マルチパーティ計算を用いたOblivious RAMの利便性及び安全性の向上 (1) ー 構想と方式概要 ー

西田 直央† 和田 紘帆† 加藤 遼† 吉浦 裕†

†電気通信大学　大学院情報理工学研究科
182-8585 東京都調布市調布ヶ丘 1-5-1
n.nishida@uec.ac.jp , wada.hiroho@uec.ac.jp

あらまし 秘密データをサーバに預けて，クライアントが安全に検索を行えるシステムのニーズが高まっている．これに応える手法の一つに Oblivious RAM があり，データのアクセスパターンを秘匿できる．しかし，クライアントは暗号鍵を管理する必要があり，一人のクライアントの利用に限られる．一方で，秘密分散したデータをマルチパーティ計算で検索する手法がある．アクセスパターンの秘匿は困難だが，鍵の管理が不要で，複数クライアントによる利用が可能である．本稿では，Oblivious RAM に秘密分散とマルチパーティ計算を取り入れて，アクセスパターンを秘匿しつつ，複数クライアントの利用が可能な手法を提案する．

Improving Security of Oblivious RAM (1)

Naohisa Nishida† Hiroho Wada† Ryo Kato† Hiroshi Yoshiura†

†Graduate School of Informatics and Engineering, The University of Electro-Communications
1-5-1, Choufugaoka, Choufu, Tokyo 182-8585, JAPAN
n.nishida@uec.ac.jp , wada.hiroho@uec.ac.jp

Abstract It becomes more and more important to balance information usability and confidentiality. Oblivious RAM is expected to meet this challenge. In this paper, we describe methods to improve security of Oblivious RAM.

1 はじめに

秘密のデータをサーバに預けて，クライアントが自由に検索を行うことが可能なクラウドサーバシステムのニーズが高まっている．その際，サーバがデータの内容を知ることができないことと，検索が効率的に行えることが重要とされている．例えば，クラウドコンピューティングの利用の拡大に伴い，企業は自社のデータベースを外部サーバに預けることも増えてきた．その際，外部からの攻撃者や内部のサーバ管理者等，内外どちらの攻撃者に対してもデータを安全に管理しつつ，クライアントによる遠隔からの利用を可能にする必要がある．

また，近年，データベースの個々のデータを秘匿しても，アクセスパターンを観察することにより，データを統計的に推測できることが明らかになっている．例えば，Pinkas ら [1] は，株取引関係のデータベースに対するアクセスを観察し，特定のアドレスへのアクセス列から特定の株取引を推定できることを明らかにした．また，Islam ら [2] は，暗号化された電子メールリポジトリについて同様の推定が可能であることを示した．したがって，データの値だけではなく，アクセスパターンを秘匿する手法が要求さ

れている．

この要求を満たす技術として，サーバに預けたデータベースを安全に運用する Oblivious Random Access Machine (ORAM) [3][4][5][6][7][8] が注目されている．ORAM は，暗号データに対して演算が行われる度に，その暗号データの格納位置を変化させてデータを管理する方法である．暗号化によるデータの秘匿に加えて，データのアクセスパターンを秘匿することが可能である．内外どちらの攻撃者も，どのデータがアクセスされたかということや検索頻度，さらにデータ間の関係も知ることができない．

ORAM の問題点として，クライアントがサーバにおける各データの格納位置を知っておく必要があるため，データの格納位置の情報を紛失すると，データの検索が不可能になってしまうことが挙げられる．さらに，データを復号するための鍵をクライアントが持つ必要があるため，クライアントの鍵が漏洩すると，サーバに預けたデータが不正に復号される危険性があり，鍵を紛失するとデータの復号が不可能になってしまう．クライアントは IT やセキュリティに精通しているとは限らないため，これらは利便性及び安全性の面で大きな問題となる．さらに，複数のクライアントが同一の鍵を利用すると安全性が低下するため，鍵を所持しているただ一人のクライアントの利用のみに限られてしまうという利便性の問題がある．

