卒業論文

題目

マルチキャスト通信のための鍵交換デーモンの拡張

指導教授

所 真理雄 教授

慶應義塾大学

理工学部 電気工学科

1997年度

69408091

濱野 淳史

論文要旨

学科	電気工学	学籍番号	69408091	氏名	濱野 淳史

(論文題目)

マルチキャスト通信のための鍵交換デーモンの拡張

近年 OS レベルでのセキュリティの強化がいくつか提案されており，そのうち IP 層での暗号通信機構である IPsec に関する研究が現在盛んに行なわれている． IPsec では通信内容を保護するために通信者間で鍵を共有する必要がある． 通信者間で秘密裡に鍵を交換し共有するためのプロトコルとして ISAKMP/Oakley が規定されており, 現在標準化が進められているが, 現行の ISAKMP/Oakley では 1 対 1 の通信における鍵交換のみしか扱えない．このため, 1 対多の通信が発生するマルチキャスト通信においては, 多数の通信者が同一の鍵を共有しなければならず, 現行の ISAKMP/Oakley では不十分である．

そこで本研究では, マルチキャストグループ内での鍵配布が可能となるように ISAKMP/Oakley の拡張を行なう．本研究で提案する機構は, グループの鍵の生成配布を行なう鍵サーバと, グループのメンバであるクライアントから構成され, 以下の 4 つの特徴をもつ：(1) マルチキャストグループ毎に鍵サーバを設け, 鍵サーバのみが鍵配布を行なうので, 全てのメンバが同一の鍵を共有することが可能となる．(2) クライアントが鍵サーバに鍵を要求する際には, クライアントは要求メッセージをマルチキャストアドレス宛に送信する．このため, クライアントは鍵サーバの存在を意識する必要がない．(3) クライアントがグループへの join と同時に鍵の要求を行うことにより, 動的なメンバの変更に対応できる．(4) 鍵配布にアクセス制御機構を導入しているため, 鍵サーバは正当なメンバのみに鍵を配布することができる．

本研究で提案した鍵交換デーモンの拡張法により, マルチキャスト通信で IPsec を利用する場合の鍵交換を自動化できるので, 暗号化および認証を考慮したマルチキャスト通信が可能となる．

慶應義塾大学理工学部

◎ 内容の要旨は約 600 字

目 次

図 目 次

表 目 次

第 1 章

序論

1.1 背景および目的

近年のパーソナルコンピュータやインターネットの普及により, 一般家庭や企業においてもインターネットという言葉が聞かれるようになった. それに伴い, 様々なアプリケーションが開発されている. その例として, インターネットを介した映画配送システムや, 自宅に居ながらにして多数の人々と会議ができるビデオ会議システム等が挙げられる. このようなアプリケーションは, 一対多通信であるマルチキャスト通信により実現することが効率的であると考えられている. ところが, このようなアプリケーションを実現する場合にはセキュリティの面で問題が生ずる. 現在のインターネットではユニキャスト, マルチキャストを問わず, ネットワーク上を暗号化されていないデータが通過するために第三者が盗聴や改竄を容易に行うことが可能である. したがって, 例えば映画配送システムの場合には料金を支払った正規のユーザ以外の者にもデータを受信されてしまう恐れがある. また, ビデオ会議システムの場合にはある企業の会議内容がライバル企業に盗聴や改竄される恐れがある. そのため, これらのアプリケーションを安全に実現するためにはインターネット上を通過するデータを暗号化し, またデータの正当性を保証することが必要である.

これまで, 通信における暗号化や認証などのセキュリティの強化は, 各アプリケーション毎において行なわれることが多かったが, 近年 OS レベルで各通信におけるセキュリティの強化を行なうことが提案されている. そのうち, IP 層でのセキュリティの強化を目的とした IPsec [Atk95c] に関する研究は現在盛んに行なわれており, 標準化のための努力がなされている. IPsec では, データの改竄と偽造を防ぐための認証ヘッダ [Atk95a] と, 改竄と偽造, 盗聴を防ぐための暗号化ペイロード [Atk95b] の 2 つが規定されている.

一般に暗号化や認証のためには送信者と受信者の間で暗号鍵を秘密裡に共有することが必要である. IPsec では, 暗号化や認証のための方法を標準化しているが, 暗号化や認証のために用いられる暗号鍵の交換方法については規定していない. その方法の 1 つとしてまず管理者が手動で設定する方法が考えられるが, 不特定多数の相手と通信することを考慮した場合, この方法は現実的ではない. そこで動的に鍵の共有を確立する方法がいくつか提案されている. IPsec における暗号化や認証に必要な鍵を動的に確立する機構としては ISAKMP/Oakley [HC97] が提案されている. ところが ISAKMP/Oakley の現在の仕様では, 1 対 1 の通信のための鍵交換方法のみが規定されており, 1 対多の通信のための鍵交換方法は規定されていない. したがって, IPsec を用いたマルチキャスト通信を行なうためには手動で鍵を設定しなければならない.

そこで本研究では, マルチキャストのための動的鍵配布が可能となるように鍵交換プ

ロトコルの拡張を行ない, その有効性を示すために, 鍵交換デーモンに拡張をほどこす形で実装を行なう. 本研究で提案する機構を実現することによりアプリケーションを変更することなく, IPsec を用いたマルチキャスト通信が可能となる.

1.2 本論文の構成

本論文は全部で 6 章より構成されている. 2 章で, 既存の鍵配布機構について概説し, 比較および検討を行なう. 3 章で, IPsec を用いるマルチキャスト通信に必要な鍵配布機構のモデルを提案し, 実装方法を述べる. 4 章で, 3 章において実装したモデルの評価を行ない, 評価結果について考察する. 5 章で今後の課題について述べ, 6 章で本研究の結論を述べる.

第 2 章

既存の鍵配布機構

本章では, 既存の鍵配布機構である ISAKMP/Oakley, GKMP, SMKD, Iolus について概説し, それらについて比較, 検討を行なう.

2.1 動的な鍵配布の必要性

安全な通信を行なうためにはホスト間で, 認証や暗号化を行なうための鍵をあらかじめ共有しておく必要がある. 鍵を共有する最も単純な方法は, 管理者が手動で鍵を設定する方法である. この方法は同一 LAN 内といった限られた範囲内では効率的であるが, 不特定多数のホストと通信を行なう場合は現実的な方法とはいえない. そこで, 不特定多数のホストと動的に鍵を交換する機構が必要となる.

1 章で例に出した映画配送システムやビデオ会議システムを実現する場合, セキュリティを考慮したマルチキャスト通信は不可欠である. しかし, マルチキャスト通信のための鍵配布は, ユニキャスト通信のための鍵の単なる拡張であると考えてはならない. 例えば, メンバ数が n のグループでマルチキャスト通信を行なう場合, 各ホストは全てのメンバとの $n-1$ 個の鍵を保持しなければならない (図 2.1). この方法では, メンバ数の増加に伴い効率が悪くなってしまうことは明かである. そのため, マルチキャストグループへの動的な鍵配布に関する研究がいくつか行なわれている.

2.2 ISAKMP/Oakley

動的な鍵交換の機構として, 長い間 **Photuris** [KS97b], [KS97a] と **SKIP** [AMP96] が提案されていた. SKIP は IP データグラムのオプション (IPv6 [DH95] では拡張ヘッダ) として定義された帯域内の鍵交換プロトコルであり, これに対し, Photuris は UDP を用いた帯域外の鍵交換プロトコルであった. しかし, これらの提案は取り下げられ, 代わりに **ISAKMP/Oakley** [HC97] を IPv6 での必須の鍵交換プロトコルとすることが IETF で決定された.

ISAKMP/Oakley は, IPsec で必要な鍵を動的に確立する機構であり, 鍵交換の枠組を提供する ISAKMP (Internet Security Association and Key Management Protocol) [MSTS97] と, ISAKMP 上で実際の鍵管理機能を定義している Oakley [Orm96] とから成り立っている.

ISAKMP/Oakley は, ISAKMP/Oakley のための SA を確立する **Phase 1** と, IPsec における AH や ESP など ISAKMP/Oakley 以外の SA を確立する **Phase 2** からなる. Phase 1 には **Main Mode** と **Aggressive Mode** の 2 つがあり, Phase 2 には **Quick Mode** の 1 つがある.

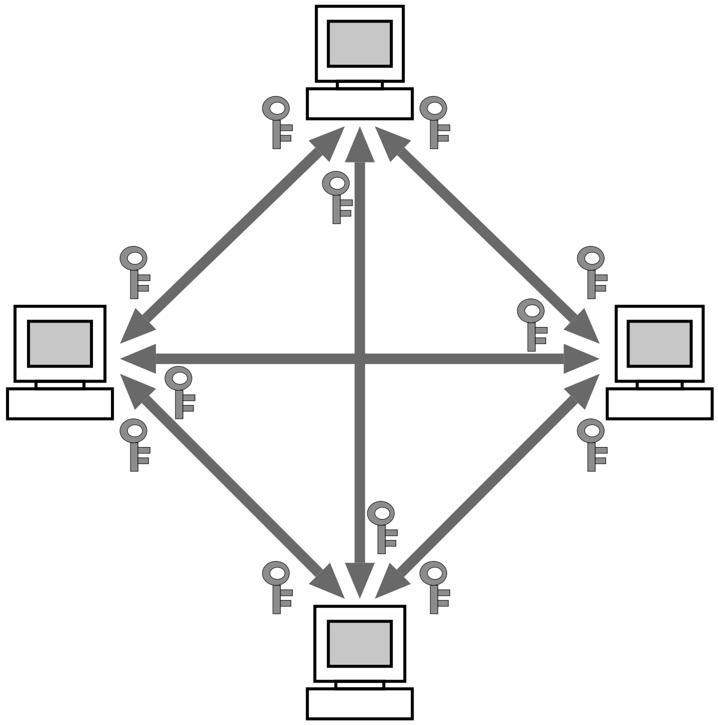

図 2.1: メンバ数が 4 の場合のマルチキャスト通信の例

本節では, これらの処理について概説する.

2.2.1 表記法

始めに, 本節で用いる表記法について述べる.

ISAKMP SA ISAKMP/Oakley のためのセキュリティ・アソシエーションを表す.

SA ISAKMP/Oakley 以外のセキュリティ・アソシエーションを表す.

HDR ISAKMP のヘッダを表す.

HDR* ISAKMP のヘッダを表す. ペイロードが暗号化されている.

SA ISAKMP SA または SA の情報を含むペイロードを表す.

<p>_b ペイロード <p> の本体を表す.

$\mathbf{SA}_i$_b SA ペイロードの本体を表す.

CKY_I, CKY_R Initiator と Responder のクッキー (疑似乱数) を表す. ISAKMP ヘッダのなかに入る.

g^{xi}**, g**^{xr} Initiator と Responder の Diffie–Hellman の公開値を表す.

g^{xy} Diffie–Hellman において共有される秘密値を表す.

KE 鍵交換ペイロードを表す.

N$_i$**, N**$_r$ Initiator と Responder の乱数を含んだペイロードを表す.

ID$_i i$**, ID**$_i r$ Initiator と Responder の ID ペイロードを表す. Phase 1 で用いる.

ID$_u i$**, ID**$_u r$ Initiator と Responder の ID ペイロードを表す. Phase 2 で用いる.

SIG 署名ペイロードを表す.

CERT 証明書ペイロードを表す.

HASH ハッシュペイロードを表す.

$prf(key, msg)$ 鍵付き疑似乱数発生関数を表す.

SKEYID 交換を行なっているホストにしかわからない秘密値から導出される記号列を表す.

SKEYID$_e$ 暗号用の鍵を表す.

SKEYID$_a$ 認証用の鍵を表す.

SKEYID$_d$ SA で用いられる鍵の種を表す.

<x>y x が鍵 y によって暗号化されていることを表す.

| 連結を表す.

[x] x がオプションであることを表す.

2.2.2 Phase 1

Phase 1 では, 通信相手の認証を行ない, ISAKMP SA を確立する. ISAKMP SA を確立するときに, ID の保護が可能な Main Mode と ID の保護が不可能な Aggressive Mode の 2 種類がある.

Main Mode では 6 回の通信が行なわれる. 始めの 2 回の通信で使用するアルゴリズム等の取り決めが行なわれ, 次の 2 回の通信で鍵交換に必要な Diffie–Hellman 鍵交換アルゴリズム [DH76] の公開値や乱数などの補助的なデータの交換が行なわれる. そして最後 2 回の通信で生成された鍵の認証が行なわれる.

Aggressive Mode では 3 回の通信が行なわれる. 始めの 2 回の通信において使用するアルゴリズム等の取り決めと鍵交換に必要な Diffie–Hellman の公開値や乱数などの補助的なデータの交換が行なわれる. さらに 2 回目の通信では, 返答者の認証も行なわれる. そして 3 回目の通信において始動者の認証と生成された鍵の認証が行なわれる. この 3 回目の通信が, 生成される ISAKMP SA により暗号化されないため ID の保護が不可能となる.

Main Mode, Aggressive Mode では, 電子署名 (以後 Signature と記す), 公開鍵暗号 (以後 Public Key Encryption と記す), あらかじめ共有鍵を所持する方法 (以後 Pre–shared Key と記す) の 3 種類の認証方法が利用可能である. 各認証方法では, 次の各式により計算される SKEYID の値が使用される.

