第 13 部

IP 層におけるセキュリティ技術

第 1 章

WHIPS

WHIPS(WIDE Happy IP Security) 分科会の目的は、「IPsec 準拠の IP security 機構の実装と検証」である。WHIPS は、1996 年秋の合宿から 1 年間の予定で活動を開始した。発足当時は、まず 1997 年の春合宿までに IPsec の基本仕様である AH(Authentication Header: 認証ヘッダ) と ESP(Encapsulating Security Payload:暗号ペイロード) を実装して相互接続性を検証し、残りの半年で鍵交換プロトコルの開発に取り組むという計画を立てた。1997 年の春合宿において、計画通り (株) 東芝、ヤマハ (株)、および、横河電機 (株) それぞれ独自の AH と EPS の実装に関して相互接続性をテストした。しかしながら、AH と EPS の仕様が変更されたことや相互接続性に若干問題があったことから、残りの半年でも AH や EPS の相互接続性を検証していく必要がある。

この章では、まず最新の AH と EPS を含めた IPsec の概要を解説し、次に分科会の成果である 3 つの実装と相互接続性テストの結果について述べる。

1.1 AH と ESP

IPsec はもともと IPv6 用のセキュリティ機構として標準化が始まったが、RFC 1825 〜 1829 として発行されるまでには IPv4 でも利用できる拡張ヘッダ形式に落ち着いた。IPsec で AH(RFC1826) と ESP(RFC1827) を分割した本質的な理由は、暗号の利用を制限している国でも認証機構だけは利用できるようにという配慮からである。ESP では適切な変換を施せば認証や完全性保護の機能も利用できる。しかしながら、RFC 1829 で定められているように必須の標準の変換 (ESP DES–CBC 変換) には認証や完全性保護の機能は無く、ESP は AH と組み合わせて使うように設計されていた。

その後 IPsec のメーリングリストで、ESP DES–CBC 変換などに暗号の性質に起因した弱点が報告された。たとえば、ESP DES–CBC 変換を用いて TCP をセッションを保護する場合を考える。このとき、DES–CBC の性質からセッション中の任意の文字列を他の文字列に置き換えることなどが可能である。そこで、暗号の機能を利用する場合は、同時に完全性を保護する必要があること分かった。

ESP で常に完全性を保護するためには、AH と組み合わせて利用するよりも完全性を保護する機能をもった変換を定めた方がよい。そこで、現在では AH は IP ヘッダを認証や

完全性保護を実現するためのみに用いることになった。さらに、AH や ESP ともにリプレイ攻撃から保護するために、通し番号を付加する必要があること分かった。

このような背景のもとに、現在新しい AH と ESP、そしてそれぞれの変換がインターネット・ドラフトとなり議論されている。以下最新の AH と ESP の書式について説明する。

1.1.1 AH

図 1.1に最新の AH の書式を示す。各フィールドの意味は以下の通りである。

次ヘッダ 引き続くヘッダの種類を示す。1 バイト。必須。

ペイロード長 AH の長さを 4 バイト単位で表す。長さには固定長である最初の 8 バイトは含まない。1 バイト。必須。

SPI Security Parameter Index。4 バイト。必須。

通し番号 対リプレイ攻撃用の通し番号。セキュリティ・アソシエーションが確立された際に 1 に初期化される。数値を使い切った場合は、セキュリティ・アソシエーションを再び確立する。4 バイト。選択 (必須になる可能性が高い)。

認証データ 変換で生成される認証データ。4 バイトを単位とした可変長。

図 1.1: AH の書式

1.1.2 ESP

図 1.2に最新の ESP の書式を示す。各フィールドの意味は以下の通りである。

SPI Security Parameter Index。4 バイト。必須。

通し番号 対リプレイ攻撃用の通し番号。セキュリティ・アソシエーションが確立された際に 1 に初期化される。数値を使い切った場合は、セキュリティ・アソシエーションを再び確立する。4 バイト。選択 (必須になる可能性が高い)。

初期ベクトル 暗号アルゴリズムなどによって必要とされるデータ領域。1 バイトを単位とした可変長。選択。

ペイロード 変換を施したペイロード。次ヘッダによって内容が指定される。1 バイトを単位とした可変長。必須。

パディング 暗号化する場合は暗号アルゴリズムによって要求されるブロック長に合わせるためのデータ領域。暗号化しない場合は、4 バイト境界に合わせる。0 〜 255 バイト。選択。

