卒業論文　　2011年度(平成23年度)

MIKEYとLKHを用いた時間単位によるグループ共有鍵更新

慶應義塾大学 環境情報学部　環境情報学科

佐藤　淳哉

指導教員

慶應義塾大学 環境情報学部

村井 純

徳田　英幸

中村 修

楠本 博之

高汐 一紀

Rodney D.Van Meter

植原 啓介

三次　仁

中澤　仁

武田 圭史

慶應義塾大学 大学院 政策・メディア研究科

朝枝 仁

平成24年2月14日

卒業論文要旨 2012 年度 (平成 24 年度)

MIKEY と LKH を用いた 時間単位による共有グループ鍵更新

本研究では，Multimedia Internet KEYing (MIKEY) と Logical Key Hierarchy (LKH) を用いた時間単位による共有グループ鍵更新の設計と実装・評価を行う.「多数のコンテンツ」と「多数のユーザ」が想定される環境での pay-per-view 型リアルタイムセキュアストリーミングにおいて, 効率的なユーザ管理と鍵更新にかかるコスト削減を行えるシステムを実現した。

セキュアな通信を実現するためには，認証・認可，鍵の更新, 暗号化・復号化などの処理が必要になる.「多数のコンテンツ」を管理する為には相応の鍵数が必要となるため、その鍵を更新し「多数のユーザ」に配信する場合、コストが膨大になりサーバに過度の負担が強いられる。従って，コンテンツ数やユーザ数に影響を受けにくい鍵の更新システムが必要である．本提案手法では，pay-per-view 型リアルタイムセキュアストリーミングに適応するため，サーバは定めた時間ごとに鍵を更新する．また，更新時間の異なる 2 種類の鍵更新を行うことにより，コンテンツの「Forward Secrecy と Backward Secrecy の保護」と「視聴権限によるユーザの区別」を可能とし、コンテンツ数、及びユーザ数に影響を受けにくい鍵管理システムを構築する。

Forward Secrecy と Backward Secrecy の保護を目的とした比較的短時間で行う鍵更新は、MIKEY を改良し実現する．また，コンテンツ時間によるユーザの区別を目的とした比較的長時間での鍵更新は，鍵を特定のユーザにのみ配信する事が可能な LKH を用いて実現する．ユーザをコンテンツ時間毎に区別することで，視聴可能なコンテンツが終了したユーザには鍵が配信されない機能を有する.

評価では、MIKEY のみの場合、LKH のみの場合，及び二つを組み合わせた場合において訂正評価・定量評価を行った．定性評価では，コンテンツグループや視聴権限グループの管理を評価項目とし、定量評価では、データ受信後の SA 作成時間を計測した．これらの評価により、本研究が提案するシステムが効率的なユーザ管理と鍵更新にかかるコスト削減を行えることを確認した。

キーワード
1．ストリーミング, 2．IP マルチキャスト, 3．鍵の更新,4．コンテンツ管理, 5．視聴権限管理

慶應義塾大学　環境情報学部

佐藤 淳哉

Abstract of graduation's Thesis

Academic Year 2012

Time-Based Shared Group Key Rekeying using MIKEY and LKH

In this thesis, a novel time-based rekey architecture is proposed for rekeying a large number of shared group keys. The architecture adopts the standard Multimedia Internet KEYing (MIKEY) and Logical Key Hierarchy (LKH) functions as the base component. It reduces the rekey costs for real-time streaming services rekeyed on a time basis. This time-based rekey architecture contributes to the improvement of the situation in which a large number of secure real-time streams uses different shared keys and many users having different conditions (i.e., pay-per-view contracts) to decode secure real-time streams. Thanks to the combination of MIKEY and LKH, we can differentiate rekey timing to protect both forward and backward secrecy by short timing rekey (e.g., 1 min.) and distinguish users by long time rekey (e.g., 30 min.). Long time rekey is done by the LKH based mechanism that changes a traffic-encrypting key (TEK), and short time rekey is done by the MIKEY based mechanism that changes a TEK Generation Key (TGK).
In this thesis, both qualitative and quantitative evaluation are described. For qualitative evaluation, we confirmed whether the proposed architecture implemented content group and user management. For quantitative evaluation, we measured the time for IPsec tunnel (i.e., Security Association) establishment.

Keywords :
1. Streaming, 2. IPmulticast, 3.Rekeying ,
4. Content Management System, 5. Time Management System

Keio University,Faculty of Environment and Information Studies

Junya Satou

目 次

第 1 章　序論 **1**
1.1 背景 1
1.2 目的 2
1.3 構成 3

第 2 章　要素技術 **4**
2.1 鍵の配信 4
2.2 鍵の更新 4
2.2.1 更新の必要性 5
2.2.2 鍵の更新手法 5
2.2.3 鍵の更新プロトコル 7
2.3 鍵の管理プロトコル 9
2.3.1 GSAKMP 9
2.3.2 GDOI 10
2.3.3 MIKEY 11

第 3 章　関連研究と問題点 **13**
3.1 MIKEY 13
3.2 An Architecture and Key Management Approach for Maintaining Privacy in Location Based Group Services 15

第 4 章　アプローチ **17**
4.1 機能要件 17
4.1.1 IP マルチキャストを用いた鍵更新 17
4.1.2 短時間の鍵更新 18
4.1.3 長時間の鍵更新 19

第 5 章　設計 **21**
5.1 Time Driven LKH 21
5.1.1 鍵配信モジュール 21
5.1.2 鍵更新モジュール 24
5.1.3 SA 作成モジュール 27
5.2 Time Driven MIKEY 28
5.2.1 seed 配信モジュール 28

5.2.2 鍵更新モジュール 29
5.3 受信ノード 29
5.3.1 鍵作成モジュール 29
5.3.2 SA 作成モジュール 30
5.3.3 動画配信モジュール 30

第 6 章 実装 **31**
6.1 実装環境 31
6.2 Time Driven LKH サーバ 31
6.2.1 鍵配信モジュール 31
6.2.2 鍵更新モジュール 34
6.2.3 SA 作成モジュール 36
6.3 Time Driven MIKEY サーバ 38
6.3.1 seed 配信モジュール 38
6.3.2 鍵更新モジュール 38
6.3.3 SA 作成モジュール 38
6.4 受信ノード 39
6.4.1 鍵作成モジュール 40
6.4.2 SA 作成モジュール 41
6.4.3 動画配信モジュール 41

第 7 章 評価 **43**
7.1 評価概要 43
7.2 Time Driven LKH 43
7.2.1 定性評価 43
7.2.2 定量評価 46
7.3 Time Driven MIKEY 48
7.3.1 定性評価 48
7.3.2 定量評価 50
7.4 Time Driven LKH と Time Driven MIKEY 51
7.4.1 定性評価 52

第 8 章 結論 **55**
8.1 まとめ 55
8.2 今後の課題 55
8.2.1 パケットロス対策 56
8.2.2 鍵の配信 56
8.2.3 insert 機能 56
8.2.4 鍵の更新 57
8.2.5 sub key SA 作成 57

図 目 次

2.1 イベント単位の鍵更新概要図 6
2.2 時間単位の鍵更新概要図 7
2.3 LKH 概要図 8
2.4 OFT 概要図 10
2.5 MIKEY 概要図 12

3.1 seed 更新に関する問題点 14
3.2 KEK 更新に関する問題点 15
3.3 LKH-MIKEY 概要図 16

4.1 ２種類の鍵更新概要図 19

5.1 全体動作概要図 22
5.2 LKH モジュール図 23
5.3 Time Driven LKH ツリー概要図 23
5.4 視聴権利情報概要図 24
5.5 2 本のツリー作成図 25
5.6 視聴時間での鍵更新図 26
5.7 insert モジュール図 27
5.8 MIKEY モジュール図 28

6.1 tree_node 構造体 32
6.2 Time Driven LKH ツリー概要図 33
6.3 LKH 構造体 34
6.4 sleep time 算出方法 36
6.5 rekey 構造体 37
6.6 KEK，sub key 配信用 SA 作成コマンド 37
6.7 seed 作成 39
6.8 seed 配信用 SA 作成コマンド 40
6.9 TEK 作成 40
6.10 コンテンツ配信用 SA 作成コマンド 41
6.11 vlc 起動コマンド 42

7.1 テスト環境 44
7.2 乱れた画像 45

7.3 SA 切り替え 45
7.4 正常な画像 46
7.5 tcpdump の結果 47
7.6 テスト環境 49
7.7 コンテンツ受信者 A(GID63996) の画像 50
7.8 コンテンツ受信者 B(GID63997) の画像 50
7.9 tcpdump の結果 51
7.10 テスト環境 52
7.11 LKH の結果 53
7.12 MIKEY の結果 53
7.13 vlc の結果 54

表 目 次

6.1 実装環境 31

7.1 計測結果 47
7.2 計測結果 50

第1章　序論

1.1　背景

ニールセン・オンライン (Nielsen Online) [1] が提供するインターネット利用者動向調査「Net View」 [2] によると，ニコニコ生放送 [3] や Ustream [4] などの 1 対 n 型リアルタイムストリーミングサービス利用者が増加している．2009 年 4 月期にはニコニコ生放送の利用者は 42 万人，Ustream の利用者は 91 万人であったのに対し，1 年後の 2010 年 4 月期にはニコニコ生放送利用者は 138 万人を突破，Ustream の利用者も 99 万人を突破した．2011 年 8 月 2 日からは，Android 版ニコニコ生放送が開始し，スマートフォンで PC と同等の画質でのストリーミングが可能となった．これにより，ストリーミング利用者が増加し，今後も増加が予想される．また，Ustream も，宇多田ヒカルの公演生中継や，タイガー・ウッズの不倫謝罪会見，TBS テレビ「地球同時多発情報 SHOW　革命×テレビ」 [5] でのインタビュー生中継など，様々な用途で使用され，その利用者数を増やしている．また，Skype などのアプリを使ったライブチャットも一般化している．ライブチャットは，企業や研究会などのミーティングで主に利用されている．文字だけのコミュニケーションには限界がある．だが、ライブチャットなどの映像ツールを使う事で，より多様なコミュニケーションが可能となる．ストリーミング利用者の増加やサービスの多様化などからストリーミングの重要性が高まっている事が分かる．

また，インターネット有料放送利用者も増加している．インターネット有料放送サービスを行っている U-NEXT [6] では 2008 年 9 月から 2009 年 8 月に加入者が約 2 万 7000 件純増し，合計約 10 万件を超えた [7]．Ustream も 2010 年 10 月に，配信ユーザが自身のストリーミングに視聴料金を設定する事ができる「Ustream Pay-Per-View」 [8] サービスを開始し，個人でのインターネット有料放送が容易になった．これにより，今までよりもインターネット有料放送の可能性が広がるとともに，有料放送を可能としている技術，セキュアストリーミングの重要性も高まっている．セキュアストリーミングはストリーミングデータを暗号化し，視聴権限のあるユーザにのみ復号化を可能とする技術である．コンテンツを買ったユーザにのみストリーミングが視聴できるようにしている．リアルタイムストリーミング利用者の増加やこういった新しいビジネスモデルの創出などから，今後，セキュアリアルタイムストリーミングの利用者は増加していくと予想される．

Web ページを共有し，資料の共有やオンラインミーティングができる Cisco WebEx Meeting Center [9] や Skype [10] などのアプリを使ったビデオ会議など，セキュアストリーミングが一般的に利用されるようになった．これらは，ストリーミングデータを暗号化し，ユーザ以外にはミーティング内容が漏えいしないようにしている．Skype を利用する事ができるテレビ「VIERA（ビエラ）」 [11] の発売からもわかるように，今後もスト

リーミングサービスは多様化していくことが予想される．また，Skype などのストリーミングアプリが普及するとともに，ビデオ会議などのセキュアリアルタイムストリーミングの利用者やコンテンツ数は爆発的に増加すると考えられる．

しかし，利用者数やコンテンツ数が増加した際，既存のセキュアストリーミングシステムでは問題が生じる．セキュアストリーミングではストリーミングデータの暗号化に鍵暗号方式を主に使用する．ストリーミングデータを鍵と呼ばれる値によって暗号化して配信する．ストリーミングデータが盗聴されても，データ内容が暗号化されている為，盗聴者に漏えいする事はなく，復号化する為の鍵を持っているユーザだけが，正常なストリーミングデータを受信できる．しかし，この鍵は使用し続けるほどに盗難の危険性が増えたり，偶発的に鍵が当てられるといった脆弱性が発生してしまう．よって，暗号化や復号化する鍵の値を変え，鍵の盗難や脆弱性に対応する必要がある．しかし，利用者数やコンテンツ数の増加に伴い，鍵を配信するサーバの計算処理コストが膨大になる．このため，利用者数やコンテンツ数に影響を受けにくい鍵更新システムが必要となる．

1.2 目的

インターネット有料放送や企業内での遠隔会議などのセキュアリアルタイムストリーミングは，視聴を許可されたユーザ以外には視聴されてはいけない．ニコニコ生放送の有料放送では，チケット購入者以外には放送が視聴されてはいけない．チケット，つまり，鍵を持っているユーザのみが暗号化されたストリーミングデータを復号化出来なければならない．

インターネット有料放送の場合，一つの事業者がドラマチャンネルやスポーツチャンネルなど複数のチャンネルを持ち，時間によって区切られたコンテンツの配信をすると考えられる．22 時から 23 時まで放送の 1 時間枠ドラマや 25 時から 28 時の海外サッカー試合生放送などのコンテンツを管理できなければならない．コンテンツの管理には鍵の管理技術が重要になってくる．ストリーミングデータをコンテンツごとに暗号化するには，コンテンツの数だけ鍵が必要になる．また，鍵一つで暗号化し続けると，鍵を盗まれたとき，全てのコンテンツが悪意あるユーザに視聴されてしまう可能性がある為，鍵はコンテンツの時間内に更新される必要がある．しかし,「多数のコンテンツ」と「多数のユーザ」が想定される場合，鍵更新などにかかるサーバの負担は非常に大きくなる．

本研究では，既存の技術が対応しきれない「多数のコンテンツ」と「多数のユーザ」が想定される環境で，一つの映像コンテンツごとに課金を行う pay-per-view 型リアルタイムセキュアストリーミングの実現を目的とする．鍵の管理，なかでも，鍵の更新に着目し，効率的なユーザ管理と，鍵更新時のユーザに対する鍵配布にかかる計算処理コストを低減する鍵管理システムの構築を目的とする．

1.3 構成

本論文は全 8 章から構成される．第 2 章では，本研究における要素技術である鍵の管理手法について述べ，鍵管理に必要な技術と既存のプロトコルについて説明する．第 3 章では，既存の研究などを考察し，その問題点を述べる．第 4 章では，第 3 章で挙げた問題を解決する為のアプローチについて述べる．第 5 章では，システムの設計，第 6 章では実装について述べる．そして，第 7 章では本システムの定性評価と定量評価を行い，第 8 章で本研究の結論を述べ，今後の課題を示す．

第2章　要素技術

本節では,「多数のコンテンツ」と「多数のユーザ」が想定される環境での鍵管理において，特に重要な技術である鍵の配信と鍵の更新に関する技術について述べる．

2.1　鍵の配信

鍵の配信方法には，１対１型のユニキャスト通信で鍵を配信する方法と，1 対ｎ型のマルチキャスト通信で鍵を配信する方法がある．ユニキャスト通信で鍵を配る場合は，クライアントとサーバが 1 対 1 の通信を行う．そのため，クライアントの数が増えれば増えるほど通信量も増加し，サーバの負荷が増大する．

マルチキャスト通信で鍵を配信する場合，1 対ｎ型の通信となるため，ユーザの数が増えてもサーバの負担は増加しない．ユーザが鍵を配信する際の通信コストは変化しないため，サーバの負荷も変化しない．しかし，鍵の配信経路での盗聴防止のため，鍵は暗号化されている必要がある．この鍵を安全に配る為に必要な鍵を KEK（Key Encryption Key）と呼ぶ．マルチキャストで鍵を配る場合，鍵を受け取る事ができるｎ人に対して事前に共通の鍵，KEK が配られている必要がある．この KEK の配布には様々な手法があるが，通常，RSA [12] や Diffie-Hellman 交換 (DH) アルゴリズム [13] などといった公開鍵暗号方式 [14] を用いた暗号通信が利用される．公開鍵暗号方式は，異なる鍵である公開鍵と秘密鍵を用いて暗号通信を行う．公開鍵で暗号されたデータは秘密鍵でのみ復号化できる．公開鍵からは秘密鍵を作成できない．よって，KEK を受け取りたいクライアントは鍵管理サーバに公開鍵を送信し，サーバは認証を行った後に KEK を公開鍵で暗号化して送信する．暗号化された KEK が盗まれても，秘密鍵を持たないため，悪意あるユーザは KEK を復号化できない．このように，KEK を共有し，その後は KEK で暗号化した鍵をマルチキャスト通信でやり取りする．

2.2　鍵の更新

鍵の更新とは，鍵の値を更新し，クライアントに安全な方法で配ることである．

2.2.1 更新の必要性

鍵は使い続けることによって様々な問題が発生する．例えば，鍵の脆弱性の問題がある．鍵の値となりうる全ての文字列の組み合わせを試していくブルートフォース・アタック [15] やハッシュ値に変換された鍵の値を解析するレインボー攻撃 [16] などの攻撃方法があるように，時間をかければ鍵の値は解析できる可能性がある．短時間の通信であれば，こういった時間のかかる攻撃は意味をなさない．しかし，本研究で想定するような，長時間の暗号通信を必要とする pay-per-view 型リアルタイムストリーミングでは，脆弱性回避のためにも鍵の更新が必要となる．

この他にも，pay-per-view 型リアルタイムストリーミングを想定すると，視聴権限でユーザを区別する必要があり，鍵の値を更新して，視聴権限の無くなったユーザがストリーミングを視聴できないようにしなければならない．

2.2.2 鍵の更新手法

鍵の更新手法は大きく二つに分けることができる．あるイベント，例えばユーザがグループに参加，離脱するなどのイベントが発生した際に鍵を更新するイベント単位と，1 分，1 時間といった定められた時間が経過すると鍵を更新する時間単位に分けられる．それぞれの手法にメリットとデメリットがある．

イベント単位のメリットは，少人数の暗号通信などに向いている点である．例えば暗号通信を用いたチャットなどがそれに当てはまる．暗号通信を用いたチャットを行う場合，時間単位で鍵を更新すると，定めた時間まで鍵を更新しないのでグループから離脱したユーザにまで，その後の暗号通信を視聴できてしまう．しかし，ユーザの参加や離脱といったイベントで鍵を更新することでその問題を解決することができる．ユーザが離脱した時点で鍵の値を更新し，離脱したユーザ以外のメンバーが共有する．これによって，離脱したユーザにチャット内容が漏れる事はなくなる．