- Signature 方式

$$SKEYID = prf(\ N_i\_b \mid N_r\_b,\ g^{xy}\ ) \tag{2.1}$$

- Public Key Encryption 方式

$$SKEYID = prf(\ hash(\ N_i\_b \mid N_r\_b\ ),\ CKY\_I \mid CKY\_R\ ) \tag{2.2}$$

- Pre–shared Key 方式

$$SKEYID = prf(\ pre\_shared\_key,\ N_i\_b \mid N_r\_b\ ) \tag{2.3}$$

Main Mode, Aggressive Mode の結果, 式 (2.4) で表される鍵の種と, 式 (2.5) で表される認証用の鍵, 式 (2.6) で表される暗号用の鍵が生成される.

$$SKEYID_d = prf(\ SKEYID,\ g^{xy} \mid CKY\_I \mid CKY\_R \mid 0\ ) \tag{2.4}$$

$$SKEYID_a = prf(\ SKEYID,\ SKEYID_d \mid g^{xy} \mid CKY\_I \mid CKY\_R \mid 1\ ) \tag{2.5}$$

$$SKEYID_e = prf(\ SKEYID,\ SKEYID_a \mid g^{xy} \mid CKY\_I \mid CKY\_R \mid 2\ ) \qquad (2.6)$$

認証に用いられるハッシュ値は, 始動者の認証には式 (2.7), 返答者の認証には式 (2.8) により計算される値が用いられる.

$$HASH_I = prf(\ SKEYID,\ g^{xi} \mid g^{xr} \mid CKY\_I \mid CKY\_R \mid SA_i\_b \mid ID_{ii}\_b\ ) \qquad (2.7)$$

$$HASH_R = prf(\ SKEYID,\ g^{xr} \mid g^{xi} \mid CKY\_R \mid CKY\_I \mid SA_i\_b \mid ID_{ir}\_b\ ) \qquad (2.8)$$

以下で各認証方式での処理の流れを概観する.

Signature による認証方式

Signature による認証方式では, 互いに獲得できるハッシュ値を用いて署名することにより認証を行なう. Main Mode における処理の流れを図 2.2 に示す. また Aggressive Mode における処理の流れを図 2.3 に示す.

各 Mode において, SIG_I, SIG_R は, HASH_I, HASH_R に電子署名アルゴリズムを適用して得られる署名されたデータである. また, オプションとして 1 つ以上の証明書ペイロードを送ることができる.

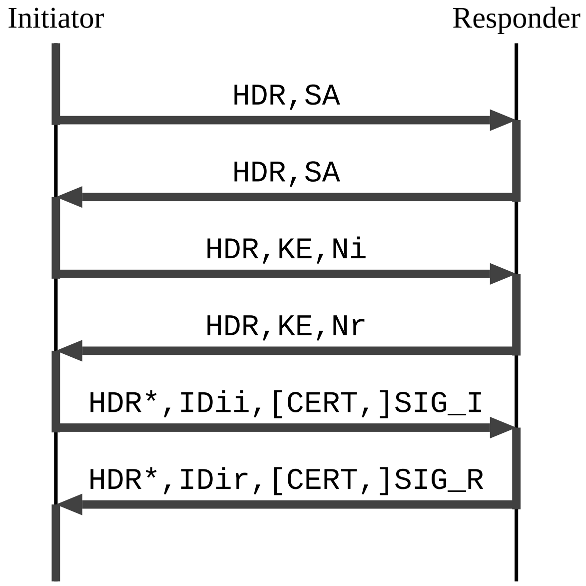

図 2.2: Main Mode with Signature

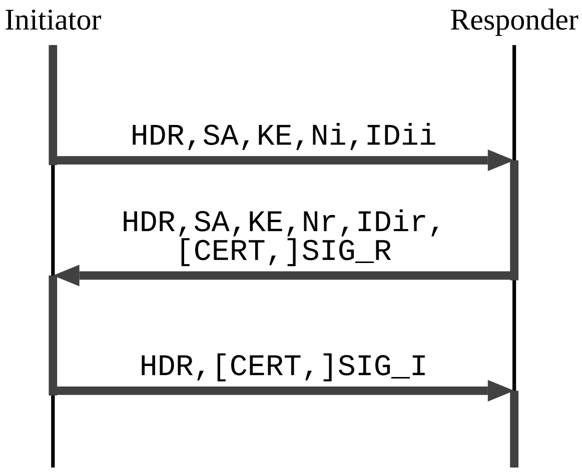

図 2.3: Aggressive Mode with Signature

Public Key Encryption による認証方式

Public Key Encryption による認証方式では, 暗号化して交換される乱数を用いてハッシュ値を計算することにより認証を行なう. Main Mode における処理の流れを図 2.4 に示す. また Aggressive Mode における処理の流れを図 2.5 に示す.

Main Mode では 3 回目, 4 回目の通信において, Aggressive Mode では 1 回目, 2 回目の通信において, 乱数の他に ID が相手の公開鍵を用いて暗号化され, 交換される. HASH(1) は, 始動者が乱数と ID を暗号化するために用いる証明書のハッシュ値である.

なお Public Key Encryption を認証方法とする場合は相手の公開鍵を用いて暗号化を行なうので, Aggressive Mode においても ID の保護が可能である.

Pre–shared Key による認証方式

Pre–shared Key による認証方式では, 帯域外で導出される鍵を認証に用いる. Main Mode における処理の流れを図 2.6 に示す. また Aggressive Mode における処理の流れを図 2.7 に示す.

Main Mode では, 始動者が ID_{ir} を処理する前に HASH_I が計算されなければならないため, 相手の IP アドレスのみから鍵が計算される. また, あらかじめ相手との鍵を共有しているため, SA を確立するために交換するメッセージは, 他の認証方法に比べて簡単なものになる.

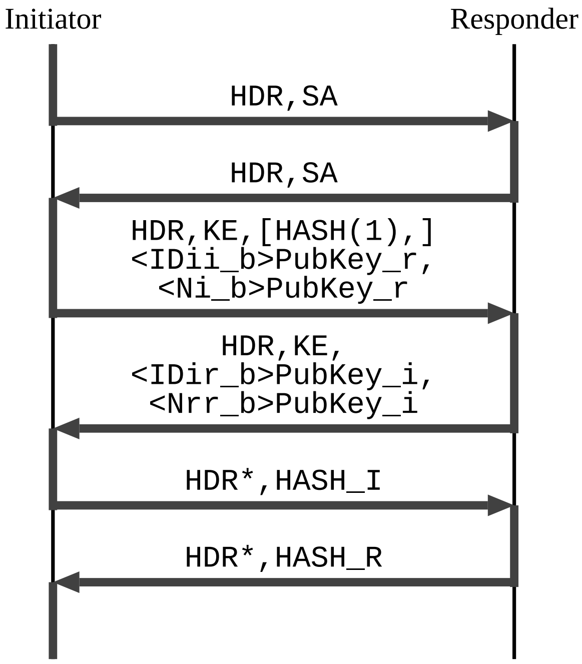

図 2.4: Main Mode with Public Key Encryption

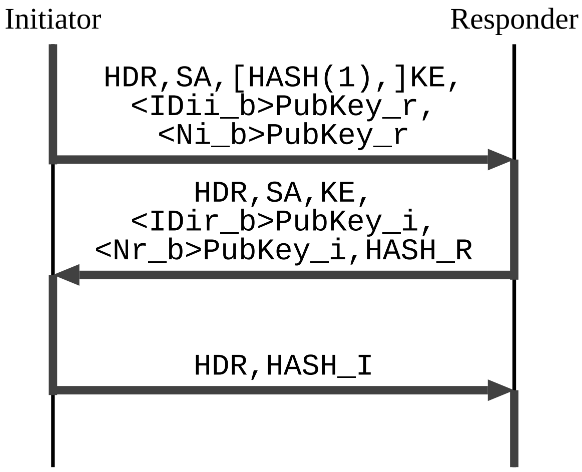

図 2.5: Aggressive Mode with Public Key Encryption

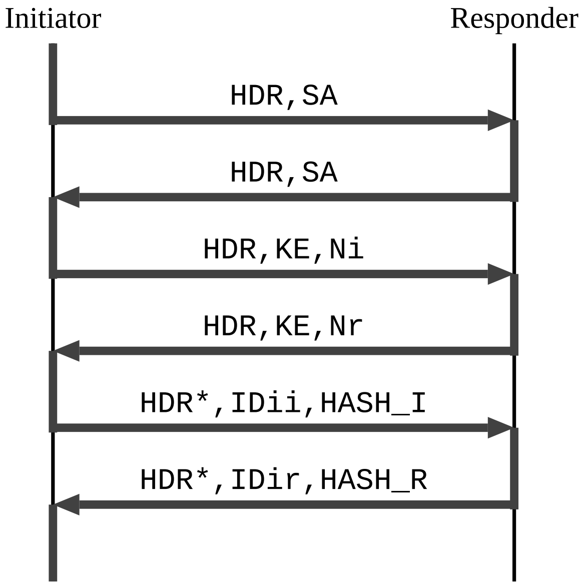

図 2.6: Main Mode with Pre–shared Key

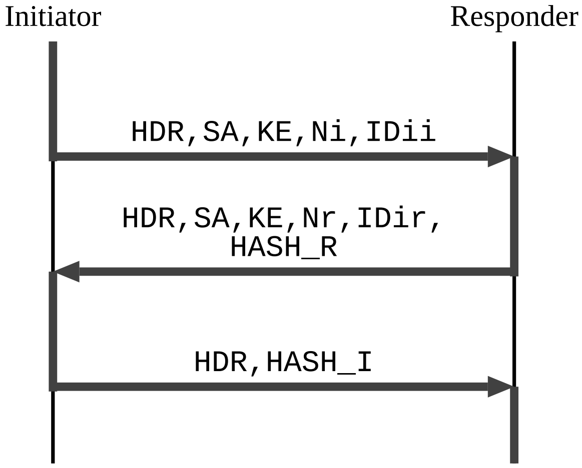

図 2.7: Aggressive Mode with Pre–shared Key

2.2.3 Phase 2

Phase 2ではQuick Modeの1つが定義されており, 実際の通信で用いられるSAを確立する. Quick Modeで交換されるメッセージのペイロードはISAKMP SAを用いて暗号化され, 保護される.

Quick Modeでは3回の通信が行なわれる. 始めの2回の通信でSAを確立するためのアルゴリズム等の取り決めが行なわれ, 3回目の通信で生成された鍵の認証を行なう. 交換されるメッセージは, ISAKMPヘッダの次にハッシュペイロードが続き, その後にSAペイロードが続くものである. このハッシュペイロードは, メッセージの認証と完全性を保証する.

Quick Modeでの処理の流れを図2.8に示す.

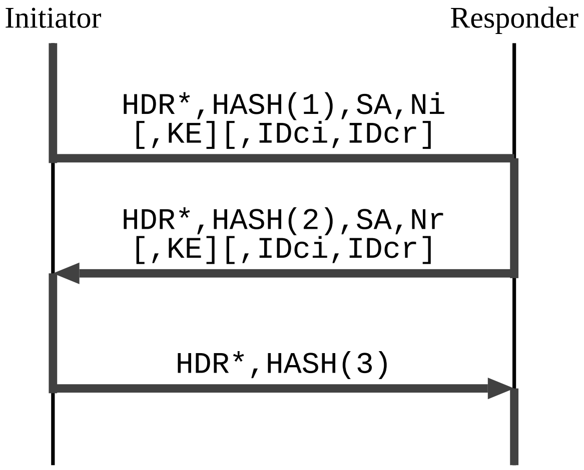

図2.8: Quick Mode

ここで, HASH(1)はISAKMPヘッダ中のメッセージID (M_ID)と, 暗号化のためのパディングを除いた全てのメッセージをprf (疑似乱数生成関数)に適用したものである. HASH(2)はN_i, N_rを除いてHASH(1)とほぼ同じものである. HASH(3)は, 1オクテット分の0とISAKMPヘッダ中のメッセージID, そしてN_i, N_rの本体をprfに適用したものである. これらは, 式(2.9) ~ (2.11)により表される.

$$HASH(1) = prf(\ SKEYID_a,\ M\_ID \mid SA \mid N_i\ [\ \mid KE\ ]\ [\ \mid ID_{ci} \mid ID_{cr}\ ]\ ) \tag{2.9}$$

$$HASH(2) = prf(\ SKEYID_a,\ M\_ID \mid N_i\_b \mid SA \mid N_r\ [\ \mid KE\ ]\ [\ \mid ID_{ci} \mid ID_{cr}\ ]\ ) \tag{2.10}$$

$$HASH(3) = prf(\ SKEYID_a,\ 0 \mid M\_ID \mid N_i\_b \mid N_r\_b\ ) \tag{2.11}$$

また両者で共有する鍵は式 (2.12) により求められる.

$$KEY = prf(\ SKEYID_d,\ [\ g(qm)^{xy}\ |\ ]\ protocol\ |\ SPI\ |\ N_i\_b\ |\ N_r\_b\ ) \tag{2.12}$$

ここで, $g(qm)^{xy}$ は Diffie–Hellman 鍵交換アルゴリズムによって算出された共有値である.

2.3 Group Key Management Protocol

マルチキャスト通信のための鍵配布機構として, **Group Key Management Protocol** (GKMP) [HM97b], [HM97a] が提案されている.

GKMP では, マルチキャストグループの鍵生成, メンバへの配布, 鍵のアクセス制御の保証, 危険にさらされているホストのアクセス拒否, グループに対しての階層的な制御等を行なう. GKMP で用いられる鍵は, 2 つのホスト間で協調的に確立される. 鍵の確立には, 公開鍵暗号である RSA [RSA78], Diffie–Hellman [DH76], 楕円暗号 [Kob87], [Mil86] 等のアルゴリズムを採用している. 鍵は正当なメンバ全てに配布されるが, 配布の過程において全てのホストは認証される.

