マルチキャスト通信のための鍵配布に関する検討

濱野 淳史[*]　所 真理雄[†]

[*]慶應義塾大学大学院 理工学研究科 計算機科学専攻

[†]ソニー株式会社

概要

マルチキャスト通信のための鍵配布に関する研究は，近年その必要性が認められ，いくつかの提案がなされている．本稿ではこれらの機構について概観し，それらで共通して生ずる共有鍵の更新時における一貫性についての問題点を指摘する．共有鍵の一貫性が保証されない場合にはデータの再送が必要となり，リアルタイム性を要求するマルチキャストアプリケーションでは特に問題となる．また，サービスを一時停止して鍵の一貫性を保証するというアプローチもあるが，この手法でもリアルタイム性は保証できない．

このような問題を解決するために，本稿では，グループのメンバシップ変更時に配布される鍵とマルチキャストデータに因果順序を付加することで，鍵の一貫性を保証する鍵配布機構を提案し，その設計の詳細を示す．

1　はじめに

近年，vic[5] や wb[4] のようなマルチキャストアプリケーションの有用性が確認されている．しかし，それらのアプリケーションを商用に用いることを考えた場合には，通信の安全性，すなわちセキュリティの提供が不可欠となる．ユニキャスト通信の場合の二者間の鍵の共有とは異なり，マルチキャスト通信の場合はグループ間で 1 つの鍵を共有する必要がある．

マルチキャスト通信のための鍵配布に関する研究は，近年いくつかの提案がなされている [3, 7, 8]．これらの機構では，グループに参加していないホストによるマルチキャストデータの盗聴を防止するために，メンバの参加，離脱時にグループの共有鍵の更新を行なう．すなわち，グループに新しくメンバが参加した場合には，そのメンバが参加する以前のデータの盗聴を不可能にし，また，グループから離脱したメンバが，離脱後もデータを不正に取得できないことを実現する．このような機構は，グループに参加していた時間や受信したデータ量により課金する場合等に特に必要となる．

しかし [3, 7] では，共有鍵の更新時に，グループの一部のホストは既に新しい鍵を取得しているにもかかわらず，遅延等により他のホストが新しい鍵を取得しておらず，まだ古い鍵のままであるという状況が生じ得る．つまり，古い鍵しか保持していないホストは，既に新しい鍵を取得したホストから送信される，新しい鍵で暗号化されたデータを受信しても，復号化が不可能となる．その場合には，同一のデータの再送が必要となるため，結果としてデータの配送に大きな遅延が伴う．この遅延はリアルタイム性を要求するマルチキャストアプリケーションでは特に望ましくない．共有鍵の更新時における一貫性を保証するアプローチ [8] も提案されているが，システムのサービスを一時停止することで共有鍵の一貫性を保証しているため，この手法でもデータの遅延が生じてしまう．

そこで本稿では，マルチキャストデータに因果順序を付加することにより，グループの共有鍵の更新時における一貫性を保証する機構を提案する．本機構ではサービスの停止もなく，データの再送も可能な限り回避できるため，メンバの参加，離脱が頻繁に起こり，かつリアルタイム性を要求するリアルタイムアプリケーションで特に有効であると考えられる．

2　既存の機構における問題点

安全なマルチキャスト通信を実現するための機構は従来よりいくつか提案されている．本章ではそれらの機構における問題点を指摘する．

マルチキャスト通信のための規模拡張性のある鍵配布の枠組として Iolus [7] が提案されている．Iolus では，マルチキャストグループを複数のサブグループに分割し，サブグループごとに異なる鍵を共有することにより規模拡張性を提供する．グ

---

[*] hamano@mt.cs.keio.ac.jp

[†] tokoro@itlab.sony.co.jp

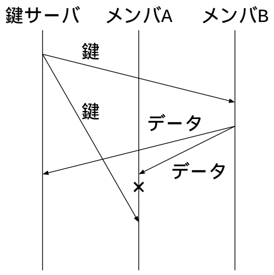


図 1: 共有鍵の更新時に一貫性が保証されない例

ループへのメンバの参加，離脱時には，各サブグループの鍵を管理するグループセキュリティエージェントが鍵の更新を行なう．この更新はグループの各メンバに対するマルチキャスト通信により行なわれる．このマルチキャスト通信は必要ならば信頼性のあるものが用いられる．

