに預託するデータを生成するエンティティである。当該データには、機密性の高いものが含まれており、サーバに預託する際には、当該データを暗号化したうえで送信する。また、暗号化等に用いる鍵の生成や配付等を行う。また、ユーザは、サーバに預託した（暗号化されている）データを後日利用する。利用したいデータに含まれる検索用のキーワードを決定したうえで、当該キーワードを暗号化してサーバに送信し、検索結果としての（暗号化された）データをサーバから受信する。最後に、ユーザは、受信した当該データを復号して利用する。

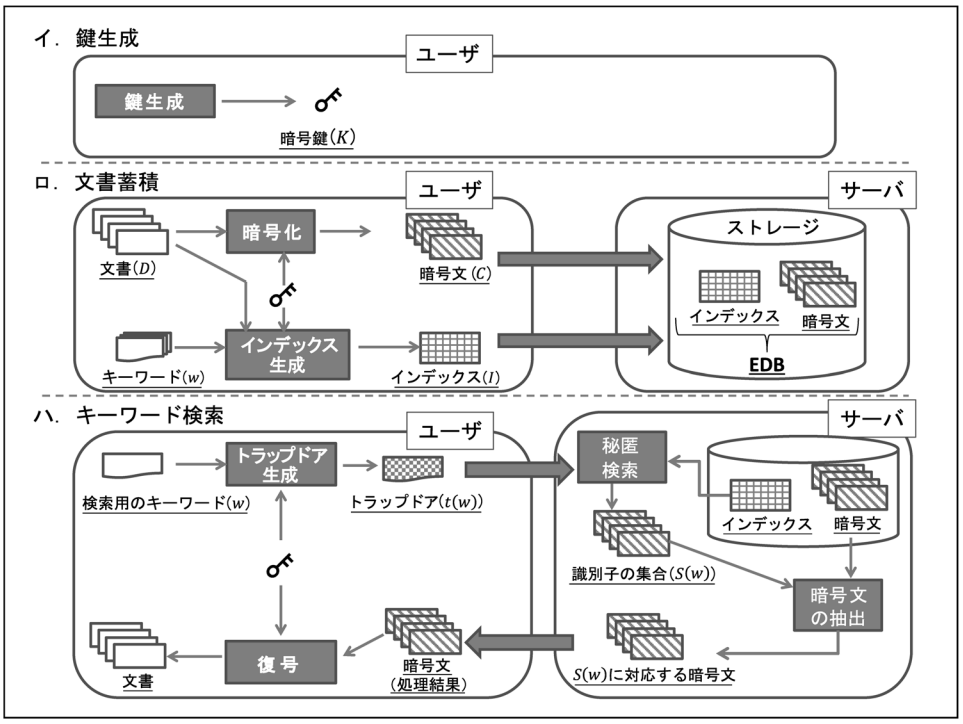
3 既存の提案方式のなかには、ユーザ、サーバに加えて、サーバに対して秘匿検索の処理を要求するエンティティ（以下、検索者）を想定したモデルを前提としているものもある（例えば、マルチ・ユーザ・モデル）。ただし、そうしたモデルにおいては、ユーザは、検索者に対して、暗号化と復号に利用可能な暗号鍵を配付することとなり、ユーザが検索者を信頼できるエンティティとみなし、サーバと結託するなどの攻撃を行わないことが前提となる。そうしたモデルは、実際上、ユーザと検索者をまとめて 1 つのエンティティと考えるモデル（本稿の対象とするもの）と同一になると考えることができるため、ここでは、ユーザとサーバのみからなるモデルに焦点を当てることとした。

サーバは、暗号化されたデータを受信・保管するとともに、ユーザから受信した（暗号化された）キーワードに基づいて検索を実施し、その結果として抽出されたデータをユーザに送信する。ここでは、ユーザから受信したデータを手掛りに、預託されたデータやキーワードの内容を推測しようとする（データの機密性に対する攻撃を試行する）可能性があると想定する[4]。

(2) 3つのフェーズと処理フロー

本節（1）のモデルにおいて、①データの暗号化等に用いる鍵（以下、暗号鍵）を準備するフェーズ（鍵生成フェーズ）、②データを暗号化してサーバに預託するフェーズ（文書蓄積フェーズ）、③サーバに預託されているデータに対してキーワード検索を実行するフェーズ（キーワード検索フェーズ）が存在する（図表1を参照）。なお、本稿で使用する記法や用語等については、補論1を参照されたい。

図表1　秘匿検索暗号のモデルにおける3つのフェーズ



4　本稿では、データの完全性については検討対象外とする。

鍵生成フェーズでは、ユーザは、暗号鍵（K）を生成する[5]。その後、ユーザは、外部に漏洩しないように暗号鍵を安全に保管する。

文書蓄積フェーズでは、ユーザは、サーバに預託したいデータ（以下、文書）を選択して暗号化するとともに、後日それらを検索する際に用いるキーワード（w）を設定する。キーワードがサーバに知られず、かつ、効率的に秘匿検索で利用できるように、ユーザは、キーワード等に一定の変換処理を施し、各キーワードを含む暗号文（$\boldsymbol{C}$）の識別子等から構成される、インデックス（$\boldsymbol{I}$）と呼ばれる暗号化されたデータを生成する。そのうえで、ユーザは、暗号化された文書（以下、暗号文）とインデックスを、サーバに送信・預託する。サーバは、ユーザから受信した暗号文とインデックスの組を暗号化データベース（$\boldsymbol{EDB}$）として蓄積し保管する。

キーワード検索フェーズでは、ユーザは、入手したい暗号文に含まれていると考えられるキーワード w をまず決定する。そのうえで、ユーザは、キーワードがサーバに知られず、かつ、秘匿検索で利用できるように、当該キーワード w に対して一定の変換処理を施したデータ $t(w)$（以下、トラップドア）をサーバに送信する。サーバは、トラップドア $t(w)$ を用いて、インデックスからキーワードを含む暗号文の識別子を検索し、検索結果として識別子の集合 $S(w)$ を出力する。最後に、ユーザは、識別子の集合に対応する暗号文をサーバから受信し、暗号鍵を用いて復号する。

(3) 検索の種類と主な実現方式

これまでに、さまざまなタイプのキーワード検索を実現する秘匿検索暗号が提案されている（図表 2 を参照）。検索の方法は、1 つのキーワードを用いるもの（以下、単一キーワード方式）と複数のキーワードを用いるもの（以下、複数キーワード方式）に分けられる。さらに、単一キーワード方式は、検索用のキーワードと預託時に用いられたキーワードとの関係に着目し、完全一致検索、部分一致検索、（これら以外の）類似検索に分類される。また、複数キーワード方式は、個々のキーワードに関して完全一致検索を実施することを前提とするものが多く、検索結果の組合せのバリエーションに応じて、論理積検索、論理和検索、論理積検索と論理和検索の組合せに分類される。

これらのほか、最近では、検索以外の機能として、いったんサーバに預託した暗号文やインデックスを後から追加したり削除したりする機能を実現する方式も提案されはじめている（例えば、黒澤ほか［2016］、平野ほか［2016］、Yavuz and Guajardo [2015]）。

5 共通鍵暗号をベースとしており、暗号化の鍵と復号の鍵は同一となる。

図表２　秘匿検索にかかる主な機能

主な検索機能		機能の概要	主な実現方式
単一キーワード方式	完全一致検索	検索用のキーワードが預託時のキーワードと一致する文書を検索する。	アシャロフらの方式（Asharov *et al.* [2016]）、黒澤らの方式（黒澤ほか［2016］）
	部分一致検索	預託時のキーワードの一部が検索用のキーワードと一致する文書を検索する。（例）検索用のキーワード「銀行」に対し、「全国銀行協会」や「地方銀行」等が預託時のキーワードに対応する文書を検索する。	平野らの方式（平野ほか［2016］）
	類似検索（完全一致、部分一致以外）	検索用のキーワードと預託時のキーワードが、一定の関係性（完全一致、部分一致以外）を有する文書を検索する。	海上らの方式（海上ほか［2016］）
複数キーワード方式	論理積検索	検索用のキーワードを複数指定し、それらがすべて預託時のキーワードと一致する文書を検索する。	小嶋らの方式（小嶋ほか［2016］）
	論理和検索	検索用のキーワードを複数指定し、それらのうち少なくとも１つが預託時のキーワードと一致する文書を検索する。	ガジェック（Gajek）の方式（Gajek [2016]）
	論理積検索と論理和検索の組合せ	上記の論理積検索と論理和検索を組み合わせて文書を検索する。	ガジェックの方式（Gajek [2016]）

(4)　検討対象とする実現方式

本稿では、さまざまな実現方式のなかでも、カートモラらの方式とアシャロフらの方式を対象として、安全性の定義やモデル、安全性の証明の方法について考察する。

レザー・カートモラ（Reza Curtmola）らは、共通鍵暗号に基づく秘匿検索暗号の安全性のモデルを初めて定義したほか、比較的効率のよい実現方式を提案した。カートモラらの方式における安全性の概念は、現在では他の実現方式でも利用されている（例えば、平野ほか［2016］）。

また、ギラッド・アシャロフ（Gilad Asharov）らは、秘匿検索のサービスを提供する際に実用性を評価する尺度の１つである局所性（locality）という概念に着目し、それを最適化した方式を提案した（詳細は後述）。アシャロフらの方式は、安全性も証明されており、実用性と証明可能安全性を兼ね備えた方式として注目されている。

3. カートモラらの方式における安全性の定式化

秘匿検索暗号方式（Π）を設計する際、例えば、サーバに対して、トラップドア $t(w)$ に対応する暗号文の識別子 $S(w)$ を求めることを許容する一方で、暗号文とインデックスの組である暗号化データベースから、キーワードと文書の識別子に関するデータベース（$\boldsymbol{DB}$）に関する情報を入手できないようにしたい[6]。既存の方式では、サーバを攻撃者として想定したうえで、サーバが入手可能な（キーワード検索にかかる）情報の多寡に基づき、安全性を検討している。本節では、こうしたサーバへの情報の漏洩にかかるカートモラらの安全性の定式化の手法について説明する。

(1) 回避不可能な情報漏洩と漏洩関数

イ. 辞書 Δ 上の秘匿検索暗号方式の場合

サーバとユーザとの間でやり取りされるデータ（以下、履歴）は、文書の集まりと一連のキーワードに依存する。一連のキーワードは、ユーザが検索したいものとして決定され、サーバに対して秘匿する必要がある。カートモラらは、履歴から漏洩する情報のうち、検索結果そのものをアクセス・パターン（access pattern）と呼び、アクセス関数として定式化している。また、検索結果から推定可能な関係をサーチ・パターン（search pattern）と呼び、サーチ関数として定式化している。履歴、アクセス関数、サーチ関数は以下のとおり定義されている。

- 履歴（H）：サーバとユーザの間で q 回のやり取りを行い、それが文書の集まり $\boldsymbol{D}$ と q 個のキーワード $\boldsymbol{w} := (w_1, w_2, \ldots, w_q)$ に関するものであるとき、その履歴を、$\boldsymbol{D}$ 上の q クエリ（query）履歴と呼び、$H := (\boldsymbol{D}, \boldsymbol{w})$ と表す[7]。
- アクセス関数（α）：サーバは、k 個のトラップドア $t(w_1)$、$t(w_2)$、…、$t(w_k)$（ただし、$1 \leq k \leq q$）に対する検索結果 $\boldsymbol{DB}(\boldsymbol{w}_k) := (\boldsymbol{DB}(w_1), \boldsymbol{DB}(w_2), \ldots, \boldsymbol{DB}(w_k))$ を入手できる。ここで、$\boldsymbol{w}_k := (w_1, w_2, \ldots, w_k)$ とする。これに基づき、アクセス関数を $\alpha(\boldsymbol{D}, \boldsymbol{w}_k) := \boldsymbol{DB}(\boldsymbol{w}_k)$ と定義する。
- サーチ関数（σ）：トラップドアが確定的に生成されるならば、サーバは、2 つ

6 例えば、カートモラらの方式では、ある文書に多種類のキーワードが含まれていること、あるいは、あるキーワードが多くの文書に含まれることなど、データベースに関する情報が漏洩することを防止するために、インデックスに出現する文書の識別子の出現回数を揃えている。

7 q はセキュリティ・パラメータ λ の多項式によって与えられる。

の異なるタイミング（例えば、第 i 番目と第 j 番目）で受信するトラップドアが一致するか否か、すなわち、$t(w_i) = t(w_j)$ が成立するか否かを調べることで、2 つのキーワードが等しいか否かを推測可能である[8]。これに基づき、サーチ関数 σ を、$\sigma(\boldsymbol{D}, \boldsymbol{w}_k) := \Theta_k$ とする（ただし、$1 \leq i, j \leq k$）。ここで、行列 Θ_k は、次のように定義される。

$$\Theta_k[i, j] := \begin{cases} 1 & \text{if } (w_i = w_j) \\ 0 & \text{if } (w_i \neq w_j). \end{cases}$$

カートモラらの安全性の定式化では、これらの情報をまとめてトレース (trace) と呼んでいる。

定義 1（トレース）．辞書（キーワードのすべての候補からなる集合）を Δ、Δ に含まれるキーワードからなる文書の集まりを $\boldsymbol{D}$ とするとき、$\boldsymbol{D}$ 上の q クエリ履歴 $H = (\boldsymbol{D}, \boldsymbol{w})$ から求められるトレースを、$\tau(H) := (|D_1|, |D_2|, \ldots, |D_n|, \alpha(H), \sigma(H))$ と定義する。ここで $|D_n|$ は文書 D_n のビット長を表す（補論 1（1）参照）。なお、Δ は公開されているとする。

カートモラらは、上記の $\tau(H)$ で示される情報の漏洩を、回避不可能な情報漏洩（acceptable information leakage）と呼んでいる。これらよりも多くの情報が漏洩するか否かが、共通鍵暗号に基づく秘匿検索暗号の安全性の評価のベンチマークの 1 つとなっている。

ロ．　キーワードの集合 $\boldsymbol{W}$ 上の秘匿検索暗号方式の場合

カートモラらの方式では、文書 $\boldsymbol{D}$ は、Δ に含まれるキーワードを組み合わせて生成されており、実際に用いられたキーワードをすべて把握することはできない、すなわち、そうしたキーワードの集合が公開されていないという状況が前提となっている。これに対して、4 節で説明するアシャロフらの方式では、実際に用いられたキーワードの集合 $\boldsymbol{W}$ が公開されているという状況を前提としている。そのうえで、文書蓄積フェーズの処理とキーワード検索フェーズの処理を分けて情報漏洩を検討し、文書蓄積フェーズの処理で回避不可能な情報漏洩を漏洩関数 $\mathcal{L}_1$ で、キーワード検索フェーズの処理で回避不可能な情報漏洩を漏洩関数 $\mathcal{L}_2$ でモデル化している[9]。

8 トラップドアが確定的に生成されるとは、同一のキーワードに対するトラップドアが毎回同一の値として生成されるケースを意味する。一方、同一のキーワードであっても毎回異なるトラップドアが生成されるケースは、トラップドアが確率的に生成されるという。

9 こうした $\mathcal{L}_1$ と $\mathcal{L}_2$ を用いたモデル化は、Chase and Kamara [2010] によって提案されたものであり、それをアシャロフらは参照している。

(2) カートモラらの構成法

イ. インデックスの生成にかかるアイデア

カートモラらの方式では、インデックス $\boldsymbol{I}$ は、キーワード w から生成したデータ（val）と文書の識別子 id から構成される。これらのデータの組（以下、エントリ）を (val, id) と表記する[10]。

キーワード検索フェーズの処理においては、サーバはトラップドアと val を照合して $\boldsymbol{I}$ に属するか否かをチェックする。したがって、トラップドアが確定的に生成される方式の場合、検索用のキーワード w を含む文書の識別子 id がアクセス・パターンとして漏洩する。すなわち、サーバは $\alpha(\boldsymbol{D}, w) = \boldsymbol{DB}(w)$ を求めることができる。

このままでは、あるキーワードが多くの文書に含まれているといった情報が漏洩することになる。対策として、カートモラらは、$\boldsymbol{I}$ に含まれる（各キーワードに対応する）識別子 id の個数を揃えるという方法を採用している。各キーワードに対応する識別子の個数を揃える場合、すべての文書に含まれるキーワードがありうるので、その個数は、キーワード w_i を含む文書の数 N_i の最大値 n となる。これに加えて、キーワードが $n_W := |\boldsymbol{W}|$ 個ありうるので、$\boldsymbol{I}$ のエントリ数 s は $s = n_W \times n$ となる。ここで $|\boldsymbol{W}|$ は、$\boldsymbol{W}$ の要素数を表す（補論 1（1）参照）。

上記の条件を満足させるために、$\boldsymbol{I}$ の id の個数を s とし、各 w_i に関して n 個の id を $\boldsymbol{I}$ に埋め込む。具体的には、各 w_i に対する n 個の val（$val_{i,1}, val_{i,2}, \ldots, val_{i,n}$）を組成したうえで、各 val に id を対応させて n 個の id_j（$1 \le j \le n$）を生成し、$\boldsymbol{I}$ における n ヵ所のエントリを埋めることを考える。ただし、それらのエントリのうち、$\boldsymbol{DB}(w_i)$ に属する id（N_i 個）を識別できるようにしたい。

