修士論文

マルチキャスト暗号通信における LKH Key Tree の効率的な構成法

An Efficient Structure of LKH Key Tree on Secure Multicast Communications

東京電機大学 大学院 工学研究科
情報通信工学専攻 修士課程
08GMC07 貝沢亮介
研究指導教員 准教授 坂本直志

目次

1. はじめに

インターネットが普及するにつれ、様々な通信がインターネット上に載せられるようになった。近年特に発展してきているのはビデオや音声の通信である。通信速度の向上やコンピュータの処理性能の向上より、YouTube などのビデオサイトの発展や、IP 電話の普及などの技術も実用化されている。このジャンルにおいて、リアルタイムの映像配信などの 1 対多通信や、遠隔会議システムなどの多対多通信を使用する形態が存在する。これは現在でも Skype などでサービスが行われている。

ビデオや音声のようなリアルタイム性の強い通信は通常 RTP を使用する。これには UDP が用いられる。そのため、マルチキャストでも利用可能である。マルチキャスト通信は、同一の情報を一度に多くの受信者に対して送信を行う多対多通信においてネットワーク負荷の軽減などの利点がある。近年オンラインゲームも発展し、大規模なサービスが登場している。MMORPG などは多くのプレイヤーキャラクターが画面上に表示される。これらキャラクターの移動情報などは多数のユーザーにリアルタイムに伝わることが要求される。このようなアプリケーションにもマルチキャスト通信の利点がある。但し、インターネットではユニキャスト通信が殆どであり、ユーザサービスレベルでのマルチキャスト通信はまだ一般的では無い。

一方で、遠隔会議システムなどでは盗聴を防ぐのが前提となる場合がある。そのため、暗号通信を行う必要がある。IP 上で暗号通信を行う枠組みとして、IPsec という仕組みがある。IPsec を用いる上で暗号鍵を管理する必要がある。そのため、ユニキャストの暗号通信では IKE という鍵管理プロトコルを使用する。一方、マルチキャストで鍵を管理する仕組みとして、GSAKMP などの鍵管理プロトコルがある。GSAKMP は鍵管理プロトコルであるが、鍵管理手法は選択可能である。鍵管理を行うひとつの手法として LKH が提案されている。LKH では鍵の管理は階層的に行う。

ここで、鍵管理における、鍵更新に関する問題について簡単に説明する。はじめに、IP 上のビデオ放送が暗号化されていて、グループ G にだけ視聴が許されているとする。そして、放送を視聴するために共有秘密鍵 k_0 が配布されていたとする。この時、グループから一人のメンバー m が離脱することを考える。メンバー m が離脱後も暗号鍵 k_0 を使いつづけると m は k_0 を持っているので盗聴ができてしまう。そのため、メンバーが脱退する時には鍵を更新しなければならない。また参加に関しても、既知の鍵を新メンバーに渡すと、過去の放送を遡って視聴できてしまう可能性がある。そのため、メンバーの参加、離脱において鍵の更新が必要となる。但し、鍵更新において、グループ全員に対して個別に更新した鍵を配布する必要はない。LKH のように、あらかじめすべてのメンバーがの葉に配置されるような鍵管理木を作り、各ノードに鍵を配置するような手法を取ると、効率的に管理されるとされてきた。

さて、オンラインゲームなどでは、サービス利用時間などユーザーによって利用パターンに傾向が見られることが報告されている[27]。例えば、曜日別に利用時間のパターンが強く表れている。将来的にはモデル化され利用時間の予測などが出来ることが期待される。本研究ではこれを踏まえ、各メンバーのグループへの参加頻度などを参考にできるという前提で、より効率のよい鍵管理方法を提案する。これは、鍵の更新の頻度が高いグループをいくつか取り出したとき、それぞれのグループは完全二分木とするが、作成された完全二分木を線形リストで結ぶものである。この方式の方が従来の方式よりも効率的になるような、グループの選択の条件を導いた。

本論文では 2 章で IP でのマルチキャスト通信とセキュリティプロトコルについて説明する。3 章でマルチキャストグループでの鍵管理の課題について説明し、この課題に対しての先行研究につ

いて述べる。4 章では提案する鍵管理手法について述べ、5 章でこの手法の有効性を解析する。そして、6 章でまとめる。

2. IP マルチキャスト暗号通信について

2.1. IP マルチキャスト通信

現在、インターネットを含めた多くの通信に Internet Protocol(IP)が利用されている。しかし、その殆どはユニキャスト通信である。ここでは現在最も普及している IP である IP Version 4(IPv4)でのマルチキャスト通信について述べる。

IPv4 では宛先や送信元をしめす 32bit のアドレス空間が利用される。IP アドレスは 8bit ごとに『. 』(ピリオド)で区切り、それぞれの区間で 0 から 255 までの 10 進法で表記される。つまり IP アドレスは、

0.0.0.0 から 255.255.255.255

まで存在する。このうち、

224.0.0.0 から 239.255.255.255

までの IP アドレスがマルチキャストアドレスとして割り当てられている[3]。

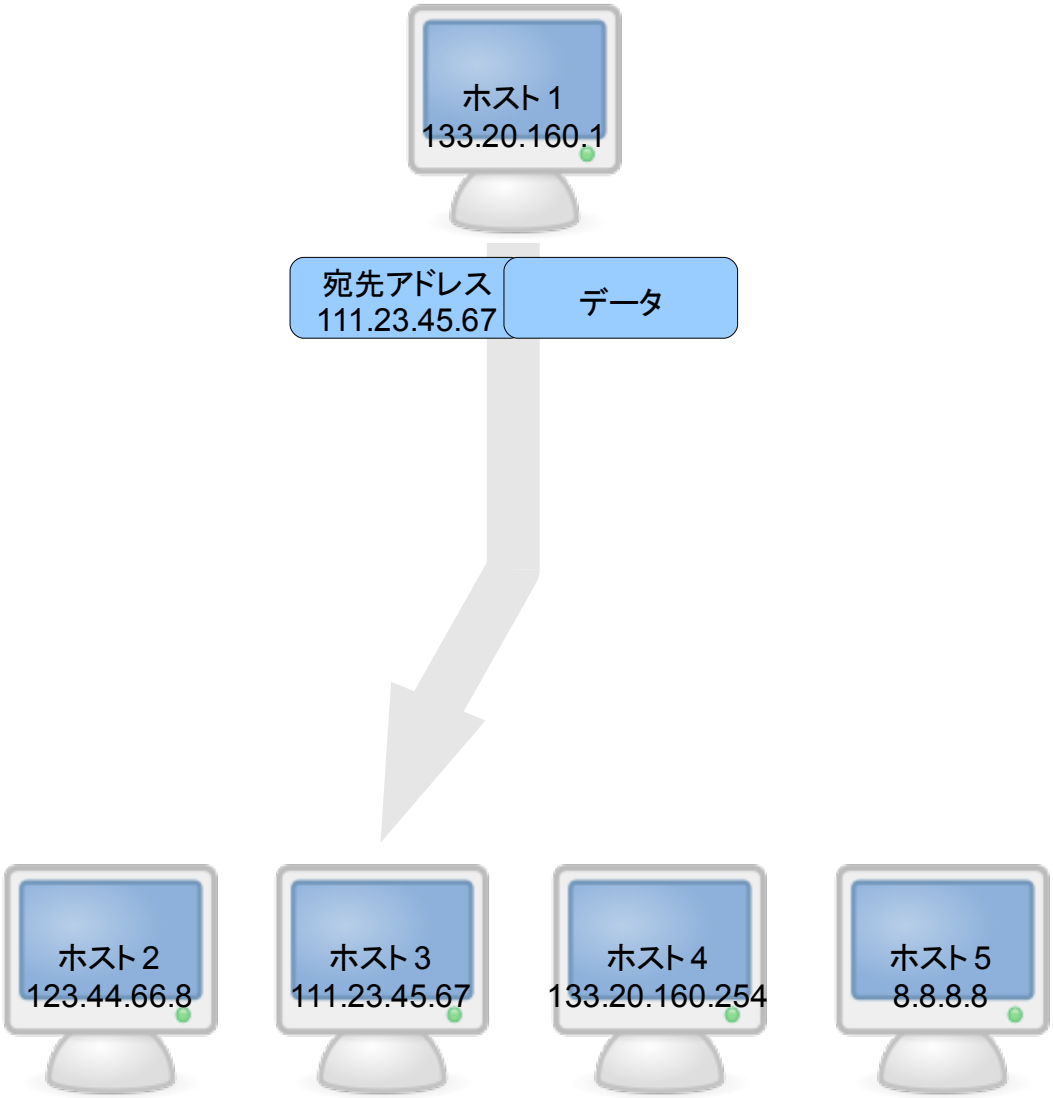

図 1:ユニキャスト通信の概要

ユニキャスト通信では宛先アドレスを指定したパケットを送信すると、パケットはそのアドレスに割り当てられているホストに到着する。このとき、1 つのアドレスに割り当てられているホストは 1 つであり、複数のホストが同一のアドレスを持つことは出来ない。ユニキャスト通信では互いのホストのアドレスを、宛先アドレスに指定してパケットを送信することで、1 対 1 の通信を行う。例えば図 1 のように、各ホストには以下のようなアドレスが割り当てられているとする。