パディング長 パディングの長さ。1 バイト。必須。

次ヘッダ ペイロードの内容。1 バイト。必須。

認証データ 変換で生成される認証データ。1 バイトを単位とした可変長。選択。

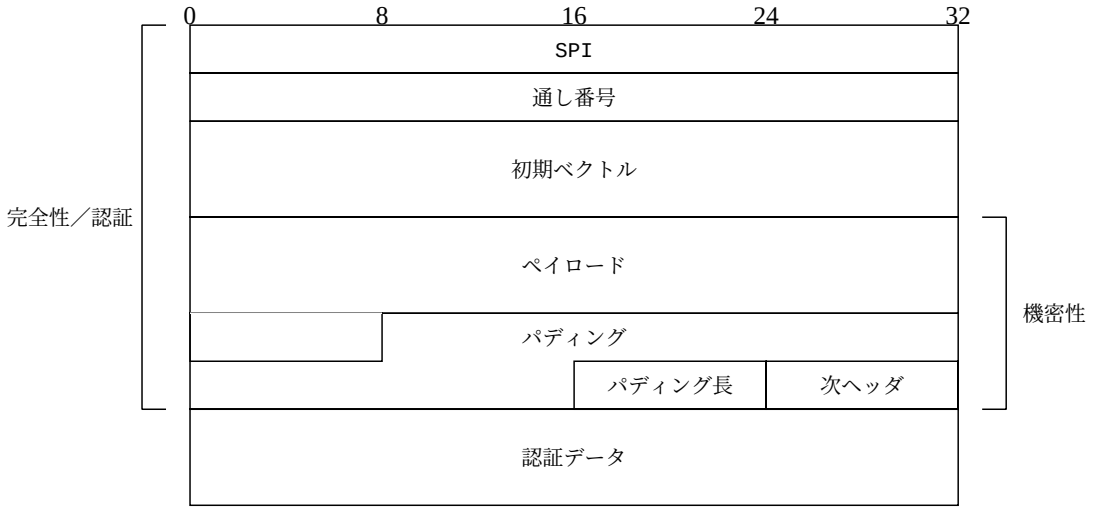

図 1.2: ESP の書式

1.2　(株) 東芝の実装

東芝では、mobile IP と IPsec の協調動作を目指している。このため、容易に修正や変更、拡張が行なえることが望ましかった。また、他の OS などへの移植が容易にできるこ

とも要望の 1 つであった。そこで、機能のほとんどをユーザー空間で動作するアプリケーションとして実装した。

1.2.1 東芝の実装の概要

東芝では、IPv4 を対象とした IPsec を BSD/OS 2.1 上で実装した。対応したプロトコルは次の通りである。

AH — Keyed-MD5(RFC1828)、Keyed–SHA1(RFC1852)

ESP — DES-CBC(RFC1829)、3DES–CBC(RFC1851)

鍵管理プロトコル — SKIP(draft-ietf-ipsec-skip-07.txt)

MD5 は基本的に RFC1321 のコードを使用している。それ以外の暗号／認証関連のアルゴリズム、すなわち SHA1、DES および 3DES、Diffie-Hellman はオリジナルの実装を使用している。

1.2.2 東芝の実装の制限

現在の実装での制限を以下に示す。

モード — トランスポート・モードは実装されていない。したがって、通信当事間の通信でも新しい IPv4 ヘッダが付加される。AH のみを使用するときは、AH の次ヘッダは IPv4 となる。

keying — 現在の実装では、セキュリティ・アソシエーションをホスト単位 (host-keying) でしか選択できない。したがって、ユーザごとに (user-keying)、あるいはセッションごとに (session-keying) 鍵を登録／変更できない。また、セキュリティ・アソシエーションの登録／削除／変更について、他プロセスとのインターフェイスは用意されていない。

経路 MTU **探索** — 経路 MTU 探索には対応していない。したがって、DF(Don't Fragment) ビットが立っている IP パケットを転送する際に IPsec でカプセル化する場合には、カプセル化されたパケットが分割されて転送される。本実装をセキュリティ・ゲートウェイとして使用したときには、通信当事間のホストが経路 MTU 探索を実装していたとしても、セキュリティ・ゲートウェイ間では分割が発生する。

1.2.3　東芝の実装方式

鍵管理プロトコル SKIP、AH および ESP のカプセル化／カプセル開放はすべてユーザ空間で動作するアプリケーション・プログラムとして実装した。また、カーネルからこの主モジュールへパケットを渡すために、PCD(Packet Capture Device) というデバイスを実装した。

ユーザ空間で動く主モジュールは、プロトコル処理部、セキュリティ・アソシエーション・マネージャ、ポリシー・マネージャの 3 つの部分に分かれる (図 1.3)。

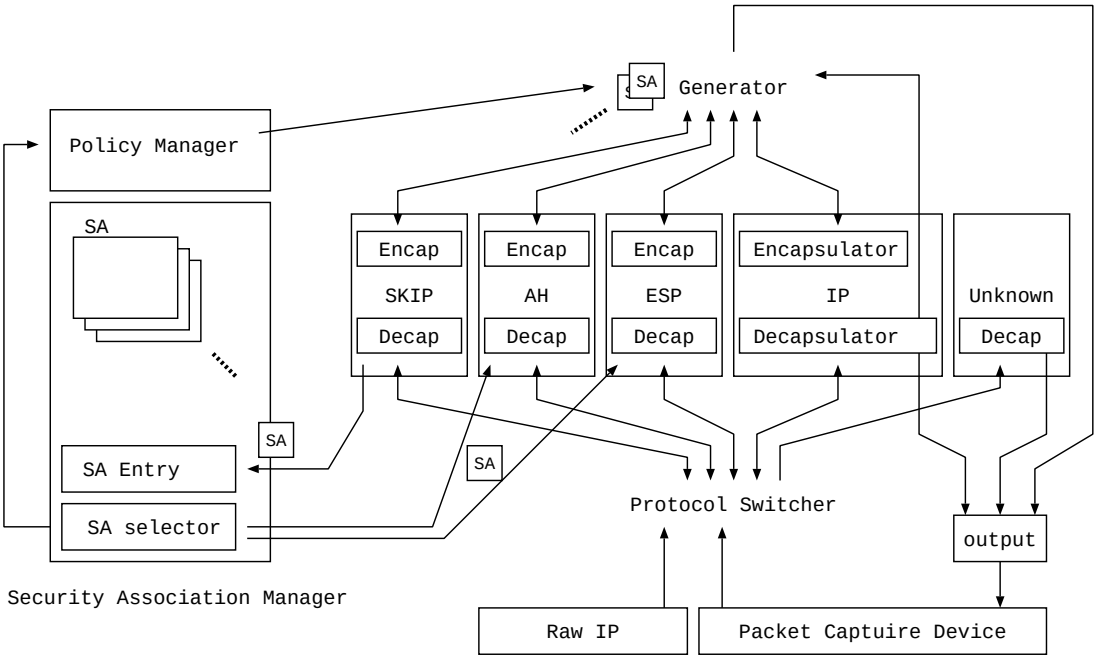

図 1.3: 東芝の実装の概念図

プロトコル処理部は各プロトコルを理解し処理する。この中は大きく受信時に使用されるカプセル開放器と、送信時にパケットを作るために使用されるカプセル格納器に分けれらる。