デメリットとしては，鍵の更新頻度を予想することが困難になる点が挙げられる．例えばユーザ数が数百人単位を超える場合，参加や離脱といったイベントが短時間に集中してしまう可能性がある．pay-per-view 型リアルタイムセキャストリーミングを考えると，人気コンテンツの開始時にユーザの参加イベントが多発し，その終わりに伴いユーザの離脱イベントが続発する可能性がある．しかし，ほぼ同時に離脱，参加を行ったユーザに対してまで鍵の値を更新する必要はなく，まとめて一つのイベントとして更新させる方がサーバの負担が少なくて済む．必要のない鍵の更新でサーバの負担を増やしてしまう危険性がある点がイベント単位のデメリットである．イベント単位の概要図を 2.1 に示す．

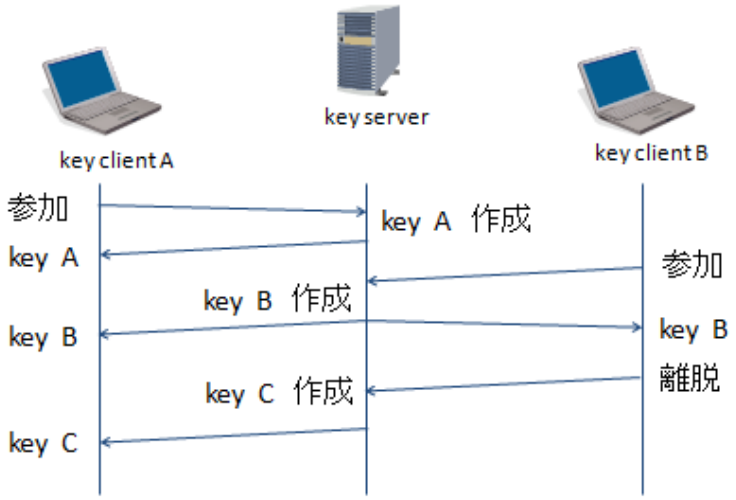

図 2.1: イベント単位の鍵更新概要図

次に, 時間単位での鍵更新について述べる．メリットとしては，鍵の更新頻度を予想しやすい点が挙げられる．1 分、1 時間といった定めた時間ごとに鍵を更新するので，イベント単位のようにユーザの参加や離脱といった鍵の更新イベントが重なり，サーバに負担がかかるといったことが発生しない．また，一定時間で鍵を更新するので鍵の更新頻度を予想しやすく，サーバの負担を想定しやすい．

しかし，鍵の更新に融通が利かないというデメリットがある．イベント単位でも述べたように，暗号通信を用いたチャットなどを想定するとき，離脱したユーザに秘密のチャット内容が漏えいしない様に，ユーザが離脱した時点で鍵の値は更新されければならない．だが，時間単位の場合，離脱が起こっても一定時間が経過するまでは鍵の更新を行わないので，チャット内容が漏えいしてしまう．また，離脱や参加が起こっていない状況でも，一定時間が経過すると鍵の値を更新してしまうので，サーバに必要のない負担を強いてしまう．

今回想定される環境は，少なくとも数百人を超える“ 多数のユーザ ”を想定している．また，pay-per-view 型リアルタイムセキュアストリーミングを想定している為，ユーザが離脱しても視聴権限のあるコンテンツは視聴できなければならない．22 時から 23 時のドラマコンテンツを購入したユーザは，一度，チャンネルを閉じ，視聴を止めたとしても，再度視聴できなければならない．イベント単位の鍵更新では，視聴を止めた時点で鍵が更新されてしまうので視聴権限の残っているユーザが視聴できなくなってしまう．これらのことから,「多数のコンテンツ」と「多数のユーザ」が想定される環境での pay-per-view 型リアルタイムセキュアストリーミングの鍵更新には時間単位の鍵更新が向いている．時

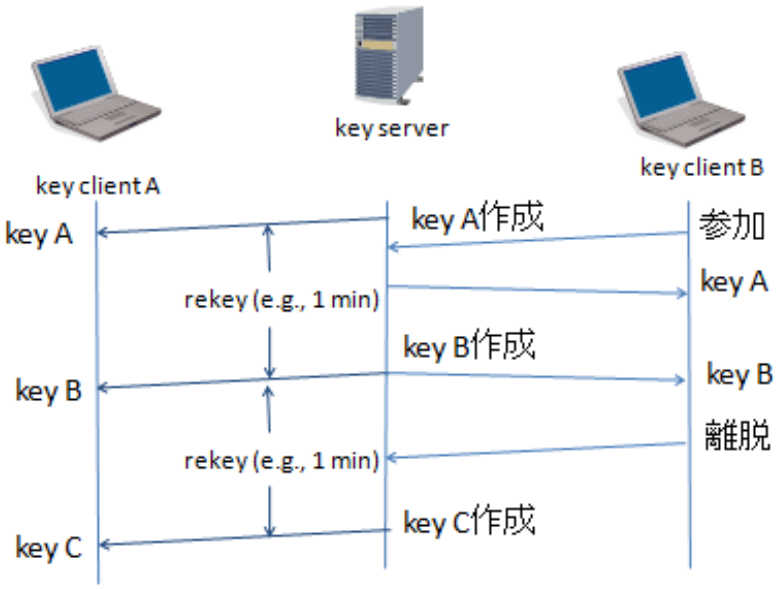

図 2.2: 時間単位の鍵更新概要図

間手法の概要図を 2.2 に示す.

2.2.3 鍵の更新プロトコル

LKH(Logical Key Hierarchy)[17][18][19][20] は鍵の値を更新し，マルチキャスト通信で安全に配布する機能と，ユーザの管理機能を両立させた時間単位での鍵更新プロトコルである．ツリー構造を用いて，複数の鍵を関連付けさせて配布，各クライアントが複数の鍵を持つことにより，マルチキャスト通信で安全に鍵の配布ができる．LKH の概要図を図 2.3 に示す.

U1 は LKH サーバに自分だけが持つ鍵 K1 を送信する．K1 は公開鍵や DH 鍵暗号方式で作られた鍵などが想定されている．LKH サーバは K1 を複数の Sub key から成るツリーに組み込み，K1 と関連付けられている sub key(K9，K13，KEK) を K1 によって暗号化して送信する．これにより U1 は LKH サーバ，他のクライアントと複数の鍵を共有することができる．U1 がこのツリーを離脱するとき，K1 と関連付けられていた sub key（K9，K13，KEK）はすべて更新され，U2 には K2 で暗号化された（K9 ’,K13 ’,KEK ’）が，U3 と U4 には K10 で暗号化された（K13 ’,KEK ’) が，U5，U6，U7，U8 には K14 で暗号化された（KEK ’）がマルチキャストで配られる．離脱する U1 が持っていない鍵で暗号化する事により，U1 には更新された鍵は配布されない．マルチキャスト通信で鍵を配布するので，ユニキャスト通信で鍵を配布するよりもサーバの負担が少ない．ユニキャス

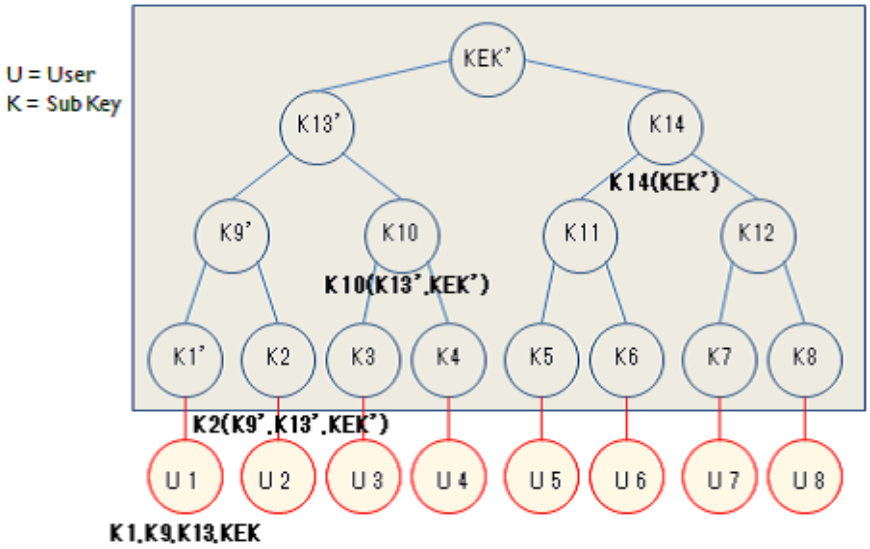

図 2.3: LKH 概要図

ト通信で鍵を配付する場合は，各ユーザに鍵を配布するので回 7 の通信が必要になる．しかし，マルチキャスト通信で鍵を配付する場合は，3 回の通信で済む．ただし，LKH はイベント単位の鍵更新方式である．鍵更新のタイミングが参加や離脱のため，頻繁にユーザが参加や離脱が繰り返される環境では，サーバが行う鍵更新の負担が大きくなる可能性がある．本研究の環境では,「多数のユーザ」を想定している．参加や離脱といったイベント単位の鍵更新では，更新が多発すると予想される．しかし，ユーザを強制的に離脱させることのできる機能やマルチキャスト通信で実現されるスケーラブルな鍵の更新機能は本研究で参考になる機能である．

OFT(The One-way Function Tree)[21] は，LKH と同様にマルチキャスト通信で安全に鍵を配布する機能と，ユーザの管理機能を両立させたイベント単位の鍵更新プロトコルである．ツリー構造を用いてる点で同様だが，鍵を関連付ける方法が異なる．LKH はサーバが複数の sub key を作った後に論理的に鍵を関連付けさせる．具体的には，list 構造を用いて鍵を関連付けさせる．root 鍵から下位に鍵を関連付けさせていくことによって，sub key 同士が直接的には依存し合わないツリーを作る事ができる．しかし OFT は，最下層を除く sub key を作る際に，下位に位置する鍵を用いて作られる．OFT の概要図を図 2.4 に示す．

OFT は LKH とは異なり，鍵の作成に二つの関数が重要になってくる．一つが一方向関数 (g())，もう一つが合成関数 (f()) である．一方向関数は，A から B を出すのは簡単だが，B から A を導き出すのが艱難な性質をもった関数である．もう一方の合成関数は A と B を特殊な処理を経て合成し，C を生み出す関数である．K1 と関連付けされている sub key(K9，K13，KEK) の作成手順を述べる．まず，サーバでは最下層の鍵 (K1，K2，

K3… K8) を作る．次に，最下層の鍵から一方向関数を使用して blinded node key (g(K1), g(K2)，g(K3)… g(K8)) を作成する．サーバは g(K 1) と g(K2) を合成関数に通して K9 を作り出す.
(K9 = f(g(k1) ，g(K2)))

同様にして，K9 と同階層に存在する K10，K11，K12 を作成し，これらを作成したのと同じように K13，K14，そして KEK を作り出す.
(K13 = f(g(k9)，g(K10)))，(KEK = f(g(k13) ，g(K14)))

OFT はイベント単位の鍵更新プロトコルである．ユーザの参加や離脱で鍵の値を更新する．U1 にユーザが参加した場合，ユーザには K1，g(K10)，g(K14) が配られる．U1 はこれらの鍵から一方向関数と合成関数を用いて K9，K13，KEK を得る.

U1 が離脱するとき，U1 が持っている鍵 (K1，K9，K13，KEK) は更新される必要がある．g(K10)，g(K14) は K9，K13，KEK が更新されるので，更新する必要はない．U2 から U8 は更新された鍵 (K1’，K9’，K13’，KEK’) を持つ必要がある．更新された鍵は，K1’ 以外，下位の鍵に依存する．(K9’ = f(g(k1’)，g (K2)))，(K13’ = f(g(k9’)，g(K10)))，(KEK’ = f(g(k1 3’)，g(K14))) となる．U2 には K2 で暗号化された g(K1’) が配られる．U2 は g(K1’) と自身が持つ g(K2) から K9’ を作成する．また，g(K9’) と g(K10) から K13’，g(K13’) と g(K14) から KEK’ を作成する．U3，U4 は K10 で暗号化された g(K9’) を受け取り，同様にして K13’，KEK’ を得て，U5 から U8 は K14 で暗号化された g(K13’) から KEK’ を得る.

LKH は鍵が更新される際，更新された鍵を複数送る必要がある。しかし，OFT を用いると blinded node key をひとつ送るだけでよいため，ツリーの階層が深ければ深いほど LKH よりも送信するデータ量が少なくて済む．しかし，ツリーの階層が浅い場合，送るデータ量はほとんど変わらない．むしろ鍵の作成が複雑な分，サーバの処理が増える.

2.3 鍵の管理プロトコル

2.3.1 GSAKMP

GSAKMP[22] はグループメンバーとグループ管理サーバが複数の SA（Security Association），GSA（Group SA）を用いて暗号通信をすることが特徴的な鍵管理プロトコルである．GSA は，レジストレーション SA と Rekey SA，データ SA からなる．レジストレーション SA は，メンバーがグループの参加や離脱をするときに使用する SA で，ユニキャスト通信で KEK や TEK（Traffic-Encrypting Key）を配信する．Rekey SA はサーバがグループメンバーにマルチキャストで更新された鍵を配るときに使う SA である．データ SA はグループメンバー同士がデータのやり取りをする SA である．鍵の更新手法には LKH が用いられる.

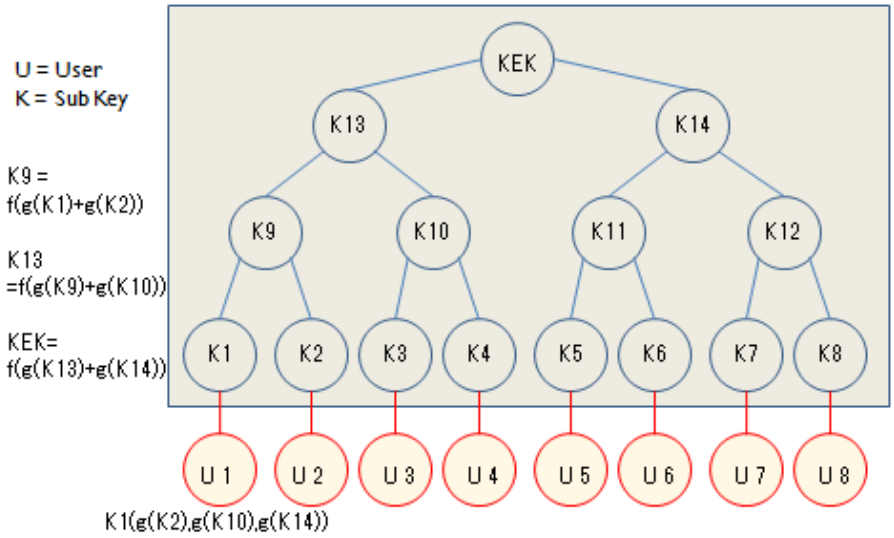

図 2.4: OFT 概要図

2.3.2 GDOI

GDOI（The Group Domain of Interpretation）[23] は，IKE（The Internet Key Exchange）を拡張したプロトコルである．IKE は IPsec（Security Architecture for Internet Protocol）で用いられるプロトコル群のうち，インターネット鍵交換で使用されているプロトコルである．IKE はマルチキャストに対応しておらず，グループ管理機能を持っていないが，GDOI は IKE をマルチキャストに対応させ，グループでのインターネット鍵交換を実現させたプロトコルである．

グループメンバーとサーバは IKE によって SA を作成する．グループメンバーはグループ情報のリクエストをサーバに送り，サーバからポリシーを得る．その後，鍵のリクエストをし，グループが使っている KEK と TEK を得る．その後は KEK と TEK で SA を張り，TEK を用いた SA ではグループメンバー内での暗号通信をし，KEK で張った SA ではサーバが送ってくる更新された鍵，KEK' や TEK' を受信する．サーバは複数の鍵を管理し，鍵の更新には LKH や OFT などを使用することができる．

2.3.3 MIKEY

MIKEY(Multimedia Internet KEYing)[24] は様々な暗号方式で KEK を配信できる鍵管理プロトコルであり，暗号通信方式に依存しない Group Key 管理プロトコルである．マルチキャスト通信，ユニキャスト通信での鍵の配布ができる．

MIKEY で KEK を配布する方法は共通鍵暗号方式と公開鍵暗号方式，DH 鍵暗号方式の 3 つがあり，それらから選択して KEK をセキュアに配信する．サーバとクライアントは配布された KEK を用いて seed の送受信を行う．図 2.5 に seed の配布方法を示す．クライアントがサーバに seed の要求をする．クライアントはサーバに seed を配付する（ユニキャストまたは，マルチキャストでの配布が可能）．seed はクライアント同士が暗号通信路である CS （Crypto Session）を張るために用いられる鍵（TEK）を作成するものである．クライアントは seed から各自が持つ GID（Group ID）を使い，TEK を作成し，同じ TEK を持つ者同士で安全な通信をする．GID によって異なる TEK が作成されるので GID 毎にグループを作成する事ができる．（TEK A ＝ seed ＋ GID A，TEK B ＝ seed ＋ GID B）

鍵の更新は，seed を更新することによって各クライアントに配布する．seed’ が配布されると各ユーザは各自が持つ GID を用いて，更新された TEK’ を作成する．サーバが各グループごとの TEK を管理する必要がないため，ほかのプロトコルよりもサーバの負担が少なくて済む．GSAKMP や GDOI では，コンテンツの数だけ鍵を作成し，更新しなければならないが，MIKEY サーバはその負担を負う必要がない．

”多数のユーザ ”と ”多数のコンテンツ ”を想定すると，サーバの負担が少ないグループ管理を実現している MIKEY が他の鍵管理プロトコルよりも参考となる．

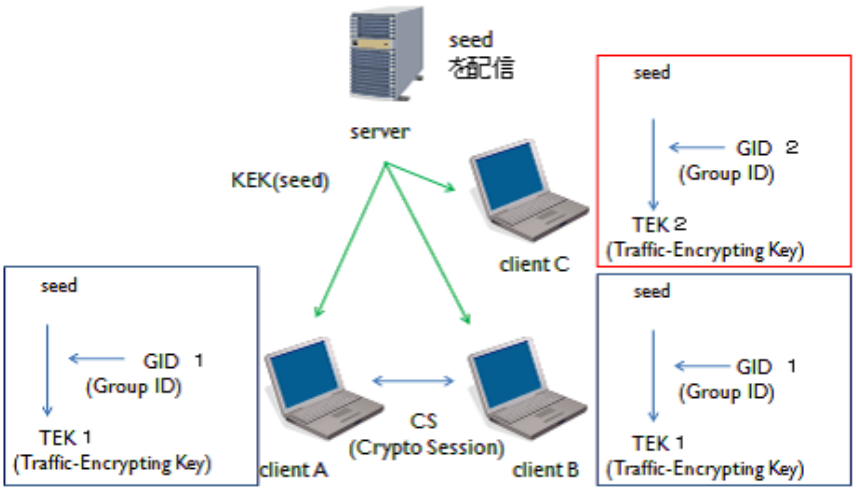

図 2.5: MIKEY 概要図

第3章　関連研究と問題点

本章では既存の技術や過去に提案された手法から本研究に関連の深いものを述べ, その問題点を明らかにする.

