IPsec と IKE の管理

Sun Microsystems, Inc.
4150 Network Circle
Santa Clara, CA 95054
U.S.A.

Part No: 817–1179–10
2003 年 4 月

原典: *IPsec and IKE Administration Guide*

Part No: 816-7264-10

Revision A

030304@5533

目次

はじめに

『*IPsec* と *IKE* の管理』は、『*Solaris* のシステム管理 *(IP* サービス*)*』の第 19、20、21 章を更新したものです。 本書では、次の作業がすでに終わっているものとします。

- SunOS™ 5.9 オペレーティングシステムのインストール
- SunOS 5.9 オペレーティングシステムの Solaris 9 4/03 リリースへのアップデート
- 使用するネットワークソフトウェアの設定

SunOS 5.9 オペレーティングシステムは Solaris 製品ファミリの一部であり、Solaris 共通デスクトップ環境 (CDE) などが含まれます。また、SunOS 5.9 は、AT&T System V リリース 4 オペレーティングシステムに準拠しています。

注 – Solaris オペレーティングシステムは、SPARC® と x86 の 2 種類のハードウェア (プラットフォーム) 上で動作します。 また、Solaris オペレーティングシステムは、64 ビットと 32 ビットの両方のアドレス空間で動作します。このマニュアルの情報は、両方のプラットフォームと両方のアドレス空間に適用されます。例外がある場合は、特別な章、節、注、箇条書き、図、表、例、またはコード例で、その旨を明記します。

対象読者

このマニュアルは、Solaris 9 リリースを実行するシステムの管理者を対象にしています。このマニュアルを活用するには、1、2 年程度の UNIX® システムの管理経験が必要です。UNIX システム管理トレーニングコースへの参加が役立つことがあります。

内容の紹介

第 1 章 では、IP セキュリティアーキテクチャの概要を説明します。IP セキュリティアーキテクチャ (IPsec) は、IP データグラムを保護します。

第 2 章では、ネットワークに IPsec (IP セキュリティ) を実装する手順について説明します。

第 3 章 では、IPsec で使用する Internet Key Exchange の概要を説明します。

第 4 章 では、IKE を実装するための手順を説明します。

用語集では、IP セキュリティに関する主な用語について説明します。

Sun のオンラインマニュアル

`docs.sun.com` では、Sun が提供しているオンラインマニュアルを参照することができます。マニュアルのタイトルや特定の主題などをキーワードとして、検索を行うこともできます。URL は、`http://docs.sun.com` です。

表記上の規則

このマニュアルでは、次のような字体や記号を特別な意味を持つものとして使用します。

表 **P–1** 表記上の規則

字体または記号	意味	例
`AaBbCc123`	コマンド名、ファイル名、ディレクトリ名、画面上のコンピュータ出力、コード例を示します。	`.login` ファイルを編集します。 `ls -a` を使用してすべてのファイルを表示します。 `system%`

表 **P–1** 表記上の規則　*(続き)*

字体または記号	意味	例
AaBbCc123	ユーザーが入力する文字を、画面上のコンピュータ出力と区別して示します。	`system% su` `password:`
AaBbCc123	変数を示します。実際に使用する特定の名前または値で置き換えます。	ファイルを削除するには、`rm` *filename* と入力します。
『 』	参照する書名を示します。	『コードマネージャ・ユーザーズガイド』を参照してください。
「 」	参照する章、節、ボタンやメニュー名、強調する単語を示します。	第 5 章「衝突の回避」を参照してください。 この操作ができるのは、「スーパーユーザー」だけです。
\	枠で囲まれたコード例で、テキストがページ行幅を超える場合に、継続を示します。	`sun% grep '^#define \` `  XV_VERSION_STRING'`

コード例は次のように表示されます。

- C シェル

  `machine_name% command y|n [filename]`

- C シェルのスーパーユーザー

  `machine_name# command y|n [filename]`

- Bourne シェルおよび Korn シェル

  `$ command y|n [filename]`

- Bourne シェルおよび Korn シェルのスーパーユーザー

  `# command y|n [filename]`

[ ] は省略可能な項目を示します。上記の例は、*filename* は省略してもよいことを示しています。

| は区切り文字 (セパレータ) です。この文字で分割されている引数のうち 1 つだけを指定します。

キーボードのキー名は英文で、頭文字を大文字で示します (例: `Shift` キーを押します)。ただし、キーボードによっては `Enter` キーが `Return` キーの動作をします。

ダッシュ (-) は 2 つのキーを同時に押すことを示します。たとえば、`Ctrl-D` は `Control` キーを押したまま `D` キーを押すことを意味します。

一般規則

- このマニュアルでは、「x86」という用語は、Intel 32 ビット系列のマイクロプロセッサチップ、および AMD が提供する互換マイクロプロセッサチップを意味します。

第 1 章

IPsec (概要)

IP セキュリティアーキテクチャー (IPsec) は、IPv4 および IPv6 ネットワークパケットで IP データグラムを暗号化して保護します。この保護には、機密性、データ完全性、データ認証、および部分的なシーケンスの完全性があります。部分的なシーケンスの完全性は再実行保護と呼ばれることもあります。IPsec は、IP モジュール内部で実行されます。IPsec は、インターネットアプリケーションの知識の有無に関係なく運用できます。正しく使用すれば、IPsec は、ネットワークトラフィックの保護に有効なツールとなります。

この章では、以下の内容について説明します。

IPsec とは

IPsec では、IP 内に安全なデータグラム認証と暗号化の機構を含むセキュリティアソシエーション (SA) を提供します。IPsec を呼び出すと、IPsec グローバルポリシーファイルで有効にしておいた IP データグラムにセキュリティ機構が適用されます。アプリケーションで IPsec を呼び出すと、ソケット単位レベルで IP データグラムにセキュリティ機構が適用されます。

図 1–1は、IPsec をアウトバウンドパケットで呼び出したときに、IP アドレス指定パケットが IP データグラムの一部として処理されるようすを示します。フロー図からわかるように、認証ヘッダー (AH) とカプセル化されたセキュリティペイロード (ESP) エンティティをパケットに適用できます。そのあとの節では、認証アルゴリズムと暗号化アルゴリズムとともに、これらのエンティティを適用する手順を説明します。

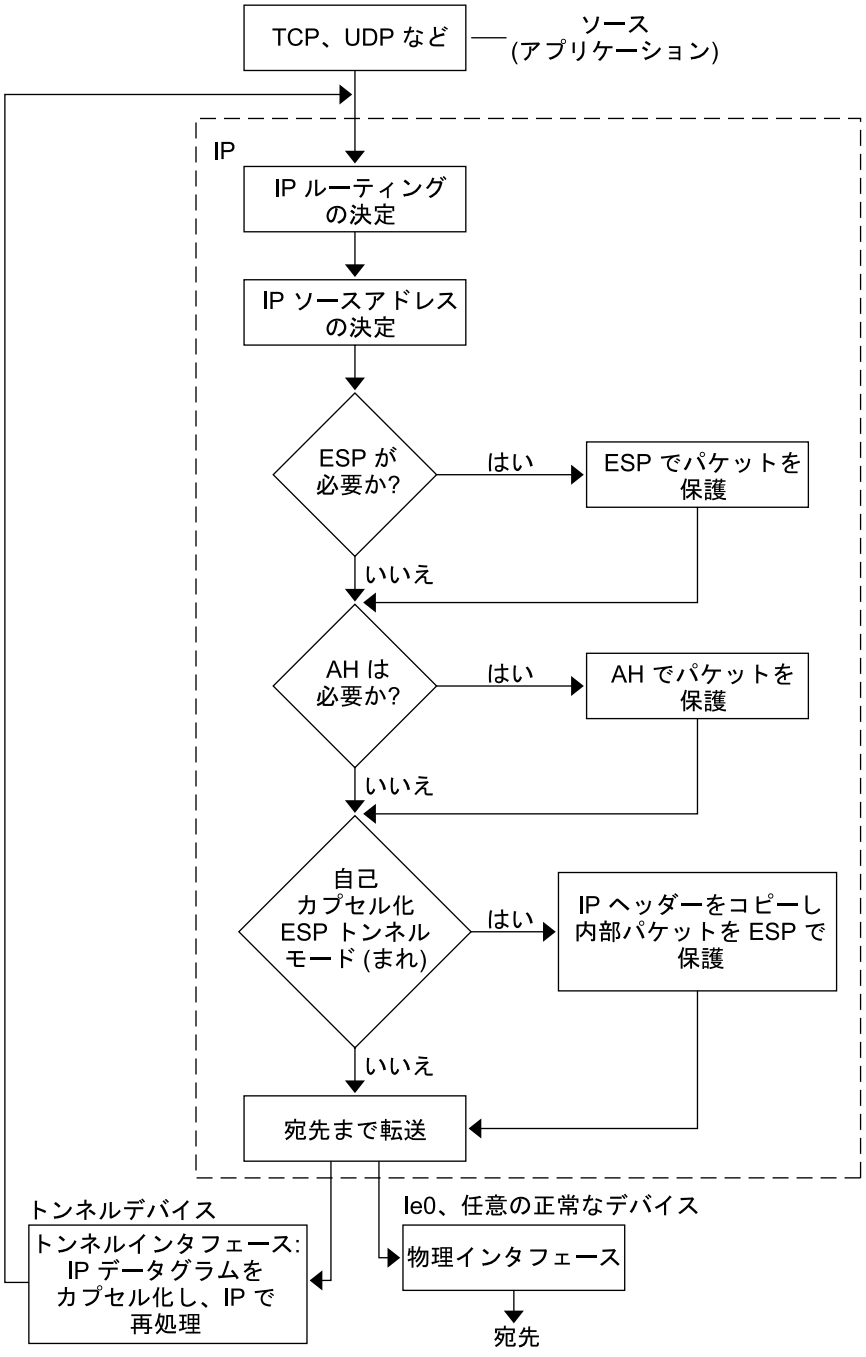


図 1–1 アウトバウンドパケットプロセスに適用された IPsec

図 1–2 は、IPsec インバウンドプロセスを示したものです。

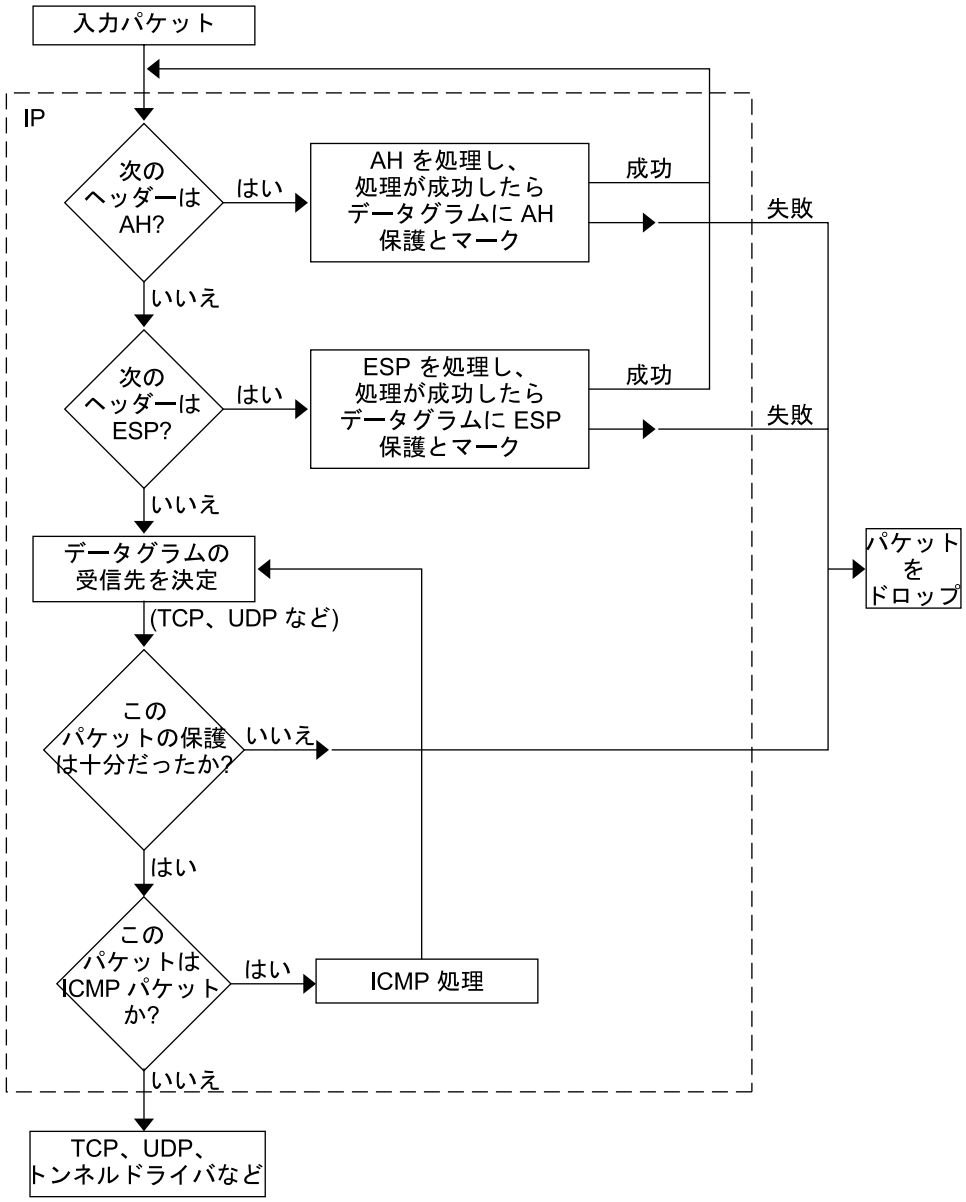


図 **1–2** インバウンドパケットプロセスに適用された IPsec

IPsec セキュリティアソシエーション

IPsec セキュリティアソシエーション (SA) では、ホスト間の通信で認識されるセキュリティ属性を指定します。一般的に、ホスト間で安全に通信するには、2 つの SA が必要です。1 つの SA は、1 方向のデータを保護します。つまり、1 つのホストかグループ (マルチキャスト) アドレスのどちらかです。ほとんどの通信は、ピアツーピアまたはクライアントとサーバー間で行われるため、両方向のトラフィックを保護するには、2 つの SA が必要です。

セキュリティプロトコル (AH または ESP) 、宛先 IP アドレス、およびセキュリティパラメータインデックス (SPI) は、IPsec SA を識別します。任意の 32 ビット値の SPI は、AH パケットまたは ESP パケットで転送されます。AH および ESP によって保護される範囲については、 `ipsecah(7P)` と `ipsecesp(7P)` のマニュアルページを参照してください。完全性チェックサム値を使用して、パケットを認証します。認証が失敗すると、パケットがドロップされます。