このようにすると、エントリの val 成分 $val_{i,j}$ ごとに $id_{i,j}$ が 1 つ定まり、$id_{i,j} \in \{id_{i_1}, id_{i_2}, \ldots, id_{i_{N_i}}\}$ ならば出力ありとして $id_{i,j}$ を出力し、$id_{i,j} \notin \{id_{i_1}, id_{i_2}, \ldots, id_{i_{N_i}}\}$ ならば出力なしとするようにして $\boldsymbol{I}$ を生成すればよい。出力なしの $id_{i,j}$ に対応するエントリをダミーのエントリと呼ぶ。

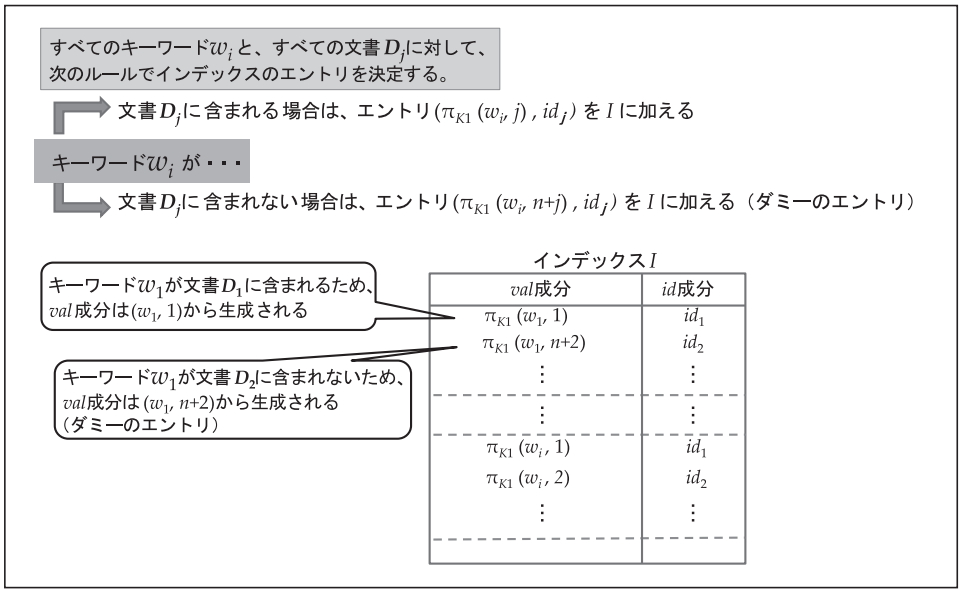
$\boldsymbol{I}$ における id の j 成分として、出力ありのとき $1 \le j \le n$ をそのまま使用し、出力なし（ダミーのエントリ）のとき $n + 1 \le n + j \le 2n$ とする。これによって、ダミーのエントリか否かを識別できる。また、$val_{i,j} = (w_i, j)$ および $val_{i,n+j} = (w_i, n + j)$ とした場合、$\boldsymbol{I}$ の生成処理は以下のとおりとなる（図表 3 を参照）。

- $\boldsymbol{I} \leftarrow \emptyset$

  for $1 \le i \le n_W = |\boldsymbol{W}|$ **do**

10 id が暗号化される方式もあるが、Curtmola *et al.* [2006] では暗号化しない。

図表 3　インデックス $\boldsymbol{I}$ の生成のアイデア



for $1 \le j \le n = |\boldsymbol{D}|$ **do**
  if $id_j \in \boldsymbol{DB}(w_i)$ **then**
    $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{((w_i, j), id_j)\}$
  else
    $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{((w_i, n + j), id_j)\}$.

上記の場合、val の値が w_i および id_j の添え字 j に対応しているので、検索前でも $\boldsymbol{I}$ から検索結果 $\boldsymbol{DB}(w_i)$ が漏れてしまう。そこで、val の値にランダム置換 π（K_1 による暗号化。詳細は後述）を施すことで、w_i と id_j の関係を秘匿するというアイデアが採用されている。

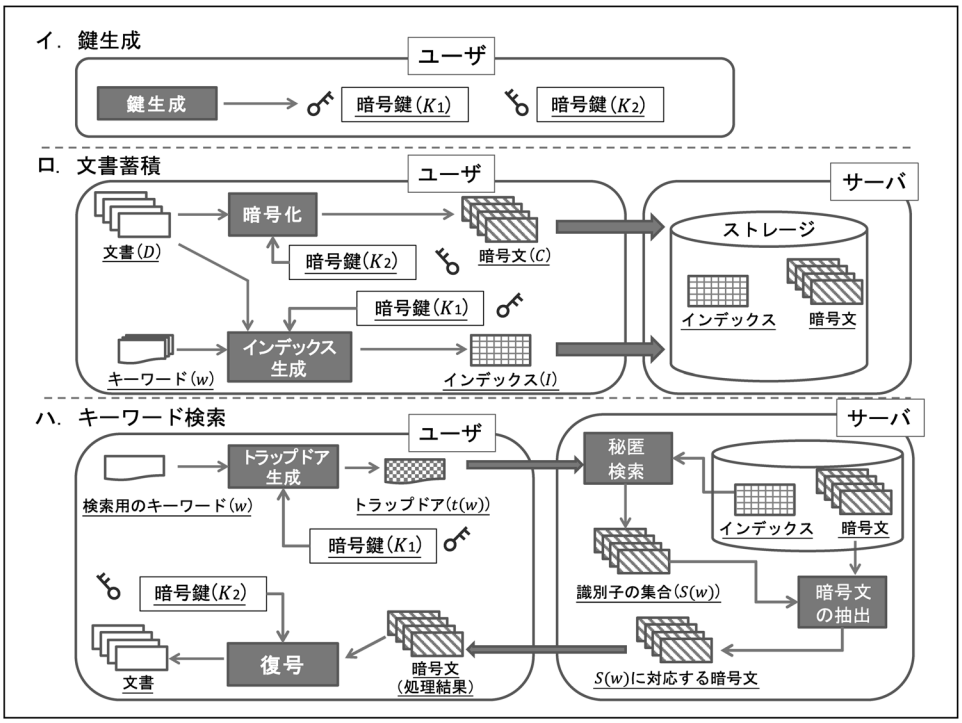
ロ．具体的な方式

こうしたアイデアに基づき、カートモラらは以下の方式を提案している。まず、ランダム置換 π を $\pi : \{0,1\}^{\lambda} \times \{0,1\}^{\ell+\log(2n)} \rightarrow \{0,1\}^{\ell+\log(2n)}$ と表す[11]。また、選択平文攻撃に対して疑似ランダム性を満たす共通鍵暗号方式（PCPA-secure symmetric key encryption scheme）を $\mathsf{SKE} = (\mathsf{Gen}, \mathsf{Enc}, \mathsf{Dec})$ と表す[12]。$n = |\boldsymbol{D}|$、$n_W = |\boldsymbol{W}|$ とするとともに、文書の集まり $\boldsymbol{D}$ の第 j 番目の文書の識別子を id_j とする $(1 \le j \le n)$[13]。また、

11 ここで λ は、セキュリティ・パラメータであり、$\ell + \log(2n)$ は、キーワード w_i に対応するインデックスの val 成分のビット長（ℓ は w_i のビット長）である。

12 選択平文攻撃とは、攻撃者が、指定した平文に対する暗号文を入手できるという状況のもとで実行さ

図表 4　カートモラらの方式の処理フロー



SKE を用いて文書（D_j）ごとに暗号文（C_j）を生成し、暗号文の集まりを $\boldsymbol{C}$ とする。このとき、カートモラらの方式は以下の 5 種類の処理から構成される（図表 4 を参照）。

- ユーザによる鍵生成（$\mathtt{KeyGen}(1^\lambda)$）：セキュリティ・パラメータ λ に対して、$K_1 \xleftarrow{u} \{0,1\}^\lambda$、$K_2 \leftarrow \mathtt{SKE.Gen}(1^\lambda)$ を生成して鍵ペア $\boldsymbol{K} = (K_1, K_2)$ を出力する。K_1 は *val* の値のランダム置換 π に使用し、K_2 は文書の暗号化に用いる。
- ユーザによる文書蓄積（$\mathtt{EDBSetup}(\boldsymbol{K}, \boldsymbol{D})$）：暗号鍵 K と文書の集まり $\boldsymbol{D}$ に対して、以下の処理を行い、インデックスと暗号文の集まりの組 $(\boldsymbol{I}, \boldsymbol{C})$ を出力する。

れる攻撃の総称である。選択平文攻撃に対して疑似ランダム性を満たす共通鍵暗号方式とは、直感的に説明すると、攻撃者が選択した平文に対応する（当該暗号方式による）暗号文と（この平文とは無関係な）疑似乱数が生成され、それらを攻撃者が受け取ったときに、当初の平文に対応する暗号文がどちらかを攻撃者が正しく回答する確率が 2 分の 1 よりも有意に大きくなることはない、つまり、暗号文が疑似乱数と区別がつかないという性質を満たすことを意味している。

13 Curtmola *et al.* [2006] では、j を $DB(w_i) = \{id_{i_1}, id_{i_2}, \ldots, id_{i_{N_i}}\}$ での第 j 番目（$1 \leq j \leq N_i$）の意味で用いている。この意味ではインデックスの構成手順は動作しないと考えられることから、ここでは、早坂らの記法を用いて方式を記述することとした（Hayasaka *et al.* [2016]）。

$\boldsymbol{I} \leftarrow \emptyset$
for $1 \le i \le n_W = |\boldsymbol{W}|$ **do**
  for $1 \le j \le n = |\boldsymbol{D}|$ **do**
    if $id_j \in \boldsymbol{DB}(w_i)$ **then**
      $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{(\pi_{K_1}(w_i, j), id_j)\}$
    else
      $\boldsymbol{I} \leftarrow \boldsymbol{I} \cup \{(\pi_{K_1}(w_i, n + j), id_j)\}$
$\boldsymbol{C} \leftarrow \emptyset$
for $1 \le j \le n$ **do**
  $\boldsymbol{C} \leftarrow \boldsymbol{C} \cup \{\mathtt{SKE.Enc}(K_2, D_j)\}$
return $(\boldsymbol{I}, \boldsymbol{C})$.

- ユーザによるトラップドア生成（$\mathtt{TokenGen}(\boldsymbol{K}, w_i)$）：暗号鍵 K_1 とキーワード w_i に対して、トラップドア $t(w_i) := (\pi_{K_1}(w_i, 1), \pi_{K_1}(w_i, 2), \ldots, \pi_{K_1}(w_i, n))$ を出力する。
- サーバによるキーワード検索（$\mathtt{Search}(\boldsymbol{I}, t(w_i))$）：インデックス $\boldsymbol{I}$ とトラップドア $t(w_i)$ に対して、以下の処理を行い、検索結果として、w_i を含む文書の識別子の集合 $S(w_i)$ を出力する。

  $S(w_i) \leftarrow \emptyset$
  for $1 \le j \le n$ **do**
    for $(\pi_{K_1}(w_i, j), id_j) \in \boldsymbol{I}$ **do**
      $S(w_i) \leftarrow S(w_i) \cup \{id_j\}$
  return $S(w_i)$.
- ユーザによる文書復号（$\mathtt{Dec}(\boldsymbol{K}, \boldsymbol{C})$）：暗号鍵 $\boldsymbol{K}$ と暗号文の集まり $\boldsymbol{C}$ に対して、文書の集まり $\boldsymbol{D} \leftarrow \mathtt{SKE.Dec}(K_2, \boldsymbol{C})$ を出力する。

(3) シミュレーションに基づいた安全性のモデル

イ. サーバのモデル化

サーバへの情報漏洩が回避不可能な情報漏洩を超えるものとなるか否かを評価するためのモデルとして、カートモラらによる、シミュレーションに基づいた安全性（Simulation-Based Security）のモデルについて説明する[14]。ここでは、本節（1）ロ.で説明したアシャロフらの記法を用いてトレースを表現するため、漏洩関数 $\mathcal{L}_1$ と

14 Curtmola *et al.* [2006] では、シミュレーションによるモデル以外のモデルも提示され、それを用いた安全性の定式化も行われている。もっとも、攻撃実施中に得られる情報に基づいて、柔軟に攻撃の方法を変化させるという比較的強力な攻撃（適応的攻撃）を想定すると、シミュレーションを用いたモデルの方が安全性の条件がより厳しくなるとの結果が示されている。

$\mathcal{L}_2$ を、$\mathcal{L}_1(\boldsymbol{D}) := (|\boldsymbol{D}_1|, |\boldsymbol{D}_2|, \ldots, |\boldsymbol{D}_n|)$、$\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w}) := (\alpha(\boldsymbol{D}, \boldsymbol{w}), \sigma(\boldsymbol{D}, \boldsymbol{w}))$ と定義して説明する。

サーバ（攻撃者）は、確率的多項式時間アルゴリズム $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_{q+1})$ を備えたものとしてモデル化されている[15]。具体的には、$\mathcal{A}_0$ から $\mathcal{A}_{q+1}$ までの（$q+2$ 個の）アルゴリズムで構成され、$\mathcal{A}_0$ は、暗号化データベースから、目的とする情報を推定しやすいように悪意を持って $\boldsymbol{D}$ を選ぶことが許される。次に、$\mathcal{A}_1$ は、正規の文書蓄積フェーズの処理（EDBSetup）から出力された暗号化データベースを用いて、当該情報を推定しやすいキーワード w_1 を何らかの方法で選ぶ。さらに、$\mathcal{A}_i$（$i = 2, 3, \ldots, q$）は、そのときの状態にかかる情報（$st_{\mathcal{A}}$）や、それまでに入手したトラップドアの集合（$\boldsymbol{T}$）も使って、当該情報をより推定しやすいキーワード w_i を再度選ぶ。これを希望の回数（q 回）繰り返した後で、（目的とする情報の推定にかかる）$\boldsymbol{D}$ についてのある判定を行う。その内容は、攻撃者 $\mathcal{A}_{q+1}$ が判定しやすい事項で決めてよい[16]。

このように、攻撃者は、暗号化データベースやトラップドア等を確認してから、動的にキーワード w_i を繰り返し選択することができるという有利な状況で攻撃（判定）を行うモデルとする。このような攻撃は、適応的選択キーワード攻撃と呼ばれ、秘匿検索暗号方式に対する強力な攻撃である。

ロ．強秘匿性

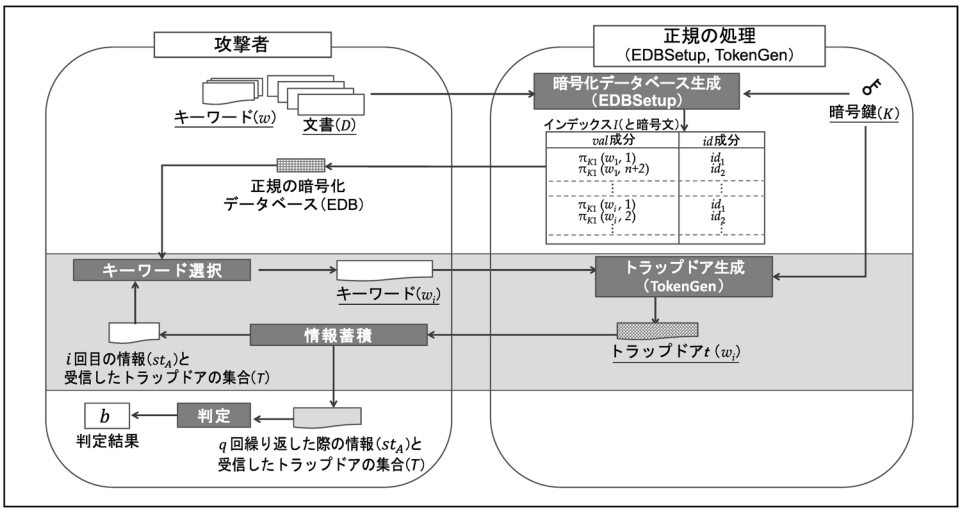
上記の攻撃者のアルゴリズム（以下、$\mathbf{SSE\text{-}Real}^{\mathrm{adpt}}_{\Pi,\mathcal{A}}$）は、暗号化データベースおよびトラップドアの生成において正規の処理（EDBSetup、TokenGen）を用いて攻撃を行っていることから、実環境と呼ぶ（図表 5 を参照）。

適応的選択キーワード攻撃を行ったとしても、サーバがいかなる内容も判定できない、つまり、何ら情報が追加的に漏洩しないということを、（方式 Π にとって理想的な環境である）トレースのみを利用して攻撃を行うシミュレータを用いて示す。シミュレータは、文書 $\boldsymbol{D}$ や鍵 $\boldsymbol{K}$ を用いずに、暗号化データベースやトラップドアの生成を何らかの方法で模倣するアルゴリズム $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_q)$ と定義され、正規の処理を行うことができないという意味で攻撃の有効性が低い。そこで、シミュレータを用いた攻撃と $\mathbf{SSE\text{-}Real}^{\mathrm{adpt}}_{\Pi,\mathcal{A}}$ の攻撃を比較すると、両方の攻撃において利用

15 確率的多項式時間アルゴリズムとは、同一の入力に対して毎回出力が異なるとともに、実行時間が、高々、パラメータのサイズを変数とする多項式によって表現することができるアルゴリズムのことである。

