(19)日本国特許庁(JP)

# (12)特許公報(B2)

(11)特許番号
特許第7505583号
(P7505583)

(45)発行日　令和6年6月25日(2024.6.25)

(24)登録日　令和6年6月17日(2024.6.17)

(51)国際特許分類
*Ｇ０９Ｃ　1/00　(2006.01)*

ＦＩ
Ｇ０９Ｃ　1/00　６５０Ｚ

請求項の数　8　（全21頁）

(21)出願番号　特願2022-567986(P2022-567986)
(86)(22)出願日　令和2年12月10日(2020.12.10)
(86)国際出願番号　PCT/JP2020/046125
(87)国際公開番号　WO2022/123744
(87)国際公開日　令和4年6月16日(2022.6.16)
審査請求日　令和5年6月1日(2023.6.1)

(73)特許権者　000004226
日本電信電話株式会社
東京都千代田区大手町一丁目５番１号
(74)代理人　100107766
弁理士　伊東 忠重
(74)代理人　100070150
弁理士　伊東 忠彦
(74)代理人　100124844
弁理士　石原 隆治
(72)発明者　市川 敦謙
東京都千代田区大手町一丁目５番１号
日本電信電話株式会社内
審査官　行田 悦資

最終頁に続く

(54)【発明の名称】　秘匿ハッシュテーブル構築装置、秘匿ハッシュテーブル構築システム、秘匿ハッシュテーブル構築方法、及びプログラム

(57)【特許請求の範囲】

【請求項１】

キーと、データがダミーか否かを示すフラグとを有するデータを複数個含む実データ列から、秘密計算により、Ｂ通りのアドレス値にそれぞれ最大Ｚ個までのデータを格納できる秘匿ハッシュテーブルを構築する秘匿ハッシュテーブル構築装置であって、

前記実データ列の各データに対する、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列と、ダミーとしての、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列とを連結した第１配列を生成し、

前記実データ列とダミーのデータ列とを連結した第２配列を生成し、

前記第１配列における第１アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第１アドレス値に対して同一データに対するランク付け操作を行い、当該ランク付けに基づいて前記第１配列と前記第２配列のそれぞれをソートし、

前記第１配列における第２アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第２アドレス値に対して同一データに対するランク付け操作を行い、

前記第１配列において、第１アドレス値と第２アドレス値のうち、ランクの低い側のアドレス値を抽出し、抽出したアドレス値とフラグを有する第３配列を生成し、

前記第３配列におけるアドレス値から計算されたランク配列の各要素とＺとの比較によ

り得られた第４配列を用いて、前記第２配列をソートし、ソートした前記第２配列におけるＢＺ個の要素を前記秘匿ハッシュテーブルとして出力する
演算部を備える秘匿ハッシュテーブル構築装置。
【請求項２】
前記演算部は、前記実データ列における各データと前記ダミーのデータ列における各データとを示すタグを有するタグの列を使用することにより、前記実データ列と前記ダミーのデータ列とを連結した前記第２配列に対して、前記第４配列に基づくソート以外のソートを行うことなく、前記秘匿ハッシュテーブルを生成する
請求項１に記載の秘匿ハッシュテーブル構築装置。
【請求項３】
前記演算部は、アクセス対象のキー値から２つのアドレス値を計算し、前記秘匿ハッシュテーブルから、前記２つのアドレス値に対応する２Ｚ個のデータを取得し、２Ｚ個のデータのうち、前記キー値と同一のキー値を有するデータを返す
請求項１又は２に記載の秘匿ハッシュテーブル構築装置。
【請求項４】
前記演算部は、削除対象のキー値から２つのアドレス値を計算し、前記秘匿ハッシュテーブルから、前記２つのアドレス値に対応する２Ｚ個のデータを取得し、２Ｚ個のデータのデータのうち、前記キー値と同一のキー値を有するデータを削除する
請求項１ないし３のうちいずれか１項に記載の秘匿ハッシュテーブル構築装置。
【請求項５】
前記演算部は、前記秘匿ハッシュテーブルにおける全データを、そのフラグに基づいてソートし、先頭の所定個数のデータを前記実データ列として取得する
請求項１ないし４のうちいずれか１項に記載の秘匿ハッシュテーブル構築装置。
【請求項６】
キーと、データがダミーか否かを示すフラグとを有するデータを複数個含む実データ列から、秘密計算により、Ｂ通りのアドレス値にそれぞれ最大Ｚ個までのデータを格納できる秘匿ハッシュテーブルを構築する秘匿ハッシュテーブル構築システムであって、
前記実データ列の各データに対する、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列と、ダミーとしての、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列とを連結した第１配列を生成し、
前記実データ列とダミーのデータ列とを連結した第２配列を生成し、
前記第１配列における第１アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第１アドレス値に対して同一データに対するランク付け操作を行い、当該ランク付けに基づいて前記第１配列と前記第２配列のそれぞれをソートし、
前記第１配列における第２アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第２アドレス値に対して同一データに対するランク付け操作を行い、
前記第１配列において、第１アドレス値と第２アドレス値のうち、ランクの低い側のアドレス値を抽出し、抽出したアドレス値とフラグを有する第３配列を生成し、
前記第３配列におけるアドレス値から計算されたランク配列の各要素とＺとの比較により得られた第４配列を用いて、前記第２配列をソートし、ソートした前記第２配列におけるＢＺ個の要素を前記秘匿ハッシュテーブルとして出力する
演算部を備える秘匿ハッシュテーブル構築システム。
【請求項７】
キーと、データがダミーか否かを示すフラグとを有するデータを複数個含む実データ列から、秘密計算により、Ｂ通りのアドレス値にそれぞれ最大Ｚ個までのデータを格納できる秘匿ハッシュテーブルを構築する秘匿ハッシュテーブル構築システムが実行する秘匿ハッシュテーブル構築方法であって、

前記実データ列の各データに対する、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列と、ダミーとしての、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列とを連結した第１配列を生成するステップと、

前記実データ列とダミーのデータ列とを連結した第２配列を生成するステップと、

前記第１配列における第１アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第１アドレス値に対して同一データに対するランク付け操作を行い、当該ランク付けに基づいて前記第１配列と前記第２配列のそれぞれをソートするステップと、

前記第１配列における第２アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第２アドレス値に対して同一データに対するランク付け操作を行うステップと、

前記第１配列において、第１アドレス値と第２アドレス値のうち、ランクの低い側のアドレス値を抽出し、抽出したアドレス値とフラグを有する第３配列を生成するステップと、

前記第３配列におけるアドレス値から計算されたランク配列の各要素とＺとの比較により得られた第４配列を用いて、前記第２配列をソートし、ソートした前記第２配列におけるＢＺ個の要素を前記秘匿ハッシュテーブルとして出力するステップと

を備える秘匿ハッシュテーブル構築方法。

【請求項８】

コンピュータを、請求項１ないし５のうちいずれか１項に記載の秘匿ハッシュテーブル構築装置における演算部として機能させるためのプログラム。

【発明の詳細な説明】

【技術分野】

【０００１】

本発明は、秘密計算に関するものであり、特に配列の順序を秘匿した操作に関するものである。

【背景技術】

【０００２】

ハッシュテーブルは、データのキー値をハッシュ関数等により符号化しデータ配列におけるアドレス値との紐付けを行うことで、キー値に基づくデータ検索を高速で行うためのデータ構造である。これは例えば、外部サーバ等にデータを預け、随時キー値による検索を行いたい場合などに有用である。

【０００３】

しかし、通常のハッシュテーブルはデータ秘匿を想定するものではなく、例えばユーザからデータ（ないしハッシュテーブル）を委託されたサーバがテーブル構造を識別でき、それによりユーザのアクセス傾向を観測しうるという問題がある。こうした問題を解決する既存の方法として、秘匿ハッシュテーブル（例えば、参考文献［４，１］）がある。なお、参考文献名については、明細書の最後にまとめて記載した。秘匿ハッシュテーブルは、サーバに対してテーブル構造を秘匿したままでハッシュテーブルを構築させる技術であり、サーバに対してユーザのアクセス傾向を秘匿できる。

【０００４】

秘匿ハッシュテーブル構築方法として自明に挙げられるのは、ユーザが全てのデータを手元で暗号化しハッシュテーブルを構築、サーバへ委託する方法であるが、この場合データ数ｎに対してユーザの記憶領域がＯ（ｎ）必要となる。上記に挙げた参考文献［４］及び［１］は、全ての暗号化データをサーバが持っている前提で、ユーザ・サーバの協調プロトコルによってサーバ上に秘匿ハッシュテーブルを構築する方法であり、この時ユーザの記憶領域はＯ（１）で十分であるという利点を持つ。

【０００５】

しかしながら、参考文献［４，１］の技術は秘匿ハッシュテーブル構築の際、ユーザ・サーバ間で非常に多くの通信を要するという課題がある。現実に、ユーザの通信環境や計

算性能が必ずしも好条件であるとは限らないことからも、上記コストは可能な限り低減されることが望ましい。これを解決した方法の１つに、非特許文献１に開示された技術が挙げられる。この文献には、複数台のサーバを利用した秘密計算により、ユーザ・サーバ間での通信を一切要さずに秘匿ハッシュテーブルを構築する手法が示されており、そのためユーザ端末の通信環境や計算性能を問わず安定した性能を実現できる。