SA は、SA データベースに保存されます。ソケットベースの管理エンジン `pf_key(7P)` インタフェースにより、特権をもつアプリケーションでそのデータベースを管理できます。`in.iked(1M)` デーモンにより、自動キー管理が可能になります。

キー管理

SA には次の情報が含まれます。

- 暗号化や認証に必要なキー情報
- 使用できるアルゴリズム
- エンドポイントの識別情報
- システムによって使用されるその他のパラメータ

SA では、認証や暗号化のためのキー情報が必要です。認証と暗号化に必要な SA のキー情報の管理をキー管理といいます。IKE (インターネットキー交換) プロトコルにより、キー管理が自動的に行われます。また、`ipseckey(1M)` コマンドを指定して、キー管理を手動で行うこともできます。現在、IPv4 パケットの SA は自動キー管理を使用できますが、IPv6 パケットの SA は手動でキー管理を行う必要があります。

IPv4 の暗号キーを IKE で自動的に管理する方法については、51 ページの「IKE の概要」を参照してください。システム管理者が `ipseckey` コマンドを指定して、暗号キーを手動で管理する方法については、24 ページの「キーユーティリティ」を参照してください。

保護機構

IPsec にはデータ保護機構が 2 つあります。

- 認証ヘッダー (AH)
- セキュリティペイロードのカプセル化 (ESP)

どちらの機構にも独自のセキュリティアソシエーションデータベース (SADB) があります。

認証ヘッダー

認証ヘッダーは、新しい IP ヘッダーです。IP データグラムに対するデータ認証、強力な完全性、再送保護を備えています。AH では大部分の IP データグラムを保護します。送信者と受信者の間で不定的に変更されるフィールドは AH では保護できません。たとえば、IP TTL フィールドの変更は予測できないので AH では保護できません。AH は IP ヘッダーとトランスポートヘッダーの間に挿入されます。トランスポートヘッダーの種類としては、TCP、UDP、ICMP、あるいは、トンネルが使用されている場合、もう 1 つ別の IP ヘッダーがあります。トンネルの詳細については、`tun(7M)` のマニュアルページを参照してください。

認証アルゴリズムと AH モジュール

IPsec による実装では、AH は IP の先頭に自動的にプッシュされるモジュールです。`/dev/ipsecah` エントリでは、`ndd(1M)` で AH を調整します。将来の認証アルゴリズムが AH の先頭にロードできます。現在の認証アルゴリズムには、HMAC-MD5 と HMAC-SHA-1 があります。どちらの認証アルゴリズムにも、それぞれのキーサイズ属性とキーフォーマット属性が用意されています。詳細については、`authmd5h(7M)` と `autsha1(7M)` のマニュアルページを参照してください。

AH におけるセキュリティについて

再送保護を有効にしておかないと、再送時攻撃が AH をおびやかす原因になります。AH では盗聴行為には対応できません。AH で保護されたデータであっても、見ようとすれば見ることができます。

セキュリティペイロードのカプセル化

AH によるサービス同様に、ESP でもカプセル化したデータの機密が守られます。ただし、保護される対象は、データグラムのうち ESP がカプセル化した部分だけです。ESP の認証サービスはオプションです。これらのサービスでは、冗長になることなく ESP と AH を同じデータグラムで同時に使用できます。ESP は暗号対応技術を使用するため、アメリカ合衆国輸出管理法が適用されます。

ESP はデータをカプセル化するため、データグラム内でその先頭に続くデータだけを保護します。TCP パケットでは、ESP は TCP ヘッダーとそのデータだけをカプセル化します。パケットが IP 内 IP データグラムの場合、ESP は内部 IP データグラムを保護します。ソケット別ポリシーでは、自己カプセル化ができるため、必要に応じて ESP では IP オプションをカプセル化できます。認証ヘッダー (AH) と異なり、ESP では複数のデータグラム保護が可能です。1 形式だけのデータグラム保護ではデータグラムを守ることはできません。たとえば、ESP で機密だけを守っても、再送時攻撃とカットアンドペースト攻撃には無防備です。同じく、ESP で完全性だけを保護しても、その保護能力は AH より弱くなります。そのようなデータグラムは盗聴には無防備です。

アルゴリズムと ESP モジュール

IPsec ESP では、IP の先頭に自動的にプッシュされるモジュールとして ESP が実装されます。 `/dev/ipsecesp` エントリでは、`ndd(1M)` で ESP を調整します。AH で使用する認証アルゴリズムに加えて、ESP では暗号化アルゴリズムをその先頭にプッシュできます。暗号化アルゴリズムには、Data Encryption Standard (DES)、Triple-DES (3DES)、Blowfish、および AES があります。どの暗号化アルゴリズムにも、それぞれのキーサイズ属性とキーフォーマット属性があります。アメリカ合衆国輸出管理法および各国の輸入管理法の適用を受けるので、すべての暗号化アルゴリズムをアメリカ合衆国外で使用できるわけではありません。

ESP におけるセキュリティについて

認証なしで ESP を使用した場合、カットアンドペースト暗号化攻撃および再送時攻撃に対しては無防備です。機密保護なしで ESP を使用した場合、盗聴に対しては AH の場合と同じく無防備です。

認証アルゴリズムと暗号化アルゴリズム

IPsec では、認証と暗号化の 2 種類のアルゴリズムを使用します。認証アルゴリズムと DES 暗号化アルゴリズムは、Solaris インストールの主要部分になります。IPsec にサポートされるその他のアルゴリズムを使用する場合には、 Solaris Encryption Kit (データ暗号化サプリメント CD) をインストールする必要があります。Solaris Encryption Kit は CD の形で提供されています。

認証アルゴリズム

認証アルゴリズムでは、データとキーに基づいて、完全性のチェックサム値すなわちダイジェストが生成されます。認証アルゴリズムのマニュアルページに、ダイジェストとキーのサイズの説明があります。次の表は、Solaris オペレーティングシステムでサポートされる認証アルゴリズムを示します。また、IPsec ユーティリティのセキュリティオプションとして認証アルゴリズムを使用する場合のアルゴリズムの形式とそのマニュアルページも示しています。

表 **1–1** サポートされる認証アルゴリズム

アルゴリズムの名前	セキュリティオプションの形式	マニュアルページ
HMAC-MD5	md5、hmac-md5	`authmd5h(7M)`
HMAC-SHA-1	sha、sha1、hmac-sha、hmac-sha1	`authsha1(7M)`

暗号化アルゴリズム

暗号化アルゴリズムでは、キーでデータを暗号化します。暗号化アルゴリズムでは、ブロックサイズごとにデータを処理します。暗号化アルゴリズムのマニュアルページに、各アルゴリズムのブロックサイズとキーサイズの説明があります。デフォルトでは、DES–CBC アルゴリズムと3DES-CBC アルゴリズムがインストールされます。

IPsec で AES アルゴリズムと Blowfish アルゴリズムを有効にするには、Solaris Encryption Kit をインストールする必要があります。このキットは、Solaris 9 インストールボックスには含まれていない別の CD から入手できます。『 *Solaris 9 Encryption Kit Installation Guide*』に、Solaris Encryption Kit のインストール方法が説明されています。

次の表に、Solaris オペレーティングシステムでサポートされる暗号化アルゴリズムを示します。また、IPsec ユーティリティのセキュリティオプションとして暗号化アルゴリズムを使用する場合のアルゴリズムの形式、そのマニュアルページ、およびそのアルゴリズムが含まれるパッケージも示しています。

表 **1–2** サポートされる暗号化アルゴリズム

アルゴリズムの名前	セキュリティオプションの形式	マニュアルページ	パッケージ
DES-CBC	des, des-cbc	`encrdes(7M)`	SUNWcsr、SUNWcarx.u
3DES–CBC または Triple-DES	3des、3des-cbc	`encr3des(7M)`	SUNWcsr、SUNWcarx.u
Blowfish	blowfish, blowfish-cbc	`encrbfsh(7M)`	SUNWcryr、SUNWcryrx
AES-CBC	aes, aes-cbc	`encraes(7M)`	SUNWcryr、SUNWcryrx

保護ポリシー機構と実施機構

IPsec では、保護ポリシー機構と実施機構を分けています。IPsec ポリシーは、次の範囲で適用できます。

- システム規模レベル
- ソケット単位レベル

`ipsecconf(1M)` コマンドは、システム規模ポリシーの設定に使用します。

IPsec は、システム規模ポリシーを入力データグラムと出力データグラムに適用します。システムで認識されるデータがあるため、出力データグラムにはその他の規則も適用できます。インバウンドデータグラムの処理は、受理されるか拒絶されるかのどちらかです。インバウンドデータグラムの受理か拒絶を決定する基準はいくつかありますが、場合によってはその基準が重複したり競合することがあります。競合の解決に当たっては、どの規則の構文解析を最初に行うかが決定されます。ただし、ポリシーエントリでトラフィックが他のすべてのポリシーを省略するように指定されている場合は、自動的に受理されます。アウトバウンドデータグラムは、保護付きまたは保護なしで送信されます。保護が適用されると、特定アルゴリズムか汎用アルゴリズムのどちらかになります。

データグラムを保護する通常のポリシーを省略することもできます。それには、システム規模ポリシーに例外を指定するか、ソケット単位ポリシーで省略を要求します。イントラシステム内トラフィックの場合、ポリシーは実施されますが、実際のセキュリティ機構は適用されません。その代わりに、イントラシステム内パケットのアウトバウンドポリシーが、セキュリティ機構の適用されたインバウンドパケットになります。

トランスポートモードとトンネルモード

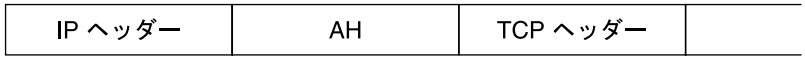
IP ヘッダーの後に、ESP または AH を呼び出してデータグラムを保護するときに、トランスポートモードを使用します。たとえば、パケットが次のヘッダーで始まる場合です。

IP ヘッダー	TCP ヘッダー	

トランスポートモードでは、ESP は次のようにデータを保護します。

暗号化部分

トランスポートモードでは、AH は次のようにデータを保護します。



AH は実際には、データグラムに出現する前のデータも保護します。その結果、 AH による保護は、トランスポートモードでも、IP ヘッダーの一部をカバーします。

データグラム全体が IPsec ヘッダーの保護下にあるとき、IPsec では、トンネルモードでデータグラムを保護しています。AH はその前にある IP ヘッダーの大部分を保護するため、トンネルモードは通常、ESP だけで実行します。先の例のデータグラムは、トンネルモードでは次のように保護されます。

暗号化部分

トンネルモードでは、内部ヘッダーは保護されますが、外部 IP ヘッダーは保護されません。外部 IP ヘッダーのソースアドレスと宛先アドレスが、内部 IP ヘッダーのものと異なることがよくあります。それでも、IPsec を認識するネットワークプログラムで ESP の自己カプセル化を使用すれば、内部と外部の IP ヘッダーを一致させることができます。ESP の自己カプセル化により、IP ヘッダーオプションが保護されます。

IPsec の Solaris 実装は基本的にトランスポートモード IPsec 実装です。トンネルモードはトランスポートモードの特殊ケースとして実装されます。そのため、IP 内 IP トンネルを特殊なトランスポートプロバイダとして処理します。`ifconfig(1M)` 設定オプションを使用してトンネルを設定する場合、オプションは、ソケットのプログラミングでソケットごとの IPsec を使用可能にするときに使用するオプションとほぼ同じです。また、トンネルモードは、ソケットごとの IPsec で使用可能にできます。ソケットごとのトンネルモードでは、内部パケットの IP ヘッダーのアドレスが外部パケットの IP ヘッダーのアドレスと同じになります。ソケットごとのポリシーの詳細については、`ipsec(7P)` のマニュアルページを参照してください。

信頼性の高いトンネル

設定したトンネルは、ポイントツーポイントインタフェースです。このトンネルで、IP パケットを IP パケット内にカプセル化できます。トンネルの設定には、トンネルソースとトンネル宛先が必要です。詳細については、`tun(7M)` のマニュアルページと、『*Solaris* のシステム管理 *(IP* サービス*)*』の「IPv6 の Solaris トンネルインターフェース」を参照してください。

トンネルでは、IP との見かけ上の物理的インタフェースが作成されます。この物理的リンクの完全性は、基本になるセキュリティプロトコルによって異なります。セキュリティアソシエーションを確実に行えば、信頼性の高いトンネルになります。トンネルのデータパケットのソースはトンネル宛先で指定したピアでなければなりません。この信頼関係があるかぎり、インタフェース別 IP 送信を利用して仮想プライベートネットワークを作成できます。

仮想プライベートネットワーク

IPsec を使用して、VPN (仮想プライベートネットワーク) を構築できます。 IPsec を使用するためには、インターネットインフラストラクチャを使用してイントラネットを作成します。たとえば、それぞれのネットワークとともに独立したオフィスを持つ組織があって、オフィス間が VPN テクノロジで接続されている場合、IPsec を利用すれば、2 つのオフィス間でトラフィックを安全にやりとりできます。

図 1–3 は、ネットワークシステムに配置した IPsec で、2 つのオフィスがインターネットを利用して VPN を形成する方法を示します。

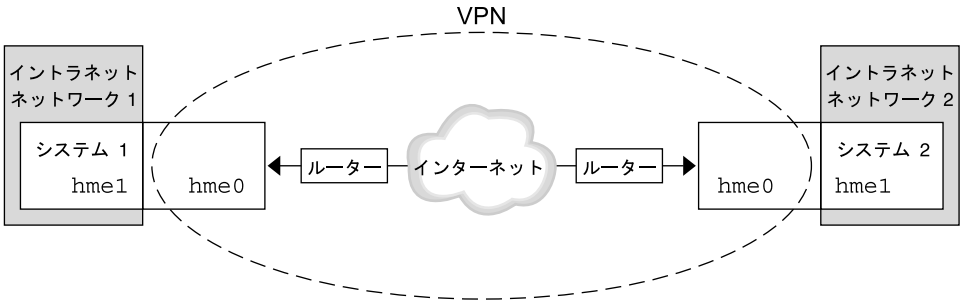


図 **1–3** 仮想プライベートネットワーク

セットアップ手順については、37 ページの「仮想プライベートネットワークを構築する方法」を参照してください。

IPsec ユーティリティおよび IPsec ファイル

この節では、IPsec を初期化する構成ファイルについて説明します。また、ネットワーク内で IPsec の管理を行うためのさまざまなコマンドについても説明します。ネットワーク内で IPsec を実装する方法については、29 ページの「IPsec の実装 (作業マップ)」を参照してください。