16 例えば、任意の文書に含まれるようなキーワードが存在するかなど、ユーザに直接質問できない事項を想定すると、$\boldsymbol{EDB}$ から $\boldsymbol{D}$ に関する何らかの漏洩情報を調べたいという攻撃者の意図をモデルとして捉えることができる。上記の事項の内容として何を採用してもよいというのが、このシミュレーションに基づいた安全性のモデルの利点である。なお、Curtmola *et al.* [2006] では、$\mathcal{A}_{q+1}$ は登場せず、その処理は識別者 (distinguisher) に担当させている。この定義はメリッサ・チェース（Melissa Chase）とセニー・カマラ（Seny Kamara）が与えた定式化である（Chase and Kamara [2010]）。

図表 5　サーバのモデル化（実環境の動作）

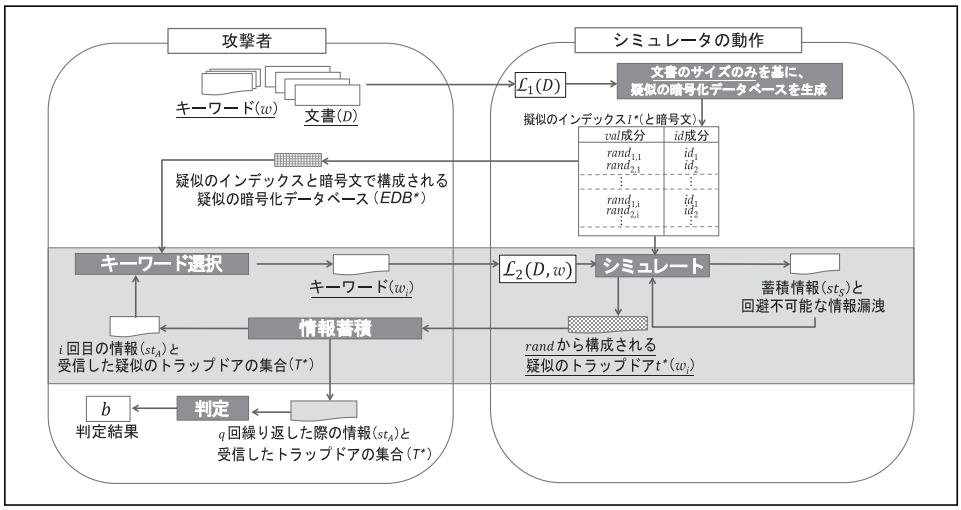


できる情報に差がほとんどなく、$\mathbf{SSE\text{-}Real}^{\mathrm{adpt}}_{\Pi,\mathcal{A}}$ は、攻撃の有効性が低いシミュレータと同程度の効果を有するに過ぎないことから、方式 Π は適応的選択キーワード攻撃に対して安全であると考える。

シミュレータ $\mathcal{S}_0$ は、暗号化データベースを出力する処理をシミュレートする（図表 6 を参照）。ここで、サーバは文書蓄積フェーズの処理における漏洩関数 $\mathcal{L}_1(\boldsymbol{D})$ から、各文書のサイズを知ることができるため、$\mathcal{S}_0$ は $\mathcal{L}_1(\boldsymbol{D})$ を用いて何らかの方法で疑似の暗号化データベースを出力することとなる。次に、シミュレータ $\mathcal{S}_i$ $(i = 1, 2, \ldots, q)$ は、$\mathcal{A}_i$ が選んだ w_i に対してトラップドアを生成する処理を、$\boldsymbol{K}$ と w_i を用いずにシミュレートする。ここで、$\mathcal{S}_i$ は、キーワード検索フェーズにおける漏洩関数 $\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w})$ を用いて疑似のトラップドア $t^*(w_i)$ を出力することとなる。上記の手順をモデル化したものがアルゴリズム $\mathbf{SSE\text{-}Ideal}^{\mathrm{adpt}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}$ であり、回避不可能な情報漏洩のみをサーバが攻撃に利用するという意味で理想環境と呼ぶことにする。

定義 2（実環境と理想環境）．Π を秘匿検索暗号方式、λ をセキュリティ・パラメータ、$\mathcal{L}_1, \mathcal{L}_2$ をそれぞれ文書蓄積フェーズの処理、キーワード検索フェーズの処理における漏洩関数とする。また、状態を表す変数を $st_{\mathcal{A}}$ とするほか、`EDBSetup` と `TokenGen` を、それぞれ暗号化データベースとトラップドアの生成アルゴリズムとする。確率的多項式時間アルゴリズム $\mathcal{A} = (\mathcal{A}_0, \mathcal{A}_1, \ldots, \mathcal{A}_{q+1})$ とシミュレータ $\mathcal{S} = (\mathcal{S}_0, \mathcal{S}_1, \ldots, \mathcal{S}_q)$ に対して、実環境 $\mathbf{SSE\text{-}Real}^{\mathrm{adpt}}_{\Pi,\mathcal{A}}(\lambda)$ と理想環境 $\mathbf{SSE\text{-}Ideal}^{\mathrm{adpt}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda)$ を次のとおり定義する。

図表 6　シミュレータの動作（理想環境）



- $\mathbf{SSE\text{-}Real}^{\mathrm{adpt}}_{\Pi,\mathcal{A}}(\lambda)$:

  $\boldsymbol{K} \leftarrow \mathtt{KeyGen}(1^\lambda)$

  $(\boldsymbol{D}, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^\lambda)$

  $\boldsymbol{EDB} = (\boldsymbol{I}, \boldsymbol{C}) \leftarrow \mathtt{EDBSetup}(\boldsymbol{K}, \boldsymbol{D})$

  $\boldsymbol{T} \leftarrow \emptyset$

  for $1 \le i \le q$,

  $(w_i, st_{\mathcal{A}}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, \boldsymbol{EDB}, \boldsymbol{T})$

  $t(w_i) \leftarrow \mathtt{TokenGen}(\boldsymbol{K}, w_i)$ and $\boldsymbol{T} \leftarrow \boldsymbol{T} \cup \{t(w_i)\}$

  $b \leftarrow \mathcal{A}_{q+1}(st_{\mathcal{A}}, \boldsymbol{EDB}, \boldsymbol{T})$, output b.

- $\mathbf{SSE\text{-}Ideal}^{\mathrm{adpt}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda)$:

  $(\boldsymbol{D}^*, st_{\mathcal{A}}) \leftarrow \mathcal{A}_0(1^\lambda)$

  $(\boldsymbol{EDB}^*, st_{\mathcal{S}}) \leftarrow \mathcal{S}_0(\mathcal{L}_1(\boldsymbol{D}))$

  $\boldsymbol{T}^* \leftarrow \emptyset$

  for $1 \le i \le q$,

  $(w_i, st_{\mathcal{A}}) \leftarrow \mathcal{A}_i(st_{\mathcal{A}}, \boldsymbol{EDB}^*, \boldsymbol{T}^*)$

  $(t^*(w_i), st_{\mathcal{S}}) \leftarrow \mathcal{S}_i(st_{\mathcal{S}}, \mathcal{L}_2(\boldsymbol{D}^*, \boldsymbol{w}_i))$ and $\boldsymbol{T}^* \leftarrow \boldsymbol{T}^* \cup \{t^*(w_i)\}$

  $b \leftarrow \mathcal{A}_{q+1}(st_{\mathcal{A}}, \boldsymbol{EDB}, \boldsymbol{T})$, output b.

このように、実環境と理想環境との差異は、暗号化データベースとトラップドアの生成の処理の部分である。理想環境では、正規の処理（EDBSetup、TokenGen）を $\mathcal{S}_0$ や $\mathcal{S}_i$ がシミュレートすることとなる。

定義 3（強秘匿性）．上記の定義 2 に基づき、次の性質を満たす $\mathcal{S}$ が存在するとき、Π を、$\mathcal{L}_1(\boldsymbol{D})$ と $\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w})$ に関して適応的選択キーワード攻撃に対する強秘匿性（adaptive semantic security）を満たすと定義する。

$$|\Pr[\mathbf{SSE\text{-}Real}^{\mathrm{adpt}}_{\Pi,\mathcal{A}}(\lambda) = 1] - \Pr[\mathbf{SSE\text{-}Ideal}^{\mathrm{adpt}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda) = 1]| \leq \mathtt{negl}(\lambda).$$

この不等式は、ある判定結果が実環境で得られる確率（$\Pr[\mathbf{SSE\text{-}Real}^{\mathrm{adpt}}_{\Pi,\mathcal{A}}(\lambda) = 1]$）と、同じ判定結果が理想環境で得られる確率（$\Pr[\mathbf{SSE\text{-}Ideal}^{\mathrm{adpt}}_{\Pi,\mathcal{L},\mathcal{A},\mathcal{S}}(\lambda) = 1]$）との差分（不等式の左辺に該当する）が、無視できる程度の値（不等式の右辺に該当する）に収まっているということを意味している。すなわち、適応的選択キーワード攻撃による効果は、シミュレーションによる弱い攻撃による効果とさほど変わらないということを示している。

ハ．カートモラらの方式における安全性証明

適応的選択キーワード攻撃に対する強秘匿性の証明について、上記のモデルに基づいて説明する。まず、理想環境のシミュレータがどのように構成されるかを説明したうえで、実環境と理想環境の間でインデックスを識別することが困難であり、理想環境よりも（攻撃を成功させるための）有利な情報を実環境においてより多く抽出することが事実上不可能であることを説明する[17]。

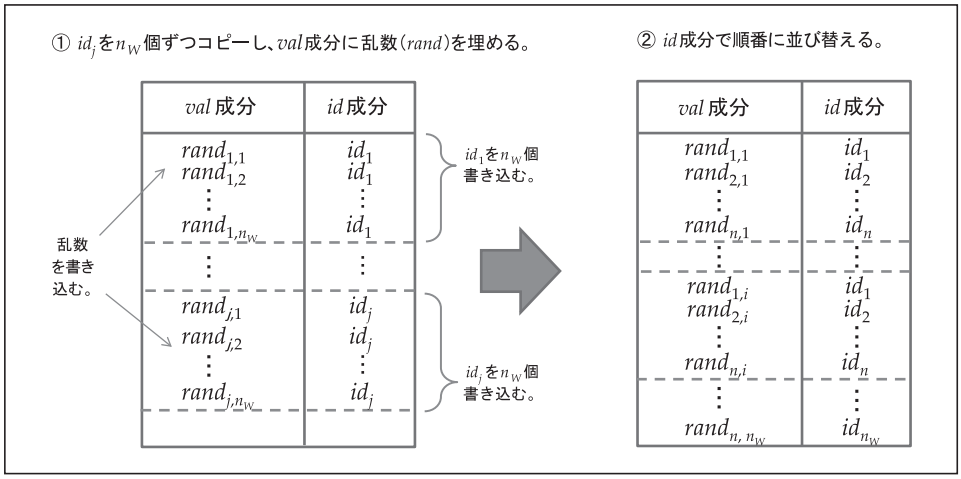
（イ） $\mathcal{S}_0$ によってシミュレートされたインデックス $\boldsymbol{I}^*$ の構成

まず、インデックスのシミュレーションの方法を説明する。シミュレートされたインデックスを $\boldsymbol{I}^*$ とする。$\mathcal{S}_0$ は、$\boldsymbol{I}^*$ の *id* 成分に $id_j (1 \leq j \leq n)$ をそれぞれ n_W 回書き込み、それぞれのエントリの *val* 成分に乱数を書き込んでエントリを入れ替える（図表 7 を参照）。$\boldsymbol{I}^*$ のエントリ数 s よりも *val* の値の候補（$\ell + \log(2n)$ ビット）が大きいので、乱数はすべて異なるように選ぶことができる。以下では、選んだ乱数を *rand* で表し、添え字をつけて特定のエントリの *val* 成分を表すことにする。

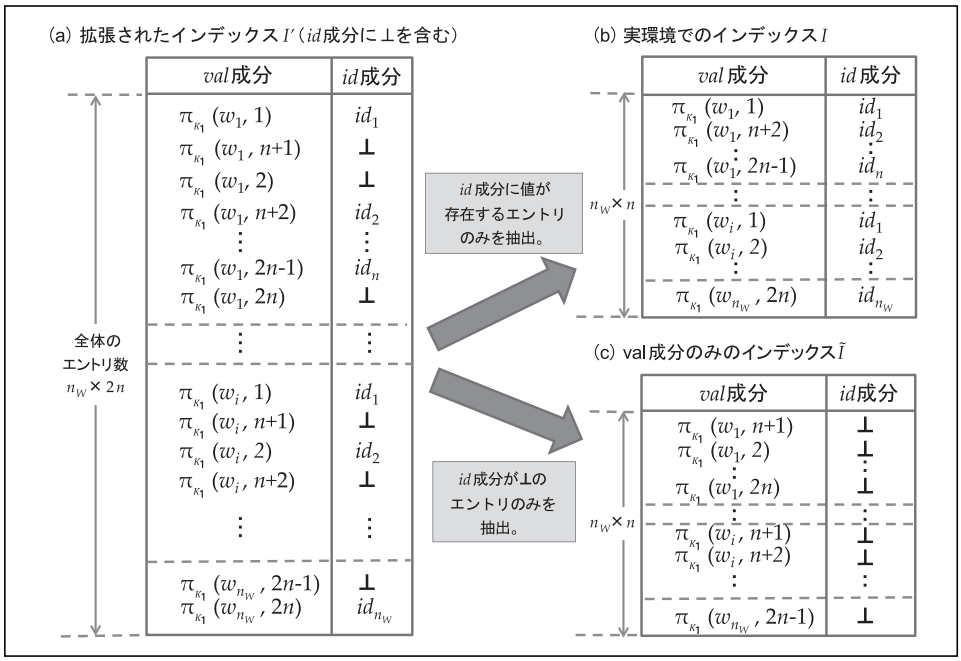
ところで、実環境のインデックス $\boldsymbol{I}$ を生成する方法として、*id* 成分に $\perp$（NULL）も許す、拡張されたインデックス（$\boldsymbol{I}'$ で表す）をまず生成し、*id* 成分に（NULL でない）*id* を持つエントリのみを抽出するという方法が考えられる（図表 8 を参照）。この場合、$\boldsymbol{I}'$ のエントリ数は $n_W \times 2n$ であり、*val* 成分の値はすべて異なっていて、$\{0,1\}^{\ell+\log(2n)}$ 上の一様な分布に従う。一方、*id* 成分に $\perp$ を持つエントリの集まりを $\boldsymbol{I}'$ から抽出してインデックス $\widetilde{\boldsymbol{I}}$ とすると、$\mathcal{S}_0$ は、$\widetilde{\boldsymbol{I}}$ もシミュレートする目的で、$\boldsymbol{I}'$ の *val* 成分として未使用だった乱数（すなわち、$\widetilde{\boldsymbol{I}}$ の *val* 成分の値）をすべて $st_{\mathcal{S}}$ に

17 ここでは、Hayasaka *et al.* [2016] の記法を用いて方式を記述した。以下の証明では、シミュレーションに n_W を明示的に用いており、カートモラらの方式は $\boldsymbol{W}$ 上の秘匿検索暗号方式の安全性証明となっている。辞書 Δ 上の秘匿検索暗号方式としての安全性証明については別途検討が必要である。

図表 7　理想環境におけるシミュレートされたインデックス I^* の例



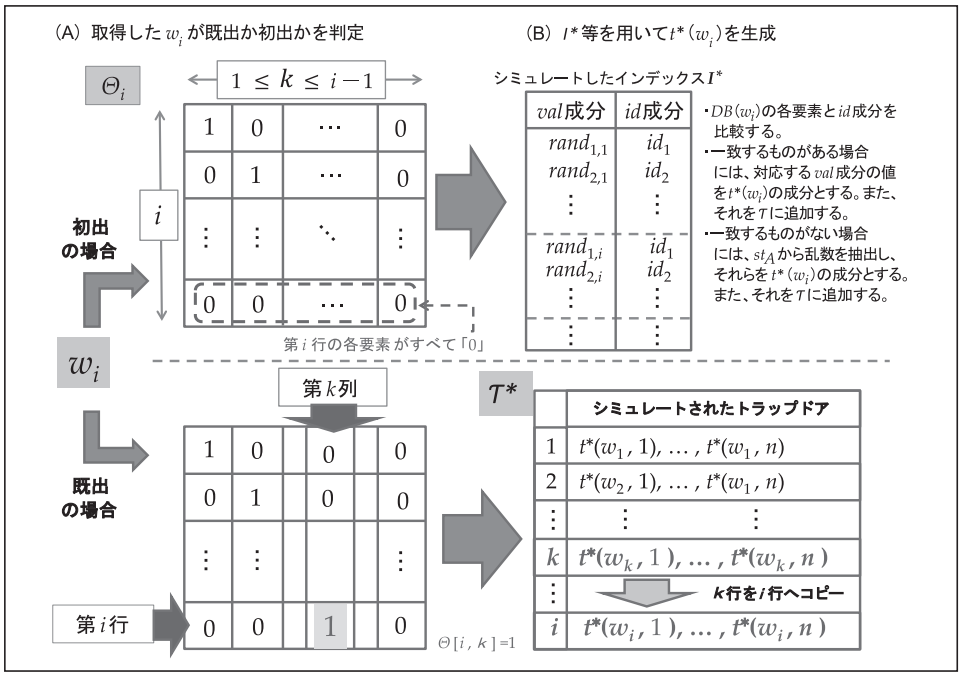
図表 8　拡張されたインデックス I'（カートモラらの方式）



書き込み記憶する[18]。

18　未使用の乱数の個数は、インデックス $\tilde{I}$ の val 成分に対応する要素を含むので、$n \times n_W$ 以上となる。

図表 9　トラップドア生成のシミュレーション例



（ロ）　$\mathcal{S}_i$ によるトラップドア生成のシミュレーション

$\mathcal{S}_i$ は、$\mathcal{A}_i$ が選定した w_i に対して $t(w_i)$ をシミュレートする（図表 9 を参照）。シミュレートした値を $t^*(w_i)$ とする。$\mathcal{S}_i$ は w_i 自体を知ることはできないが、漏洩関数 $\mathcal{L}_2$ からサーチ・パターン $\sigma(\boldsymbol{D}, \boldsymbol{w}) = \Theta$ が得られるので、$1 \le \forall k \le i-1$ の k に対して $\Theta[i,k]$ を計算することで、w_i が初出か既出かを判定できる。

w_i が既出の場合、それまでに生成したトラップドアの集合 $\boldsymbol{T}^*$ から w_i に対応する値 $t^*(w_i)$ を出力する。

w_i が初出の場合、$\boldsymbol{T}^*$ から出力できないので、別途 $t^*(w_i)$ を生成する必要がある。$\mathcal{S}_i$ は、シミュレートした $\boldsymbol{I}^*$ のエントリのうち、w_i に対応すると考えられるものを検索・抽出することになる。すなわち、$\mathcal{S}_i$ は、漏洩関数 $\mathcal{L}_2$ のうち、アクセス・パターン $\alpha(\boldsymbol{D}, \boldsymbol{w}_i)$ から $\boldsymbol{DB}(w_i)=(id_{i_1}, id_{i_2}, \ldots, id_{i_{N_i}})$ を知りうるので、id_j（$1 \le j \le n$）が $\boldsymbol{I}^*$ に含まれるか否かを順々に調べ、$\boldsymbol{I}^*$ に含まれる場合、id_j に対応するエントリのうち、未使用の最初の *val* 成分の値 $rand_{j,i}$ を $t^*(w_i)$ の一部として使う。$\boldsymbol{I}^*$ に含まれない場合は、$st_{\mathcal{A}}$ に含まれる初出の乱数を使う。以上をまとめて示すと、$1 \le j \le n$

