2005－DSM－39（9）
2005／10／14

鍵の同期を考慮した鍵の配布・更新の提案と実装

浅野　歩†　岸田　崇志†　前田　香織‡　河野　英太郎‡

†広島市立大学大学院情報科学研究科
‡広島市立大学情報処理センター
〒731-3194 広島市安佐南区大塚東 3-4-1

E-mail: †{asano,takashi}@v6.ipc.hiroshima-cu.ac.jp, ‡{kouno,kaori}@ipc.hiroshima-cu.ac.jp

あらまし マルチキャストのデータを暗号化するための鍵を，複数のメンバに対して配布し更新する技術は幾つか標準化が進んでいる．セキュリティのために，メンバの参加と離脱毎に鍵を更新する必要があるが，メンバ間で鍵を一致させる方法についてはあまり考慮されていない．そのため，鍵の不一致による通信劣化が生じることもある．本研究ではこの問題を解決するため，鍵を定期的に更新する仕組みを提案し，その実装と評価を行う．

キーワード マルチキャストセキュリティ，鍵更新，IPsec, 暗号化

A Proposal and Implementation of Key Distribution and Rekey Considering Key Synchronization

Ayumu ASANO†　Takashi KISHIDA†　Kaori MAEDA‡　Eitaro KOHNO‡

†Graduate School of Information Science, Hiroshima City University
‡Information Processing Center, Hiroshima City University

E-mail: †{asano,takashi}@v6.ipc.hiroshima-cu.ac.jp, ‡{kouno,kaori}@ipc.hiroshima-cu.ac.jp

Abstract Some technologies to distribute key for encryption of multicast data are being standardized. Rekey is important for security when group membership changes. However, synchronization of keys of each members is not enough considered. Then, it may affect communication quality by key conflict. In this research, we propose a periodic rekey mechanism to solve this problem. In this paper, we show this mechanism and its evaluation.

Keyword Multicast security, Rekey, IPsec, Encryption

1. はじめに

近年，ネットワークの広帯域化と高機能化により，複数拠点でのテレビ会議など，マルチキャストを用いるアプリケーションを使う機会も増えている．

一般に，マルチキャストグループへは誰でも参加することができるため，無関係な第三者に通信を盗聴される可能性がある．そのため，データを暗号化し，受信を許可するメンバにだけに鍵を渡しておくことで，受信可能なメンバを限定する方法がある[1]．また，メンバの参加状況は変わるので，新規受信者が参加許可前のデータを復号できないように，もしくは離脱後のデータを復号できないようにするために，参加・離脱毎に鍵を更新する必要がある．

マルチキャストの通信を暗号化するために，複数のメンバへ鍵を配布・更新する方法はいくつか提案されており，例えば LKH[2]や OFT[3]などがある．これらは主に放送のような一方向の通信での利用を想定しており，数万以上のメンバへ鍵を配布・更新することが可能である．この方式をテレビ会議のような複数の送信者が存在する双方向の通信に適用した場合，鍵管理が困難になる場合がある．例えば，鍵更新時に送信するメンバは全メンバに新しい鍵が届いたことを確認した後で，新しい鍵を用いる必要がある．確認を行わず，新しい鍵を得ていないメンバがいる状態で，他のメン

バが新しい鍵を用いてデータを暗号化して送信した場合，まだ新しい鍵を得ていないメンバは復号に失敗する．つまり，受信するための新しい鍵を全メンバが得た状態で，送信するメンバは新しい鍵を使い始めることが重要である．本稿では全員が同じ鍵を使える状態になることを鍵の同期と呼ぶ．鍵の同期のためには，全メンバに鍵が届いたことを確認する方法と，鍵を使い始めるタイミングを通達する方法が必要であるが，複数拠点でこれを実現するのは困難である．

そこで，鍵を定期的に更新することで，鍵の同期を実現する仕組みを提案する．ここでは，複数拠点でマルチキャストを用いてセキュアなテレビ会議をするような場合を対象とする.本稿では鍵同期の提案と実装，評価について記述し，提案方法の有用性を示す．２章では定期的鍵更新を実現する仕組みについて概要を述べる．３章では実装について説明し，4 章で開発したシステムの評価を行う．最後に，5 章でまとめと今後の課題について述べる．