GKMP では, 隣接するメンバ同士の認証の過程が規定されている. プロトコルを実装したホストは, グループの鍵を配布する過程の一部として許可証明書 (permission certificates) を配布する. 許可証明書は, ホストより上位の機関 (例えば, 各サイトの管理者) により署名され割り当てられた, そのサイトへのアクセス制御情報を含んでいる. したがって, 各ホストは他のホストを認証することはできるが, アクセス情報を更新することはできない. 各ホストは, 他のホストからあるサービスの要求を受けたら, 許可証明書により, 要求を出したホストがそのサービスを受けることができる正当なホストであるかを確認し, 不当なホストであると判断された場合にサービスを拒否する.

GKMP では, 危険にさらされているホストのリスト (Compromise Recovery List, CRL) が鍵管理のメッセージと共に各メンバに配布される. このリストにより, 各メンバは危険にさらされているホストとのコネクションを切断する. GKMP では, CRL の配布を簡単にするために特定のグループコントローラにグループの管理を委任する. 鍵管理のメッセージに CRL のバージョンが含まれており, CRL の更新が認められたら新しい CRL をダウンロードする. CRL は上位の機関により署名されているため, 各ホストにおいて認証される.

GKMP はアプリケーション層のプロトコルであり, 下層の通信プロトコルに依存しない. しかしながら, マルチキャスト通信が利用可能ならばグループの鍵の再配布のためにそれを利用する.

グループ鍵の生成と配布の過程では, 鍵を配布するホストが必要となる. このホストは

グループコントローラと呼ばれ, 鍵を生成し, 鍵配布メッセージを送信し, 受信者からの鍵の受け取り確認を行なう. 全てのメンバはグループコントローラになる権限をもっており, 割り当てられたホストがグループコントローラの役割を果たす.

グループコントローラは鍵の無矛盾性を維持する. グループは同じ鍵を共有していなければならず, もしグループの一部が鍵を無くしたり, 不適切な変更を行なったりした場合, 正しい鍵で動作しているホストとのデータの送受信が行なえなくなる. したがって, 鍵の生成と変更には曖昧さが伴ってはならず, グループコントローラが鍵の同期を取らなければならない.

グループのメンバとは, グループコントローラの役割を割り当てられていない, グループのホストのことである. グループのメンバは, グループコントローラが鍵を生成する場合にはそれに協調する. また, コントローラに鍵を要求し, 受け取る. そして, 配布された CRL リストを管理し, 常に他のメンバに対して危険性の検査を行なう.

以下に GKMP での処理例を挙げる (図 2.9).

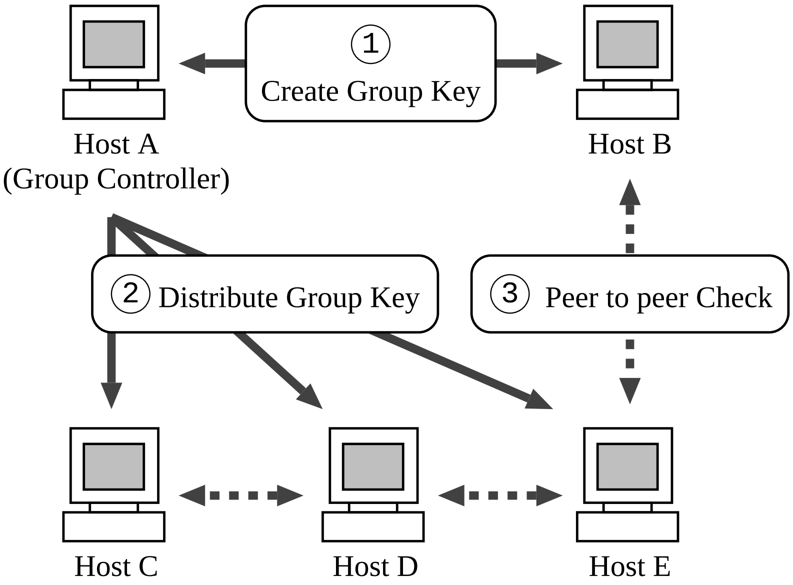

図 2.9: GKMP の例

1. 最初にグループのメンバとなったホスト (Host A) と次にメンバになったホスト (Host B) との間で協調してグループの鍵を生成する.

2. メンバにグループコントローラ (Host A) がグループの鍵や CRL を配布する.

3. 他のメンバが危険にさらされていないかを検査し, 発見した場合グループコントローラに伝える.

2.4 Scalable Multicast Key Distribution

Scalable Multicast Key Distribution (SMKD) [Bal96] は, **Core Based Trees** (CBT) [BFC93] に基づくマルチキャスト通信のための鍵配布機構である.

SMKD で提案するセキュリティ機構では, CBT マルチキャスト配送木への安全な join と, マルチキャストグループの鍵の配布の両方を規定している. マルチキャストの鍵配布は, 配送木への join プロセスのオプションとされており, グループの鍵の要求がなければ, 鍵の配布は省略される.

SMKD が提案するモデルは中央化された鍵配布局の存在に依存しない. さらに, SMKD が提案する鍵配布局は, 鍵配布のみを行なう必要はない. SMKD では, 各グループが自分のグループ鍵配布局をもっており, それが提供する機能は, 配送木に join してくる他のノードに委任することができる. ゆえに, SMKD では分散化された鍵の配布が可能となり, 規模拡張性が高くなる. また SMKD では, 専用の鍵配布局を必要としない. SMKD では, CBT のプライマリのコアがグループ鍵配布局の役割を果たす.

マルチキャストグループのアクセス制御リストは, CBT のグループが作られたときに, グループの一番最初のメンバ (グループ始動者) によって作成される. このリストは認証可能となるように, 電子署名されている方が望ましい. グループ始動者は, グループのプライマリコアに順々にアクセス制御リストを CBT を用いずにユニキャストで送信する. アクセス制御リストの電子署名により, 完全性が保証される. もし, グループのメンバシップそのものが他人に知られたくない情報ならば, 送信前にプライマリコアの公開鍵を用いて, 付加的にアクセス制御リストを暗号化することができる.

アクセス制御リストは, グループメンバシップ情報から成り, ホストベースでもユーザベースでもよいことになっている. さらに, CBT 配送木の一部となる, ネットワーク中のある “信頼された” ルータのみに鍵配布の権限を与えることができるように, 鍵配布の権限を制限することは望ましい. この場合, 信頼されたネットワークノードを構成するために付加的なアクセス制御リストが必要となる.

プライマリコアはアクセス制御リストを含むメッセージを受信し, 認証した後, グループで共有する鍵を生成する. プライマリコアは, 古い鍵の期限が切れる直前にグループの鍵を再配布するために用いられる, 鍵を暗号化する鍵も生成する.

アクセス制御リスト, グループの鍵, 鍵を暗号化する鍵は, セカンダリコアが配布木の

一部になるときに配布される. この情報をもつルータは join してくるメンバを認証できるため, 既に認証された配送木のルータ上では, 安全な配送木への join と, 鍵配布がプライマリコアの代わりに行なわれる.

2.5 Iolus

Iolus [Mit97] は規模拡張性のある安全なマルチキャスト通信の枠組を提供する.

マルチキャストグループでは, メンバシップを変更した場合にはグループ全てのメンバが影響を受けるため, 規模拡張性に乏しい. Iolus では, 単一の平面的で安全なマルチキャストグループの代わりに, 安全な配送木の概念を採用することにより, 規模拡張性の問題を解決する. 安全な配送木は, 階層的に配置された単一の仮想的で安全なマルチキャストグループを生成するために, 多数の安全な “サブグループ” に分割する (図 2.10).

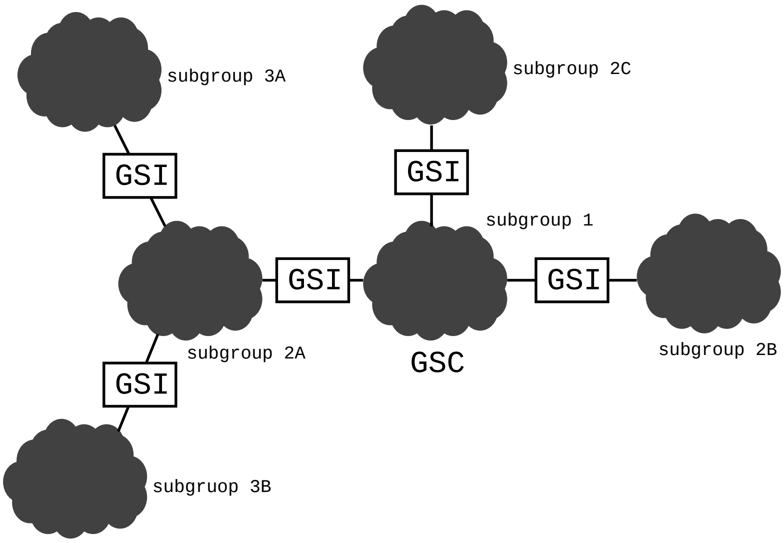

図 2.10: Iolus における安全な配送木の例

規模拡張性は, 各サブグループを比較的独立したものと考えることにより達成される. 安全な配送木の各サブグループは, それ自身がマルチキャストグループとマルチキャストアドレスをもっており, 適切なマルチキャストルーティングプロトコルを用いて単一の仮想的なグループを構成する. そして, 各グループはそのグループ鍵 (K_{SGRP}) はもつが, 単一の仮想的なグループの鍵 (K_{GRP}) はもたない. したがって, メンバの join や leave は, 各サブグループで管理すればよく, ローカルな K_{SGRP} のみを変更すればよい. これにより, 規模拡張性は達成される.

Iolus では, 一番上位のサブグループの管理を行なうグループセキュリティコントローラ (group security controller, GSC) と, 他のサブグループを管理するための, サブグループに 1 つずつあるグループセキュリティ媒介者 (group security intermediate, GSI) と呼ばれるエージェントを設ける. 一般的にこれらはグループセキュリティエージェント (group security agent, GSA) と呼ばれる.

Iolus では, 送受信者における変更はほとんど無く, マルチキャストグループのメンバは, 普段通りの方法で送受信を行なう. 唯一異なる点は, 単一のマルチキャストグループに join するのではなく, 別々のローカルなマルチキャストグループに join するということである. そして GSA が各サブグループを結合し, 仮想的なグループ中の全てのサブグループにローカルなマルチキャストのデータを協調して配送する.

つまり, GSA はサブグループの階層構造を作り上げる. GSC は, 安全な配送木の根において, 最上位のサブグループの制御を行ない, グループ全体のセキュリティの責任を負う. GSI は, GSC か親の GSI の代理として行動する権限をもった, 特別に信頼されたサーバであり, ローカルなサブグループを制御する. GSI は, 安全な配送木の範囲内で, レベルに応じてグループ化され, 1 つ上位のサブグループか GSC のサブグループに join する. GSI はサブグループ間の橋の役目をし, 上位のサブグループのマルチキャストデータを受信したら下位に, 下位のサブグループのマルチキャストデータを受信したら上位にマルチキャストし直す. ここで隣接するサブグループの K_{SGRP} は異なるが, GSI が 2 つのグループの K_{SGRP} をもつことにより通信を行なうことができる.

2.6 鍵交換のための機構についての検討

本節では, 前節までに概説した 4 つの機構の特徴を挙げ, それらの比較および検討を行なう.

2.2 節で述べた ISAKMP/Oakley は IPsec に必要な SA を動的に確立するための鍵交換プロトコルである. しかし, ISAKMP/Oakley はユニキャスト通信しか対応していない.

2.3 節で述べた GKMP はマルチキャストグループのための鍵配布の機構を規定している. グループ毎に鍵を配布するグループコントローラを設けて鍵を配布する. 鍵の配布にはいかなる通信プロトコルも用いることができる. しかし, この機構は一極集中となってしまうため, 規模拡張性に欠ける. また, グループへの鍵配布は可能であるが, IPsec の SA を確立することはできない. IPsec の SA を確立するためには大幅な拡張が必要となる.

2.4 節で述べた SMKD も GKMP と同様にマルチキャストグループのための鍵配布機構を規定している．実際のマルチキャスト通信にはいかなるマルチキャストプロトコルを用いることができるが，鍵配布にはマルチキャストルーティングプロトコルである CBT を用いなければならないため，CBT がほとんど普及していない現状では大きな問題である．CBT ではグループ毎にコアルータがグループのメンバ管理を行なっており，SMKD では鍵の配布もコアルータが行なう．SMKD では鍵の配布とメンバ管理を他のルータに委任することができるため，規模拡張性に富む．しかし，GKMP と同様に IPsec の SA を確立することはできないため，大幅な拡張が必要である．

2.5 節で述べた Iolus は規模拡張性のある安全なマルチキャスト通信の枠組だけを規定しているにすぎない．

これらを比較したものを表 2.1 に示す．

表 2.1: 鍵交換のための機構の比較

	ISAKMP/Oakley	GKMP	SMKD	Iolus
規模拡張性	○	×	○	○
マルチキャストグループの鍵配布	×	○	○	○
鍵配布に必要なプロトコル	特に無し	特に無し	CBT	–
IPsec の SA の確立	可能	不可能	不可能	–
普及度	○	×	×	–

表 2.1 を見てもわかるように，全ての項目を完全に満足しているものは存在していない．そこで本研究では，表 2.1 中の 5 項目のうち，IPsec の SA の確立と普及度という点に注目する．理由を以下に示す．

- IPsec は IP 層でセキュリティを提供する機構なので，アプリケーションに透過的にセキュリティを提供できるから
- IPsec は IPv6 では必須の機構であり，IPv6 への移行が完了すると世界中で利用可能となる
- 広く普及している機構を用いることにより，より多くの人がサービスを受けることができるから

よって，この IPsec の SA の確立と普及という点を同時に満足する ISAKMP/Oakley をセキュリティを考慮したマルチキャスト通信のための鍵配布機構として利用する．しかし ISAKMP/Oakley は，マルチキャスト通信のための鍵配布機構に対応していないため，

本研究では ISAKMP/Oakley のマルチキャスト通信のための拡張を行なう. 詳しい設計方法については次章で述べる.

