単一の鍵で多重帰属できるグループファイル共有システムの評価

長澤　悠貴†　白石　善明†　毛利　公美‡　福田　洋治††　野口　亮司‡‡
名古屋工業大学†　岐阜大学‡　愛知教育大学††　(株)豊通シスコム‡‡

1. はじめに

クラウドコンピューティングの拡大により，クラウド内のサーバで提供されるサービスの利用場面が拡大している．クラウドサービスの一つにグループファイル共有サービスがあるが，ファイルをサービス提供側のサーバで保管するため，サービス提供者の不正が懸念される．一方，サービス提供者が全てのファイルを厳重に管理することは困難であり，利用者側で対策をしてもらうことが望ましい．我々は既に，利用者側での暗号化によりサービス提供者にファイルの内容を知られず，かつ利用者が複数のグループに多重帰属する場合でも鍵管理の負担を軽減するための単一の鍵で多重帰属できるグループファイル共有プロトコル[1]を提案している．本稿では，提案プロトコルのスケーラビリティを評価し，クラウドサービスとして実用的に利用できることを示す．

2. 単一の鍵で多重帰属できるグループファイル共有システム

我々が提案している単一の鍵で多重帰属できるグループファイル共有システムでは，ファイルを共通鍵方式によりファイル復号鍵で暗号化・復号し(図 1の①)，そのファイル復号鍵をグループで公開鍵暗号方式により，グループ公開鍵で暗号化(図 1の②)し，グループ秘密鍵で復号(図 1の③)して共有することを前提としている．そのグループ秘密鍵をグループメンバで共有することでアクセス制御を行う．サーバとグループメンバ間で(2,2)閾値秘密分散法を繰り返し適用することで，グループ秘密鍵を分散管理する．分散して生成された二つの鍵のうち，メンバが管理する鍵を**メンバ部分復号鍵**，サーバが管理する鍵を**サーバ部分復号鍵**と呼ぶ．分散管理しているグループ秘密鍵は，サーバと任意のメンバの二者が協力すれば復号可能である．グループ秘密鍵を分散して管理する様子を図 1に示す．以降，グループ秘密鍵を分散して形成された木を**鍵分散木**と呼ぶ．

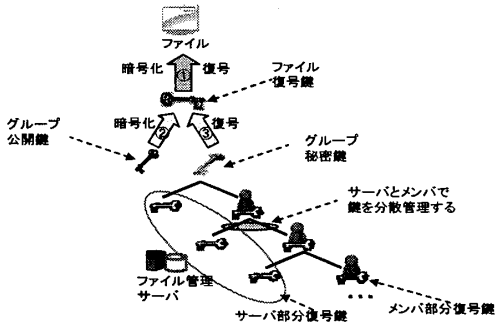

図 1：提案システム内で使用する鍵の関係とグループ秘密鍵を管理する様子

提案システムは以下の5種類のプロトコルから構成される．

[グループ構築プロトコル]
サーバとグループメンバ一人が分散情報生成プロトコル[2]を実行してグループ秘密鍵を生成する．グループ秘密鍵の生成後，グループ公開鍵を生成する．

[メンバ追加プロトコル]
鍵分散木のリーフノードメンバのメンバ部分復号鍵に(2,2)閾値秘密分散法を適用し，サーバと新規メンバに分散することで，新規メンバをグループに追加する(図 2の①)．新規メンバが既に他のグループに所属している場合は，文献[1]で提案しているシェア・コントロールを新規メンバが実行し，サーバ部分復号鍵を調節することで(図 2の②)，新規メンバは親ノードにあたるメンバ部分復号鍵との関係を維持しながら，現在のメンバ部分復号鍵でファイル取得が可能になる．

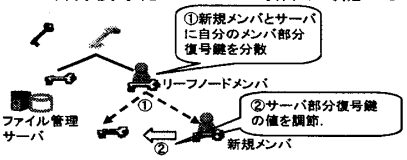

図 2：メンバ追加プロトコルを実行する様子

[メンバ離脱プロトコル]
離脱メンバの親ノードのメンバ部分復号鍵を，サーバと離脱メンバの子ノードメンバに再分散して離脱メンバのメンバ部分復号鍵をそのグループでは使用できないようにする．メンバ離脱プロトコル実行前後の鍵分散木の様子を図 3に示す．