2. 定期的鍵更新の提案

2.1. 定期的鍵更新の仕組み

提案する定期的鍵更新では，メンバの参加離脱に関わらず,期間ごとに鍵サーバから配布される鍵を使い，鍵更新を行う．各メンバは同じ期間中は同一の鍵で送信時に暗号化，受信時に復号を行う．この時，送信と受信に使う鍵を別のタイミングで使い始める方法をとっている．図 1に，あるメンバの鍵の所持状態の例を示す．他のメンバも同様に鍵を所持している．期間 1～3 で使用される鍵を，それぞれ鍵 1～3 とする．

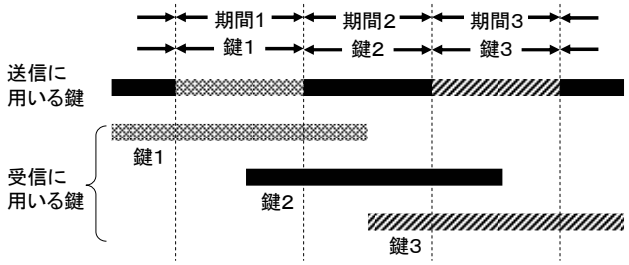

図 1．定期的鍵更新における鍵の所持状態

各メンバが次の期間に入ると同時に新しい鍵を使い始めることができるように，前の期間中に新しい鍵を配布しておく．例えば，鍵 2 は前の期間 1 の間に各メンバに配布される．各メンバは鍵 2 を用いて暗号化されたデータをいつでも復号できるように，期間 1 の間に受信に用いる鍵として復号可能な状態にしておく．そして,期間 2 に入ると鍵 2 を送信の暗号化に用いる．メンバ間で期間を正確に同期させることは困難であるが，鍵更新前後ではどちらの鍵でも復号できるため，ある程度のずれは許容できる．

各メンバがこのような定期的鍵更新の処理を行うことで，受信したデータが復号できない状態を回避することできる．このような仕組みで鍵の同期を保証しようとするのが，提案の基本的アイデアである．

一方，メンバの参加離脱毎に鍵更新を行う方法もある．この方法も，すぐに鍵更新をするのではなく，定期的鍵更新と同様に送信と受信に使う鍵を分け，新しい鍵を使い始めるまでに適当な期間をおくことで，鍵の同期が可能である.参加離脱の起こる回数が少なく，鍵更新の頻度が低い場合は2.2で説明する鍵配布の処理が少なくなり，鍵更新に必要なトラフィックなどの負荷が軽減される．しかし，同じ鍵が使われ続ける危険性があるのに加え，メンバの参加離脱の頻度によっては鍵更新の頻度が高くなり，逆に鍵更新に必要な負荷が大きくなる可能性がある．よって，今回は定期的に鍵更新を行うことで同じ鍵が使われ続ける危険性を小さくし，メンバの参加離脱に関わらず鍵更新の負荷が一定である定期的鍵更新を実装することにした．

2.2. 鍵の配布方法

定期的な鍵更新を行うにあたって, 2.1で事前に新しい鍵を配布する仕組みを説明した．このとき，新しい鍵は前の期間内に全メンバに配布されなければならない．また，参加離脱によってメンバは動的に変化することから，許可された特定のメンバにだけに新しい鍵を配布することが求められる.これは LKH で可能であるが，実装の容易さから以降で述べる簡略化した方法を用いる．

鍵配布の方法を図 2を用いて説明する．ここでは，２種類の鍵を用いる．１つは TEK (Traffic Encryption KEY)と呼び，図 1の鍵 1～3 にあたるもので，実際にデータを暗号化するための鍵，もしくは鍵を生成するためのマテリアルとして用いられる．もう一つは KEK (Key Encryption Key)と呼び, 共通鍵として TEK を暗号化して配布するために用いられる．

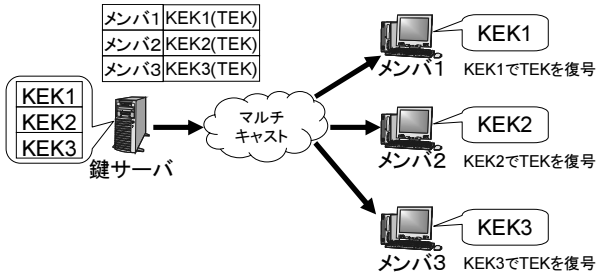

図 2．鍵配布・更新の仕組み

鍵を配布する役割を担うのが鍵サーバである．鍵サーバはメンバごとに異なる KEK を持っており，図 2 で KEK1～3 として示している.ここでは既に鍵サーバとメンバ間で KEK が共有されているものとする. TEK を各メンバに配布する場合，鍵サーバは各メンバの