【０００６】

特に、非特許文献１にて示される手法のうち"Ｏｂｌｉｖｉｏｕｓ　Ｇｒｅｅｄｙ　Ｈａｓｈｉｎｇ"と呼ばれる秘匿ハッシュ法は、格納されるべき各データに対してそれぞれ２通りのアドレス値（すなわち、テーブルのどの位置に格納されるべきかを示す値）を与えることで、各データが２つの格納先のより空いている方へと格納されるようなテーブル構築アルゴリズムを実現している。これにより、他の既存手法に比較してテーブルサイズを小さく抑えられるといった利点がある。

【先行技術文献】

【非特許文献】

【０００７】

【文献】市川敦謙，濱田浩気，菊池亮，五十嵐大．3 パーティ計算上の最適秘匿ハッシュと劣対数効率のOblivious RAM．暗号と情報セキュリティシンポジウム(SCIS)2020 予稿集(2020)．

【発明の概要】

【発明が解決しようとする課題】

【０００８】

非特許文献１に開示されているＯｂｌｉｖｉｏｕｓ　Ｇｒｅｅｄｙ　Ｈａｓｈｉｎｇはテーブルサイズが小さく、参考文献［４，１］に開示の技術と違いユーザ・サーバ間での通信が不要であるといったメリットの一方で、テーブル構築の際、単純にシステム全体の通信量で比較した場合に非常に多くの通信を要する欠点もある。具体的には、ｎをテーブルに格納するデータの総数とすると、参考文献［４］や［１］の手法で秘匿ハッシュテーブルを構築する際にはＯ（ｎｌｏｇ$^{2}$ｎ）ビットの通信を要するのに対し、Ｏｂｌｉｖｉｏｕｓ　Ｇｒｅｅｄｙ　ＨａｓｈｉｎｇはＯ（Ｚ×ｎｌｏｇ$^{2}$ｎ）ビットの通信が必要となる。ここで、Ｚとは方式の安全性にまつわるパラメータであり、非特許文献１ではＺ＝２ｌｏｇ$^{\varepsilon}$ｎが例示されている。

【０００９】

本発明は上記の点に鑑みてなされたものであり、テーブルサイズを小さくしつつ、通信量を増大させないで秘匿ハッシュテーブルを構築する技術を提供することを目的とする。

【課題を解決するための手段】

【００１０】

開示の技術によれば、キーと、データがダミーか否かを示すフラグとを有するデータを複数個含む実データ列から、秘密計算により、Ｂ通りのアドレス値にそれぞれ最大Ｚ個までのデータを格納できる秘匿ハッシュテーブルを構築する秘匿ハッシュテーブル構築装置であって、

前記実データ列の各データに対する、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列と、ダミーとしての、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列とを連結した第１配列を生成し、

前記実データ列とダミーのデータ列とを連結した第２配列を生成し、

前記第１配列における第１アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第１アドレス値に対して同一データに対するランク付け操作を行い、当該ランク付けに基づいて前記第１配列と前記第２配列のそれぞれをソートし、

前記第１配列における第２アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第２アドレス値に対して同一データ

に対するランク付け操作を行い、
前記第１配列において、第１アドレス値と第２アドレス値のうち、ランクの低い側のアドレス値を抽出し、抽出したアドレス値とフラグを有する第３配列を生成し、
前記第３配列におけるアドレス値から計算されたランク配列の各要素とＺとの比較により得られた第４配列を用いて、前記第２配列をソートし、ソートした前記第２配列におけるＢＺ個の要素を前記秘匿ハッシュテーブルとして出力する
演算部を備える秘匿ハッシュテーブル構築装置が提供される。
【発明の効果】
【００１１】
開示の技術によれば、テーブルサイズを小さくしつつ、通信量を増大させないで秘匿ハッシュテーブルを構築する技術が提供される。
【図面の簡単な説明】
【００１２】
【図１】秘匿ハッシュテーブル構築システムの構成図である。
【図２】サーバの機能構成図である。
【図３】装置のハードウェア構成例を示す図である。
【図４】実施例１の処理手順を示すフローチャートである。
【図５】実施例２の処理手順を示すフローチャートである。
【図６】実施例３の処理手順を示すフローチャートである。
【図７】実施例４の処理手順を示すフローチャートである。
【図８】実施例５の処理手順を示すフローチャートである。
【発明を実施するための形態】
【００１３】
以下、図面を参照して本発明の実施の形態（本実施の形態）を説明する。以下で説明する実施の形態は一例に過ぎず、本発明が適用される実施の形態は、以下の実施の形態に限られるわけではない。
【００１４】
（実施の形態の概要）
本実施の形態では、複数台のサーバを用いてユーザ・サーバ間通信コストを削減した効率的な秘匿ハッシュテーブル構築方法、及び操作方法について説明する。本実施の形態に係る技術により、各データに２通りの格納先を付与することでＯｂｌｉｖｉｏｕｓ　Ｇｒｅｅｄｙ　Ｈａｓｈｉｎｇと同等にテーブルサイズを小さく抑えつつも、サーバ間通信量Ｏ（ｎｌｏｇ$^{2}$ｎ）ビットのみでテーブル構築が可能である。以下、本実施の形態のシステム構成、及び処理手順について詳細に説明する。
【００１５】
（システム構成）
図１に、本実施の形態における秘匿ハッシュテーブル構築システムの構成例を示す。図１に示すように、本実施の形態の秘匿ハッシュテーブル構築システムは、複数のサーバ１００－１～１００－Ｎが通信ネットワーク上２００に備えられた構成を有する。各サーバはコンピュータである。当該コンピュータは物理マシンであってもよいし、クラウドにより提供される仮想マシンであってもよい。
【００１６】
図１に示すユーザ端末３００が、秘匿ハッシュテーブル構築システムにアクセスして、例えばキーに対応するデータを取得する。
【００１７】
本実施の形態では、これら複数のサーバ１００－１～１００－Ｎ間の協調プロトコルにより秘密計算で秘匿ハッシュテーブル構築に係る処理を実行する。ただし、本発明の実施にあたって、複数サーバの協調プロトコルを使用することは例であり、これに限定されるわけではない。本発明は、秘密計算を行うことができるどのような技術にも適用可能である。例えば、１つのサーバで秘密計算を行うことができる場合に、そのサーバで本発明に

係る技術を実行することも可能である。

【００１８】

図２に、サーバ１００の構成例を示す。図２に示すサーバ１００の構成例は、複数のサーバ１００－１～１００－Ｎを仮想的に１つのサーバとみなした場合のその構成である。この場合、図２に示すサーバ１００を秘匿ハッシュテーブル構築システムと呼んでもよい。また、当該サーバ１００を、秘匿ハッシュテーブル構築装置と呼んでもよい。また、１つのサーバで秘密計算を行う場合におけるそのサーバが図２に示すサーバ１００であってもよい。

【００１９】

図２に示すように、サーバ１００は、入力部１１０、演算部１２０、出力部１３０、データ格納部１４０を有する。演算部１２０は、秘匿ハッシュテーブル構築及び操作に係る処理を実行する。データ格納部１４０には、計算にあたって予め用意しておくデータ、計算途中の秘匿した配列等のデータ、テーブルが格納される。入力部１１０は、例えば、ユーザ端末３００からの要求を受信し、入力する。出力部１４０は、例えば、ユーザ端末３００からの要求に対する演算結果をユーザ端末３００に返す。

【００２０】

（ハードウェア構成例）

本実施の形態におけるサーバ１００（秘匿ハッシュテーブル構築装置）は、例えば、コンピュータに、本実施の形態で説明する処理内容を記述したプログラムを実行させることにより実現可能である。なお、この「コンピュータ」は、物理マシンであってもよいし、クラウド上の仮想マシンであってもよい。仮想マシンを使用する場合、ここで説明する「ハードウェア」は仮想的なハードウェアである。

【００２１】

上記プログラムは、コンピュータが読み取り可能な記録媒体（可搬メモリ等）に記録して、保存したり、配布したりすることが可能である。また、上記プログラムをインターネットや電子メール等、ネットワークを通して提供することも可能である。

【００２２】

図３は、上記コンピュータのハードウェア構成例を示す図である。図３のコンピュータは、それぞれバスＢで相互に接続されているドライブ装置１０００、補助記憶装置１００２、メモリ装置１００３、ＣＰＵ１００４、インタフェース装置１００５、表示装置１００６、入力装置１００７、出力装置１００８等を有する。

【００２３】

当該コンピュータでの処理を実現するプログラムは、例えば、ＣＤ－ＲＯＭ又はメモリカード等の記録媒体１００１によって提供される。プログラムを記憶した記録媒体１００１がドライブ装置１０００にセットされると、プログラムが記録媒体１００１からドライブ装置１０００を介して補助記憶装置１００２にインストールされる。但し、プログラムのインストールは必ずしも記録媒体１００１より行う必要はなく、ネットワークを介して他のコンピュータよりダウンロードするようにしてもよい。補助記憶装置１００２は、インストールされたプログラムを格納すると共に、必要なファイルやデータ等を格納する。