3.1　MIKEY

前述した鍵の管理プロトコルは，GSAKMP や GDOI，MIKEY などマルチキャストが使えるものだけでも数多く存在する．本研究では，少なくとも数十以上の「多数のコンテンツ」を有する環境を想定するため，コンテンツ数が増えてもサーバの負担が変わらない MIKEY を選択し，その問題点を挙げていく．

MIKEY サーバは，KEK で暗号化された通信によって seed を配信する．このとき，ユニキャスト通信とマルチキャスト通信，どちらの通信方法も利用できる．各クライアントは受け取った seed と，それぞれが持つ GID で TEK を作成する．同じ GID を持つクライアントは，同じ TEK を共有する事により，グループを形成する．他のプロトコルが，グループごとに鍵を作成しサーバが管理するのに対し，MIKEY はサーバが一つの seed を配信するだけでよいため，コンテンツ数の増減にかかわらず，負担が一定である．しかし，「多数のコンテンツ」と「多数のユーザ」が想定される環境での pay-per-view 型リアルタイムセキュアストリーミングを想定すると，seed の更新手法と seed の暗号通信に問題が発生する．seed の更新問題については図 3.1 に，seed の暗号通信にかかわる問題は図 3.2 に示す.

seed の更新は，ユーザの更新リクエストを受信したときに行う．更新リクエストにはマルチキャストアドレスなどが含まれ，サーバはそのアドレスに更新した seed' を KEK で暗号化して送る．seed' が送られるマルチキャストアドレスを知っているユーザのみが更新した seed' を受け取る事ができる．また，そのとき KEK で暗号化しているため，KEK を持つユーザ以外，seed' を受信する事は出来ない．図 3.1 にある MIKEY クライアントグループ A のユーザは，鍵を更新するとき，MIKEY サーバに更新リクエスト（rekey message）を送信する．rekey message にはマルチキャストアドレス（ex，239.7.7.7）が含まれる．サーバは認証を行い，その後，seed' を KEK で暗号化し 239.7.7.7 に送信する．MIKEY クライアントグループ A は 239.7.7.7 に join しているため，暗号化された seed' を受け取る事ができる．暗号化された seed' を復号化し，seed' と GID を使って TEK' を作成する．グループ B も同様にして seed" を受け取る．つまり，MIKEY はユーザからの鍵更新リクエストを受信したときのみ鍵を更新するイベント単位の鍵更新プロトコルである．しかし，

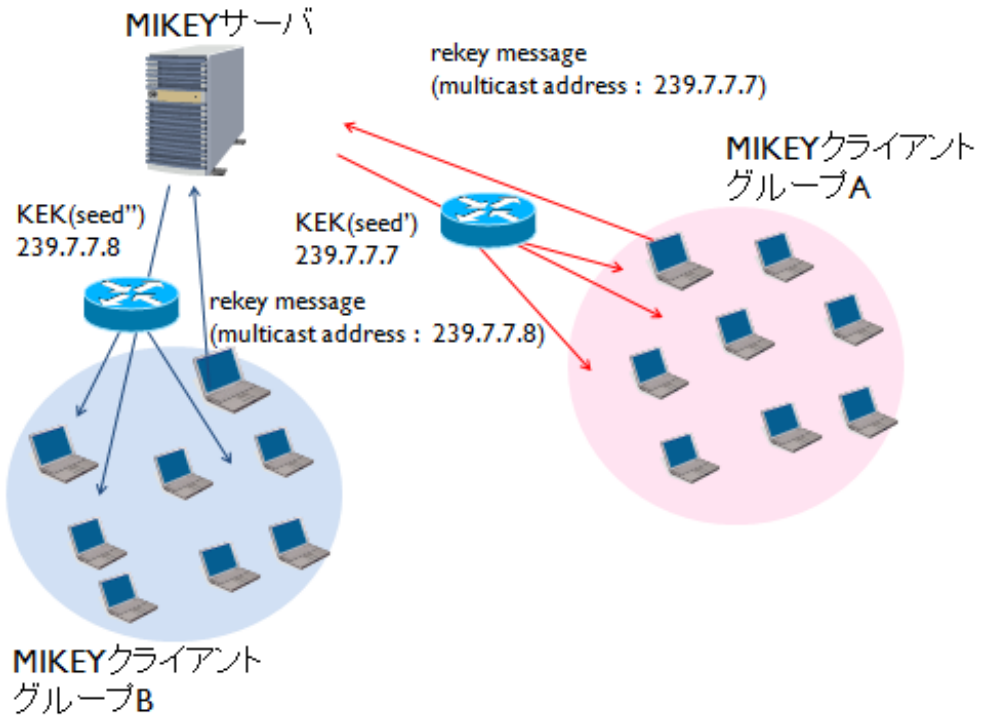

図 3.1: seed 更新に関する問題点

「多数のユーザ」を想定した環境では，イベント単位の鍵更新ではなく，時間単位を用いる必要がある．

seed を安全に配る暗号通信には KEK が用いられる．KEK は，事前に公開鍵暗号方式や DH 鍵暗号方式により各クライアントに配信され，グループ内での暗号通信が終了するまで使用される．MIKEY ではｎ対ｎ，もしくは１対ｎ通信が想定されている．また，短時間のセキュアストリーミンググループを複数管理することが前提とされる鍵管理プロトコルであるため，長時間使い続ける必要はない．よって，KEK の更新については定義されていない．これでは，一度参加したユーザはグループ内の暗号通信をいつまでも視聴する事ができる.「多数のコンテンツ」と「多数のユーザ」が想定される環境での pay-per-view 型リアルタイムセキュアストリーミングでは，一度参加したユーザが永遠にコンテンツを視聴できてはいけない．例えば，3 時間のドラマを放送するとき，30 分の視聴権限を購入したユーザには 30 分以上コンテンツが視聴できてはいけない．また，悪意ある第三者がユーザの KEK を盗んだ場合，悪意ある第三者はいつまでも seed を受け続ける事ができてしまう．もちろん，GID がなければ TEK は作成できないが，GID も盗難されたとき，悪意ある第三者がコンテンツを盗聴できてしまう．これらの問題を回避する為にも KEK の値を定期的に更新し，盗難に対応する必要がある．

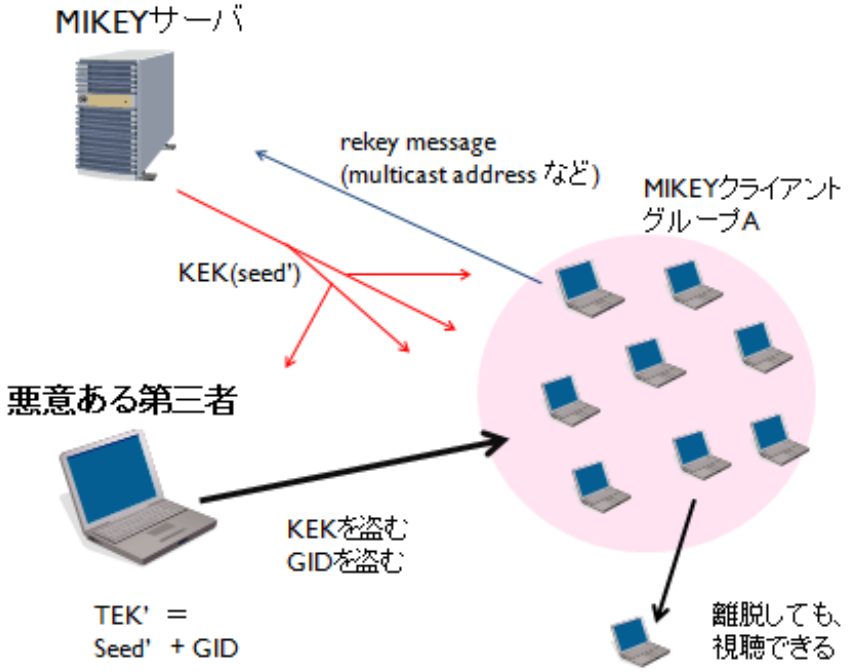

図 3.2: KEK 更新に関する問題点

3.2 An Architecture and Key Management Approach for Maintaining Privacy in Location Based Group Services

本論文 [25] では，MIKEY の KEK を更新する為に LKH を用いたシステムが提案されている．図 3.3 に概要図を示す．LKH は，複数の鍵を共有する事により，更新された鍵をスケーラブルに配信する事ができる．ユーザが離脱する際，サーバはユーザが持っていた KEK と複数の sub key を更新し，まだ参加しているユーザに配信する．更新した KEK’, sub key’ を配信するとき，サーバは参加中のユーザのみが持つ sub key で暗号化して送る．マルチキャストで送るため，同じ sub key を持つユーザには一度に更新された鍵を送る事ができる．これにより，各クライアントにユニキャスト通信で配信するよりもスケーラブルに KEK’ や sub key’ を配信する事ができる．また，MIKEY のグループからユーザが離脱する際，LKH の更新手法を用いて残っているユーザだけがコンテンツを受信できるようにしている．

しかし，本研究で想定している「多数のコンテンツ」と「多数のユーザ」がいる環境では，参加や離脱が頻繁に発生するため，ユーザ数に影響を受けにくい時間単位の鍵更新が

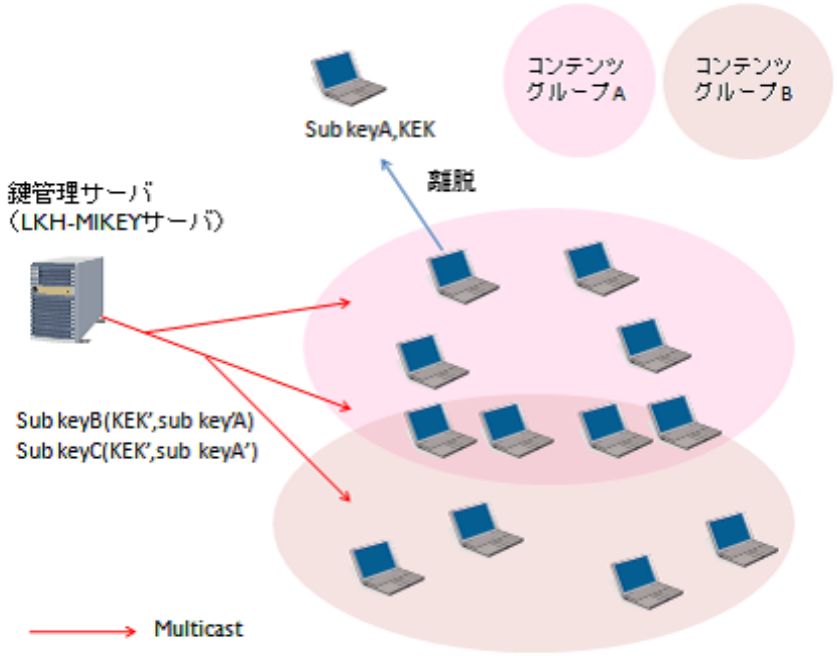

図 3.3: LKH-MIKEY 概要図

必要となる．また，pay-per-view 型のリアルタイムセキュアストリーミングを想定しているため，ユーザは所定の時間区分だけストリーミングを視聴できなければならない．離脱などのイベントで鍵を更新すれば，視聴権限がまだ残っているにもかかわらず，離脱した瞬間にコンテンツを視聴できなくなってしまう．つまり，3 時間分のコンテンツを購入したユーザは，一度視聴を止めても，再び視聴を始めたときにコンテンツが視聴できなければならならない．LKH を用いた KEK 更新手法はユーザの排除やマルチキャスト通信によるスケーラビリティ性などのメリットも多く存在するため，本研究では LKH を応用した時間による鍵更新手法を提案し，実現する．

第4章　アプローチ

本章では，本研究で想定する ”多数のコンテンツ ”と ”多数のユーザ ”といった環境下において発生する鍵更新に関する問題点へのアプローチについて述べる．

4.1　機能要件

本研究が想定する効率的なユーザ管理と鍵更新時のユーザに対する鍵配信にかかる計算処理コストを低減する鍵管理システムでは，以下の 3 点を満たすべき機能とする．

1. IP マルチキャストを用いた鍵更新
2. 短時間の鍵更新
3. 長時間の鍵更新

4.1.1　IP マルチキャストを用いた鍵更新

更新した鍵を配布する方法は，ユニキャスト通信で各ユーザに配信する方法と，マルチキャスト通信で同時に複数のユーザへ配信する方法がある．”多数のユーザ ”が想定される場合，各ユーザに鍵を配信するユニキャスト通信ではサーバの負担が増えてしまう．しかし，マルチキャストで鍵を配信する場合，一回の通信で複数のユーザに鍵を配信できるのでサーバの負担は増加しない．よって想定環境には，鍵更新にマルチキャストを用いる鍵管理システムが適している．

マルチキャストを用いる鍵管理システムは GSAKMP や GDOI，MIKEY などがある．サーバが配信した seed から各ユーザが鍵を作成する MIKEY は，コンテンツ数の増加がサーバの負担にならない．GSAKMP などのシステムは，コンテンツごとに鍵を作成し，配布するため，コンテンツ数の増加とともにサーバの負担が増える.「多数のコンテンツ」に対応したシステムを考えると MIKEY がもっとも適しているため,「多数のコンテンツ」と「多数のユーザ」を想定した鍵管理システムの構築は MIKEY をベースに進めていく．

しかし，MIKEY にはいくつか問題がある．seed の更新は，クライアントからの更新リ

クエストを受信したときに行うため，”多数のユーザ ”が想定される場合，更新リクエストが頻繁に起こる可能性がある．このため，更新頻度を予想しやすい時間単位の鍵更新に改善する必要がある．また，seed を暗号化してクライアントに送るときに使用する KEK の更新方法は，MIKEY が短時間の暗号通信を想定している為，定義されていない．暗号通信を短時間しか利用しない場合，ブルートフォース・アタックなどによって KEK が盗まれる可能性は少ない．だが，今回想定されるような暗号通信を長時間利用する環境下では，KEK が盗まれる可能性が高くなる．よって，KEK の更新手法が必要となる．

4.1.2 短時間の鍵更新

鍵は盗難される可能性がある．ブルートフォース・アタックやクラッキングなどにより，TEK が盗まれ，視聴権限を持たないユーザが視聴できてしまう可能性がある．よって，TEK が盗まれた場合の対策を講じなければならない．

TEK が盗まれると，TEK で暗号化したコンテンツ，全てが視聴できてしまう．よって，TEK が盗まれた際のリスクを下げる為には，TEK を更新し，TEK’ によってコンテンツを暗号化しなければならない．しかし，30 分のコンテンツがあった場合，29 分経過した時点で TEK を更新するのでは意味がない．30 分間に複数回，TEK を更新し，鍵が盗まれた際のリスクを下げる必要がある．このとき，更新にかかるサーバの負担があまりにも増えすぎないように，注意しなければならない．コンテンツの安全性とサーバの負担を見極めた TEK の更新頻度を考える必要がある．

MIKEY は，リクエストがあった時にのみ，seed を更新することにより，TEK を更新する．seed の値が変わればクライントで作られる TEK の値も変わる．(GID は更新されないが，長時間使うときは GID も更新する必要がある．)

しかし，リクエストを受け付けたときのみ TEK を更新するイベント単位の鍵更新では、鍵の更新頻度を予測する事が難しい．ユーザによってはある集中した時間内にリクエストを何度も行う可能性があるため，鍵の更新頻度を予想しやすい時間単位の鍵更新に改善する必要がある．30 分のコンテンツがあるとして，1 分間毎に TEK を更新すると決めれば，ユーザがどれだけ増加しても，サーバの負担は変わらない．また，TEK が盗まれたとしてても，1 分間のコンテンツしか盗聴することができず，その前後のコンテンツは保護される．鍵の更新によって前後のコンテンツが保護されている事を Forward Secrecy と Backward Secrecy という．図 4.1 に詳しく示す．コンテンツの Forward Secrecy と Backward Secrecy を保護する為に行われるコンテンツ時間内（短時間）での鍵更新を Short Period Rekey という．

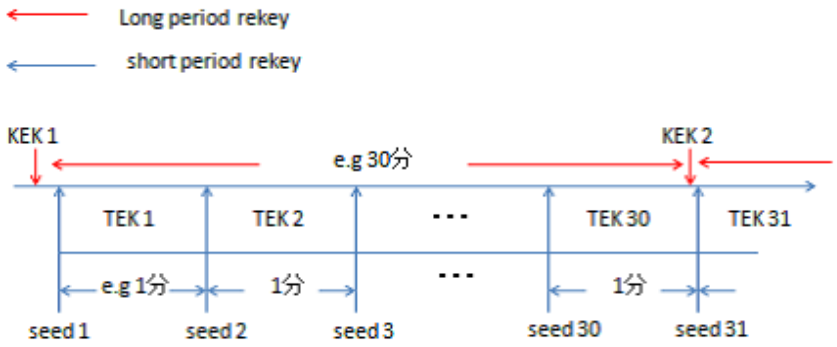

図 4.1: ２種類の鍵更新概要図

4.1.3 長時間の鍵更新

pay-per-view 型セキュアストリーミングにおいて守られるべきはストリーミングコンテンツである．料金を支払ったユーザのみがコンテンツを視聴できなければならない．料金を支払ったユーザであっても，料金以上のコンテンツが視聴できてはならない．

図 4.1 に示した通り，18 時から 18 時 30 分に流れるコンテンツを買ったユーザは，18 時以前のコンテンツを視聴できてはならずまた，18 時以降のコンテンツも視聴できてはいけない．よって，一定時間ごとに鍵の値が更新され，視聴権限を持つユーザにのみ配信される必要がある．

MIKEY の鍵更新では視聴権限によるユーザの区別ができない．鍵更新時，MIKEY サーバは，マルチキャスト通信で seed’ を配信するだけであるため，視聴権限があるユーザとないユーザを区別する事ができない．一度，KEK が配られたユーザは，永遠にストリーミングコンテンツを視聴できてしまう．よって，KEK を視聴時間ごとに更新し，権限のあるユーザにのみ配布する鍵更新手法が必要である．