KEK で TEK や鍵の生存期間などの情報を暗号化し，まとめてマルチキャストで各メンバへ配布する．各メンバは，各自所持している KEK で暗号化された TEK を復号することで鍵を得る．鍵更新は，新たな TEK を各メンバの KEK で暗号化して送ることで実現する．

特定の許可されたメンバにだけ鍵を配布する方法は，鍵サーバが鍵配布時に特定のメンバのエントリを追加・削除することで実現する．例えば，メンバ 3 の受信を不許可にする場合は，メンバ 1 と 2 のエントリだけを配布する．また，メンバ 3 の受信を許可する場合は，メンバ 3 のエントリを追加する．このようにして，会議の管理者が鍵サーバを通してメンバの参加許可，不許可を操作することができる．

マルチキャストで鍵を配布するため，パケット損失によって鍵を正しく配布できない可能性がある．この問題に対しては，鍵配布のパケットを複数回送ることで鍵到達の信頼性を高める．再送などの機構を取り入れることも考えられるが，各メンバからの確認応答の集中や，各メンバの鍵の所持状態を把握しておく必要があるため，スケーラビリティの問題や処理の複雑化を招くと考えた．そこで，提案手法では，ネットワークに対する負荷を考慮しつつ，鍵を複数回送ることで信頼性を高める方法を用いた．期間内に何度鍵を配布すれば信頼性を向上できるかは4.6，鍵配布に必要なトラフィックは4.7で考察する．

3. 実装

ここでは，2．で述べた定期的鍵更新の基本的な動作を実装したプロトタイプシステムについて説明する．以降，このシステムを本システムと呼ぶ．

3.1. システム構成

IETF の Multicast Security Working Group で標準化された RFC3740[4]に，マルチキャストセキュリティを実現するための GSA (Group Security Association)の枠組みについて記述されている．その中で，図 3のように Data SA，Re-key SA，Registration SA という 3 種類の SA (Security Association)が，GSA を構築する機能を大別する意味で定義されている．

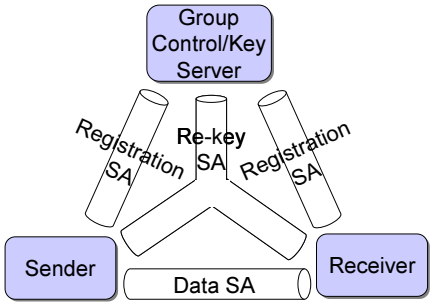

図 3．GSA の構成の大枠

本システムに当てはめると，Group Control/Key Server は鍵サーバにあたり，Sender と Receiver はメンバに当たる．Data SA はストリームなどの実際のデータを暗号化するための機能で，IPsec[5]や SRTP[6]，個々のアプリケーション固有の暗号化の方法に相当する．Re-key SA は，Data SA を構築するための鍵などの情報をマルチキャストで配布するための機能で，LKH や ISAKMP[7]の phase 2 の機能に相当する．また，Registration SA は，Re-key SA を構築するために必要な鍵などの情報を交換する機能で，ISAKMP の phase 1 に相当する．

本稿で提案する定期的な鍵配布は，Re-key SA の部分に相当する．鍵配布でメンバの参加許可の制御や，鍵サーバとメンバ間で KEK を共有する必要があるが，これは Registration SA に相当する．今回はメンバの参加許可の制御はせず，あらかじめ KEK を共有できているものとしており，Registration SA の部分は実装していない．Data SA は実際にアプリケーションを暗号化する部分であるが，多くのアプリケーションの暗号化に適用可能な IPsec を用いている．IPsec の仕様として，図 1で説明したように複数の鍵でパケットを受信できる状態にしておくことができ，途切れない鍵更新を実現できるのも大きな利点である．