第 3 章

設計および実装

本章では, マルチキャストグループへの鍵配布が可能となるように ISAKMP/Oakley を拡張する手法を述べ, さらに実装方法について説明する.

3.1 ISAKMP/Oakley のマルチキャストのための拡張

本章では, IPsec を用いたマルチキャスト通信を行なうため, マルチキャストグループへの鍵配布が可能となるように, 鍵交換プロトコル ISAKMP/Oakley を拡張する方法について述べる. つまり SA を確立するために, 鍵交換プロトコルである ISAKMP/Oakley の拡張を行なう.

3.1.1 設計方針

本研究では以下のことを提案する.

- グループ内で用いる鍵の無矛盾性の維持
  グループで共有する鍵の無矛盾性を保つため, 鍵を配布するサーバをグループごとに決め, サーバがそのマルチキャストグループで用いる鍵の管理を行なう. なお, この方法では一極集中の機構となってしまうため, 規模拡張性の面で問題が生じてしまう. サーバを階層化し, 階層的なメンバ管理を行なうことで規模拡張性の問題は解決されるが, 本研究ではサーバの階層化は行なわない. この点は今後本研究で解決すべき今後の問題である.

- マルチキャストグループの動的な変更
  マルチキャストグループへあらたにメンバが加わる (以後 join するという) 際にグループの鍵を確実に獲得するために, グループに対して鍵の要求は join と同時に行なう. また, メンバがマルチキャストグループから抜ける (以後 leave するという) 際には単にグループの鍵を廃棄する. 鍵の廃棄は, 鍵の生存時間を設け join しているメンバに対して鍵を一定周期で再配布することで実現する.

- アクセス制御機構
  アクセス制御機構は, 鍵配布を行なうサーバの機能とする. ただし, グループの正当なメンバを登録したアクセス制御リスト (Access Control List, ACL) はあらかじめ設定してあるものとする. また, IP 層でのアクセス制御を考えているため, ユーザベースでなくホストベースのアクセス制御を行なう. そのため, ユーザベースのアクセス制御は各アプリケーション毎で行なうべきである. なお, 本研究では ACL の作成, 変更は手動で行なうことにする.

3.1.2 提案モデル

本小節では, 設計方針を基にした ISAKMP/Oakley の拡張方法について述べる.
本研究で提案するモデルは以下の 2 種類の鍵交換デーモンにより構成される.

- 鍵サーバ
  各マルチキャストグループに 1 つ存在し, グループで用いる鍵の生成と配布, およびメンバのアクセス制御を行なう鍵交換デーモン. 起動時にマルチキャストグループの鍵, セキュリティ・パラメータ・インデックス (SPI), DES–CBC [KMS95] 等の暗号化に用いる初期ベクタ (Initialization Vector, IV) を生成し, マルチキャストのメンバへ配布する.

- クライアント
  グループのメンバで, 鍵サーバ以外の鍵交換デーモン.

なお, 鍵サーバ, クライアント共にユニキャスト通信のための SA の確立は可能である.
鍵サーバとクライアントの間では以下の処理を行なう

- Phase 1
  クライアントがマルチキャストアドレスに join メッセージを送信する. 鍵サーバが join メッセージを受け取り, 両者の間で ISAKMP SA を確立する. join メッセージは本研究で新しく提案するものであるが, ISAKMP SA の確立は従来の ISAKMP/Oakley Phase 1 の処理に全く手を加えずに用いる.

- Phase 2
  鍵サーバがグループの鍵情報をクライアントに配布する. クライアントは鍵の完全性の確認メッセージを返送する. Phase 2 での処理は従来の ISAKMP/Oakley Phase 2 と全く異なるものであり, 本研究であらたに提案する部分である.

3.1.3 小節, 3.1.4 小節では, 各フェーズについて詳しく説明する.

3.1.3 Phase 1: 鍵の要求と ISAKMP/Oakley のための SA の確立

本小節では Phase 1 における, クライアントと鍵サーバでの処理を示す (図 3.1). この場合, クライアントが ISAKMP/Oakley における Initiator となり, 鍵サーバが Responder となる.

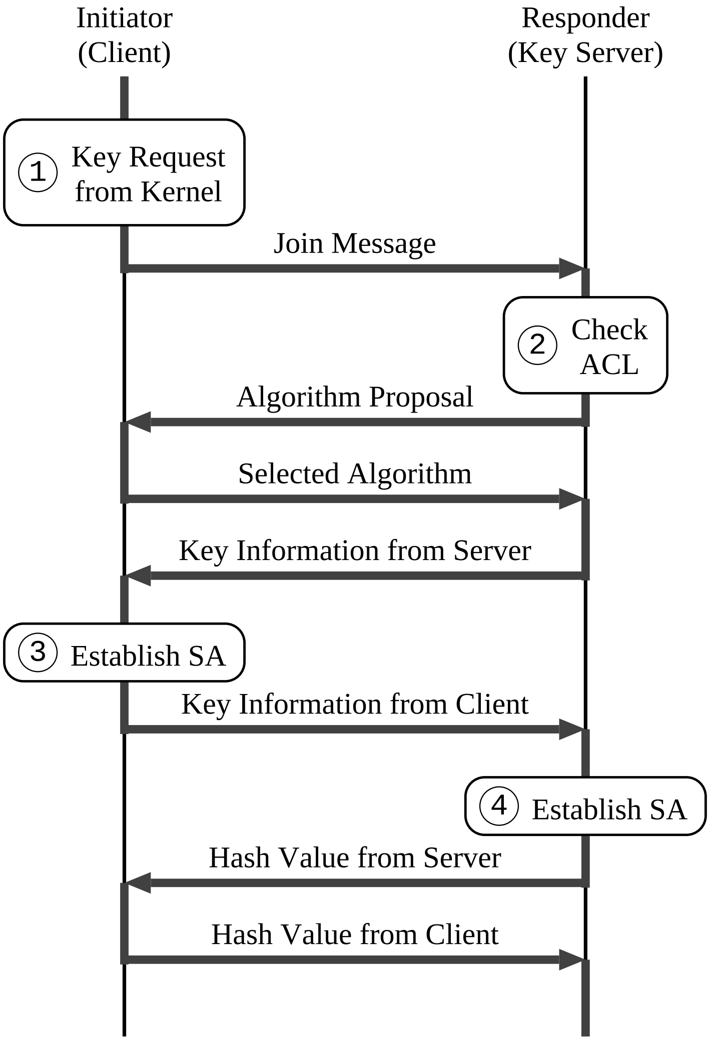

図 3.1: Phase 1 での処理

カーネルからの鍵の要求

クライアントに対して, マルチキャストグループの鍵の要求は以下のようにして発生する (図 3.1 ①, 図 3.2).

1. ホスト上で動作しているプロセスが IP マルチキャストグループに join するか, マルチキャストアドレス宛に IP データグラムを送信しようとする.
2. カーネルがそのグループに対応する SA が無いと判断する.
3. PF_KEY [MMP97] に規定されているとおり, カーネルが鍵交換デーモンに対して, そのマルチキャストアドレスとの SA を確立するように要求する.
4. カーネルから要求された鍵の種類 (AH か ESP か) とマルチキャストグループを含む join メッセージを作成し, マルチキャストアドレス宛に送信する.

join メッセージをマルチキャストアドレス宛に送信し, 鍵サーバのみがこれに返答するというモデルを採用することで, クライアントは鍵サーバの IP アドレスを知る必要はない.

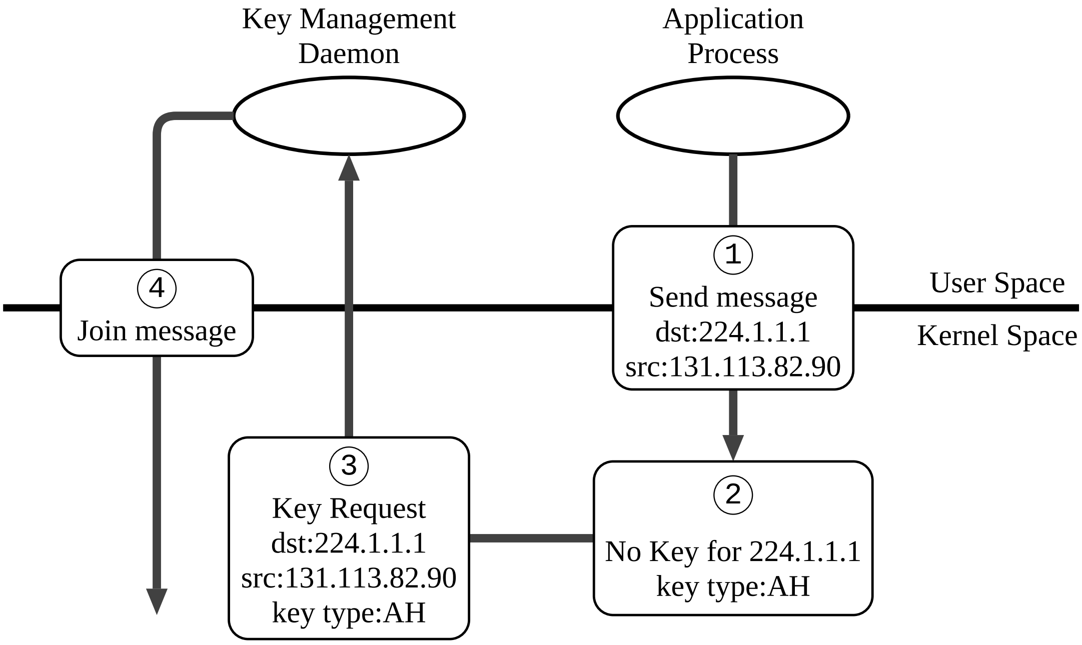

図 3.2: カーネルからの鍵の要求

アクセス制御

鍵サーバは join メッセージを受信すると, join メッセージを送信したクライアントの IP アドレスを調べ, あらかじめ設定してあるアクセス制御リストと比較する (図 3.1 ②). アクセス制御リストに登録されているホストからの要求である場合は, ISAKMP SA を確立する処理に移る. そうでない場合は, 不当なホストからの要求であると判断し, 単に要求を無視する (図 3.3).

なお, 鍵サーバとクライアントとの ISAKMP SA が既に確立されている場合は, Phase 2 の処理へと移る.

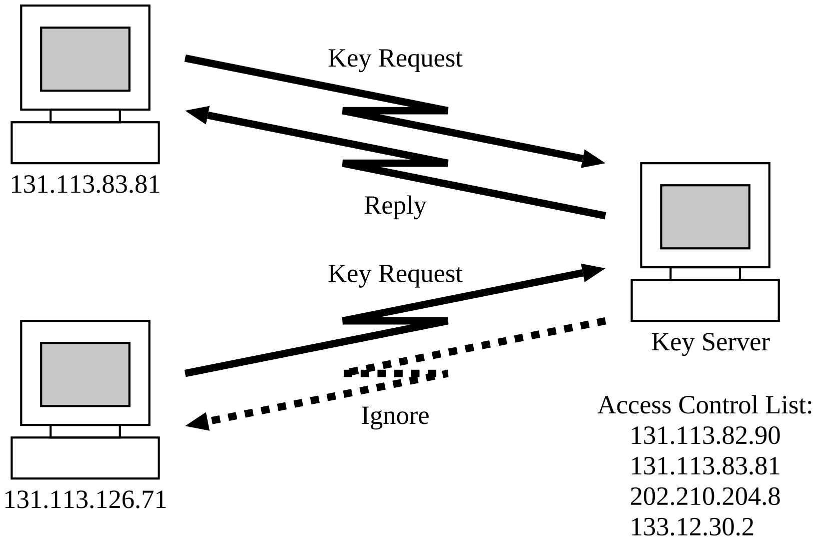

図 3.3: アクセス制御

ISAKMP/Oakley のための SA の確立

鍵サーバが join メッセージを受信した後の Phase 1 での処理は, 2.2 小節で述べた ISAKMP/Oakley の Phase 1 での処理と全く同じものである. 2.2 小節では Main Mode と Aggressive Mode の 2 つがあると述べたが, 本実装においては Main Mode のみサポートする.

鍵サーバとクライアントは ISAKMP SA を確立するために, 以下の処理を行なう (図 3.1).

1. 鍵サーバは ISAKMP SA を確立するために利用可能なアルゴリズムを希望する順に並べ, クライアントに送信する.

2. クライアントはその中から利用可能で, かつ鍵サーバが希望するもっとも優先順位の高いアルゴリズムを選択し, 鍵サーバに送信する.

3. 鍵サーバは, クライアントが ISAKMP SA を確立するために必要な情報を送信する.

4. クライアントは受信した情報を基に ISAKMP SA を生成する (図 3.1 ③). そして, 鍵サーバが SA を確立するために必要な情報を送信する.