セキュリティ・アソシエーション・マネージャは、セキュリティ・アソシエーションの登録、削除、選択などを受け持つ。現在の実装では、セキュリティ・アソシエーションには AH、ESP 双方のための鍵、アルゴリズム等の情報が含まれている。AH と ESP を同時に使用するときには同じセキュリティ・アソシエーションを使うようになっている (本来は異なるべきである)。ただし、ESP のための SPI を指定して、(同じセキュリティ・アソシエーションを使っているが)AH ／ ESP で異なる SPI を使用できる。

ポリシー・マネージャは送信の際にどのセキュリティ・アソシエーションを使用するかを決定する機構である。ただし、現在の実装では host-keying しかできないので、IPsec の

処理を施すパケットに対して、どのゲートウェイのセキュリティ・アソシエーションを使用すればよいかしか判断していない。

パケットが入力されると、それはまず IP プロトコル処理部に渡される。IP プロトコル処理部は、入力されたパケットが (1) パケットを転送すべきか、(2) カプセル化するべきか、(3) カプセルを開放すべきかを判断する。

カプセル化する場合は、パケットは発生器に渡される。発生器は、ポリシー・マネージャからそのパケットに適切なセキュリティ・アソシエーションを受け取る。次に発生器は、セキュリティ・アソシエーションの内容に従い、パケットをカプセル格納器に渡していく。最後に、カプセル格納器は IPsec のカプセル化を行なう。

カプセル開放の際は、パケットに付けられているヘッダの順に各プロトコル処理部のカプセル開放器に渡される。AH、ESP ではそのカプセル開放器の中でセキュリティ・アソシエーション・マネージャから入力されたパケットに合うセキュリティ・アソシエーションを受け取って、その内容に従ってカプセルを開放する。

PCD (図 1.4) は、カーネルが出力するパケットをアプリケーションへ渡すためのデバイスであり、基本的にはトンネル・デバイスを拡張したものである。PCD では BPF(Berkeley Packet Filter) のようにパケットのコピーを渡すのではなく、パケットを “横取り” でき、アプリケーションが取ったパケットは出力されない。

4.4BSD の実装において、IP パケットの出力関数である`ip_output()` は IP パケットを実際に送信するためにインターフェイスの出力関数を呼ぶ。実際には`ifnet{}`構造体中の`if_output[]` に出力関数へのポインタが保存されており、`ip_output()` はこの`if_output[]` を参照する。

PCD は、自分が`open()` されると、すべての`ifnet{}`構造体を走査し、`if_output[]` の値を PCD 内の配列`if_output[]` へ保存し、同時に`if_output[]` を PCD の出力関数へと変更する。この操作によってインターフェイスに出力されるすべてのパケットは PCD に渡されることになる。

PCD は簡単なパケット・フィルタを持っており、PCD を使うアプリケーションは自分がどのようなパケットを必要としているかをこのフィルタに設定しておく。現在の実装では、始点と終点のアドレスを条件に指定できる。上位層が必要としていないパケットはそのまま、本来出力されるはずであったインターフェイスの出力関数へと渡される。そうでなければ、PCD 内のパケット・キューに入れられる。ユーザ・プロセスがキャラクタ・デバイスとしての PCD から`read()` すると、このキューからパケットが上位層に渡される。また、キャラクタデバイスとしての PCD に `write()` すると、これは `ip_output()` を通して出力される。ただし、PCD に `write()` されたパケットがもう 1 度上位層に渡ることがないようになっている。