3.2. システム詳細

鍵配布のための独自のプロトコルを設計した．図 4 にパケットフォーマットを示す．

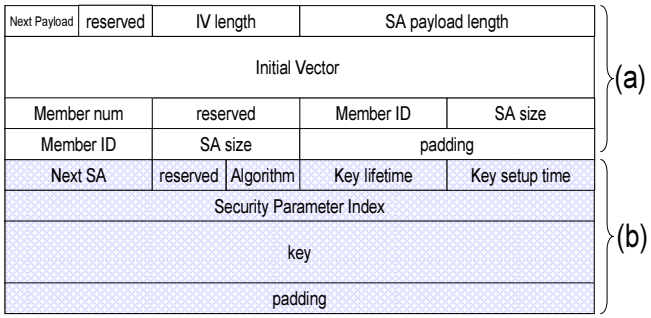

図 4．鍵配布プロトコルのパケットフォーマット

(a)の部分はヘッダで，パケットのシーケンスナンバや暗号化に用いる IV(Initial Vector)など，全メンバに通知する情報で構成される．(b)の部分は SA ペイロードで，各メンバが IPsec の SA 構築に必要な暗号化すべき情報で構成される．鍵やその生存期間，暗号化アルゴリズムの種類などの情報をメンバごとに複数伝えることができる柔軟な設計となっている．KEK で暗号化されるのは SA ペイロードの部分で，メンバの数だけ，暗号化したデータを連結する．よって，メンバ数や IPsec で用いる SA の鍵長，SA ペイロードの暗号化方法によって可変長である．

上記のデータを UDP ヘッダの後に付加し，マルチキャストで送信することで鍵配布を行う．

3.3. 開発環境

開発は C 言語で行い，FreeBSD 5.4-RELEASE と Debian GNU/Linux kernel 2.6.12 で動作を確認している．SA ペイロードの暗号化は OpenSSL の暗号化ライブラリを用いている．IPsec スタックは各 OS に標準で搭載されているものを用いており，PF_KEY Key management API, version 2[8]を通して SAD を管理している．また，IPsec は ESP を用いて暗号化だけをしており，認証は行っていない．鍵の配布は IPv6/v4 の両方に対応している．

Linux kernel 2.6.12 は IPsec スタックに一部制約があり，本システムを正しく動作させるためには双方向通信を行わず，送信か受信の片方だけにする必要がある．FreeBSD を用いる場合はそのような制約はなく，送信，受信の両方を同時に行うことができる．また，IPsec の動作モードはトランスポートモードでの使用を想定して実装しているが，トンネルモードにも対応可能である．

4. 評価

4.1. 実験環境

以降の評価実験に用いたネットワーク構成とマシンの仕様を図 5に示す．

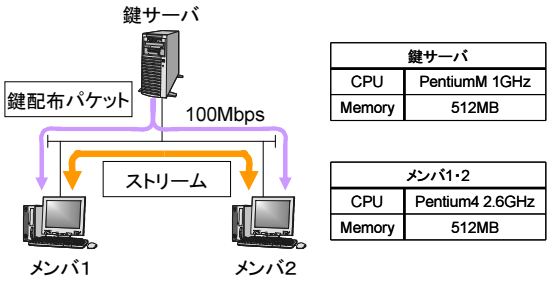

図 5. ネットワーク構成とマシンスペック

4.2. 鍵の同期確認

メンバ間で，鍵の同期が正しく行われているかを調べるために，各メンバが所持している鍵を比較する．IPsec では，鍵や鍵の生存期間，プロトコル，SPI(Security Parameter Index)などの情報をまとめて SA と呼んでいる．SPI は SA を識別するための数値である．ここでは，SA と鍵をほぼ同じ意味で使っている．図 6はメンバ 1 を送信側，メンバ 2 を受信側としたときの，SA 所持状態のプロットを 3 期間分(T-1, T, T+1)取り出したものである．kernel が保持する SA の取得には IPsec の手動管理ツールである setkey[9]を用い，1 秒間隔に出力した．鍵の更新間隔は 10 秒，生存期間は 24 秒に設定した．

ここでは，期間 T に注目する．T の間は，SPI が 333 の SA を用いる．図 6で，送信，受信側のそれぞれで SPI が 333 の SA を所持しているところを網掛けしている．受信側では，T-1 の間に T で使う SPI が 333 の SA が配布され，既に受信可能な状態になっていることがわかる．また，送信側は T から SPI が 333 の SA を使い始めていることが確認できる．この時，既に受信側では SPI が 333 の SA を所持しているため，受信したパケットを正しく復号することができる．他の期間においても常に 1 期間先の SA が送信，受信側それぞれに配布されており，鍵の同期を保証した鍵配布ができていることが確認できた．