また別の手法として，Diffie–Hellman 鍵交換アルゴリズム [6] の拡張を用いてグループの共有鍵の生成，更新を行なう CLIQUES[3] が提案されている．CLIQUES では，グループの鍵管理は特定のホストが行なうのではなく，全メンバが協調して鍵を生成し，それをマルチキャストすることにより更新を行なう．

これらの手法では，共有鍵の更新時における一貫性は保証していない．つまり，更新時にグループ内に新しい鍵を保持するメンバと古い鍵しか保持しないメンバが混在してしまい，鍵が更新されたメンバが古い鍵で暗号化されたデータを受信した場合に復号化できない状況や，その逆の状況が生じる．図 1にこのような状況の例を示す．図 1では，鍵サーバが鍵を配布した際にメンバ A への鍵が遅延し，新しい鍵で暗号化されたメンバ Bからのデータをメンバ A において復号化できない．

一方，共有鍵の更新時の一貫性を保証するアプローチ [8] も提案されている．[8] では，鍵の更新時にサービスを一時停止することにより一貫性を保証する．図 2にこの手法の例を示す．鍵を更新する際には，鍵サーバが各メンバに対してユニキャストで新しい鍵を配布し，その確認応答を回収する．全メンバからの確認応答を得られた時点で鍵サーバは無効メッセージをマルチキャストする．各メンバは無効メッセージを受信するとサービスを停止し，確認応答を返送する．全メンバからの確認応答を得られた時点で鍵サーバは有効メッセージをマルチキャストする．そして，各メンバが有効メッセージを受信した時点から新しい鍵を用いてサービスを再開する．この手法では，鍵の更新時の一貫性は保証できてもサービスが停止してしまうので，リアルタイム性を要求するマルチキャストアプリケーションでは望ましくない．

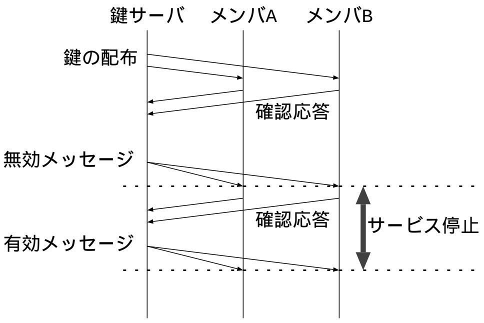


図 2: サービスを停止して一貫性を保証する例

したがって，どのような場合でも鍵の一貫性を保証でき，かつサービスが停止しないことが必要である．そこで次章では，これらの問題に対する解決策を示す．

3 解決策

基本方針は以下のとおりとする．

(A). 因果順序マルチキャスト通信 [2] の概念を用いる

(B). 鍵配布のためのマルチキャスト通信には信頼性のあるものを用いる

(C). 各メンバは常に 1 つの鍵しか保持しない

(A) によりマルチキャストデータに因果順序をつけることができ，鍵の更新がネットワークの遅延等により遅れた場合にも受信データの配送順序を変更できる．(B) により配布された鍵が必ず全てのメンバに送信されることが保証できる．さらに (C) により 1 つのホストで同時に 2 つ以上の鍵が存在する状態を避けることができる．以下では (A) の詳細について述べる．

図 3，4は，鍵の一貫性が保証されない 3 つの状態を示している．図の縦線は時間の経過を表し，矢印はデータの流れを表している．また，$\{K'\}K$は新しい鍵 K'が古い鍵 Kにより暗号化されたデータであることを示し，$\{M\}K$はデータ Mが鍵 Kにより暗号化されたデータであることを示している．

図 3は，鍵サーバからメンバ A へ配布される鍵が遅延したことにより，メンバ A において新しい鍵を保持していないにもかかわらず新しい鍵で暗号化されたデータを受信する場合である．この場合は，メンバ Bからのデータに因果順序を付加す