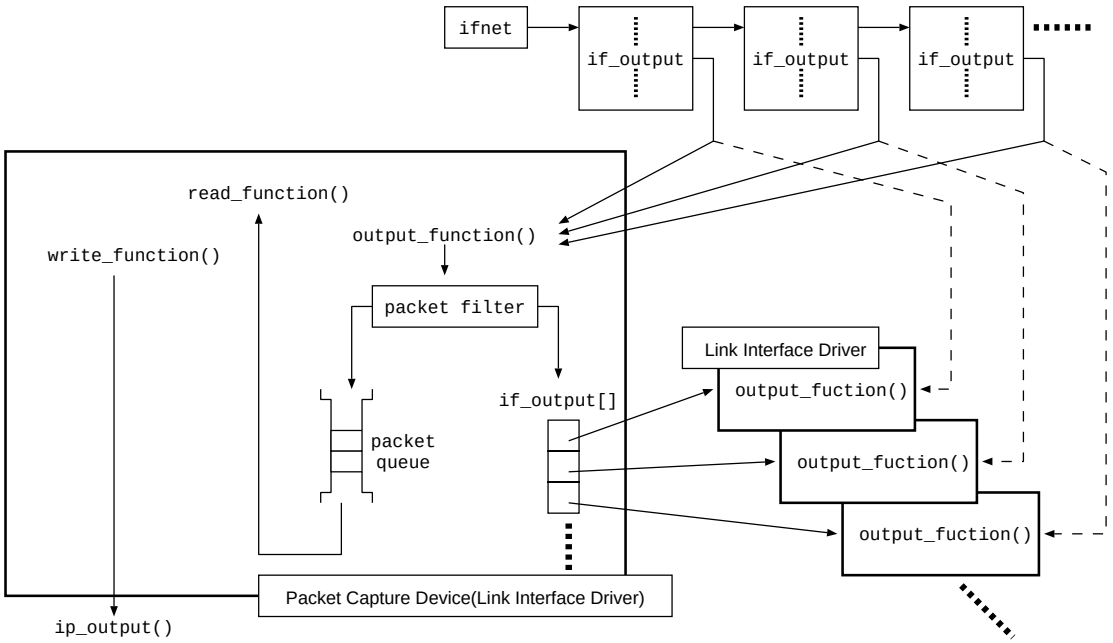

図 1.4: Packet Capture Device

1.2.4　東芝の性能

本実装の性能を測定するために、2 台のホスト間での TCP の転送速度を測定した。測定には Netperf[1] を使用した。テストに使用したホストは 2 台とも BSD/OS 2.1、Pentium 100MHz である。結果を表 1.1に示す。

(1) は、IPsec モジュールを全く使用しない場合の速度である。(2) から (6) は鍵管理プロトコル SKIP を使用しない場合、(7) から (11) は SKIP を使用した場合の速度である。(1) と (2) の速度差はかなり大きい。MD5 の計算量はそれほど大きくないことを考えると、速度低下の大部分はおそらくカーネル空間とユーザ空間とのデータ交換のオーバヘッドであろうと考えられる。(2)〜(6) と (7)〜(11) をそれぞれを比較すると、それほど速度差がないことがわかる。SKIP では鍵情報をヘッダ内に埋め込み、これを暗号化[2]する。そのため、各送受信パケットそれぞれについて SKIP を使用することによるオーバヘッドが発生するが、この結果をみるとこのオーバヘッドはそれほど問題にならないといえる。(3)〜(6), (9)〜(11) をみると、暗号化の処理は非常に重く、最終的にこれが転送速度を決定している。

[1] URL: http://www.cup.hp.com/netperf/NetperfPage.html
[2] 本実験では暗号化に Triple-DES を使用した。

	プロトコル			Mbit/sec
(1)	IP (no IPsec)			8.32
(2)		AH(MD5)		3.89
(3)			ESP(DES)	2.43
(4)		AH(MD5)+	ESP(DES)	2.24
(5)			ESP(TDES)	1.43
(6)		AH(MD5)+	ESP(TDES)	1.35
(7)	SKIP+	AH(MD5)		3.77
(8)	SKIP+		ESP(DES)	2.42
(9)	SKIP+	AH(MD5)+	ESP(DES)	2.25
(10)	SKIP+		ESP(TDES)	1.43
(11)	SKIP+	AH(MD5)+	ESP(TDES)	1.37

表 1.1: 性能測定の結果

1.2.5 東芝の実装の課題

現在の実装の制限を順次解決していきたい。特に、現在はホスト単位でしかセキュリティ・アソシエーションを定められないが、セッション／ユーザ単位でのセキュリティ・アソシエーションの選択を可能にすべきである。現在提案されている Simple IPsec API に類似した機構を実装したいと考えている。このため、現在は IPsec をサポートするほとんどすべてのモジュールおよびデータがユーザ空間にあるが、この内のいくつかはカーネルに移動しなければならなくなると考えている。また現在我々は、独自に IPv6 のプロトコルスタックを実装しており、これにも IPsec を組み込む予定である。IPv6 では IPsec は拡張ヘッダとして定義されているので、アプリケーションとしての実装が難しい。我々には高い移植性という要求があるため、ユーザ／カーネル空間の切り分けについて考えていきたい。

1.3 ヤマハ (株) の実装

ヤマハ (株) では、ISDN リモート・ルータ RT100i ／ RT102i ／ RT200i (以降、RT と呼ぶ) を基にして、IPv4 用の IPsec を試験的に実装した。AH の変換には Keyed MD5 の 1 種類、ESP の変換には DES–CBC および 3DES–CBC の 2 種類を実装した。また、鍵管理プロトコルは未実装で、鍵の管理は手動である。

なお、MD5 は PGP バージョン 2.1.2 に付属するフリーのコードを、DES は LSI ジャパンの森公一郎氏が作成したフリーのコードを改造して利用している。

1.3.1　ヤマハの実装の詳細

RT 上で動いているファームウェアは BSD などの UNIX ベースではなく、まったく独自である。そのため、実装の詳細については図を用いて説明を進めていくこととする。

図 1.5は RT での IP パケット処理の概念図である。入力インタフェースから入って来たパケットはまず入力フィルタを通り、そこで落すべきパケットを落す。

次に IP ルーティング・エンジンを通してパケットの送り先を決定する。この送り先には自らの上位層も含まれている。送り先が外部への出力となった場合、出力フィルタで選別をかけられ、通過したパケットだけが出力インタフェースから出力される。