ホスト 1: 133.20.160.1

ホスト 2: 123.44.66.8

ホスト 3: 111.23.45.67

ホスト 4: 133.20.160.254

ホスト 5: 8.8.8.8

このとき、ホスト 1 がホスト 3 にデーターを送信したい場合、宛先アドレスに『111.23.45.67』を指定した IP パケットを送信すれば、ホスト 3 に到着する。同じデータを複数の相手に送信したい場合、それぞれの宛先を指定したパケットを相手の数だけ送信する必要がある。

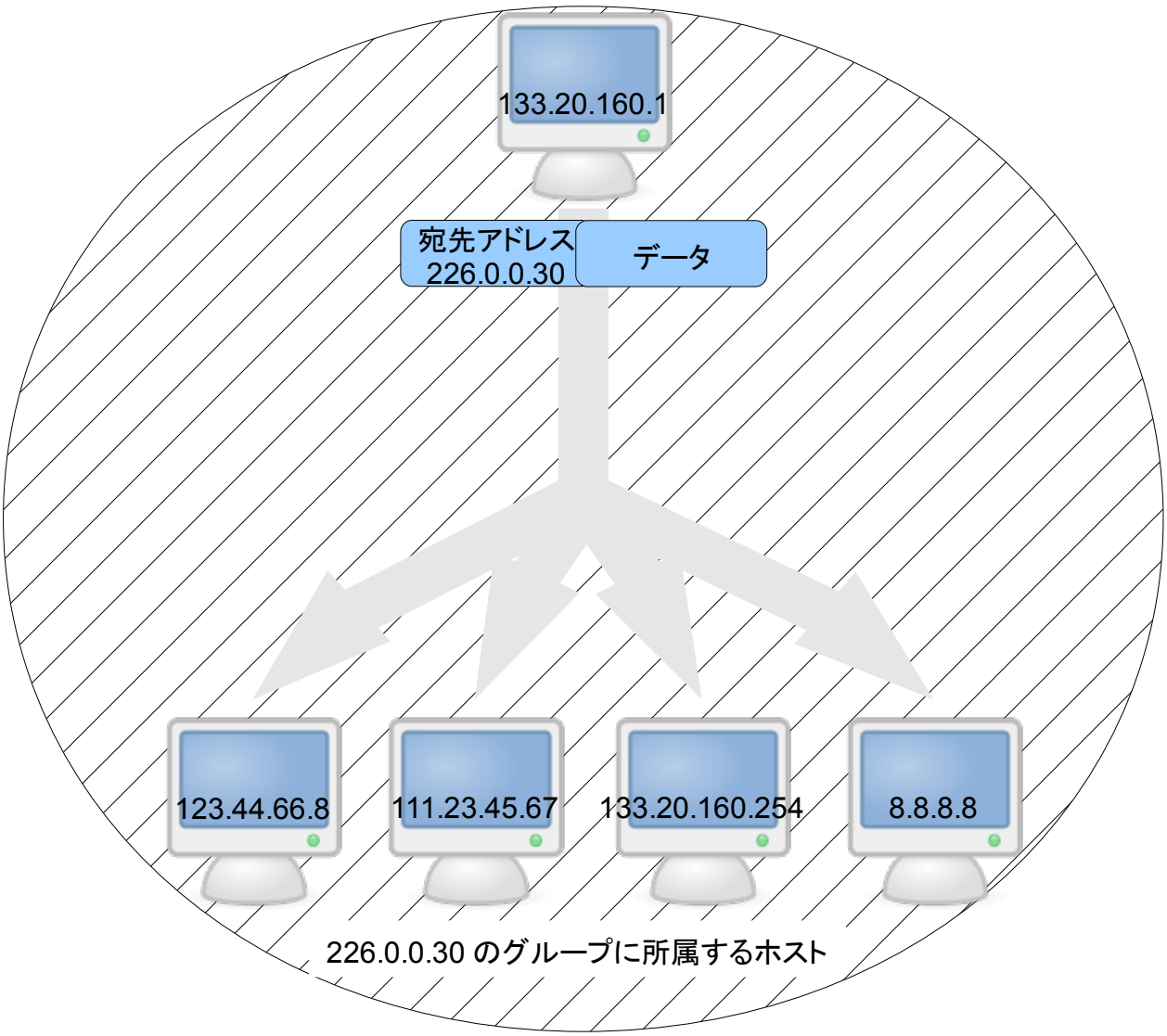

図 2: マルチキャスト通信の概要

マルチキャスト通信[1][2]では、1 つのマルチキャストアドレスは 1 つのグループに割り当てられる。このとき宛先アドレスにマルチキャストアドレスを指定して送信すると、そのアドレスのグループに所属する全ホストにパケットは到達する。例えば、図 2 の各ホストは図 1 と同様にそれぞれユニキャストアドレスが割り当てられている。さらに、ホスト 1 からホスト 5 はマルチキャストアドレス『226.0.0.30』のグループに所属しているとする。このとき、ホスト 1 がグループに所属する全ホストにデーターを送信する場合、宛先アドレスに『226.0.0.30』を指定したパケットを送信すれば、グループ内の全てのホストに到着する。このように、複数のホストに同じデータを送信する場合、送信元のホストは単一のパケットを送信するだけでよいので、ホストの負荷抑制が期待できる。

2.2. IPsec

インターネットでは通常、通信は全て平文で行われる。このため第三者による通信の改ざんや盗聴が行われる可能性がある。そこで、Security Architecture for Internet Protocol (IPsec)[6]という暗号技術を用いて、IP 通信の改ざん防止や盗聴防止機能を提供するプロトコルが定められている。これ自体は IP 通信をセキュアに行うものなので、コネクションレス型である。さて、セキュリティプロトコルによって保護された擬似的な通信路は Security Association (SA)と呼ばれ、特に IPsec

による SA は IPsec SA と呼ぶ。IPsec SA は単方向の通信路である。そのため IPsec を利用した双方向通信には逆方向の IPsec SA がもう一つ必要である。

IPsec は主に、

- セキュリティポリシー
- 暗号通信プロトコル
- 鍵管理プロトコル

の 3 つの要素から構成される。暗号通信プロトコルと鍵管理プロトコルでは、それぞれで複数のプロトコルが存在し、必要な機能に応じて利用するプロトコルの選択が可能である。

2.2.1. セキュリティポリシー

セキュリティポリシーはホストに出入りする IP パケットに対して、IPsec を適用するかどうかが判定される。セキュリティポリシーが記述されているセキュリティポリシーデータベース(SPD)を参照して、出入りする IP パケットの宛先アドレス、送信元アドレスなどのヘッダ情報から、その IP パケットの扱いが決定される。扱いは、

- パケットの破棄(DISCARD)
- IPsec を適用しない(BYPASS)
- IPsec を適用する(PROTECT)

の3つである。セキュリティポリシーによって PROTECT として扱われる場合、使用する暗号通信プロトコルや暗号化アルゴリズムも SPD に記述される。

2.2.2. 暗号通信プロトコル

セキュリティポリシーによって PROTECT として扱われた IP パケットは SPD の記述に従って暗号通信プロトコルが利用される。暗号通信プロトコルはパケットへの暗号化技術の適用を行う。暗号通信プロトコルは、

- Authentication Header (AH)[7]
- Encapsulating Security Payload (ESP)[8]

の2種類存在する。

AH は IP パケットのヘッダ部分を含めた改ざん防止機能を提供する。しかしパケットの暗号化は行わないので、盗聴防止機能は提供しない。

ESP は IP パケットのデータ部分の改ざん防止や盗聴防止機能を提供する。ただし IP パケットのヘッダ部分には機能を提供しないので、送信元の IP アドレスを含めたヘッダ部分の改ざんは検出しない。AH と ESP は一方のみの利用だけでなく両方同時の利用も可能である。よって同時に利用すればデータ部分の盗聴防止と、データ部分とヘッダ部分の改ざん防止が可能である。改ざん防止や盗聴防止機能を提供するのに使われる暗号鍵は共有秘密鍵方式のものが利用される。

2.2.3. 鍵管理プロトコル

IPsec で利用される暗号通信プロトコルの用いる暗号鍵は共有秘密鍵である。そのため暗号通信を行うホストは、事前に暗号鍵を共有する必要がある。また、暗号鍵は時間をかければ解読される可能性がある。そのため一定時間利用した後に新しい暗号鍵に更新して、通信の安全性を維持する必要がある。この暗号鍵の更新を Rekey と呼ぶ。暗号鍵の事前の共有や Rekey を手動で行うことは、運用する上で現実的ではない。そのため、暗号鍵の共有と Rekey を行うのに鍵管理プロトコルを使用する。