を満たす j に対して、

$$t^*(w_i, j) := \begin{cases} rand_{j,i} & \text{if } (id_j \in \boldsymbol{DB}(w_i)) \\ \text{初出の乱数} & \text{if } (id_j \notin \boldsymbol{DB}(w_i)) \end{cases}$$

として $t^*(w_i) := (t^*(w_i, 1), t^*(w_i, 2), \ldots, t^*(w_i, n))$ とおく。さらに $t^*(w_i)$ を $\boldsymbol{T}^*$ に追加し、使用した初出の乱数を $st_{\mathcal{S}}$ から削除して $st_{\mathcal{S}}$ を更新するとともに、$t^*(w_i)$ を出力する。

（ハ） シミュレーションの妥当性と強秘匿性の証明

実環境のアルゴリズムを実行するとき、id_j を id 成分に持つ $\boldsymbol{I}$ のエントリが利用されるのは高々 n_W ヵ所においてであった。一方、理想環境で q 個のトラップドアをシミュレートするときに、id_j を id 成分に持つ $\boldsymbol{I}^*$ のエントリが利用されるのも高々 n_W ヵ所であるため、$\boldsymbol{I}^*$ を構成する際に、それぞれの id_j を n_W 個コピーするとともに val 成分にすべて異なる乱数を設定すれば、$\boldsymbol{I}$ のシミュレーションは完了する。

また、$\mathcal{S}_0$ が作成した $st_{\mathcal{S}}$（シミュレートした $\widetilde{\boldsymbol{I}}$）のエントリ数は少なくとも $n \times n_W$ であった。一方、$\mathcal{S}_i$ が $t^*(w_i)$ をシミュレートするために使う初出の乱数は高々 n 個である。$\mathcal{S}_1$ から $\mathcal{S}_q$ が最大で n_W 個の w に対して $t^*(w)$ をシミュレートするために使う初出の乱数は、高々 $n \times n_W$ である。したがって、$\mathcal{S}_0$ が作成した $st_{\mathcal{S}}$ から初出の乱数を与えることが可能であり、トラップドアのシミュレートを完了することができる。

実環境の $\boldsymbol{I}$ の val 成分はランダム置換 π で暗号化されているので、すべて異なるようにランダムに選んだ $\boldsymbol{I}^*$ の val 成分と区別できない。シミュレートされたトラップドア $t^*(w_i)$ の実行結果は $\boldsymbol{DB}(w_i)$ となっているので、攻撃者（$\mathcal{A}_{q+1}$）は実環境と理想環境を区別できない。したがって、$\mathcal{A}_{q+1}$ は、実環境と理想環境において同じ判定結果を出力することとなり、$\mathcal{S}$ が本節（3）ロ．の強秘匿性の定義を満たすシミュレータであるといえる。

この $\mathcal{S}$ は $\boldsymbol{D}$ や $\boldsymbol{K}$ を知らずに動作しており、$\mathcal{S}$ が $\mathcal{A}$ に漏らす情報は高々 $\mathcal{L}_1(\boldsymbol{D})$ と $\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w}_i)$（$1 \le i \le q$）となる。これらは正規のプロトコル実行時に元々 $\mathcal{A}$ が入手可能な情報であり、それらを用いて w_i を適応的に選択しても、実環境、理想環境ともに、$\boldsymbol{D}$ についての判定に有利にはならない。理想環境では $\boldsymbol{D}$ について何も判定できず（回避不可能な情報漏洩以外は漏れない）、実環境でも同様であるといえる。仮に、$\mathcal{A}_{q+1}$ が実環境で正しく判定することが可能ならば、$\boldsymbol{D}$ や $\boldsymbol{K}$ について何も知らない $\mathcal{S}$ だけがシミュレーションに失敗することとなり、理想環境で $\mathcal{A}_{q+1}$ は実環境とは異なる動作をすることになる。これは、$\mathcal{A}_{q+1}$ が実環境と理想環境とで同じ動作を行うという前提と矛盾しており、$\mathcal{A}_{q+1}$ が実環境で正しく判定することが可

能であることが偽となる。

4. アシャロフらの方式における安全性の定式化

本節では、アシャロフらの方式の構成について説明したうえで、その安全性（適応的選択キーワード攻撃に対する強秘匿性）について説明する。

(1) 検索処理におけるスケーラビリティの課題

既存の秘匿検索暗号方式においては、スケーラビリティが十分に備わっていない場合が少なくない。すなわち、文書数 n やキーワード数 n_W が非常に大きくなると、検索処理にかかる計算量が大きくなり、処理速度が非常に遅くなるという問題が知られている。これは、通常、メモリ領域上に（インデックスの）エントリが物理的に隣接せずに配置されており、各検索依頼に対して、トラップドアに対応する値を有するエントリを悉皆的に探索するために、メモリ領域への莫大な回数のアクセスが必要となることによる（これを、poor locality という）。このように、スケーラビリティを向上させるためには、キーワード検索におけるメモリ領域上でのアクセスをいかに巧みに制御するかが重要となる。

カートモラらの方式のキーワード検索フェーズの処理（Search）では、$t(w)$ の各成分と $\boldsymbol{I}$ の val 成分が一致するか否かを逐次チェックして $\boldsymbol{DB}(w)$ を求める。このとき、$\boldsymbol{I}$ のエントリ数 s は $s = n_W \times n$ であり、n と n_W が大きいと巨大な値となることから、$\boldsymbol{I}$ をチェックする処理の実行時間が大きくなると考えられる。これを念頭に、アシャロフらは、キーワード検索フェーズの処理にかかる時間の増加問題を局所性（locality）という観点で解決する構成法について検討している[19]。

局所性と読出効率（reading efficiency）は、性能評価のための尺度であり、デビッド・キャッシュ（David Cash）とステファノ・テサロ（Stefano Tessaro）によって導入された（Cash and Tessaro [2014]）。局所性とは、メモリ領域上で物理的に離れて保管されているデータをそれぞれ抽出する際に、当該データの先頭アドレスへのアクセス回数として定義される。読出効率とは、キーワード検索の依頼に対して、正しい検索結果として得られるデータのビット数に対する、メモリ領域から読み出されるデータ全体のビット数の比率である。いずれの尺度も、値が小さいほど処理性

19 Asharov *et al.* [2016] では、w を含んだ文書の識別子を $\boldsymbol{DB}(w)$ の要素とするケースと、w を含んだ文書そのものを要素とするケースを想定している。本稿では、前者のケースを対象にしている。

図表 10　領域効率・局所性・読出効率の関係

方式	領域効率	局所性	読出効率
カートモラらの方式	$O(N)$	$O(\|\boldsymbol{DB}(w)\|)$	$O(1)$
キャッシュとテサロの方式	$O(N \log N)$	$O(\log N)$	$O(1)$
アシャロフらの方式 1（ワンチョイス法）	$O(N)$	$O(1)$	$O(\log N)$
アシャロフらの方式 2（ツーチョイス法）	$O(N)$	$O(1)$	$O(\log \log N)$
理論値（Cash and Tessaro [2014]）	N の 1 次関数	$O(1)$	$O(1)$

備考：アシャロフらの方式における読出効率の向上に関する追加的な考察については補論 4 を参照。

能が相対的に高いことを表すものである。キャッシュとテサロは局所性と読出効率の関係を理論的に検討しており、それらをベースに、キーワード検索フェーズにかかる処理性能の面で優れた方式をアシャロフらは提案した（図表 10 を参照）。

(2)　局所性を高めるためのアシャロフらのアイデア

既存の秘匿検索暗号方式は、キーワードを含む文書の識別子をメモリ領域上にどのように配置するかという観点で、次の 2 つのアプローチに分類できる。第 1 のアプローチは、$\boldsymbol{DB} = (\boldsymbol{DB}(w_1), \boldsymbol{DB}(w_2), \ldots, \boldsymbol{DB}(w_{n_w}))$ が与えられたときに、$\boldsymbol{DB}$ の要素を、N 個の配列（array）のエントリに 1 つずつ一様に割り当てるというものである[20]。このとき、キーワード検索フェーズにおいて $\boldsymbol{DB}(w)$ をなるべく効率よく復元するために、$\boldsymbol{DB}(w)$ に属する文書の識別子を、次の文書の識別子の位置情報と一緒に配列に格納する。$\boldsymbol{DB}(w)$ の要素は配列に一様に対応付けられるので、$\boldsymbol{DB}(w)$ を復元するためには、サーバが $|\boldsymbol{DB}(w)|$ 回だけ配列にアクセスしなければならない。このアプローチでは、$\boldsymbol{EDB}$ を格納するために必要となるメモリ領域の大きさ（以下、領域効率）は N のオーダーで増加するほか、メモリ領域から読み出すデータの量は最適化され、読出効率は定数となる。他方、サーバはメモリ領域にランダムにアクセスすることから、局所性の観点で劣る（図表 10 のカートモラらの方式を参照）。

第 2 のアプローチは、$\boldsymbol{DB} = (\boldsymbol{DB}(w_1), \boldsymbol{DB}(w_2), \ldots, \boldsymbol{DB}(w_{n_w}))$ が与えられたときに、$\boldsymbol{DB}(w)$ の要素を、$|\boldsymbol{DB}(w)|$ 個の隣接したエントリの集合に、他の $\boldsymbol{DB}(w)$ の要素と重ならないように、一様に対応付けるというものである。$\boldsymbol{DB}(w)$ を復元する際には、サーバは、その隣接したエントリの集合の先頭アドレスに 1 回だけアクセスし、その位置に隣接する $|\boldsymbol{DB}(w)|$ 個のエントリから識別子を読み出すこととする。このアプローチでは、局所性と読出効率の観点から最適となるが、領域効率の増加が問題

20 カートモラらの方式は、このアプローチを採用している。

となる。理論的には、領域効率の下限は N に対する 1 次関数で与えられることが示されている（Cash and Tessaro [2014]）。

これらのアプローチでは、各要素の位置情報から $\boldsymbol{DB}$ の構造（要素数等）に関する情報が漏洩することとなる[21]。これを防止するためには、$|\boldsymbol{DB}(w)|$ の情報を秘匿するためのダミーのデータの詰込み（padding）が必要になる。具体的には、$|\boldsymbol{DB}(w)|$ を、$\max_{1\leq i\leq n_w}|\boldsymbol{DB}(w_i)|$ に揃えるための詰込みが必要となり、領域効率を N に対する 1 次関数で抑えられなくなってしまう。

こうしたことから、局所性、読出効率、領域効率を同時に満たす優れた方式をどう構成するかが重要な課題となる。キャッシュとテサロは、3 つの尺度すべてを同時に最適化することが不可能であることを理論的に証明したほか、局所性を、多項式から対数のオーダーに削減できる方式（図表 10 の理論値を参照）を提案した（Cash and Tessaro [2014]）。

アシャロフらの方式は、上記の 2 つのアプローチの組合せであるといえる。アシャロフらは、領域効率と局所性について最適化するとともに、カートモラらの方式における安全性と同様に、適応的選択キーワード攻撃に対する強秘匿性を証明可能である方式を提案した（Asharov *et al.* [2016]）。その方式のアイデアは、$\boldsymbol{DB}(w)$ の構造をある程度メモリ領域上に保存しつつ、$\boldsymbol{DB}(w)$ の文書の識別子を隣接したエントリの集合に直接対応付けるのではなく、隣合う識別子の間に、他のキーワード w' に対応する文書の識別子をランダムに挟み込むというものである。ランダムに挟み込むという意味での柔軟性をもたせることで、読出効率を若干犠牲にしつつも、安全性（強秘匿性）を証明可能とするとともに、領域効率の増加を線形に抑え、かつ、局所性を最適化している（図表 10 のアシャロフらの方式 1 および 2 を参照）。

(3) アシャロフらの構成法

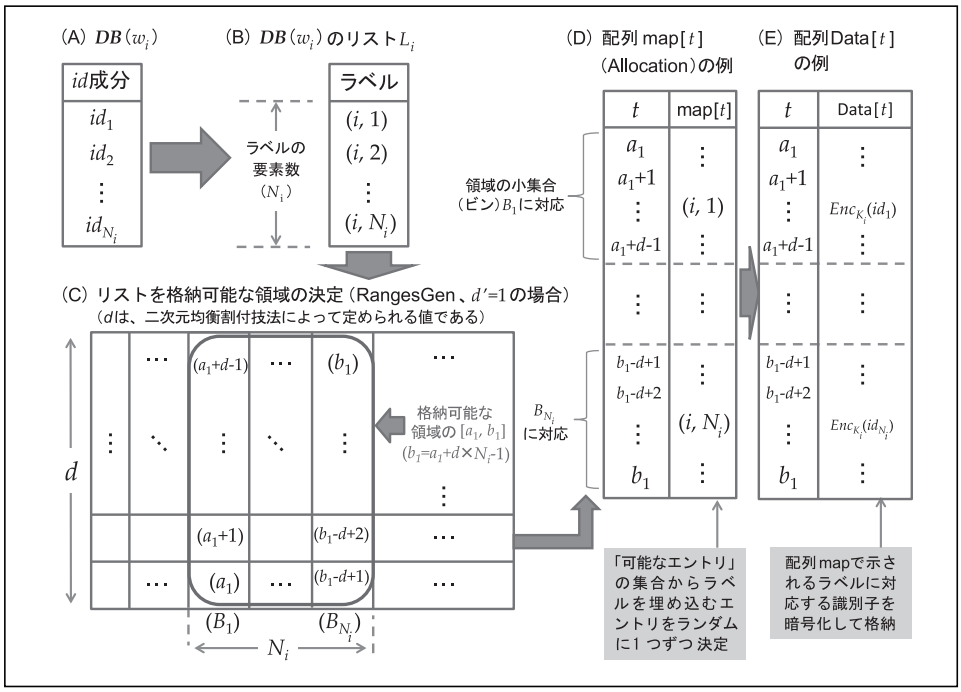
イ. 割振りアルゴリズム

データベース $\boldsymbol{DB}(w)$ のメモリ領域への配置は、まず、①メモリ領域（配列で表現される）のうち、識別子を格納可能な位置を特定したうえで、②上記①で特定したメモリ領域のうち、どこにどの識別子を格納するかを決定し、③最後に、上記②で決定したとおりに識別子をそれぞれ格納するという流れとなる。これらのうち、上記①と②は割振りアルゴリズムと呼ばれている。

割振りアルゴリズムは、文書の識別子を特定するラベル（例えば、*id* の添え字の

21 w ごとの文書数 $|\boldsymbol{DB}(w)|$ は、検索後にサーバに漏洩することが許容されるが、検索前に漏洩することは許されない。

図表 11　割振りアルゴリズムの動作イメージ



(i, j)）の集合（以下、リスト）の要素数 N_i（識別子の数と同じ）が与えられたときに、各ラベルを格納する位置を指定する配列 map を出力するものである[22]。メモリ領域上のある位置がラベルを格納しうるときには、それを、可能な（possible）エントリと呼び、実際にラベルを格納したときには、それを、現実（actual）のエントリと呼ぶことにする。