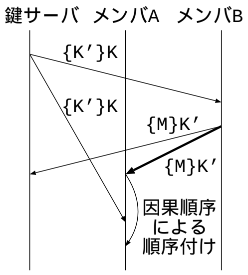
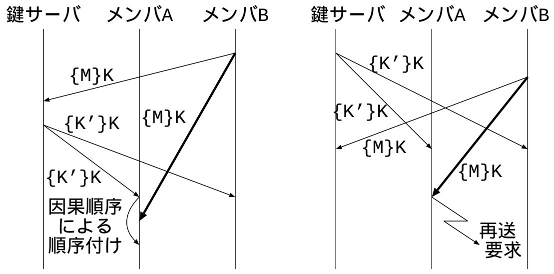


図 3: 因果順序性を考慮したマルチキャスト通信の例 (鍵が遅延する場合)



(a) 鍵サーバはデータを受信する場合

(b) 鍵サーバもデータを受信しない場合

図 4: 因果順序性を考慮したマルチキャスト通信の例 (データが遅延する場合)

ることにより，そのデータのアプリケーションへの配送を新しい鍵の受信後に遅らせることができる．

図 4(a) は，メンバ A に対するメンバ Bからのデータが遅延したことにより，メンバ A において既に新しい鍵に更新してしまった後に古い鍵で暗号化されたデータを受信する場合である．この場合は，鍵の更新メッセージに因果順序を付加することにより，メンバ Bからのデータを受信するまでメンバ内の鍵の更新を遅らせることができる．

図 4(b) は，図 3，図 4(a) の場合とは異なり，鍵の更新メッセージとメンバ Bからのデータに互いの因果順序を付加することができないため，メンバ B からの古い鍵で暗号化されたデータを受信する際に因果順序を考慮して鍵の更新を遅らせることができない．この場合にのみ，復号化できなかったデータの再送が必要となる．

以上により，どのような場合でも鍵の一貫性を保証でき，かつサービスが停止しない機構が実現できる．

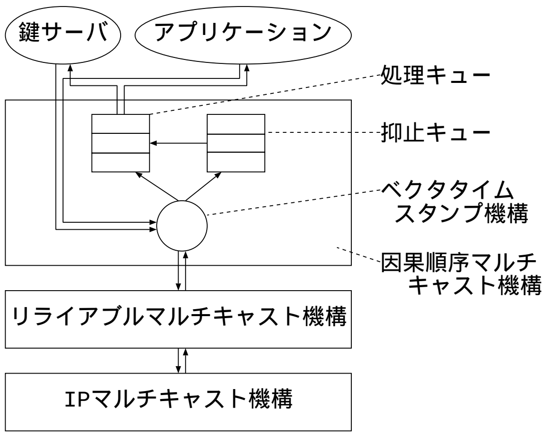


図 5: 提案する鍵配布機構

4 提案する鍵配布機構

本稿で提案するマルチキャストグループは，グループの共有鍵の管理を行なう鍵サーバとグループのメンバとから構成される．鍵サーバは，メンバの参加，離脱時に鍵を更新し，マルチキャスト通信によりメンバに配布する．このマルチキャスト通信は信頼性のあるものである必要があるので，ネットワークが信頼性のないものである場合にはSRM[4] などを用いる．

因果順序性はリライアブルマルチキャスト通信よりも上位層で提供する．因果順序の実現は，各データに含まれるベクタタイムスタンプを検査することにより行なう．

図 5に本稿で提案する鍵配布機構を示す．まずIP マルチキャスト通信の上位にマルチキャスト通信に信頼性を保証するリライアブルマルチキャスト機構があり，その上位に因果順序性を提供する因果順序マルチキャスト機構がある．そしてその上位に鍵サーバと各マルチキャストアプリケーションが位置する．

マルチキャストデータを送信する場合には，鍵サーバ，マルチキャストアプリケーションともにデータを因果順序マルチキャスト機構内のベクタタイムスタンプ機構に配送する．ベクタタイムスタンプ機構ではタイムスタンプをマルチキャストデータに挿入し，リライアブルマルチキャスト機構に配送する．

マルチキャストデータを受信した場合には，データは全てベクタタイムスタンプ機構に配送される．そこでタイムスタンプを検査し，そのデータよりも因果順序が前のデータを全て受信している場合には，そのデータを処理キューに配送する．そう