2.2.3.1. IKE

Internet Key Exchange (IKE)[11]は IPsec を利用した暗号通信を行う2者間で鍵管理を行うプロトコルである。Internet Security Association and Key Management Protocol (ISAKMP)[9]と Oakley[10]という2つのプロトコルを組み合わせて構成されている。現在では後継である IKEv2[12]の仕様が公開されている。ここでは IKEv2 の仕組みを説明する。

IKEv2 では、

- IKE_SA
- CHILD_SA

の 2 種類の SA を構築する。IKE_SA は IKEv2 がやりとりをするメッセージを保護するための SA である。IKE_SA は IPsec SA とは異なり双方向通信路である。IKE_SA の保護に利用される共有秘密鍵は Diffie-Hellman 鍵共有アルゴリズム[28]を用いて生成される。この IKE_SA を用いてそれぞれの方向に向いた 2 本の CHILD_SA を生成する。この CHILD_SA は IPsec SA である。CHILD_SA の保護に利用される共有秘密鍵は IKE_SA を用いて交換される。また、CHILD_SA の Rekey も IKE_SA を用いて行われる。

IKE はユニキャスト暗号通信における鍵管理を想定して設計されている。そのため IPsec を利用したマルチキャスト通信を行うには、マルチキャストに対応した鍵管理プロトコルが必要である。

2.2.3.2. GSAKMP

Group Secure Association Key Management Protocol (GSAKMP)[19]は暗号通信で保護されたグループを構築し、管理するプロトコルである。GSAKMP は Group Key Management Protocol (GKMP)[14][15]を拡張したプロトコルである。どちらのプロトコルもマルチキャストグループに対応した Group Security Association (GSA)の概念を元に構成されている。

GSA は、1 つのマルチキャストグループに参加しているホストあるいはユーザーの集合である。このホストあるいはユーザーのことをメンバーと呼び、あるマルチキャストグループに参加しているメンバーをグループメンバーと呼ぶ。GSA に参加するホストには、

- Group Member (GM)
- Group Controller Key Server (GC/KS)

が存在する。GM はその名の通りグループメンバーのことである。GC/KS はグループの管理とグループで用いられる暗号鍵の管理を行うサーバーである。暗号鍵の管理として、暗号鍵の生成や配布、そして Rekey のための構造の構築、管理を行う。

GSA は、

- レジストレーション SA
- Rekey SA
- データ SA

の 3 つの SA を含む。レジストレーション SA はメンバーがグループへの参加・離脱をする際に GC/KS と行うユニキャスト通信を保護する SA である。レジストレーション SA は各 GM と GC/KS 間ごとに生成される。Rekey SA は Rekey を行う際の GC/KS から GM へのマルチキャスト通信を保護する SA である。データ SA は GM 同士でやりとりされるデータのマルチキャスト通信を保護する SA である。IPsec ではこのデータ SA が IPsec SA である。

GSAKMP では主に参加、離脱、Rekey の 3 つの動作でメンバの管理を行う。

2.2.3.2.1. 参加

メンバーがグループに加わる時の動作である。メンバーは参加したいグループを管理している GC/KS に対し、参加要求との通知とレジストレーション SA の構築を試みる。レジストレーション SA の保護に利用される共有秘密鍵は Diffie-Hellman 鍵共有アルゴリズムを用いて生成される。レジストレーション SA が構築されたら、グループで用いられている Rekey SA とデータ SA に加わるための共有秘密鍵を交換し、グループへの参加が可能となる。

2.2.3.2.2. 離脱

メンバーがグループから離れるときの動作である。メンバーが自発的に離脱する場合、明示的に離脱要求をすると、そのメンバと GC/KS とのレジストレーション SA は消失する。また、Rekey SA とデータ SA に使われている共有秘密鍵の Rekey を行う。この Rekey は離脱するメンバーには行われないため、これ以降のグループの SA 上の全ての情報にアクセスできなくなる。この離脱の動作は、メンバーからの要求以外にも、メンバーによる共有秘密鍵の漏洩などによる安全性の低下からマルチキャストグループを保護する必要が生じた際は、GC/KS 側から行われる。

2.2.3.2.3. Rekey

各 SA で使われている共有秘密鍵を新しいものに更新し、それを配布する動作である。これは Rekey SA 上で行われる。同じ共有秘密鍵を長時間使い続けることによる脆弱性の増加を防ぐため、メンバーの離脱などの後以外にも、定期的に行われる。Rekey は安全性を確保しつつ、効率的に鍵の配布を行うことが望まれる。GSAKMP では LKH という鍵管理構造を用いることで、この要求に応えている。LKH については後ほど述べる。

3. グループ鍵管理の課題

IPsec などのマルチキャスト暗号通信が可能なセキュリティプロトコルは、マルチキャスト通信の暗号化に共有秘密鍵を用いる。この共有秘密鍵を Traffic Encryption Key (TEK)と呼ぶ。TEK をメンバーに配布する際に、改ざんや盗聴から保護するために用いる暗号鍵を Key Encryption Key (KEK)とよぶ。Rekey では一定時間使用した TEK や、メンバーの参加・離脱によって更新が必要になった TEK や KEK に変わる新しい鍵を生成し、他の安全な KEK を用いてメンバーに配布する作業を行う。

TEK は、マルチキャストグループに所属する全てのメンバーに共有される必要がある。ここで、グループへ新たなメンバーが参加する場合を考える。今まで使っていた TEK を新メンバーと共有してしまうと、新メンバーが参加する以前の通信内容を読み取ることが出来てしまう。また、グループからメンバーが離脱する場合、今まで使っていた TEK は旧メンバーにも共有されている。このため、旧メンバーが離脱した以後の通信内容を読み取ることが出来てしまう。これらを防ぐためにはメンバーの参加や離脱が起きた直後に TEK を更新し、速やかに配布され、現在のグループメンバーのみに共有される必要がある。

3.1. 共有するメンバーを限定した Rekey

Rekey 時に新しい TEK を現在のメンバーのみに配布し共有するには、鍵サーバーがメンバーに TEK を配布する際に利用する KEK がメンバーごとに異なっていればよい。離脱したメンバー間で共有していた KEK を用いなければ、旧メンバーに更新した TEK を共有されることはない。しかしこの手法では、現在のグループメンバーがn人だとすると、その全員に更新した TEK を配布するには、TEK をn回暗号化する必要がある。このためメンバー数が大きくなると鍵サーバーの暗号処理に必要なコストも大きくなる。また、KEK で暗号化した TEK の送信もn回行う必要がある。このためメンバー数が大きくなると TEK が全メンバーに共有されるまでにかかる時間も大きくなる。

3.2. 速やかな配布が可能な Rekey

Rekey 時に新しい TEK を速やかに配布し共有するには、単一の KEK を用いてマルチキャスト通信で TEK を配布することである。このとき KEK は全てのメンバーで同一のものを用いるとマルチキャスト通信を利用した鍵配布の効率は最もよくなる。ユニキャスト通信に比べマルチキャスト通信は、同じ情報を大人数に送信する必要があるときに効果的である。しかし、この手法では KEK は離脱したメンバーも含めた全てのメンバーで同一なため、共有するメンバーを限定することは難しい。

3.3. LKH

先行研究の一つに Logical Key Hierarchy（LKH)[16][21]がある。これは配布するユーザーを任意に指定できる能力と、マルチキャスト通信を用いた鍵配布の能力を両立した手法である。LKH は図 3 のような木構造を利用して鍵を管理する。全てのノードには KEK(k_i)が配置されている。これらの KEK は全て異なる鍵である。グループに含まれるメンバー(u_j)は葉に配置され、各メンバーは自分の配置されている葉から根までのパス上にある全ての鍵を鍵サーバーと共有する。なお、根に配置された鍵は全てのメンバーで共有される。また葉に配置された鍵は、葉に配置