5. 鍵サーバは受信した情報をもとに ISAKMP SA を生成する (図 3.1 ④). この時点で鍵サーバとクライアント間で ISAKMP SA を確立できたこととなる. これ以降, 暗号化, 復号化にはこの ISAKMP SA を用いる.

6. 鍵サーバとクライアントで共有している鍵が正しいことを確認するために, 鍵サーバは両者が共有している情報と鍵を基にハッシュ値を計算し, それを暗号化してクライアントに送信する.

7. クライアントは復号化を行ない, 両者が共有している情報と鍵をもとにハッシュ値を照合し, SA が正しく確立されたことを確認する. そして, またハッシュ値を計算し, それを暗号化して鍵サーバに送信する.

8. 鍵サーバは復号化を行ない, 同様にハッシュ値を照合し, SA が正しく確立されたことを確認する. その後, Phase 2 の処理に移る.

3.1.4 Phase 2: IPsec を用いるマルチキャストのための SA の確立

本小節では Phase 2 における, クライアント (Initiator) と鍵サーバ (Responder) での処理を示す (図 3.4).

1. 鍵サーバはグループの鍵, SPI, 初期ベクタを含む鍵情報メッセージを, Phase 1 において確立された ISAKMP SA を用いて暗号化し, クライアントに送信する.

2. クライアントは復号化を行ない, その中から鍵情報を取り出しカーネルに渡す. これによりクライアントでは, IPsec を用いるマルチキャスト通信のための SA をカーネルが所持したことになる.

3. IPsec を用いるマルチキャスト通信のための SA が正しく確立されたことを確認するために, クライアントは確立された SA からハッシュ値を計算し, ISAKMP SA を用いて暗号化し, 鍵サーバに送信する.

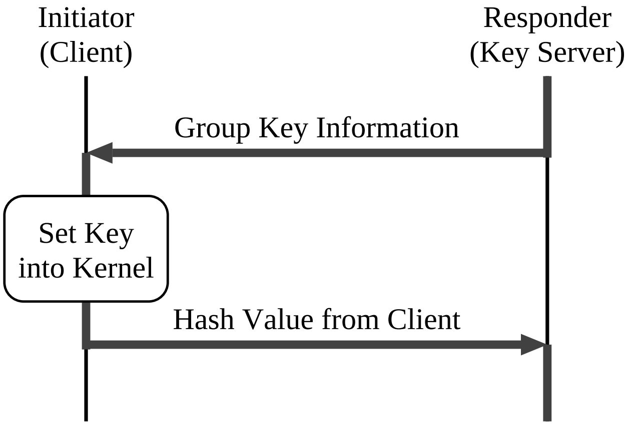

図 3.4: Phase 2 での処理

4. 鍵サーバは復号化を行ない, マルチキャストのための SA からハッシュ値を照合し, SA が正しく確立されたことを確認する. もし SA が正しく確立されていなければ, クライアントにエラーメッセージを送信し, 確立した SA を無効とする.

カーネルへの鍵のセットアップ

カーネルへの鍵のセットアップは図 3.5 のように行なわれる.

1. 鍵サーバから受け取った鍵情報からグループの鍵, SPI, 初期ベクタ, アルゴリズムを取り出し, 鍵の種類 (AH か ESP か), グループのアドレスと, 通信しようとするインタフェースの IP アドレスと共に PF_KEY [MMP97] に規定されているとおりにカーネルに渡す.
2. カーネルはそれらの情報を用いて, マルチキャストグループの SA をセットアップする.
3. 以降, マルチキャスト通信では, 確立された SA を用いて通信する.

以上で IPsec を用いたマルチキャスト通信が行なえるようになる. ここでは例として, 確立された SA を用いた IP マルチキャスト通信における処理の様子を 3.6 に示す. ここでは IPsec の認証ヘッダのみについて例を示すが, 暗号化ペイロードを用いる場合も同様の処理を行なう.

1. アプリケーションプロセスはカーネルを通じて IP マルチキャスト通信を行なう. この過程は, IPsec を利用しない通信でも同様に行われる.

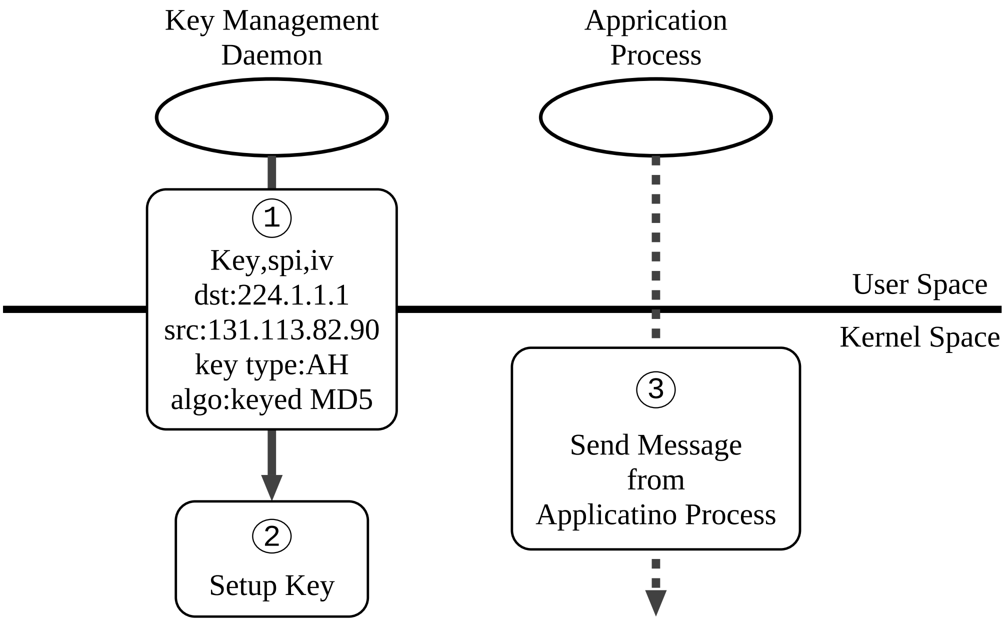

図 3.5: カーネルへの鍵のセットアップ

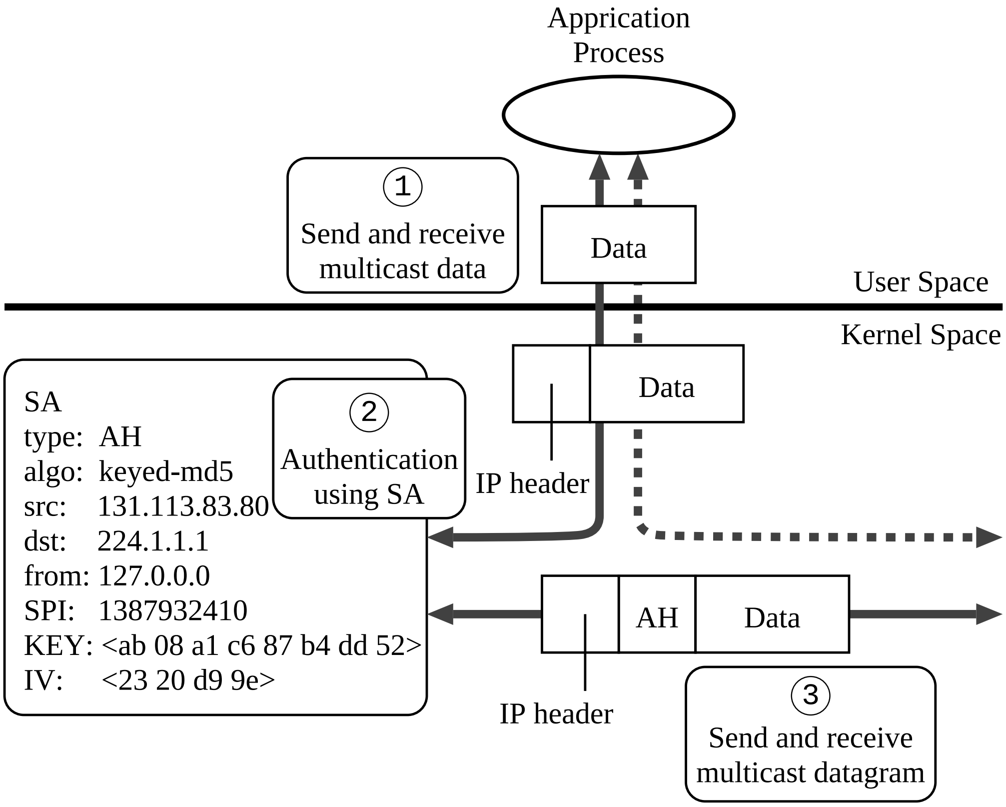

図 3.6: 確立された SA を用いたマルチキャスト通信

2. カーネルは, IPsec が設定されており, さらに IP マルチキャスト通信のための SA が確立されている場合のみ, SA を用いて IP データグラムの認証を行なう. 認証が失敗した場合はカーネルで IP データグラムが暗に廃棄される.
3. カーネルは IP マルチキャストアドレス宛に IP データグラムを送信し, また IP マルチキャストアドレス宛に送信された IP データグラムを受信する.

3.2 実装

本節では, 本研究で行なった実装方法について述べる. 実装は FreeBSD 2.2.1–RELEASE, WIDE プロジェクトで開発された IPsec カーネル上で行な った.

なお, 本実装では, 鍵交換デーモン同士の通信には全て UDP 500 番ポートを用いる.

3.2.1 データ構造

本小節では本実装において新しく定義したメ ッセージのデータ構造を示す.

join メ ッセージ

クライアントは Phase 1 において join メ ッセージをマルチキャストアドレス宛に送信する. join メ ッセージは ISAKMP ヘ ッダと join ペイロードからなる. join ペイロードは図 3.7 に示す join ペイロードヘ ッダのみからなる.

```
struct isakmp_join
{
    u_int8_t         isaj_np;        /* Next Payload             */
    u_int8_t         isaj_reserved;  /* Reserved (Must be zero) */
    u_int16_t        isaj_length;    /* Payload Length           */
    struct sockaddr  isaj_group;     /* Multicast Group Address */
    u_int8_t         isaj_keytype;   /* Type of Key              */
    u_int8_t         isaj_zero[3];   /* Must be zero             */
};
```

図 3.7: join ペイロードヘ ッダ

マルチキャストのための鍵情報メッセージ

鍵サーバは Phase 2 においてマルチキャストのための鍵情報メッセージをクライアントに送信する. 鍵情報メッセージは ISAKMP ヘッダとハッシュペイロード, セキュリティ・アソシエーションペイロード, そして鍵情報ペイロードからなる. 鍵情報ペイロードは図 3.8 に示す鍵情報ペイロードヘッダとグループの SPI, 鍵, IV とからなる.

```
struct isakmp_multicast_key
{
    u_int8_t         isamk_np;         /* Next Payload              */
    u_int8_t         isamk_reserved;   /* Reserved (Must be zero) */
    u_int16_t        isamk_length;     /* Payload Length            */
    struct sockaddr  isamk_group;      /* Multicast Group Address */
    u_int16_t        isamk_spilen;     /* Length of SPI             */
    u_int16_t        isamk_keylen;     /* Length of Key             */
    u_int16_t        isamk_ivlen;      /* Length of IV              */
    u_int16_t        isamk_zero;       /* Must be zero              */
};
```

図 3.8: マルチキャストのための鍵情報ペイロードヘッダ

3.2.2 Phase 1 の実装

Phase 1 における処理を図 3.9 に示す. Phase 1 において, 本研究であらたに定義した部分は, join メッセージの送信と鍵サーバでのアクセス制御機構である. なお, 図 3.9 では 2.2.1 小節で述べた表記法を用いている.

join メッセージの送信

カーネルからマルチキャストグループの鍵の確立要求がきたら, クライアントは join メッセージを作成し, そのマルチキャストアドレス宛に送信する. join メッセージは ISAKMP ヘッダと join ペイロードとからなる. なお, 図 3.9 では join ペイロードは JOIN と表記してある.

join メッセージはマルチキャストグループに join している全てのメンバが受信するが, 鍵サーバでないメンバはそのメッセージを単に無視する.

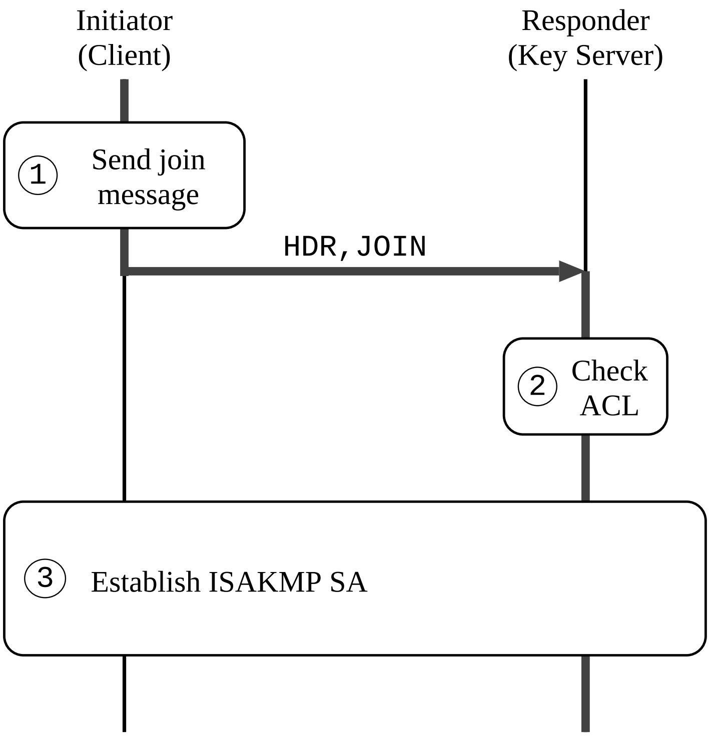

図 3.9: Phase 1 の実装

アクセス制御

join メッセージを受け取った鍵サーバは, 鍵の要求をしているホストが正当なメンバであるかを確認してから鍵の配布を行なわなければならない. したがって, 鍵サーバにおいてアクセス制御の機構を実装しなければならない.