割振りアルゴリズムは RangesGen と Allocation から構成される（図表 11 を参照）。RangesGen は、N_i に対して d' 個の領域の集合 R_i を出力する[23]。Allocation は、n_W 組の (R_i, N_i) に対して配列 map を出力する。割振りアルゴリズムの入力 $(N_1, N_2, \ldots, N_{n_W})$ は、各リストの要素数を表す n_W 個の整数値であり、$N = \sum_{i=1}^{n_W} N_i$ とする。そのうえで、以下の 1 から 3 の処理を行う。

1. $1 \le i \le n_W$ を満たす i に対して、$R_i \leftarrow \mathrm{RangesGen}(N, N_i)$ を生成する。
   $\mathrm{RangesGen}(N, N_i)$ は、第 i 番目のリストを格納することが可能な領域とし

22 配列 map のエントリをパラメータ t で指定する。例えば、map[t] によって、配列 map の第 t 番目のエントリを表す。

23 d' の値については、割振りアルゴリズムのベースとなっている二次元均衡割付技法によって特定され、N を変数とする関数で与えられる。具体的には、「ボールとビン」ゲームに基づいて特定される（補論 2 を参照）。ワンチョイス法では $d' = 1$ であり、ツーチョイス法では $d' = 2$ である。

て $R_i = \{[a_1, b_1], [a_2, b_2], \ldots, [a_{d'}, b_{d'}],\}$ を出力する（図表 11 の（C）に対応）。$[a_i, b_i]$ はメモリ領域上の「a_i」で示される位置から「b_i」で示される位置までの間の領域を示す[24]。

2. map ← Allocation$((N_1, R_1), (N_2, R_2), \ldots, (N_{n_W}, R_{n_W}))$ を生成する。
   配列 map は、リストが格納される位置（actual location）を情報として持つ。すなわち、配列 map の各エントリには、第 i 番目のリストの第 j 番目のラベルを表すデータ (i, j)、あるいは、空のエントリを表す NULL のいずれかが埋め込まれる[25]。
3. 配列 map を出力する。

ロ．割振りアルゴリズムを用いた方式の構成

割振りアルゴリズムが与えられたとき、アシャロフらが提案した秘匿検索暗号方式を構成できる一般的な変換方法（generic transformation）の概要を説明する（図表 12 を参照）。

文書蓄積フェーズでは、ユーザはキーワードの集まり $\boldsymbol{W}$ と文書の集まり $\boldsymbol{D}$ からリストの集合 $\{L_i\}_{1 \le i \le n_W}$ を作成し（図表 11（A）（B）に対応）、RangesGen と Allocation を用いて配列 map を作成する。map[t] = (i, j) のとき、$\boldsymbol{DB}(w_i)$ の第 j 番目の文書の識別子 $id_{i,j}$ を暗号化して Data[t] に埋め込む（図表 11（E）に対応）[26]。map[t] = NULL のとき、Data[t] に乱数を詰め込む。$\boldsymbol{DB}(w_i)$ のサイズ N_i をサーバに伝えるためにハッシュ表 HT を生成する[27]。

キーワード検索フェーズでは、ユーザは、TokenGen によって、キーワード w に対応するトラップドア τ_{w_i} を疑似ランダム関数 $\mathrm{PRF}_K(w_i)$ によって生成する（図表 13 を参照）。$\tau_{w_i} = (\ell_i, k_i, r_i)$ に含まれる ℓ_i と k_i は、HT と組み合わせて $N_i = |\boldsymbol{DB}(w_i)|$ を復号するために用いられる。残りの r_i は配列 map の生成で使われる。

24 この領域は、二次元均衡割付技法によって定まる値 d（補論 2（1）の深さの値に対応する）のもとで、最大で $d \times N_i$ 個の識別子を格納可能なように求められる。したがって、$b_i = a_i + d \times N_i$ となる。

25 この処理をやや詳しく説明すると、次のとおりである。領域 $[a_i, b_i]$ は、N_i 個の領域の小集合（例えば、図表 11（C））における N_i 個の列に分割され、識別子のラベルは、各小集合に対応するエントリの集合に対して、1 つずつ割り振られることとなる（図表 11（D）を参照）。領域の小集合は、それぞれ、d 個のエントリからなるが、これらのうちの可能なエントリのなかの 1 つにラベルが埋め込まれる。どのエントリにラベルが埋め込まれるかはランダムに決定される。残りのエントリには NULL が埋め込まれる。このエントリのランダムな決定が、二次元均衡割付技法における「ボールとビン」ゲームでの（ビン内部の）ボールのシャッフルに対応する。

26 $\boldsymbol{DB}(w_i)$ に属する文書の識別子は、疑似ランダム関数 PRF_K を用いてキーワード w_i から求めた鍵 K_i で暗号化する。なお、暗号化に用いられる暗号方式は、疑似ランダムな暗号文を出力し、判別不可能で効率的に検証可能な値域集合（elusive and efficiently verifiable range）を持つものとする。Asharov *et al.* [2016] によれば、このような暗号方式は、疑似ランダム関数から容易に構成できる。

27 HT については、補論 1（3）ハ．を参照されたい。

図表 12　割振りアルゴリズムを用いた方式の構成法

パラメータ：キーワードの集合を $W = \{w_1, w_2, \ldots, w_{n_W}\}$、識別子のデータベースを $DB = \{DB(w_1), DB(w_2), \ldots, DB(w_{n_W})\}$ とする。また、$N = \sum_{i=1}^{n_W} |DB(w_i)|$、$N_i = |DB(w_i)|$ とする。暗号鍵の生成については、アルゴリズム $\mathtt{KeyGen}(1^\lambda)$ を用いて疑似ランダム関数 PRF の鍵 K をサンプリングして出力する。

ユーザによる文書蓄積：アルゴリズム $\mathtt{EDBSetup}(DB, K)$ が次のように動作する。

1. 集合 S を空集合に初期化する。
2. キーワード $w_i \in W$ に対して以下を実行する。
   (a)　$DB(w_i) = \left\{id_{i,1}, id_{i,2}, \ldots, id_{i,N_i}\right\}$ を構成する。
   (b)　$(\ell_i, k_i, r_i, K_i) = \mathrm{PRF}_K(w_i)$ を計算する。
   (c)　$n_i = N_i \oplus k_i$ を計算する。
   (d)　$R_i = \mathtt{RangesGen}(N, N_i; r_i)$ を計算する。
3. N 個の乱数要素の組を S に詰め込んだ後、S を一様にシャッフルして、$\mathtt{HT} \leftarrow \mathtt{HTSetup}(S)$ を計算する。
4. $\mathtt{map} \leftarrow \mathtt{Allocation}(\{(N_i, R_i)\}_{i=1}^{n_W})$ を計算する。$\mathtt{map} = \perp$ のときには $\perp$ を出力する。その他の場合には、サイズ $s(N)$ のデータ・ブロック Data を、$1 \le t \le s(N)$ となる t に対して次のように定義する。

$$\mathtt{Data}[t] = \begin{cases} \mathrm{Enc}_{K_i}(id_{i,j}) & \mathtt{map}[t] = (i, j) \text{ の場合} \\ U_\ell & \text{その他の場合。} \end{cases}$$

   U_ℓ は、エントリごとに一様独立にサンプリングした ℓ-ビット列を表す。
5. $EDB = (\mathtt{Data}, \mathtt{HT})$ を出力する。

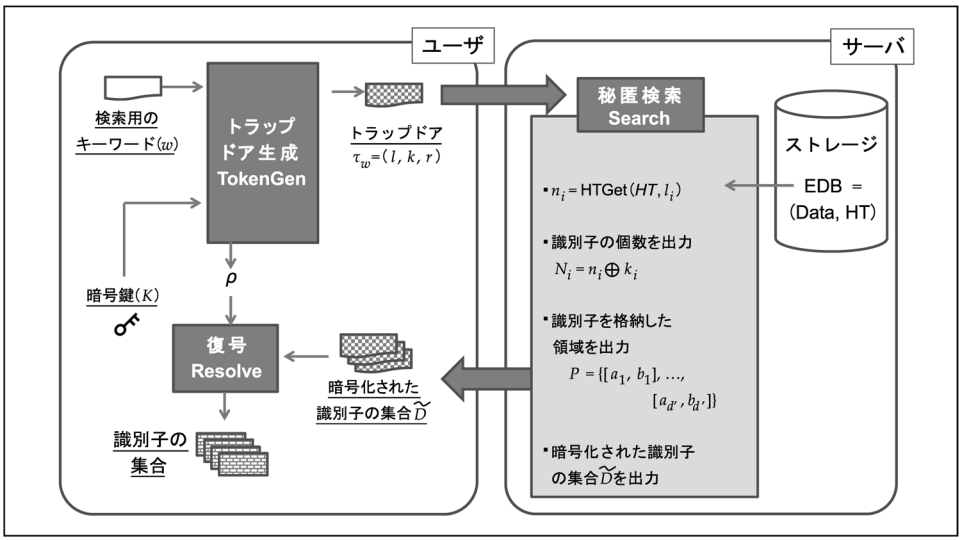
ユーザによるトラップドア生成：アルゴリズム $\mathtt{TokenGen}(K, w_i)$ が、トラップドア $\tau_i = (\ell_i, k_i, r_i)$ を生成するとともに、アルゴリズム Resolve によって用いられる $\rho_i = K_i$ を出力する。

サーバによるキーワード検索：アルゴリズム $\mathtt{Search}(\tau_i, EDB)$ が、$\tau_i = (\ell_i, k_i, r_i)$ と $EDB = (\mathtt{Data}, \mathtt{HT})$ を用いて次の処理を行う。

1. $n_i \leftarrow \mathtt{HTGet}(\mathtt{HT}, \ell_i)$ を生成し、$N_i = n_i \oplus k_i$ を計算する。
2. $\mathtt{RangesGen}(N, N_i; r_i)$ を計算し、暗号化された識別子を格納した領域として $P = \{[a_1, b_1], [a_2, b_2], \ldots, [a_{d'}, b_{d'}]\}$ を出力する。
3. 暗号化された識別子の集合として $\widetilde{D} = \bigcup_{[a_j,\ b_j] \in P} \mathtt{Data}[a_j, \ldots, b_j]$ を出力する。ただし、$\mathtt{Data}[a_j, \ldots, b_j] = \bigcup_{t=a_j}^{b_j} \mathtt{Data}[t]$ である。

ユーザによる識別子の復号：アルゴリズム $\mathtt{Resolve}(\rho_i, \widetilde{D})$ が、鍵 K_i を使って $\widetilde{D}$ に属する要素（識別子）を復号し出力する。

図表 13　キーワード検索フェーズの動作（アシャロフらの方式）



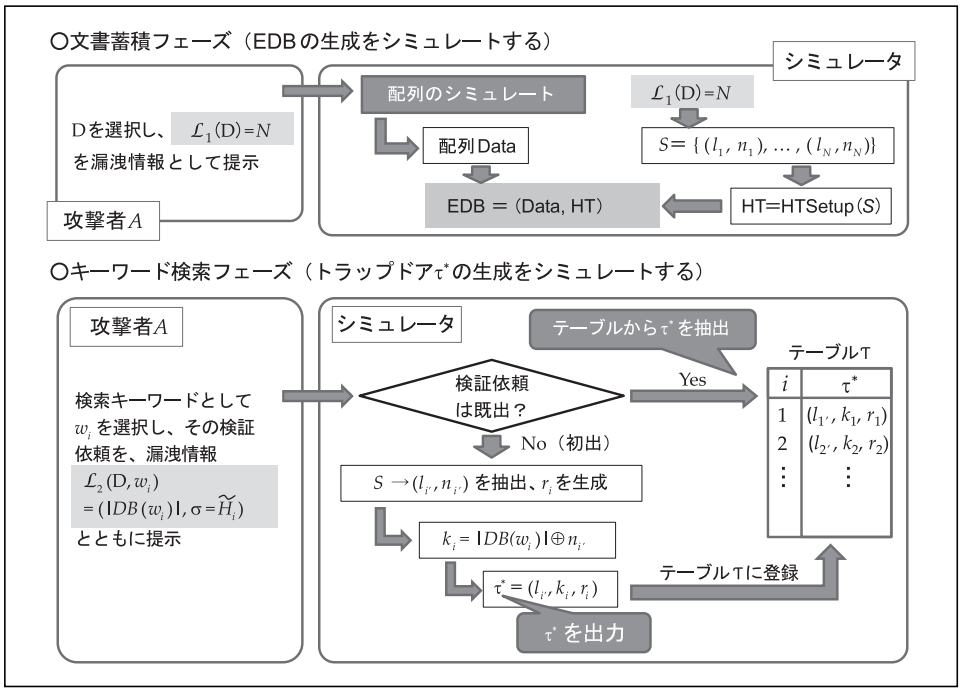
サーバは、$\mathtt{Search}(\tau_i, \boldsymbol{EDB})$ によって ℓ_i、k_i、HT から N_i を復号し、$\mathtt{RangesGen}(N, N_i; r_i)$ によって $P = \{[a_1, b_1], [a_2, b_2], \ldots, [a_{d'}, b_{d'}]\}$ を入手する[28]。なお、サーバは、乱数 r を用いて、w に対応するリストのラベルが格納されているエントリのみを出力する。このとき、他のリストで用いる乱数を知らないので、他のリストのラベルが格納されているエントリに関する情報を入手することができない。

サーバは、検索結果として、P に属する $\mathtt{Data}[t]$ の全体（暗号化された識別子と乱数）を $\widetilde{\boldsymbol{D}}$ として回答する。ユーザは、Resolve を用いて、キーワード w から求めた暗号鍵 K_i によって $\mathtt{Data}[t] \in \widetilde{\boldsymbol{D}}$ をすべて復号し、キーワードを含む文書の識別子を入手する。

暗号化された文書の識別子は、$\mathtt{Data}[t]$ のいずれかの領域に格納されているので、$\boldsymbol{DB}(w)$ は正しく復号される。暗号方式は、判別不可能な値域集合と疑似ランダムな暗号文を有することから、サーバが鍵 K を知ったとしても、他の鍵 $K' = h(w')$ を用いて暗号された $\mathtt{Enc}_{K'}(id')$ を K で復号することはできず、$\mathtt{Enc}_{K'}(id')$ はランダムにみえる。また、$\mathtt{Data}[t]$ に埋め込まれた乱数 $rand$ が $rand = \mathtt{Enc}_K(id')$ を満たすことがあるとしても、それは偶然でしかなく、$\boldsymbol{DB}(w)$ として誤った出力を生じる確率は無視できるほど小さい。したがって、キーワード検索フェーズと復号の処理は正しく動作するといえる。

28　P を得ることは、二次元均衡割付技法における「ボールとビン」ゲームにおける方式 1（ワンチョイス法）の場合、先頭が $h(w)$ で指定されたビン B_α（深さ d）を特定することに対応する。方式 2（ツーチョイス法）の場合、先頭が $h_1(w)$ と $h_2(w)$ で指定されたビン B_{α_1} とビン B_{α_2}（深さ d）を特定することに対応する。

図表 14　***EDB*** 生成とトラップドア生成のシミュレーション（アシャロフらの方式）



(4)　アシャロフらの方式における安全性証明

上記の構成法に基づくアシャロフらの 2 つの方式は、適応的選択キーワード攻撃に対して強秘匿性を有することが証明可能である。漏洩関数を $\mathcal{L}_1(\boldsymbol{D}) = N$、$\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w}_i) = (|\boldsymbol{DB}(\boldsymbol{w}_i)|, \sigma = \widetilde{H}_i)$ と定義すると、次の定理 1 が成り立つ[29]。

定理 1.　疑似ランダム関数を PRF、疑似ランダムな暗号文を出力して判別不可能かつ効率的に検証可能な値域集合（elusive and efficiently verifiable range）を有する共通鍵暗号方式を $(\mathsf{Enc}, \mathsf{Dec})$ とする。このとき、アシャロフらの構成法は、漏洩関数 $\mathcal{L}_1(\boldsymbol{D}) = N$ と $\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w}) = (|DB(w_i)|, \sigma = \widetilde{H}_i)$ のもとで、適応的選択キーワード攻撃に対する強秘匿性を満たす。

以下では、3 節（3）ハ. と同様に、シミュレータ $\mathcal{S}$ によって実環境での攻撃をどう模倣できるかに注目して、安全性証明の概要を説明する（図表 14 を参照）。

29　Asharov *et al.* [2016] では、$\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w}_i) = |\boldsymbol{DB}(\boldsymbol{w}_i)|$ と読み取れるような記述がある。この定義では、検索依頼が初出か既出かをシミュレータが判断できなくなり、実環境と理想環境で異なる動作が生じてしまうため、安全性を証明できなくなると考えられる。

文書蓄積フェーズにおいて、$\mathcal{S}_0$ は、$\mathcal{L}_1(\boldsymbol{D}) = N$ を入手したうえで、適切な長さのランダムなビット列からなる要素を $s(N)$ 個有する配列 Data を生成する。次に、N 組のランダムな $(\ell_{i'}, n_{i'})$ を要素とする集合 S を生成する。さらに、S をシャッフルして $\mathtt{HT} := \mathtt{HTSetup}(S)$ を計算し、$\boldsymbol{EDB} = (\mathtt{Data}, \mathtt{HT})$ を出力する。