されているメンバーと鍵サーバーのみで共有される。

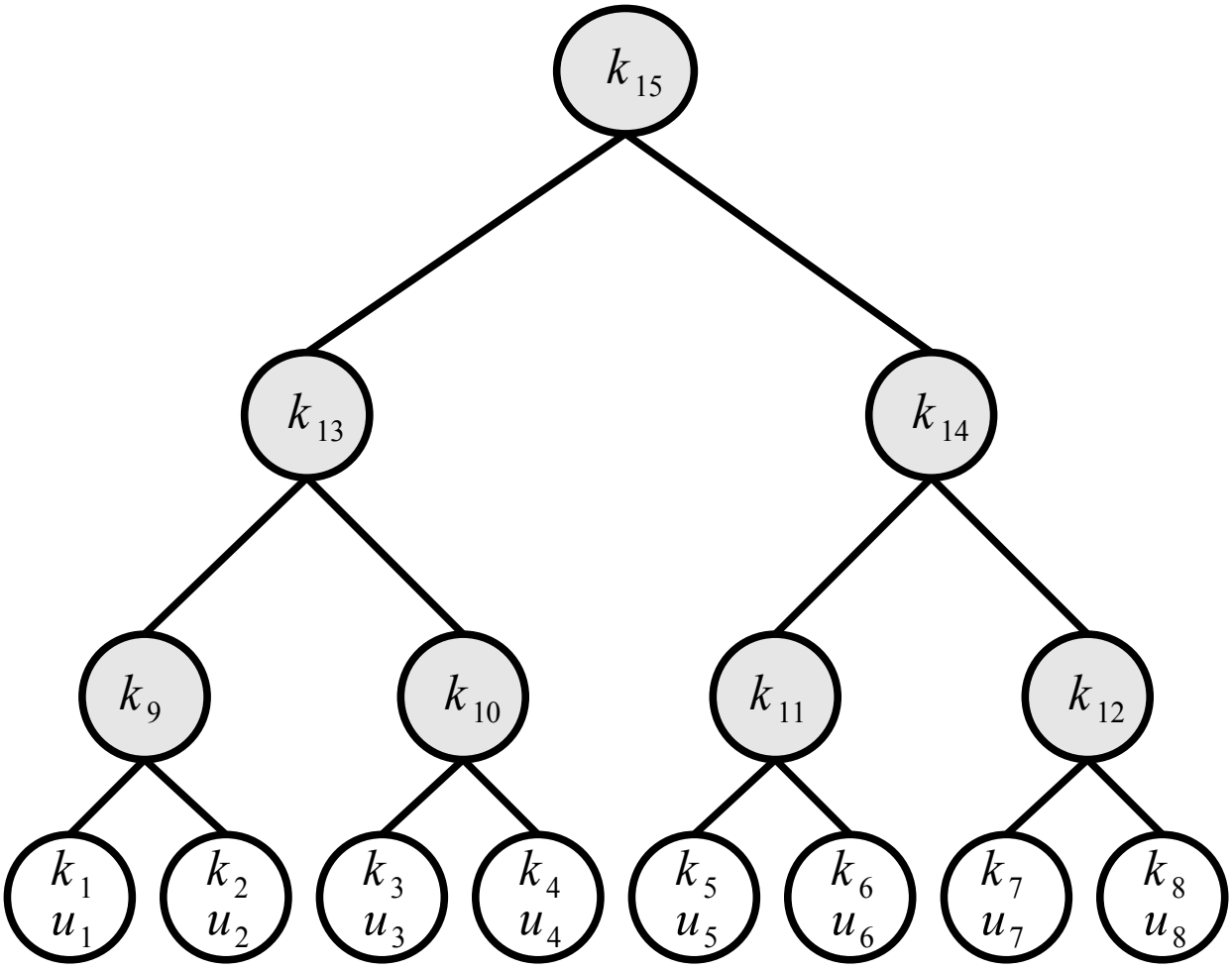

図 3:LKH の Key Tree

例えば、図 3 で最も左の葉に配置されたメンバー u_1 は k_1 、 k_9 、 k_{13} 、 k_{15} の KEK を所有している。このように鍵を管理する木を Key Tree と呼ぶ。図 3 の Key Tree は 2 分木で構成されているが、多分木でも構成できる。簡単のため、特に断りがなければ Key Tree は 2 分木であるとする。

図 3 の Key Tree で管理されているマルチキャストグループで Rekey が発生したとき、Key Tree がどのような動作で Rekey を行うか説明する。まず、メンバーの参加や離脱がなく、一定の間隔で行われる Rekey での動作を説明する。この場合、特定のメンバーを除外した Rekey をおこなう必要はない。よって、鍵 k_{15} を用いて更新した TEK を暗号化し、メンバーにマルチキャストする。 k_{15} は全てのメンバーに共有されているので、メンバーは更新された TEK を取り出すことが出来る。

次に、メンバーが参加や離脱する場合の Rekey での動作を説明する。メンバーが離脱する場合、TEK の更新が必要なのと同時に、メンバーが所有していた KEK も更新する必要がある。例えば、メンバー u_1 が離脱するとき破棄される KEK は k_1 、 k_9 、 k_{13} 、 k_{15} である。このときメンバー u_1 の配置されているノードは必要なくなるため、 k_1 とともに破棄される。一方、残りの

k_9 、 k_{13} 、 k_{15} はそれぞれ k'_9 、 k'_{13} 、 k'_{15} に更新され、更新前と同一のノードに配置される。更新された鍵は複数の KEK を用いて複数回送信される。 u_1 が所有していなくて、 u_5 、 u_6 、 u_7 、 u_8 に共有されているのは k_{14} である。この KEK を用いて k'_{15} を暗号化しメンバーにマルチキャストをすると、 u_5 、 u_6 、 u_7 、 u_8 の 4 メンバーは k'_{15} を取り出すことが出来る。また、 u_1 が所有していなくて、 u_3 、 u_4 に共有されているのは k_{10} である。よって u_3 、 u_4 にたいしては、この KEK を用いて k'_{13} 、 k'_{15} を暗号化しマルチキャストする。 u_1 の兄弟である u_2 にたいしては、 u_2 のみが所有する k_2 を用いて k'_9 、 k'_{13} 、 k'_{15} が暗号化され送信される。メンバーが参加する場合も、メンバーの配置されるノードから根までのパス上にある全ての鍵が更新され、離脱と同じ手法で配布される。

このように Rekey するときに Key Tree は動作する。Key Tree の更新の必要な鍵数は高々 Key Tree の高さになる。Key Tree が完全二分木ならば、鍵更新数は $\lceil \log_2 n \rceil$ である。

用いる KEK のある深さによって暗号化して送信する鍵の本数が増える。先述の例では、用いる KEK のうち最も浅い位置に配置されている k_{14} は 1 つの鍵を暗号化する。一つ深い階層の k_{10} は 2 つの鍵を暗号化する。さらに 1 つ深い階層の k_2 は 3 つの鍵を暗号化する。1 つ階層が深くなるに従って暗号化する鍵の本数が 1 つ増える。従って、Key Tree の更新の必要な通信コストは高々 Key Tree の高さの 2 倍になる。Key Tree が完全二分木ならば、通信コストは $2\lceil \log_2 n \rceil$ である。

3.4. Batch LKH

LKH は 3.1、3.2 の Rekey 方法よりも効率がよい。しかし、それでもマルチキャストグループが大きくなると、メンバーの参加や離脱も頻繁に発生すると考えられる。これにより Rekey の頻度も多くなり、鍵サーバーの高負荷になるなどの点で課題が残る。

有料放送などのサービスを考えたとき、1 契約者の視聴契約時間は最小で 1 番組単位で設定することはよく行われている。このように、一定の期間ごとにメンバーの参加や離脱の動作をすればよいケースが考えられる。Batch LKH[22]はこのようなサービスのケースを踏まえ、LKH の Key Tree を用いて鍵管理をおこなう。つまり、期間内にあった参加や離脱要求に対する Rekey 作業を一度に処理する。これにより、例えば参加や離脱の度に必ず根の KEK が更新されるので、2 つ以上の更新をまとめることで、更新を 1 回にできるなど、参加や離脱要求毎に鍵更新をおこなうよりも効率のよい鍵管理を行うことができる。ただし、参加や離脱のタイミングにより、迅速な Rekey が求められるようなサービスでは、LKH の方が適する。

3.5. Balanced Batch LKH

LKH や Batch LKH は Key Tree を用いて鍵管理をおこなう。Rekey 時に必要な鍵更新数は Key Tree の深さに影響される。よって LKH での Key Tree は出来るだけ浅く、平衡状態であることによって最も効率のよい Rekey が可能であると考えられている。しかし、LKH はメンバーの参加や離脱を繰り返すことによって、不平衡な Key Tree になってしまう場合がある。Balanced Batch LKH[23]は、不平衡化を防ぐため、Batch LKH による Rekey のタイミングで Key Tree の再構成をおこない、Key Tree を可能な限り平衡に保つ手法である。

4. 提案手法

LKH では全てのノードに暗号鍵を配置した Key Tree の葉にグループメンバーを配置する。これにより、メンバーの参加や離脱時に発生する Rekey 作業で効率的な鍵配布が可能である。Key Tree が平衡な状態であるとき、グループメンバーの総数を L_0 とするとメンバーの参加や離脱毎に発生する鍵更新数は $\lceil \log_2 L_0 \rceil$ である。これは全てのメンバーを平等に扱う場合には最も効率のよい値であると考えられる。