でない場合には抑止キューに配送し，因果順序が満足されるまで待機させ，その後処理キューに配送する．処理キューでは，FIFO 方式で鍵サーバあるいはアプリケーションにデータを配送する．

以下で各モジュールの詳細について解説する．

4.1 リライアブルマルチキャスト機構

本稿で提案する鍵配布機構では信頼性が保証されているマルチキャスト通信が必要である．なぜならば，一部のマルチキャストデータあるいは配布される鍵が喪失してしまった場合，因果順序が保証できなくなるからである．

IP マルチキャストでは信頼性が保証されていないため，データが確実に配送されるという，最低限の信頼性を保証する機構が必要となる．このようなリライアブルマルチキャスト機構はいくつか提案されているが，本稿では，SRM[4] を採用する．

4.2 因果順序マルチキャスト機構

因果順序マルチキャスト機構は，出力データにベクタタイムスタンプを挿入し，入力データのベクタタイムスタンプを検査するベクタタイムスタンプ機構，入力データの因果順序が満足されるまで鍵サーバあるいはマルチキャストアプリケーションへのデータの配送を遅らせる抑止キュー，鍵サーバあるいはマルチキャストアプリケーションへデータを配送するための処理キューにより構成される．

ベクタタイムスタンプ機構は，入力データ，出力データにより処理が異なる．以下ではプロセス p_iのベクタタイムスタンプを $VT(p_i)$ と表記し，プロセス p_iのベクトルの要素を $VT(p_i)[i]$ で表す．また，送信されるデータに挿入するベクタタイムスタンプを $VT(m)$ と表記する．なお，本稿で用いるベクタタイムスタンプは [1] に基づくものとする．

以下にベクタタイムスタンプの更新アルゴリズムを示す．

1. プロセス起動時に $VT(p_i)$ の全要素を 0 に初期化
2. 送信時に $VT(p_i)[i]$ を 1 増加
3. 送信データに $VT(p_i)$ を挿入 (これを $VT(m)$ とする)
4. p_iからのデータを p_jが受信した場合には (1) 式に従い $VT(p_j)$ を更新 (ただし n をメンバ数とすると，$\{\forall k : k \in n\}$)

$$VT(p_j)[k] = \max(VT(p_j)[k], VT(m)[k]) \quad (1)$$

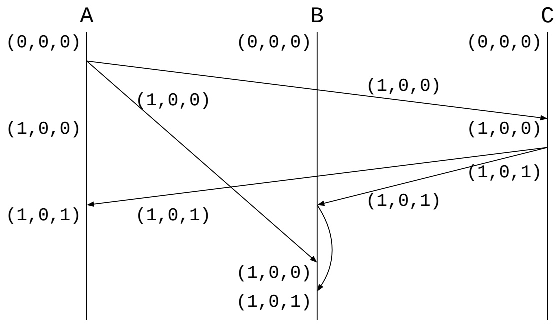


図 6: ベクタタイムスタンプの例

ベクタタイムスタンプ機構では，マルチキャストデータの受信時に (2) 式の条件を満足していない場合には抑止キューにマルチキャストデータを配送する．条件を満足している場合には，処理キューに配送し，ベクタタイムスタンプを (1) 式に従い更新する．

$$\begin{cases} VT(m)[k] = VT(p_j)[k] + 1 & k = i \\ VT(m)[k] \leq VT(p_j)[k] & k \neq i \end{cases} \quad (2)$$

ただし，n をメンバ数とすると，$\{\forall k : k \in n\}$ である．

図 6にベクタタイムスタンプによる因果順序の実現例を示す．A，B，Cの 3 つがグループのメンバであり，A がデータを送信した場合，`(1,0,0)` が $VT(m)$ としてデータに挿入され，自分の $VT(A)$ も`(1,0,0)` に更新する．そしてそれを Cが受信すると，自分の $VT(C)$ と $VT(m)$ を比較して，(2) 式を満足するので，処理キューにデータを配送し，$VT(C)$ を`(1,0,0)` と更新する．その後，Cがデータを送信したとすると，`(1,0,1)` が $VT(m)$ としてデータに挿入される．そのデータを受信した Bでは，(2) 式を満足しないので，データを抑止キューに配送する．そして，$VT(m)$ が`(1,0,0)` である A からのデータを受信した時点で (2) 式が満足されるので，処理キューに配送する．そして $VT(B)$ が`(1,0,0)` に更新され，(2) 式を満足するようになった時点で，Cからのデータを抑止キューから処理キューに配送する．これにより，A からのデータが Cからのデータよりも遅れた場合に因果順序を満足できる．