一方で，一つのデータベースを複数のデータベースに分散して管理する秘密分散データベース [9][10][11] という手法がある．データは秘密分散法により秘匿されており，一度に閾値以上の秘密分散データベースが侵入されない限り，情報は漏洩しない．各秘密分散データベースの管理者が異なる場合には，閾値以上の管理者が結託しない限り情報は漏洩しない．また，マルチパーティ計算と呼ばれる計算方法を用いることにより，各秘密分散データベースが通信を行い，所持する分散情報を復号することなく，元のデータベースと同様の演算を行うことが可能である．秘密分散データベースは，クライアントが暗号処理を行わないので，鍵を保持する必要が無い．

本稿では，ORAM に秘密分散データベースの手法を取り入れることにより，クライアントによる格納位置及び鍵の管理を不要化する．その結果，クライントに対して ORAM におけるアクセスパターンの秘匿を維持しつつ，クライアントに関わる脆弱性を取り除き，さらに，複数のクライアントによる利用を可能にする手法を提案する．

2 先行研究

2.1 ORAM

ORAM[3][4][5][6][7][8] は，O.Goldreich らにより提案された，ソフトウェアへの不正侵入を防ぐためのシステムである．クライアントの鍵を用いて暗号化されたデータを，サーバ等の外部記憶領域に預ける．そして，クライアントからの検索が行われる度に，アクセスされたデータの格納位置を変更する．ORAM は以下の要件を満たす．

- データの値の秘匿
  サーバに預けたデータは，クライアントの鍵により暗号化されているため，サーバの管理者と外部からの攻撃者共に，データの値を知ることができない．
- アクセスパターンの秘匿
  データがアクセスされる度にデータの格納位置が変更されるので，同一データに対するアクセス頻度を推定することが困難である．また，アクセスを観察しても，データ間の関係を推定することができない．
- 範囲検索の実現
  特定の値による検索だけではなく，二つの値の間に含まれるデータの検索 (範囲検索) が可能である．

ORAM の通信コストは，主に Bandwidth によって見積もられる．Bandwidth は，一つのデータを検索するために必要となるデータの通信量のことである．詳細は 3.5 節で述べる．Bandwidth を小さくするための様々な手法が提