そこで, 本研究では正当なメンバのリストであるアクセス制御リストを, 鍵サーバがもつというアプローチをとることにする. アクセス制御リストは管理者が `/etc/pluto.conf` というファイルに手動で設定するものとする. アクセス制御リストには図 3.10 のように, 正当なメンバの IP アドレスを列挙する.

join メッセージを受け取った鍵サーバの役割を果たす鍵交換デーモンは, メッセージの送信者の IP アドレスとアクセス制御リストの IP アドレスとを比較し, 一致すれば次の Phase 1 の処理へと移る. もし, アクセス制御リスト全てと比較しても一致しなければ, 要求を出したホストは正当なメンバでないということなので, そのメッセージを無視する.

```
131.113.82.90
131.113.83.81
202.210.204.8
133.12.30.2
```

図 3.10: アクセス制御リスト `/etc/pluto.conf` ファイルの例

3.2.3 Phase 2 の実装

Phase 2 における処理を図 3.11 に示す．Phase 2 は，本研究においてあらたに定義した部分である．なお，図 3.11 では 2.2.1 小節で述べた表記法を用いている．

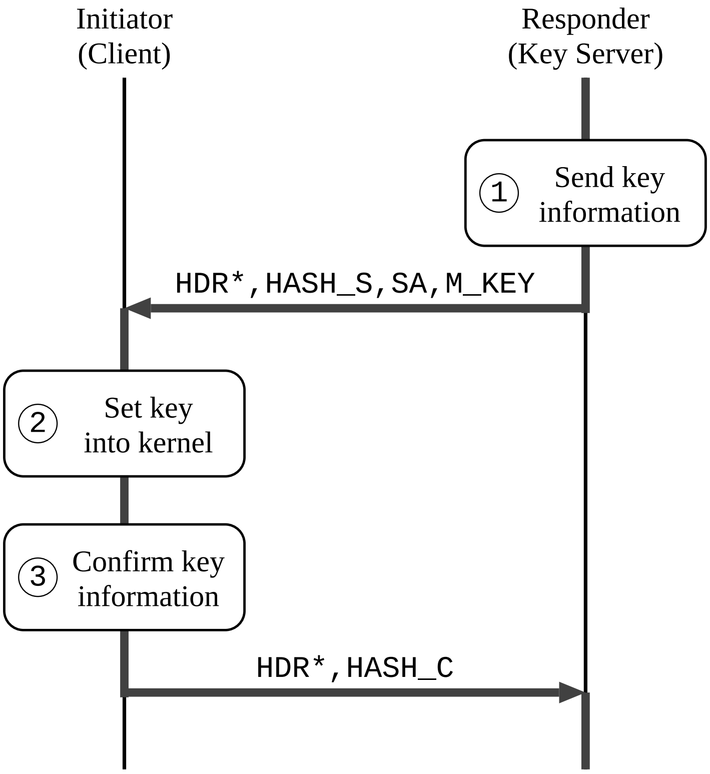

図 3.11: Phase 2 の実装

鍵情報の配布

Phase 1 の処理が完了し ISAKMP/Oakley のための SA が確立されたら，鍵サーバはクライアントに鍵情報メッセージを送信する．鍵情報メッセージは ISAKMP ヘッダ，ハッシュペイロード，SA ペイロード，鍵情報ペイロードからなる．なお，図 3.11 では鍵情報

ペイロードは M_KEY と表記してある.

鍵情報ペイロードは Phase 1 で確立した ISAKMP SA を用いて ISAKMP ヘッダ以降を暗号化してあるので, 鍵情報を保護することができる. またハッシュペイロードにより, メッセージの完全性と ISAKMP SA が期限内であることが保証される. HASH(S) は式 (3.1) により求められる値である.

$$HASH(S) = prf(\ SKEYID_a,\ SA \mid M\_KEY\ ) \tag{3.1}$$

カーネルへの鍵のセットアップ

クライアントは鍵情報メッセージから以下のものを取り出し, [MMP97] に規定されているとおりにカーネルに渡すことにより, マルチキャストグループの SA を確立する.

- グループの鍵, SPI, 初期ベクタ
- グループのアドレス, 通信するインタフェースのアドレス
- 鍵の種類と使用するアルゴリズム

鍵情報の確認

クライアントは, 鍵サーバから配布された鍵が正確にセットアップされたことを確認するために鍵情報の確認メッセージを返送する. 鍵情報の確認メッセージは ISAKMP ヘッダとハッシュペイロードからなる.

鍵情報の確認ペイロードは Phase 1 で確立した ISAKMP SA を用いて ISAKMP ヘッダ以降を暗号化してあり, 認証情報を保護することができる. 鍵情報の確認はハッシュペイロードの値により鍵サーバにおいて行なわれる. このハッシュ値は式 (3.2) により計算される.

$$HASH(C) = prf(\ SKEYID_a,\ SPI \mid KEY \mid IV\ ) \tag{3.2}$$

第 4 章

評価および議論

本研究では，マルチキャストグループへ鍵配布が可能となるように, 鍵交換プロトコル ISAKMP/Oakley の拡張を行なった.

そこで本章では，従来の機構でも可能であったユニキャスト通信用の鍵を交換するのに必要な時間と，本研究で新たに提案したマルチキャスト通信用の鍵を交換するのに必要な時間をそれぞれ測定し，測定結果について比較および検討を行なう.

4.1 評価項目

Phase 1 では Signature, Public Key Encryption, Pre–shared Key の 3 種類の認証方式それぞれについて測定を行なった (以後，認証方式が Signature によるものは *sig*, Public Key Encryption によるものは *pub*, Pre–shared Key によるものは *pre* と記す).

評価を行なう項目の詳細は以下のとおりである．

1. 各メッセージの処理に有する時間
2. 各 Phase に有する時間
3. 通信の開始から 通信相手と SA を確立するまでに有する時間

上記の項目を図で示すと, ユニキャスト通信用の鍵を交換する場合は図 4.2, マルチキャスト通信用の鍵を交換する場合は図 4.3となり，各図中の番号を以後評価結果を示す際に用いる．

4.2 評価環境

本研究で用いた評価環境を図 4.1 に示す.

なお，2 ホスト上で動いている OS および鍵交換デーモンは以下の通りである．

- OS
  FreeBSD2.2.1-RELEASE に，伊藤純一郎氏が開発した IPsec カーネルへの変換パッチ[*]を当てたもの
- 鍵交換デーモン
  祢宜知孝氏が開発した鍵交換デーモン[†] をベースに，本研究で提案した拡張を加えたもの

---

[*] ftp://ftp.itojun.org/pub/ipsec/FreeBSD221-ipsec.970714.tar.gz.pgp

[†] http://www.mt.cs.keio.ac.jp/person/negi/implement-j.html

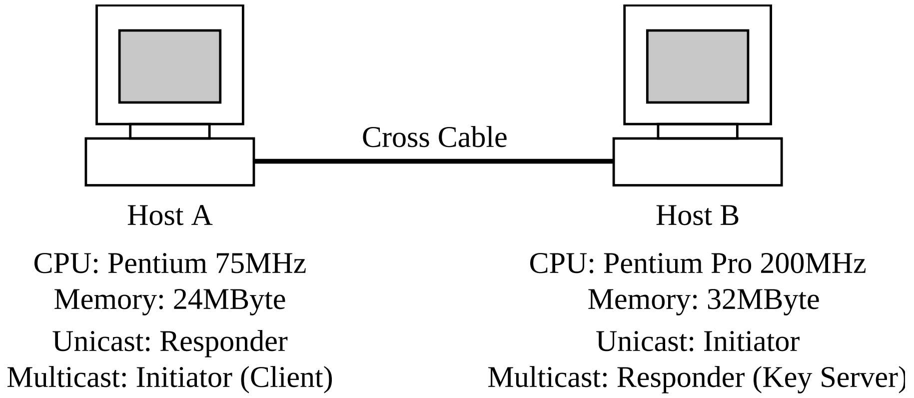

図 4.1: 評価環境

2 台をクロスケーブルで接続し, ユニキャスト通信の場合はホスト A を Responder, ホスト B を Initiator として測定を行ない, マルチキャスト通信の場合はホスト A を Initiator, ホスト B を Responder として測定を行なった . ISAKMP SA の確立するための第一メッセージ (SA を確立するために利用可能なアルゴリズムを通信相手に送信するメッセージ) を送信するのは , ユニキャスト通信用の鍵を交換する際では Initiator であり , マルチキャスト通信用の鍵を交換する際では Responder(すなわち鍵サーバ) であるため , このように 2 つの測定において , Initiator と Responder を入れ替えて測定を行った .

4.3 評価結果および検討

本節では各項目における評価結果を示し , 検討を行なう.

4.3.1 各メッセージの処理にに有する時間

ユニキャスト通信用およびマルチキャスト通信用の鍵を交換する際の各メッセージに有する処理時間を表 4.1 に示す. なお, ユニキャスト通信の場合とマルチキャスト通信の場合では Initiator と Responder の関係ホストが入れ替わっているため, 表 4.1 では比較検討が容易になるよう , 対応する処理を並べて表示している .

本研究で拡張した部分は, Phase 1 開始直後の join メッセージ, Phase 2 の鍵情報メッセージおよび鍵情報確認メッセージの 3 箇所である. したがってその 3 箇所以外の, 図 4.2 における Ru–1–1 から Ru–1–3 までの 5 つの処理と, 図 4.3 における Im–1–2 から Im–1–4 までの 5 つの処理はそれぞれ対応し, 同じ処理を行なうことが予想される . 表 4.1 に示したとおり, Iu–1–2 から Ru–1–3 までの処理と Rm–1–2 から Im–1–4 までの処理に

表 4.1: 各メッセージの処理に有する時間

	unicast ($msec$)				multicast ($msec$)		
	sig	*pub*	*pre*		*sig*	*pub*	*pre*
–	–	–	–	Im–1–1	7.0	7.0	6.8
Iu–1–1	4.4	4.4	4.3	Rm–1–1	3.9	3.7	3.8
Ru–1–1	6.9	6.8	8.1	Im–1–2	0.8	0.8	2.1
Iu–1–2	4.5	12.0	5.1	Rm–1–2	4.3	11.9	4.3
Ru–1–2	142.0	732.1	144.1	Im–1–3	138.8	764.4	140.6
Iu–1–3	84.7	152.5	31.6	Rm–1–3	78.9	152.3	31.2
Ru–1–3	229.3	6.1	6.2	Im–1–4	222.8	6.1	6.2
Iu–1–4	9.9	6.0	5.9	Rm–1–4	6.8	2.9	2.9
Ru–1–4	11.1	11.5	10.8	Im–1–5	1.6	1.6	1.6
Iu–1–5	0.7	0.8	0.7	Rm–1–5	0.1	0.1	0.1
Ru–1–5	1.7	1.7	1.6	–	–	–	–

おいて，それぞれ対応する処理時間はほぼ同じであることが確認できる．しかし, Ru–1–1 と Im–1–2 の処理時間に関しては大きく異なっており, 本研究の設計から予測できない評価結果が生じている．この部分については再度測定を行い，改めて検討することとする．

また, Rm–1–1 の処理では, Iu–1–1 の処理に加えて受信した join メッセージの処理とアクセス制御を行なうため, Rm–1–1 の処理時間は Iu–1–1 の処理時間よりも大きい値を取ることが予想される．しかし, 表 4.1 では認証方式に関わらず， Iu–1–1 の処理時間の方が Rm–1–1 の処理時間よりも大きくなっている．この部分についても，先程の場合と同様に再度測定を行い，改めて検討することとする．

ここで, 本研究において拡張した部分の処理時間について考察する. Im–1–1 の処理ではクライアントが join メッセージを作成し, マルチキャストアドレス宛に送信する. この処理時間は 7 $msec$ である．また, 鍵サーバが鍵情報メッセージを送信する Rm–1–4 の処理時間は，電子署名による認証方式の場合には 6.8 $msec$, 他の 2 つの認証方式では 2.9 $msec$ である．Rm–1–4 において認証方式による処理時間の差が生じるのは, 電子署名方式では, 相手の電子署名を確認する必要があるためと考えられる．

さらに，Rm–1–4 の処理時間は対応する Iu–1–4 の処理時間と比較して, 3 $msec$ 短縮されている．この理由は Iu–1–4 の処理は, Rm–1–4 の処理では必要としない乱数や ID を獲得する必要があるためと考えられる．

またクライアントが鍵情報から，マルチキャストのための SA をカーネルにセットアップし, 鍵情報確認メッセージを返送する Im–1–5 の処理時間, およびカーネルが鍵情報メッ

セージを受信し, 鍵の完全性を確認するRm–1–5の処理時間は，それぞれ1.6 *msec*, 0.1 *msec*となる．この2つの処理時間はユニキャスト通信の場合のPhase 2における処理時間と比較して大幅に短縮されていることが分かる．