ベクタタイムスタンプの要素数はグループのメンバシップが変わるたびに変化する．すなわち，メンバが新しく参加する場合にはベクタタイムスタンプの要素数は 1 大きくなり，離脱する場合には 1 小さくなる．ただし，離脱したメンバが使用していたフィールドを他のメンバに再割り当てすると

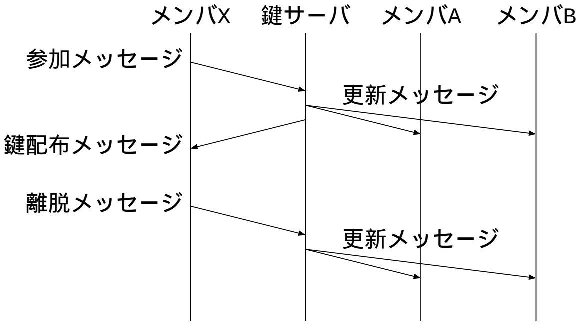


図 7: メンバの参加と離脱

きにはセキュリティ上の問題が生じない様に注意しなければならない．各メンバはメンバシップの変化を鍵の更新メッセージから知ることができる．

4.3 鍵サーバ

鍵サーバは，鍵の生成，更新，グループのメンバシップの管理を行なうものであり，各グループに 1 つ存在する．鍵の更新はメンバの参加，離脱時に必要となる．図 7にその例を示す．

メンバ Xがグループに参加したい場合には，鍵サーバに対して参加メッセージをユニキャストで送信する．参加メッセージを受信することにより，鍵サーバはグループの新しい鍵を生成し，ベクタタイムスタンプについての情報とともに，現在参加している各メンバに現在のグループの共有鍵で暗号化してマルチキャストで送信する．これを参加メッセージに対する鍵の更新メッセージとする．また，新しいメンバ Xには，Xの公開鍵を用いてグループの共有鍵を暗号化して，ベクタタイムスタンプについての情報とともに，ユニキャストで送信する．これを鍵配布メッセージとする．

メンバ Xがグループから離脱する場合には，鍵サーバに対して離脱メッセージをユニキャストで送信する．Xからの離脱メッセージを受信することにより，鍵サーバはグループの新しい共有鍵を生成し，それを各メンバの公開鍵で暗号化したものを全メンバ分まとめたものを，ベクタタイムスタンプについての情報とともに，マルチキャストで送信する．以上により，離脱したメンバによる新しい共有鍵の不正な取得を防止できる．

以下に参加メッセージ，離脱メッセージ，鍵の更新メッセージ，鍵配布メッセージを示す．ただし，$group\ ID$ はグループ識別子，$source\ ID$ は送信者の識別子，$public\ key$ はそのメンバの公開鍵，$new\ key$ はグループの新しい共有鍵，$old\ key$ はグループの古い共有鍵，$VT\ infomation$ はベクタタイムスタンプについての情報を意味する．

- 参加メッセージ

$$(group\ ID, source\ ID, public\ key)$$

- 離脱メッセージ

$$(group\ ID, source\ ID)$$

- 参加メッセージに対する鍵の更新メッセージ

$$\{new\ key, VT\ infomation\}_{old\ key}$$

- 離脱メッセージに対する鍵の更新メッセージ

$$\sum\{new\ key, VT\ infomation\}_{public\ key}$$

- 鍵配布メッセージ

$$\{new\ key, VT\ infomation\}_{public\ key}$$

4.4 マルチキャストアプリケーション

マルチキャストアプリケーションは，マルチキャストグループ参加時に鍵サーバに参加メッセージを送信する必要があり，また離脱するときも離脱メッセージを送信する必要がある．さらに各プロセス毎に公開鍵と秘密鍵を保持する必要がある．しかし，それらの点を除けば，通常のマルチキャストアプリケーションをそのまま用いることができる．