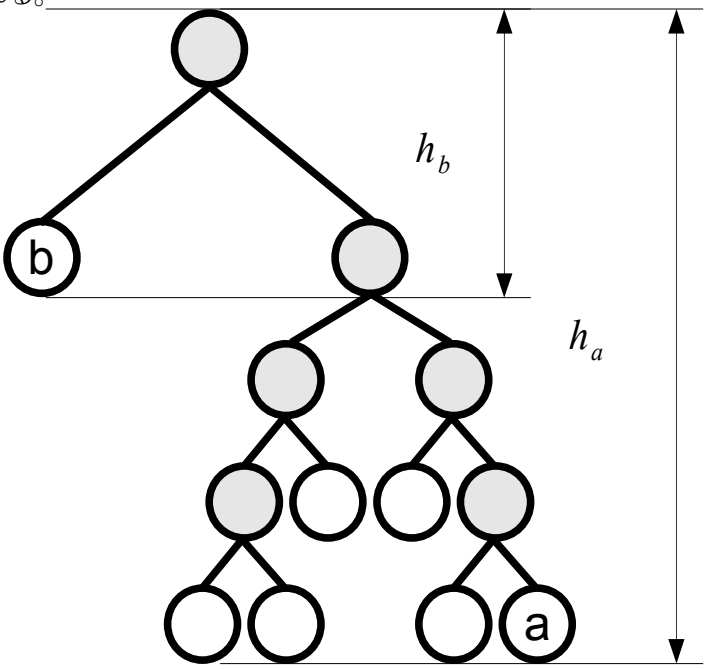

図 4:不平衡な Key Tree とノードの深さ

ここで、図 4 のように平衡でない Key Tree があるとケースと考える。このとき、メンバー a が離脱する場合、発生する鍵更新数は h_a とする。これは Key Tree が平衡な状態であるときと比べて大きくなっている。しかしメンバー b が離脱する場合、発生する鍵更新数は h_b である。これは平衡な状態であるときと比べて小さくなっている。もし、他のグループメンバーよりメンバー b が先に離脱するとわかっているならば、この例のように意図的にメンバー b を Key Tree の浅い部分へ配置すれば、鍵更新数を少なくできると考えられる。

本研究ではこの考えを元に、グループメンバーの離脱する時間があらかじめわかっていると仮定したときに、効率のよい鍵管理が出来る鍵管理構造を提案する。

4.1. 提案手法の構造

4.1.1.参加時間別 Key Tree への分割

各メンバーのグループへの平均参加時間がわかっているものとする。このときメンバーを平均参加時間の長さ別に複数のグループに分割する。そして分割されたグループを 1 つ、または複数含んだ Key Tree を構築する。各 Key Tree の全てのノードに暗号鍵が配置され、葉には各メンバーが配置される。これは LKH と同様の Key Tree 構造である。

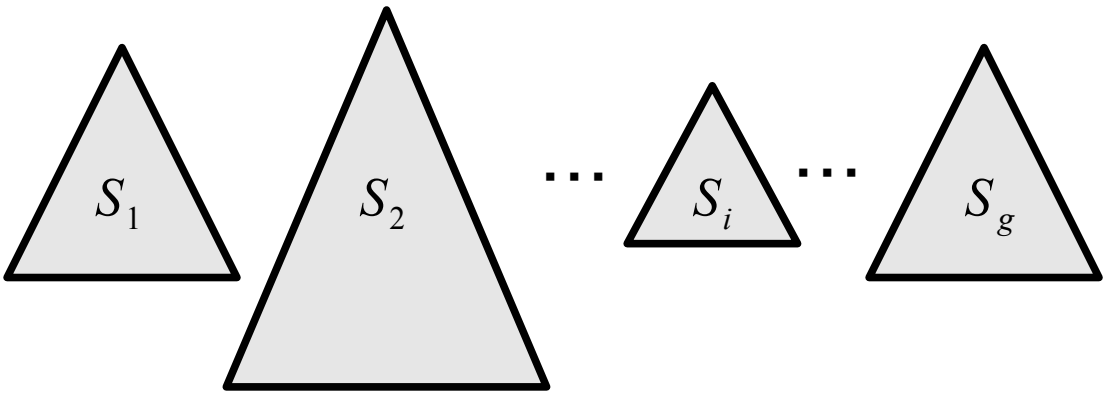

図 5:複数の分割された Key Tree

図 5 のように、そして分割されたグループを 1 つ、または複数含んだ Key Tree が g 個できたとすると、Key Tree $S_i(1 \le i \le g)$ が構築される。各 Key Tree に所属するメンバーの人数は各 Key Tree で異なる。

4.1.2. 線形リストによる Key Tree の管理

図 5 のように S_1 から S_g までの複数の Key Tree が構築されている。これらの Key Tree から一つの Key Tree を構築する。複数の Key Tree を束ねるのに以下の方法を用いる。

まず、新しく根となるノード l_{g-1} を生成する。このノードの右の子孫は S_g で、左の子孫は S_{g-1} となる。次に、今構築したノード l_{g-1} が根の Key Tree と S_{g-2} を束ねる。そのために新しく根となるノード l_{g-2} を生成する。そしてこのノードの右の子孫は l_{g-1} が根の Key Tree で、左の子孫は S_{g-2} となる。これを全ての Key Tree が 1 つになるまで繰り返す。すると図 6 のように各 Key Tree が線形リストで結ばれた 1 つの Key Tree になる。

分割された Key Tree は全て束ねられ、ノード l_1 を根とする 1 つの Key Tree が構築される。LKH と同様に、全てのノードには暗号鍵が配置され、メンバーは葉に配置される。また、各メンバーは自らの葉から根 l_1 までのパスに存在する全てのノードにある暗号鍵を所有している。

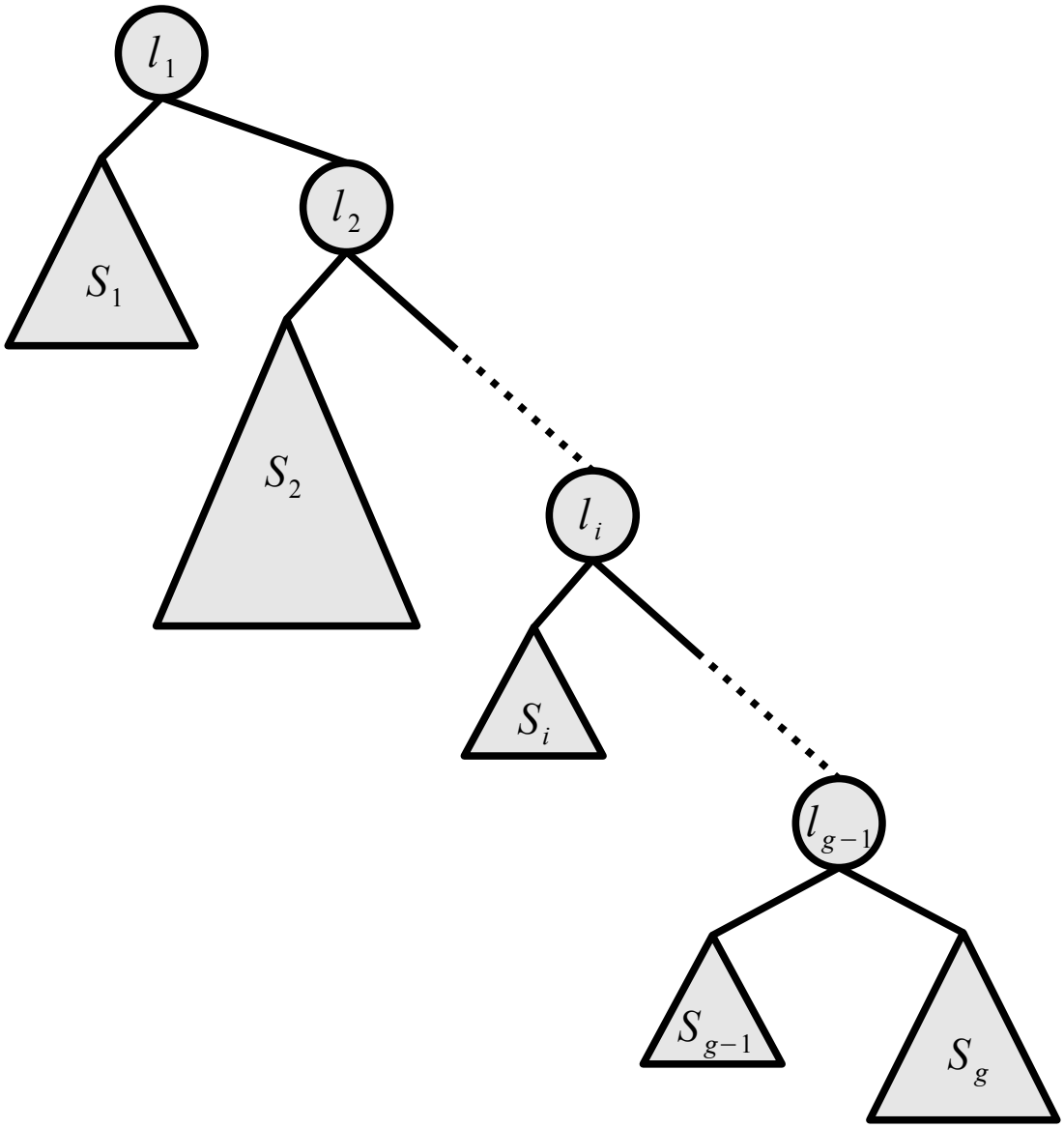

図 6:提案手法の Key Tree

4.2. 提案手法の検討

参加や離脱による鍵更新数は、参加や離脱を起こしたメンバーのいる深さに影響する。よって、図 6 のような Key Tree を構築すると、浅い位置に配置された Key Tree は 1 回 Rekey による鍵更新数が少なくなる。また、深い位置に配置された Key Tree は 1 回 Rekey による鍵更新数が大きくなる。このとき、鍵更新の要因である参加や離脱を頻繁に起こすユーザーのグループを浅い位置に配置し、参加や離脱を頻度が小さいユーザーのグループを深い位置に配置する。すると頻繁に鍵更新が必要なグループに必要な鍵更新数は小さくなる。その代わり、頻度が小さいグループによって Rekey が発生した場合の鍵更新数は大きくなる。