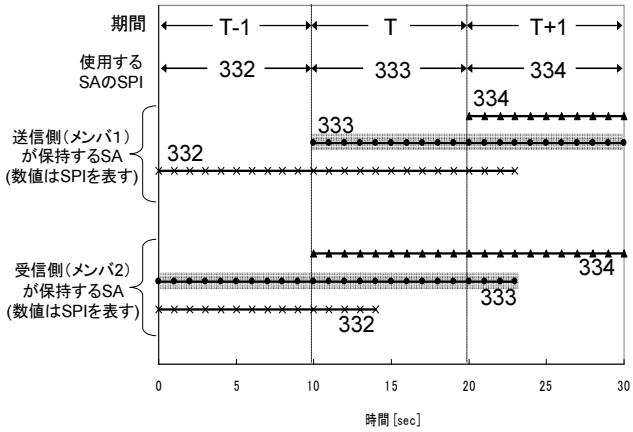

図 6. 各メンバの鍵の所持状態

4.3. アプリケーションを用いた動作確認

双方向のマルチキャストで音声の伝送が可能な RAT[10]と，片方向のマルチキャストで音声や動画の伝送が可能な VLC[11]の 2 種類のアプリケーションで動作確認を行う．両方とも IPv4/v6 に対応しており，それぞれについて動作確認を行う．RAT 送信データとしては，エンコーディングが 16bit 非圧縮 PCM，サンプリングレートが 44kHz のステレオを使った．よって，RAT が送出する帯域は約 1.4Mbps である．VLC 送信データとしては，エンコーディングが MPEG2 で，送信帯域は約 3Mbps である．まず，本システムを用いず暗号化を行わない通常時で動作可能であるかを検証し，その後に本システムを用いて暗号化した通信が可能であるかを調べた．その結果を表 1に示す．

表 1. アプリケーションの動作確認

			RAT		VLC	
			通常時	本システム使用時	通常時	本システム使用時
FreeBSD	片方向	IPv4	○	○	○	○
		IPv6	○	○	○	○
	双方向	IPv4	○	○		
		IPv6	○	○		
Linux kernel 2.6.12	片方向	IPv4	○	○	○	○
		IPv6	○	○	○	○
	双方向	IPv4	○	△		
		IPv6	○	△		

通常時に動作したアプリケーションは，ほとんどの場合，本システムを用いて暗号化して通信することが可能であることが確認できた．表 1の△は，送信と受信の片方だけで RAT の使用が可能であることを表している．これは Linux kernel 2.6.12 の IPsec スタックでは，宛先アドレスが同じで，送信元アドレスが異なる SA は別であるにも関わらず，同じ SA として扱うため，送信用 SA と受信用 SA を区別できないからである．

4.4. ストリーム伝送への影響

鍵更新時に鍵の同期を保証する仕組みによって，鍵の不一致によるパケット損失が発生しないことを確認する．ストリーム伝送ツールとして，FreeBSD 上で RAT を動作させた場合を調べる．本システムと RAT は共に IPv6 で動作させた．RAT に設定したパラメータは4.3と同様である．IPsec の拡張ヘッダを含めたパケット長は 652byte，パケット送信間隔は平均して約 3.3msec である．受信側メンバで tcpdump[12]を実行し，IPsec の SPI が変化して鍵更新が行われた時刻を調べる．RAT のログから鍵更新された時間前後の情報を抜き出し，受信時刻とシーケンスナンバでグラフ化したものが図 7である．

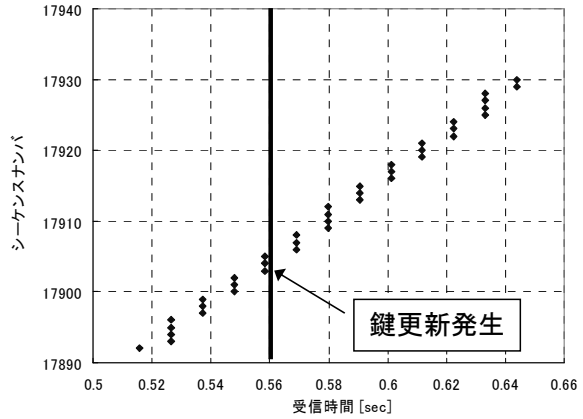

図 7. 鍵更新時の影響

鍵更新前後で，パケットの到着間隔に変化が見られないことが分かる．よって，本システムを用いて鍵の同期をとることで，ストリーム伝送の品質を低下させることなく，鍵更新が可能であることが確認できた．