【００２４】

メモリ装置１００３は、プログラムの起動指示があった場合に、補助記憶装置１００２からプログラムを読み出して格納する。ＣＰＵ１００４は、メモリ装置１００３に格納されたプログラムに従って、当該サーバ１００に係る機能を実現する。インタフェース装置１００５は、ネットワークに接続するためのインタフェースとして用いられる。表示装置１００６はプログラムによるＧＵＩ（Ｇｒａｐｈｉｃａｌ　Ｕｓｅｒ　Ｉｎｔｅｒｆａｃｅ）等を表示する。入力装置１００７はキーボード及びマウス、ボタン、又はタッチパネル等で構成され、様々な操作指示を入力させるために用いられる。出力装置１００８は演算結果を出力する。

【００２５】

（秘密計算に係る基本的な処理について）

まず、サーバ１００の演算部１２０により実行される秘匿ハッシュテーブル構築や操作の処理において前提となる、秘密計算に係る基本的な処理について説明する。これら基本的な処理自体は既存技術である。

【００２６】

＜秘匿化・復元＞

以下では、値ｘの秘匿値を［［ｘ］］と表し、処理ｘ→［［ｘ］］をｘの秘匿化、［［ｘ］］→ｘをｘの復元と呼ぶ。本実施の形態では、秘匿化・復元処理を行うための技術として秘密分散法（参考文献［６，５］）を使用することを想定しているが、同等の機能及び安全性を満たす方法ならばどのような秘匿化・復元処理技術を用いてもよい。

【００２７】

＜基本演算＞

秘匿値同士の加算・減算・乗算をそれぞれ以下のように表す。

【００２８】

［［ａ＋ｂ］］←［［ａ］］＋［［ｂ］］

［［ａ－ｂ］］←［［ａ］］－［［ｂ］］

［［ａ×ｂ］］←［［ａ］］×［［ｂ］］

ａ、ｂがｌｅｎビットで表現される空間にあるとすると、参考文献［７］の技術では加算と減算を通信コスト０、乗算を通信コストＯ（ｌｅｎ）ビットで実行できる。

【００２９】

＜比較＞

秘匿値同士の比較を以下のように表す。

【００３０】

［［ｃ］］←［［［ａ］］＝？［［ｂ］］］

上記の比較は、ａとｂが等しいかどうかの判定結果がｃ（１ビット数）であることを示す。正しければ１、そうでなければ０になる。

【００３１】

［［ｄ］］←［［［ａ］］≦？［［ｂ］］］

上記の比較は、ａがｂ以下であるか否かの判定結果がｄ（１ビット数）であることを示す。正しければ１、そうでなければ０になる。

【００３２】

比較の演算は、ａ、ｂがｌｅｎビットで表現される空間にあるとすると、参考文献［７］の技術では通信コストＯ（ｌｅｎ）ビットで実行できる。

【００３３】

＜内積＞

秘匿値のベクトル又は配列［［ａ］］＝（［［$a_0$］］，…，［［$a_{n-1}$］］），［［ｂ］］＝（［［$b_0$］］，…，［［$b_{n-1}$］］）の内積を以下のように表す。

【００３４】

［［ｃ］］←＜［［ａ］］，［［ｂ］］＞

ベクトルａ，ｂの各要素がそれぞれｌｅｎビットで表現される空間にあるとすると、参考文献［７］の技術では内積を通信コストＯ（ｌｅｎ）ビットで実行できる。

【００３５】

＜配列のシャッフル＞

秘密の配列［［Ａ］］をシャッフルする処理を以下のように表す。

【００３６】

［［Ａ′］］←Ｓｈｕｆｆｌｅ（［［Ａ］］）

Ａがｌｅｎビットの要素をｎ個持つ配列であるとき、参考文献［３］の技術ではこの処理を通信コストＯ（ｌｅｎ・ｎ）ビットで実行できる。

【００３７】

＜安定ソート＞

秘密のキー配列［［Ｋ］］及びソート対象配列［［Ａ］］に対し、［［Ａ］］を［［Ｋ

]]の各値に基づきソートする処理を以下のように表す。

【００３８】

[[A']]←Sort([[K]],[[A]])

この処理はKに関して昇順・降順のいずれも可能である。Kがlenビットの要素をn個持つ配列、Aがmビットの要素をn個持つ配列であるとき、参考文献［３］の技術ではこの処理を通信コストO（len・nlogn＋mn）ビットで実行できる。

【００３９】

＜同一データのランク付け＞

配列[[A]]＝([[$a_0$]],...,[[$a_{n-1}$]])を、ソート済みの配列の配列とする。このとき、同一データに対するランク付け操作を下記のように表す。

【００４０】

[[B]]←Rank([[A]])

ただし、[[B]]＝([[$b_0$]],...,[[$b_{n-1}$]])であり、各$b_i$は以下を満たす：

・$b_0=1$

・全てのi∈{1,...,n－1}において、$a_{i-1}\neq a_i$ならば$b_i=1$

・全てのi∈{1,...,n－1}において、$a_{i-1}=a_i$ならば$b_i=b_{i-1}+1$

具体的に、例えばA＝（０，０，０，０，１，１，２，２，２）のとき、B＝（１，２，３，４，１，２，１，２，３）となる。参考文献［１］に開示されているRange Prefix Sumを用いれば、この操作を通信コストO（nlogn）で実行できる。

【００４１】

＜擬似ランダム関数＞

値a，sをそれぞれ、擬似ランダム関数の入力及び秘密鍵であるとする。秘匿値[[a]]，[[s]]に対し、秘密の疑似ランダム値を計算する操作を以下のように表す。

【００４２】

[[r]]←PRF([[a]],[[s]])

aのビット長をlenとすると、この秘密計算処理は例えば参考文献［２］の技術を用いて通信コストO（len）で実現できる。

【００４３】

＜秘密計算による暗号化＞

値a，sをそれぞれ、ブロック暗号の平文及び秘密鍵であるとする。秘匿値[[a]]，[[s]]に対し、秘密計算により暗号文を生成する操作を以下のように表す。

【００４４】

[[c]]←Enc([[a]],[[s]])

aのビット長をlenとすると、この秘密計算処理は参考文献［２］の技術を用いて通信コストO（len）で実現できる。

【００４５】

以下、本実施の形態に係るサーバ１００が実行する処理の具体例について、実施例１～５を用いて説明する。以下の処理はサーバ１００の演算部１２０により実行される。また、予め用意しておくデータ、計算により得られる配列、テーブル等はデータ格納部１４０に格納される。演算部１２０は、データ格納部１４０から読み出したデータに対して演算を行い、演算結果をデータ格納部１４０に格納することを繰り返しながら処理を進める。

【００４６】

なお、下記の各実施例における処理は一例である。例えば、１と０の意味や、昇順／降順などが、下記の各実施例におけるものと逆であってもよい。

【００４７】

（実施例１）

まず、実施例１を説明する。本実施例１では、サーバ１００が、データ格納部１４０に格納されているキー$k_i$とフラグ$e_i$からなるデータ$a_i$＝（$k_i$，$e_i$）を用いて秘匿ハッシュテーブルを構築する方法を示す。キー$k_i$はデータ毎に固有の識別子であり、アク

セスの際などにデータを一意に特定する目的に用いる。

【００４８】

フラグ$e_i$はデータがダミーか否かを判定する１ｂｉｔの値であり、ここでは$e_i=0$なら実際のデータ、$e_i=1$ならダミーデータであるとする。なお、データの総数がｎの場合において、$k_i$のビット長は最小でｃｅｉｌ（ｌｏｇｎ）となるが、冗長性があっても構わないとする。また、各データにはキーとフラグ以外にも、任意の情報$v_i$を付与し、$a_i=(k_i, e_i, v_i)$のようにしても良いとする。

【００４９】

前提として、サーバ１００が、データ格納部１４０の中に、秘匿化されたデータ列$[[A]]=([[a_0]],\ldots,[[a_{n-1}]])$；$[[a_i]]=([[k_i]],[[e_i]])$を保持しているものとする。

【００５０】

また、以下では事前に秘匿化された秘密鍵$[[s_0]]$，$[[s_1]]$を生成し、データ格納部１４０に保持しておくものとするが、プロトコル中に逐次生成しても構わない。

【００５１】

本実施例１で構築する秘匿ハッシュテーブルは、サイズＢ×Ｚのデータ配列であり、別の表現をするならばＢ通りのアドレス値にそれぞれ最大Ｚ個までのデータを格納できるデータ構造である、とする。また、本実施例１で用いる擬似ランダム関数は、$PRF([[k]],[[s]])\in\{0,\ldots,B-1\}$を満たすとする。

【００５２】

上記の前提の下、サーバ１００は、以下で説明する処理により、秘匿ハッシュテーブルを構築する。図４のフローチャートの手順に沿って、サーバ１００が実行する処理を説明する。