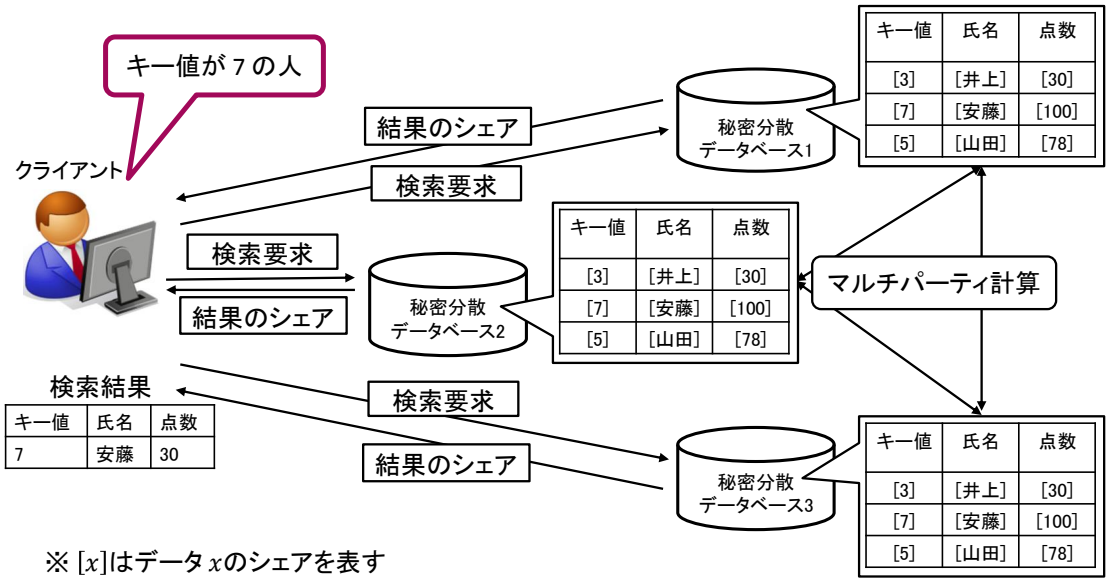

図 1: 秘密分散データベース

案されている．例えば，データ数を M として，E. Kushilevitz ら [6] は，Bandwidth が $O(\log^2 M/\log\log M)$，C.Gentry ら [7] は，$O(\log^3 M/\log\log M)\cdot\omega(1)$，E.Stefanov ら [8] は，$O(\log M)$ でのデータの検索が可能な手法を提案した．

次に，ORAM の問題点を以下に挙げる．

- クライアントによる格納位置の保持

  暗号データの検索のために，クライアントは各データについてサーバ上の格納位置を知っている必要がある．そのため，データの格納位置の情報を紛失すると，データの検索が困難になる．

- クライアントの鍵管理

  サーバに預けるデータは，クライアントの鍵を用いて暗号化されているため，鍵管理が必須である．そのため，クライアントの所持する鍵が漏洩すると，攻撃者によりデータを復号されてしまう危険性がある．また，クライアントが鍵を紛失した際には，データの復号が不可能になってしまう．

- 一クライアントのみの利用

  複数のクライアントが同一の鍵を所持すると，鍵が漏洩する確率が高まるため安全性の低下に繋がる．そのため，同じ鍵で暗号化されたデータに対して一人のクライアントしか演算要求を出すことができない．これにより，利便性が低下する．

2.2 秘密分散データベース

秘密分散データベースは，秘密分散法を用いて一つのデータベースを複数のデータベースに分散して管理する手法である [9][10][11]．具体的な秘密分散法としては，Shamir(k,n) 閾値秘密分散法 [12] や，加法的 (k,n) 閾値秘密分散法が用いられる．また，分散されたデータベース間でマルチパーティ計算 (MPC) を行うことにより，データベースのレコードの値を復号することなく，各種の検索や統計計算を実現する [11]．秘密分散データベースの概要を図 1 に示す．

秘密分散データベースは以下の要件を満たす．

- 外部の攻撃者に対する秘匿
  秘密分散法の性質より，閾値以上の秘密分散データベースが一度に漏洩しない限り，元のデータベースの漏洩には繋がらない．

- 管理者に対する秘匿
  閾値以上の管理者が結託しない限り，管理者は元のデータベースの内容を知りえない．

- 鍵管理の不要化
  データの復号に鍵を用いないので，クライアント及びサーバは鍵を管理する必要がない．

秘密分散データベースの問題点として，MPC に多量の通信を要することが挙げられる．MPC の通信コストは，ラウンド数及び通信量により見積もられる．ラウンド数は，積プロトコルを