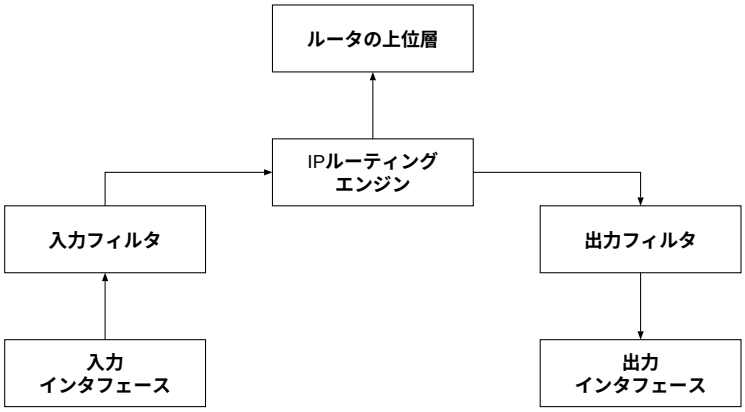

図 1.5: RT における IP パケット処理

この構造に IPsec を組み込んだのが図 1.6である。IP 経路制御エンジンの入口にカプセル格納器が、自らの上位層にはカプセル開放器が付加される。カプセル開放器で処理されたパケットはまた、IP パケット処理の入口に戻される。

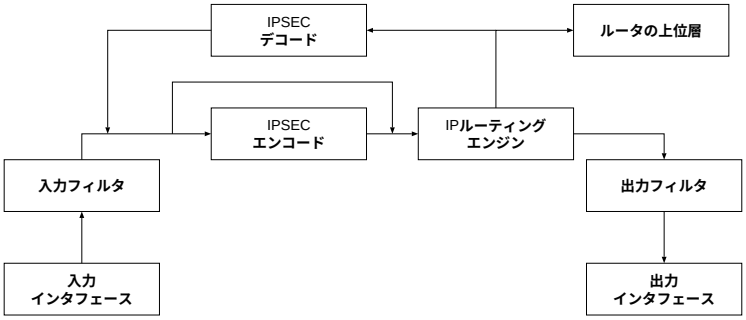

図 1.6: IPsec モジュールを入れたパケット処理

カプセル格納器は、どのパケットをカプセル化すべきかをフィルタと同じルールにより

選別する。これにより、パケットの始点／終点アドレスはもとより、TCP ／ UDP などのプロトコルやポート番号など、カプセル化すべきパケットを細かく指定できる。

カプセル開放器は、IPsec パケットとして妥当なものであればすべてカプセル開放を試みる。ここではどこからパケットが送られてきたかなどは一切考慮しない。必要があれば、これらを入力フィルタで処理する。

カプセル開放器からのパケットが IP パケット処理の入口に差し戻されることからも分かるとおり、RT の実装では IP ヘッダごとカプセル化するトンネル・モードが必須となっている。いわゆる AH トンネルは実装されていないため、ESP トンネルを用いる必要がある。理由は RT をルータとして IPsec トンネルを構築することを主眼に実装を進めたためであり、ルータ自らの上位層が IPsec を用いて通信することはあまり考えられていないからである。

1.3.2 ヤマハの実装における AH と ESP の独立性

ESP トンネルが必須となっているため、コードとしても AH は ESP から切り離しては使えないようになっている。これは単に最終的な機能の実現のために余計な部分を削ったためであり、技術的には AH のコードを ESP から切り離すことは問題ない。将来 AH トンネルもサポートするようになれば切り離すべきであろう。ヤマハでは今のところ RT 以外の実装へは考えていないため、上位層で IPsec を使いやすくするように改造する予定はない。

1.3.3 ヤマハの実装の特徴

ヤマハの実装の特徴は AH 単独では使えず、常に ESP を必要とすることである。これは、IPsec の用途を IPsec トンネルを張ることに限定したためであり、ルータ自身の上位層が IPsec の端点となって動作することをほとんど考慮していない。(しかしながら、相互接続テストのために必要となるため、実際には ping などが IPsec 上で実行できる。)

また、フィルタと同様なパケット選択方式を取っているため、IPsec を利用するパケットを細かく指定できるのも特徴である。これにより、たとえばメールは ESP で暗号化したいが、WWW は暗号化せず転送速度を低下させないなどの細かい設定が可能である。

欠点としては、鍵管理プロトコルを実装していないにも関わらず、暗号アルゴリズムとして DES–CBC などの鍵長の短いものしか用意していない点があげられる。鍵長の短いアルゴリズムは暗号化／復号化にかかる時間が短くてすみ、RTT(Round Trip Time) をあまり増加させないことが利点ではあるが、その分解読されやすい。暗号としての強度をあげるためには定期的に鍵を変更する必要があるが、手動での設定は適切ではない。よって、実用化のためには鍵管理プロトコルの実装が必須であると考えられる。

1.4　横河電機 (株) の実装

横河電機の YIPS(Yokogawa IP Security) は、組織間の安全な通信を保証するための IPsec の実装である。YIPS は、IPsec アーキテクチャ(RFC1825) を採用し、アドレスとマスク、ポート番号を元にしたセキュリティ・アソシエーションを作成する。また、プロキシ・アドレスを組み合わせることにより、セキュリティ・ゲートウェイとしての動作も可能である。セキュリティ・アソシエーションを構成するセキュリティ・プロトコルとして ESP(RFC1827)、変換は DES-CBC(RFC1829) を採用[3]している。さらに、送受信に関するポリシーをシステムに対して設定可能にすることにより、様々なポリシーに対応できる。

YIPS は BSD/OS 2.1 patch-level 27、libdes 3.2 以上で動作する。YIPS は、セキュリティ・アソシエーションを管理する部分とセキュリティ・プロトコルを処理する部分に分類できる。