鍵の同期を考慮していない場合，あるメンバへの鍵配布パケットの遅延や，パケット損失によって，他のメンバがそのメンバの所持していない鍵を使って暗号化したパケットを送る可能性がある．例えば，他のメンバに比べて鍵の取得が 50msec 遅れた場合，図 7中の鍵更新付近で約 50msec の間はパケットの復号に失敗する危険性がある．定期的鍵更新は，ネットワークの遅延に比べて長い時間を想定しており，ある程度の鍵配布にかかる遅延は吸収することができる．

4.5. 鍵更新時の遅延に関する考察

定期的鍵更新において，メンバの参加や離脱時に遅延時間が発生する．これを図 8を用いて説明する．

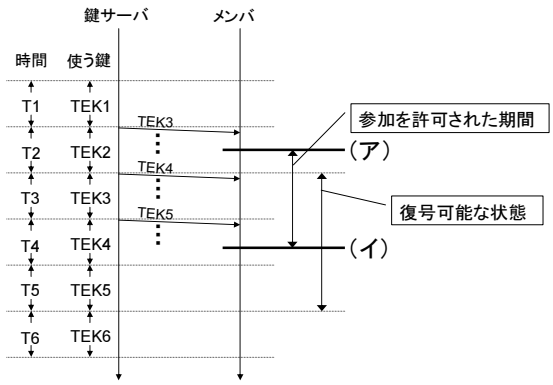

図 8. 定期的鍵更新による遅延時間

メンバは（ア）の時点で参加を許可され，次の期間で使われる TEK3 を得ることができるが，T2 の間は TEK2 が使われるため，T3 に入るまでは復号不可能な状態が続く．そのため，すぐに参加ができないことから，利便性に問題がある．次に，（イ）の時点で離脱するとする．このメンバは T4 の初めに配布された，TEK5 持っているので，T5 が終わるまでは復号可能となる．よって，不許可後もしばらく受信を続けることができるため，セキュリティ的問題がある．このように，定期的鍵更新はメンバの参加の許可状態を直ちに反映することができない．それに対して，メンバの参加離脱毎に鍵更新する方法は，参加の許可状態がすぐに反映される．

4.6. 鍵の配布回数に関する考察

本システムでは，マルチキャストで鍵を配布するため，鍵を複数回配布することによって鍵の到達する信頼性を高めている．ここでは，期間内に何回鍵を配布すれば十分な信頼性を得られるかについて考察する．

例として，5 秒間隔で鍵更新しながら（1 期間は 5 秒），1 時間のテレビ会議を行う場合を考える．その間，全期間で全メンバに鍵が到達すれば成功とし，1 期間でも鍵が到達しないメンバが存在すれば失敗とする．そして，期間中に何回鍵を配布すれば 99%以上の確率で成功するかを求める．メンバ中で最もパケット損失率が高いメンバのパケット損失率を p とする．また，期間中の鍵配布回数を n とする．1 期間中に、最もパケット損失率の高いメンバに鍵が到達する確率は $1-p^n$ となる．鍵更新間隔は 5 秒なので，1 時間の間に 720 回の鍵更新が行われる．よって，全期間で鍵が到達する確率は $(1-p^n)^{720}$ となる．このメンバに 99%以上の確率で全期間に鍵を到達させるための鍵配布回数は，$(1-p^n)^{720} \geqq 0.99$ を満たす n となる．この数式のグラフを図 9に示す．

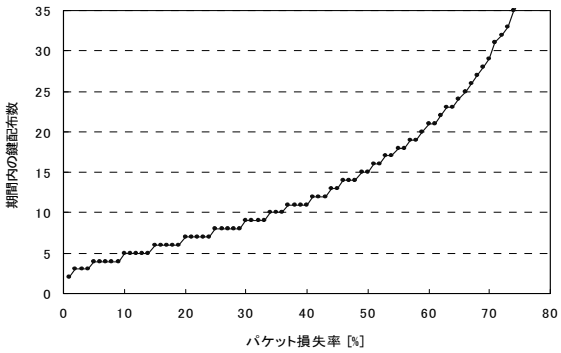

図 9. 鍵配布回数

例えばパケット損失率 p が 10%というストリーム伝送には劣悪な通信品質の場合でも，少なくとも 1 期間中に 5 回ほど同じ鍵を配布することで，全期間での鍵到達率を 99%にすることができる．この鍵配布回数であれば，よりパケット損失率の低い他のメンバにも鍵が到達すると考えることができる．