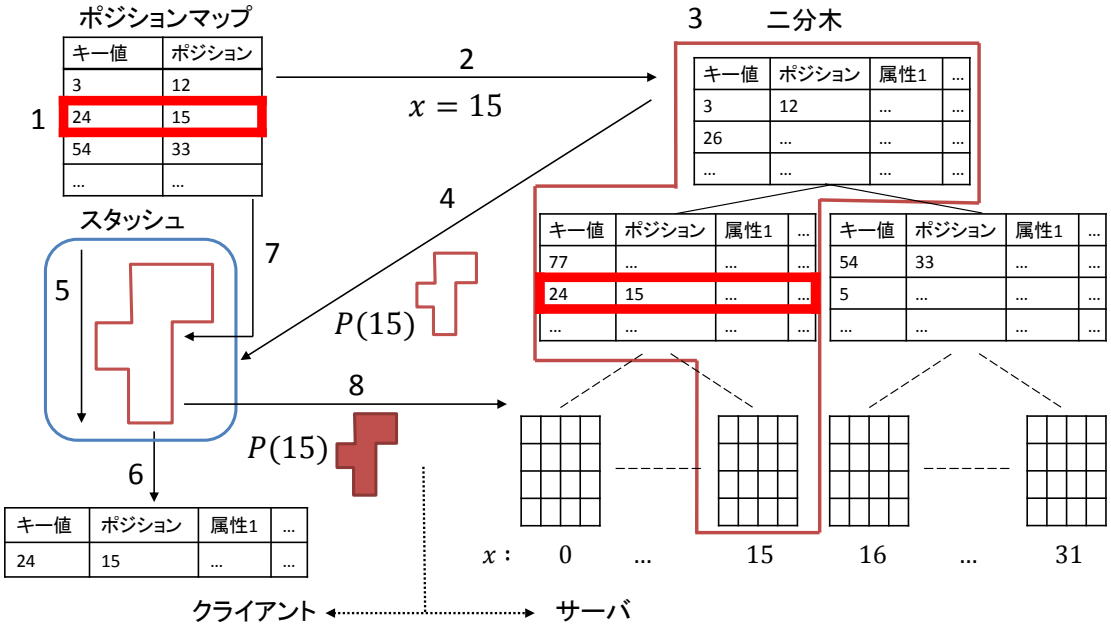

図 2: PathORAM の検索

並列実行した場合の段数を表し，通信量は，当該計算に含まれる積プロトコルの回数を表す．

2.3 達成条件

秘密分散データベースの手法を取り入れることにより，ORAM の問題を解決する．そのため，提案手法が達成すべき条件として以下が挙げられる．

- 鍵管理を不要化し，複数クライアントの利用を可能にする．
- 外部の攻撃者とサーバの管理者に対して，データの値とアクセスパターンを秘匿する．
- MPC の通信コスト (ラウンド数及び通信量) を少なく抑える．

3 PathORAM

本稿では，ORAM の代表例として，現在最も効率が良いとされている E.Stefanov ら [8] による PathORAM を取り上げる．この手法は，検索や書き換えが要求される度に，サーバに預けたデータの木構造を変化させることでアクセスパターンを秘匿する．検索の概要を図 2 に示す．

3.1 サーバの記憶領域

サーバに預けた暗号データは，木構造を用いて記憶される．木は多分木でもよいが，本稿では簡単のために二分木を用いて説明する．

二分木 サーバは深さ $L+1$，葉の数 2^L の二分木を所持する．バケット数を N として，$L = \lceil \log_2 N \rceil - 1$ である．二分木の根を深さ 0 番目，葉を L 番目とする．

バケット 二分木の各ノードをバケットと呼び，各バケットには Z 個の暗号データが格納されている．実際の格納データが Z よりも少なければ，ダミーデータを格納することにより，バケットのサイズ Z を維持する．

パス $x \in \{0, 1, \ldots, 2^L - 1\}$ を，左から x 番目の葉ノードとする．葉ノード x から根ノードへ辿る経路をパスと呼び，$P(x)$ で表す．任意のデータは，一つ以上のパス上に存在する．

3.2 クライアントの記憶領域

クライアントは，スタッシュとポジションマップの二つの記憶領域を平文で所持する．

スタッシュ プロトコルの実行中に，少数のデータが木のバケットから溢れてしまう可能性がある．そのため，クライアントはスタッシュと呼ばれる記憶領域に，これらの溢れたデータを一時的に格納する．

ポジションマップ 二分木における各データの格納位置を表す表である．格納位置をポジショ

ンと呼ぶ．具体的には，各データがどのパス上にあるかを表し，データ d について，$d \in P(x)$ となる x を d のキー値と共に記憶する．データは，$P(x)$ かスタッシュのどちらかに存在する．