5 検討事項

本章では，本稿で提案した機構について検討を行なう．

5.1 実装についての検討

本稿で提案した鍵配布機構の実装方針として，2 つの方法が考えられる．1 つは，リライアブルマルチキャスト機構と因果順序マルチキャスト機構をデーモンとしてアプリケーション層に実装する方法であり，もう 1 つはこれらをカーネル内に実装する方法である．

デーモンとして実装する場合には，鍵サーバ，アプリケーションとカーネルとの間でデータを横取りし，マルチキャスト通信に信頼性と因果順序を提供する必要がある．

カーネル内に実装する場合には，図 5に示した階層構造をカーネル内で実現でき，シンプルな実装となる．

したがって，今後は本機構をカーネル内に実装していく予定である．

5.2 本機構の有効性

本稿で提案した機構は，マルチキャストグループのメンバシップの変化が激しく，かつリアルタイム性を要求するアプリケーションにおいて特に有効であるだろう．このような性質を持つアプリケーションとしては，vic[5] 等に代表されるビデオ会議システムが挙げられる．このようなアプリケーションにおいては，本機構の導入によって回避可能なデータの再送および不必要なサービスの停止時間の減少が予想される．実際にどの程度の性能の向上が見られるかについては，今後評価を行なう．

5.3 規模拡張性

本機構では，規模拡張性のための枠組みは特に設けていない．したがって，グループの規模が大きくなった場合に規模拡張性の問題が生じる．

まず，各ホストのベクタタイムスタンプの要素数は，グループの規模に比例することが挙げられる．したがって，グループの規模が大きくなるほど，各ホストで必要となる空間計算量とデータ中のベクタタイムスタンプが占める割合が大きくなってしまう．

また，鍵サーバの負荷もグループの規模に比例する．したがって，鍵サーバもマルチキャストデータを受信すべきかが問題となる．しかし，配布する鍵とマルチキャストデータとの因果順序を定めるためには，鍵サーバもマルチキャストデータを受信する必要がある．鍵サーバは 1 つのグループしか管理しないので，マルチキャストデータを受信することにより鍵サーバが受ける負荷は，メンバがマルチキャストデータを受信する負荷と等しいので，この点はさほど問題とはならない．

さらに，離脱メッセージに対する鍵の更新メッセージの大きさもグループの規模に比例する．

したがって，規模の大きいグループで本機構を用いる場合には，なんらかの枠組みが必要となるだろう．規模拡張性を提供するためには，Iolus[7] の枠組みが有効であると思われる．

6 まとめ

本稿では，マルチキャストグループでの共有鍵の更新時における一貫性を保証するために，データの因果順序性を考慮した鍵配布機構を提案した．本機構により鍵の一貫性を保証でき，また可能な限り再送を避けるので，リアルタイム性も提供できると考えられる．

今後は，本機構の実装を完成させ，本機構を導入することによる性能の向上についての評価を行なっていく予定である．

参考文献

[1] A. Schiper et al. A new algorithm to implement causal ordering. In *3rd International Workshop on Distributed Algorithms*, Springer–Verlag, 1989.

[2] K. Birman et al. Lightweight Causal and Atomic Group Multicast. In *ACM Transactions on Computer Systems*, ACM, 1991.

[3] M. Steiner et al. CLIQUES: A New Approach to Group Key Agreement. In *ICDCS '98*, IEEE, 1998.

[4] S. Floyd et al. A Reliable Multicast Framework for Light–weight Sessions and Application Level Framing. In *SIGCOMM '95*, ACM, 1995.

[5] S. McCanne et al. vic: A Flexible Framework Framework for Packet Video. In *ACM Multimedia '95*, ACM, 1995.

[6] W. Diffie et al. New Directions in Cryptograpy. In *Transactions on Information Theory*, IEEE, 1976.

[7] S. Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *SIGCOMM '97*, ACM, 1997.

[8] 朴 美娘 他. セキュアマルチキャスト通信方式に関する検討. In *DiCoMo '98*, 情報処理学会, 1998.