キーワード検索フェーズにおいては、攻撃者 $\mathcal{A}_i$ が発行した検索依頼に対して、$\mathcal{S}_i$ は、$\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w}_i)$ を用いて、検索依頼が初出か既出かを検証する[30]。検索依頼が初出の場合、$\mathcal{S}_i$ は S から $(\ell_{i'}, n_{i'})$ を選び、S から消去する。次に $\mathcal{S}_i$ は、鍵成分 $k_i := |DB(w_i)| \oplus n_{i'}$ を計算する[31]。ここで、$|DB(w_i)|$ は $\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w}_i)$ から与えられる。さらに、$\mathcal{S}_i$ は、ランダムに r_i を選び、シミュレートしたトラップドアとして $\tau_i^* = (\ell_{i'}, k_i, r_i)$ を出力して、テーブル T に (i, τ_i^*) を登録する。検索依頼が既出の場合、T から一致するものを読み出して τ_i^* として出力する。

ここで、カートモラらの方式と同様に、シミュレータは w_i を知ることができないものの、検索依頼の初出と既出の判断は、$\sigma = \widetilde{H}_i$ を用いることによって可能である[32]。すなわち、$1 \le \forall j \le i-1$ の j に対して $\widetilde{H}_i(j, i)$ を計算し、一致する j の存在を調べることで、w_i の初出と既出を判定できる。

実環境と理想環境においてそれぞれ生成された $\boldsymbol{EDB}$ と $\tau = \{\tau_i\}_{w_i \in \boldsymbol{w}}$ を識別できないことは、PRF が疑似ランダム関数であること、および、$(\mathtt{Enc}, \mathtt{Dec})$ が判別不可能かつ効率的に検証可能な値域集合を有する共通鍵暗号方式であることから示すことができる[33]。

30 $\mathcal{S}_i$ は w_i を直接知ることができないことに注意する。

31 実環境では、$n_i := |DB(w_i)| \oplus k_i$ で N_i の暗号文 (n_i) を計算しているが、理想環境では、$k_i := |DB(w_i)| \oplus n_i$ として暗号鍵を計算し、いわば、後付けで辻褄をあわせるトリックで、安全性証明に成功している。

32 サーチ関数 $\widetilde{H}_i$ は、Curtmola *et al.* [2006] で定義されたものとは異なっている。$\widetilde{H}_i$ は、キーワード検索を実行したときにサーバが観測できる情報を示すものであり、サーバからの回答 $\widetilde{\boldsymbol{D}}$ を含む。

33 $\boldsymbol{EDB}$ と $\tau = \{\tau_i\}_{w_i \in \boldsymbol{w}}$ を、識別できないことから、カートモラらの方式にかかる安全性評価のケースと同様に、攻撃者は、双方の環境において同じ判定結果を出力することとなり、実環境において理想環境よりも判定に有利にならないといえる。アシャロフらの方式 1 と 2 において鍵 K_i が漏洩した場合等の安全性について、やや詳しくみると、次のとおりである。まず、方式 1 では、暗号鍵 K_i が漏洩したとしても、他のリストに属する文書の識別子が格納されている領域の情報が漏洩しない。そのため、サーバが観察するアクセス・パターン（異なるキーワード w' に対する $h(w')$）からは、データベースについて余計な情報が漏洩しないといえる。また、方式 2 は、他のリストに属する文書の識別子が格納された可能性がある 2 つの位置が互いに独立となっているほか、サーバに K_i が与えられておらず、暗号化された識別子を復号できない。したがって、サーバが観察するアクセス・パターン（異なるキーワード w' に対する $h_1(w')$ と $h_2(w')$）からは、データベースについて余計な情報が漏洩しないといえる。

5. 主な技術的課題

(1) 回避不可能な情報漏洩にかかる厳密な評価

3節と4節では、カートモラらの方式とアシャロフらの方式において、サーバに対するデータの安全性（すなわち、情報漏洩の度合い）の意味やそれを示すためのモデル、安全性証明の概略等についてそれぞれ説明した。いずれの方式も、各モデルのもとで、回避不可能な情報漏洩よりも多くの情報が漏洩するか否かという点について、攻撃者であるサーバが、自分でキーワードを適応的に選択しつつ、何らかの情報を入手しようとしても、それが困難である（適応的選択キーワード攻撃に対する強秘匿性を有する）ことが証明可能である。

もっとも、これらの方式では、回避不可能な情報漏洩にかかるモデルが異なっており、サーバへの情報漏洩の内容が異なっている点に留意が必要である。カートモラらの方式では、漏洩が回避不可能な情報として、トレース $\tau(H) := (|D_1|, |D_2|, \ldots, |D_n|, \alpha(H), \sigma(H))$ が定義されている。これは、履歴（サーバとユーザとの間でやり取りされる文書の集まりと一連のキーワード）、アクセス・パターン（検索結果となる文書の識別子）、サーチ・パターン（異なるタイミングで用いられたトラップドアが一致するか否か）から構成されている。一方、アシャロフらの方式では、情報漏洩のタイミングを文書蓄積フェーズとキーワード検索フェーズに分けたうえで、それぞれの情報漏洩を $\mathcal{L}_1(\boldsymbol{D})$ と $\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w})$ によって表すという方法を採用している。2つの方式における回避不可能な情報漏洩の差異は、キーワード検索フェーズの処理の結果として漏洩する識別子 $\boldsymbol{DB}(\boldsymbol{w})$ において発生し、カートモラらの方式では、$\boldsymbol{DB}(\boldsymbol{w})$ そのものが漏洩するのに対して、アシャロフらの方式では、そのサイズ $|\boldsymbol{DB}(\boldsymbol{w})|$ が漏洩するのに過ぎない。この結果、サーバに漏洩する情報は、アシャロフらの方式において相対的に少ない。

このように、適応的選択キーワード攻撃に対する強秘匿性が証明可能であったとしても、トレースや漏洩関数のパラメータによっては、安全性（ここでは機密性を意味する）のレベルが変化することがある。複数の秘匿検索暗号方式を安全性の観点から比較する際には、回避不可能な漏洩情報に着目し、その差異を確認したうえで、それが実用上問題となるか否か（アプリケーションのセキュリティ要件に影響するか否か）について検討することが求められるといえる。

秘匿検索暗号を提案する側に関していえば、今後、新しい実現方式を提案する際にシミュレーションに基づくモデルによって安全性を評価する場合には、漏洩関数を厳密に評価するとともに、それらによって達成される安全性のレベルを論文等に

明記すべきであろう。

また、カートモラらの方式とアシャロフらの方式においては、同一のキーワードに対して毎回同一のトラップドアが生成されることから、サーチ関数 σ が、回避不可能な情報漏洩の一部として定義されている。一方、最近では、確率的なトラップドアを実現する方式が提案されるようになってきており、そうした方式を今後提案する際には、サーチ関数の定義を見直す必要があると考えられる。

(2) キーワードの集合にかかる拡張性

本稿では、Asharov *et al.* [2016] の記法を採用し、$\boldsymbol{W}$ を n_W 個のキーワードから構成された集合と定義したうえで、$\boldsymbol{W}$ 上の秘匿検索暗号方式を対象に考察した。一方、Curtmola *et al.* [2006] は、キーワードのすべての候補を Δ と定義して辞書と呼び、Δ 上の方式を提案している[34]。

現状では、$\boldsymbol{W}$ 上の方式の研究が中心となっているが、$\boldsymbol{W}$ を予め限定しない Δ 上の方式について検討することも、キーワードに関する拡張性を実現するうえで必要であると考えられる。秘匿検索暗号方式を実際に利用する場合、取り扱う文書が増加するにつれて、キーワードも増加していくことが想定される。したがって、キーワードの集合を最初に固定しなければならない方式は、拡張性の観点で望ましくないと考えられる。また、文書の内容も多様化し、日本語だけの文書から外国語の単語が含まれる文書も増えてくるとすれば、そうした外国語の単語をキーワードにしたいというニーズも高まってくると考えられる。

こうしたニーズに対応するために、例えば、辞書に関して、$\Delta^{(1)} \subset \Delta^{(2)} \ldots$ と拡張する機能を実現する方式や、キーワードの集合について、$\boldsymbol{W}^{(1)} \subset \boldsymbol{W}^{(2)} \ldots$ と拡張する機能を実現する方式の研究が重要となる。安全性を確保しつつ、こうした機能を実現する方式の研究開発の進展が期待される[35]。

(3) 文書の集合にかかる拡張性

ネットワーク上で生成されるデータの量が拡大するとともに、それらをクラウド

34 このとき、文書 D はキーワードだけから構成されることになる。

35 もっとも、現時点では、Δ について拡張性を有する方式の提案は容易でないと考えられる。例えば、Curtmola *et al.* [2006] において、キーワードのサイズを決定するために、最大のサイズとなる文書が持ちうるキーワードの種類を数えたうえで、それをキーワードのサイズとするアイデアが示されている。しかし、最大のサイズとなる文書が持ちうるキーワードの種類をどう決定するかという問題が存在する。

等のストレージに保管して有効活用するというニーズが高まるなかで、実用上の観点から、文書の集まり $\boldsymbol{D}$ を固定して秘匿検索暗号方式を利用し続けることは想定しづらい。通常、ユーザがクラウド上で秘密に登録・保管したいと考える文書の数は増加すると見込まれることから、$\boldsymbol{D}^{(1)} \subset \boldsymbol{D}^{(2)} \subset \ldots$ と、秘匿検索暗号方式の対象となる文書の増加に対応する機能の実現が求められると考えられる。また、登録・保管された文書を削除する機能も求められるであろう。

例えば、$\boldsymbol{D}^{(2)}$ を前提に生成されたトラップドア（$t^{(2)}(w)$）によって $\boldsymbol{D}^{(1)}$ も検索対象にすることができれば、ユーザは、最新の文書の集まりにかかるトラップドアを生成するための鍵のみを管理すればよく、運用上の負担が軽減されることになる。こうした機能を有する秘匿検索暗号は、ダイナミック SSE（Searchable Symmetric Encryption）と呼ばれており、今後、一般的な構成法の開発が期待される。また、同時に、こうした機能を付加することに伴う新たな情報の漏洩の可能性についても配慮し、安全性にかかるモデルの改良等についても検討を行うことが重要であろう。

参考文献

海上勇二・松崎なつめ・山田翔太・アッタラパドゥンナッタポン・松田隆宏・花岡悟一郎、「ユークリッド距離に基づく類似検索可能暗号」、『2016 年暗号と情報セキュリティシンポジウム予稿集』、電子情報通信学会、2016 年

金融情報システムセンター、「平成 27 年度金融機関アンケート調査結果」、『金融情報システム』増刊 79 号 No. 338、金融情報システムセンター、2015 年

黒澤　馨・佐々木圭佑・太田清比古・米山一樹、「UC 安全性を満たす効率的で動的な検索可能暗号」、『2016 年暗号と情報セキュリティシンポジウム予稿集』、電子情報通信学会、2016 年

小暮　淳・下山武司・安田雅哉、「暗号化したまま検索が可能な秘匿検索技術」、『電子情報通信学会誌』Vol. 98 No. 3、電子情報通信学会、2015 年、202～206 頁

小嶋陸大・品川和雅・金山直樹・西出隆志・岡本栄司、「共通鍵完全準同型暗号を用いた安全なブルームフィルタ」、『2016 年暗号と情報セキュリティシンポジウム予稿集』、電子情報通信学会、2016 年

清藤武暢・四方順司、「高機能暗号を活用した情報漏えい対策『暗号化状態処理技術』の最新動向」、『金融研究』第 33 巻第 4 号、日本銀行金融研究所、2014 年、97～132 頁

日本電気、「世界初、データベースを暗号化したまま処理できる秘匿計算技術を開発～情報セキュリティ対策の強化に貢献～」、日本電気、2013 年（http://jpn.nec.com/press/201311/20131106_01.html、2017 年 9 月 27 日）

日立ソリューションズ、「SharePoint Online 上の重要情報を守るセキュリティ強化ソリューションを提供開始　クラウド上にアップロードされる情報をすべて暗号化し、第三者による不正参照を防止」、日立ソリューションズ、2016 年（http://www.hitachi-solutions.co.jp/company/press/news/2016/0829.html、2017 年 9 月 27 日）

平野貴人・川合　豊・太田和夫・岩本　貢、「共通鍵暗号型の秘匿部分一致検索（その 1）」、『2016 年暗号と情報セキュリティシンポジウム予稿集』、電子情報通信学会、2016 年

富士通研究所、「暗号化したまま検索が可能な秘匿検索技術を開発」、富士通研究所、2014 年、（http://pr.fujitsu.com/jp/news/2014/01/15.html、2017 年 9 月 27 日）

三菱電機、「『部分一致対応秘匿検索基盤ソフトウェア』を開発」、三菱電機、2016 年（http://www.mitsubishielectric.co.jp/news/2016/0204.html、2017 年 9 月 27 日）

NTT テクノクロス、「クラウド向け 情報漏えい対策・データ保護ソリューション TrustBind/Secure Gateway」、NTT テクノクロス、2013 年（https://www.ntts.co.jp/products/trustbind/sgw/、2017 年 9 月 27 日）

Asharov, Gilad, Moni Naor, Gil Segev, and Ido Shahaf, “Searchable Symmetric Encryp-

tion: Optimal Locality in Linear Space via Two-Dimensional Balanced Allocations," *Proceedings of the 48th Annual ACM Special Interest Group on Algorithms and Computation Theory Symposium on Theory of Computing*, Association for Computing Machinery, 2016, pp. 1101–1114.

Cash, David, and Stefano Tessaro, "The Locality of Searchable Symmetric Encryption," *Advances in Cryptology—EUROCRYPT 2014*, Lecture Notes in Computer Science, 8441, Springer-Verlag, 2014, pp. 351–368.

Chase, Melissa, and Seny Kamara, "Structured Encryption and Controlled Disclosure," *Advances in Cryptology—ASIACRYPT 2010*, Lecture Notes in Computer Science, 6477, Springer-Verlag, 2010, pp. 577–594.

Curtmola, Reza, Juan Garay, Seny Kamara, and Rafail Ostrovsky, "Searchable Symmetric Encryption: Improved Definitions and Efficient Constructions," *Proceedings of the 13th ACM Conference on Computer and Communications Security*, Association for Computing Machinery, 2006, pp. 79–88.

Gajek, Sebastian, "Dynamic Symmetric Searchable Encryption from Constrained Functional Encryption," *Topic in Cryptology—CT-RSA 2016*, Lecture Notes in Computer Science, 9610, Springer-Verlag, 2016, pp. 75–89.

Hayasaka, Kenichiro, Yutaka Kawai, Yoshihiro Koseki, Takato Hirano, Kazuo Ohta, and Mitsugu Iwamoto, "Probabilistic Generation of Trapdoors: Reducing Information Leakage of Searchable Symmetric Encryption," *Cryptology and Network Security—CANS 2016*, Lecture Notes in Computer Science, 10052, Springer-Verlag, 2016, pp. 350–364.

Yavuz, Attila A., and Jorge Guajardo, "Dynamic Searchable Symmetric Encryption with Minimal Leakage and Efficient Update on Commodity Hardware," *Proceedings of SAC 2015*, Lecture Notes in Computer Science, 9566, Springer-Verlag, 2015, pp. 241–259.