4.7. 鍵配布に必要なトラフィック

パケットフォーマットについては3.2で述べ，その中でヘッダに加えて SA ペイロードはメンバ数や鍵長によってサイズが増加することを説明した．ここでは，例を挙げて鍵配布に必要なトラフィックについて考察する．128bit の鍵を 1 つ配布するために，1 メンバあたり約 34byte を必要とする．MTU が 1500byte の場合に分割せずに一度に鍵配布できるメンバ数は，IPv6 ヘッダ，UDP ヘッダも考慮して 43 メンバであり，この場合はパケット長が 1492byte になる．より長い鍵長を配布すれば 1 メンバあたりに必要な情報量が増え，一度に配布できるメンバ数は減る．また，より多くのメンバに配布したい場合もある．このような場合は，鍵配布を複数回に分けることで対応可能であるが，今回は実装していない．

4.6の結果をふまえて，鍵配布に必要なトラフィックを計算する．1 期間が 5 秒である場合，その間に鍵配布を 5 回行えば信頼できる鍵到達率になった．43 メンバに鍵配布をした場合は，1492byte のパケットを毎秒送ることになり，必要な帯域は約 12kbps となる．一般的なストリーミングアプリケーションに比べて必要とする帯域は小さく，無視できると言える．

4.8. 評価のまとめ

本システムが正しく動作し，鍵の同期をとることで，鍵更新時に RAT や VLC といったアプリケーションの通信に影響を与えないことを確認した．また，鍵配布には多くて 10 数 kbps の帯域を必要とするが，テレビ会議に用いられるアプリケーションと比べて必要とする帯域は小さく，多くの場合は問題にならないと考えている．安全性は暗号化アルゴリズムの強度に依存するため，解読されない保証はないが，日常で行うテレビ会議のセキュリティを高める用途には利用できる．以上の結果から，メンバ数が数 10 までのテレビ会議を行う場合に，本システムを用いることで通信を暗号化し，セキュリティを高めることが可能である．

5. おわりに

本稿では，双方向マルチキャストの通信において鍵の同期を保証するために，定期的に鍵更新する方法を提案し，実装と評価を行った．そして，アプリケーションの品質を落とすことなく鍵更新が可能であることを示した．今後は，メンバの管理機能を実装し，実際のテレビ会議に本システムを用いるなど，さらに評価を進めていきたい．

謝辞

本研究に際し，日頃からご指導いただく広島市立大学情報科学部石田賢治教授に感謝します．

参考文献

[1] 上野英俊，田中希世子，原下貴志，鈴木偉元，石川憲洋，高橋修, “マルチキャストセキュリティアーキテクチャの提案と実装”, DICOMO 2003, pp.113-116 (2003)

[2] D. Wallner, E. Harder, R. Agee, “Key Management for Multicast: Issues and Architectures”, RFC2627 (1999).

[3] Balenson, D., McGrew, P.C., and A. Sherman, “Key Management for Large Dynamic Groups: One-Way Function Trees and Amortized Initialization”, IRTF Work in Progress (2000).

[4] T. Hardjono, B. Weis, “The Multicast Group Security Architecture”, RFC3740 (2004).

[5] S. Kent, R. Atkinson, “Security Architecture for the Internet Protocol”, RFC2401 (1998).

[6] M. Baugher, D. McGrew, M. Naslund, E. Carrara, K. Norrman, “The Secure Real-time Transport Protocol”, RFC3711 (2004).

[7] D. Maughan, M. Schertler, M. Schneider, J. Turner, “Internet Security Association and Key Management Protocol “, RFC2408 (1998).

[8] D. McDonald, C. Metz, B. Phan, “PF_KEY Key Management API, Version 2”, RFC2367 (1998).

[9] “IPsec-Tools”, http://ipsec-tools.sourceforge.net/.

[10] “RAT”, http://www-mice.cs.ucl.ac.uk/multimedia/software/rat/.

[11] “VLC”, http://www.videolan.org/.

[12] “tcpdump”, http://www.tcpdump.org/.