表 **1–3** 選択される IPsec ファイルと IPsec コマンドのリスト

IPsec ファイルまたは **IPsec** コマンド	説明
`/etc/inet/ipsecinit.conf` ファイル	IPsec ポリシーファイル。このファイルがある場合、IPsec はブート時に起動する
`ipsecconf` コマンド	IPsec ポリシーコマンド。起動スクリプトは、`ipsecconf` を使って `/etc/inet/ipsecinit.conf` ファイルを読み込み、IPsec を起動する。現在の IPsec ポリシーの表示および変更や、テストを行うときに役立つ
`PF_KEY` ソケットインタフェース	SA データベースのインタフェース。手動キー管理および自動キー管理を処理する
`ipseckey` コマンド	IPsec SA 保守およびキーコマンド。`ipseckey` は、`PF_KEY` インタフェースに対するコマンド行フロントエンド。`ipseckey` では、セキュリティアソシエーションの作成、削除、または変更ができる
`/etc/inet/secret/ipseckeys` ファイル	IPsec SA のキー。`ipsecinit.conf` ファイルがある場合、`ipseckeys` ファイルはブート時に自動的に読み込まれる
`/etc/inet/ike/config` ファイル	IKE 構成およびポリシーファイル。このファイルがある場合、IKE デーモン `in.iked(1M)` により、`/etc/inet/ike/config` ファイルが開始およびロードされる。54 ページの「IKE ユーティリティおよび IKE ファイル」を参照

IPsec ポリシーコマンド

`ipsecconf(1M)` コマンドを使用して、ホストの IPsec ポリシーを構成します。このコマンドを実行してポリシーを構成すると、`ipsecpolicy.conf` という名前の一時ファイルが作成されます。このファイルには、`ipsecconf` コマンドによってカーネルに設定された IPsec ポリシーエントリが格納されます。システムは、カーネル内 IPsec ポリシーエントリを使用して、すべてのアウトバウンドおよびインバウンド IP

データグラムがポリシーに沿っているか検査します。転送されたデータグラムは、このコマンドで追加されたポリシー検査の対象外になります。転送されたパケットを保護する方法については、ifconfig(1M) と tun(7M) のマニュアルページを参照してください。

ipsecconf コマンドを呼び出すには、スーパーユーザーになるか、同等の役割を引き受ける必要があります。このコマンドは、両方向のトラフィックを保護するエントリ、および 1 方向のみのトラフィックを保護するエントリを受け入れます。

ローカルアドレスとリモートアドレスというパターンのポリシーエントリは、1 つのポリシーエントリで両方向のトラフィックを保護します。たとえば、指定されたホストに対して方向が指定されていない場合、laddr host1 と raddr host2 というパターンをもつエントリは、両方向のトラフィックを保護します。したがって、各ホストにポリシーエントリを 1 つだけ設定すれば済みます。ソースアドレスから宛先アドレスへというパターンのポリシーエントリは、1 方向のみのトラフィックを保護します。たとえば、saddr host1 daddr host2 というパターンのポリシーエントリは、インバウンドかアウトバウンドのどちらかのトラフィックのみを保護します。両方向ともは保護しません。したがって、両方向のトラフィックを保護するには、saddr host2 daddr host1 のようなエントリも ipsecconf コマンドに渡す必要があります。

引数を指定しないで ipsecconf コマンドを実行すると、システムに構成されているポリシーを確認できます。各エントリが、インデックスとその後に番号が付いて表示されます。-d オプションでインデックスを指定すると、システム内の指定されたポリシーが削除されます。このコマンドで表示されるエントリの順序はエントリが追加された順であり、必ずしもトラフィックを照合する順序ではありません。トラフィックの照合が行われる順序を確認するには、-l オプションを使用します。

ipsecpolicy.conf ファイルは、システムのシャットダウン時に削除されます。マシンのブート時に IPsec ポリシーを起動させるには、マシンのブート時に inetinit スクリプトによって読み込まれる IPsec ポリシーファイル /etc/inet/ipsecinit.conf を作成する必要があります。

IPsec ポリシーファイル

Solaris オペレーティングシステムを起動したときに IPsec セキュリティポリシーを呼び出すには、特定の IPsec ポリシーエントリを利用して、IPsec を初期化する構成ファイルを作成します。ファイルの名前は、/etc/inet/ipsecinit.conf とします。ポリシーエントリとその形式の詳細については、ipsecconf(1M) のマニュアルページを参照してください。ポリシーの構成後、ipsecconf コマンドを使用してポリシーを一時的に削除したり、既存の構成を表示したりすることができます。

例 – ipsecinit.conf ファイル

Solaris ソフトウェアには、IPsec ポリシーファイルの例が含まれています。このサンプルファイルの名前は ipsecinit.sample です。このファイルをテンプレートとして独自の ipsecinit.conf ファイルを作成することができます。
ipsecinit.sample ファイルには、次のエントリが含まれています。

```
#
#ident    "@(#)ipsecinit.sample    1.6  01/10/18 SMI"
#
# Copyright (c) 1999,2001 by Sun Microsystems, Inc.
# All rights reserved.
#
# This file should be copied to /etc/inet/ipsecinit.conf to enable IPsec
# systemwide policy (and as a side-effect, load IPsec kernel modules).
# Even if this file has no entries, IPsec will be loaded if
# /etc/inet/ipsecinit.conf exists.
#
# Add entries to protect the traffic using IPsec. The entries in this
# file are currently configured using ipsecconf from inetinit script
# after /usr is mounted.
#
# For example,
#
#     {rport 23} ipsec {encr_algs des encr_auth_algs md5}
#
# Or, in the older (but still usable) syntax
#
#    {dport 23} apply {encr_algs des encr_auth_algs md5 sa shared}
#    {sport 23} permit {encr_algs des encr_auth_algs md5}
#
# will protect the telnet traffic originating from the host with ESP using
# DES and MD5. Also:
#
#     {raddr 10.5.5.0/24} ipsec {auth_algs any}
#
# Or, in the older (but still usable) syntax
#
#    {daddr 10.5.5.0/24} apply {auth_algs any sa shared}
#    {saddr 10.5.5.0/24} permit {auth_algs any}
#
# will protect traffic to or from the 10.5.5.0 subnet with AH
# using any available algorithm.
#
#
# To do basic filtering, a drop rule may be used. For example:
#
#    {lport 23 dir in} drop {}
#    {lport 23 dir out} drop {}
#
# will disallow any remote system from telnetting in.
#
#
# WARNING:    This file is read before default routes are established, and
#         before any naming services have been started. The
```

```
#       ipsecconf(1M) command attempts to resolve names, but it will
#       fail unless the machine uses files, or DNS and the DNS server
#       is reachable via routing information before ipsecconf(1M)
#       invocation.  (that is, the DNS server is on-subnet, or DHCP
#       has loaded up the default router already.)
#
#       It is suggested that for this file, use hostnames only if
#       they are in /etc/hosts, or use numeric IP addresses.
#
#       If DNS gets used, the DNS server is implicitly trusted, which
#       could lead to compromise of this machine if the DNS server
#       has been compromised.
#
```

ipsecinit.conf と ipsecconf のセキュリティについて

たとえば、/etc/inet/ipsecinit.conf ファイルを、NFS マウントファイルシステムから送信すると、ファイル内のデータが不正に変更される可能性があります。また、設定ポリシーも変更される可能性があります。そのため、ipsecinit.conf ファイルのコピーをネットワークで送信しないでください。

TCP ソケットまたは UDP ソケットに対して、 connect(3SOCKET) または accept(3SOCKET) を実行した場合、これらのソケットのポリシーを変更することはできません。ポリシーの変更ができないソケットを、ラッチされたソケットと呼びます。新しいポリシーエントリを追加しても、ラッチされたソケットは変更されません。

ポリシーは通信を開始する前にセットアップしてください。新しいポリシーエントリを追加すると既存の接続が影響を受けることがあるためです。同じ理由から、通信の途中ではポリシーを変更しないでください。

ネーミングシステムを保護してください。次の 2 つの条件に該当する場合、そのホスト名は信頼できません。

- ソースアドレスが、ネットワークを介して参照できるホストである
- ネーミングシステムの信頼性に問題がある

セキュリティの弱点の多くは、実際のツールではなく、ツールの使用方法にあります。ipsecconf コマンドを使用するときは注意が必要です。安全に操作するため、コンソールなどの、ハード接続の TTY を使用してください。

IPsec のセキュリティアソシエーションデータベース

IPsec セキュリティサービスのキー情報は、セキュリティアソシエーションデータベース (SADB) に保存されます。セキュリティアソシエーションは、インバウンドパケットとアウトバウンドパケットを保護します。ユーザープロセス (場合によっては

マルチ連携プロセス) では、特殊なソケットからのメッセージを送信することで SADB を管理します。SADB を保守するこの方法は、 route(7P) のマニュアルページで説明している方法に類似しています。SADB にアクセスできるのは、スーパーユーザーか、同等の役割を引き受けた人だけです。

オペレーティングシステムは、外部イベントに対する応答としてメッセージを自動的に発信する場合があります。たとえば、システムがアウトバウンドデータグラムに対する新しい SA を要求したり、既存の SA の期限切れを報告する場合です。先に説明したソケットコールを使用して、SADB 制御メッセージを伝えるためのチャンネルを開いてください。システムごとに複数のキーソケットを開くことができます。

メッセージには、小さいベースヘッダーがあり、そのあとにいくつかの拡張メッセージが続きます。拡張メッセージの数はゼロの場合もあれば、1 以上の場合もあります。メッセージの中には、追加データが必要なものもあります。ベースメッセージと拡張メッセージのいずれも 8 バイト配列である必要があります。たとえば GET メッセージの場合、ベースヘッダー、SA 拡張メッセージ、ADDRESS_DST 拡張メッセージが必要です。詳細については、pf_key(7P) を参照してください。

キーユーティリティ

IKE プロトコルは、IPv4 アドレスの自動キーユーティリティです。IKE の設定方法については、第 4 章 を参照してください。 手動でキーを操作するユーティリティには、ipseckey(1M) コマンドがあります。

ipseckey コマンドを使用して、ipsecah(7P) と ipsecesp(7P) の保護機構で SA データベースを手動で操作できます。また、自動キー管理が無効な場合に、通信パーティ間の SA をセットアップするときも、ipseckey コマンドを使用します。例としては、IPv6 アドレスを持つ通信パーティ間が挙げられます。

ipseckey コマンドには少数の一般オプションしかありませんが、多くのコマンド言語をサポートしています。マニュアルキー操作に固有のプログラムインタフェースで要求を配信するように指定することもできます。詳細については、pf_key(7P) のマニュアルページを参照してください。引数なしで ipseckey を呼び出すと、対話モードになり、エントリを入力できるプロンプトが表示されます。コマンドによっては、明示的なセキュリティアソシエーション (SA) タイプが必要ですが、それ以外は、ユーザーが SA を指定すれば、すべての SA タイプで動作します。

ipseckey におけるセキュリティについて

ipseckey コマンドを使用すると、特権ユーザーは微妙な暗号キー情報を入力できます。場合によっては、不正にこの情報にアクセスして IPsec トラフィックのセキュリティを損なうことも可能です。キー情報を扱う場合および ipseckey コマンドを使用する場合には、次のことに注意してください。

1. キー情報を更新しているかどうか。定期的にキーを更新することが、セキュリティの基本作業となります。キーを更新することで、アルゴリズムとキーの脆弱性が暴かれないように保護し、公開されたキーの侵害を制限します。

2. TTY がネットワークに接続されているか。ipseckey コマンドは対話モードで実行されているか。
   - 対話モードの場合には、キー情報のセキュリティは、TTY のトラフィックに対応するネットワークパスのセキュリティになります。clear-text telnet や rlogin セッションでは、ipseckey コマンドを使用しないでください。
   - ローカルウィンドウでも、ウィンドウを読み取ることのできる隠密プログラムからの攻撃には無防備です。
3. ファイルはネットワークを介してアクセスされているか。ファイルは外部から読み取り可能か。-f オプションを使用しているか。
   - ネットワークマウントファイルの読み取り時に、不正に読み取ることができます。外部から読み取れるファイルにキー情報を保存して使用しないでください。
   - ネーミングシステムを保護してください。次の 2 つの条件に該当する場合、そのホスト名は信頼できません。
     - ソースアドレスが、ネットワークを介して参照できるホストである
     - ネーミングシステムの信頼性に問題がある

セキュリティの弱点の多くは、実際のツールではなく、ツールの使用方法にあります。ipseckey コマンドを使用するときには注意が必要です。安全に操作するため、コンソールなどの、ハード接続の TTY を使用してください。

その他のユーティリティに対する IPsec 拡張機能

ifconfig コマンドには、トンネルインタフェースで IPsec ポリシーを管理するオプションがあります。また、snoop コマンドを使用して AH ヘッダーと ESP ヘッダーを構文解析できます。

ifconfig コマンド

IPsec をサポートするため、ifconfig(1M) に次のオプションが追加されました。

- auth_algs
- encr_auth_algs
- encr_algs

auth_algs

このオプションを設定すると、指定した認証アルゴリズムで、トンネルにIPsec AH を使用できます。auth_algs オプションの書式は次のとおりです。

auth_algs *authentication_algorithm*

アルゴリズムには、番号またはアルゴリズム名を指定できます。特定のアルゴリズムが指定されないようにするパラメータ *any* も使用できます。IPsec トンネル属性は、すべて同じコマンド行に指定します。トンネルセキュリティを無効にするには、次のオプションを指定します。

```
auth_algs none
```

サポートされる認証アルゴリズムとその詳細を説明したマニュアルページのリストについては、表 1–1 を参照してください。

encr_auth_algs

このオプションを設定すると、指定した認証アルゴリズムで、トンネルにIPsec ESP を使用できます。`encr_auth_algs` オプションの書式は次のとおりです。

`encr_auth_algs` *authentication_algorithm*

アルゴリズムには、番号またはアルゴリズム名を指定できます。特定のアルゴリズムが指定されないようにするパラメータ *any* も使用できます。ESP 暗号化アルゴリズムを指定し、認証アルゴリズムを指定しない場合、ESP 認証アルゴリズム値はデフォルトのパラメータ *any* になります。

サポートされる認証アルゴリズムとその詳細を説明したマニュアルページのリストについては、表 1–1 を参照してください。

encr_algs

このオプションでは、暗号化アルゴリズムを指定したトンネルで IPsec ESP を使用できます。オプションの書式は次のとおりです。

`encr_algs` *encryption_algorithm*

アルゴリズムには、番号またはアルゴリズム名を指定できます。IPsec トンネル属性は、すべて同じコマンド行に指定します。トンネルセキュリティを無効にするには、次のオプションを指定します。

```
encr_algs none
```