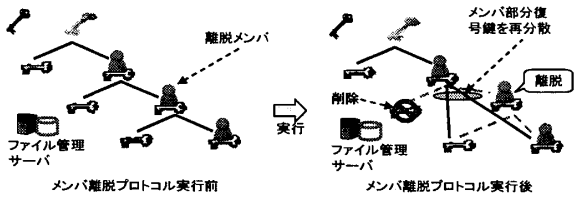

図 3：メンバ離脱プロトコル実行前後の様子

[鍵更新プロトコル]
サーバと全グループメンバが協力し，鍵分散木の各ノード間の関係を維持しながらメンバ部分復号鍵を更新する．関係を維持するために，ルートメンバから順に実行し，子ノードメンバに自身のメンバ部分復号鍵の更新量を伝播させる必要がある．また，鍵更新プロトコルの実行中は鍵分散木の構造が変化するメンバ追加プロトコルとメンバ離脱プロトコル，メンバ部分復号鍵とサーバ部分復号鍵を使用するファイル共有プロトコルを実行することはできない．鍵更新プロトコル実行の様子を図 4に示す．①，②，③は鍵更新プロトコルの実行順を表す．

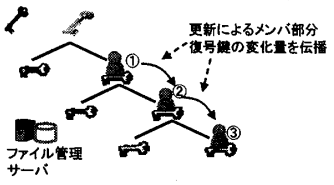

図 4：鍵更新プロトコル実行の様子

[ファイル共有プロトコル]
メンバがグループ公開鍵でファイルを暗号化する．サーバとメンバでグループ秘密鍵を復号し，ファイルを復号する．

このうち，鍵分散木の深さが実行時間に影響すると考えられる鍵更新プロトコルについてシミュレーションを行い，木の深さがプロトコルの実行時間にどのような影響を与えるかを調べる．なお，他のプロトコルは木の深さなどに依存しないので評価の対象としない．

3. 評価用シミュレータの構築

システム全体の振る舞いを再現するために，マルチエージェントシミュレーションを行う．

Evaluation of the Key Management in Multiple Association With A Single Key For A Group File Sharing System

†Yuuki NAGASAWA and Yoshiaki SHIRAISHI・Nagoya Institute of Technology
‡Masami MOHRI・Gifu University
††Youji FUKUTA・Aichi University of Education
‡‡Ryoji NOGUCHI・Toyotsu Syscom Corporation

本シミュレーションはターン毎に 1)新規タスク生成フェーズ，2)鍵更新プロトコル実行の判定フェーズ，3)ホスト行動フェーズという順に実行する．新規タスク生成フェーズでは，ホスト一台に対して鍵更新プロトコル以外のプロトコルを生成する．鍵更新プロトコル実行の判定フェーズでは，最後に更新が行われてから一定ターンが経過していれば所属メンバに対して鍵更新プロトコルを生成する．ホスト行動フェーズではホストがプロトコルを実行する．

次にシミュレータの構成要素を説明する．本シミュレータはホスト，サーバ，ファイル，グループ，ホスト管理者により構成される．

[ホスト]

サービス利用者が使用する端末．利用者は各端末を利用してファイル共有などを行う．ホストは複数のタスクを同時に処理することはできず，処理しきれないタスクがある場合はキューに保持する．図 5に示したホスト行動フェーズでは，プロトコル実行の対象グループが更新中であればプロトコルを実行できない．この場合，ホストは実行中のタスクをキューへ戻し，次のタスクをキューから取り出して別のプロトコルの実行を試みる．ただし，キューからタスクを取り出せるのは 1 ターンにつき 1 タスクまでである．

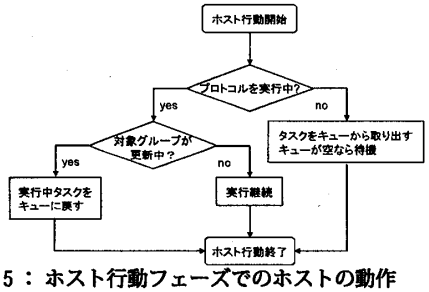

図 5 ： ホスト行動フェーズでのホストの動作

ホストがとりうる状態は，各プロトコルの実行中である，グループ構築プロトコル実行中，メンバ追加プロトコル実行中・メンバ離脱プロトコル実行中・鍵更新プロトコル実行中・ファイル取得中・ファイル保管中，プロトコルを実行していない待機中，サーバのコネクション確立ができなかったときに遷移するコネクション確立待ちの 8 状態である．ホストの状態遷移図を図 6に示す．