【００５３】

＜Ｓ１０１（ステップ１０１）＞

Ｓ１０１において、サーバ１００は、$[[A]]$に対してシャッフルを行い$[[A]]\leftarrow Shuffle([[A]])$とする。ただしこの操作は、入力となる$[[A]]$の状態や満たすべき安全性によっては実行しなくてもよいものとする。

【００５４】

＜Ｓ１０２＞

Ｓ１０２において、サーバ１００は、$[[A]]$に対して、格納先データ配列$[[ADDR]]=((([[addr^0_0]],[[addr^1_0]],[[e_0]],[[rank^0_0]],[[rank^1_0]]),\ldots,([[addr^0_{n-1}]],[[addr^1_{n-1}]],[[e_{n-1}]],[[rank^0_{n-1}]],[[rank^1_{n-1}]]))$を計算する。

【００５５】

ただし、$[[addr^p_i]]\leftarrow PRF([[k_i]],[[s_p]])$；$i\in\{0,\ldots,n-1\}$，$p\in\{0,1\}$とし、また$[[rank^p_i]]\leftarrow[[0]]$と初期化する。ここで$addr^0_i$，$addr^1_i$はそれぞれAのｉ番目のデータ$a_i=(k_i,e_i)$に対応する２通りのアドレス値に相当し、また$rank^p_i$は後にランク付けをする際のランク保持用領域である。

【００５６】

なお、データのイメージを説明する便宜上、秘匿の記号$[[]]$なしで説明を行うこととすると、この時点でのＡＤＤＲは、例えば、ＡＤＤＲ＝（（３，１，０，０，０），（１８，９５，０，０，０），……，（８，７８，０，０，０））といった配列（ｎ個のデータの配列）になる。以下、適宜、$[[]]$なしで説明を行う場合がある。

【００５７】

＜Ｓ１０３＞

Ｓ１０３において、サーバ１００は、ダミーの格納先データ配列$[[ADDR_d]]=([[d_0]],\ldots,[[d_{BZ-1}]])$；$d_i=(floor(i/Z),floor$

（ｉ／Ｚ），１，０，０）を生成する。

【００５８】

これは２通りのアドレス値$addr^0_i$，$addr^1_i$が同じ値ｆｌｏｏｒ（ｉ／Ｚ）∈｛０，…，Ｂ－１｝を指しているダミーの格納先データを、アドレス値０，…，Ｂ－１ごとにＺ個ずつ保持する配列である。

【００５９】

サーバ１００は、このダミーの［［$ADDR_d$］］を、実データに関する格納先データ配列［［ＡＤＤＲ］］の末尾に連結し、長さｎ＋ＢＺの配列［［ＡＤＤＲ′］］＝［［ＡＤＤＲ］］｜｜［［$ADDR_d$］］とする。

【００６０】

この時点でのＡＤＤＲ´は、例えば、ＡＤＤＲ´＝（（３，１，０，０，０），（１８，９５，０，０，０），……，（８，７８，０，０，０），（０，０，１，０，０），（０，０，１，０，０），……，（８，８，１，０，０））といった配列（ｎ＋ＢＺ個のデータの配列）になる。

【００６１】

なお、以下では格納先データ配列［［ＡＤＤＲ′］］に対して、記法［［ＡＤＤＲ′．$addr_0$］］を、［［ＡＤＤＲ′］］から１番目のアドレス値［［$addr^0_i$］］のみを取り出した配列（［［$addr^0_0$］］，…，［［$addr^0_{n+BZ-1}$］］）であると定義する。同様に、記法［［ＡＤＤＲ′．$addr_1$］］，［［ＡＤＤＲ′．$rank_0$］］，［［ＡＤＤＲ′．$rank_1$］］を、それぞれ［［ＡＤＤＲ′］］から［［$addr^1_i$］］，［［$rank^0_i$］］，［［$rank^1_i$］］を取り出した配列と定義する。

【００６２】

さらに、記法［［ＡＤＤＲ′．$pos_0$］］，［［ＡＤＤＲ′．$pos_1$］］によって、それぞれ［［ＡＤＤＲ′］］からペア（［［$addr^0_i$］］，［［$e_i$］］）、（［［$addr^1_i$］］，［［$e_i$］］）を抽出した配列であるとする。

【００６３】

例えば、ＡＤＤＲ´＝（（３，１，０，０，０），（１８，９５，０，０，０），……，（８，７８，０，０，０），（０，０，１，０，０），（０，０，１，０，０），……，（８，８，１，０，０））であるとすると、ＡＤＤＲ′．$pos_0$＝（（３，０），（１８，０），……，（８，０），（０，１），（０，１），……，（８，１））となる。

【００６４】

＜Ｓ１０４＞

Ｓ１０４において、サーバ１００は、ダミーのデータ列［［$A_d$］］＝（［［$a^d_0$］］，…，［［$a^d_{BZ-1}$］］）；$a^d_i$＝（ｄｕｍｍｙ，１）を生成する。ただし、ここでｄｕｍｍｙは予め定めておいた、いずれのキー$k_i$とも異なる値であるとする。この［［$A_d$］］を実データ列［［Ａ］］の末尾に連結し、長さｎ＋ＢＺの配列［［Ａ′］］＝［［Ａ］］｜｜［［$A_d$］］とする。

【００６５】

この時点でのＡ´は、例えば、Ａ´＝（（１１，０），（１０１，０），……，（３，０），（ｄｕｍｍｙ，１），（ｄｕｍｍｙ，１），……，（ｄｕｍｍｙ，１））といった配列（ｎ＋ＢＺ個のデータの配列）になる。

【００６６】

＜Ｓ１０５＞

Ｓ１０５において、サーバ１００はまず、１番目のアドレス値$addr^0_i$とフラグ$e_i$に基づいてデータ配列と格納先データ配列のそれぞれのソートを行う。すなわち、［［Ａ′］］←Ｓｏｒｔ（［［ＡＤＤＲ′．$pos_0$］］，［［Ａ′］］）、及び［［ＡＤＤＲ′］］←Ｓｏｒｔ（［［ＡＤＤＲ′．$pos_0$］］，［［ＡＤＤＲ′］］）を計算する。

【００６７】

この時、データ配列及び格納先データ配列はアドレス値による昇順かつ同アドレス値においては実データ＞ダミーデータの順で並べ替えられるものとするが、以後のソートやラ

ンク付けの昇順・降順を調整することでソート順序を変更することもできる。

【００６８】

ソート後のＡ´は、例えば、Ａ´＝（（３，０），（ｄｕｍｍｙ，１），（ｄｕｍｍｙ，１），……，（１１，０），（ｄｕｍｍｙ，１），，……，）といった配列（ｎ＋ＢＺ個のデータの配列）になる。

【００６９】

＜Ｓ１０６＞

Ｓ１０６において、サーバ１００は、ソート済みのアドレス値に基づきランク付けを行う。すなわち、[[ＡＤＤＲ′．ｒａｎｋ$_0$]]←Ｒａｎｋ（[[ＡＤＤＲ′．ａｄｄｒ$_0$]]）を計算する。

【００７０】

ＡＤＤＲ′．ｒａｎｋ$_0$は、例えば、ＡＤＤＲ′．ｒａｎｋ$_0$＝（１，２，３，４，….，１，２，３，….，１，２，３，４，５）といった配列（ｎ＋ＢＺ個のデータの配列）になる。

【００７１】

＜Ｓ１０７＞

Ｓ１０７において、サーバ１００は、データ配列及び格納先データ配列を、１番目のランクに基づき降順でソートする。すなわち、[[Ａ′]]←Ｓｏｒｔ（[[ＡＤＤＲ′．ｒａｎｋ$_0$]]，[[Ａ′]]）及び[[ＡＤＤＲ′]]←Ｓｏｒｔ（[[ＡＤＤＲ′．ｒａｎｋ$_0$]]，[[ＡＤＤＲ′]]）を、$rank^0{}_i$が大きい順に並ぶよう計算する。なお、この操作は単に各配列を昇順でソートしたうえ、逆順に並べ替えることでも同様に実現できる。

【００７２】

＜Ｓ１０８＞

Ｓ１０８において、サーバ１００は、Ｓ１０５での処理と同様に、今度は２番目のアドレス値$addr^1{}_i$とフラグに基づいてソートを行う。すなわち、[[Ａ′]]←Ｓｏｒｔ（[[ＡＤＤＲ′．ｐｏｓ$_1$]]，[[Ａ′]]）及び[[ＡＤＤＲ′]]←Ｓｏｒｔ（[[ＡＤＤＲ′．ｐｏｓ$_1$]]，[[ＡＤＤＲ′]]）を計算する。この時、安定ソートを用いれば、同アドレス値においては実データ＞ダミーデータかつ高ランク＞低ランクの順で並べ替えることができている。

【００７３】

＜Ｓ１０９＞

Ｓ１０９において、サーバ１００は、Ｓ１０６での処理と同様に、２番目のソート済みアドレス値に基づきランク付けを行う。すなわち、[[ＡＤＤＲ′．ｒａｎｋ$_1$]]←Ｒａｎｋ（[[ＡＤＤＲ′．ａｄｄｒ$_1$]]）を計算する。