ユーザを区別する鍵の更新手法には，OFT や LKH などがある．いずれも，ツリー構造を用いた鍵更新手法である．ツリーに関連付けられた複数の sub key をユーザに持たせ，グループを抜けるユーザが持たない sub key を使って KEK’ を配信する．このとき，多くのユーザが持つ sub key で暗号化し，マルチキャスト通信で配信する事で，各ユーザに１対１で KEK’ を配信するよりもサーバの負担は少なくて済む．

LKH は KEK が更新される際，更新された複数の sub key’ を送信する必要がある．しかし，OFT を用いると sub key をひとつ送信するだけで，ユーザが複数の sub key を作る．よって，ツリーの階層が深ければ深いほど LKH よりも送信するデータ量が少なくて済む．しかし，ツリーの階層が浅い場合，送信するデータ量はほとんど変わらない．鍵の作成が複雑な分，サーバの処理が増加する．

LKH，OFT ともにイベント単位の鍵更新プロトコルである．しかし，今回の想定環境ではイベント単位よりも時間単位の鍵更新が適している．よって，時間単位での鍵更新に改善していく必要がある．

時間単位の鍵更新に改善する場合，OFT よりも LKH が適している．30 分，60 分などの視聴権限でユーザをグルーピングする為，イベント単位のようにユーザの数だけ階層を増やさなくてもよい．それどころか，階層が多いほど鍵の数が増え，それだけ暗号通信路を作らなければならない為，サーバの処理が増える．時間単位を用いる場合は，ツリーの階層は浅い方がよい．階層が浅い場合，LKH の方が OFT よりも適している為，本研究では LKH を改善していく．

第5章　設計

本章では，本研究で提案するシステムの設計について述べる．概要図を図 5.1 に示す．また，本システムの動作を，Time Driven LKH，Time Driven MIKEY，受信ノードに分けて述べる．

5.1　Time Driven LKH

本節では Time Driven LKH の動作概要について述べていく．LKH の機能を参考に時間単位での鍵更新機能を新たに加えた LKH を Time Driven LKH と呼ぶ．図 5.2 に Time Driven LKH の動作モジュール図を示す．

5.1.1　鍵配信モジュール

鍵配信モジュールを，図 5.2 に示す．

Time Driven LKH サーバ（以下，LKH サーバ）は設定された時間をもとにツリーを作成する．ツリーとは複数のノードをリスト構造によって関連付けしたものである．LKH サーバはこのツリーを二本作成する．ツリーを二本作成する理由は， 5.1.2 項で述べる．

ツリーを構成するノードには様々な情報が含まれている．それぞれのノードが鍵番号（key ID）や鍵（sub key）といった値を持っている．LKH サーバと Time Driven LKH クライアント（以下，LKH クライアント）は，この key ID と sub key を用いて LKH サーバと SA を張る．この SA は，GSAKMP などで定義されている Rekey SA に相当する．詳しくは 5.1.3 項で述べる．

ツリーの末端に位置するノードを葉ノードと呼び，もっとも上位に存在するノードをルートノードと呼ぶ．葉ノードからルートノードまでの一連の繋がりを枝と呼ぶ．枝には，そこに属する各ノードの key ID や sub key の情報が含まれている．LKH サーバは，この枝に 30 分，60 分といった時間情報を加え，管理する．時間が過ぎた枝は，枝に含まれている各ノードの sub key を更新し，新たな時間情報が加えられる．ツリーの概要図を図 5.3 に示す．

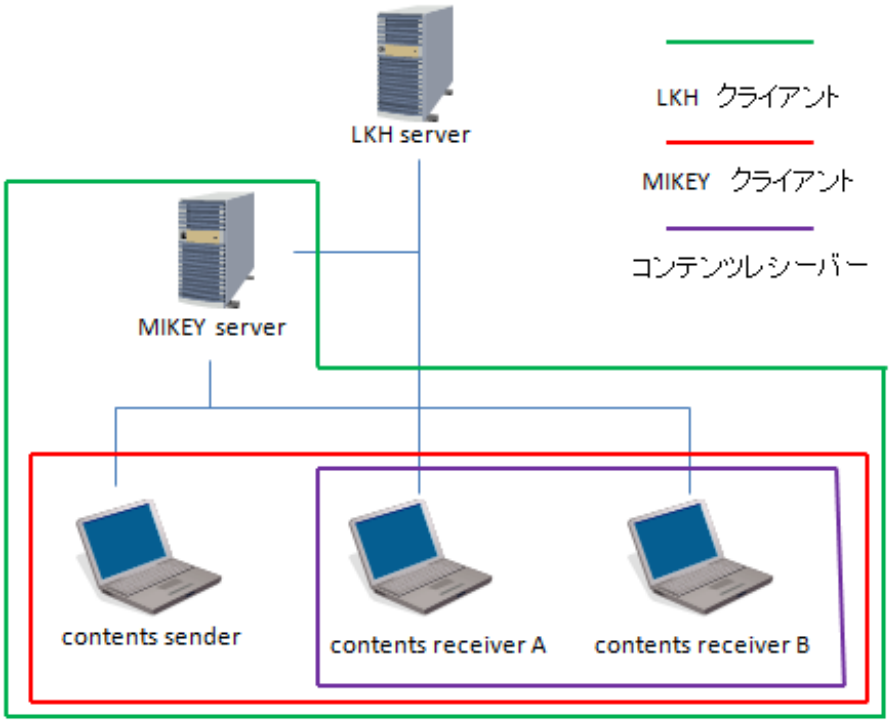

図 5.1: 全体動作概要図

LKH クライアントは LKH サーバに公開鍵と視聴権利情報が含まれるリクエストメッセージを送信する．この視聴権利情報とは，視聴可能な時間ではなく，LKH サーバが作成している時間情報を指す．つまり，10 時から 10 時 30 分まで放送されるドラマコンテンツがある場合，LKH クライアントが 10 時 15 分に視聴を開始すると，10 時 45 分まで視聴できるのではなく，10 時 30 分のドラマ終了までコンテンツを視聴できる事になる．概要図を図 5.4 に示す．通常，視聴可能な時間で課金はできない．しかし， 5.1.2 項で述べる insert 機能を応用する事で，視聴時間による課金も可能となる．

公開鍵が正当なクライアントからの公開鍵かどうかを調べるために，LKH サーバはパスワードによる認証を行う．リクエストメッセージを受け取った LKH サーバは，予め作成していたツリーから，視聴権利に見合った枝を選択し，KEK と共に公開鍵で暗号化して L クライアントに送信する．L クライアントは受信した暗号データを秘密鍵で復号化し，枝に含まれる key ID や sub key，そして KEK を得る．LKH クライアントは得た key ID と sub key から L サーバとの Rekey SA を張る．この Rekey SA は得られた sub key の数だけ張る．KEK は L クライアント同士で暗号通信を行う為に用いられる．この KEK で張られる SA も GSAKMP などで定義される Rekey SA に相当する．sub key でも Rekey SA が張られるのだが張られるのだが，KEK で張られる Rekey SA との違いは後述する．

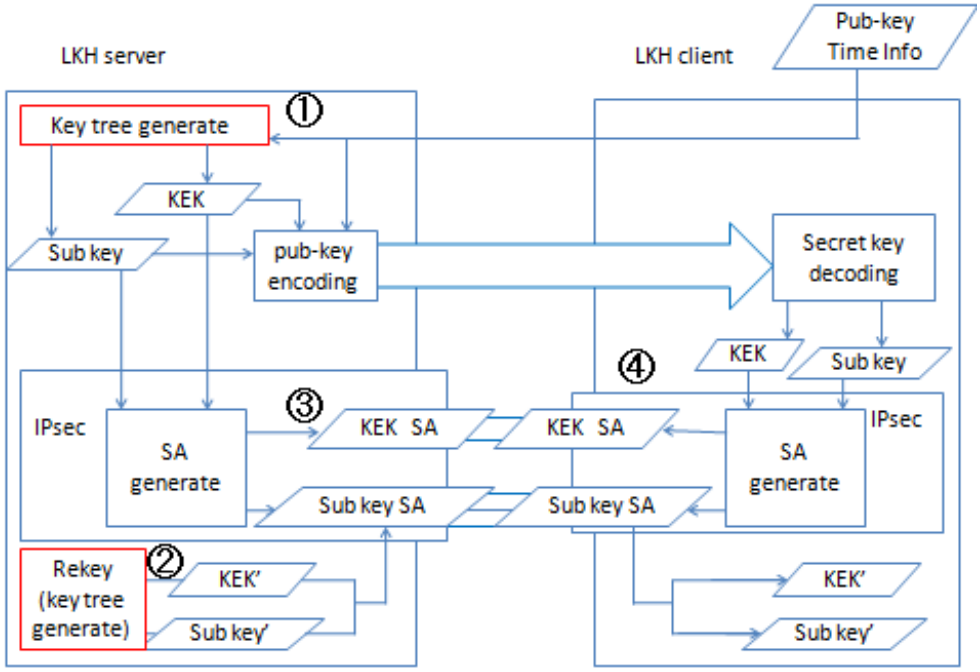

図 5.2: LKH モジュール図

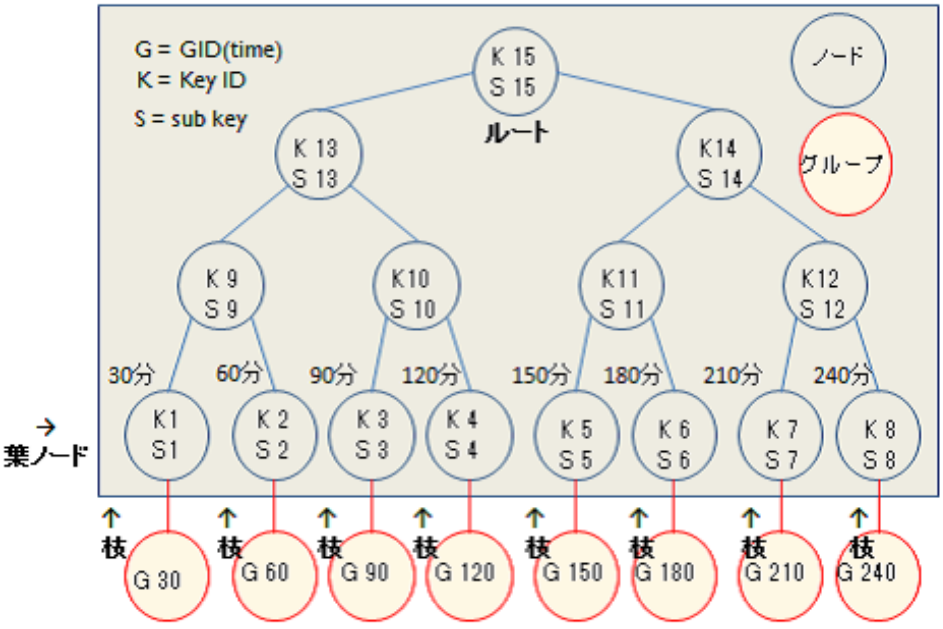

図 5.3: Time Driven LKH ツリー概要図

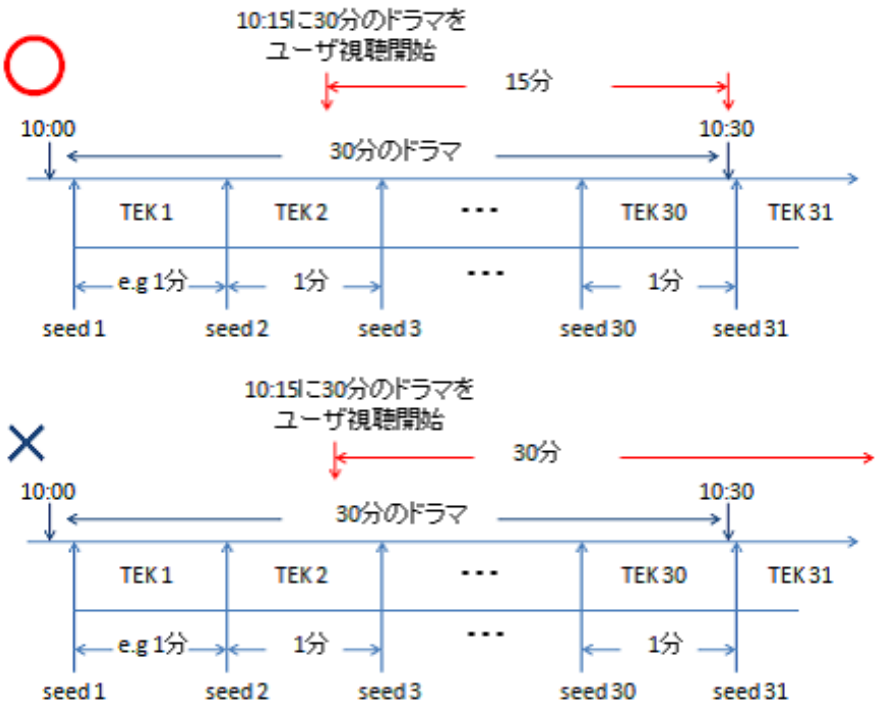

図 5.4: 視聴権利情報概要図

5.1.2 鍵更新モジュール

鍵更新モジュールを，図 5.2 に示す．

Time Driven LKH の鍵更新は時間単位によって行われる．LKH サーバが定めた時間を過ぎると，枝に存在する各 LKH ノードの sub key は更新される．30 分ごとの鍵更新を設定すると，LKH サーバは 30 分，60 分，90 分と枝の sub key を更新していく．このとき，KEK も更新される．この更新された鍵を，それぞれ sub key’，KEK’ と呼ぶ．そして，新しく設定された時間を EndTime とする．

sub key’ と KEK’，EndTime は sub key で張られた SA を通じて LKH クライアントに送信される．図 5.2 に概要図を示す．30 分経過すると，sub key’ と KEK’，EndTime は 30 分の視聴権限ユーザが張っていない SA を通じて配信される．つまり，K14，K10，K2 で張られた sub keySA で配信される．このときの配信はマルチキャスト通信を用いる．本手法により，視聴権限を持つ LKH クライアントのみが復号化でき，更新された KEK’ と sub key’，EndTime を得る事ができる．そして，ユニキャスト通信では 7 回の通信が必要な処理を，マルチキャスト通信を用いる手法により 3 回の通信で行うことができる．

時間単位の鍵更新では，ツリーを 2 本作成する必要がある．1 本のみだと設定できる時間に限界がある．図 5.3 のツリーは 8 本の枝があり，視聴権限を 30 分ごとと設定している．8 本目の枝は 240 分の視聴権限が与えられる枝である．240 分以上の視聴権限を持つ LKH クライアントは 240 分の sub key を得る．ツリーが更新されていき，270 分，300 分の枝ができると，LKH サーバは 270 分，300 分と EndTime を送信する．270 分の EndTime を

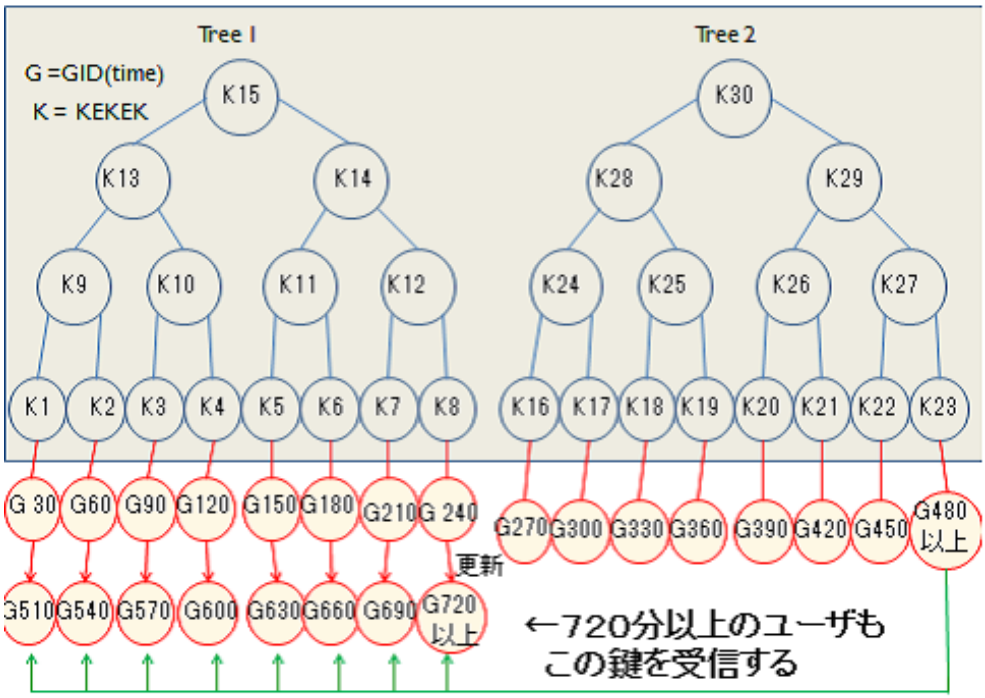

図 5.5: 2 本のツリー作成図

受け取った 270 分の視聴権限を持つ LKH クライアントは，sub key’ と KEK’ を公開鍵暗号方式のユニキャスト通信で LKH サーバから受信する．このようにして，まだツリーに存在しない視聴権限に対応する．ツリーが 1 本の場合，最後の枝である 240 分の枝を更新するとき，240 分で視聴が終了するユーザと，240 分以上の視聴権限を持つユーザの区別ができない．最後の枝が更新されるとき，まだ視聴権限を持つユーザにだけ sub key’ と KEK’ を配信する方法がなくなる．よって，240 分以上の視聴権限を持つ LKH クライアントは，コンテンツが視聴できなくなる．しかし，ツリーを 2 本作成する事でこの問題を解決する事ができる．1 本目の最後の枝が更新されるとき 240 分の枝に残っている 240 分以上の視聴権限を持つユーザには，予め作成していた 2 本目の枝の鍵を配信すれば良い．2 本目が全て更新された場合，1 本目の鍵をもらいに行く．このようにして，2 本のツリーを使い続ける事で問題を解決した．図 5.5 に概要図を示す．

葉ノードを新たに加える事 (insert 機能) によって，鍵の更新時間を柔軟に調整できる．通常，LKH サーバが 30 分ごとの鍵更新を設定した場合，30 の倍数分のコンテンツ，10 時から 10 時 30 分まで放送される 30 分ドラマコンテンツや，10 時から 11 時まで放送される 60 分ドラマコンテンツのように限定された時間でしかコンテンツを作成できない．しかし，この insert 機能によって，10 時から 10 時 15 分までの 15 分間のコンテンツや，10 時から 10 時 21 分までの 21 分間コンテンツといった細かい時間でコンテンツを作成できる．また，この機能を応用する事で視聴時間での課金も可能となる．通常，コンテンツ単位で