4.3.2 各**Phase**に有する時間

ユニキャスト通信用およびマルチキャスト通信用の鍵を交換する際の各Phaseに有する処理時間を表4.2に示す. なお, ユニキャスト通信の場合とマルチキャスト通信の場合ではInitiatorとResponderの関係ホストが入れ替わっているため, 表4.2では比較検討が容易になるよう，対応する処理を並べて表示している．

表4.2: 各Phaseに有する時間

	unicast (*msec*)				multicast (*msec*)		
	sig	*pub*	*pre*		*sig*	*pub*	*pre*
Iu–2–1	475.2	913.1	196.3	Rm–2–1	432.7	933.4	186.1
Ru–2–1	466.4	911.8	191.6	Im–2–1	457.3	940.9	193.2
Iu–2–2	17.6	22.4	17.2	Rm–2–2	10.1	5.0	5.0
Ru–2–2	16.7	17.7	16.1	Im–2–2	0.1	0.1	0.1

Iu–2–1, Rm–2–1, Ru–2–1 ,Im–2–1の各処理において認証方式毎にそれぞれ比較するとどの認証方式においても，必要な処理時間はほぼ等しいといえる．また，マルチキャスト通信の場合はjoinメッセージの送受信とアクセス制御を行なうため，多少の処理時間を有することは4.3.1小節の結果で示したが，その処理時間はPhase 1全体の処理時間と比較すると，無視できるほど小さいことが分かる．

したがって 本研究で新たに導入したjoinメッセージの送受信やアクセス制御を行うための実行コストはPhase 1全体を考えた場合，非常に小さいと判断できる．

次に, Iu–2–2とRm–2–2の処理, Ru–2–2とIm–2–2の処理をそれぞれ比較すると, マルチキャスト通信の場合のPhase 2に必要な処理時間はユニキャスト通信の場合と比較して非常に小さいことがわかる. ユニキャスト通信の場合はホスト間でIPsecのためのSAを生成しなければならないが, 一方マルチキャスト通信の場合は単にグループの鍵情報を配布するだけですむため，このような結果となると考えられる．

4.3.3 通信の開始から 通信相手と SA を確立するまでに有する時間

ユニキャスト通信用およびマルチキャスト通信用の鍵を交換する際の通信の開始から通信相手とSAを確立するまでに有する処理時間を表4.3に示す．なお，ユニキャスト通信の場合とマルチキャスト通信の場合ではInitiatorとResponderの関係ホストが入れ替わっているため，表4.3では比較検討が容易になるよう，対応する処理を並べて表示している．

表4.3: SAが確立されるまでの時間

	unicast ($msec$)				multicast ($msec$)		
	sig	*pub*	*pre*		*sig*	*pub*	*pre*
Iu–3	491.7	934.4	212.6	Rm–3	442.8	938.5	191.1
Ru–3	489.7	933.0	210.3	Im–3	467.1	942.4	194.8

4.3.2小節の結果から考察するとSAが確立されるまでの処理時間はマルチキャスト通信の場合の方がユニキャスト通信の場合と比較して短縮されることが分かる．しかし，認証方式に公開鍵暗号 (*pub*) を用いた場合のみ，マルチキャスト通信のためのSA確立に有する処理時間が，ユニキャスト通信の場合と比較して大きくなっていることが表4.3から確認される．この理由は，公開鍵暗号の認証処理ではファイルアクセスを必要とし，それにより処理時間の分散が非常に大くなったためであると考えられる．この部分についても，先程の場合と同様に再度測定を行い，改めて検討することとする．

4.3.4 本研究で提案した拡張によるオーバーヘッド

4.3.1, 4.3.2, 4.3.3小節より，以下のことが確認できる．

- マルチキャストグループへの鍵配布を可能とするために，本研究で新たに導入したjoinメッセージの送受信とアクセス制御に有する処理時間は，Phase 1全体の処理時間と比較すると無視できる程度のオーバーヘッドにすぎない．
- マルチキャスト通信の場合のPhase 2に必要な処理時間はユニキャスト通信の場合と比較して非常に小さくなる．

したがって，本研究で提案した拡張によるオーバーヘッドは，非常に小さいことが分かる．

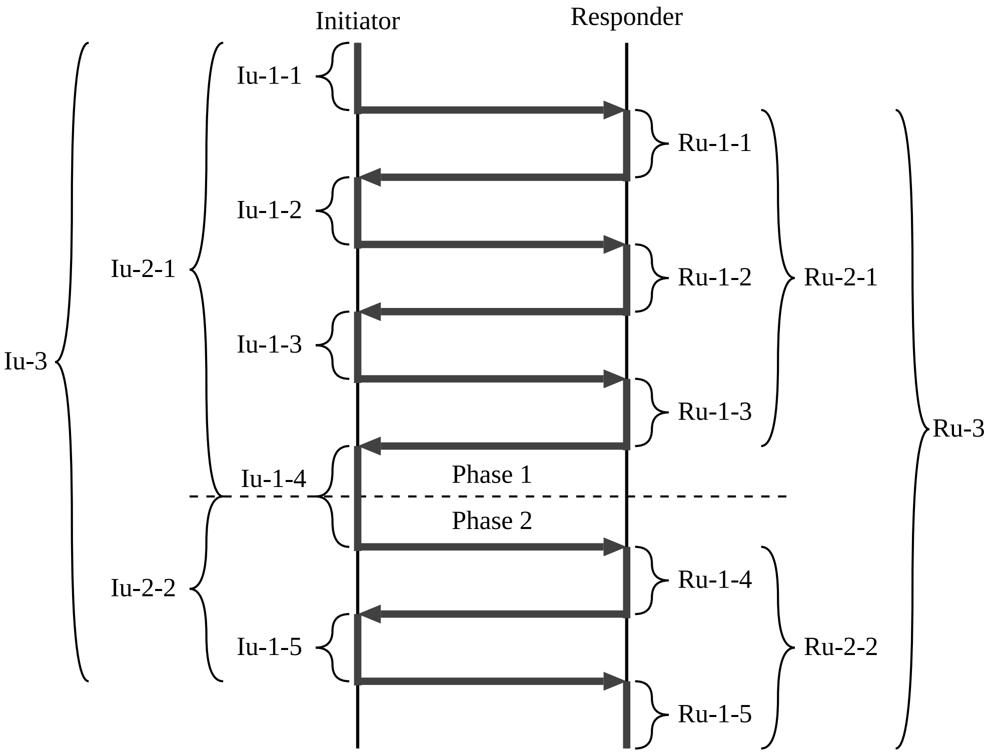

図 4.2: ユニキャストのための鍵交換を行なう場合の評価項目

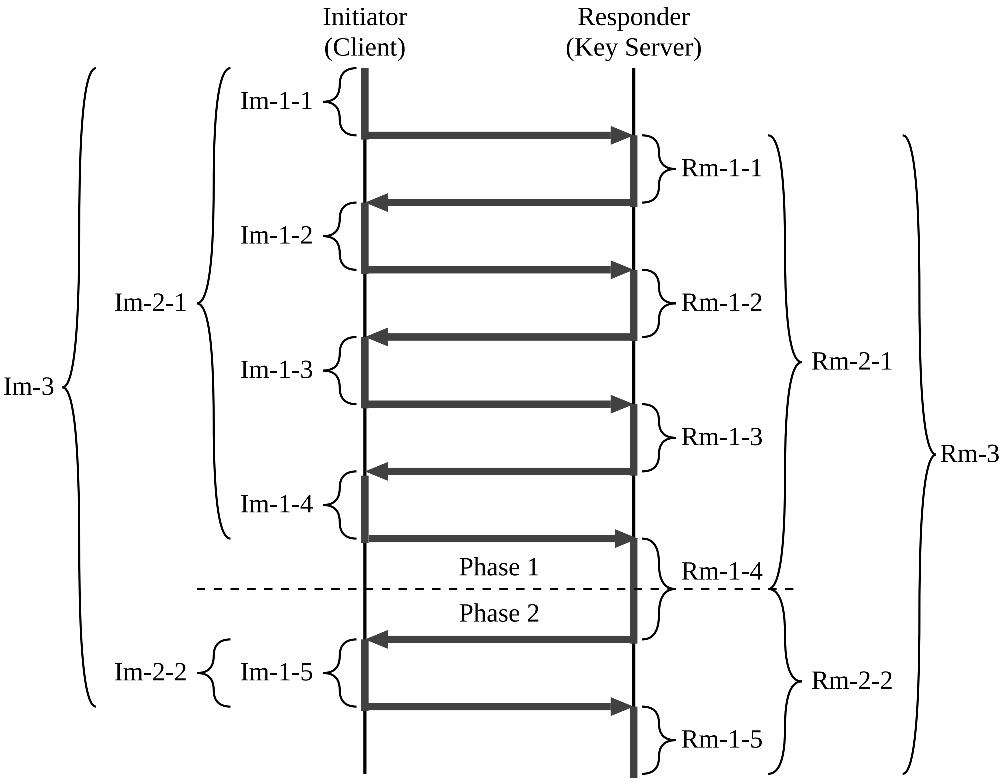

図 4.3: マルチキャストのための鍵交換を行なう場合の評価項目

第 5 章

今後の課題

本章では, 本研究の今後の課題について述べる.

5.1 IPv6 のサポート

20 年前に設計された現在のインターネットプロトコル (IPv4) は, 急激に普及した一方で近年, 以下のような問題が指摘されるようになってきた [Hui96].

- アドレス空間の不足
- 実際の利用状況と使用のひずみ

以上の問題を解決するために, 次世代インターネットプロトコルとして, **IP version 6 (IPv6)** [DH95] が提案された. IPv6 では上記の問題の他に, セキュリティ強化策として Appendix A に付記した IPsec が必須となっている. IPv6 に関する研究は実装段階をほぼ終え, 現在 6 Bone において相互接続実験が行われており, 近い将来 IPv4 から IPv6 へ移行することが十分考えられる.

本研究は IPv4 上で実装しており, IPv6 には対応していない. そのため, 本研究の IPv6 への対応は急務であるといえる. しかし, IPsec や ISAKMP/Oakley の仕様において, IPv4 から IPv6 への移行に関する相違点はほとんどないため, 比較的容易に実現可能であると考えられる.

5.2 規模拡張性

本研究では, マルチキャストグループへの鍵配布機構として各グループ毎に 1 つの鍵サーバを設けた. しかし, 各グループ毎に 1 つの鍵サーバを設ける手法では, グループのメンバ数の増加に伴い鍵サーバの負荷が増大してしまうため, 規模拡張性に欠ける.

この問題の解決策の 1 つとして, 各グループ毎に鍵サーバを複数設け, 各鍵サーバへの負荷を軽減させるという方法が考えられる.

しかし, 複数の鍵サーバを設けることにより, グループで共有する鍵の無矛盾性を維持することが難しくなる可能性があるため, 注意深く設計することが必要である. このようにグループで共有する鍵の無矛盾性を維持しつつ, 鍵サーバへの負荷を軽減することは今後の課題である.

第 6 章

結論

本研究では, 現在標準化が進められている鍵交換プロトコル ISAKMP/Oakley を, マルチキャストグループに対して鍵の配布が可能となるよう拡張を行なった.

本研究で提案した拡張は以下の 4 つの特徴をもつ.

- マルチキャストグループ毎に鍵サーバを設け, メンバに鍵配布を行なう
- クライアントは鍵の要求メッセージをマルチキャストアドレス宛に送信するので, 鍵サーバの存在を意識する必要がない
- クライアントがグループへの join と同時に鍵の要求を行なうことにより, 動的なメンバの変更に対応できる
- アクセス制御機構により正当なメンバのみに鍵を配布する

これらは, ISAKMP/Oakley への比較的少ない変更で実現できた. また, 提案したモデルを実装し, 評価を行った結果, 拡張による処理時間のオーバーヘッドは非常に小さいことが確認できた.

本研究の今後の課題としては大きく分けて以下がある.

- IPv6 への対応
- 規模拡張性のための階層的鍵サーバの導入

今後はこれらの拡張を行なう予定である.

謝辞

まず始めに, 本研究の機会を与えて下さり, 懇切丁寧な御指導を頂きました慶應義塾大学理工学部電気工学科教授 所 真理雄 博士に心から感謝いたします.

本研究の方向付けをして下さり, 深い議論や御査読など貴重な時間を割いて頂きました本研究室博士課程 伊藤 純一郎 氏に深く感謝いたします.

また, 多大なる御示唆や御指導, そして御査読を頂きました本研究室博士課程 福田 健介 氏, 修士課程 祢宜 知孝 氏に深く感謝いたします.

本研究の全般に渡り, 有意義な御指導を頂きました本研究室修士課程 久曽神 宏 氏, 三輪 泰孝 氏, 矢崎 啓之 氏, 北島 輝一 氏, 舌間 一宏 氏に感謝いたします.

苦しい中を互いに励まし合った本研究室学部4年 小川 孝義 氏, 京田 耕司 氏, 小巻 賢二郎 氏, 田谷 文彦 氏, 山本 祐歌 さんに感謝いたします.

最後に, 数多くの友と, 影から声援を送ってくれた家族に感謝いたします.

1998年2月

Appendix A

IP 層におけるセキュリティの提供

A.1 IPsec

IPsec は [Atk95c] により規定されている, ネットワーク層 (IP 層) でのセキュリティの機構であり, 具体的には IPv6 では拡張ヘッダとして, IPv4 では IP オプションヘッダとして定義されている. またヘッダとして, 認証ヘッダ (Authentication Header, AH) [Atk95a] と暗号ペイロード (Encapsulating Security Payload, ESP) [Atk95b] の 2 つが定義されており, 認証ヘッダは IP データグラムに完全性と認証を, 暗号ペイロードは IP データグラムに完全性, 認証, 機密性を提供するために設計されている. それらは共に用いることもでき, あるいは別々に用いることもできる.