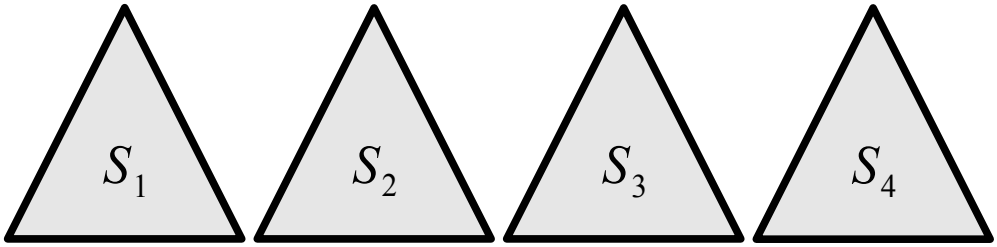

図 7:メンバー数が同一の複数の Key Tree

ここで、図 7 のような分割された Key Tree を考える。各 Key Tree に所属するメンバー数はいずれも L であり平衡状態であるとする。このとき、各 Key Tree の鍵更新数は $\lceil \log_2 L \rceil$ である。図 8 はこれらの Key Tree を線形リストによって束ねた Key Tree である。この状態での鍵更新数は、各 Key Tree の鍵更新数である $\lceil \log_2 L \rceil$ に加え、線形リストによって配置された深さ分の鍵更新数が必要である。例えば、階層 1 に配置されている S_1 に所属するメンバーにより鍵更新が必要になった場合、鍵更新数は $\lceil \log_2 L \rceil + 1$ である。同様に、階層 4 に配置されている S_4 に所属するメンバーにより鍵更新が必要になった場合、鍵更新数は $\lceil \log_2 L \rceil + 4$ である。この 4 つの Key Tree に含まれるユーザー数と同一のユーザー数を含む Key Tree を考えると鍵更新数は $\lceil \log_2 4L \rceil$ である。よって効率を改善するには、図 8 の Key Tree 全体の平均鍵更新数が $\lceil \log_2 4L \rceil$ よりも小さくなればよい。

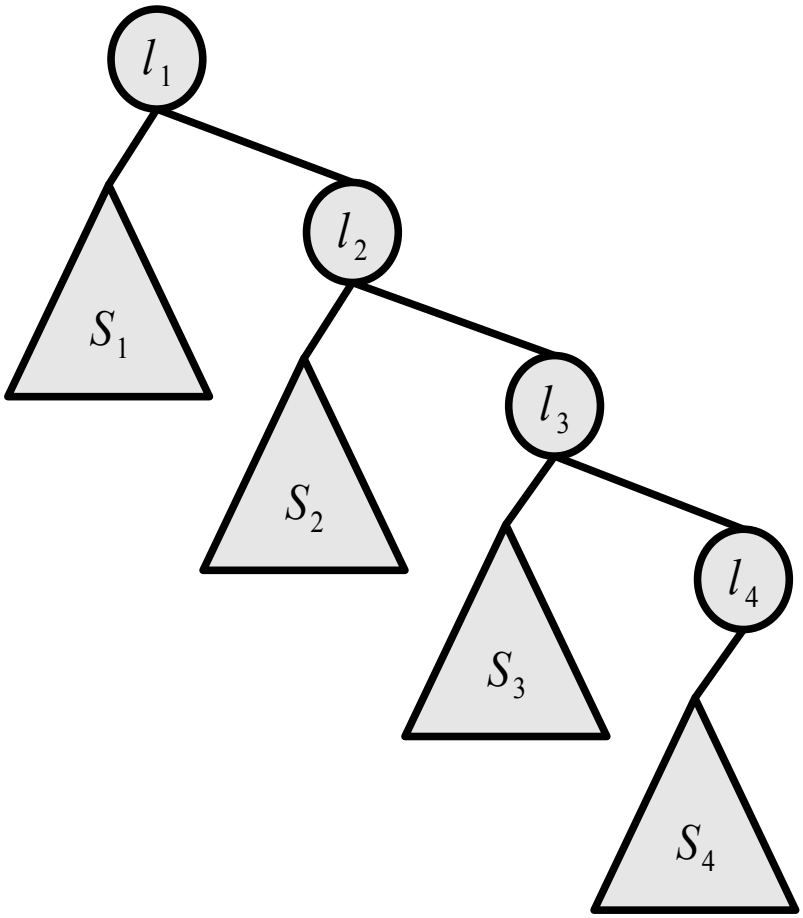

図 8:提案手法の Key Tree

4.2.1. 各 Key Tree のユーザー数の上限

提案手法の Key Tree を構築することで平均的な鍵更新数が小さくなることが期待される。しか

し、ユーザーの参加・離脱時に更新の必要な鍵数は、そのユーザーのノードの深さによって決まる。また分割された Key Tree が平衡状態であるとすると、その Key Tree に所属するユーザー数が増加すると Key Tree の深さは増加する。そのため更新頻度の大きい Key Tree であっても大人数が所属していれば、Key Tree を浅い位置に配置しても Key Tree 自体の深さによって効率を相殺してしまうことが考えられる。

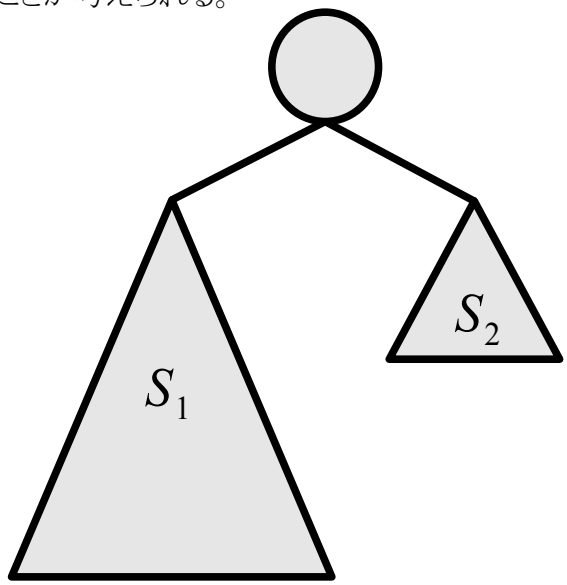

図 9:大きさの異なる Key Tree の統合

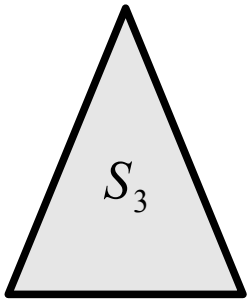

図 10:図 7 と同数のメンバを含む Key Tree

たとえば、図 9 のような Key Tree を考える。ここで、S_1 は S_2 より更新頻度が十分に大きい Key Tree であるとする。また S_1 は S_2 より所属するユーザー数が多く Key Tree が深いとする。一方、図 10 の S_3 は S_1 と S_2 をあわせたユーザー数が所属する。また S_3 は平衡状態である。Key Tree を構築したときに最も深さが小さくなるのは平衡状態の時である。図 9 の Key Tree を全体から見れば平衡ではない。このため、S_1 の深さは S_3 より大きい。S_1 は S_2 より更新頻度が十分に大きいため頻繁に更新すべき鍵が発生する。さらに S_1 の深さは S_3 より大きいので一度の参加や離脱によって更新すべき鍵数は平衡状態の S_3 より多くなってしまう。このため図 9 と図 10 の Key Tree で発生する参加や離脱の総数が単位時間あたりで同一の場合、従来方式より効率が悪化してしまうと考えられる。