1.4.1　YIPS の制限

YIPS の制限を以下に示す。

- IPv4 上での実装である。IPv6 に関しては今後対応するかもしれない。
- ESP トランスポート・モードはサポートしない。セキュリティ・ゲートウェイは、IP ヘッダさえも機密性を保たなければならない場合がある。トランスポート・モードを実装する手間と、これを採用したときのセキュリティ・ゲートウェイにおける利点を比べて実装しないことにする。ただし、今後帯域の問題に直面した場合に考慮するかもしれない。
- AH はセキュリティ・アソシエーション管理において指定はできるが、カーネルには実装していない。これは実装している途中で、認証が必要な場合は ESP の適切な変換を使用すればよいと判断したためである。ただし、今後中継認証などの認証のみを必要とする問題に直面した場合に考慮するかもしれない。
- セキュリティ・アソシエーション管理をする方法については手動方式のみをサポートする。その他のセキュリティ・アソシエーション管理の方法については今後考慮しなければならない。
- 本実装でのマスクは、連続しているもののみ使用可能とする。
- マルチキャスト、マルチレベル・セキュリティ、IPSO、QoS については、考慮していない。

[3] 3DES-CBC(RFC1851) も実装したが、アルゴリズムを正確に理解していなかったため WIDE における相互接続テストに失敗した。今後見直す予定。

1.4.2 YIPS のセキュリティ・アソシエーション管理機構

YIPS では、ユーザ空間からセキュリティ・アソシエーションを管理するために、PF_KEY[4] と呼ばれるインターフェイスを採用した。これは、route(4) と同様に RAW ソケットを介してカーネルと通信する機構である。以下に使用可能なメッセージについて説明する。

YIPS では以下の様なメッセージが使用できる。

SA_ADD — カーネルに対し、新規セキュリティ・アソシエーションの登録を要求する。

SA_DELETE — カーネルに対し、指定したセキュリティ・アソシエーションの削除を要求する。カーネルは対応するセキュリティ・アソシエーションの状態を SA_DEAD とするだけである。

SA_GET — カーネルに対し、指定したセキュリティ・アソシエーション情報の返答を要求する。カーネルは、対応するセキュリティ・アソシエーションの情報をユーザへ返す。

SA_DUMP — カーネルは、全てのセキュリティ・アソシエーションの情能をユーザへ返す。

SA_FLUSH — カーネルは、全てのセキュリティ・アソシエーションを削除する。

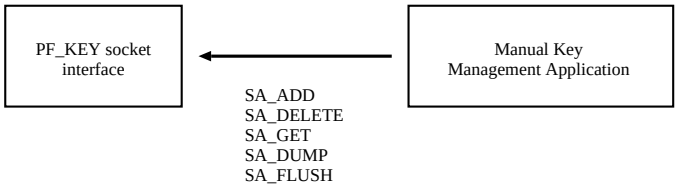

図 1.7: 手動のセキュリティ・アソシエーション管理で使用可能なメッセージ

これらは全て手動のセキュリティ・アソシエーション管理に必要なメッセージである (図 1.7)。今後動的なセキュリティ・アソシエーション管理 (図 1.8) が必須となるので、以下の様なメッセージも考慮し実装していかなければならない。

SA_GETSPI — カーネルは、システム内で一意な新規 SPI を取得し、セキュリティ・アソシエーションの状態を SA_LARVAL としてセキュリティ・アソシエーション管理テーブルに登録する。取得した SPI をユーザへ返す。

SA_UPDATE — SA_GETSPI で取得した SPI を持つセキュリティ・アソシエーションを上書き更新し、セキュリティ・アソシエーションを使える状態にする。