【００７４】

＜Ｓ１１０＞

Ｓ１１０において、サーバ１００は、格納先データ配列[[ＡＤＤＲ′]]の各要素[[$b_i$]]＝（[[$addr^0{}_i$]]，[[$addr^1{}_i$]]，[[$e_i$]]，[[$rank^0{}_i$]]，[[$rank^1{}_i$]]）について、それぞれランクの低い側のアドレス値を抽出する。すなわち、大小比較[[$z_i$]]←[[[[$rank^0{}_i$]]≦？[[$rank^1{}_i$]]]]に基づき、[[$addr_i$]]←[[$addr^1{}_i$]]＋[[$z_i$]]×（[[$addr^0{}_i$]]－[[$addr^1{}_i$]]）を計算する。その後、選ばれたアドレス値とフラグを併せた配列[[$ＡＤＤＲ^{fin}$]]＝（（[[$addr_0$]]），[[$e_0$]]），…，（[[$addr_{n+BZ-1}$]]，[[$e_{n+BZ-1}$]]））を計算する。

【００７５】

＜Ｓ１１１＞

Ｓ１１１において、サーバ１００は、新たに得た配列[[$ＡＤＤＲ^{fin}$]]に基づき、各データを昇順でソートする。すなわち、[[Ａ′]]←Ｓｏｒｔ（[[$ＡＤＤＲ^{fin}$]]，[[Ａ′]]）及び[[$ＡＤＤＲ^{fin}$]]←Ｓｏｒｔ（[[$ＡＤＤＲ^{fin}$]]，[

［ＡＤＤＲ$^{fin}$］］）を計算する。

【００７６】

＜Ｓ１１２＞

Ｓ１１２において、サーバ１００は、［［ＡＤＤＲ$^{fin}$］］のアドレス値部分のみを抽出した配列［［ＡＤＤＲ$^{fin}$．ａｄｄｒ］］を用いて、ランク配列［［Ｒ］］＝Ｒａｎｋ（［［ＡＤＤＲ$^{fin}$．ａｄｄｒ］］）を計算し、さらに［［Ｒ］］の各要素［［$r_i$］］に対して大小比較［［$y_i$］］←［［［$r_i$］］≦？Ｚ］を計算した結果の配列［［Ｙ］］＝（［［$y_0$］］，…，［［$y_{n+BZ-1}$］］）を得る。このとき、$y_i$＝１のデータはテーブルに格納され、$y_i$＝０のデータは削除されることを示している。

【００７７】

＜Ｓ１１３＞

Ｓ１１３において、サーバ１００は、［［Ｙ］］を用いてデータ配列を昇順でソートする。すなわち、［［Ａ′］］←Ｓｏｒｔ（［［Ｙ］］，［［Ａ′］］）を計算する。その後、末尾のＢＺ個の要素のみをハッシュテーブルとして出力し、その他は削除する。また、ハッシュテーブルに付随するアクセス用情報として２つの秘密鍵［［$s_0$］］，［［$s_1$］］も出力する。

【００７８】

本実施例１において、［［$k_i$］］をＯ（ｌｏｇＮ）ビットとし、Ｂ×Ｚ＝Ｏ（ｎ）とすれば、サーバ間通信量はソート、ランク付け及び比較によりＯ（ｎｌｏｇ$^2$ｎ）ビットとなる。

【００７９】

（実施例２）

次に、実施例２を説明する。本実施例２では、実施例１と同等のテーブル構築を、より効率的に行う方法を示す。実施例１では、常に格納先データ配列と共にデータ配列［［Ａ］］（ないし［［Ａ′］］）がソートされており、単にソート回数が増していた他、仮にキーとフラグに対して大きなデータ$v_i$が付与されていた場合、データ配列のソートのために効率悪化を招く可能性もあった。

【００８０】

本実施例２では原則として実施例１のアルゴリズムを踏襲しつつも、秘密計算上での暗号化を利用してソート回数を削減し、以下のように秘匿ハッシュテーブルを構築する。以下、図５のフローチャートの手順に沿って説明する。

【００８１】

＜Ｓ２０１＞

Ｓ２０１において、サーバ１００は、実施例１のＳ１０１と同様に、［［Ａ］］に対してシャッフルを行い［［Ａ］］←Ｓｈｕｆｆｌｅ（［［Ａ］］）とする。ただしこの操作は、入力となる［［Ａ］］の状態や満たすべき安全性によっては実行しなくてもよいものとする。

【００８２】

＜Ｓ２０２＞

Ｓ２０２において、サーバ１００は、秘匿化されたブロック暗号の秘密鍵［［ｓ］］を生成し、［［Ａ］］に対応する秘密のタグ配列［［Ｔａｇ］］＝（［［$tag_0$］］，…，［［$tag_{n-1}$］］）；［［$tag_i$］］←Ｅｎｃ（［［ｉ］］，［［ｓ］］）を計算する。なお、秘密鍵［［ｓ］］はプロトコルの開始以前に生成していても構わないとする。

【００８３】

＜Ｓ２０３＞

Ｓ２０３において、サーバ１００は、実施例１のＳ１０２と同様に、格納先データ配列［［ＡＤＤＲ］］＝（（［［$addr^0_0$］］，［［$addr^1_0$］］，［［$e_0$］］，［［$rank^0_0$］］，［［$rank^1_0$］］，［［$tag_0$］］），…，（［［$addr^0_{n-1}$］］，［［$addr^1_{n-1}$］］，［［$e_{n-1}$］］，［［$rank^0_{n-1}$］］，［［$rank^1_{n-1}$］］，［［$tag_{n-1}$］］））を計算する。実施例１とは、タグ［［ｔａｇ

$_i$]]が含まれていることのみが異なる。

【００８４】

＜Ｓ２０４＞

Ｓ２０４において、サーバ１００は、実施例１のＳ１０３と同様に、ダミーの格納先データ配列[[$ADDR_d$]]＝([[$d_0$]],…,[[$d_{BZ-1}$]])；[[$d_i$]]＝([[floor(i／Z)]],[[floor(i／Z)]],[[1]],[[0]],[[0]],[[$tag^d_i$]])を生成し、[[ADDR′]]＝[[ADDR]]||[[$ADDR_d$]]とする。ただし、[[$tag^d_i$]]←ＰＲＦ([[n＋i]],[[s]])であるとする。

【００８５】

＜Ｓ２０５＞

Ｓ２０５において、サーバ１００は、実施例１のＳ１０４と同様に、ダミーのデータ列[[$A_d$]]＝([[$a^d_0$]],…,[[$a^d_{BZ-1}$]])；$a^d_i$＝(dummy,１)を生成し、[[A′]]＝[[A]]||[[$A_d$]]とする。また、本実施例２のＳ２０４で計算したタグの列[[$Tag_d$]]＝([[$tag^d_0$]],…,[[$tag^d_{BZ-1}$]])を用いて、[[Tag′]]←[[Tag]]||[[$Tag_d$]]とする。

【００８６】

＜Ｓ２０６＞

Ｓ２０６において、サーバ１００は、配列[[A′]]及び[[Tag′]]の並び順を秘匿・ランダマイズするために、([[A′]],[[Tag′]])←Ｓｈｕｆｆｌｅ([[A′]],[[Tag′]])を計算する。この操作では、２つの配列の対応関係を保ったままランダマイズするため、並列に同じシャッフル処理を実行している。

【００８７】

＜Ｓ２０７＞

Ｓ２０７において、サーバ１００は、実施例１のＳ１０５～Ｓ１１３と同様の処理を行う。ただし、実施例２において、実施例１のＳ１０５～Ｓ１１２に対応する各手順においては配列[[A′]]には一切手を触れず格納先データ配列のみに操作を行うこととし、また、実施例１のＳ１１０に対応する手順で新たな格納先データ配列を得る際には[[$ADDR^{fin}$]]の各要素を([[$addr_i$]],[[$e_i$]],[[$tag_i$]])として、タグ情報を引き継ぐようにする。

【００８８】

＜Ｓ２０８＞

Ｓ２０８において、サーバ１００は、実施例１のＳ１１２に対応する手順で得た[[Y]]を用いて[[$ADDR^{fin}$]]を昇順でソートする。すなわち、[[$ADDR^{fin}$]]←Ｓｏｒｔ([[Y]],[[$ADDR^{fin}$]])を計算する。その後、末尾のＢＺ個の要素のみ残し、その他は削除する。

【００８９】

＜Ｓ２０９＞

Ｓ２０９において、サーバ１００は、タグ配列[[Tag′]]の全ての要素を復元し、平文に戻す。同時に、[[$ADDR^{fin}$]]に含まれる全てのタグを復元し、平文に戻す。データ配列[[A′]]の中から、対応するタグが[[$ADDR^{fin}$]]に含まれるような要素ＢＺ個を取り出し、[[$ADDR^{fin}$]]の並び順の通りに並べることでハッシュテーブルとする。最後にハッシュテーブルと秘密鍵[[$s_0$]],[[$s_1$]]を出力する。