ESP 認証アルゴリズムを指定し、暗号化アルゴリズムを指定しない場合、ESP 暗号化アルゴリズム値はデフォルトのパラメータ *null* になります。

サポートされる暗号化アルゴリズムとその詳細を説明したマニュアルページのリストについては、`ipsecesp(7P)` のマニュアルページまたは表 1–2 を参照してください。

snoop コマンド

snoop コマンドでも、AH ヘッダーと ESP ヘッダーを構文解析できるようになりました。ESP はそのデータを暗号化するので、snoop は ESP で暗号化されて保護されたヘッダーを読み取ることができませんが、AH ではデータは暗号化されないので、snoop でトラフィックを確認できます。パケットに AH が使用されている場合、snoop -V オプションで表示できます。詳細については、snoop(1M) のマニュアルページを参照してください。

保護されたパケットに対する snoop の冗長出力例については、47 ページの「パケットが保護されていることを確認する方法」を参照してください。

第 2 章

IPsec の管理 (手順)

この章では、ネットワークに IPsec を実装する手順について説明します。これらの手順を表 2–1 に示します。

IPsec の概要については、第 1 章を参照してください。`ipsecconf(1M)`、`ipseckey(1M)`、`ifconfig(1M)` の各マニュアルページの「EXAMPLES」セクションにも、有益な手順が記載されています。

IPsec の実装 (作業マップ)

表 **2–1** IPsec の実装 (作業マップ)

作業	説明	参照先
IPv6 システム間のトラフィックの保護	`/etc/inet/ipnodes` ファイルに対するアドレスの追加、`/etc/inet/ipsecinit.conf` ファイルに対する IPsec ポリシーの入力、`ipseckey` コマンドを使用した手動によるキーの追加、 `ipsecinit.conf` ファイルの呼び出し	30 ページの「2 つのシステム間のトラフィックを保護する方法」
IPsec ポリシーによる Web サーバーの保護	`ipsecinit.conf` ファイルに対するさまざまなポートの異なるセキュリティ要件の入力とそのファイルの呼び出しによる、保護トラフィックだけの有効化	35 ページの「Web サーバーを保護する方法」

表 **2–1** IPsec の実装 (作業マップ)　*(続き)*

作業	説明	参照先
仮想プライベートネットワークのセットアップ	IP 送信のオフ、IP の厳密宛先マルチホーム、大半のネットワークサービスとインターネットサービスの無効化、セキュリティアソシエーションの追加、IPsec ポリシーの設定、保護トンネルの設定、IP 送信のオン、デフォルトルートの設定、ルーティングプロトコルの実行	37 ページの「仮想プライベートネットワークを構築する方法」
乱数の生成	Solaris の `/dev/random` デバイスから数値を生成する	44 ページの「乱数を生成する方法」
手動によるセキュリティアソシエーションの作成	追加のインタフェースを保護する際に、`ipseckey` コマンドを使ってセキュリティアソシエーションを作成する	44 ページの「IPsec セキュリティアソシエーションを手動で作成する方法」
現在のセキュリティアソシエーションの変更	新しいキー情報を入力する前に、現在のセキュリティアソシエーションをフラッシュする	47 ページの「例 — IPsec セキュリティアソシエーションの置き換え」
IPsec がパケットを保護しているかどうかの検査	スヌープ出力を調べ、IP データグラムがどのように保護されているかを示すヘッダーをチェックする	47 ページの「パケットが保護されていることを確認する方法」

IPsec 作業

この節では、2 つのシステム間のトラフィックを保護し、Web サーバーを保護し、仮想プライベートネットワークをセットアップするための手順について説明します。システム名 `enigma` と `partym` は一例として説明しているだけです。よって、`enigma` と `partym` を各自使用しているシステムの名前に置き換えてください。

役割を使って IPsec を管理する方法については、『*Solaris* のシステム管理 *(*セキュリティサービス*)*』の「役割によるアクセス制御 (手順)」を参照してください。

▼ 2 つのシステム間のトラフィックを保護する方法

この手順では、次の設定がすでになされているものとします。

- 各システムには、2 つのアドレス (IPv4 アドレスと IPv6 アドレス) がある
- 各システムは、128 ビットのキーを必要とする MD5 アルゴリズムを使って AH 保護を呼び出す
- 各システムは、192 ビットのキーを必要とする 3DES アルゴリズムを使って ESP 保護を呼び出す

- IPsec は、共有セキュリティアソシエーションを使用する

  共有セキュリティアソシエーションでは、2 つのシステムを保護するのに 1 組だけの SA を必要とします。

1. システムコンソールから、スーパーユーザーになるか、同等の役割を引き受けます。

注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

---

2. システムごとに、他のシステムのアドレスとホスト名を **/etc/inet/ipnodes** ファイルに追加します。次のように、**1** つのシステムのエントリは連続してそのファイルに入力します。

   a. **partym** という名前のシステムでは、**ipnodes** ファイルに次のように入力します。

   ```
   # Secure communication with enigma
   192.168.116.16 enigma
   fec0::10:20ff:fea0:21f6 enigma
   ```

   b. **enigma** という名前のシステムでは、**ipnodes** ファイルに次のように入力します。

   ```
   # Secure communication with partym
   192.168.13.213  partym
   fec0::9:a00:20ff:fe7b:b373 partym
   ```

   システムの名前は、一例として使用しているだけです。実際にシステム間のトラフィックを保護する場合には、各自のシステムの名前を使用してください。

   これで、起動スクリプトでは、存在しないネーミングサービスに依存することなくシステム名を使用できます。

3. システムごとに、**/etc/inet/ipsecinit.conf** ファイルを作成します。

   /etc/inet/ipsecinit.sample ファイルを /etc/inet/ipsecinit.conf にコピーすることができます。

4. **ipsecinit.conf** ファイルに **IPsec** ポリシーエントリを追加します。

   a. **enigma** で、次のポリシーを **ipsecinit.conf** ファイルに追加します。

   ```
   {laddr enigma raddr partym} ipsec {auth_algs any encr_algs any sa shared}
   ```

   b. **partym** で、同じポリシーを **ipsecinit.conf** ファイルに追加します。

   ```
   {laddr partym raddr enigma} ipsec {auth_algs any encr_algs any sa shared}
   ```

   IPsec ポリシーエントリの構文については、ipsecconf(1M) のマニュアルページを参照してください。

5. システムごとに、**2** つのシステム間の **IPsec** セキュリティアソシエーションの組を追加します。

システムごとに、読み取り専用の `/etc/inet/secret/ipseckeys` ファイルを編集します。読み取り専用ファイルのアクセス権は 400 です。`ipseckeys` ファイル で、ESP 保護と AH 保護のセキュリティアソシエーションの組は、次の形をとります。

```
add protocol spi random-hex-string dst local-system \
   encr_alg protocol-algorithm \
   encrkey random-hex-string-of-algorithm-specified-length

add protocol spi random-hex-string dst local-system \
   auth_alg protocol-algorithm \
   authkey random-hex-string-of-algorithm-specified-length

add protocol spi random-hex-string dst remote-system \
   encr_alg protocol-algorithm \
   encrkey random-hex-string-of-algorithm-specified-length

add protocol spi random-hex-string dst remote-system \
   auth_alg protocol-algorithm \
   authkey random-hex-string-of-algorithm-specified-length
```

protocol	`esp` または `ah`。`ah` プロトコルでは、`auth_alg` と `authkey` 引数を使用する。`esp` プロトコルでは、`encr_alg` と `encrkey` 引数も使用する。`esp` では、さらに、`ah` で使用する `auth_alg` と `authkey` 引数も使用する
random-hex-string	16 進数形式の最大 8 桁の乱数。SPI が受け取れる以上の桁数を入力すると、超過部分は無視される。SPI が受け取れるより少ない桁数を入力すると、パディングが行われる
local-system	ローカルシステム名
remote-system	リモートシステム名
protocol-algorithm	ESP または AH のアルゴリズム。それぞれのアルゴリズムには、特定の長さのキーが必要 認証アルゴリズムには MD5 と SHA がある。暗号化アルゴリズムには 3DES と AES がある
random-hex-string-of-algorithm-specified-length	アルゴリズムによって必要とされる長さをもつ 16 進数の乱数。たとえば、MD5 アルゴリズムでは、128 ビットキーのため 32 桁の乱数が必要。3DES アルゴリズムでは、192 ビットキーのため 48 桁の乱数が必要

a. 乱数を生成します。

アウトバウンドトラフィックとインバウンドトラフィックにはそれぞれ、3 種類の乱数が必要です。したがって、システムごとに次の乱数が必要になります。

- spi キーワードの値として、2 つの 16 進数の乱数。1 つの乱数はアウトバウンドトラフィック用、もう 1 つの乱数はインバウンドトラフィック用。それぞれの乱数の最大桁数は 8 桁
- AH の MD5 アルゴリズム用として、2 つの 16 進数の乱数。各乱数は 32 桁でなければならない。1 つの乱数は dst enigma 用、もう 1 つの乱数は dst partym 用
- ESP の 3DES アルゴリズム用として、2 つの 16 進数の乱数。192 ビットキーの場合、各乱数は48 桁でなければならない。1 つの乱数は dst enigma 用、もう 1 つの乱数は dst partym 用

乱数発生関数がすでにある場合は、それを使用してください。ない場合は、od コマンドを使用できます。この手順については、44 ページの「乱数を生成する方法」を参照してください。

b. たとえば、**enigma** では、**ipseckeys** ファイルの内容は次のようになります。

```
# for inbound packets
add esp spi c83f5a4b dst enigma encr_alg 3DES \
      encrkey b6a8f89213a796bde03c601029861eae91c65783368165a6
#
add ah spi 2f526ae6 dst enigma auth_alg MD5
      authkey 305ec56369ca62c2ae804690c5713e18

# for outbound packets
add esp spi 0cecc4b2 dst partym  encr_alg 3DES \
      encrkey 802e89f9f9b929ea2b615641b71ac7034a540d3cbeeaf6a9
#
add ah spi a75bbe5f dst partym auth_alg MD5 \
      authkey 2ae8b94967e6b9b0dd16e6d4b7ea7278
```

c. **partym** の **ipseckeys** には、同一のキーを使用します。コメントが異なっていますが、これは、**dst enigma** が **enigma** ではインバウンド、**partym** ではアウトバウンドであるためです。

```
# for outbound packets
add esp spi c83f5a4b dst enigma encr_alg 3DES \
        encrkey b6a8f89213a796bde03c601029861eae91c65783368165a6
#
add ah  spi 2f526ae6 dst enigma auth_alg MD5
        authkey 305ec56369ca62c2ae804690c5713e18

# for inbound packets
add esp spi 0cecc4b2 dst partym  encr_alg 3DES \
        encrkey 802e89f9f9b929ea2b615641b71ac7034a540d3cbeeaf6a9
#
add ah  spi a75bbe5f dst partym auth_alg MD5 \
        authkey 2ae8b94967e6b9b0dd16e6d4b7ea7278
```

注 – これらのキーと SPI は、セキュリティアソシエーションごとに変更できます。セキュリティアソシエーションごとに、異なるキーと異なる SPI を割り当てるべきです。

6. リブートします。

```
# /usr/sbin/reboot
```

7. パケットが保護されているかどうかを確認するには、**47** ページの「パケットが保護されていることを確認する方法」を参照してください。

例 — リブートなしでの IPv6 アドレス間のトラフィックの保護

次の例では、IPv6 アドレスを持つシステム間の保護トラフィックをテストする方法について説明します。実際の稼動環境では、`ipsecconf` コマンドを実行するよりもリブートする方が安全です。

1. 30 ページの「2 つのシステム間のトラフィックを保護する方法」の手順 5 までを実行します。
2. リブートする代わりに、`ipseckey` コマンドを使ってセキュリティアソシエーションをデータベースに追加します。

```
# ipseckey -f /etc/inet/secret/ipseckeys
```

3. 次のように、`ipsecconf` コマンドを使用して IPsec ポリシーを有効にします。

```
# ipsecconf -a /etc/inet/ipsecinit.conf
```

注 – このコマンドの実行時には警告を読んでください。ソケットがすでに使用中 (ラッチされた) の場合には、システムのセキュリティが低下します。

例 — IPv4 アドレス間のトラフィックの保護

次の例では、IPv4 アドレスを持つシステム間のトラフィックを保護する方法について説明します。この例では、自動キー管理 (IKE) を使用してセキュリティアソシエーションを作成します。IKE は、管理者が介在する必要が少なく、大量のトラフィックを容易に保護するようにスケーリングします。

1. 前の手順の手順 2 の`/etc/inet/ipnodes` ファイルを `/etc/hosts` ファイルに置き換えます。

   `partym` システムで、`enigma` を `/etc/hosts` ファイルに追加します。

```
# echo "192.168.116.16 enigma">> /etc/hosts
```

enigma システムで、partym を /etc/hosts ファイルに追加します。

```
# echo "192.168.13.213 partym">> /etc/hosts
```

2. ipsecinit.conf ファイルを編集して IPsec ポリシーエントリを追加します (手順 4 を参照)。
3. キーは、次のどちらかの方法で作成できます。
   - IKE を設定してキーを自動的に生成する。IKE はキーを自動的に更新します。IKE を設定するには、表 4–1 の設定手順のどれかに従ってください。IKE 設定ファイルの構文については、ike.config(4) のマニュアルページを参照してください。

     キーの生成や保守を手動で行う必要が特にない場合は、IKE を設定すべきです。
   - IKE デーモン in.iked を起動しない場合は、キーを手動で作成できます。これについては、30 ページの「2 つのシステム間のトラフィックを保護する方法」の手順 5 を参照してください。
4. リブートします。

   リブートせずにトラフィックを保護する場合は、ipseckey コマンドと ipsecconf コマンドを使用します。

```
# ipseckey -f /etc/inet/secret/ipseckeys
# ipsecconf -a /etc/inet/ipsecinit.conf
```

注 – このコマンドの実行時には警告を読んでください。ソケットがすでに使用中 (ラッチされた) の場合には、システムのセキュリティが低下します。

---

5. パケットが保護されているかどうかを確認するには、47 ページの「パケットが保護されていることを確認する方法」を参照してください。

▼ Web サーバーを保護する方法

セキュリティ保護された Web サーバーでは、Web クライアントであれば Web サービスと通信できます。セキュリティ保護された Web サーバーでは、Web トラフィック以外のトラフィックは、セキュリティ検査を通る必要があります。次の手順には、Web トラフィックの検査省略手順が含まれています。さらに、この Web サーバーでは、セキュリティ保護されていない DNS クライアント要求を出すことができます。その他のすべてのトラフィックでは、Blowfish と SHA-1 アルゴリズムによる ESP が必要です。他のトラフィックではさらに、アウトバウンドトラフィックに共有 SA を使用します。共有 SA を使用すると、生成しなければならないセキュリティアソシエーションの数が少なくて済みます。