補論 1.　記法や用語等の基本的な概念

(1)　記法

正の実数値の集合を $\mathbb{R}^+$ と記述する。また、関数 $\mathtt{negl} : \mathbb{N} \to \mathbb{R}^+$ が無視可能（negligible）とは、任意の正値多項式 p と $n \geq n_0$ なるすべての n に対して、$\mathtt{negl}(n) < 1/p(n)$ を満たす整数 $n_0 \in \mathbb{N}$ が存在することを意味するものとする。n についての不特定の多項式を $\mathtt{poly}(n)$ と記述する。

ある事象が少なくとも確率（$1 - \mathtt{negl}(n)$）で成り立つときに、圧倒的な確率で成り立つという。

確率的アルゴリズム $\mathcal{A}$ が、一様にランダムな乱数テープを用いて入力 x によって実行されるときに、出力が y となることを、$y \leftarrow \mathcal{A}(x)$ と表す。$\mathcal{A}^{\mathcal{O}}$ は、$\mathcal{A}$ が $\mathcal{O}$ にオラクル・アクセスして実行することを表す[36]。

S が有限集合ならばその要素数を $|S|$ で表すほか、S がビット列ならばそのビット長を $|S|$ と表す。また、S が有限集合の場合、S から一様に s を選ぶことを $s \xleftarrow{u} S$ と表す。

(2)　用語

本稿における秘匿検索暗号の説明の用語については、Asharov *et al.* [2016] を採用する。

キーワードを w、n_W 個のキーワードから構成された集合を $W \subseteq \{0,1\}^{\ell}$ とする[37]。キーワードにおいて、特に順序に注目する場合には、集まり（collection）と呼び、$\boldsymbol{W} = (w_1, w_2, \ldots, w_{n_W})$ と表す。文書を $D \in \{0,1\}^*$、n 個の文書の集まりを $\boldsymbol{D} = (D_1, D_2, \ldots, D_n)$ で表す。

D_i の暗号文を $C_i (1 \leq i \leq n)$ とするとともに、$\boldsymbol{C} = (C_1, C_2, \ldots, C_n)$ によって $\boldsymbol{D}$ の暗号文の集まりを表す。暗号文 C_i は（一意の）ν ビットの識別子（unique identifier）として $id_i \in \{0,1\}^{\nu}$ を持つとする。キーワード w_i を含む文書の識別子の集合を $DB(w_i) = \{id_{i_1}, id_{i_2}, \ldots, id_{i_{N_i}}\}$ と表し、$DB(w_i)$ の要素数を N_i （$\leq n$）とするほ

36 オラクル・アクセスするとは、次のような性質を持つ理想的なハッシュ関数に問合せすることである。①過去に照会のなかった問合せには、完全にランダムな答えを返す。②過去に照会のあった問合せには、必ず前回と同一の答えを返す。

37 カートモラらは、キーワードのすべての候補を Δ で定義して辞書と呼んで $\boldsymbol{D} \subseteq 2^{\Delta}$ と仮定している。これにより、$\forall D \in \boldsymbol{D}$ はキーワードだけから構成されることになる。D が同じキーワードを複数回含むこともある。

か、$N = \sum_{i=1}^{n_W} N_i$ とする[38]。

順序に注目するときには $\boldsymbol{DB}(w_i) = (id_{i_1}, id_{i_2}, \ldots, id_{i_{N_i}})$ と表す。$\boldsymbol{W}$ に属するすべてのキーワードに対する DB の集まりを $\boldsymbol{DB} = (\boldsymbol{DB}(w_1), \boldsymbol{DB}(w_2), \ldots, \boldsymbol{DB}(w_{n_W}))$ と表してデータベースと呼ぶ。

カートモラらは、文書 D に含まれるキーワードの集合を $\delta(D)$、文書の集まり $\boldsymbol{D}$ に含まれるキーワードの集合を $\delta(\boldsymbol{D})$ と表している。

(3) 暗号学的な基本概念等 (Cryptographic Primitives)

イ. 疑似ランダム性

関数 $f : \{0,1\}^\lambda \times \{0,1\}^m \to \{0,1\}^n$ が疑似ランダム性（pseudo-randomness）を満たすとは、f がセキュリティ・パラメータ λ について多項式時間で計算可能であり、かつ、任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対して、以下の不等式が成り立つことを意味する。

$$|\Pr[1 \leftarrow \mathcal{A}^{f(K,\cdot)}(1^\lambda) \mid K \xleftarrow{u} \{0,1\}^\lambda] - \Pr[1 \leftarrow \mathcal{A}^{g(\cdot)}(1^\lambda) \mid g \xleftarrow{u} \mathcal{F}_{m,n}]| \leq \mathtt{negl}(\lambda).$$

ここで、$\mathcal{F}_{m,n}$ は $\{0,1\}^m$ から $\{0,1\}^n$ への写像の集合である。

また、3 つのアルゴリズム Gen、Enc、Dec から構成される共通鍵暗号方式 $\mathtt{SKE} = (\mathtt{Gen}, \mathtt{Enc}, \mathtt{Dec})$ が、選択平文攻撃に対して疑似ランダム性を満たすこと（pseudo-randomness against chosen plaintext attacks: PCPA）を、以下のとおり定義する。

定義 4（選択平文攻撃に対する疑似ランダム性）. 任意の攻撃者 $\mathcal{A}$ と SKE に対する次の実験を $\mathbf{PCPA}_{\mathcal{A},\mathtt{SKE}}(\lambda)$ とする。

1. 鍵 K を $\mathtt{SKE.Gen}(1^\lambda)$ で生成する。
2. $\mathcal{A}$ は平文 m を出力する。
3. 2 つの暗号文 $c_0 \leftarrow \mathtt{SKE.Enc}(K, m)$ および $c_1 \xleftarrow{u} \mathcal{C}$（ここで $\mathcal{C}$ は暗号文の集合を表す）を生成する。乱数ビット $b \xleftarrow{u} \{0,1\}$ を選択し、c_b を $\mathcal{A}$ に与える。
4. $\mathcal{A}$ は $\mathtt{SKE.Enc}(K, \cdot)$ にオラクル・アクセスし、ビット $b' \in \{0,1\}$ を出力する。
5. $b' = b$ ならば 1 を、その他の場合には 0 を出力する。

任意の攻撃者 $\mathcal{A}$ に対して以下が成り立つとき、SKE は PCPA-安全（PCPA-secure）

38 キーワード w を含む文書の識別子の集合の要素数を $N_w = |DB(w)|$ と表すこともある。

という。

$$\left|\Pr[\mathbf{PCPA}_{\mathcal{A},\mathrm{SKE}}(\lambda)=1]-\frac{1}{2}\right|\leq \mathtt{negl}(\lambda).$$

ロ．判別不可能かつ検証可能な値域集合

適応的攻撃に対して安全な秘匿検索暗号方式を構成するときに、識別子が属するリストごとに異なった鍵で識別子を暗号化することがある。その際、ある暗号文を正しい鍵で復号したときにのみ有効（valid）であることを容易に確認できることを保証したい。また、ユーザがリストに乱数を詰め込むことによって、ユーザが偶然にある鍵で有効な暗号文を設定できる状態にならないこと、および、暗号文がそれぞれの鍵と独立にみえることを保証したい。

こうした特性を実現するために用いる共通鍵暗号方式として、判別不可能かつ検証可能な値域集合（elusive and verifiable range）を有する疑似ランダム性を満たす共通鍵暗号方式 $(\mathtt{Enc},\mathtt{Dec})$ を以下のとおり定義する。

定義 5（判別不可能かつ検証可能な値域集合を有する疑似ランダム性）． まず、共通鍵暗号方式 $(\mathtt{Enc},\mathtt{Dec})$ において、$K\leftarrow\{0,1\}^\lambda$ の値域集合を $\mathtt{Range}(K):=\{\mathtt{Enc}_K(x)\}_{x\in\{0,1\}^n}$ と表す。次の 3 つの性質が成り立つ場合、$(\mathtt{Enc},\mathtt{Dec})$ は判別不可能かつ検証可能な値域集合を有する疑似ランダム性を満たすという。

1. $(\mathtt{Enc},\mathtt{Dec})$ は判別不可能な値域集合（elusive range）を持つ：任意の確率的多項式時間アルゴリズムの攻撃者 $\mathcal{A}$ に対して、$\Pr[\mathcal{A}(1^\lambda)\in\mathtt{Range}(K)]\leq\mathtt{negl}(\lambda)$ が成り立つ。
2. $(\mathtt{Enc},\mathtt{Dec})$ は効率的に検証可能な値域集合（efficiently verifiable range）を持つ：$c\in\mathtt{Range}(K)$ ならば $\mathcal{M}(1^\lambda,K,c)=1$ を出力する確率的多項式時間マシン $\mathcal{M}$ が存在する。
3. $(\mathtt{Enc},\mathtt{Dec})$ は疑似ランダムな暗号文（pseudo-random ciphertexts）を持つ：任意の確率的多項式時間アルゴリズム $\mathcal{A}$ に対して、$|\Pr[\mathcal{A}^{\mathtt{Enc}_K(\cdot)}(1^\lambda)=1]-\Pr[\mathcal{A}^{\mathcal{R}(\cdot)}(1^\lambda)=1]|\leq\mathtt{negl}(\lambda)$ が成り立つ。

ここで、$\mathcal{R}(\cdot)$ は、初出の入力に対して $\mathtt{Enc}_K(\cdot)$ の出力長の一様な値を新たにサンプリングして出力する確率的オラクルである。

ハ．ハッシュ表

ハッシュ表（static hash table）とは、2 つのアルゴリズムの組 $(\mathtt{HTSetup},\mathtt{HTGet})$ で構成される。$\mathtt{HTSetup}$ は、入力 $S=\{(\ell_i,d_i)\}_{i=1}^k$ に対して、ハッシュ表の値 $HT(S)$ を出力する。$\mathtt{HTGet}$ は、入力 $(HT(S),\ell)$ に対して、$(\ell,d)\in S$ ならば d を出力し、その

他の場合には $\perp$ を出力する[39]。

(4) 文法（Syntax）

既存の秘匿検索暗号方式では、キーワードで構成される辞書 (Δ) 上の方式と、キーワードの集まり ($\boldsymbol{W}$) 上の方式という 2 つが存在する。カートモラらは、キーワードのすべての候補を辞書 Δ と呼び、$\boldsymbol{D} \subseteq 2^{\Delta}$ とモデル化して、Δ 上の方式を定義している。一方、アシャロフらは、$\boldsymbol{W} \subseteq \{0,1\}^{\ell}$ を n_W 個のキーワードから構成された集まりとして、$\boldsymbol{W}$ 上の方式について議論している。$\boldsymbol{W}$ と n_W はアシャロフらの方式では公開情報であるが、カートモラらの方式では非公開（漏洩が許されない）情報である。現在はキーワード集合 $\boldsymbol{W}$ 上の方式が研究の中心となっているが、初めて安全性が定式化されたのは、辞書上の方式であった。

イ. 辞書上の秘匿検索暗号方式

カートモラらは、秘匿検索暗号方式 Π の安全性を定式化した際に、$\boldsymbol{EDB}$ の形をインデックスと暗号文の集まりに限定した以下の文法を採用した。この場合、$\boldsymbol{D}$ が与えられたときに $\boldsymbol{DB}$ が定まることとなる。

定義 6（カートモラらの方式）. $\Pi = (\mathsf{KeyGen}, \mathsf{EDBSetup}, \mathsf{TokenGen}, \mathsf{Search}, \mathsf{Dec})$ を次のアルゴリズムで構成する。

- $K \leftarrow \mathsf{KeyGen}(1^{\lambda})$: 確率的に暗号鍵 K を生成する。λ はセキュリティ・パラメータである。
- $(\boldsymbol{I}, \boldsymbol{C}) \leftarrow \mathsf{EDBSetup}(K, \boldsymbol{DB})$: インデックス $\boldsymbol{I}$ と暗号文の集まり $\boldsymbol{C} = (C_1, \ldots, C_n)$ を生成する。これらを暗号化データベース $\boldsymbol{EDB}$ と呼ぶ。
- $t(w) \leftarrow \mathsf{TokenGen}(K, w)$: 秘密鍵 K とキーワード $w \in \boldsymbol{W}$ からトラップドア $t(w)$ を出力する。
- $S(w) \leftarrow \mathsf{Search}(\boldsymbol{I}, t(w))$: 暗号文の識別子の集合 $S(w)$ を出力する。
- $D_i \leftarrow \mathsf{Dec}(K, C_i) : S(w_i)$ で指定された識別子に対応する暗号文 C_i を復号し文書 D_i を出力する。

Δ 上の方式 Π が正しい（correct）とは、任意の $\lambda \in \mathbb{N}$、$w \in \Delta$、$\mathsf{KeyGen}(1^{\lambda})$ が出力した K、$\mathbf{D}$、$\mathsf{EDBSetup}(K, \boldsymbol{DB})$ が出力した $\boldsymbol{EDB}$ である $(\boldsymbol{I}, \boldsymbol{C})$ に対して、次の条件が成り立つことを意味する。

39 ℓ は $1 \leq \ell \leq n_W$ の範囲で動く。d はデータを表しており、$1 \leq d \leq \max_{1 \leq i \leq n_W}\{N_i\}$ の範囲で動く。

$\mathtt{Search}(\boldsymbol{I}, \mathtt{TokenGen}(K, w)) = DB(w)$ かつ $\mathtt{Dec}(K, C_i) = D_i$ $(1 \le i \le n)$.

ロ．キーワードの集まり上の検索可能暗号方式

アシャロフらは $\boldsymbol{W}$ 上の秘匿検索暗号方式 Π を次のように定義した。

定義 7（アシャロフらの方式）．次のアルゴリズムによって秘匿検索暗号方式 $\Pi = (\mathtt{KeyGen}, \mathtt{EDBSetup}, \mathtt{TokenGen}, \mathtt{Search}, \mathtt{Resolve})$ を構成する。

- $K \leftarrow \mathtt{KeyGen}(1^\lambda)$: 確率的に暗号鍵 K を生成する。λ はセキュリティ・パラメータである。
- $\boldsymbol{EDB} \leftarrow \mathtt{EDBSetup}(K, \boldsymbol{DB})$: データベース $\boldsymbol{DB}$ から暗号化データベース $\boldsymbol{EDB}$ を生成する。
- $t(w) = (\tau, \rho) \leftarrow \mathtt{TokenGen}(K, w)$: 秘密鍵 K とキーワード $w \in \boldsymbol{W}$ からトラップドア $t(w) = (\tau, \rho)$ を出力する。τ をトークン、ρ を内部状態と呼ぶ。
- $M(w) \leftarrow \mathtt{Search}(\tau, \boldsymbol{EDB})$: 暗号化された識別子の集合 $M(w)$ を出力する。
- $S \leftarrow \mathtt{Resolve}(\rho, M(w))$: $M(w)$ を復号し、その結果を S として出力する。

アシャロフらは、上記のモデルで、1 ラウンド方式と 2 ラウンド方式という 2 つのケースを扱った。4 節での証明は、2 ラウンド方式を対象としている。1 ラウンド方式では、サーバは文書の識別子を知り、かつその識別子に対応する暗号文を 1 回の検索依頼で回答する。2 ラウンド方式では、サーバは文書の識別子を知ることなく、暗号化された識別子をユーザに答える。ユーザはそれを復号し文書の識別子を求めてサーバに送ることで、サーバはその識別子に対応する暗号化された文書を回答する。

$\boldsymbol{W}$ 上の方式 Π が正しいとは、任意の攻撃者 $\mathcal{A}$ に対して、次の実験 $\mathtt{correct}_{\mathcal{A},\boldsymbol{DB}}(1^\lambda)$ において圧倒的な確率で 1 が出力されることを意味する。

1. $\mathcal{A}$ はデータベース $\boldsymbol{DB}$ と引継情報 $st_{\mathcal{A}}$ を出力する。
2. 鍵を $K \leftarrow \mathtt{KeyGen}(1^\lambda)$ で選び、$\boldsymbol{EDB} \leftarrow \mathtt{EDBSetup}(K, \boldsymbol{DB})$ で $\boldsymbol{DB}$ を暗号化する。
3. $\mathcal{A}$ は $(\boldsymbol{EDB}, st_{\mathcal{A}})$ を入力として質問 w_i を繰り返し行い、次の回答を得る。
   $(\tau_i, \rho_i) \leftarrow \mathtt{TokenGen}(K, w_i)$, $M_i \leftarrow \mathtt{Search}(\tau_i, \boldsymbol{EDB})$, $S_i = \mathtt{Resolve}(\rho, M_i)$.
4. すべての w_i に対して $S_i = DB(w_i)$ ならば 1 を、その他の場合には 0 を出力する。

上記の枠組みは、$\rho = \emptyset$ とみなすと、$S = M(w) = S(w)$ となって、定義 6 を自然に拡張したものであることがわかる。

補論 2.　二次元均衡割付技法と「ボールとビン」ゲーム

(1)　概要

二次元均衡割付技法は、アシャロフらの方式 1 および 2 の暗号化データベースにおいて、$\boldsymbol{DB}(w)$ をメモリ領域に配置する際に用いられる割当てアルゴリズムのベースとなるものであり、「ボールとビン」ゲームが利用されている。「ボールとビン」ゲームは、付番された複数のボールとビンが準備され、ボールを順々にビンに投げ込むというゲームであり、各種のアルゴリズムの動作を解析・構築する際に用いられる。ここでは、ボールは、各識別子を特定するためのラベル（例えば、識別子 *id* の添え字）に対応付けされる。このラベルの集合をリスト L_i という。ビンは、メモリ領域のうち識別子を格納可能な領域の集合に対応付けされる。アシャロフらは、こうしたゲームを利用して 2 つの二次元均衡割付技法、ワンチョイス（One-Choice）法とツーチョイス（Two-Choice）法を提案している。

以下では、1 列に隣接した d 個の領域の集合をビンと呼び、m 個のビンの集合 $\{B_1, B_2, \ldots, B_m\}$ を考える。なお、B_i の最後の領域と、B_{i+1} の最初の領域は隣接しているとする。

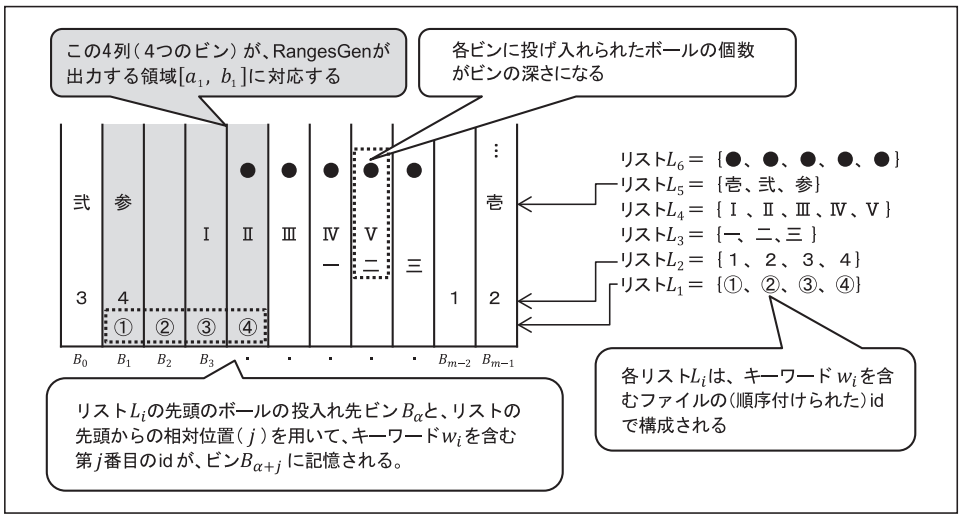
(2)　方式 1（ワンチョイス法）

方式 1 は、次の（a）、（b）の処理から構成される。まず 処理（a） では、リスト L_i に属するボールのラベルの個数 N_i に対して、N_i 個の隣接したビンの組 $R_i = (B_{\alpha_i \bmod m}, B_{(\alpha_i+1) \bmod m}, \ldots, B_{(\alpha_i+N_i-1) \bmod m})$ を選択し、領域の組 $(R_1, R_2, \ldots, R_k)$ を出力する。次に、処理（b） では、L_i の第 j 番目のボールのラベル (i, j) を格納する位置を、R_i を構成するビン B_{α_i+j} に属する可能なエントリから選択し、当該エントリを現実のエントリとする。この処理を $i = 1, 2, \ldots, k$ に対して実行する。