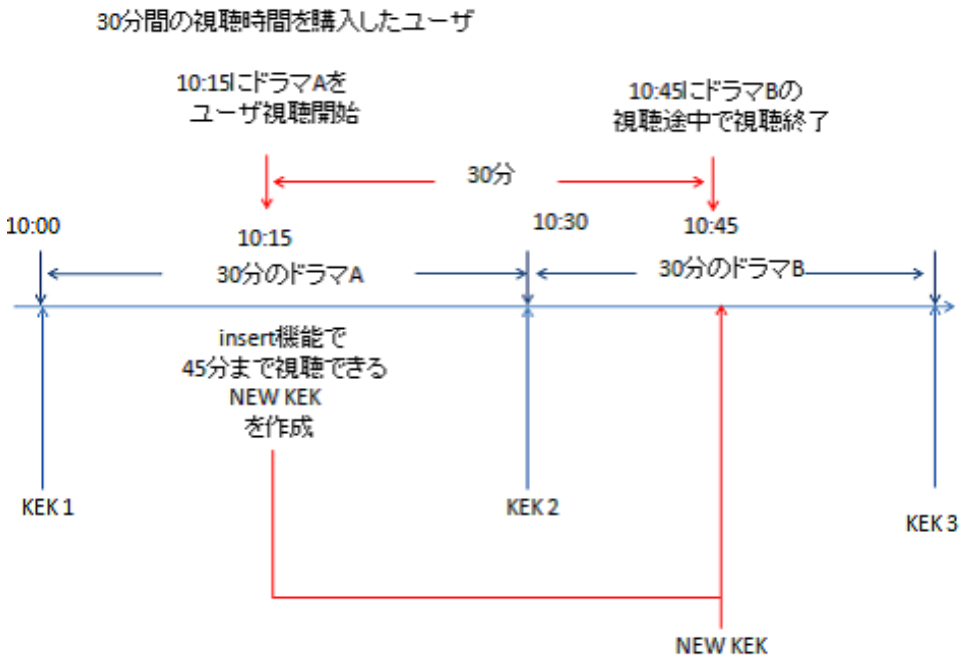

図 5.6: 視聴時間での鍵更新図

課金している為，10 時から 10 時 30 分までのコンテンツを購入した場合，10 時 15 分で視聴を開始すると 10 時 30 分までしか視聴できない．しかし，insert 機能を利用すると 10 時 15 分から 10 時 45 分までの視聴時間での課金もできるようになる．詳細図を図 7.5 に示す．10 時 15 分の時点で 30 分間ドラマチャンネルを視聴したいユーザがいた場合，LKH サーバが 45 分の枝を作成し，鍵を配信する．現在の実装では，視聴を中止した際も視聴可能時間は経過してしまうので，視聴中止時に残り視聴可能時間を計算し，再度視聴を開始した際に，残り時間に見合った鍵を LKH サーバに取りに行く機能が必要である．

75 分の葉ノードを加える場合を図 5.7 に示す．75 分の葉ノードを加える場合，90 分の枝に新しいノードを加える．新しい葉ノードは，自分が持つ時間情報を既存の葉ノードの視聴権限と比べていく．そして，初めて自分の視聴権限よりも大きくなった枝に加わる．75 分の場合，30 分，60 分と比べていき，自分よりもはじめて大きくなる 90 分の枝に加わる．90 分のノード鍵はすでに各ユーザに配信されているので，90 分のノードの上位に新しいノードを設ける．図の場合，それが K16 ノードとなる．75 分ノードは K16 ノードに加わる．鍵が更新されて，75 分ノードが不要になるときは，逆の動作で K16 と 75 分ノードを 90 分の枝から削除する．

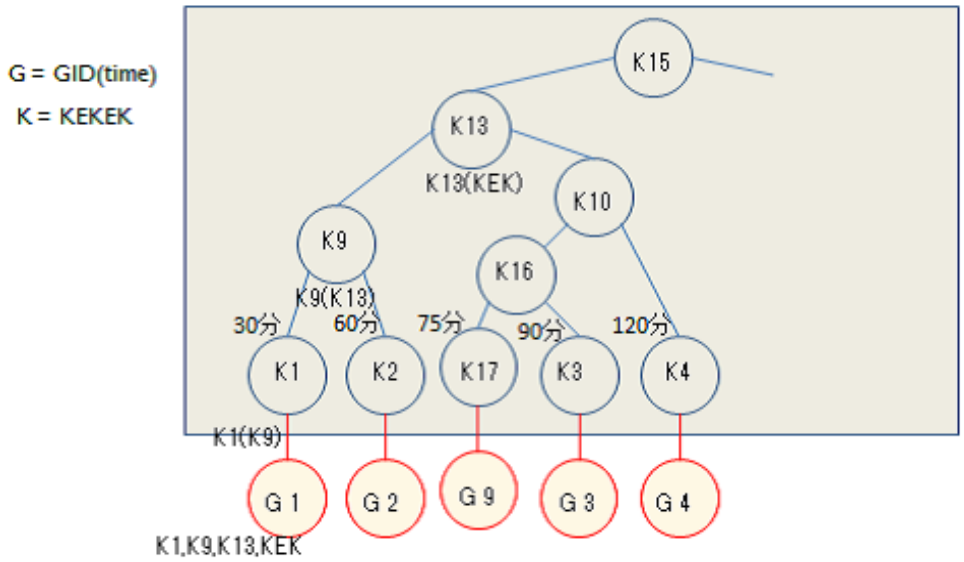

図 5.7: insert モジュール図

5.1.3 SA 作成モジュール

SA 作成モジュールを，図 5.2 に示す．

暗号化には，IPsec（Security Architecture for Internet Protocol）[26] を使用する．Time Driven LKH は，鍵の更新プロトコルの為，暗号化を行わない．SRTP などの暗号化プロトコルも使用できるが，今回はインストールが容易な IPsec を使用する．IPsec は，暗号化をネットワーク層のレベルで行うプロトコルである．AH (Authentication Header) による認証，ESP (Encapsulated Security Payload) によるデータの暗号化，IKE を用いた鍵交換などができる．しかし，本研究では，前述したとおりパスワードによって認証を行うので AH は必要がなく，また，鍵交換は Time Driven LKH が行うため IKE も必要ない．このため，ESP によるデータの暗号化機能だけを利用する．

IPsec で SA を張るには送信先アドレスと受信先アドレス，SPI (Security Parameter Index)，key などの値が必要である．key は，sub key や KEK を用いる．SPI は，SA を識別する為に必要な値であるため，サーバ，クライアントにかかわらず送信先，受信先，鍵が同じ SA には同じ spi を指定しなければならない．よって，ここでは spi に key ID を利用する．

Time Driven LKH では，sub key を key として張る SA と KEK を key として張る SA がある．この 2 種類の SA は，GSAKMP などで定義されている Rekey SA に相当する．

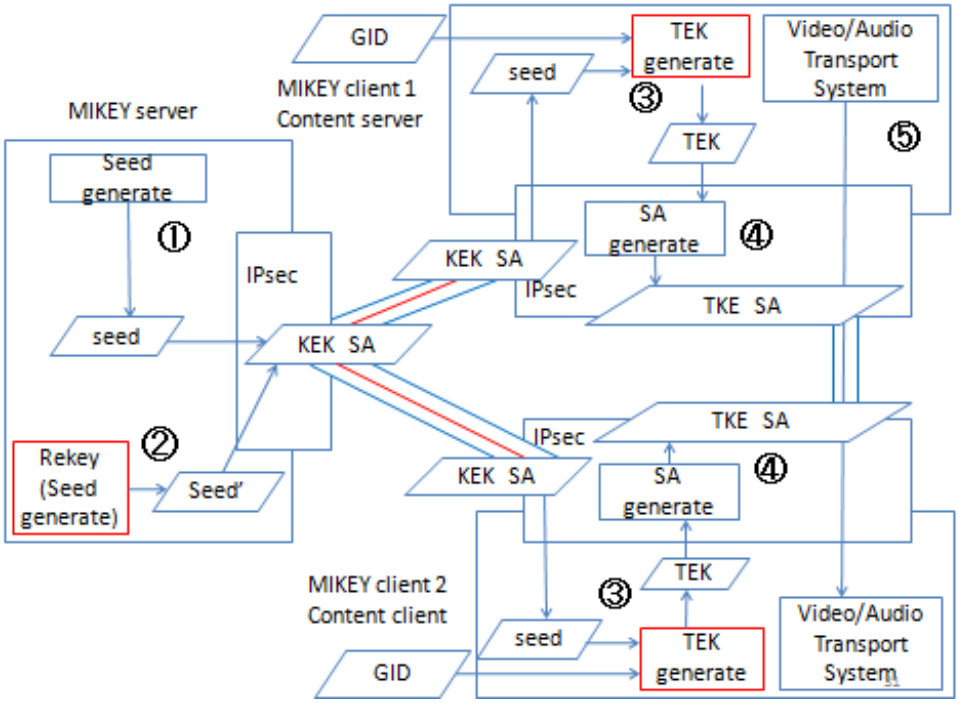

図 5.8: MIKEY モジュール図

5.2 Time Driven MIKEY

MIKEY の機能を参考に時間単位での鍵更新機能を新たに加えた MIKEY を Time Driven MIKEY と呼ぶ．図 5.8 に Time Driven MIKEY の動作モジュール図を示す．

5.2.1 seed 配信モジュール

seed 配信モジュールを，図 5.8 に示す．

Time Driven MIKEY サーバ（以下，MIKE サーバ），Time Driven MIKEY クライアント（以下，MIKEY クライアント），そのどちらも上記の，いずれも前述した KLH クライアントとして動作する．

MIKEY サーバは，seed を作成し配信する．KEK で張られた SA を用いて，マルチキャスト通信で seed を配信する．KEK を持たないユーザはたとえ受信しても，暗号化された seed を復号化できない．このため，MIKEY サーバは安全に seed を配信する事ができる．配信された seed から TEK を作成する部分は受信ノードで述べる．

5.2.2 鍵更新モジュール

鍵更新モジュールを，図 5.8 に示す.

一定時間が経過すると TEK は更新されなければならない．TEK が盗まれた場合，その TEK で暗号化したコンテンツが視聴権限のないユーザに視聴できてしまうため，その危険を最小限に抑える為に KEK で暗号化されるよりも短い時間間隔で鍵を更新し，コンテンツの Forward Secrecy と Backward Secrecy を確保しなければならない.

TEK の更新は，MIKEY サーバが seed を更新して配布する事によって実現する．更新された seed を seed' とする．seed' を受信することで MIKEY クライアントは，TEK' を作成する．この TEK' で新しく SA を張る事によって鍵の更新となる．詳細な動作は，受信ノードの 6.4.2 で述べる.

5.3 受信ノード

受信ノードにおける動作モジュール図を図 5.2，図 5.8 に示す．受信ノードは seed を受信すると TEK を作成して SA を張る．また，LKH サーバから sub key や KEK を受信すると同様に SA を張る．TEK で張った SA を通して，コンテンツ配信者はコンテンツを送信し，コンテンツ受信者はコンテンツを受信する.

5.3.1 鍵作成モジュール

鍵作成モジュールを，図 5.2 に示す.

受信ノードは MIKEY サーバから seed を受信したときに TEK を作成する．TEK の作成には seed と GID が必要となる．seed と GID を組み合わせる事により TEK を作成する．seed と GID，それぞれ二つの値が一致した受信ノード同士でなければ TEK による SA を張る事ができない．seed と GID A からは，TEK A が作成され，seed と GID B からは TEK B が作成される．配信される seed は同一の物である，GID の違いにより，コンテンツグループが管理される.

一定の時間が経過すると M サーバは seed' をマルチキャスト通信で送信する．受信ノードは seed' を受信すると，新たに seed' と GID から TEK '' を作成し，SA を張る.

5.3.2 SA 作成モジュール

受信ノードは 2 種類に分けられる．MIKEY サーバの受信ノードと MIKEY クライアントの受信ノード，いずれも LKH クライアントとして，KEK で張った SA を用いて seed の配信と受信を行う． MIKEY サーバの受信ノードは KEK SA と sub keySA しか張らないが，MIKEY クライアントの受信ノードはその 2 つのほかに TEK SA を張る．GSAKMP などで定義されているデータ SA がこれに相当する．MIKEY クライアントの受信ノード同士で TEK SA を張り，ストリームの送受信を行う．鍵が更新されるたびに，これらの SA は張りなおされる．張りなおす際，パケットロスが起きない様にこの SA は複数張っておく必要がある．

5.3.3 動画配信モジュール

MIKEY クライアントの受信ノードには，コンテンツ配信者と受信者が存在する．TEK SA でコンテンツ配信者は配信をし，コンテンツ受信者はストリームを受信する．このときの，ストリームはマルチキャスト通信で配信される．TEK SA を通して配信されるため，TEK SA を張っているユーザにしか，コンテンツは受信されない．TEK SA を張っていないユーザは復号化できない為，安全性が保たれている．動画配信モジュール図を図 5.8 に示す．

第6章　実装

本章では，第 5 章で述べた設計をもとに，時間手法の鍵管理システムの実装に関して述べる．Time Driven LKH サーバと Time Drivem MIkey サーバ，受信ノードに分けて説明する．

6.1　実装環境

本システムの実装を行った環境を図 6.1 に示す.

ＯＳ	Debian Linux 2.6.26-2-686
プログラム言語	C 言語
コンパイラ	gcc-4.3.2

表 6.1: 実装環境

6.2　Time Driven LKHサーバ

本節では Time Driven LKH サーバの実装概要について述べる．鍵配信モジュール，鍵更新モジュール，SA 作成モジュールに分けて説明する．

6.2.1　鍵配信モジュール

LKH クライアントに配信する鍵などの情報は，ツリーによって関連付けられている．sub key や Key ID など SA を張る為に必要な情報などがツリーノードと呼ばれる構造体に格納されている．list 構造によってノード同士がツリー上に関連付けられている．図 6.1 に構造体を示す.

```
typedef struct tree_node
{

        struct tree_node *left;
        struct tree_node *right;
        struct tree_node *parent;

        char key[256];

        char KEK[256];

        int value;

        int balanced;

        int flag;

        int Leafflag;

        int TimeID;

}tree_node;
```

図 6.1: tree_node 構造体

図 6.2 にツリーを示す．*left は，ツリー上に位置づけられる自分の左下に位置するノードのアドレスが格納される構造体ポインタである．K13 ノードからすると K9 ノードのアドレスが*left には格納されている．*right は，ツリー上に位置づけられる自分の右下に位置するノードのアドレスが格納される構造体ポインタである．K13 ノードからすると K10 ノードのアドレスが格納されている．*parent は，ツリー上に位置づけられる自分の 1 つ上位に位置するノードのアドレスが格納される構造体ポインタである．K13 ノードか

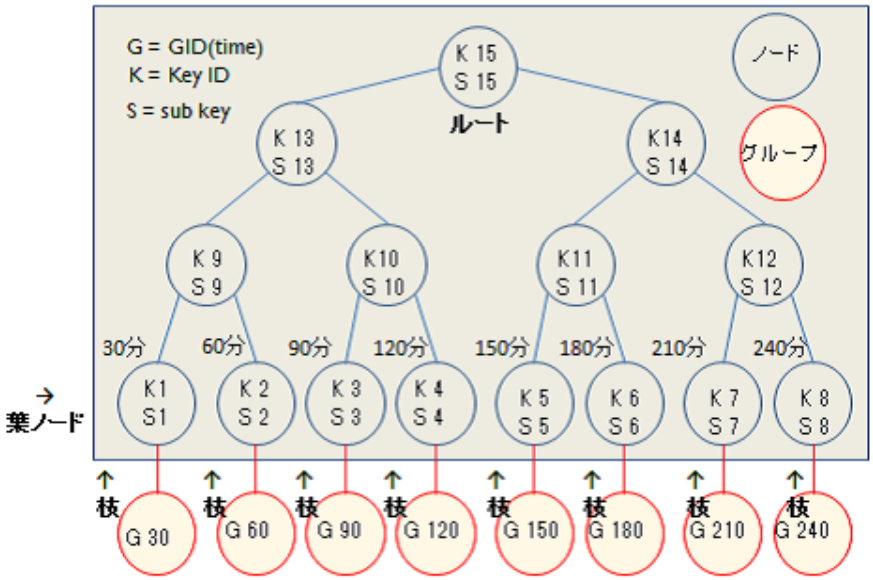

図 6.2: Time Driven LKH ツリー概要図

らすると K15 ノードのアドレスが格納されている．このようにノードを list 構造によってツリー上に関連付けている．

ノード 1 つ 1 つには様々な情報が含まれている．key は，LKH サーバと LKH クライアントが SA を張るときに必要となる鍵，つまり sub key である．value は，SA を張るときに SPI として使用する．TimeID と Leafflag，KEK は，主にツリーの最下層に属する葉ノードで必要となる．TimeID には，30 分，60 分といった時間情報が入力される．LKH クライアントが，権利情報として 30 などを入力してきた際に識別する為に必要となる．Leafflag は，他のノードと葉ノードを識別するときに必要となる．葉ノードには KEK 情報が入力される．flag は，鍵の更新時に必要となる．詳細は 6.2.2 項で述べる．value と balanced は，ツリー作成時に必要となる．

ツリーを作成する場合，value を比較してノードを位置づけていく．1 つめに加えられるノードをまず一番上層に位置するルートノードとする．新しいノードを加えるとき，ルートノードよりも value が大きい場合は*left に新しいノードを格納する．ルートノードよりも value が小さい場合は*right にノードを格納する．このようにノードを格納していくと，ルートノードの左側と右側で偏りができてしまう．効率的な鍵の配信をする為にはツリーに偏りができてはいけない．よって，rotateLeft 関数と rotateRight 関数を用いてバランスをとる．ノードが左に偏り過ぎた場合，balanced に左超過を表す値が代入される．この値が代入されると rotateLeft 関数が起動し，一部左側に関連付けられているノードがルートノードの右側に移動する．反対に，balanced に右側超過の値が代入されると，rotateRight 関数が起動し，一部右側に関連付けられているノードをルートノードの左側に移動する．これによって，ツリー作成時にバランスのとれたツリーが作成される．

```
struct LKH{

        char LKHkeys[256*7];

        char KeyID[128];

        char KEK[256];

        int TimeID;

        struct tree_node *Leafnode;
};
```