このことから、浅い位置に配置する利点を生かせる Key Tree の人数を知る必要がある。

5. 解析

ここでは主に提案手法の Key Tree を構築するとき、分割された Key Tree に含まれるメンバー数をどう定めるべきかについて解析を行う。各グループ G_i において、平均参加者数は L_i 、平均参加時間は W_i であり、定常状態であるとする。この時、リトルの公式により、

$$\text{到着率 } \lambda_i = \frac{L_i}{W_i}$$

が求まる。グループ G_i のメンバーで構成される Key Tree S_i を構築することを考える。このときグループ G_i のメンバは平等に扱われる。 S_i は常に平衡状態を維持すると仮定すると、単位時間内の平均鍵交換数は、

$$2\lambda_i \lceil \log L_i \rceil$$

になる。ここで、各グループに対して鍵交換生起確率 p_i を、

$$p_i = \frac{2\lambda_i}{\sum 2\lambda_j}$$

と定義する。

まず、 g 個のグループを提案手法で分割することなく、従来の LKH と同様に 1 つの Key Tree S_0 で管理することを考える。このとき、 S_0 は平衡状態であるとすると、一回の鍵交換に対する平均鍵交換数 K_0 は

$$K_0 = \lceil \log \sum_{i=1}^{n} L_i \rceil \qquad (1)$$

となる。このとき、 $n=1$ であれば、従来の LKH 方式になる。また 3 章でも触れたように、平均通信コストは、 $2K_0$ となる。

5.1. Key tree の分割

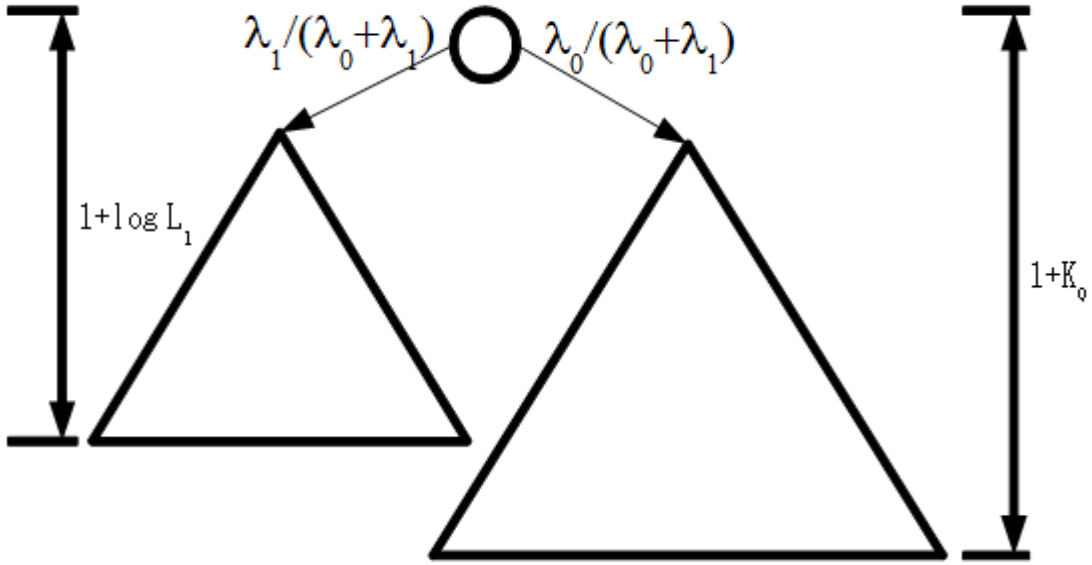

図 11:木の分割

平均鍵交換数が K_0 の Key Tree S_0 から、単位時間あたりの鍵交換数 λ_1 で人数 L_1 人のグループ G_1 を取り出して Key Tree S_1 を作り、図 11 のように Key Tree を分割して二分木として接続することを考える。このとき S_0 および S_1 は常に平衡状態を維持すると仮定する。ここで、元々の鍵管理木の単位時間あたりの鍵交換数を、

$$\lambda_0+\lambda_1$$

と置く。さらに、元々の鍵管理木の平均鍵交換数は高々完全二分木のモデルの平均鍵交換数以下であるとする。つまり、全体の人数に L_0 対して、

$$K_0 \le (\lambda_0+\lambda_1)\lceil \log L_0 \rceil$$

が成立すると仮定する。この様にした場合に鍵交換数が改善する条件を求める。

この時、 L_1 人取り出しても残った鍵管理木の効率が K_0 のまま改善しない場合まで考慮すると、求める平均鍵交換数 K_1 は次のようになる。

$$K_1 \le \frac{\lambda_1}{\lambda_0+\lambda_1}(1+\lceil \log L_1 \rceil)+\frac{\lambda_0}{\lambda_0+\lambda_1}(1+K_0) \qquad (2)$$

これが鍵交換数を改善するためには、

$$K_0-K_1>0$$

である必要がある。従って、これを計算すると次のようになる。

$$K_0-K_1 \ge K_0-\frac{\lambda_1}{\lambda_0+\lambda_1}(2+\log L_1)-\frac{\lambda_0}{\lambda_0+\lambda_1}(1+K_0)$$

$$=\frac{\lambda_1}{\lambda_0+\lambda_1}(K_0-\log L_1-1)-1$$

これが正になるには次の条件を満たせば良い。

$$\log L_1 < K_0 - 2 - \frac{\lambda_0}{\lambda_1} \qquad (3)$$

例えば、メンバー 2^{16} 人が常時参加しているサービスを考える。ここで更新頻度が 20%のメンバーを取り出し、Key Tree を分割して、図 11 のような構成にする。分割前の Key Tree が平衡状態であったとき、取り出したメンバーの人数を何人に抑えれば、分割前より平均鍵更新数を減らせるかを求める。

分割前の Key Tree では 1 回あたりの平均鍵更新数は $\lceil \log_2 2^{16} \rceil = 16$ である。求めたい人数を L とおくと、式 3 により、

$$\log_2 L < 16 - 2 - \frac{1 - 0.2}{0.2} = 10$$

よって、

$$L < 2^{10}$$

であればよいことがわかる。

5.2. 線形リスト

前節で述べたのは完全二分木以上に効率の良い鍵管理木において、特定の条件を満たすようなグループを分割し、頂点を一個拡張して二分木にすると効率が良くなるということである。これは図 12 のように任意の回数繰り返すことができるので、分離した木を線形リストにつなぐと効率が良くなるということである。

具体的には、次のようにして行う。

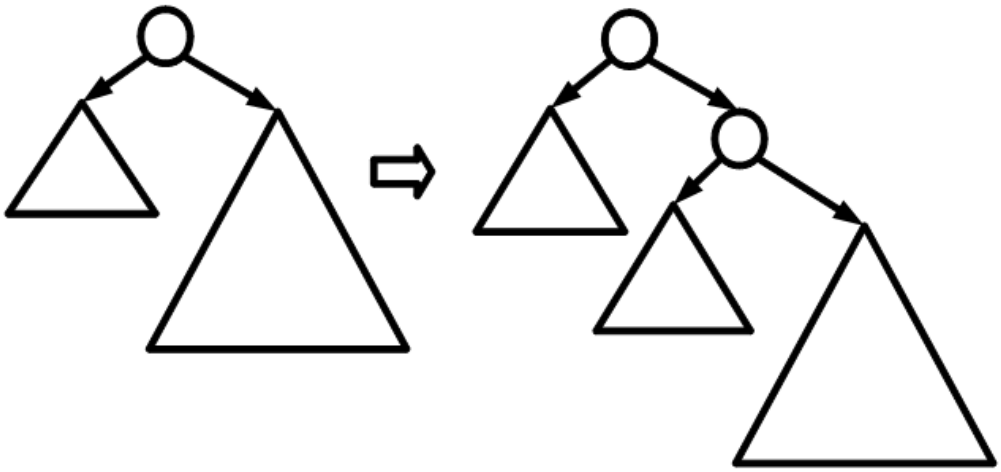

図 12:線形リストの構築

(1) まず、式 3 を満たす、確率が大きくて、小さなグループ G_1 を取り出し、二分木を作る。

(2) 残りのメンバーに対して、残りのメンバーの中で確率が大きくて小さなグループを再び取り出して、二分木を拡張する。

(3) これをグループが取り出せなくなるまで繰り返す。

確率の多いグループを取り出してから、次のグループを取り出すので、最初は条件を満たさず取り出せなかったグループを取り出すことができる。

さて、今、グループ G_1 、 G_2 の参加平均人数が L_1 、 L_2 、鍵更新生起確率 p_1 、 p_2 で、残りのメンバーの平均鍵更新数が K であると仮定する。この時、グループ G_1 、 G_2 で線形リストを作った場合の平均鍵更新数 K_3 と、 G_1 と G_2 のグループを統合して二つの木を統合した二分木の平均鍵更新数 K_2 について考える。各値は次の式で求まる。

$$K_3 = p_1(1+\lceil \log L_1 \rceil) + p_2(2+\lceil \log L_2 \rceil) + (1-p_1-p_2)(2+K)$$

$$K_2 = (p_1+p_2)(1+\lceil \log(L_1+L_2) \rceil) + (1-p_1-p_2)(1+K)$$

これに対して、単に二分割して二分木を作るのではなく、線形リストにした方が効率が良くなる条件、つまり

$$K_2 - K_3 > 0$$

なる条件を求める。

$$K_2 - K_3 \geq (p_1+p_2)\lceil \log(L_1+L_2) \rceil - p_1\lceil \log L_1 \rceil - p_2\lceil \log L_2 \rceil - (1-p_1) \qquad (4)$$