動作例を図表 A-1 に示す。各列はボールの投入れ先の候補となるビンを表す。リスト L_i の先頭のボールの投入れ先のビン B_{α_i} と、リストの先頭からの相対位置 (j) を用いて、キーワード w_i を含む第 j 番目のボールがビン B_{α_i+j} に記憶される。各ビンに投げ入れられたボールの個数をビンの「深さ」と呼ぶことにする。

図表 A-1 にあるように、複数の異なるボール（リストの要素）が、同一のビンに投げ入れられることがある。リスト L_1 を構成するボール①と②は、それぞれ隣接したビン B_1 と B_2 に格納されるが、リスト L_2 と L_5 にそれぞれ含まれるボール「4」と「参」がビン B_1 に投げ入れられて、ボール①と②がボール「4」と「参」を挟み

図表 A-1　方式 1 における「ボールとビン」ゲームの動作例



込む形となる。これによって、4 節（2）で説明した柔軟性を実現している。

ビン B_i に投げ入れられるボールの個数を、深さ d_i と定義して解析すると、$d_1 = \cdots = d_m$ とする m 個のビンの個数 (m) と深さ d には次の関係が成り立つことがアシャロフらによって証明されている（Asharov *et al.* [2016]）。

定理 2.　任意の k（リストの数）に対して、任意の $N_1, \ldots, N_k$（各リストの要素数）が与えられたとする。$N := \sum_{i=1}^{k} N_i$、かつ、ビンの個数を $m := \frac{N}{\log N \log\log N}$ とおくと、「ボールとビン」ゲーム終了時に、任意のビンの最大の深さ (d) は、少なくとも確率 $1 - N^{-\omega(1)}$ で、高々 $3 \log N \log\log N$ である[40]。

この定理 2 により、当該方式に必要なメモリ領域のサイズを見積ることが可能となる。

（3）　方式 2（ツーチョイス法）

方式 2 は、次の（a）、（b）の処理から構成される。処理（a）では、L_i に属するボールのラベルの個数 N_i に対して、互いに交わらない隣接したビンの組 $R_i = ((B_{\alpha_{i_1}} \bmod m, B_{(\alpha_{i_1}+1)} \bmod m, \ldots, B_{(\alpha_{i_1}+N_i-1)} \bmod m), (B_{\alpha_{i_2}} \bmod m, B_{(\alpha_{i_2}+1)} \bmod m, \ldots, B_{(\alpha_{i_2}+N_i-1)} \bmod m))$ を選択し、領域の組 $(R_1, R_2, \ldots, R_k)$ を出力する。次に、処理（b）では、L_i の第 0 番目のボールのラベル $(i, 0)$ を格納する位置を、R_i を構成するビン $B_{\alpha_{i_1}}$ と

40　これを圧倒的な確率ということにする。

$B_{\alpha_{i_2}}$ に属する可能なエントリから選択し、当該エントリを現実のエントリとする。ただし、ラベル $(i,0)$ が格納される現実のエントリが属するビンを B_α としたとき、(i,j) が格納されるエントリは、$B_{\alpha+j}$ の可能なエントリから選択される。この処理を $i = 1,2,\ldots,k$ に対して実行する。

この方式では、$(i,0)$ の格納先のエントリが属するビンとして、B_{α_1} と B_{α_2} の 2 つの候補を挙げ、何らかの判断基準を設定して一方を選択するという点が方式 1 と異なる。選択したビンの添え字を α とする。一般の場合の深さの解析は難しいため、Asharov *et al.* [2016] では、任意の N_i が 2 のべき乗の値であるとともに、$N_1 \geq \cdots \geq N_k$ となるケースを前提として説明している。この条件のもとでの動作例を図表 A-2 に示す。

ボールは、各ビンの下位のレイヤから順に投げ入れられるものとして、深さを見積る。上記の $\{N_i\}$ の前提により、同一の超ビン (super bin) に含まれるビンはすべて同じ深さになるので、超ビンの構成要素のビン $B_{n_i\alpha+j}$ に既に投げ入れられたボールの個数を超ビンの深さとする[41]。図表 A-2 において、n_i のレイヤまで塗りつぶされた領域は、n_i 個のビンで構成される超ビンを表す。投入れ先の候補としてビン $B_{n_i\alpha_1}$ から始まる超ビンと $B_{n_i\alpha_2}$ から始まる超ビンが指定されたとき、深さの小さい方の超ビンを投入れ先とする。図表 A-2 の例では、$B_{n_i\alpha_1}$ の深さが 3、$B_{n_i\alpha_2}$ の深さが 2 なので、$B_{n_i\alpha_2}$ から始まる超ビンが選択される。この操作を繰り返すとき、次の定理 3 が証明されている。

定理 3. 任意の k に対して、任意の N_i が 2 のべき乗の値であるとともに、$N_1 \geq \cdots \geq N_k$ とする。$N := \sum_{i=1}^{k} N_i$ とおいたときに、最大の $|\boldsymbol{DB}(w)|$ を $N^{1-\frac{1}{(\log\log N)}}$ とする。このとき $m := \frac{N}{\log\log N(\log\log\log N)^2}$ とおくと、ゲーム終了時に任意のビンの最大の深さは、圧倒的な確率で高々 $O((\log\log N)(\log\log\log N)^2)$ である。

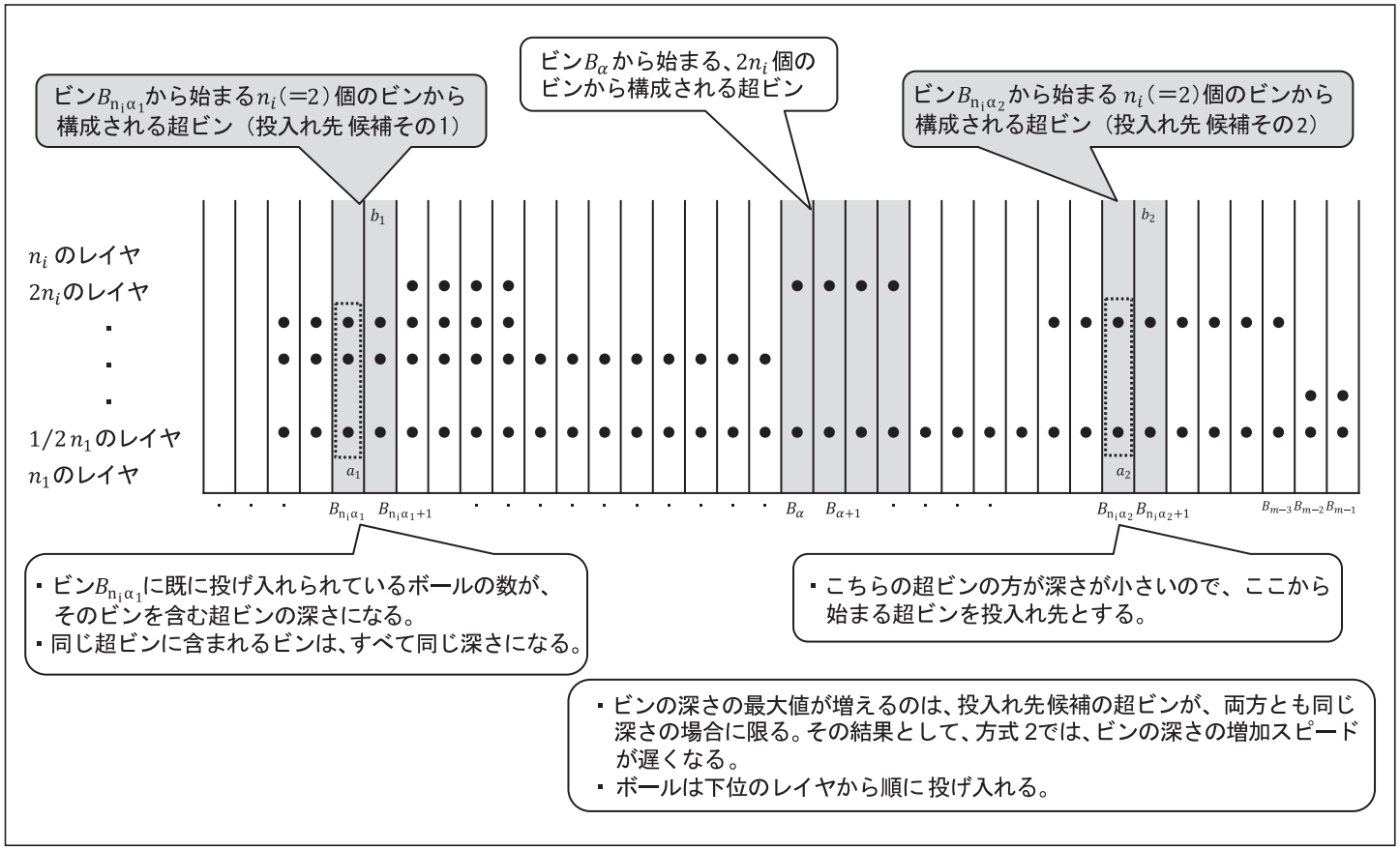
(4) 3 つの評価尺度の見積り

方式 1 および 2 が 4 節（2）の図表 2 における 3 つの尺度（オーダー）を実現することを確認する。領域効率については、$\boldsymbol{EDB} = (\mathtt{Data}, \mathtt{HT})$ のサイズで表される。定理 2 および 3 より、$|\mathtt{Data}| \in O(d \times m) = O(N)$、かつ、$|\mathtt{HT}| \in O(N)$ となるため、$|\boldsymbol{EDB}| = O(N+N) = O(N)$ である。局所性は、RangesGen が出力する d' 個の領域

41 複数のビンを一塊にしたものを超ビンという。超ビンの深さを、その超ビンを構成するビンの深さの平均と定義する。任意の N_i が 2 のべき乗の値であるとともに、$N_1 \geq \cdots \geq N_k$ となるとの条件のもとでは、同じ超ビンに属する普通のビンの深さはすべて等しくなり、超ビンの深さは各ビンの深さと一致する。

（方式 1 では $d' = 1$、方式 2 では $d' = 2$）の先頭アドレスへのアクセス回数で表される。このアクセス回数は、方式 1 では、1 または 2 であり、方式 2 では、2 または 3 であることから、いずれの方式においても局所性は $O(1)$ である。読出効率は、RangesGen が出力する領域のサイズを $|\boldsymbol{DB}(w)|$ で割った値で表されるが、方式 1 では $(\log N) \times$ (定数) であるため、$O(\log N)$ であり、方式 2 では $(\log \log N) \times$ (定数) であるため、$O(\log \log N)$ である。

図表 A-2　方式 2 における「ボールとビン」ゲームの動作例



補論 3.　2 つの方式のトラップドアの比較

カートモラらの方式では、トラップドアは $t(w) = (t(w,1), t(w,2), \ldots, t(w,n))$ となり文書数 (n) に依存する（Curtmola *et al.* [2006]）。また、キーワード検索フェーズにおいて、サーバにインデックス $\boldsymbol{I}$ を開示し、トラップドアの各成分 $t(w,j)$ が $\boldsymbol{I}$ に含まれているか否かをそれぞれ調べて $\boldsymbol{DB}(w)$ を算出することとしている。適応的選択キーワード攻撃に対する強秘匿性の証明では、予め確定した $\boldsymbol{I}$ と漏洩情報となる $\boldsymbol{DB}(w)$ が整合的となるように、$t^*(w) := DB(w) \oplus$（$\boldsymbol{I}$ の対応する要素）を計算してトラップドアをシミュレートした。

一方、アシャロフらの方式では、トラップドアは $\tau_w = (\ell, k, r)$ である（Asharov *et al.* [2016]）。ℓ、k は、HT と組み合わせて $N_w = |DB(w)|$ を復号するために用いられる。r は「ボールとビン」ゲームに乱数性を与えるために領域の集合 R の生成で用いられる。このとき、ℓ は検索依頼の種別数を表しているので、$1 \le \ell \le n_W$ の範囲で動く[42]。k は N_i を暗号化するための鍵を表しているので、$1 \le k \le N_{\max} := \max_{1 \le i \le n_W} N_i$ の範囲で動く。r はボールの投入れ先のビンを指定するために用いられるので、$1 \le r \le m$ の範囲で動く。また、$N_i \le n$ なので $N_{\max} \le n$ が成り立つ。

このように整理すると、$n_W \times N_{\max} \times m \le 2^n$ が成り立つときに、同一の安全性証明を与える 2 つの方式のうち、アシャロフらの方式において、必要なトラップドアのサイズが相対的に小さくなることから、より効率的であるといえる。なお、$N := \sum_{i=1}^{n_W} N_i \approx n_W \times N_{\max}$ かつ $m \approx N$ に注意すると、左辺、すなわち、アシャロフらの方式におけるトラップドアのサイズは $O(N^2)$ であり、n に依存しないことがわかる。

アシャロフらの方式において、トラップドアのビット長を文書数 (n) に依存しないように構成することができた理由について考察する。アシャロフらの方式では、深さ d でボールを格納できる m 個のビンを構成するために、$s := d \times m$ のエントリからなる領域を準備する必要があり、定理 2 および 3 より $s = O(N)$ である。領域は、RangesGen で可能なエントリが決定された後、Allocation によって現実のエントリとなる場合があるが、他のリストに対して現実のエントリとなることもある[43]。このように、複数のリストに対して同一の領域を可能なエントリとして使い回すことが可能であり、最終的に現実のエントリとならず配列 Data において乱数が詰め込まれるエントリの個数が、カートモラらの方式における $\boldsymbol{I}$ より少なくなる

42 Asharov *et al.* [2016] では、N 組の (ℓ, k, r, K_i) で集合 S を構成しているが、検索依頼の種別の上限が n_W であることから、この範囲で動くと考えられる。

43 配列 Data のエントリへの暗号文の割振りは、ゲームの実行結果に基づいているので、現実のエントリが上書きされることはない。

と考えられる[44]。

44 カートモラらの方式では、乱数が詰め込まれるエントリは、L_i ごとに区別されているので、$\boldsymbol{I}$ が大きくなっていた。$\boldsymbol{I}$ のサイズは $s = n \times n_W$ である一方、アシャロフらの方式における Data のサイズは $s = d \times m = O(N)$ である。

補論 4.　アシャロフらの方式 1 および 2 を変形する試み

アシャロフらの方式 1 および 2 について定理 1 で安全性が証明されていることから、それらの改良方式を検討し安全性について考察する。図表 10 に示した理論研究の成果より、読出効率を向上させるように改良することを試み、その結果、同一の安全性が維持できないことを明確にする。これによって、アシャロフらの方式が優れていることを確認することとしたい。

まず、方式 1 において、暗号鍵 K_i をトラップドアに含め、$\tau_i = (\ell_i, k_i, r_i, K_i)$ とし、ユーザによる識別子の復号の処理を回避することができるように変形することを考える。サーバが `Resolve` の処理を実行するので、サーバの処理量が増えるものの、ユーザによる処理量を削減するとともに、データの通信量を $1/d$ に削減できるというメリットがあることから、実用上意味があると考えられる。

しかし、この場合、$\mathcal{L}_2(\boldsymbol{D}, \boldsymbol{w}_i)$ に $\boldsymbol{DB}(w_i)$ が含まれ、サーバが K_i を用いて $\boldsymbol{DB}(w_i)$ を復号することになる一方、もともと K_i を知らない $\mathcal{S}_i$ がこれと整合的な K_i^* をシミュレートするのは困難であり、シミュレーションに成功しないと考えられる[45]。

また、サーバの領域効率の向上を目的として、方式 1 において、トラップドアに $N_i = |\boldsymbol{DB}(w_i)|$ を含め、$\tau_i = (r_i, N_i)$ とするという変形を施すことを考える。この場合、N_i を秘密にする必要がないため、`HT` が不要になり、領域効率の向上が期待できる。しかし、`HT` を用いないので、安全性の証明のポイントである鍵成分 k_i を活用できなくなり、安全性証明が有効でなくなると考えられる。

方式 2 において、ユーザによる `Resolve` をサーバに代行させるために、暗号鍵 K_i をトラップドアに含めることを考えると、K_i をサーバに開示することとなり、$h_1(w)$ と $h_2(w)$ のどちらの領域が採用されたかという情報が漏洩してしまう。その結果、方式 1 への変形の場合と同様に、安全性が低下してしまうこととなる。

45 この考察により、方式 1 が 4 節（4）で示した安全性を満たすうえで、無駄な情報を含んだ $\overline{\boldsymbol{D}}$ を許すことが本質的であるといえる。なお、無駄な情報とは、リスト L_i の隣同士のラベルの間に挟み込まれた（他のリスト $L_{i'}$ の）ラベルのことである。