図 6.3: LKH 構造体

LKH 構造体は，鍵を送信する際に使用する．LKH 構造体に含まれる情報を 図 6.3 に示す．LKH 構造体は葉ノードの数だけ作成される．LKH 構造体には各枝の葉ノードの情報が代入される．*Leafnode には，葉ノードのアドレスが格納される．LKHkeys には枝に属する全ノードの sub key 情報が代入される．図 6.2 の K1 が属する枝の場合，K1，K9，K13，K15 の sub key が入力される．KeyID には同じく枝に属する全ノードの value が入力されている．KEK には，葉ノードに含まれていた KEK が代入され，TimeID には各枝の葉ノードが持つ時間情報が入力されている．LKH サーバは，LKH クライアントから権利情報を入力されると TimeID と比較し，同一の TimeID が属する LKH 構造体から LKHkeys と KeyID，KEK を送信する．この送信は RSA 暗号を用いた公開鍵暗号方式で行う．

6.2.2 鍵更新モジュール

鍵は一定時間 sleep 関数で停止されてから更新される．更新時間は予め LKH サーバを起動させるときに，引数で入力しておく．LKH サーバは引数で入力された値の倍数ごとに TimeID に情報を入力していく．引数が 30 の場合，30，60，90 と TimeID に値を入力していく．LKH サーバは起動から 30 秒，60 秒，90 秒後に鍵を更新していくが，insert 機

能によって，TimeID に 30 の倍数以外の数が入力されている場合がある．よって，単純に入力された引数の倍数時間ごとに鍵を更新するのではなく，insert 機能に対応した sleep の入力時間算出法が必要となる．図 7.5 に鍵の更新時間を設定するプログラムを載せる．time は LKH 構造体の番号である．更新されるたびに数を 1 つずつ増やし，更新される枝をずらしていく．numberOfUsers は 1 本のツリーに属する葉ノードの数である．ツリーは 2 本作成するので葉ノードの数は numberOfUserf の 2 倍ある．

if(time == ((numberOfUsers*2)-1)) は，全ての枝を更新し終えた事を意味している．全葉ノードから 1 を引いているのは，time が 0 から数を数えている為である．else には，最後の枝を更新する場合以外の動作が書かれている．次に更新される枝の TimeID から，今回更新する枝の TimeID を引いた値が次の更新までの時間，sleeptime となっている．これにより，次に更新される枝が insert 機能によって新しく作成された枝だとしても対応できる．最後の葉ノードの場合のみ，1 本目の枝から最後の枝の TimeID を引いたものになっている．しかし，sleeptime には，rekey にかかる時間が含まれていない．rekey に 0.1 秒かかる場合，1 回 rekey する毎に 0.1 秒ずつ遅れてしまう．100 回の rekey が行われると 10 秒 rekey を開始する時間が遅れてしまう．よって，rekey 動作にかかった時間を計算に入れた sleeptime を算出しなければならない．

sleep 関数を抜けると鍵の更新が始まる．枝ごとに鍵の値を更新し，更新した値を適切な SA を通じて，適切な数だけ配信する．図 6.1 を用いて述べる．K1 の枝が更新される場合，K1 から*parent を利用して，K9，K13，K15 と鍵などを更新していく．更新される際，遡ってきたノードとは反対に属するノードの flag を立てて行く．K1 から K9 に遡った場合，K9 は*left に属する K1 から遡ってきた事になるので，K2 の flag に 1 を代入する．同じように，K13 では K10 に 1 を代入し，K15 では K14 に 1 を代入する．次に，ルートノードである K15 から，*left もしくは*right ノードに flag が立っていないかを調べていく．K15 ノードの場合，K14 ノードに flag が立っているので，K14 ノードの value と key を rekey 構造体の KeyID と key に代入する．rekey 構造体は図 6.5 に示す．また，K15 ノードの KeyID と key を rekey 構造体の LKHkeysID と LKHkeys に代入する．K14 ノードの flag を初期化し，新たな rekey 構造体を作成し，反対のノードである K13 ノードに移る．K13 ノードでも，両ノードの flag を確認する．K13 ノードの場合，K10 に flag が立っているので，K15 のときと同様に，K10 の KeyID と key を新しい rekey 構造体に代入する．rekey 構造体の LKHkeysID と LKHkeys には，これまで通過してきたノードの keyID と key を入力する．K15 と K13 の KeyID と key を新しい rekey 構造体の LKHkeysID と LKHkeys に入力する．その後，K10 ノードの flag を初期化し，次の rekey 構造体に移動する．このように，葉ノードに到達するまで rekey 構造体に入力していく．葉ノードにまで到達し，全ての情報を入力し終わると，rekey 構造体の KeyID と key で SA を張り，LKHkeysID と LKHkeys を送信していく．

```
if(time == ((numberOfUsers*2)-1))
            {

          sleepTime =
          (((LKH->TimeID) - (LKH+time)->TimeID));

              }
              else{

              sleepTime =
   (((LKH+(time+1))->TimeID) - ((LKH+time)->TimeID));

                 }
            sleep(sleepTime);

}
```

図 6.4: sleep time 算出方法

6.2.3 SA 作成モジュール

図 7.6 は setkey コマンドを使用して，SA を張るコマンドである．LKHserveraddress は LKH サーバのアドレス，LKHmulticastaddress は，KEK' や sub key' を送信するマルチキャストアドレスを指している．keyid は，サーバから送られてきた KeyID である．今回は，KeyID を SPI として使用している．key が sub key である．SPD も同様に張るのだが，LKH サーバでは，鍵を出力するので out，LKH クライアントでは鍵を受信するので in になっている事に注意しなければならない．

マルチキャストで SA を張る場合，受信側の SA でもサーバからマルチキャストアドレスに SA を張る．これは，受信者に届くデータの送信元と送信先が，サーバからマルチキャストアドレスになっている為である．

LKH クライアントは届いた sub key 全てを用いて SA を張る．図 6.2 で説明する．60 分の視聴権利がある LKH クライアントには K15，K13，K9，K2 の鍵が配信される．LKH クライアントはそれらすべての sub key で SA を張る．30 分の枝が更新される場合，更新

```
struct rekey{

        char LKHkeys[256*7];

        char key[256];

        char LKHkeysID[32];

        int keyID;

        char KEK[256];
};
```

図 6.5: rekey 構造体

```
送信者
add $LKHserveraddress $LKHmulticastaddress esp $keyid -E rijndael-cbc 0x$key;
spdadd $LKHserveraddress $LKHmulticastaddress any -P out ipsec
esp/transport//require;
受信者
add $LKHserveraddress $LKHmulticastaddress esp $keyid -E rijndael-cbc 0x$key;
spdadd $LKHserveraddress $LKHmulticastaddress any -P in ipsec
esp/transport//require;
```

図 6.6: KEK，sub key 配信用 SA 作成コマンド

された K15’，K13’，K9’ は K2 で張られた SA を通じて 60 分の視聴権利がある LKH クライアントに配信される．LKH クライアントはすでに K2 で SA を張っているので更新された鍵を受信する事ができる．

通常，IPsec を用いる場合，IKE によって必要となるパラメータを交換し，適切な SA のみ張る．しかし，本実装では受信した全ての sub key で SA を張っている．メリットとして，IKE によるネゴシエーションを省略する事ができる．デメリットとしては，常に必要のない SA を張っている点が挙げられる．IKE を用いるメリットとしては，LKH サー

バが使用する SA を事前に各 LKH クライアントに通知するので，不必要な SA を張らないで済む点が挙げられる．しかし，各 LKH クライアントが IKE によるネゴシエーションをしなければならないデメリットがある．

6.3 Time Driven MIKEY サーバ

Time Driven MIKEY サーバの実装概要について，seed 配信モジュール，鍵更新モジュール，SA 作成モジュールに分けて述べる．

6.3.1 seed 配信モジュール

seed 作成部分を述べる．図 7.7 に seed 作成部分を載せる．seed は OpenSSL に定義されている RANDseed 関数で作成される．引数に入れた値から seed を作成する．本実装では，time 関数で暦時刻を取り，RANDseed 関数に引数として入力している．できた seed はマルチキャスト通信によって MIKEY クライアントに配信される．seed から TEK を作成する部分は， 6.4.1 項で述べる．

6.3.2 鍵更新モジュール

引数によって入力された時間ごとに鍵を更新する．10 と入力された場合，MIKEY サーバが起動してから 10，20，30 秒と seed を更新していく．seed は，新たに time 関数で暦時刻を取り，RANDseed 関数を再度起動する事で更新する．更新された seed はマルチキャスト通信によって配信され，MIKEY クライアントが更新された TEK を作成する．

6.3.3 SA 作成モジュール

seed 配信用 SA の作成方法を図 7.8 に載せる．MIEYserveraddress は MIKEY サーバのアドレスを入力し，MIKEYmulticastaddress には seed を配信するマルチキャストアドレスを入力する．

```
uint8_t *mikey_generate_rand(uint8_t randlen) {
        uint8_t *rand;
        time_t timeval;

        rand = (uint8_t *)malloc(randlen);

        if (rand == NULL) {
                MIKEY_LOG("memory allocatioin failure.");
                goto ret;
        }

        timeval = time(NULL);

        RAND_seed(&timeval, sizeof(timeval));

        if (RAND_pseudo_bytes(rand, randlen) != 1) {
                MIKEY_LOG("random generation failure.");
                return NULL;
        }

ret:
        return rand;
}
```

図 6.7: seed 作成

6.4 受信ノード

鍵作成モジュール，SA 作成モジュール，動画配信モジュールに分けて受信ノードの動作を述べる．

送信者
add $MIKEYserveraddress $MIKEYmulticastaddress esp $keyid -E rijndael-cbc 0x$key;
spdadd $MIKEYserveraddress $MIKEYmulticastaddress any -P out ipsec esp/transport//require;
受信者
add $MIKEYserveraddress $MIKEYmulticastaddress esp $keyid -E rijndael-cbc 0x$key;
spdadd $MIKEYserveraddress $MIKEYmulticastaddress any -P in ipsec esp/transport//require;

図 6.8: seed 配信用 SA 作成コマンド

```
static void xor(uint8_t p1[], uint8_t p2[], uint32_t n) {
        uint32_t i;

        for (i = 0; i  n; i++) {
                p1[i] ^= p2[i];
        }
}
```

図 6.9: TEK 作成

6.4.1 鍵作成モジュール

MIKEY サーバから seed を受信すると，GID と組み合わせて TEK を作成する．TEK を作成するプログラムを図 6.9 に示す.

p1 には seed のビット列が入力されており，p2 には GID のビット列が入力されている．ビット毎の XOR を求める演算子で p1 に TEK のビット列を入力していく．その後，ビット列を文字列に変換して SA を作成する.

送信側
add $senderaddress $address1 esp $x -E rijndael-cbc 0x$str[$x];
spdadd $senderaddress $address1 any -P out ipsec esp/transport//require;
受信側
add $senderaddress $address1 esp $x -E rijndael-cbc 0x$str[$x];
spdadd $senderaddress $address1 any -P in ipsec esp/transport//require;

図 6.10: コンテンツ配信用 SA 作成コマンド

6.4.2 SA 作成モジュール

TEK からコンテンツを配信する SA を張る．コマンドを図 6.10 に示す．senderaddress にはコンテンツ配信者のアドレスが入力され，address1 にはコンテンツを配信するマルチキャストアドレスが入力される．str[x] は，鍵を入力する配列である．

6.4.3 動画配信モジュール

図 7.10 は vlc を起動するときのコマンドである．送信者側と受信者側のコマンドを載せる．動画配信は vlc 以外も使用できるが，今回はマルチキャストによる配信の容易さから，vlc を使用していく．file には配信するコンテンツを入力し，address1 には配信に使用するマルチキャストアドレスを入力する．myaddress には，配信に使用されているマルチキャストアドレスを入力する．

送信者
cvlc $file –sout udp:$address1:1234 –loop
受信者
vlc udp://@$myaddress

図 6.11: vlc 起動コマンド

第7章　評価

本章では，第 6 章において実装された本システムに対する評価について述べる．

7.1　評価概要

本システムは，Time Driven LKH と Time Driven MIKEY を組み合わせている．よって，Time Driven LKH と Time Driven MIKEY のそれぞれと，組み合わせたときの定性評価を行う．また，鍵の更新にかかる時間の測定やコンテンツ配信中にパケットロスが起こらない SA の数を計測する．

7.2　Time Driven LKH

実検環境を図 7.1 に示す．本評価環境は，1 台の LKH サーバと 2 台の LKH クライアントによって構成される．LKH クライアントは届いた鍵で SA を張り，vlc で動画の送受信を行う．

7.2.1　定性評価

Time Driven LKH の性能を評価する．第 5 章の設計通りの動作をしているか実験をして確認する．LKH クライアント A をコンテンツ送信者，LKH クライアント B をコンテンツ受信者とする．LKH クライアント A の視聴権利情報を 480 秒，LKH クライアント B の視聴権利情報を 360 秒とし，360 秒後に LKH クライアント B がコンテンツを視聴できなくなるか実験する．

実検中，コンテンツ視聴の際に映像が乱れる事があった．図 7.2 に，その画像を載せる．これは，LKH クライアントが LKH サーバから配信された KEK を受信し，SA を張り替えるときにおこる現象である．LKH クライアント A と LKH クライアント B の SA 張り

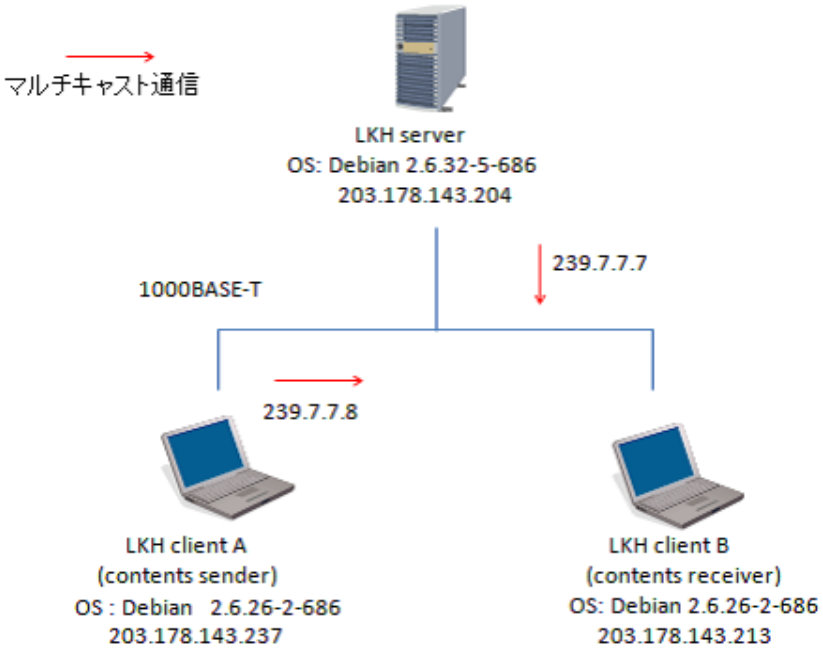

図 7.1: テスト環境

替えにかかるタイムラグが原因であった．LKH サーバから配信された KEK' で LKH クライアント A が SA を張り替え，コンテンツを暗号化して配信する．そのとき，LKH クライアント B が SA を張り替え終わっていない場合，コンテンツを復号化できない．また，LKH クライアント B が SA を張り替え終わったにもかかわらず，LKH クライアント A が SA を張り替え終わっていないときにもこのような現象が現れる．図 7.3 に概要図を示す．

この現象の解決方法として，SA を複数張る事にした．この現象は LKH クライアント A，LKH クライアント B のどちらかが，KEK' の SA を張り遅れる事で起る．よって，複数の KEK を予め配信し，複数の SA を張っておく事で，SA の張り遅れで起るこの現象を解決する．

LKH クライアントは KEK' を受信して KEK' SA を張っても，KEK SA をすぐには消さない．一定時間 KEK SA でデータのやり取りをする．LKH クライアント A をデータ送信者とすると，LKH クライアント A は一定時間 KEK SA でデータを送信したのち，KEK SA を消去し，KEK SA' でデータを送信する．データ受信者である LKH クライアント B は，KEK' SA で暗号化されたデータを受信すると，KEK SA を消去する．データ送信者が SA を切り替えるタイミングは固定にしているが，データ受信者全員が KEK' を受信した事が確認できなければならない．　複数の SA を張り実験をした．実験の結果，SA が切り替わっても映像が乱れる事はなくなった．正常に視聴できている画像を図 7.4 に示す．また，コンテンツが暗号化されているか tdpdump で調べた．図 7.5 に示す．LKH クライ

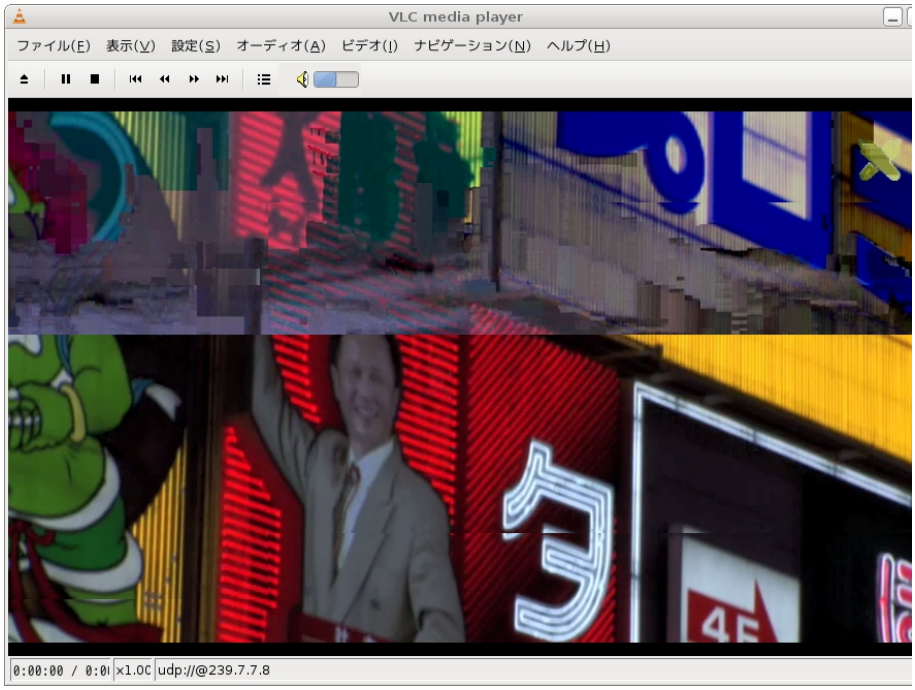

図 7.2: 乱れた画像

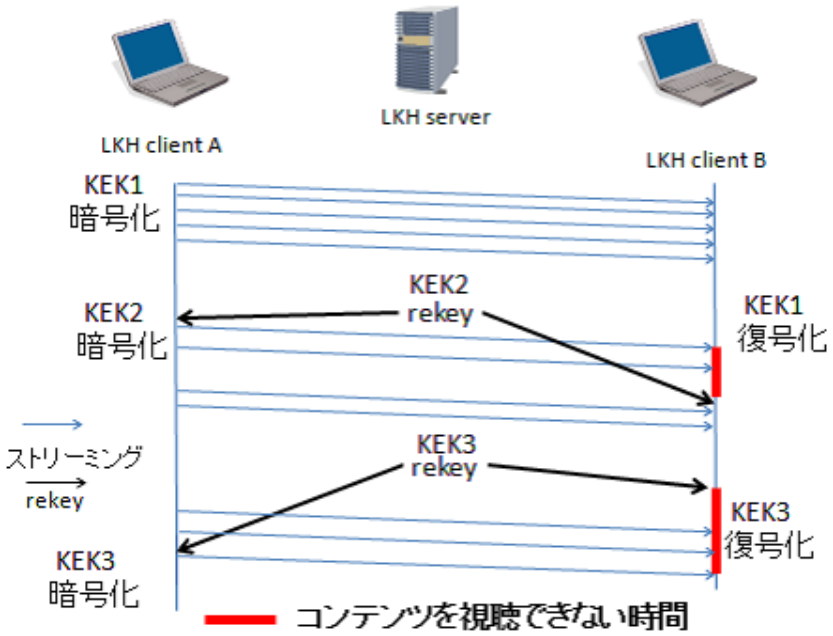

図 7.3: SA 切り替え

アント A からマルチキャストアドレス (239.7.7.8) に暗号化されたデータが配信されているのが分かる.