【００９０】

本実施例２において、[[$k_i$]]をＯ(ｌｏｇｎ)ビットとし、Ｂ×Ｚ＝Ｏ(ｎ)とすれば、サーバ間通信量はソート、ランク付け及び比較によりＯ($n\log^2 n$)ビットとなる。実施例１では任意のデータ$v_i$を含む組($k_i$,$e_i$,$v_i$)のデータサイズがω(ｌｏｇｎ)ビットとなる場合にサーバ間通信量がω($n\log^2 n$)ビットとなってしまうのに対し、本実施例は$v_i$のサイズに関わらずＯ($n\log^2 n$)ビットで済む利点

がある。

【００９１】

（実施例３）

次に、実施例３を説明する。実施例３では、実施例１、２で構築した秘匿ハッシュテーブルへのデータ参照方法について説明する。前提として、サーバ１００は、実施例１ないし２で構築された秘匿ハッシュテーブル（サイズＢ×Ｚ）と秘密鍵［［$s_0$］］、［［$s_1$］］をデータ格納部１４０に保持しているとする。また、ユーザ端末３００はアクセスしたいキーｋを持っているとするが、これはサーバ同士の合意に基づいてサーバ自身で選定しても構わない。以下、図６のフローチャートに示す手順に沿って説明する。

【００９２】

＜Ｓ３０１＞

Ｓ３０１において、ユーザ端末３００は、サーバ１００に対し、アクセスしたいデータに対応するキーの秘匿値［［ｋ］］を送る。ただし、この秘匿キー値はユーザ端末３００からサーバ１００への要求だけでなく、サーバ間合意に基づくサーバ自身による生成も可能である。

【００９３】

＜Ｓ３０２＞

Ｓ３０２において、サーバ１００は、疑似ランダム関数を用いて、２つのアドレス値［［$addr^0$］］←ＰＲＦ（［［ｋ］］，［［$s_0$］］），［［$addr^1$］］←ＰＲＦ（［［ｋ］］，［［$s_1$］］）を計算し、これを復元して$addr^0$，$addr^1$を得る。

【００９４】

＜Ｓ３０３＞

Ｓ３０３において、サーバ１００は、データ格納部１４０に格納されているハッシュテーブルから$addr^0$、$addr^1$に対応する２Ｚ個の要素を取り出す。すなわち、ハッシュテーブルを配列［［Ｔａｂｌｅ］］＝（［［$a_0$］］，…，［［$a_{BZ-1}$］］）と読み替えた場合に、［［ａ］］＝（［［$a_{Z\times addr0}$］］，…，［［$a_{Z\times addr0+Z-1}$］］，［［$a_{Z\times addr1}$］］，…，［［$a_{Z\times addr1+Z-1}$］］）を取得する。

【００９５】

＜Ｓ３０４＞

Ｓ３０４において、サーバ１００は、［［ａ］］の各データ［［$a_j$］］について、キーの比較を行い［［ｃ］］＝（［［$c_0$］］，…，［［$c_{2Z-1}$］］）；［［$c_j$］］＝［［［$k_j$］］＝？［［ｋ］］］を計算する。

【００９６】

＜Ｓ３０５＞

Ｓ３０５において、サーバ１００は、内積［［ａ］］←＜［［ａ］］，［［ｃ］］＞を計算し、これをユーザ端末３００へ返送するか、又はサーバ間合意に基づき復元する。あるいは、これを復元せず、全く別の秘密計算処理に用いてもよい。

【００９７】

本実施例３において、ユーザ・サーバ間の通信量Ｏ（ｌｏｇｎ）ビットであり、サーバ間通信量はＯ（Ｚｌｏｇｎ）ビットである。

【００９８】

（実施例４）

本実施例４では、実施例１、２で構築した秘匿ハッシュテーブルでのデータ削除方法を説明する。前提として、サーバ１００は、実施例１ないし２で構築されたハッシュテーブル（サイズＢ×Ｚ）と秘密鍵［［$s_0$］］，［［$s_1$］］をデータ記憶部１４０に保持しているとする。また、ユーザは削除したいデータのキーｋを持っているとするが、これはサーバ同士の合意に基づいてサーバ自身で選定しても構わない。以下、図７のフローチャートに示す手順に沿って説明する。

【００９９】

＜Ｓ４０１＞

Ｓ４０１において、ユーザ端末３００は、サーバ１００に対し、削除したいキーの秘匿値［［ｋ］］を送る。ただし、この秘匿キー値はユーザ端末３００からサーバ１００への要求だけでなく、サーバ間合意に基づくサーバ自身による生成も可能である。

【０１００】

＜Ｓ４０２＞

Ｓ４０２において、サーバ１００は、疑似ランダム関数を用いて、２つのアドレス値［［ａｄｄｒ$^{0}$］］←ＰＲＦ（［［ｋ］］，［［ｓ$_{0}$］］），［［ａｄｄｒ$^{1}$］］←ＰＲＦ（［［ｋ］］，［［ｓ$_{1}$］］）を計算し、これを復元してａｄｄｒ$^{0}$，ａｄｄｒ$^{1}$を得る。

【０１０１】

＜Ｓ４０３＞

Ｓ４０３において、サーバ１００は、データ格納部１４０に保持しているハッシュテーブルからａｄｄｒ$^{0}$、ａｄｄｒ$^{1}$に対応する２Ｚ個の要素を取り出す。すなわち、ハッシュテーブルを配列［［Ｔａｂｌｅ］］＝（［［ａ$_{0}$］］，…，［［ａ$_{BZ-1}$］］）と読み替えた場合に、［［ａ］］＝（［［ａ$_{Z\times addr0}$］］，…，［［ａ$_{Z\times addr0+Z-1}$］］、［［ａ$_{Z\times addr1}$］］，…，［［ａ$_{Z\times addr1+Z-1}$］］）を取得する。

【０１０２】

＜Ｓ４０４＞

Ｓ４０４において、サーバ１００は、［［ａ］］の各データ［［ａ$_{j}$］］について、キーの比較に基づきデータ（［［ｋ］］，［［ｅ］］）の削除処理を行う。すなわち、全てのｊに関して［［ｋ$_{j}$］］＝［［ｋ$_{j}$］］＋［［［［ｋ$_{j}$］］＝？［［ｋ］］］×（ｄｕｍｍｙ－［［ｋ$_{j}$］］），［［ｅ$_{j}$］］＝［［ｅ$_{j}$］］＋［［［［ｋ$_{j}$］］＝？［［ｋ］］］を計算する。

【０１０３】

＜Ｓ４０５＞

最後に、サーバ１００は［［ａ］］の各要素をハッシュテーブルの元の位置に上書きする。

本実施例４の通信量は実施例３と等価である。

【０１０４】

（実施例５）

本実施例５では、実施例１、２で構築したハッシュテーブルの解体・全データの抽出方法について説明する。前提として、サーバ１００は実施例１ないし２で構築されたハッシュテーブル（サイズＢ×Ｚ）をデータ格納部１４０に保持しているとする。以下、図８のフローチャートに示す手順に沿って説明する。

【０１０５】

＜Ｓ５０１＞

Ｓ５０１において、サーバ１００は、ハッシュテーブルの全データを、そのフラグに基づき昇順でソートする。すなわち、テーブルを配列［［Ｔａｂｌｅ］］＝（［［ａ$_{0}$］］，…，［［ａ$_{BZ-1}$］］）と読み替え、またテーブルからフラグのみを抽出した配列を［［Ｅ］］＝（［［ｅ$_{0}$］］，…，［［ｅ$_{BZ-1}$］］）とした場合に、［［Ｔａｂｌｅ］］＝Ｓｏｒｔ（［［Ｅ］］，［［Ｔａｂｌｅ］］）とする。

【０１０６】

＜Ｓ５０２＞

Ｓ５０２において、サーバ１００は、［［Ｔａｂｌｅ］］の先頭ｎデータを配列［［Ａ］］とし、残りを削除する。

【０１０７】

本実施例では、１ビットの情報に基づくソート１回のみ行うため、サーバ間通信量はＯ（ｎｌｏｇｎ）となる。

【０１０８】

（実施の形態の効果）

本実施の形態によれば、秘密計算を用いることでユーザとの通信を介さず秘匿ハッシュ

テーブルを構築できる。その際、ランク付けという秘密計算処理を利用することで、テーブル構築に必要な秘匿ソートの回数を大きく減らし、従来法よりも通信コストを大きく削減できる。

【０１０９】

すなわち、秘匿ハッシュテーブルの構築において、各データに２通りのアドレス値を割り振ることでテーブルサイズを小さくするという従来法のメリットをそのままに、サーバ間通信量をω（ｎｌｏｇ$^{2\cdot 5}$ｎ）からＯ（ｎｌｏｇ$^{2}$ｎ）へと削減できる。構築済みのテーブルへの操作も可能である。

【０１１０】

（付記）

本明細書には、少なくとも下記各項の秘匿ハッシュテーブル構築装置、秘匿ハッシュテーブル構築システム、秘匿ハッシュテーブル構築方法、及びプログラムが開示されている。

（第１項）

キーと、データがダミーか否かを示すフラグとを有するデータを複数個含む実データ列から、秘密計算により、Ｂ通りのアドレス値にそれぞれ最大Ｚ個までのデータを格納できる秘匿ハッシュテーブルを構築する秘匿ハッシュテーブル構築装置であって、