1. システムコンソールから、スーパーユーザーになるか、同等の役割を引き受けます。

注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

2. セキュリティポリシー検査を省略するサービスを指定します。

   Web サーバーの場合、TCP ポート 80 (HTTP) と 443 (保護 HTTP) が該当します。Web サーバーが DNS 名検査をするときは、TCP と UDP の両方にポート 53 も組み込む必要がある場合もあります。

3. **Web** サーバーポリシー用のファイルを **/etc/inet/** ディレクトリに作成します。このファイルにその目的を表す名前を与えます **(**たとえば、**IPsecWebInitFile)**。このファイルに次のように入力します。

```
# Web traffic that Web server should bypass.
    {sport 80 ulp tcp} bypass {dir out}
    {dport 80 ulp tcp} bypass {dir in}
    {sport 443 ulp tcp} bypass {dir out}
    {dport 443 ulp tcp} bypass {dir in}

    # Outbound DNS lookups should also be bypassed.
    {dport 53} bypass {dir out}
    {sport 53} bypass {dir in}

    # Require all other traffic to use ESP with Blowfish and SHA-1.
    # Use a shared SA for outbound traffic, in order to avoid a
    # large supply of security associations.
    {} permit {encr_algs blowfish encr_auth_algs sha}
    {} apply {encr_algs blowfish encr_auth_algs sha sa shared}
```

   これで、保護トラフィックだけがシステムにアクセスできるようになります。ただし、先の手順で説明した、検査を省略するトラフィックは例外です。

4. 先の手順で作成したファイルを **/etc/inet/ipsecinit.conf** に読み込みます。

```
# vi  /etc/inet/ipsecinit.conf
 :r IPsecWebInitFile
 :wq!
```

5. **IPsecWebInitFile** ファイルを読み取り専用アクセス権で保護します。

```
# chmod 400 IPsecWebInitFile
```

6. リブートせずに **Web** サーバーを保護するために、**ipseckey** コマンドと **ipsecconf** コマンドを使用します。

```
# ipseckey -f /etc/inet/secret/ipseckeys
# ipsecconf -a /etc/inet/ipsecinit.conf
```

注 – このコマンドの実行時には警告を読んでください。ソケットがすでに使用中 (ラッチされた) の場合には、システムのセキュリティが低下します。

リブートすることもできます。システムをリブートすると、IPsec ポリシーがすべての TCP 接続に適用されます。リブート時に、IPsec ポリシーのファイルで指定したポリシーが TCP 接続でラッチされます。

これで、Web サーバーでは、Web サーバートラフィックとアウトバウンド DNS 要求と応答だけを処理するようになりました。他のサービスは、IPsec をリモートシステムで有効にしないと機能しません。

リモートシステムと Web サーバーの間で非 Web トラフィックを安全に通信したい場合は、両方のポリシーが一致していなければなりません。リモートシステムの `ipsecinit.conf` ファイルに次のポリシーが設定されていれば、リモートシステムと Web サーバーは安全に通信できます。

```
# Communicate with Web server about non-Web stuff
#
    {} permit {encr_algs blowfish encr_auth_algs sha}
    {} apply {encr_algs blowfish encr_auth_algs sha sa shared}
```

▼ 仮想プライベートネットワークを構築する方法

この手順では、インターネットで VPN を構築して組織内の 2 つのネットワークを接続する方法について説明します。また、そのネットワーク間のトラフィックを IPsec で保護する方法について説明します。

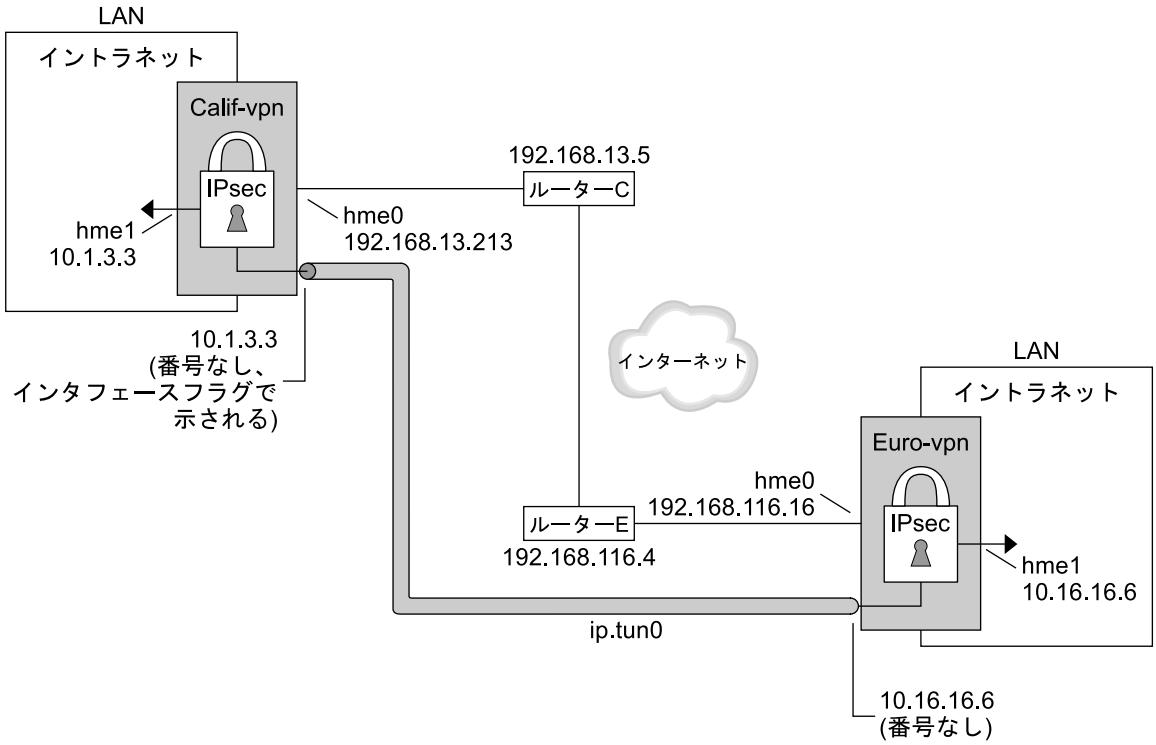
この手順は、30 ページの「2 つのシステム間のトラフィックを保護する方法」 の手順を拡張するものです。この手順では、2 つのマシンを接続するだけでなく、これら 2 つのマシンに接続している 2 つのイントラネットを接続します。この手順における 2 つのマシンはゲートウェイとして機能します。

この手順では、次の設定がすでになされているものとします。

- 各システムが IPv4 アドレス空間を使用している
- 各システムには 2 つのインタフェースがある。`hme0` インタフェースはインターネットに接続している。この例では、インターネット IP アドレスは `192.168` で始まる。`hme1` インタフェースは社内の LAN (イントラネット) に接続している。この例では、イントラネット IP アドレスは `10` で始まる
- 各システムは、MD5 アルゴリズムを使って AH 保護を呼び出す。MD5 アルゴリズムには 128 ビットキーが必要
- 各システムは、3DES アルゴリズムを使って ESP 保護を呼び出す。3DES アルゴリズムには 192 ビットキーが必要
- 各システムは、インターネットに直接アクセスするルーターに接続できる

- IPsec は、共有セキュリティアソシエーションを使用する

VPN については、19 ページの「仮想プライベートネットワーク」を参照してください。次の図は、この手順によって設定される VPN を表しています。



hme0 = IP 転送をオフ

hme1 = IP 転送をオン

ip.tun = IP 転送をオン

ルーターC ― Calif-vpn 用 /etc/defaultrouter

ルーターE ― Euro-vpn 用 /etc/defaultrouter

この手順では、次の構成パラメータを使用します。

パラメータ	ヨーロッパ	カリフォルニア
ホスト名	enigma	partym

パラメータ	ヨーロッパ	カリフォルニア
システムイントラネットインタフェース	hme1	hme1
システムインターネットインタフェース	hme0	hme0
システムイントラネットアドレス (手順 8 の *-point* アドレスでもある)	10.16.16.6	10.1.3.3
システムインターネットアドレス (手順 8 の *-taddr* アドレスでもある)	192.168.116.16	192.168.13.213
インターネットルーターの名前	router-E	router-C
インターネットルーターのアドレス	192.168.116.4	192.168.13.5
トンネル名	`ip.tun0`	`ip.tun0`

1. どれかのシステムのシステムコンソールで、スーパーユーザーになるか、同等の役割を引き受けます。

注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

---

2. 次のコマンドを入力して **IP** 送信をオフにします。

```
# ndd -set /dev/ip ip_forwarding 0
```

IP 送信をオフにすると、このシステムを経由したネットワーク間のパケット送信ができなくなります。`ndd` コマンドについては、`ndd(1M)` のマニュアルページを参照してください。

3. 次のコマンドを入力して **IP** の厳密宛先マルチホームをオンにします。

```
# ndd -set /dev/ip ip_strict_dst_multihoming 1
```

IP 厳密宛先マルチホームをオンにすると、システムの宛先アドレスに宛てたパケットは、正しい宛先アドレスに必ず到着します。

`ndd` コマンドを使用して IP 送信をオフにし、IP 厳密宛先をオンにすると、システムを流れるパケットの数は少なくなります。マルチホームによって、システムのアドレス宛のパケットを除き、パケットの流れがシャットダウンされるからです。システムアドレスの場合は、マルチホームによって、宛先 IP アドレスに対応するインタフェースに到着するパケットだけが送信されます。

4. 必要に応じて次の手順を行い、**Solaris** システム上の大部分 **(**場合によってはすべて**)** のネットワークサービスを無効にします。

   a. **`inetd.conf`** を編集し、重要なサービス以外のすべてのサービスを削除してから、次のコマンドを入力します。

```
# pkill -HUP inetd
```

注 – VPN ルーターは、ほとんどの入力要求を受け付けません。入力トラフィックを受信するすべてのプロセスを無効にする必要があります。たとえば、`inetd.conf` ファイルの一部をコメントにしたり、SNMP を停止したりします。あるいは、35 ページの「Web サーバーを保護する方法」で使用したようなテクニックを使うこともできます。

---

b. 重要なサービス以外のすべてのサービスを削除するための **inetd.conf** の編集をしていない場合は、次のコマンドを入力します。

```
# pkill inetd
```

c. 必要に応じて、次の例のようなコマンドを **1** つまたは複数入力して **SNMP**、**NFS** など他のインターネットサービスを無効にします。

```
# /etc/init.d/nfs.server stop
# /etc/init.d/sendmail stop
```

ネットワークサービスを無効にすると、IP パケットによるシステムへの妨害がなくなります。たとえば、SNMP デーモン、telnet、rlogin を最大限に活用できます。

5. システムごとに、**2** つのシステム間のセキュリティアソシエーションの組を追加します。

IKE デーモンは、セキュリティアソシエーションを作成するように設定されていれば、セキュリティアソシエーションを自動的に作成します。VPN に IKE を設定するには、次のいずれかの手順を実行します。

- 62 ページの「事前共有鍵による IKE の設定方法」
- 70 ページの「自己署名付き公開証明書による IKE の設定方法」
- 72 ページの「認証局による署名付き公開鍵による IKE の設定方法」

システムで IPv6 アドレスを使用している場合には、手動でセキュリティアソシエーションを作成する必要があります。手順については、44 ページの「IPsec セキュリティアソシエーションを手動で作成する方法」を参照してください。

6. システムごとに、**/etc/inet/ipsecinit.conf** ファイルを編集して **VPN** ポリシーを追加します。

a. たとえば、**enigma** で、**ipsecinit.conf** ファイルに次のエントリを入力します。

```
# LAN traffic can bypass IPsec.
   {laddr 10.16.16.6 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
   {} ipsec {encr_algs 3des encr_auth_algs md5}
```

b. たとえば、**partym** で、**ipsecinit.conf** ファイルに次のエントリを入力します。

```
# LAN traffic can bypass IPsec.
   {laddr 10.1.3.3 dir both} bypass {}

# WAN traffic uses ESP with 3DES and MD5.
   {} ipsec {encr_algs 3des encr_auth_algs md5}
```

ipsec エントリは、リモートシステムがクリアパケットを送信してくるのを防止します。bypass エントリを指定すると、LAN に属するノードでは、VPN ルーターを、それが LAN の一部であるかのように扱うことができます。

7. **(省略可能)** これより高いレベルのセキュリティが必要な場合は、**LAN bypass** エントリを削除します。

ipsecinit.conf のエントリは次のようになります。

```
# All traffic uses ESP with 3DES and MD5.
   {} ipsec {encr_algs 3des encr_auth_algs md5}
```

これによって、LAN 上の各システムが VPN ルーターと通信するには、IPsec の起動が必要になります。

8. システムごとに、保護トンネル **ip.tun0** を設定します。

このトンネルは、IP から見たもう 1 つの物理的インタフェースを追加します。3 つの ifconfig コマンドを入力してポイントツーポイントインタフェースを作成します。

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 system1-point system2-point \
tsrc system1-taddr tdst system2-taddr encr_algs 3DES encr_auth_algs MD5

# ifconfig ip.tun0 up
```

a. たとえば、**enigma** で次のコマンドを入力します。

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 10.16.16.6 10.1.3.3   \
tsrc 192.168.116.16 tdst 192.168.13.213 encr_algs 3DES encr_auth_algs MD5

# ifconfig ip.tun0 up
```

b. たとえば、**partym** で次のコマンドを入力します。

```
# ifconfig ip.tun0 plumb

# ifconfig ip.tun0 10.1.3.3 10.16.16.6  \
tsrc 192.168.13.213 tdst 192.168.116.16 encr_algs 3DES encr_auth_algs MD5

# ifconfig ip.tun0 up
```

ifconfig コマンドに渡すポリシーは、ipsecinit.conf ファイルに指定されているポリシーと同じでなければなりません。リブート時に、各システムは ipsecinit.conf ファイルに指定されているポリシーを使用します。

9. システムごとに、**hme1** と **ip.tun0** インタフェースの **ip_forwarding** をオンにします。

```
# ndd -set /dev/ip hme1:ip_forwarding 1

# ndd -set /dev/ip ip.tun0:ip_forwarding 1
```

ip_forwarding は、別のインタフェースから到着したパケットを転送できることを意味します。ip_forwarding はまた、送信するパケットがもともとは別のインタフェースから発信されたパケットである可能性も、意味します。パケットを正しく転送するには、受信インタフェースと送信インタフェースの ip_forwarding をオンにしておきます。

hme1 インタフェースはイントラネットの内部にあるため、hme1 のip_forwarding はオンにしておきます。さらに、ip.tun0 はインターネットを通してこれら 2 つのシステムに接続されているため、ip.tun0 のip_forwarding はオンにしておきます。

hme0 インタフェースの ip_forwarding はオフです。そのため、外部からパケットが保護イントラネットに侵入するのを防ぐことができます。外部とはインターネットを意味します。