認証ヘッダ, 暗号ペイロードは, それらを実装している 2 つ以上のホスト間, 2 つ以上のゲートウェイ間, 一組のホストとゲートウェイの間のセキュリティを提供する. ホストでなくゲートウェイのみにおいて IPsec を実装することの利点は, 図 A.1 のようにあるサブネットワークにおいてゲートウェイのみが IPsec を実装しており, 他のホストは実装していない場合を考えてみると明らかである. つまり, サブネットワーク A のホストがサブネットワーク B のホストと通信したい場合, 各サブネットワークのゲートウェイより内側にあるホストが信頼できるとすれば, ゲートウェイしか IPsec を実装していないにもかかわらず, そのゲートウェイより内側にある全てのホストが安全に通信をすることができるのである.

次に IPsec を実装している各ホストにおいて, 認証ヘッダと暗号ペイロードを持つ IP データグラムをどのように処理するかを見ることにする. 一般的に認証や暗号化を行な

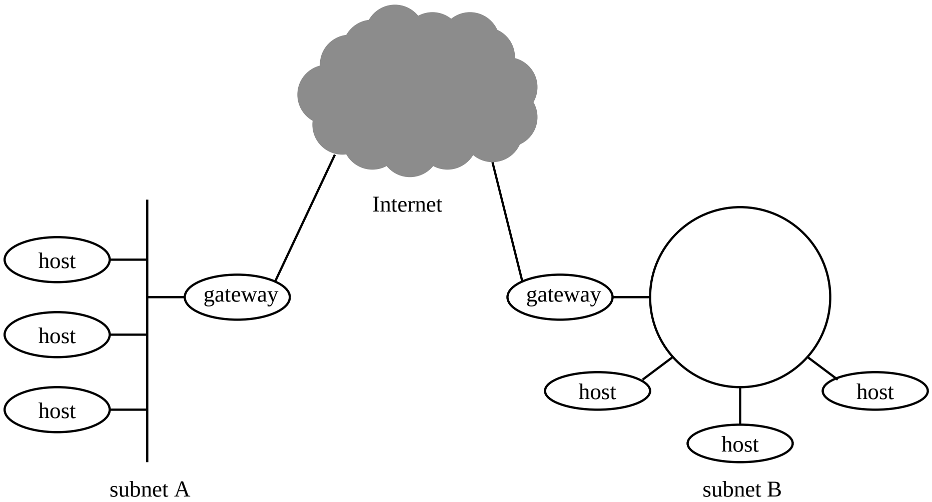

図 A.1: インターネットを介した, IPsec を実装していないホスト間の通信

うためには, 鍵, 認証や暗号化のアルゴリズム, 鍵の長さやアルゴリズムの具体的な利用方法などのパラメータについてあらかじめ送信者と受信者が合意している必要がある. IPsec ではこれらの取り決めを, “セキュリティ・アソシエーション (SA)” と呼んでいる. 各ホストは認証ヘッダや暗号ペイロードによって示されるセキュリティ・パラメータ・インデックス (SPI) と終点アドレスの組み合わせから一意に特定の SA を識別する. 送信ホストは, 始点アドレス, 終点アドレスから該当する SA を選び, その鍵を用いて認証や暗号化を行ない, その SPI 値を認証ヘッダ, 暗号ペイロードに格納する. 一方受信ホストは, 認証ヘッダ, 暗号ペイロードから SPI 値を抽出し, SPI 値と終点アドレスから該当する SA を選び, 認証や復号化を行なう. また SA は通常一方向であるため, 2 つのホスト間で通信するときは, 通常各方向に 1 つづつ, 合計 2 つの SPI 値を用いる. つまり特定の SPI 値と特定の終点アドレスの組み合わせにより一意にセキュリティ・アソシエーションを識別する. なお, 終点アドレスはユニキャストアドレスでもマルチキャストアドレスでも良いことになっている. また, SA の確立は IPsec とは別の枠組で提供されている. これについては 2.2 節で述べた.

A.2 認証ヘッダ

認証ヘッダは [Atk95a] で規定されており, 認証情報をデータグラムに加えることにより完全性と認証を提供することができる. 認証情報は, 配送途中で変化しない IP データグラムのフィールド全てを用いて計算される.

認証ヘッダは, 配送途中で変更される可能性のあるヘッダの後におかれる. 認証ヘッダのフォーマットを図 A.2 に示す. 各フィールドは以下のとおりである.

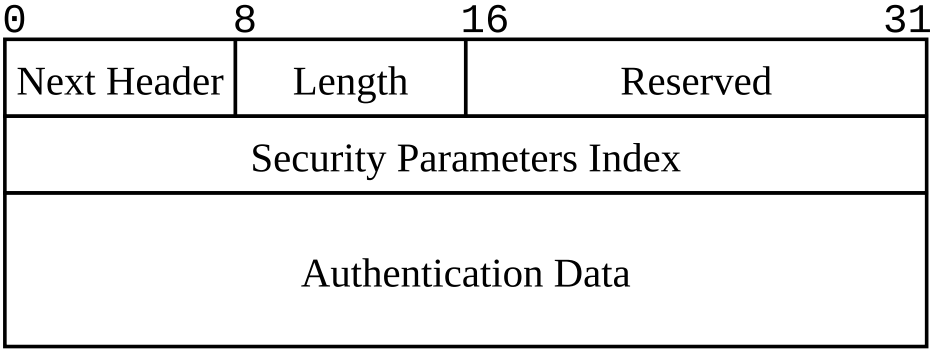

図 A.2: 認証ヘッダのフォーマット

Next Header 認証ペイロードの次のペイロードを指定

Length 32 ビットワード単位での認証データフィールドの長さ

Reserved 未使用, 0 でうめる

SPI SA を指定するための SPI 値

Authentication Data 認証データ, 32 ビットワードの整数倍 (可変長)

通常認証データはメッセージダイジェストアルゴリズムを用いて計算される. このアルゴリズムは通信相手と取り決めることにより, 自由に選択することができるが, 最低でも [MS95] で規定される Keyed MD5 はサポートしなければならないことになっている.

認証データは IP データグラムのヘッダも含めたデータグラム全体から計算されるが, 配送途中で変更されるものは全てのビットが 0 であるものとして計算される. 配送途中で変更されるものには以下のものがある.

- IPv4
  - TTL (Time To Live)
  - ヘッダチェックサム
  - いくつかのオプションフィールド
- IPv6
  - Hop Limit
  - いくつかの拡張ヘッダ

A.3 暗号ペイロード

暗号ペイロードは [Atk95b] で規定されており, IP データグラムのペイロードを暗号化することにより完全性と機密性を提供する.

図 A.3 に暗号ペイロードのフォーマットを示す. 各フィールドは以下のとおりである.

SPI SA を指定するための SPI 値

Transform Data 暗号化されたペイロードの本体 (可変長)

暗号ペイロードのある IP データグラムでは, 暗号ペイロードの SPI フィールドまでが平文であり, それ以降は暗号化アルゴリズムにより暗号化されている. この暗号化アルゴ

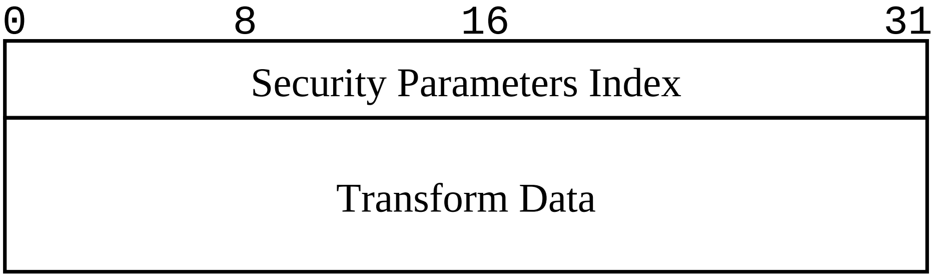

図 A.3: 暗号ペイロードのフォーマット

リズムは, 通信相手と取り決めることにより, 自由に選択することができるが, [KMS95]で規定されるDES–CBCは必ずサポートしなければならないことになっている.

なお, IPデータグラムのなかで暗号ペイロードの置かれる部分により, トンネルモードとトランスポートモードに分けられる.

トンネルモード

トンネルモードの場合, IPデータグラム全体が暗号化される. 送信者はIPデータグラムを作成した後, それを完全に暗号化して暗号ペイロードを生成し, それにあらためてIPヘッダ等を付加して送信する. この場合, 暗号化されていないIPヘッダが始点から終点までのルーティングに用いられる. これにより, 暗号化されたIPデータグラムが安全に終点ホストまでトンネルされる.

トランスポートモード

トランスポートモードでは, UDP/TCP/ICMPなどのトランスポート層のデータのみ, 暗号化を行なう. この場合, IPヘッダやIPオプションが暗号化されていないので, 帯域幅が節約される.

参考文献

[AMP96] Ashar Aziz, Tom Markson, and Hemma Prafullchandra. *Simple Key-Management For Internet Protocols (SKIP) <draft–ietf–ipsec–skip–07.txt>*. Internet Draft, *IPSEC Working Group*, August 1996. work in progress.

[Atk95a] Randall Atkinson. *IP Authentication Header*. RFC 1826, *IETF*, August 1995.

[Atk95b] Randall Atkinson. *IP Encapsulating Security Payload (ESP)*. RFC 1827, *IETF*, August 1995.

[Atk95c] Randall Atkinson. *Security Architechture for IP*. RFC 1825, *IETF*, August 1995.

[Bal96] Tony Ballardie. *Scalable Multicast Key Distribution*. RFC 1949, *IETF*, May 1996.

[BFC93] Tony Ballardie, Paul Francis, and Jon Crowcroft. Core Based Trees (CBT) An Architecture for scalable Inter-Domain Multicast Routing. In *SIGCOMM '93*, ACM, 1993.

[Com95] Douglas E. Commer. *Internetworking With TCP/IP Vol.1: Principles, Protocols, and Architecture, Third Edition*. *Prentice Hall*, 1995.

[DH76] W. Diffie and M. Hellman. New Directions in Cryptograpy. In *Transactions on Information Theory*, IEEE, 1976.

[DH95] Stephen E. Deering and Robert M. Hinden. *Internet Protocol, Version 6 (IPv6) Specification*. RFC 1883, *IETF*, December 1995.

[HC97] Dan Harkins and Dave Carrel. *The resolution of ISAKMP with Oakley <draft–ietf–ipsec–isakmp–oakley–05.txt>*. Internet Draft, *IPSEC Working Group*, November 1997. work in progress.

[HM97a] Hugh Harney and Carl Muckenhirn. *Group Key Management Protocol (GKMP) Architecture*. RFC 2094, *IETF*, July 1997.

[HM97b] Hugh Harney and Carl Muckenhirn. *Group Key Management Protocol (GKMP) Specification.* RFC 2093, *IETF*, July 1997.

[Hui96] Christian Huitema. *IPv6: The New Internet Protocol.* *Prentice Hall*, 1996.

[Kes97] Srinivasan Keshav. *An Engineering Approach to Computer Networking: ATM Networks, the Internet, and the Telephone Network.* *Addison–Wesley*, 1997.

[KMS95] Phil Karn, Perry Metzger, and William Allen Simpson. *The ESP DES-CBC Transform.* RFC 1829, *IETF*, August 1995.

[Kob87] N. Koblitz. Elliptic Curve Cryptosystems. *Mathematics of Computation*, 48(177):203–209, 1987.

[KS97a] Phil Karn and William Allen Simpson. *Photuris: Extended Schemes and Attributes <draft–simpson–photuris–schemes–03.txt>.* Internet Draft, *Network Working Group*, July 1997. work in progress.

[KS97b] Phil Karn and William Allen Simpson. *Photuris: Session–Key Management Protocol <draft–simpson–photuris–16.txt>.* Internet Draft, *Network Working Group*, September 1997. work in progress.

[Mil86] V. S. Miller. Use of Elliptic Curves in Cryptography. In *Advances in Cryptology — CRYPTO '85*, 1986.

[Mit97] Suvo Mittra. Iolus: A Framework for Scalable Secure Multicasting. In *SIGCOMM '97*, ACM, 1997.

[MMP97] Daniel L. McDonald, Craig Metz, and Bao G. Phan. *PF_KEY Key Management API, Version 2 <draft–mcdonald–pf–key–v2–04.txt>.* Internet Draft, *Network Working Group*, September 1997. work in progress.

[MS95] Perry Metzger and William Allen Simpson. *IP Authentication using Keyed MD5.* RFC 1828, *IETF*, August 1995.

[MSTS97] Douglas Maughan, Mark Schneider, Jeff Turner, and Mark Schertler. *Internet Security Association and Key Management Protocol (ISAKMP) <draft–ietf–ipsec–isakmp–08.txt>.* Internet Draft, *IPSEC Working Group*, July 1997. work in progress.

[Orm96] Hilarie Orman. *The OAKLEY Key Determination Protocol <draft–ietf–ipsec–oakley–02.txt>*. Internet Draft, *IPSEC Working Group*, 1996. work in progress.

[RSA78] R. Rivest, A. Shamir, and L. Adleman. A Method for Obtaining Digital Signatures and Public Key Cryptosystems. In *Communications of the ACM*, ACM, 1978.

[Sch96] Bruce Schneier. *Applied Cryptography, Second Edition. John Wiley and Sons, Inc.*, 1996.