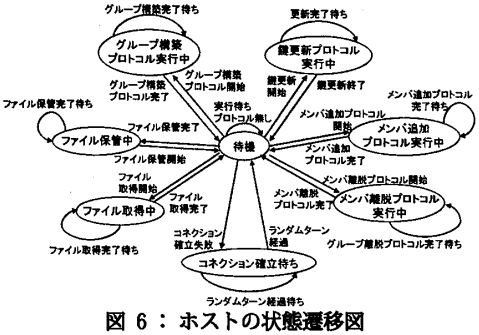

図 6 ： ホストの状態遷移図

[サーバ]

サービス提供者のサーバ．各グループのサーバが管理する鍵とファイルの保管，ホストからリクエストのあったプロトコルを実行する．また，グループ情報を管理する

[グループ]

ホストによって形成されるグループ．この単位でファイル共有を行う．

[ファイル]

グループで共有されるファイル．ファイルサイズはファイル共有プロトコル生成時に決定され，サイズの分布は Double Pareto Distributions[3]に従う．

[シミュレーションコントローラ]

新規タスクの生成など，シミュレータの各要素の動作管理を行う．

4. シミュレーションによる評価

4. 1. 実験目的

鍵分散木の深さに応じて鍵更新プロトコルの実行時間がどのように変化するかを調べる．

4. 2. 実験条件

シミュレーションに登場するホスト数は 2048 台で，通信スループット，通信遅延は全て 10Mbps と 30ms になっている．各プロトコルの生成確率は，ファイル共有プロトコルを 1 日 100 回実行し，グループ生成プロトコル・メンバ追加プロトコル・メンバ離脱プロトコルは 1 ヶ月に 3 回の頻度で実行されると考え，ファイル共有プロトコルを 0.7，他の 3 つのプロトコルを 0.0007 とし．残りの 0.2979 を待機に割り当てた．鍵更新プロトコルは，一定ターン毎に実行される．サーバの同時最大接続数は，ホストのリクエストには全て応答可能となるよう 2048 とし．最大ファイルサイズは 100GB，最大グループ数は 100 グループとした．

また，事前に構成要素ごとにプロトコルの処理時間を測定し，各プロトコルの処理を 1 ターンで完了することができるように 1 ターンを 150 ミリ秒と定義した．シミュレーション実行前に，メンバ数が 2048 人のグループが 1 グループ存在する状況でシミュレータを 1 ヶ月分実行しておき，システムがある程度使用されている状況を再現しておく．この状態をシミュレーションの初期状態として評価を行う．このとき，グループ数は 100 とし，2048 人のグループ以外のメンバ数は数人程度である．

4. 3. 実験結果とまとめ

実験結果を図 7に示す．このとき，メンバ数が 2048 人のグループが一つ，メンバ数が数人のグループが 99 グループ存在している．結果より，鍵分散木の深さが 2000 のとき，鍵更新プロトコルが実行に要するターンは約 7000 ターンであることがわかる．これを実時間に換算すると約 20 分弱である．通常，システムをメンバが 24 時間連続して利用することは無いと考えられるため，鍵更新プロトコルの実行はシステムが利用されない時間帯(真夜中など)に行えばシステムとして問題なく利用できることがわかる．あわせて，メンバ数が 2000 人を超えるグループが存在することは考えづらく，通常は高々40 人程度であろう．このようなグループが 100 グループ存在する場合の鍵更新プロトコルの実行時間は 15 秒程度であり，もし鍵更新プロトコル実行中にシステムを利用する場合でも利用者の業務に支障をきたさないと考えられる．よって，鍵更新プロトコルがシステムに悪影響を与えることはないことが確認できた．今後は鍵分散木の深さがファイル共有プロトコルの鍵復号部にどのような影響を与えるかを評価する予定である．

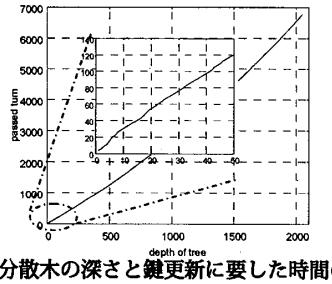

図 7 ： 鍵分散木の深さと鍵更新に要した時間の関係

参考文献

[1] 内田真理子 他, “多重帰属の鍵管理が容易な(2,2)閾値秘密分散を用いたグループファイル共有”, 信学技報, ISEC2008-113, 2009 年 3 月.

[2] T. Pedersen, “A threshold cryptosystem without a trusted party, ”Proc. of Eurocrypt’91, LNCS No.547.

[3] Michael Mitzenmacher, ”Dynamic Models for File Sizes and Double Pareto Distributions”, Harvard Univ., Internet Mathematics Vol. I, No. 3: 305-333