10. システムごとに、次のコマンドを入力して、ルーティングプロトコルによってイントラネット内のデフォルトのルートが通知されないようにします。

```
# ifconfig hme0 private
```

hme0 の ip_forwarding がオフになっていても、ルーティングプロトコルの実装によっては、このインタフェースを通知することがあります。たとえば、in.routed プロトコルは、イントラネット内のピアにパケットが転送される際に hme0 を有効なインタフェースとして通知する場合があります。インタフェースの *private* フラグを設定すれば、このような通知を防止できます。

11. **hme0** 経由のデフォルトルートを手動で追加します。

このルートは、インターネットに直接アクセスできるルーターでなければなりません。

```
# pkill in.rdisc

# route add default router-on-hme0-subnet
```

a. たとえば、**enigma** で次のルートを追加します。

```
# pkill in.rdisc
# route add default 192.168.116.4
```

b. **partym** で次のルートを追加します。

```
# pkill in.rdisc
# route add default 192.168.13.5
```

hme0 インタフェースはイントラネットの一部ではありませんが、インターネットを介してそのピアシステムにアクセスする必要があります。hme0 は、自身のピアを見つけるために、インターネットルーティング情報を必要とします。インターネットの残りの要素にとって、VPN システムは、ルーターというよりもホストのような存在です。したがって、デフォルトルーターを使用するか、ルーター発見プ

ロトコルを実行すれば、ピアシステムを見つけることができます。詳細については、route(1M) と in.routed(1M) のマニュアルページを参照してください。

12. リブート後に **hme0** がデフォルトルートを使用するように、**defaultrouter** ファイルを作成します。

    /etc/defaultrouter ファイルに hme0 のデフォルトルーターの IP アドレスを入力します。この手順により、in.rdisc デーモンがリブート時に起動しなくなります。

    a. たとえば、**enigma** で、そのインターネットルーターを **/etc/defaultrouter** ファイルに入力します。

    ```
    # vi /etc/defaultrouter

    192.168.116.4 router-E
    ```

    b. **partym** のインターネットルーターを **partym** の **/etc/defaultrouter** ファイルに入力します。

    ```
    # vi /etc/defaultrouter

    192.168.13.5 router-C
    ```

13. システムごとに、ブートシーケンスの初期にルーティングが起こらないようにします。これによって、セキュリティの脆弱性が軽減されます。

    ```
    # touch /etc/notrouter
    ```

14. **VPN** がリブート後に開始するように、**/etc/hostname.ip.tun0** ファイルを編集します。

    system1-point system2-point tsrc *system1-taddr* \
    tdst *system2-taddr* encr_algs 3des encr_auth_algs md5 up

    a. たとえば、**enigma** で、**hostname.ip.tun0** ファイルに次のように追加します。

    ```
    10.16.16.6 10.1.3.3 tsrc 192.168.116.16 \
    tdst 192.168.13.213 encr_algs 3DES encr_auth_algs MD5 up
    ```

    b. **partym** で、**hostname.ip.tun0** ファイルに次のように追加します。

    ```
    10.1.3.3 10.16.16.6 tsrc 192.168.13.213 \
    tdst 192.168.116.16 encr_algs 3DES encr_auth_algs MD5 up
    ```

15. システムごとに、**VPN** パラメータの一部をブート時に設定するファイルを作成します。このファイルに **/etc/rc3.d/S99vpn_setup** という名前を付け、次のように入力します。

    ```
    ndd -set /dev/ip hme1:ip_forwarding 1
    ndd -set /dev/ip ip.tun0:ip_forwarding 1
    ifconfig hme0 private
    in.routed
    ```

    in.routed を使用する代わりに、手動で /etc/rc3.d/S99vpn_setup ファイルにルートを追加することもできます。

16. システムごとに、次のコマンドを入力してルーティングプロトコルを実行します。

```
# in.routed
```

▼ 乱数を生成する方法

乱数発生関数がすでにある場合は、それを使用してください。ない場合は、Solaris の /dev/random デバイスを入力として od コマンドを実行することができます。詳細は、od(1) のマニュアルページを参照してください。

1. ランダムなキーを生成します。

   Solaris システムでは、od コマンドを使用できます。

```
# od -X -A n file
```

-x	8 進数ダンプを 16 進数形式で表示する。16 進数形式はキー情報を表すのに役立つ。16 進数を 4 文字単位で表示する
-X	8 進数ダンプを 16 進数形式で表示する。16 進数を 8 文字単位で表示する
–A n	表示から入力オフセットベースを取り除く
file	乱数のソース

   たとえば、次のコマンドを入力すると、16 進数の乱数がそれぞれ次のように表示されます。

```
# od -X -A n /dev/random | head -2
         d54d1536 4a3e0352 0faf93bd 24fd6cad
         8ecc2670 f3447465 20db0b0c c83f5a4b
# od -x -A n /dev/random | head -2
         34ce 56b2 8b1b 3677 9231 42e9 80b0 c673
         2f74 2817 8026 df68 12f4 905a db3d ef27
```

2. これらの乱数を組み合わせて、適切な長さのキーを作成します。

   同じ行にある乱数間のスペースを取り除き、32 文字のキーを作成します。32 文字キーの長さは 128 ビットです。SPI の場合は、8 文字の乱数 1 個を使用できます。

▼ IPsec セキュリティアソシエーションを手動で作成する方法

システムで IPv6 アドレスを使用している場合には、手動で IPsec セキュリティアソシエーションを作成する必要があります。

注 – IPv4 ネットワークを使用している場合は、IKE を使ってセキュリティアソシエーションを管理します。IKE を使って SA を管理する方法については、61 ページの「IKE の実装 (作業マップ)」を参照してください。

---

1. どれかのシステムのシステムコンソールで、スーパーユーザーになるか、同等の役割を引き受けます。

注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

---

2. 次のコマンドを入力して **ipseckey** コマンドモードを有効にします。

```
# ipseckey

>
```

> プロンプトは、ipseckey コマンドモードになったことを示します。

3. セキュリティアソシエーションを作成したり、フラッシュしたばかりのセキュリティアソシエーションを置き換えたりするには、次のコマンドを実行します。

```
> add protocol spi random-hex-string \
src addr dst addr2 \
protocol_alg protocol-algorithm   \
protocolkey random-hex-string-of-algorithm-specified-length
```

random-hex-string	16 進数形式の最大 8 桁の乱数。SPI が受け取れる以上の桁数を入力すると、超過部分は無視される。SPI が受け取れるより少ない桁数を入力すると、パディングが行われる
protocol	esp または ah
addr	システムの IP アドレス
addr2	*addr* のピアシステムの IP アドレス
protocol-algorithm	ESP または AH のアルゴリズム。それぞれのアルゴリズムには、特定の長さのキーが必要 認証アルゴリズムには MD5 と SHA がある。暗号化アルゴリズムには 3DES と AES がある
random-hex-string-of-algorithm-specified-length	アルゴリズムによって必要とされる長さをもつ 16 進数の乱数。たとえば、MD5 アルゴリズムでは、128 ビットキーのため 32 桁の乱数が必要。3DES アルゴリズムでは、192 ビットキーのため 48 桁の乱数が必要

a. たとえば、**enigma** で、次のコマンドを入力してアウトバウンドパケットを保護します。生成した乱数を使用します。

```
> add esp spi 8bcd1407 src 192.168.116.16 dst 192.168.13.213 \
encr_alg 3DES \
encrkey d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d

> add ah spi 18907dae src 192.168.116.16 dst 192.168.13.213 \
auth_alg MD5 \
authkey e896f8df7f78d6cab36c94ccf293f031

>
```

注 – ピアシステムでは、同じキー情報を使用する必要があります。

---

b. 引き続き **ipseckey** モードを使って、**enigma** で、次のコマンドを入力してインバウンドパケットを保護します。生成した乱数を使用します。

```
> add esp spi 122a43e4 src 192.168.13.213 dst 192.168.116.16 \
encr_alg 3des \
encrkey dd325c5c137fb4739a55c9b3a1747baa06359826a5e4358e

> add ah spi 91825a77 src 192.168.13.213 dst 192.168.116.16 \
auth_alg md5 \
authkey ad9ced7ad5f255c9a8605fba5eb4d2fd

>
```

注 – これらのキーとSPI は、セキュリティアソシエーションごとに変更できます。セキュリティアソシエーションごとに、異なるキーと異なる SPI を割り当てるべきです。

---

4. **Control-D** か **quit** を使って **ipseckey** コマンドモードを終了します。

5. リブート時に **IPsec** がキー情報を使用できるように、**enigma** の **/etc/inet/secret/ipseckeys** ファイルにキー情報を追加します。

```
add esp spi 8bcd1407 dst partym  encr_alg 3DES \
   encrkey  d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d
#
add ah spi  18907dae  dst partym auth_alg MD5  \
   authkey  e896f8df7f78d6cab36c94ccf293f031
#
#
add esp spi 122a43e4 dst enigma encr_alg 3DES \
    encrkey 137fb4739a55c9b3a1747baa06359826a5e4358e
#
add ah spi  91825a77  dst enigma auth_alg MD5  \
   authkey  ad9ced7ad5f255c9a8605fba5eb4d2fd
```

6. **partym** で、手順 **1** から手順 **5** を繰り返します。
   両システムのキー情報は同じでなければなりません。

例 — IPsec セキュリティアソシエーションの置き換え

暗号化システムを破る時間的な猶予を与えないためには、キー情報を更新する必要があります。あるシステムの SA を置き換える場合は、それと通信しているシステムの SA も置き換える必要があります。

セキュリティアソシエーションを置き換える場合は、古いキーを削除してから新しいキーを追加します。古いキーを削除するには、`ipseckey` コマンドモードで `flush` コマンドを実行します。そのあとに新しいキー情報を追加します。

```
# ipseckey
> flush
> add esp spi ...
```

▼ パケットが保護されていることを確認する方法

パケットが保護されていることを確認するには、`snoop` コマンドで接続をテストします。`snoop` 出力に表示される接頭辞は、次のとおりです。

- `AH:` 接頭辞 – AH がヘッダーを保護していることを示す。`AH:` が表示されるのは、`auth_alg` を使ってトラフィックを保護している場合
- `ESP:` 接頭辞 – 暗号化されたデータが送信されていることを示す。`ESP:` が表示されるのは、`encr_auth_alg` か `encr_alg` を使ってトラフィックを保護している場合

注 – snoop 出力を読むためには、root であるか、それと同等の役割でなければなりません。さらに、接続をテストするためには、両方のシステムにアクセスできなければなりません。

---

1. 一方のシステム **(**たとえば、**partym)** で **root** になります。

```
% su
Password: root-password
#
```

2. 端末ウィンドウで、別のシステム **(**たとえば、**enigma)** から来るパケットの **snoop** を開始します。

```
# snoop -v enigma
Using device /dev/hme (promiscuous mode)
```

3. 別の端末ウィンドウで、**enigma** システムにリモートからログオンします。パスワードを入力します。次に、**root** になり、**enigma** からのパケットを **partym** システムに送信します。

```
% rlogin enigma
Password: your-password
% su
Password: root-password
# ping partym
```

4. **partym** の **snoop** ウィンドウに次のような出力が表示されます。

```
IP:   Time to live = 64 seconds/hops
IP:   Protocol = 51 (AH)
IP:   Header checksum = 4e0e
IP:   Source address = 192.168.116.16, enigma
IP:   Destination address = 192.168.13.213, partym
IP:   No options
IP:
AH:  ----- Authentication Header -----
AH:
AH:  Next header = 50 (ESP)
AH:  AH length = 4 (24 bytes)
AH:  <Reserved field = 0x0>
AH:  SPI = 0xb3a8d714
AH:  Replay = 52
AH:  ICV = c653901433ef5a7d77c76eaa
AH:
ESP:  ----- Encapsulating Security Payload -----
ESP:
ESP:  SPI = 0xd4f40a61
ESP:  Replay = 52
ESP:      ....ENCRYPTED DATA....

ETHER:  ----- Ether Header -----
ETHER:
ETHER:  Packet 20 arrived at 9:44:36.59
ETHER:  Packet size = 98 bytes
ETHER:  Destination = 8:0:27:aa:11:11, Sun
ETHER:  Source      = 8:0:22:aa:22:2, Sun
ETHER:  Ethertype = 0800 (IP)
ETHER:
IP:   ----- IP Header -----
IP:
IP:   Version = 4
IP:   Header length = 20 bytes
IP:   Type of service = 0x00
IP:         xxx. .... = 0 (precedence)
IP:         ...0 .... = normal delay
IP:         .... 0... = normal throughput
IP:         .... .0.. = normal reliability
IP:         .... ..0. = not ECN capable transport
IP:         .... ...0 = no ECN congestion experienced
IP:   Total length = 84 bytes
IP:   Identification = 40933
IP:   Flags = 0x4
IP:         .1.. .... = do not fragment
IP:         ..0. .... = last fragment
IP:   Fragment offset = 0 bytes
```

```
IP:   Time to live = 60 seconds/hops
IP:   Protocol = 51 (AH)
IP:   Header checksum = 22cc
...
```

第 3 章

インターネットキー交換 (概要)

IP データグラムのセキュリティ保護された伝送に必要な IPsec SA (セキュリティアソシエーション) のキー情報の管理を、キー管理といいます。自動キー管理では、キーの作成、認証、および交換のため、セキュリティ保護された通信チャネルを必要とします。Solaris オペレーティングシステムでは、インターネットキー交換 (IKE) を使用してキー管理を自動化します。IKE を使用すれば、セキュリティ保護されたチャネルを大量のトラフィックに割り当てるために容易にスケーリングできます。IPv4 パケットの IPsec SA では、IKE の利点を生かすことができます。

Sun™ Crypto Accelerator 1000 カードを搭載したシステムで IKE を使用する場合には、公開鍵の操作はこのカードで行われます。公開鍵の操作のためにオペレーティングシステムのリソースが使用されることはありません。

この章では、以下の内容について説明します。

IKE の概要

インターネットキー交換 (IKE) デーモン `in.iked(1M)` では、保護された方法でセキュリティアソシエーションのキー情報のネゴシエーションと認証を行います。また、SunOS™ によって提供される内部機能からキーのランダムシードを使用します。IKE は、PFS (Perfect Forward Secrecy) をサポートしています。PFS では、データ伝送を保護するキーを使用しないで追加キーを取得し、データ伝送のキーの作成に使用するシードを再利用しません。