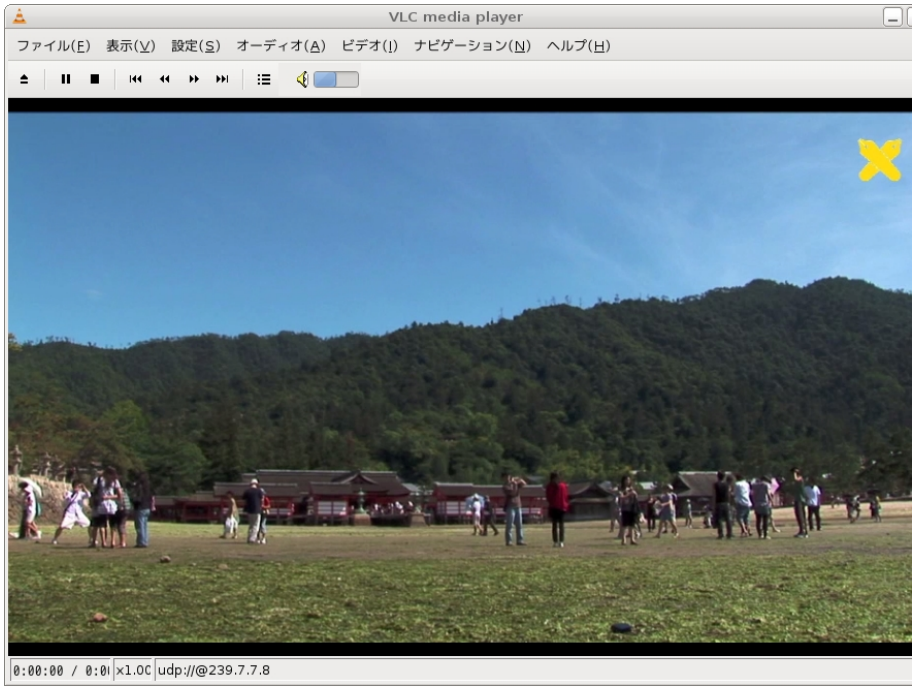

図 7.4: 正常な画像

7.2.2 定量評価

LKH サーバから SA 作成に必要な情報を受信してから，SA を張るまでの LKH クライアントの動作時間を計測する．LKH クライアントが LKH サーバから受信するメッセージは，更新して新しくできた時間情報，SPI に使用する key ID，sub key，KEK である．これらを受信し，ファイルに書き込み，wrapper がファイルから読み込んで，setkey コマンドを利用して SA が張り終わるまでの時間を計測した．

計測されたデータは大きく 2 種類の時間に分けられる．1 つは公開鍵暗号方式で鍵を受信して SA を張る時間．もう一方は，SA を通じて鍵を受信して SA を張る時間である．公開鍵暗号方式を用いる通信と IPsec を用いる通信は，どのような場合に使用するかは第 5 章で述べている．

IPsec を用いる場合，さらに細かく分類できる．IPsec を用いる鍵の更新では，sub key の数が異なる．sub key の数は，ツリーの階層の深さによって変化する．本実験では，ツリーの階層を 4 層としている．

表 7.1 に公開鍵暗号方式，IPsec を用いた場合の SA を張るまでにかかった最長時間

図 7.5: tcpdump の結果

(MAX)，最短時間（MIN)，平均（AVE)，標準偏差 (STD)，試行回数を示す．IPsec を用いた場合は，最も sub key の数が少ない sub key0 の場合と，最も sub key の数が多い sub key3 の場合で実験する．

	MAX	MIN	AVE	STD	試行回数
公開鍵暗号方式	3.99864912	3.841345787	3.9573496	0.029509278	35 回
IPsec(sub key 0 の場合)	0.484231949	0.455322981	0.458900408	0.003554133	280 回
IPsec(sub key 3 の場合)	2.25898695	1.718291044	1.849399018	0.117487291	20 回

表 7.1: 計測結果

公開鍵暗号方式の場合，最大 3.99 秒，平均して 3.95 秒の時間が必要となり，IPsec の場合は，sub key が 0 のとき，最大 0.48 秒，平均で 0.45 秒必要となる．sub key が 3 のときは，最大 2.2 秒，平均して 1.8 秒となる．公開鍵暗号方式の方が IPsec を用いた場合と比較して時間が多く必要とするのが分かる．これは，公開鍵暗号方式が暗号化されたデータを受信して，復号化の処理をしている為である．IPsec の場合，recv 関数でデータを受信するときには復号化の処理が済んでいる．しかし，公開鍵暗号方式はプログラム内でデー

タを復号化するので，その時間だけより多くの時間を必要とする．また，公開鍵暗号方式ではTCPで通信をしている為，UDPで通信をするIPsecよりもより多くの時間を必要とする．また，IPsecを用いた場合でも，受信する鍵の数や張るSAの数が増えれば，それだけ多くの時間を必要とする．

公開鍵暗号方式でSAを張るのに必要とする時間は最大で3.9秒となる．それ以下の場合，SAを張るのが間に合わず，暗号化されたデータを受信しても復号化できない．2本あるツリーの内，どちらか一方のツリーが全て更新された場合，LKHクライアントはLKHサーバに最新のsub keyを公開鍵暗号方式で取りに行く．このとき，LKHクライアントのアクセスが集中しない様に，次の枝が更新されるまで，分散して鍵を取りに行くように実装している．次の枝の更新時間が30分(1800秒)後の場合，ランダムな値を1800秒で割り，その余りの時間分だけsleep関数で待機してからLKHサーバにアクセスする．だが，公開鍵暗号方式は鍵を受信してからSAを張るまでに3.9秒必要とする．また，LKHサーバにアクセスしてから鍵を受信するまでにかかる時間は，インターネットの回線状況が関わってくる．回線の状況は予測の仕様がないため，多めに時間を見積もる必要がある．また，sub keyの数が増加する事で，SAを張るまでにかかる時間は増えていく．sub keyが3つ増えた事で，試行回数の差はあるが，平均して約1.4秒，SAを張るまでにかかる時間が増えた．ツリーの階層を増やし，sub keyの数を増加させる事は，LKHサーバだけではなく，LKHクライアントの負担にもなる．従って，ツリーの階層は深くなり過ぎないようにしなければならない．

7.3 Time Driven MIKEY

実検環境を図7.6に示す．本評価環境は，1台のMIKEYサーバと3台のMIKEYクライアントによって構成される．MIKEYクライアントは届いた鍵でSAを張り，vlcで動画の送受信を行う．鍵の送受信はマルチキャストアドレス239.7.7.8で送受信し，コンテンツの配信はマルチキャストアドレス239.7.7.5で行う．

7.3.1 定性評価

MIKEYクライアントは，1台がコンテンツ送信者で2台がコンテンツ受信者とする．コンテンツ受信者Aは，コンテンツ送信者と同じGIDを持ち，コンテンツ受信者Bは，コンテンツ送信者とは異なるGIDを持つ．コンテンツ受信者Aがコンテンツを視聴でき，コンテンツ受信者Bがコンテンツを視聴できない事を確認する．また，tcpdumpでコンテンツが暗号化されている事を確認する．そのとき，SPIの変化を調べSAが更新されているかを確認する．

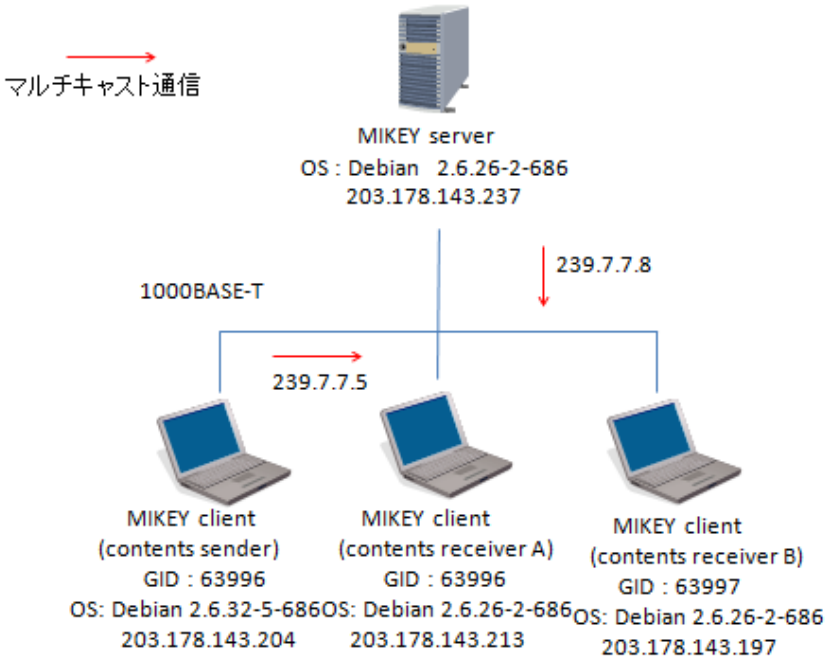

図 7.6: テスト環境

実験の結果，コンテンツ受信者 (GID63996) と同じ GID を持つコンテンツ受信者 A (GID63996) はコンテンツを視聴する事ができた．コンテンツ送信者と異なる GID を持つコンテンツ受信者 B(GID63997) はコンテンツを視聴する事ができなかった．その後，コンテンツ送信者の GID をコンテンツ送信者 B と同じ値 (GID63997) にするとコンテンツ受信者 A が視聴できなくなり，コンテンツ受信者 B は視聴できるようになった．

SA の数は LKH の結果を踏まえ，予め複数の SA を張った．これにより，SA の切り替えで映像が乱れる事はなかった．

図 7.9 の結果を見ると，コンテンツ送信者から配信されているコンテンツが暗号化されているのが分かる．また，0x00000d62 から 0x000006d1 に SPI が変化した事から鍵が更新されて，SA が切り替わっている事が確認できる．

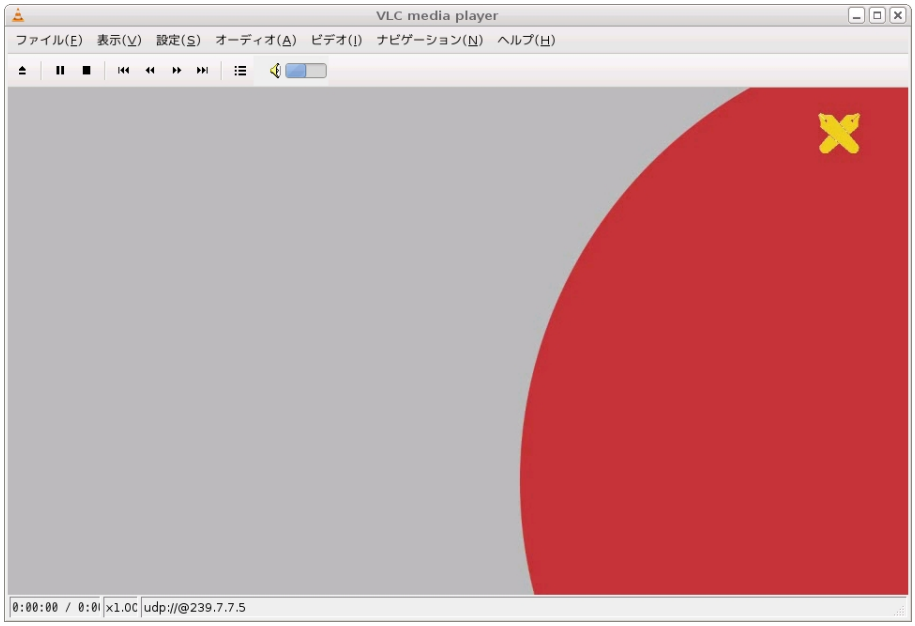

図 7.7: コンテンツ受信者 A(GID63996) の画像

図 7.8: コンテンツ受信者 B(GID63997) の画像

7.3.2 定量評価

MIKEY クライアントが MIKEY サーバからマルチキャスト通信で配信された seed を受信し，seed と GID を組み合わせて TEK を作成，TEK をファイルに書き込み，wrapper が読み込んで SA を張るまでにかかる時間を計測する．

表 7.2 に SA を張るまでにかかった最長時間（MAX），最短時間（MIN），平均（AVE），標準偏差 (STD)，試行回数を載せる．

	MAX	MIN	AVE	STD	試行回数
IPsec	2.142891884	0.396062851	0.455462404	0.118535517	450 回

表 7.2: 計測結果

計測の結果，最大 2.1 秒，平均 0.45 秒となった．平均時間は LKH の IPsec の結果とほ

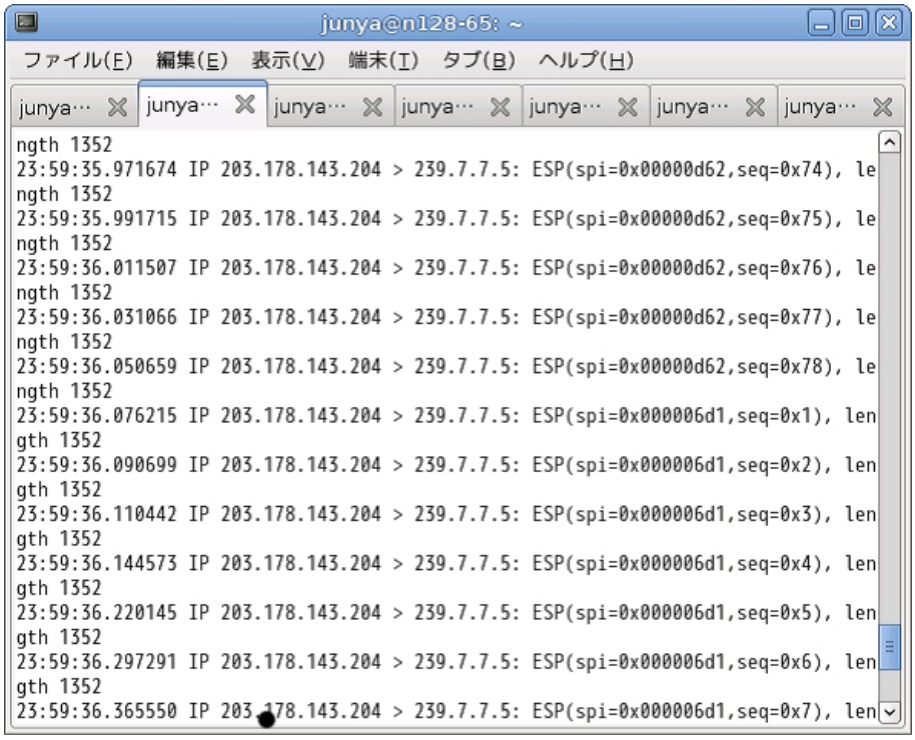

図 7.9: tcpdump の結果

とんど変化がない．しかし，最大時間は 1.5 秒ほど異なる．MIKEY と LKH の違いは，MIKEY が seed を GID と組み合わせて TEK を作成し SA を張るのに対し，LKH は複数の鍵を受信し，そのまま SA を張る所にある．複数の鍵を受信するのも TEK を作成するのも平均で見ればあまり大きな違いはない．しかし，MIKEY の鍵を作成する作業は受信された seed 値などによって必要とする時間に幅がある．今回の結果では，450 回の試行で 3 回の試行で SA を張るのに 1 秒以上かかっている．これらの結果を踏まえ，MIKEY の鍵更新時間の設定は最低でも 5 秒以上が良いことが分かる．

7.4 Time Driven LKH と Time Driven MIKEY

全部で 5 台の実験機を用意する．1 台は LKH サーバ，1 台は MIKEY サーバとする．残りの 3 台は受信ノードとし，1 台をコンテンツ送信者，2 台をコンテンツ受信者とする．LKH サーバは 30 秒ごとに鍵更新をし，MIKEY サーバは 5 秒ごとに鍵更新をするように設定した．コンテンツ受信者 A とコンテンツ受信者 B は，コンテンツ送信者と同じ GID を持たせる．コンテンツ受信者 A とコンテンツ受信者 B の権利情報は異なるものとする．詳細を図 7.10 に示す．