前記実データ列の各データに対する、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列と、ダミーとしての、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列とを連結した第１配列を生成し、

前記実データ列とダミーのデータ列とを連結した第２配列を生成し、

前記第１配列における第１アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第１アドレス値に対して同一データに対するランク付け操作を行い、当該ランク付けに基づいて前記第１配列と前記第２配列のそれぞれをソートし、

前記第１配列における第２アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第２アドレス値に対して同一データに対するランク付け操作を行い、

前記第１配列において、第１アドレス値と第２アドレス値のうち、ランクの低い側のアドレス値を抽出し、抽出したアドレス値とフラグを有する第３配列を生成し、

前記第３配列におけるアドレス値から計算されたランク配列の各要素とＺとの比較により得られた第４配列を用いて、前記第２配列をソートし、ソートした前記第２配列におけるＢＺ個の要素を前記秘匿ハッシュテーブルとして出力する

演算部を備える秘匿ハッシュテーブル構築装置。

（第２項）

前記演算部は、前記実データ列における各データと前記ダミーのデータ列における各データとを示すタグを有するタグの列を使用することにより、前記実データ列と前記ダミーのデータ列とを連結した前記第２配列に対して、前記第４配列に基づくソート以外のソートを行うことなく、前記秘匿ハッシュテーブルを生成する

第１項に記載の秘匿ハッシュテーブル構築装置。

（第３項）

前記演算部は、アクセス対象のキー値から２つのアドレス値を計算し、前記秘匿ハッシュテーブルから、前記２つのアドレス値に対応する２Ｚ個のデータを取得し、２Ｚ個のデータのうち、前記キー値と同一のキー値を有するデータを返す

第１項又は第２項に記載の秘匿ハッシュテーブル構築装置。

（第４項）

前記演算部は、削除対象のキー値から２つのアドレス値を計算し、前記秘匿ハッシュテーブルから、前記２つのアドレス値に対応する２Ｚ個のデータを取得し、２Ｚ個のデータのデータのうち、前記キー値と同一のキー値を有するデータを削除する

第１項ないし第３項のうちいずれか１項に記載の秘匿ハッシュテーブル構築装置。

（第５項）
前記演算部は、前記秘匿ハッシュテーブルにおける全データを、そのフラグに基づいてソートし、先頭の所定個数のデータを前記実データ列として取得する
第１項ないし第４項のうちいずれか１項に記載の秘匿ハッシュテーブル構築装置。
（第６項）
キーと、データがダミーか否かを示すフラグとを有するデータを複数個含む実データ列から、秘密計算により、Ｂ通りのアドレス値にそれぞれ最大Ｚ個までのデータを格納できる秘匿ハッシュテーブルを構築する秘匿ハッシュテーブル構築システムであって、
前記実データ列の各データに対する、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列と、ダミーとしての、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列とを連結した第１配列を生成し、
前記実データ列とダミーのデータ列とを連結した第２配列を生成し、
前記第１配列における第１アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第１アドレス値に対して同一データに対するランク付け操作を行い、当該ランク付けに基づいて前記第１配列と前記第２配列のそれぞれをソートし、
前記第１配列における第２アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第２アドレス値に対して同一データに対するランク付け操作を行い、
前記第１配列において、第１アドレス値と第２アドレス値のうち、ランクの低い側のアドレス値を抽出し、抽出したアドレス値とフラグを有する第３配列を生成し、
前記第３配列におけるアドレス値から計算されたランク配列の各要素とＺとの比較により得られた第４配列を用いて、前記第２配列をソートし、ソートした前記第２配列におけるＢＺ個の要素を前記秘匿ハッシュテーブルとして出力する
演算部を備える秘匿ハッシュテーブル構築システム。
（第７項）
キーと、データがダミーか否かを示すフラグとを有するデータを複数個含む実データ列から、秘密計算により、Ｂ通りのアドレス値にそれぞれ最大Ｚ個までのデータを格納できる秘匿ハッシュテーブルを構築する秘匿ハッシュテーブル構築システムが実行する秘匿ハッシュテーブル構築方法であって、
前記実データ列の各データに対する、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列と、ダミーとしての、第１アドレス値及び第２アドレス値と、フラグと、アドレス値毎のランクとを含む格納先データ配列とを連結した第１配列を生成するステップと、
前記実データ列とダミーのデータ列とを連結した第２配列を生成するステップと、
前記第１配列における第１アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第１アドレス値に対して同一データに対するランク付け操作を行い、当該ランク付けに基づいて前記第１配列と前記第２配列のそれぞれをソートするステップと、
前記第１配列における第２アドレス値とフラグに基づいて、前記第１配列と前記第２配列のそれぞれをソートし、ソートした前記第１配列の第２アドレス値に対して同一データに対するランク付け操作を行うステップと、
前記第１配列において、第１アドレス値と第２アドレス値のうち、ランクの低い側のアドレス値を抽出し、抽出したアドレス値とフラグを有する第３配列を生成するステップと、
前記第３配列におけるアドレス値から計算されたランク配列の各要素とＺとの比較により得られた第４配列を用いて、前記第２配列をソートし、ソートした前記第２配列におけるＢＺ個の要素を前記秘匿ハッシュテーブルとして出力するステップと
を備える秘匿ハッシュテーブル構築方法。
（第８項）

コンピュータを、第１項ないし第５項のうちいずれか１項に記載の秘匿ハッシュテーブル構築装置における演算部として機能させるためのプログラム。

【０１１１】

以上、本実施の形態について説明したが、本発明はかかる特定の実施形態に限定されるものではなく、特許請求の範囲に記載された本発明の要旨の範囲内において、種々の変形・変更が可能である。

【０１１２】

［参考文献］

[1] T-H. H. Chan, Y. Guo, W-K. Lin, and E. Shi. Oblivious hashing revisited, and applications to asymptotically efficient ORAM and OPRAM. Cryptology ePrint Archive, Report 2017/924, 2017.
[2] K. Chida, K. Hamada, D. Ikarashi, R. Kikuchi, and B. Pinkas. High-throughput secure AES computation. In WAHC@CCS 2018, pages 13-24, 2018.
[3] K. Chida, K. Hamada, D. Ikarashi, R. Kikuchi, N. Kiribuchi, B. Pinkas. Anefficient secure threeparty sorting protocol with an honest majority. Cryptologyeprint Archive, Report 2019/695 (2019), https://eprint.iacr.org/2019/695
[4] O. Goldreich and R. Ostrovsky. Software protection and simulation on oblivious RAMs. J. ACM, 43(3):431-473, May 1996. 8
[5] M. Ito, A. Saito, and T. Nishizeki. Secret sharing schemes realizing general access structures. Proceedings of the IEEE Global Telecommunication Conference, Globecom 87, pp. 99-102, 1987.
[6] A. Shamir. How to share a secret. Commun. ACM, Vol. 22, No. 11, pp. 612-613, 1979.
[7] 桐淵直人，五十嵐大，濱田浩気，菊池亮．プログラマブルな秘密計算ライブラリMEVAL3．暗号と情報セキュリティシンポジウム(SCIS)2018 予稿集(2018)．

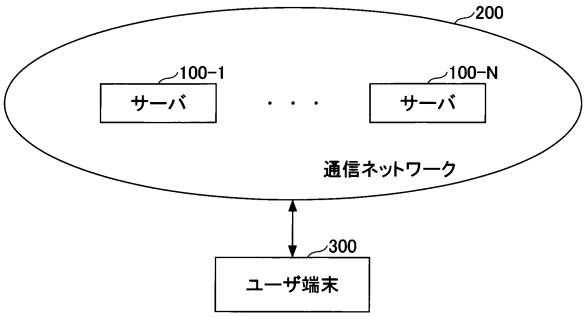
【符号の説明】

【０１１３】

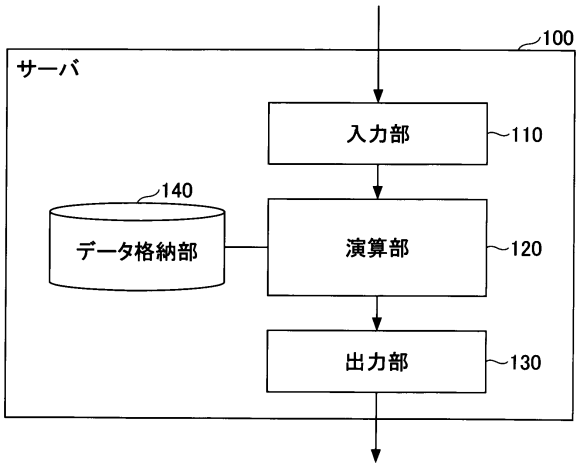
１００　サーバ
１１０　入力部
１２０　演算部
１３０　出力部
１４０　データ格納部
２００　通信ネットワーク
３００　ユーザ端末
１０００　ドライブ装置
１００１　記録媒体
１００２　補助記憶装置
１００３　メモリ装置
１００４　ＣＰＵ
１００５　インタフェース装置
１００６　表示装置
１００７　入力装置