IKE デーモンによってリモートホストの公開暗号鍵が検出されると、ローカルシステムではその鍵を使用できるようになります。ローカルシステムは、リモートホストの公開鍵を使用してメッセージを暗号化します。メッセージを読み取れるのは、このリモートホストだけです。IKE デーモンでは、そのジョブを交換と呼ばれる 2 つのフェーズで実行します。

フェーズ 1 交換

フェーズ 1 交換はメインモードといいます。フェーズ 1 交換では、IKE は公開鍵暗号方式を使用して、ピア IKE エンティティによる IKE 自体を認証します。その結果が ISAKMP (Internet Security Association and Key Management Protocol) セキュリティアソシエーションで、IKE で IP データグラムのキー情報のネゴシエーションを行うためのセキュリティ保護されたチャネルとなります。IPsec SA とは異なり、ISAKMP セキュリティアソシエーションは双方向であるため、1 つだけ必要です。

IKE でキー情報のネゴシエーションを行う方法は、フェーズ 1 交換で設定可能です。IKE では、`/etc/inet/ike/config` ファイルから設定情報を読み取ります。設定情報には、影響するインタフェース、使用するアルゴリズム、認証方式、および PFS 使用の有無が含まれています。認証方式には、事前共有鍵と公開鍵証明書の 2 つがあります。公開鍵証明書は、自己署名付きにすることも、PKI (Public Key Infrastructure) 機関から認証局 (CA) によって発行することもできます。PKI 機関には、Sun™ Open Net Environment (Sun ONE) Certificate Server、Entrust、および Verisign があります。

フェーズ 2 交換

フェーズ 2 交換はクイックモードといいます。フェーズ 2 交換では、IKE は IKE デーモンを実行するホスト間の IPsec SA を作成および管理します。また、フェーズ 1 で作成したセキュリティ保護されたチャネルを使用して、キー情報の伝送を保護します。IKE デーモンは、`/dev/random` を使用して乱数発生関数からキーを作成します。また、IKE デーモンは、キーを一定の割合 (構成可能) で更新します。このキー情報は、IPsec ポリシーの構成ファイルに指定されているアルゴリズムによって使用されます。

IKE 構成の選択

2 つの IKE デーモンが相互を認証するためには、IKE ポリシーの構成ファイル `ike.config(4)` が有効でなければなりません。さらに、キー情報も必要です。この構成ファイルには、IKE ポリシーのエントリが格納されています。これらのエントリは、フェーズ 1 交換の認証方式を決定します。選択肢は、事前共有鍵か公開鍵証明書のどちらかです。

鍵のペア auth_method preshared は、事前共有鍵が使用されることを示します。auth_method の値が preshared 以外の場合には、公開鍵証明書が使用されることを示します。公開鍵証明書は、自己署名付きにすることも、PKI 機関から発行することもできます。

事前共有鍵の使用

事前共有鍵は、1 つのシステムの管理者によって作成され、通信するシステムの管理者とアウトオブバンドで共有します。管理者は、大量のランダム鍵の作成、そのファイルとアウトオブバンド伝送の保護に十分注意する必要があります。鍵は、各システムの /etc/inet/secret/ike.preshared ファイルに保存されます。IPsec の場合は ipseckeys ファイルですが、IKE の場合は ike.preshared(4) ファイルとなります。ike.preshared ファイルにある鍵に問題があると、その鍵から導出されるすべての鍵に問題が発生します。

1 つのシステムの事前共有鍵は、通信するシステムの鍵と同一にする必要があります。鍵は特定の IP アドレスに連結され、そのセキュリティ保護は管理者が通信するシステムを制御する場合に最も強化されます。

公開鍵証明書の使用

公開鍵証明書を使用すると、通信するシステムが秘密鍵情報をアウトオブバンドで共有する必要がなくなります。公開鍵では、鍵の認証とネゴシエーションに Diffie-Hellman 方式を採用します。公開鍵証明書には、2 つの方法があります。公開鍵証明書は、自己署名付きにすることも、認証局 (CA) が認証することもできます。

自己署名付き公開鍵証明書は、管理者によって作成されます。ikecert certlocal -ks コマンドを実行して、システムの公開鍵と非公開鍵のペアの非公開部分を作成します。その後、管理者は通信するシステムから X.509 形式で自己署名付き証明書の出力を取得します。通信するシステムの証明書は、鍵のペアの公開部分の ikecert certdb コマンドに入力されます。自己署名付き証明書は、通信するホストの /etc/inet/ike/publickeys ディレクトリに保存されます。

自己署名付き証明書は、事前共有鍵と CA 間の中間ポイントになります。事前共有鍵とは異なり、自己署名付き証明書は移動体システムまたは再番号付けされる可能性があるシステムで使用できます。証明書に自己署名するには、管理者は DNS (www.example.org) または EMAIL (root@domain.org) の代替名を使用します。

公開鍵は、PKI または CA 機関で配信できます。公開鍵とそれに関連する CA は、管理者によって /etc/inet/ike/publickeys ディレクトリに格納されます。また、ベンダーは証明書無効リスト (CRL) も発行します。管理者は鍵と CA を格納するだけでなく、CRL を /etc/inet/ike/crls ディレクトリに格納する責任があります。

CA には、サイトの管理者ではなく、外部の機関によって認証されるといった特長があります。その点では、CA は公証された証明書となります。自己署名付き証明書と同様に、CA は移動体システムまたは再番号付けされる可能性があるシステムで使用できます。その一方、自己署名付き証明書とは異なり、CA は通信する多くのシステムを保護するために容易にスケーリングします。

IKE と加速ハードウェア

IKE アルゴリズムは、とりわけそのフェーズ 1 交換において、多くの処理を要します。大量の交換を処理するシステムでは、Sun Crypto Accelerator 1000 カードを使って公開鍵の操作を処理することができます。IKE の計算負荷をアクセラレータカードに移すための IKE 構成方法については、80 ページの「IKE で Sun Crypto Accelerator 1000 カードを使用する方法」を参照してください。

IKE ユーティリティおよび IKE ファイル

この節では、IKE ポリシーの構成ファイルと、IKE を実装するさまざまなコマンドについて説明します。IPv4 ネットワークに IKE を実装する方法の手順については、61 ページの「IKE の実装 (作業マップ)」を参照してください。

表 **3–1** IKE ファイルおよび IKE コマンドのリスト

ファイルまたはコマンド	説明
`in.iked(1M)` デーモン	インターネットキー交換 (IKE) デーモン。自動キー管理を有効にする
`ikeadm(1M)`	IKE 管理コマンド。IKE ポリシーの表示および変更用
`ikecert(1M)`	認証データベース管理コマンド。ローカル公開鍵の認証データベースの操作用
`/etc/inet/ike/config` ファイル	IKE ポリシーの構成ファイル。インバウンド IKE 要求のマッチングとアウトバウンド IKE 要求の準備に関するサイトの規則が含まれる。このファイルがある場合には、`in.iked` デーモンがブート時に自動的に開始する

表 3–1 IKE ファイルおよび IKE コマンドのリスト　(続き)

ファイルまたはコマンド	説明
/etc/inet/secret/ike.preshared ファイル	事前共有鍵のファイル。フェーズ 1 認証の秘密鍵情報が含まれる。事前共有鍵を使って IKE を構成するときに使用
/etc/inet/secret/ike.privatekeys ファイル	非公開鍵のディレクトリ。公開鍵と非公開鍵のペアの非公開部分が含まれる
/etc/inet/ike/publickeys ディレクトリ	公開鍵と証明書ファイルを保存するディレクトリ。公開鍵と非公開鍵のペアの公開部分が含まれる
/etc/inet/ike/crls ディレクトリ	公開鍵と証明書ファイルの無効リストを保存するディレクトリ

IKE デーモン

in.iked(1M) デーモンを実行すると、Solaris ホスト上の暗号キーの管理が自動化されます。また、同じプロトコルを実行するリモートホストとのネゴシエーションを行い、認証されたキー情報が、保護された方法でセキュリティアソシエーションに提供されます。このデーモンは、セキュリティ保護された通信を行うすべてのホストで実行されている必要があります。

IKE ポリシーの構成ファイル /etc/inet/ike/config がある場合には、IKE デーモンがブート時に自動的にロードされます。デーモンは構成ファイルの構文を検査します。

IKE デーモンを実行すると、フェーズ 1 では、システムがそのピア IKE エンティティに対してそのシステム自体を認証します。そのピアは、認証方式として IKE ポリシーファイルに定義されています。その後、フェーズ 2 では、セッションのキーが設定されます。ポリシーファイルで指定した時間間隔で、IKE キーが自動的に更新されます。in.iked デーモンを実行すると、ネットワークから着信する IKE 要求と、 PF_KEY ソケット経由のアウトバウンドトラフィックの要求を待機します。詳細については、pf_key(7P) のマニュアルページを参照してください。

2 つのプログラムで IKE デーモンをサポートします。ikeadm(1M) コマンドを実行すると、管理者は IKE ポリシーを表示できます。さらに、このコマンドを実行して IKE ポリシーを変更することもできます。ikecert(1M) コマンドを実行すると、管理者は 2 つの公開鍵データベース ike.privatekeys と publickeys を表示したり管理したりできます。

IKE ポリシーファイル

IKE ポリシーの構成ファイル /etc/inet/ike/config により、IKE デーモン自体のキー情報、およびそのファイルが管理する IPsec SA のキー情報が提供されます。IKE デーモン自体は、フェーズ 1 交換でキー情報を要求します。ike/config ファイルに

ある規則に基づいてキー情報が設定されます。ポリシーファイルにある有効な規則にはラベルが含まれています。その規則により、キー情報を使用するホストまたはネットワークが特定され、認証方式が指定されます。有効なポリシーファイルの例については、62 ページの「IKE 作業」を参照してください。そのパラメータの例と説明については、`ike.config(4)` のマニュアルページを参照してください。

IPsec SA は、IPsec ポリシーの構成ファイル `/etc/inet/ipsecinit.conf` で設定されるポリシーに従って保護される IP データグラムで使用されます。IKE ポリシーファイルにより、IPsec SA の作成時に PFS を使用するかどうかが決定されます。

`ike/config` ファイルのセキュリティに関する注意点は、`ipsecinit.conf` ファイルのセキュリティと同様です。詳細については、23 ページの「`ipsecinit.conf` と `ipsecconf` のセキュリティについて」を参照してください。

IKE 管理コマンド

`ikeadm` コマンドを実行すると、次のことができます。

- IKE デーモンプロセスの要素の表示
- IKE デーモンに渡すパラメータの変更
- 統計情報の表示
- IKE プロセスのデバッグ

例とオプションの詳細については、`ikeadm(1M)` のマニュアルページを参照してください。実行する IKE デーモンの権限レベルにより、表示および変更可能な IKE デーモンの要素が決まります。権限レベルには、次のものがあります。

0x0 (基本レベル)	基本レベルの権限では、キー情報を表示できない。また、キー情報の変更もできない。基本レベルは、`in.iked` デーモン実行時のデフォルトレベル。
0x1 (modkeys レベル)	modkeys レベルの権限では、事前共有鍵を削除、変更、または追加できる
0x2 (keymat レベル)	keymat レベルの権限では、`ikeadm` コマンドを指定して実際のキー情報を表示できる

`ikeadm` コマンドのセキュリティに関する注意点は、`ipseckey` コマンドのセキュリティと同様です。詳細については、24 ページの「`ipseckey` におけるセキュリティについて」を参照してください。

事前共有鍵ファイル

/etc/inet/secret/ ディレクトリには、ISAKMP SA と IPsec SA の事前共有鍵が格納されています。共有鍵を手動で作成すると、ike.preshared ファイルには ISAKMP SA の事前共有鍵、ipseckeys ファイルには IPsec SA の事前共有鍵が格納されます。secret ディレクトリは 0700 で保護されています。secret ディレクトリの中にあるファイルは 0600 で保護されています。

- ike/config ファイルが事前共有鍵を要求したときに、管理者は ike.preshared ファイルを作成します。ike.preshared ファイルには、ISAKMP SA (つまり、IKE 認証) のキー情報が含まれています。フェーズ 1 交換の認証に事前共有鍵を使用するため、このファイルを in.iked デーモンの開始前に有効にする必要があります。
- ipseckeys ファイルには、IPsec SA のキー情報が含まれています。IPv6 ホストの場合、このファイルのキーは手動で作成および更新します。ファイルを手動で管理する例については、30 ページの「IPsec 作業」を参照してください。IKE デーモンでは、このファイルを使用しません。IKE によって IPsec SA に対して生成されるキー情報は、カーネルに保存されます。

IKE 公開鍵のデータベースおよびコマンド

ikecert(1M) コマンドを実行すると、ローカルホストの公開鍵データベースを操作できます。このコマンドは、ike/config ファイルが公開鍵証明書を要求するときに使用します。IKE ではそれらのデータベースを使用してフェーズ 1 交換を認証するため、in.iked デーモンを起動する前に、それらのデータベースに必要な情報が含まれていなければなりません。3 つのサブコマンド certlocal、certdb、certrldb をそれぞれ実行して、3 つのデータベースを処理します。

ikecert certlocal コマンド

certlocal サブコマンドを実行して、/etc/inet/secret/ike.privatekeys ディレクトリにある非公開鍵データベースを管理します。このサブコマンドを選択すると、非公開鍵の追加、表示、および削除を行うことができます。また、自己署名付き証明書または証明書要求のいずれかを作成できます。-ks オプションを選択すると、自己署名付き証明書が作成され、-kc オプションを選択すると、証明書要求が作成されます。

非公開鍵を作成する場合、certlocal サブコマンドには、ike/config ファイルの情報が必要です。certlocal オプションと ike/config エントリの対応を次の表に示します。

表 **3–2** ike certlocal の値と ike/config の値の対応表

certlocal オプション	**ike/config** エントリ	注
-A 対象の代替名	cert_trust 対象代替名	証明書を一意に識別するニックネーム。指定可能な値は、IP アドレス、電子メールアドレス、およびドメイン名
-D *X.509* 識別名	*X.509* 識別名	国、組織名、組織単位、共通名を含む認証局のフルネーム
-t dsa-sha1	auth_method dss_sig	RSA よりもわずかに遅くなる。特許は登録されていない
-t rsa-md5 -t rsa-sha1	auth_method rsa_sig	DSA よりもわずかに速くなる。特許の期限切れは 2000 年 9 月 RSA 公開鍵は、最大ペイロードを暗号化するのに十分な長さが必要。一般的に識別名などの ID ペイロードが最大
-t rsa-md5 -t rsa-sha1	auth_method rsa_encrypt	RSA 暗号化により、IKE にある ID が不正侵入者から保護されますが、IKE ピアには互いの公開鍵の認識が要求されます。

ikecert certlocal -kc コマンドを指定して証明書要求を実行する場合、そのコマンドの出力を PKI 機関に送信します。会社が独自の PKI を運営している場合は、出力を PKI 管理者に送信します。機関または PKI 管理者はこれに基づいてキー情報を作成します。ユーザーは、返されたキー情報を certdb と certrldb サブコマンドへの入力として使用します。

ikecert certdb コマンド

certdb サブコマンドを実行して、公開鍵データベース /etc/inet/ike/publickeys を管理します。このサブコマンドを選択すると、証明書と公開鍵を追加、表示、および削除できます。また、通信するシステムで ikecert certlocal -ks コマンドを実行して作成された証明書を入力として受け入れます。手順については、70 ページの「自己署名付き公開証明書による IKE の設定方法」を参照してください。さらに、PKI または CA から受信する証明書も入力として受け入れます。手順については、72 ページの「認証局による署名付き公開鍵による IKE の設定方法」を参照してください。

ikecert certrldb コマンド

certrldb サブコマンドを実行して、証明書無効リスト (CRL; Certificate Revocation List) データベース /etc/inet/ike/crls を管理します。crls データベースには、公開鍵の無効リストが保存されています。よって、このリストには、すでに有効でな