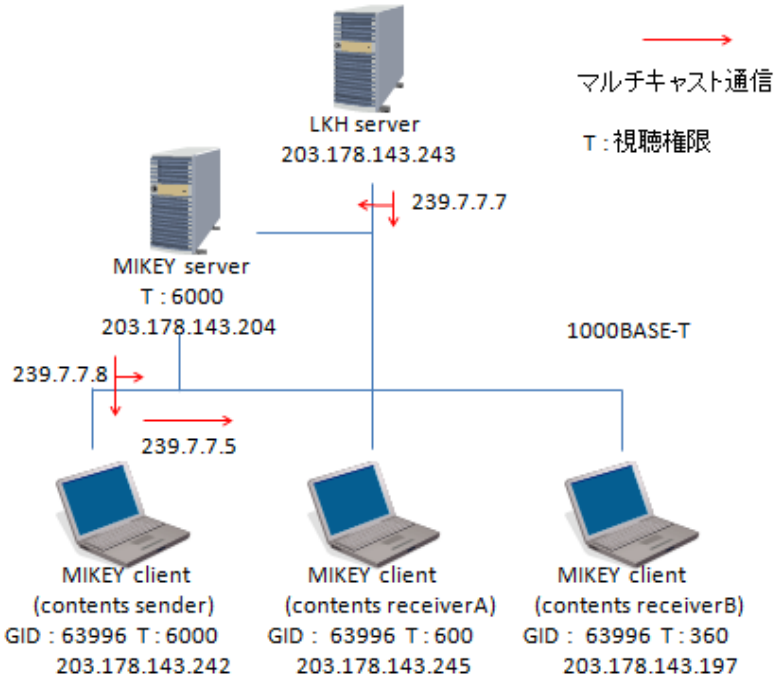

図 7.10: テスト環境

7.4.1 定性評価

実験の結果，360 秒後にコンテンツ受信者 B はコンテンツを視聴できなくなり，コンテンツ受信者 A だけがコンテンツを視聴できるようになった．図 7.11 に LKH サーバから送信されるデータが暗号化されている事を示す tcpdump の結果を載せ，図 7.12 には MIKEY サーバから送信されるデータが暗号化されている事を示す tcpdump の結果を載せる．また，コンテンツが暗号化されている事を示す tcpdump の結果を図 7.13 に示す．

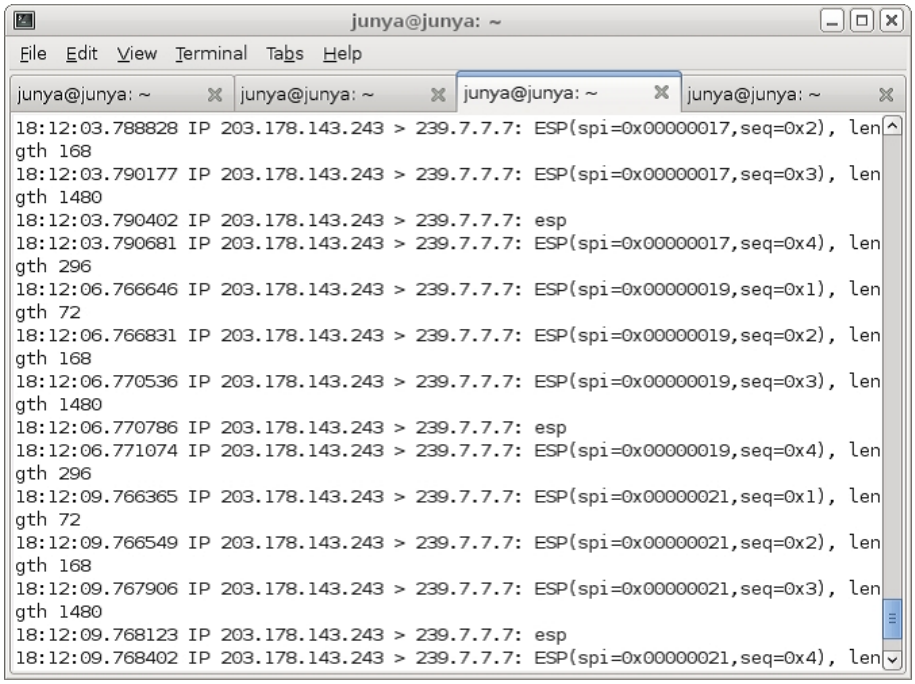

図 7.11: LKH の結果

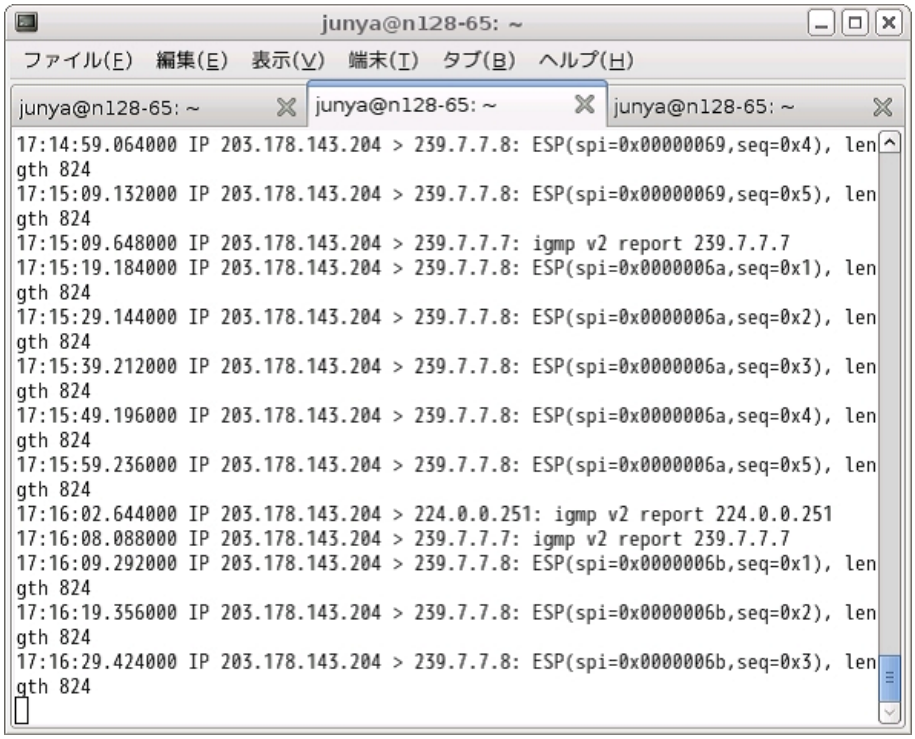

図 7.12: MIKEY の結果

junya@n128-65: ~
ファイル(F) 編集(E) 表示(V) 端末(T) タブ(B) ヘルプ(H)
junya@n128-65: ~
junya@n128-65: ~
length 1368
16:54:44.586737 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x00000ae0,seq=0x1068),
length 1368
16:54:44.590657 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x00000ae0,seq=0x1069),
length 1368
16:54:44.594892 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x00000ae0,seq=0x106a),
length 1368
16:54:44.597866 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x00000ae0,seq=0x106b),
length 1368
16:54:44.602057 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x00000ae0,seq=0x106c),
length 1368
16:54:44.621630 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x000003e2,seq=0x1), len
gth 1368
16:54:44.621649 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x000003e2,seq=0x2), len
gth 1368
16:54:44.621656 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x000003e2,seq=0x3), len
gth 1368
16:54:44.621730 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x000003e2,seq=0x4), len
gth 1368
16:54:44.621780 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x000003e2,seq=0x5), len
gth 1368
16:54:44.621786 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x000003e2,seq=0x6), len
gth 1368
16:54:44.630802 IP 203.178.143.242 > 239.7.7.5: ESP(spi=0x000003e2,seq=0x7), len

図 7.13: vlc の結果

第8章　結論

本章では，本研究をまとめ，今後の課題を述べる．

8.1　まとめ

本研究では，鍵更新時のユーザに対する鍵配布にかかる計算処理コストを低減する鍵管理システムの構築を目的として LKH と MIKEY を参考にして，Time Driven LKH と Time Driven MIKEY の実装を行った．本システムを用いる事により，コンテンツ数に影響されないグループ管理が可能となり，また，ユーザ数の増加に影響されにくい鍵の更新ができるようになった．

時間単位の鍵更新を用いた事で鍵の更新頻度が想定でき，従来の鍵更新システムよりもサーバの負担が少なくない．LKH のツリー構造を応用する事で時間区分でのユーザ管理を可能とした．

Time Drive MIKEY では，鍵の作成を MIKEY クライアントが行うことにより，MIKEY サーバがグループ毎に存在する鍵を管理する必要が無くなった．また，マルチキャスト通信を用いた時間単位での鍵更新に改善した事で鍵更新の度にユーザのアクセスが集中し，サーバの負担が増加するということもなくなった．

Time Driven LKH では，ツリーが行う鍵の更新を時間単位にした事で，30 分コンテンツを視聴できるユーザと 60 分視聴できるユーザとを区別する事ができるようになった．また，その際鍵の配信を可能な限りマルチキャスト通信で行う事でサーバの負担を可能な限り減らす事を可能とした．

本研究により，鍵更新時のユーザに対する鍵配布にかかる計算処理コストは低減した．これにより，pay-per-view 型リアルタイムセキュアストリーミングのサービスを行おうとするユーザは，既存のシステムが対応しきれなかったユーザ数とコンテンツ数に対応できるようになった

.

8.2　今後の課題

本節では，今後の課題について述べる．

8.2.1 パケットロス対策

本システムでは，マルチキャスト通信を利用している．マルチキャスト通信はUDP（User Datagram Protocol）の為，パケットロスが起こった場合，再送信しない．つまり，鍵更新の際，更新された鍵をマルチキャスト通信で配信する為，インターネットの状況次第ではパケットロスが起こり，鍵が届かないユーザが出てくる可能性がある．そういった場合，ユーザはコンテンツが視聴できなくなる．

Time Driven MIKEY の場合，比較的短時間の鍵更新を想定している為，被害は少なくて済む．例えば，10 秒間おきの鍵更新を想定している場合，seed が届かなかったユーザは 10 秒間コンテンツが視聴できなくなる．だが，Time Driven LKH は 30 分，1 時間といった長い時間での鍵更新の為，被害は大きい．また，sub key が届かなかった場合，3 時間視聴できるユーザが，今後更新された鍵を受け取る事が出来なくなり，1 時間しか視聴できないといった状況になる．

よって，この問題を解決する為に，クライアント側にて届いていない鍵を認識し，再送信を要求する機能が必要である．

8.2.2 鍵の配信

Time Driven LKH と Time Drivrn MIKEY は，初めの SA を張る為にサーバにユニキャストで鍵を収得する．しかし，pay-per-view 型リアルタイムセキュアストリーミングサービスを考えると，コンテンツの開始時間にアクセスが集中し，サーバに多大な負担をかける可能性がある．

この問題を解決する為に，サーバに最初の鍵を収得するのではなく，クライアントから鍵を収得するといった解決法が必要となる．

8.2.3 insert 機能

コンテンツの視聴権限だけではなく，コンテンツの視聴時間で課金サービスを行えるようにするには，insert 機能を改善していく必要がある．現在の実装では，10 時 15 分から視聴を開始した 30 分間視聴可能なユーザは，途中で視聴を止めても 10 時 45 分までしかコンテンツを視聴できない．よって，視聴を止めた時間から残りの視聴時間を計算し，その残り視聴分の鍵を新たに作成する機能が必要となる．

8.2.4 鍵の更新

現在の実装では，rekey 動作開始時間を次回更新する枝の視聴権限時間から今回更新する枝の視聴権限時間を引いたものにしている．次回更新する枝の視聴権限時間が 150 秒で今回更新する枝の視聴権限時間が 120 秒の場合，rekey 動作開始時間は 150 秒から 120 秒を引き 30 秒後となる，30 秒間 sleep 関数で待機した後，今回更新する枝の rekey 動作を開始する．しかし，この時間計算には rekey にかかる時間が含まれていない．rekey に 0.1 秒必要とする場合，1 回更新を行うと 0.1 秒ずつ rekey 動作開始時間が遅れてしまう．100 回 rekey 動作を行った場合，10 秒も rekey 動作開始時間に遅れが生じる．よって，rekey 動作にかかる時間を考慮に入れた rekey 動作開始時間の計算が必要となる．

8.2.5 sub key SA 作成

現在の実装では，LKH サーバから受信した全ての sub key で SA を張っている．これにより，鍵の更新時，LKH サーバが用いた SA を事前に LKH クライアントへと伝える必要が無くなる．しかし，デメリットとしては LKH クライアントが必要のない SA を張らなければならなくなる点が挙げられる．これに対し，IKE を用いる場合，事前に必要となる SA を LKH クライアントへと通知する事ができるので複数の SA を張る必要が無くなる．しかし，デメリットとして各 LKH クライアントが IKE によるネゴシエーションをしなければならなくなる．必要のない SA を張る動作と，IKE によるネゴシエーション動作，どちらが LKH サーバと LKH クライアントにかかる負担が少ないかを検証し，負担のかからない方で実装する必要がある．

謝辞

本研究を進めるにあたり，慶應義塾大学 大学院 政策・メディア研究科 特任准教授の朝枝仁博士には多大なご尽力をしていただきました．正直に言いますと朝枝博士のご協力がなければ卒業論文を書けなかったのではないかというほど，研究の着想の時点からご協力していただきました．実装から評価に至るまで，様々なご質問をさせていただきました．朝枝仁博士には感謝の言葉しかございません．本当に，ありがとうございます．

慶應義塾大学 大学院 政策・メディア研究科 後期博士課程の三島和宏氏，久松剛氏，松園和久氏，Achmad Basuki 氏，岡田耕司氏にも大変ご協力していただきました．三島和宏氏には，実装のとき，どのような関数が必要か迷って尋ねに行くと「これを調べればいい」という助言を何度も頂きました．また，卒業論文の赤入れもしていただきました．ありがとうございます．久松剛氏には，実検環境作成時などに実験機の起動エラーなどで自分がどうしようもなくなったときなど助けていただきました．自分一人では解決できなかったであろうエラーがたくさんありました．お忙しい中，ありがとうございました．松園和久氏には，いつも自分を気にかけていただきました．卒業論文の進み具合などを私に訊きにきて下さり，私が手間取っていると「こうしたらいい」などの助言をいただきました．また，身体の心配などもしていただきました．ありがとうございます．Achmad Basuki 氏も度々，卒業論文の進み具合などを気にかけてくださいました．ありがとうございます．岡田耕司氏には，mistral ブースに人がいないときに何度か質問にお答えして頂きました．ありがとうございます．

現在、KDDI 株式会社で働いていらっしゃる峯木厳氏には大変ご協力していただきました．ありがとうございます．休日にわざわざお手伝いに来ていただき，また，随時卒業論文の進み具合を気にしていただきました．手伝っていただいたテスト環境を結局のところ使わなかった事には謝罪の言葉しかございません．誠に，申し訳ありませんでした．現在、エリクソン・ジャパン株式会社で働いていらっしゃる内田陽豪氏にもたくさんの助言をいただきました．在学中に相談に乗っていただいたり，謝恩会のときも気にかけていただきました．ありがとうございます．慶應義塾大学 SFC 研究所 上席所員の小川晃通博士にも大変身体の事を気にかけていただきました．生協で買った野菜などをたくさんいただきました．ありがとうございます．慶應義塾大学学部生の小野山裕己氏，山中俊樹氏，吉原洋樹氏，中村尚生氏，澁田拓也氏，数井翔氏にも大変ご協力していただきました．小野山裕己氏，吉原洋樹氏とは共に励まし合う事で卒業論文を進めていく事ができました．ありがとうございます．山中俊樹氏，吉原洋樹氏，中村尚生氏，澁田拓也氏とは学食でご飯を食べながら漫画の話などをして，卒業論文で張りつめた気持ちを何度も緩和していただきました．ありがとうございます．数井翔氏には，笑顔で何度も和ませていただきました．ありがとうございます．慶應義塾大学環境情報学部　教授の村井純博士，中村修博

士，武田圭史博士，准教授の植原啓介博士，Rodney D.Van Meter 博士には，発表会でご質問，ご意見をいただきました．ご質問，ご意見は自分の研究内容をもう一度見つめるきっかけとなりました．大変，ありがとうございます．有限会社 LEFT の小川浩司氏にも，MIKEY の改善時には大変ご協力いただきました．ありがとうございます．

まだまだ様々な方にご迷惑，ご協力をいただいて本研究はすすめられてきました．ここに書かれていない方にもご協力していただいているのに申し訳ありません．本研究を進めていくうえで，ご協力いただいた全ての方々，友人，家族に深く感謝します．ありがとうございます．

以上を持って謝辞とさせていただきます．

参考文献

[1] ニールセン・オンライン（Nielsen Online）. http://www.netratings.co.jp/.

[2] Nielsen NetView. http://www.netratings.co.jp/solution/netview.html.

[3] ニコニコ生放送. http://live.nicovideo.jp/.

[4] Ustream. http://www.ustream.tv/.

[5] 革命×テレビ CH ５. http://www.ustream.tv/channel/KakumeiTV-ch05.

[6] U-NEXT. http://www.unext.jp/.

[7] ITPRO. http://itpro.nikkeibp.co.jp/article/COLUMN/20100104/342806/.

[8] Ustream Pay-Per-View. http://www.ustream.tv/ppv.

[9] Cisco WebEx Meeting Center. http://www.webex.co.jp/microsite/service/meeting.html.

[10] Skype. http://www.skype.com/intl/ja/home/.

[11] VIERA. http://panasonic.jp/viera/products/vt2/.

[12] R. Rivest, A. Shamir and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, 1978.

[13] E. Rescorla. Diffie-hellman key agreement method. *RFC 2631*, 1999.

[14] Web で学ぶ 情報処理概論 公開鍵暗号方式 (Public key encryption system). http://www.infonet.co.jp/ueyama/ip/glossary/public_key.html.

[15] セキュリティ用語事典. http://www.atmarkit.co.jp/aig/02security/bruteforceattack.html.

[16] 日立ソリューションズの情報セキュリティブログ. http://securityblog.jp/words/609.html.

[17] Debby M. Wallner, Eric J.Harder, and Ryan C. Agee. Key management for multicast: Issues and architectures. *RFC 2627*, 1999.

[18] Chung Kei Wong, Mohamed Gouda, and Simon S. Lam. Secure group communications using key graphs. *IEEE/ACM Transactions on Networking, vol. 8*, pages pp.16–30, 2000.

[19] Josep Pegueroles, Francisco Rico-Novella. Balanced batch lkh: New proposal, implementation and performance evaluation. *Eighth IEEE Symposium on Computers and Communications*, 2003.

[20] Xiaozhou Steve Li, Yang Richard Yang, Mohamed G. Gouda. Batch rekeying for secure group communications. *Proceedings of Tenth International World Wide Web Conference,*, 2001.

[21] D. McGrew, A. David, T. Alan, and A. Sherman. *Keyestablishment in large dynamic groups using one-way function trees.* TIS Report no.0755, TIS Labs at Network Associates, Inc., Glenwood, MD, 1998.

[22] H. Harney, U.Meth, A. Colegrove, and G. Gross. Gsakmp: Group association key management protocol. *RFC 4535*, 2006.

[23] M. Baugher, B. Weis, T. Hardjono, and H. Harney. The group domain of interpretation. *RFC 3547*, 2003.

[24] J. Arkko, E. Carrara, and F. Lindholm. Mikey: Multimedia internet keying. *RFC 3830*, 2004.

[25] Y. Sun, P. Liu, P. Kermani. An architecture and key management approach for maintaining privacy in location based group services. *Networking & Security Res*, 2006.

[26] R. Atkinson. Security architecture for the internet protocol. *RFC1825*, 1995.