【図面】

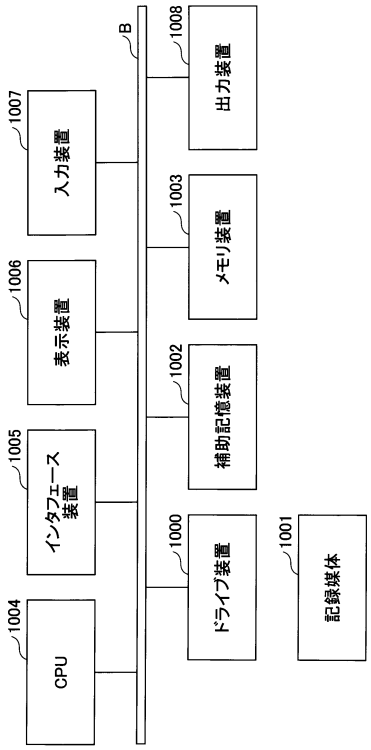
【図１】


200
100-1
サーバ
・・・
100-N
サーバ
通信ネットワーク
300
ユーザ端末

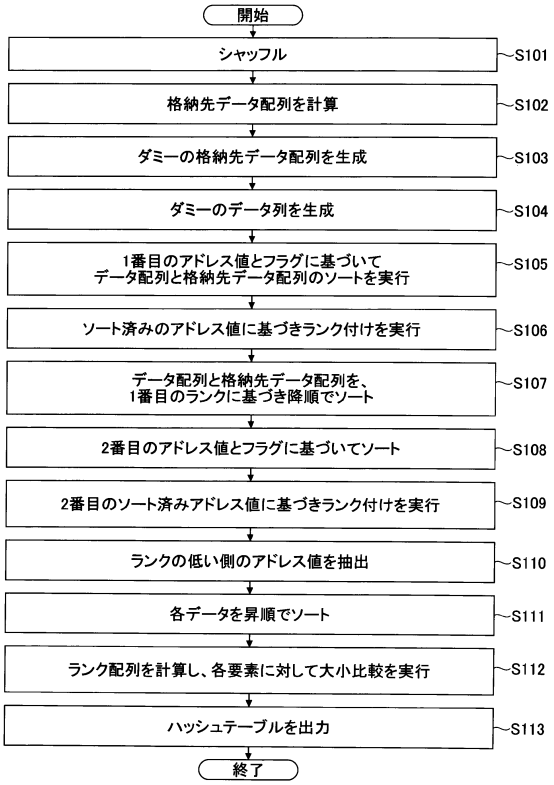
【図２】


100
サーバ
入力部
110
140
データ格納部
演算部
120
出力部
130

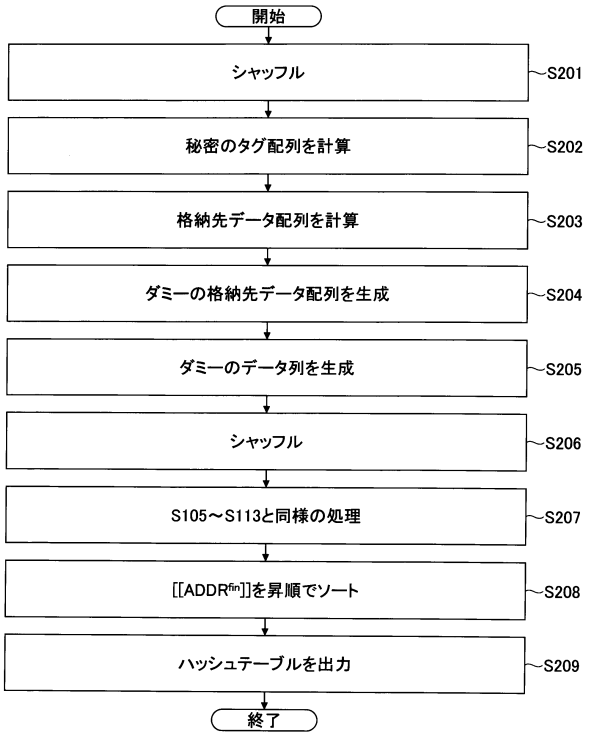
【図３】


B
1008
出力装置
1007
入力装置
1003
メモリ装置
1006
表示装置
1002
補助記憶装置
1005
インタフェース装置
1000
ドライブ装置
1001
記録媒体
1004
CPU

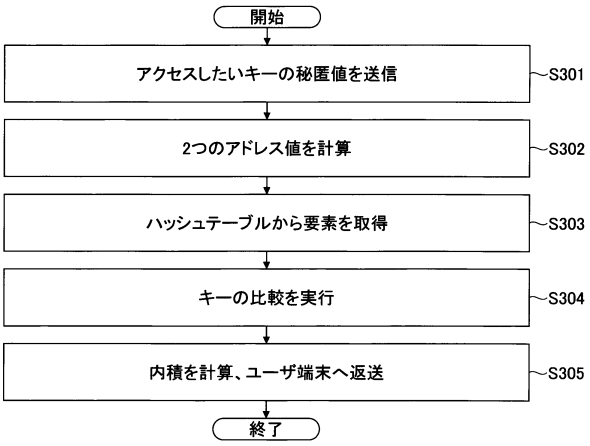
【図４】


開始
シャッフル
S101
格納先データ配列を計算
S102
ダミーの格納先データ配列を生成
S103
ダミーのデータ列を生成
S104
1番目のアドレス値とフラグに基づいて
データ配列と格納先データ配列のソートを実行
S105
ソート済みのアドレス値に基づきランク付けを実行
S106
データ配列と格納先データ配列を、
1番目のランクに基づき降順でソート
S107
2番目のアドレス値とフラグに基づいてソート
S108
2番目のソート済みアドレス値に基づきランク付けを実行
S109
ランクの低い側のアドレス値を抽出
S110
各データを昇順でソート
S111
ランク配列を計算し、各要素に対して大小比較を実行
S112
ハッシュテーブルを出力
S113
終了

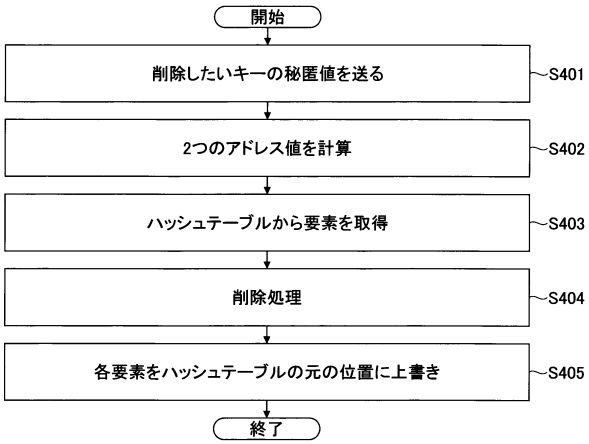
【図５】


開始
シャッフル
S201
秘密のタグ配列を計算
S202
格納先データ配列を計算
S203
ダミーの格納先データ配列を生成
S204
ダミーのデータ列を生成
S205
シャッフル
S206
S105～S113と同様の処理
S207
[[ADDR^fin]]を昇順でソート
S208
ハッシュテーブルを出力
S209
終了

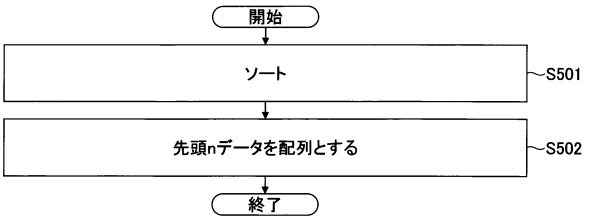
【図６】


開始
アクセスしたいキーの秘匿値を送信
S301
2つのアドレス値を計算
S302
ハッシュテーブルから要素を取得
S303
キーの比較を実行
S304
内積を計算、ユーザ端末へ返送
S305
終了

【図７】


開始
削除したいキーの秘匿値を送る
S401
2つのアドレス値を計算
S402
ハッシュテーブルから要素を取得
S403
削除処理
S404
各要素をハッシュテーブルの元の位置に上書き
S405
終了

【図８】


開始
ソート
S501
先頭nデータを配列とする
S502
終了

フロントページの続き

(56)参考文献 CHAN, T-H. H. et al.，Perfectly Secure Oblivious Parallel RAM，Cryptology ePrint Archive: Report 2018/364，Ver. 20181002:155559，2018年10月02日，pp.1-54，[online], [令和3年3月26日検索], インターネット〈URL:https://eprint.iacr.org/2018/364〉
ASHAROV, G. et al.，OptORAMa: Optimal Oblivious RAM，Cryptology ePrint Archive: Report 2018/892，Ver.20191012:174809，2019年10月12日，pp.1-71，[online], [令和2年1月31日検索], インターネット〈URL:https://eprint.iacr.org/2018/892〉
KUSHILEVITZ, E. et al.，Sub-logarithmic Distributed Oblivious RAM with Small Block Size，v3，2018年11月17日，pp.1-42，[online], [令和2年1月31日検索], インターネット〈URL:https://arxiv.org/abs/1802.05145〉

(58)調査した分野 (Int.Cl.，ＤＢ名)
Ｇ０９Ｃ　１／００