3.3 検索・書き換えプロトコル

検索は，以下に示す手順で実行される．例として，キー値 24 のデータを検索するものとする．

1. クライアントはポジションマップを参照して，キー値のポジション情報 $x = 15$ を取得し，ランダムに選ばれた新たなポジション x' に変更する．
2. 取得したポジション情報 $x = 15$ をサーバに送信する．
3. サーバは受け取ったポジションのパス $P(15)$ を取得し，クライアントに送信する．
4. クライアントは受け取ったパス $P(15)$ を復号した後，スタッシュに格納する．
5. スタッシュ内にあるキー値 24 のデータを検索する．
6. 検索終了後，キー値 24 の当該データを出力する．
7. クライアントは，ポジションマップを参照しつつ，スタッシュに格納されている $P(15)$ 上のデータの配置を並び替えて変化させる．
8. 並び替え後の $P(15)$ 上のデータを暗号化してサーバに送信し，二分木のバケットに上書きする．

書き換えの際は，手順 5 の実行時に当該データを新しいデータに書き換え，その後手順 7 から続行する．

3.4 再帰的実行

3.3 節の手順 5 において，パス内の検索にかかる計算量は $\mathrm{O}(\log N)$ で見積もられるため効率が良いが，手順 1 において，ポジションマップ内の検索にかかる計算量は $\mathrm{O}(N)$ で見積もられるため効率が悪い．

これに対し，3.3 節のプロトコルを再帰的に実行する．具体的には，ポジションマップ内のデータを複数個まとめたものを一つの暗号データとして，上述とは別の二分木に格納し，再帰的にプロトコルを実行する．この際，手順 6 では当該データではなく，複数個のデータがまとめられた新しいポジションマップを出力する．

新しい二分木は，木の深さ及び葉ノード数が元の二分木よりも少なくなるため，再帰的に実行することにより，ポジションマップの検索や書き換えにかかる計算量を抑制することが可能である．詳細は，Shi らによる文献 [13] を参照していただきたい．

検索プロトコルにかかる計算量を改めて示す．

一回の再帰により，新しい二分木のサイズが $1/M$ になるとすると，再帰の回数は $\mathrm{O}(\log N/\log M)$ で見積もられる．i 番目の再帰における Bandwidth は $\log(N/\chi^i)$ であるため，再帰的実行にかかる全体の Bandwidth は，$\sum_{i=1}^{n} \log(N/\chi^i)$ で計算され，$\mathrm{O}(\log^2 N/\log \chi)$ で見積もられる．

データはクライアントの鍵により暗号化されているため，サーバはデータの値を知ることができない．さらに，検索の度にサーバの二分木におけるパスを変更するので，アクセスパターンを秘匿することが可能である．安全性の詳細については，Stefanov らによる文献 [8] を参照していただきたい．

一方で，2.1 節で述べた問題点，つまり，クライアントによる格納位置の保持，クライアントの鍵管理，一クライアントのみの利用という問題点は残っている．

3.5 Bandwidth

Bandwidth とは，一つのデータを検索するために送受信されるデータの通信量のことである．PathORAM の場合，サーバは検索プロトコルの手順 3 において，パス上にある $Z \log N$ 個の暗号データをクライアントに送信する．同様に，クライアントは手順 8 において，スタッシュ内にあるパスの $Z \log N$ 個の暗号データをサーバに送