[4] draft-mcdonald-pf-key-v2-00.txt

SA_ACQUIRE — カーネルが、IPsec を適用するために必要なセキュリティ・アソシエーションの生成要求をする。

SA_REGISTER — SA_ACQUIRE を受けるためにカーネルへプロセス ID を登録する。

SA_EXPIRE — セキュリティ・アソシエーションの生存期限が切れたことをセキュリティ・アソシエーション管理プロセスへ知らせる。

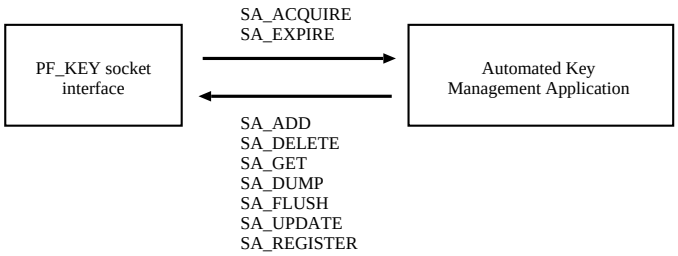

図 1.8: 自動セキュリティ・アソシエーション管理で使用可能なメッセージ

カーネル内では、以下の様なセキュリティ・アソシエーションの状態を管理する。また、これらの関係を図 1.9に示す。

SA_SPAWN — カーネル内でセキュリティ・アソシエーションが割り当てられ、初期化時に一時的に存在する状態である。

SA_USED — 使用可能または使用中。SA_USED 状態のセキュリティ・アソシエーションだけが、SA_GET コマンドに応答する。

SA_LARVAL — SPI は割当済みだが、セキュリティ・アソシエーションとしては未完成な状態。

SA_ZOMBIE — 期限切れだが、使用可能。

SA_DEAD — 削除待状態。

外向きのセキュリティ・アソシエーションは、以下の条件で全て一意に識別される。

- 終点アドレス/終点マスク
- 始点アドレス/始点マスク
- 終点ポート
- 始点ポート

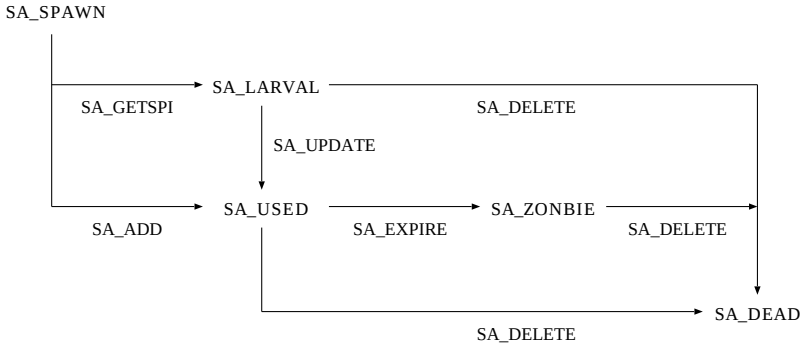

図 1.9: セキュリティ・アソシエーションの状態遷移図

- 次ヘッダ

内向きのセキュリティ・アソシエーションは、終点アドレス、または終点プロキシ・アドレスが自ホストならば SPI のみで一意となる。それ以外は、外向きのトラフィックと同様である。

アドレス、マスク、上位層プロトコルとポート番号、プロキシ・アドレスを組み合わせることにより、以下ようなセキュリティを提供できる。当然これ以外の組合せも考えられる。

- ホスト単位 — 与えられた IP アドレス同士の通信に対して、1 つのセキュリティ・アソシエーションを割り当てる。
- セッション単位 — 与えられた IP アドレス、上位層プロトコル、ポート番号に対して 1 つのセキュリティ・アソシエーションを割り当てる。
- セキュリティ・ゲートウェイ — 与えられたサブネットまたはホストの通信に対して、セキュリティを提供する。

1.4.3 セキュリティ・プロトコル処理部

処理の概要を図 1.10に示す。

送信処理はまず、`ip_output()` のラベル sendit の直後、分割処理が行なわれる前で `ipsec_output()` を呼び出す。`ipsec_output()` では、ホストに設定されたセキュリティ・ポリシーを検査し、それに従い送信処理方法を決定する。アドレスとマスク、ポート番号からセキュリティ・アソシエーションを検索し、それに従い処理する。

受信処理はまず、`ipintr()` でホストに設定されたセキュリティ・ポリシーを検査し、それに従い受信処理方法を決定する。再構成した後、次ヘッダが `IPPROTO_ESP` ならば、`ipsec_input()` を呼び出す。`ipsec_input()` では、SPI によりセキュリティ・アソシエー

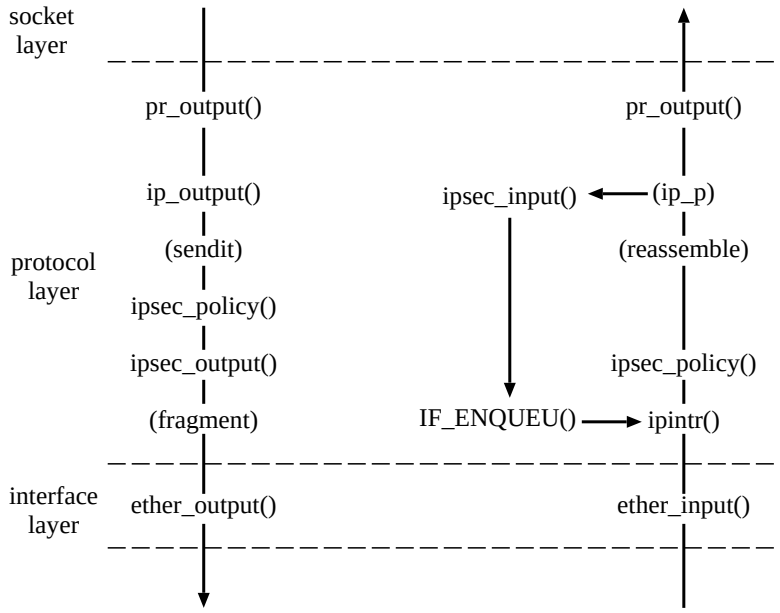

図 1.10: 横河の実装での処理の概要図

ションを検索し、それに従い処理する。セキュリティ・アソシエーションを取得できないパケットは破棄し、`icmp_error()` を呼び出す。

1.4.4　セキュリティ・ポリシー

システムに以下のポリシーレベルを用意し、デフォルトの動作を指定可能とする。これらは、`ip_output()` と `ipintr()` で検査される。

`IPSEC_LEVEL_NONE` — 外向きのパケットは IPsec を適用しない。内向きのパケットは無条件で受信処理を行なう。

`IPSEC_LEVEL_AVAIL` — 外向きのパケットに対するセキュリティ・アソシエーションが存在すればこれを使用する。それ以外は `IPSEC_LEVEL_NONE` と同様。内向きのパケットは無条件で受信処理を行なう。

`IPSEC_LEVEL_WALL` — 外向きのパケットに対するセキュリティ・アソシエーションが存在すればこれを使用する。それ以外は `IPSEC_LEVEL_NONE` と同様。内向きのパケットは IPsec の適用を必須とする。つまり IPsec が適用されていないパケットは受信処理を行なわない。