ここで、 $L_2 > L_1$ を仮定し、 $\lceil \log(L_1+L_2) \rceil > \log L_2$ を使用すると、

$$式4 > p_1(\log L_2 - \log L_1) - 1 - P_2 \qquad (5)$$

が得られる。これが正という条件なので、次が求めるべき条件になる。

$$\log L_2 > \log L_1 + \frac{1+p_2}{p_1} \qquad (6)$$

なお、 $L_1 > L_2$ を仮定し、 $\lceil \log(L_1+L_2) \rceil > \log L_1$ を使用した場合は、

$$\log L_2 < \log L_1 - \frac{1+p_1}{p_2} \qquad (7)$$

が得られる。

5.3. グループの順番

さてここで、複数のグループが線形リスト状に並べられているとき、 j 階層と $j+1$ 階層の二つの隣り合ったグループ G_1 、 G_2 を、 G_1 、 G_2 の順に並べた平均鍵交換数 K_{12} との、 G_2 、 G_1 順に並べた平均鍵交換数 K_{21} のどちらが良いのかを検討する。 G_1 、 G_2 以外の平均鍵交換数を K とするとこれは K_{12} 、 K_{21} によらず一定になる。この時、それぞれの値を求め、差を求めると次が得られる。(下記の K はそれぞれのグループ以外のメンバーの平均鍵交換数)

$$K_{12} = p_1(j+\lceil \log L_1 \rceil) + p_2(j+1+\lceil \log L_2 \rceil) + (1-p_1-p_2)K$$

$$K_{21} = p_2(j+\lceil \log L_2 \rceil) + p_1(j+1+\lceil \log L_1 \rceil) + (1-p_1-p_2)K$$

$$K_{12} - K_{21} = p_2 - p_1$$

つまり、確率の大きい順に並べるのが得策であることがわかる。

例えば、メンバー 2^{16} 人が常時参加しているサービスを考える。ここで更新頻度が 20%のメンバー 2^9 人のグループ G_1 と、更新頻度が 10%のメンバー L 人のグループ G_2 を取り出し、

Key Tree を分割する。分割前の Key Tree が平衡状態であったとき、L を何人に抑えれば、 G_1 のみ分割したときより平均鍵更新数を減らせるかを求める。

分割前の Key Tree では 1 回あたりの平均鍵更新数は $\lceil \log_2 2^{16} \rceil = 16$ である。また、 G_1 だけ分割した場合の平均鍵更新数はおよそ、

$$0.2 \times (1+9) + 0.8 \times (1+16) = 15.6$$

より、15.6 回である。 G_2 も分割したときの平均鍵更新数 K は、

$$K < (1-0.1-0.2)(2+16) + 0.2 \times (2+10) + 0.1 \times (1+\lceil \log_2 L \rceil)$$

$$= 14.9 + 0.1 \lceil \log_2 L \rceil$$

これが、15.6 より小さくなるように L を求めると、

$$14.9 + 0.1 \lceil \log_2 L \rceil < 15.6$$

$$L < 2^7$$

よって、 2^7 人以下であればよい。

6. まとめ

本研究では、マルチキャストでのグループ鍵管理に関して、メンバーのグループへの平均参加時間がわかっている場合に、既存のグループ鍵管理手法である LKH を発展させ、より効率のよい手法を提案した。また、ある程度の大きさの確率で発生するグループを抜き出して、線形リスト上に接続することで効率が向上する条件を求めた。

先行研究で挙げた Batch LKH は LKH に対する一つの改善手法である。本提案手法と Batch LKH は排他的な関係ではなく、同時に適用することも可能だと思われる。マルチキャストグループの利用ケースによっては、同時に適用することで、より効率の高い鍵管理を実現できる可能性があると考えられる。

本提案手法の解析では、分割された Key Tree が常に平衡状態を維持すると仮定している。実際の運用においては、本提案手法は LKH と同様に分割された Key Tree は不平衡な状態になる可能性がある。よって Balanced Batch LKH のような平衡状態を維持する手法の適用も必要であると考えられる。

今後はこれらの可能性を踏まえて、本提案手法のさらなる改善を行おうと思う。

7. 参考文献

[1] Thomas A. Muafer, 楠本 博之 訳, “IP マルチキャスト入門”, 共立出版, 2001.

[2] S. Deering, “Host Extensions for IP Multicasting”, RFC 1112, 1989.

[3] Z. Albanna, K. Almeroth, D. Meyer, M. Schipper, “IANA Guidelines for IPv4 Multicast Address Assignments”, RFC 3171, 2001.

[4] 馬場 達也, “マスタリング IPsec 第 2 版”, オライリー・ジャパン, 2006.

[5] 谷口 功, 水澤 紀子, “マスタリング TCP/IP IPsec 編”, オーム社, 2006.

[6] Kent S., K. Seo, “security Architecture for the Internet Protocol”, RFC 4301, 2005.

[7] Kent S., “IP Authentication Header”, RFC 4302, 2005.

[8] Kent S., “IP Encapsulating Security Payload (ESP)”, RFC 4303, 2005.

[9] D. Maughan, M. Schertler, M. Schneider, J. Turner, “Internet Security Association and Key Management Protocol (ISAKMP)”, RFC 2408, 1998.

[10] H. Orman, “ The OAKLEY Key Determination Protocol”, RFC 2412, 1998.

[11] D. Harkins, D. Carrel, “The Internet Key Exchange (IKE)”, RFC 2409, 1998.

[12] C. Kaufman, “Internet Key Exchange (IKEv2) Protocol”, RFC 4306, 2005.

[13] T. Ballardie, “Scalable Multicast Key Distribution”, RFC 1949, 1996.

[14] H. Harney, C Muckenhirn. “Group Key Management Protocol (GKMP) Specification”, RFC 2093, 1997.

[15] H. Harney, C. Muckenhirn, “Group Key Management Protocol (GKMP) Architecture”, RFC 2094, 1997.

[16] Debby M. Wallner, Eric J.Harder, Ryan C. Agee, “Key Management for Multicast: Issues and Architectures”, RFC 2627, 1999.

[17] Hardjono, T. and B. Weis, “The Multicast Group Security Architecture”, RFC 3740, 2004.

[18] M. Baugher, R. Canetti, L. Dondeti, F. Lindholm, “Multicast Security (MSEC) Group Key Management Architecture”, RFC 4046, 2005.

[19] H. Harney, U.Meth, A. Colegrove, G. Gross, “GSAKMP: Group Association Key Management Protocol”, RFC 4535, 2006.

[20] Bibo Jiang, Xiulin Hu, “A Survey of Group Key Management”, csse, vol. 3, pp.994-1002, 2008 International Conference on Computer Science and Software Engineering, 2008.

[21] Chung Kei Wong, Mohamed Gouda, Simon S. Lam, “Secure Group Communications Using Key Graphs”, IEEE/ACM Transactions on Networking, vol. 8, No. 1, pp.16-30, 2000.

[22] Xiaozhou Steve Li, Yang Richard Yang, Mohamed G. Gouda, Simon S. Lam, “Batch Rekeying for Secure Group Communications”, Proceedings of Tenth International World Wide Web Conference, 2001.

[23] Josep Pegueroles, Francisco Rico-Novella, “Balanced Batch LKH: New Proposal, Implementation and Performance Evaluation.”, iscc, pp.815, Eighth IEEE Symposium on Computers and Communications, 2003

[24] Takahito Sakamoto, Takashi Tsuji, Yuichi Kaji, “Group Key Rekeying Using the LKH Technique and the Huffman Algorithm”, ISITA, 2008.

[25] Deuk-Whee Kwak, SeungJoo Lee, JongWong Kim, Eunjin Jung, “An efficient LKH balancing algorithm for Group Key Management”, IEEE Communications Letters, Vol. 10, No.3, March 2006.

[26] 秋丸 春夫, 川島 幸之助, “情報通信トラヒック -基礎と応用- 改訂版”, 電気通信協会, 2000.

[27] Pin-Yun Tarng, Kuan-Ta Chen, Polly Huang, “An Analysis of WoW Players' Game Hours”, Proceedings of the 7th ACM SIGCOMM Workshop on Network and System Support for Games, pp.47-52, 2008.

[28] Diffie, W., M. Hellman, “New Directions in Cryptography”, IEEE Transactions on Information Theory, vol IT-22, Nov 1976, pp.644-654.