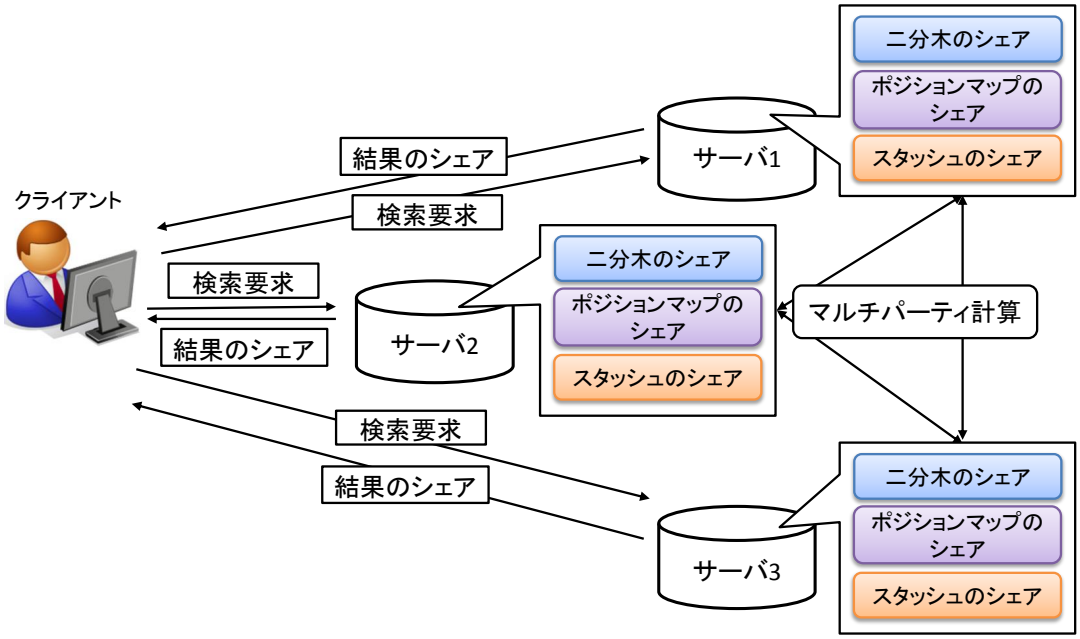

図 3: 秘密分散 PathORAM の概要

信する．結果として，1 回の検索にかかる合計の Bandwidth は $2Z \log N$ となる．ここで，一般に Z は定数であるため，Bandwidth は $O(\log N)$ で見積もられる．また，3.4 節で述べた再帰的実行を適用すると，Bandwidth は $O(\log^2 N)$ で見積もられる．詳細は，Stefanov らによる文献 [8] を参照していただきたい．ここで，データのサイズを可変にすることにより，$O(\log N)$ に抑制することが可能だが，再帰の実行回数が増加してしまう．つまり，サーバとクライアント間の通信回数が増加してしまうため，本稿では $O(\log^2 N)$ を採用することにする．

4 提案手法

本稿では基本方針として，3 章で述べた PathORAM に，2.2 節で述べた秘密分散データベースの手法を取り入れる．

4.1 基本構成

PathORAM に用いる 3 種類の記憶領域（二分木・ポジションマップ・スタッシュ）を全てサーバ側に配置する．概要を図 3 に示す．二分木・ポジションマップ・スタッシュの情報を秘匿するために，これらを秘密分散して各サーバにシェアを格納する．本稿では，Shamir(k, n) 閾値秘密分散法を用いることにする．

以上の構成の上で，3.3 節で述べた PathORAM の検索プロトコルに相当する処理を，マルチパーティ計算 (MPC) により実現する．この構成によると，クライアントはポジションマップやスタッシュ等，何も管理しなくてよいため，クライアントに関わる利便性及び安全性の問題を解決することが可能である．

4.2 秘密分散データベースの記憶領域

各サーバが所持する記憶領域について説明する．以下のデータは全てシェアの形で格納される．

二分木　PathORAM と同様に，バケットにはデータかダミーデータを格納する．

スタッシュ　プロトコルの実行中に溢れたデータを一時的に格納する．

ポジションマップ　二分木における各データのポジションを表す表である．

4.3 課題と目標

提案手法の課題を以下に挙げる．

- クライアント側の処理の変換
  PathORAM では，クライアントは攻撃者として想定されていないので，ポジションマップ及びスタッシュの検索をクライアント側で平文により行うことが可能である．しかし，提案手法ではこれらの処理をサー

バ側で行う．サーバは攻撃者になり得るため，クライアントが平文で行っていた処理を，MPC により安全に計算を行う必要がある．その際，データの値とアクセスパターンを共に秘匿する必要がある．

- MPC による通信コスト
  MPC はサーバ間で通信を行う必要があるため，平文で行えていた計算を MPC に変換すると，サーバ間の通信コストが膨大になってしまう．そのため，通信コストを抑制する必要がある．また，ORAM では通信コストを Bandwidth で見積もっていたが，提案手法は MPC を用いるので，ラウンド数及び通信量で見積もることにする．