`IPSEC_LEVEL_USE` — 外向きのパケットに対するセキュリティ・アソシエーションが存在すればこれを使用する。セキュリティ・アソシエーションが存在しなければセキュリ

ティ・アソシエーション確立要求[5]を出し、確立された後これを使用する。内向きのパケットは無条件で受信処理を行なう。

IPSEC_LEVEL_REQUIRE — 外向きのパケットに対するセキュリティ・アソシエーションが存在すればこれを使用する。セキュリティ・アソシエーションが存在しなければセキュリティ・アソシエーション確立要求を出し、確立された後これを使用する。内向きのパケットは IPsec の適用を必須とする。

IPSEC_LEVEL_UNIQUE — 外向きのパケットに対するセキュリティ・アソシエーションが存在すればこれを使用する。セキュリティ・アソシエーションが存在しなければセキュリティ・アソシエーション確立要求を出し、確立された後これを使用する。さらに、そのセキュリティ・アソシエーションが他のセッションに使用されないことを保証する。内向きのパケットは IPsec の適用を必須とする。

以上のセキュリティ・ポリシーは、今後エンド・ノードのためにソケット単位で設定可能とする予定である。また、ソケット単位のセキュリティ・ポリシーが設定可能となった場合には、以下のそれも設定可能とする予定である。

IPSEC_LEVEL_DEFAULT — システムのデフォルトを使用する。

IPSEC_LEVEL_BYPASS — システムのいかなるセキュリティ・ポリシーをもバイパスする。セキュリティ・アソシエーション管理デーモンなどの特権アプリケーションにのみ設定可能とする。

1.4.5 YIPS の課題

以下に今後考慮しなければならない大きな問題を列挙する。

- 認証と対リプレイ攻撃を伴った ESP の変換[6] の実装。
- セキュリティ・アソシエーションと鍵の生存時間。
- 安全な通信路を確保する準備としてセキュリティ・アソシエーションを交換しなければならない。現状は手動のセキュリティ・アソシエーション管理方式であるが、大規模なネットワークを考えると動的な管理方式が必須となる。今後はこの方式を検討し実装する必要がある。
- ユーザが独自にセキュリティ・ポリシーを設定できるよう、ソケット単位での設定に対応するべきである。

[5] 現状のセキュリティ・アソシエーション管理方法は手動方式のみなので、手で設定する必要がある。
[6] draft-ietf-ipsec-esp-des-md5-03.txt

- カーネルではセキュリティ・アソシエーションを線形で管理している。管理しなければならないセキュリティ・アソシエーションが増えるにしたがい性能にに影響が出るはずである。セキュリティ・アソシエーション管理の最適化が必要だろう。

1.5　1996 年 WIDE 春合宿での相互接続実験

1996 年 WIDE 春合宿において、(株) 東芝、ヤマハ (株)、横河電機 (株) からそれぞれ IPv4 用の IPsec の実装を遠鉄エンパイア・ホテル内の一室に持ち寄り、1996 年 3 月 17 日から 18 日にかけて相互接続性を検証した。それぞれの実装の概要を表 1.2に示す。

	AH	ESP	鍵管理	備考
東芝	Keyed MD5	DES-CBC, 3DES-CBC	SKIP	BSD/OS 2.1 のユーザプログラムとして実装
ヤマハ	Keyed MD5	DES-CBC, 3DES-CBC	なし	ISDN リモートルータ (独自 OS) に実装
横河	なし	DES-CBC, 3DES-CBC	なし	BSD/OS 2.1 のカーネルに実装

表 1.2: 実装概要

接続試験は 1 台のハブに 3 つの実装 (東芝、横河はノート PC、ヤマハは RT100i) を接続し、お互いに ping、traceroute を用いて接続を確認するという方法で行った。また、東芝と横河の間は telnet でも試験を行った。パケットモニタには全く別の PC 上で動作する tcpdump を利用した。

接続試験に用いたアルゴリズムは、AH としては Keyed MD5、ESP としては DES-CBC と 3DES-CBC を試験した。鍵管理プロトコルは東芝しか実装していなかったため、今回は試験の対象外とした。

ESP(DES–CBC) では 3 者間で相互に通信できた。ESP(3DES–CBC) ではヤマハと東芝間で通信できた。ESP(DES–CBC) + AH(Keyed MD5) では、ヤマハと東芝間で通信できた。これらの試験結果を表 1.3〜1.5にまとめる。なお、ping や traceroute などの試験方法による差はみられなかったので、それらに関する情報は載せていない。

結果的にほとんどの項目で接続を確認できたが、横河とヤマハは試験期間中に相当量のコードを書き直したり書き足したりした。はじめ接続ができなかった原因としては、RFC やアルゴリズムに対する理解が間違っていたことが挙げられる。東芝はほとんど変更無しであった。

速度に関しては評価してないが、実際にテストした感触では IPsec なしに比べるとかなり遅くなる。実用上ではいかに速いハードウェア提供できるかが鍵になってくるであろう。

→	東芝	ヤマハ	横河
東芝	—	○	○
ヤマハ	○	—	○
横河	○	○	—

表 1.3: ESP(DES-CBC)

→	東芝	ヤマハ	横河
東芝	—	○	×
ヤマハ	○	—	×
横河	×	×	—

表 1.4: ESP(3DES-CBC)

→	東芝	ヤマハ	横河
東芝	—	○	—
ヤマハ	○	—	—
横河	—	—	—

表 1.5: ESP(DES-CBC)+AH(Keyed MD5)