次に，提案手法の通信コストに関する目標を以下に挙げる．

- ラウンド数の維持
  PathORAM において，サーバとクライアント間の通信は $O(\log N)$ ラウンドで見積もられるため，このラウンド数を維持する．つまり，全体のラウンド数を (マルチパーティプロトコルのラウンド数)·$O(\log N)$ にする．
- Bandwidth の維持
  PathORAM の Bandwidth は $O(\log^2 N)$ で見積もられるため，このコストを維持する．つまり，通信量を (マルチパーティプロトコルの通信量)·$O(\log^2 N)$ にする．

5 主要プロトコル

主要プロトコルを以下に示す．具体的なプロトコルについては，著者らによる文献 [14] を参照していただきたい．

5.1 ポジションマップの参照プロトコル

与えられたキー値のポジション情報を出力する．キー値及びポジション情報はシェアになっているため，全てのキー値に対してマルチパーティプロトコルを行う必要がある．計算量は $O(N)$ になってしまい効率が悪いが，5.3 節で述べるポジションマップの参照を再帰的に行うことにより，この問題を解決する．

5.2 検索プロトコル

秘密分散された二分木・スタッシュ・ポジションマップを入力して，3.3 節で述べた検索プロトコルと等価な検索処理を行う．この際，各サーバに対してデータの値及びアクセスパターンを秘匿する．

5.3 再帰的実行プロトコル

上述した検索プロトコルを再帰的に実行し，3.4 節の再帰的実行と等価な処理を実現する．

6 提案手法の貢献

6.1 ORAM の利便性及び安全性の向上

従来の ORAM では，クライアントがポジションマップを保持したり，鍵管理を行う必要がある．しかし，一般に，クライアントは IT やセキュリティに精通しているとは限らないため，ポジションマップを紛失したり，鍵を漏洩させてしまう危険性がある．これに対し，秘密分散データベースの手法を取り入れることにより，クライアントによる鍵管理を行う必要が無くなるため，利便性及び安全性が向上すると考えられる．

また，クライアントによる鍵管理の問題の解消に伴い，複数のクライアントによる利用が可能になるため，利便性が向上すると考えられる．利便性向上のイメージ図を図 4 に示す．

6.2 秘密分散データベースの安全性の向上

秘密分散データベースは，ハッシュテーブルなどを用いることにより，アクセスパターンを秘匿できる可能性もあるが，そのことと範囲検索を両立することは困難だと考えられる．しかし，ORAM は範囲検索が可能であることから，

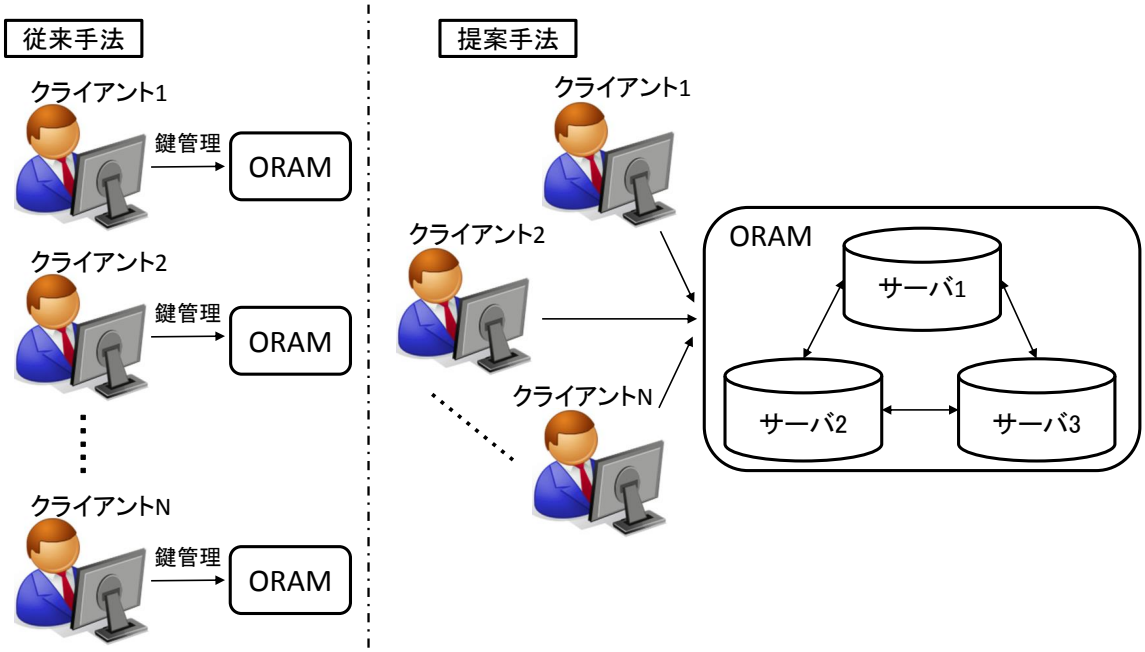

図 4: 提案手法による利便性の向上

提案手法では秘密分散データベースにおいて，アクセスパターンの秘匿と範囲検索を両立できる可能性がある．

7 結論と今後の課題

ORAM に秘密分散とマルチパーティ計算を取り入れて，アクセスパターン，特に検索頻度を秘匿しつつ，複数クライアントの利用が可能な手法を提案した．提案手法では，鍵管理や鍵漏洩時のセキュリティの問題，消失時の検索不能性の問題を解決することが可能である．

今後の課題は，5 章で述べたプロトコルを設計し，コストと安全性を評価することである．

参考文献

[1] B.Pinkas and T.Reinman. Oblivious ram revisites. In CRYPTO, 2010.

[2] M.Islam, M.Kuzu, and M.Kantarcioglu. Access pattern disclosure on searchable encryption:Ramification, attack and mitigation. In NDSS, 2012.

[3] O.Goldreich, and R.Ostrovsky. Software protection and simulation on oblivious rams. In JACM, pp.431-473, 1996.

[4] O.Goldreich. Towards a theory of software protection and simulation by oblivious rams. In STOC, pp.182-194, 1987.

[5] R.Ostrovsky. Efficient computation on oblivious rams. In STOC, pp.514-523, 1990.

[6] E.Kushilevitz, S.Lu, and R.Ostrovsky. On the (in)security of hash-based oblivious RAM and a new balancing scheme. In SODA, 2012.

[7] C.Gentry, K.Goldman, S.Halevi, C.Julta, M.Raykova, and D.Wichs. Optimizing oram and using it efficiently for secure computation. In PETS, pp1-18, 2013.

[8] E.Stefanov, M.Dijk, E.Shi, C.Fletcher, L.Ren, X.Yu, and S.Devadas. Path ORAM: An Extremely Simple Oblivious RAM Protocol. In CCS, pp. 299-310, 2013.

[9] 桜井友二, 齊藤泰一. 情報漏えい防止データベースシステムの提案. In SCIS, 2006.

[10] 濱田浩気, 五十嵐大, 菊池亮, 千田浩司, 諸橋玄武, 冨士仁, 高橋克巳. 実用的な速度で統計分析が可能な秘密計算システム MEVAL. In CSS, 2013.

[11] 志村正法, 遠藤つかさ, 宮崎邦彦, 吉浦裕. 安全で機能制限のないデータベースを実現するマルチパーティプロトコルを用いた関係代数演算. In 情報処理学会研究報告. CSEC, pp.187-193, 2008.

[12] A.Shamir. How to Share a Secret. In Communications of the ACM, Vol.22,No.11, pp.612-613, 1979.

[13] E.Shi, T.-H.H.Chan, E.Stefanov, and M.Li. Oblivious RAM with $O(\log N)^3$ worst-case cost. In ASIACRYPT, pp.197-214, 2011.

[14] 和田紘帆, 西田直央, 加藤遼, 吉浦裕. マルチパーティ計算を用いた Oblivious RAM の利便性及び安全性の向上 (2)— プロトコルの設計と評価 —. In CSS, 2014.