い証明書が明記されます。PKI によって CRL が提供されるときに、ikecert certrldb コマンドを指定して CRL データベースにそれらの CRL を格納します。手順については、77 ページの「証明書無効リストにアクセスする方法」を参照してください。

/etc/inet/ike/publickeys ディレクトリ

/etc/inet/ike/publickeys ディレクトリには、公開鍵と非公開鍵のペアの公開部分とファイルにあるその証明書、つまり「スロット」が格納されています。/etc/inet/ike ディレクトリは 0755 で保護されます。ikecert certdb コマンドを使用して、そのディレクトリを読み込みます。

そのファイルには、別のシステムで生成された証明書の X.509 識別名がコード化形式で含まれています。自己署名付き証明書を使用する場合、そのコマンドへの入力として、通信するシステムの管理者から受信する証明書を使用します。PKI からの証明書を使用する場合、PKI から受け取る 2 つのキー情報をこのデータベースに格納します。PKI に送信した情報に基づいた証明書を格納します。また、PKI からの CA も格納します。

/etc/inet/secret/ike.privatekeys ディレクトリ

ike.privatekeys ディレクトリには、公開鍵と非公開鍵のペアの一部である非公開鍵ファイル、ISAKMP SA のキー情報が格納されています。このディレクトリは 0700 で保護されています。このデータベースにある非公開鍵は、publickeys データベースの公開鍵とペアにする必要があります。ikecert certlocal コマンドを実行して、コマンドのディレクトリを読み込みます。非公開鍵は、ペアとなる公開鍵、自己署名付き証明書や CA が /etc/inet/ike/publickeys ディレクトリに格納されてから有効になります。

/etc/inet/ike/crls ディレクトリ

/etc/inet/ike/crls ディレクトリには証明書無効リスト (CRL) ファイルが含まれています。各ファイルは、/etc/inet/ike/publickeys/ ディレクトリにある公開鍵証明書ファイルに対応しています。PKI 機関により、それらの証明書の CRL が提供されます。ikecert certrldb コマンドを使用して、そのデータベースを読み込みます。

第 4 章

インターネットキー交換 (手順)

この章では、IKE の実装手順について説明します。使用するシステムのために構成された IKE は、そのネットワークにおける IPsec のキー情報を自動的に生成します。IKE の実装手順を 表 4–1 に示します。

IKE の概要については、第 3 章を参照してください。`ikeadm(1M)`、`ikecert(1M)`、`ike.config(4)` の各マニュアルページの「EXAMPLES」セクションには、有益な手順が記載されています。

IKE の実装 (作業マップ)

表 4–1 IKE の実装 (作業マップ)

作業	説明	参照先
事前共有鍵による IKE の設定	有効な IKE ポリシーファイルと `ike.preshared` ファイルを作成する。また、システムをブートして IKE によって生成された鍵を使用する前に、IPsec ファイルも設定する	62 ページの「事前共有鍵による IKE の設定方法」
実行中の IKE システムでの事前共有鍵の更新	IKE 権限レベルをチェックし、通信するシステムの最新キー情報に応じて `ipseckeys` ファイルを編集する	65 ページの「既存の事前共有鍵を更新する方法」
実行中の IKE システムへの事前共有鍵の追加	IKE 権限レベルをチェックし、通信するシステムの最新キー情報に応じて `ikeadm` コマンドを実行する	66 ページの「新しい事前共有鍵を追加する方法」

表 **4–1** IKE の実装 (作業マップ)　*(続き)*

作業	説明	参照先
自己署名付き公開鍵証明書による IKE の設定	`ikecert certlocal -ks` コマンドを指定して自己署名付き証明書を作成し、`ikecert certdb` コマンドを指定して通信するシステムからの公開鍵を追加する	70 ページの「自己署名付き公開証明書による IKE の設定方法」
PKI 認証局による IKE の設定	`ikecert certlocal -kc` コマンドから PKI 機関に出力を送信し、その機関からの公開鍵、CA、CRL を格納する	72 ページの「認証局による署名付き公開鍵による IKE の設定方法」
CA 無効リストの更新	一元的なディストリビューションポイントを通して PKI 機関の CRL にアクセスする	77 ページの「証明書無効リストにアクセスする方法」
IKE に対する Sun Crypto Accelerator 1000 カードの使用	このデバイスのために PKCS#11 ライブラリへのパスを設定する	80 ページの「IKE で Sun Crypto Accelerator 1000 カードを使用する方法」

IKE 作業

この節では、IPv4 アドレスを使用する 2 つのシステム間でトラフィックを保護する鍵を自動的に管理する手順について説明します。IKE 実装では、鍵の長さが異なるさまざまなアルゴリズムが提供されます。鍵の長さは、サイトのセキュリティに応じて選択します。一般的に、鍵の長さが長いほど、セキュリティが高くなります。

役割の使用方法については、『*Solaris* のシステム管理 *(*セキュリティサービス*)*』の「役割によるアクセス制御 (手順)」を参照してください。

▼ 事前共有鍵による IKE の設定方法

1. システムコンソールから、スーパーユーザーになるか、同等の役割を引き受けます。

   注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

   ---

2. システムごとに、**`/etc/inet/ike/config.sample`** ファイルを **`/etc/inet/ike/config`** にコピーします。
3. システムごとに、規則とグローバルパラメータを **`ike/config`** ファイルに入力します。

これらの規則やグローバルパラメータは、システムの ipsecinit.conf ファイルに設定されている IPsec ポリシーが正しく動作するものでなければなりません。次の ike/config の例は、30 ページの「2 つのシステム間のトラフィックを保護する方法」の ipsecinit.conf の例に対応しています。

a. たとえば、**enigma** システムの **/etc/inet/ike/config** ファイルを次のように変更します。

```
### ike/config file on enigma, 192.168.116.16

## Global parameters
#
## Phase 1 transform defaults
p1_lifetime_secs 14400
p1_nonce_len 40
#
## Defaults that individual rules can override.
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg des }
p2_pfs 2
#
## The rule to communicate with partym

{ label "Enigma-Partym"
  local_addr 192.168.116.16
  remote_addr 192.168.13.213
  p1_xform
   { auth_method preshared oakley_group 5 auth_alg md5 encr_alg 3des }
  p2_pfs 5
    }
```

注 – auth_method のすべての引数は同じ行になければなりません。

---

b. **partym** システムのファイルを次のように変更します。

```
### ike/config file on partym, 192.168.13.213
## Global Parameters
#
p1_lifetime_secs 14400
p1_nonce_len 40
#
p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg des }
p2_pfs 2

## The rule to communicate with enigma

{ label "Partym-Enigma"
  local_addr 192.168.13.213
  remote_addr 192.168.116.16
  p1_xform
   { auth_method preshared oakley_group 5 auth_alg md5 encr_alg 3des }
```

```
  p2_pfs 5
}
```

注 – システム名は一例として使用しているだけです。実際にシステム間のトラフィックを保護する場合には、各自のシステムの名前とアドレスを使用してください。

---

4. システムごとに、次のように指定してファイルが有効であるかどうかをチェックします。

```
# /usr/lib/inet/in.iked -c -f /etc/inet/ike/config
```

5. ランダム鍵を生成します。

   Solaris システムでは、od コマンドを使用できます。たとえば、次のコマンドを入力すると、16 進数の数値が 2 行に渡って表示されます。

```
# od -X -A n /dev/random | head -2
         f47cb0f4 32e14480 951095f8 2b735ba8
         0a9467d0 8f92c880 68b6a40e 0efe067d
```

   コマンドの説明については、44 ページの「乱数を生成する方法」と od(1) のマニュアルページを参照してください。

6. システムごとに **/etc/inet/secret/ike.preshared** ファイルを作成します。各ファイルに事前共有鍵を書き込みます。

   この例の認証アルゴリズムは MD5 です (手順 3 を参照)。事前共有鍵として推奨する最小のサイズは、ハッシュのサイズ (つまり、認証アルゴリズムの出力のサイズ) で決まります。MD5 アルゴリズムの出力は 128 ビットすなわち 32 文字です。鍵の長さは長いほうがよいので、この例の鍵の長さは 56 文字にします。

   a. たとえば、**enigma** システムの **ike.preshared** は次のようになります。

```
# ike.preshared on enigma, 192.168.116.16
#...
{ localidtype IP
    localid 192.168.116.16
    remoteidtype IP
    remoteid 192.168.13.213
    # enigma and partym's shared key in hex (192 bits)
    key f47cb0f432e14480951095f82b735ba80a9467d08f92c88068b6a40e
    }
```

   b. **partym** システムの **ike.preshared** は次のようになります。

```
# ike.preshared on partym, 192.168.13.213
#...
{ localidtype IP
    localid 192.168.13.213
    remoteidtype IP
    remoteid 192.168.116.16
    # partym and enigma's shared key in hex (192 bits)
```

```
key f47cb0f432e14480951095f82b735ba80a9467d08f92c88068b6a40e
}
```

注 – 事前共有鍵は同一にする必要があります。

これで、IKE は IPsec で使用できるように構成されました。

例 — 事前共有鍵が同一であるか検査する

通信する各システムの事前共有鍵が同一でない場合は、次のエラーメッセージが表示されます。

```
# rup system2
system2: RPC: Rpcbind failure
```

事前共有鍵を表示するためには、`in.iked` デーモンが特権レベル 0x2 で動作していなければなりません。システムごとに、`ikeadm` コマンドを次のように実行して、事前共有鍵の情報をダンプします。

```
# /usr/sbin/ikeadm get priv
Current privilege level is 0x2, access to keying material enabled
# ikeadm dump preshared
PSKEY: Pre-shared key (24 bytes): f47cb.../192
LOCIP: AF_INET: port 0, 192.168.116.16 (enigma).
REMIP: AF_INET: port 0, 192.168.13.213 (partym).
```

両方のダンプを比較します。両者の事前共有鍵が同じでない場合は、`/etc/inet/secret/ike.preshared` ファイルで、一方の鍵を他方の鍵で置き換えます。

▼ 既存の事前共有鍵を更新する方法

この手順では、リブートすることなく、一定の間隔で既存の事前共有鍵を置き換えたい場合を想定しています。3DES や Blowfish などの強力な暗号化アルゴリズムを使用するときは、両方のシステムのリブート時に鍵を変更するようスケジュールしたほうがよい場合もあります。

1. システムコンソールから、スーパーユーザーになるか、同等の役割を引き受けます。

注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

---

2. ランダム鍵を生成して、それらのいずれか **1** つを選択します。

Solaris システムでは、od コマンドを使用できます。たとえば、次のコマンドを入力すると、16 進数の数値が 2 行に渡って表示されます。

```
# od -X -A n  /dev/random | head -2
         03efe016 216e60ac e316f663 a2f073e0
         7f90d069 316d99b5 00f8384c 2142610a
```

コマンドの説明については、44 ページの「乱数を生成する方法」と od(1) のマニュアルページを参照してください。

3. システムごとに **/etc/inet/secret/ike.preshared** ファイルを編集して、現在の鍵を新しい鍵に変更します。

   たとえば、ホスト enigma と partym で、key の値をそれと同じ長さの新しい数値で置き換えます。

4. **in.iked** デーモンがキー情報の変更を許可するかどうか確認します。

   ```
   # /usr/sbin/ikeadm get priv
   Current privilege level is 0x2, access to keying material enabled
   ```

   コマンドから 0x1 または 0x2 の権限レベルが戻された場合には、キー情報を変更できます。レベル 0x0 の場合には、キー情報を操作できません。デフォルトでは、in.iked デーモンは 0x0 の権限レベルで実行されます。

5. **in.iked** デーモンがキー情報の変更を許可する場合は、**ike.preshared** ファイルの新しいバージョンを読み込みます。

   たとえば、次のように指定します。

   ```
   # ikeadm read preshared
   ```

6. **in.iked** デーモンがキー情報の変更を許可しない場合は、デーモンを強制終了してから再起動します。

   デーモンは起動時に ike.preshared ファイルの新しいバージョンを読み込みます。

   たとえば、次のように指定します。

   ```
   # pkill in.iked
   # /usr/lib/inet/in.iked
   ```

▼ 新しい事前共有鍵を追加する方法

事前共有鍵を使用する場合は、ipsecinit.conf ファイルのポリシーエントリごとに 1 つの事前共有鍵が必要です。IPsec と IKE が動作している間に新しいポリシーエントリを追加すれば、in.iked デーモンはそれらの新しいキーを読み込むことができます。この手順では、次の条件がすでにそろってるものとします。

- in.iked デーモンが動作している。
- IPsec を使って保護したいインタフェースが、次のように、両システムの /etc/hosts ファイルのエントリとして存在する。

  ```
  192.168.15.7 ada
  ```

- 両システムの /etc/inet/ipsecinit.conf ファイルに新しいポリシーエントリが追加されている。たとえば、enigma のエントリが次のようになっている。

  ```
  {laddr enigma raddr ada} ipsec {auth_algs any encr_algs any sa shared}
  ```

  たとえば、ada のエントリが次のようになっている。

  ```
  {laddr ada raddr enigma} ipsec {auth_algs any encr_algs any sa shared}
  ```

- 両システムの /etc/inet/ike/config ファイルの ada 上にそのインタフェースの規則が作成されている。たとえば、enigma の規則が次のようになっている。

  ```
  ### ike/config file on enigma, 192.168.116.16
  ...
  ## The rule to communicate with ada
  { label "Enigma-ada"
    local_addr 192.168.116.16
    remote_addr 192.168.15.7
    p1_xform
    { auth_method preshared oakley_group 5 auth_alg md5 encr_alg blowfish }
    p2_pfs 5
      }
  ```

  たとえば、ada の規則が次のようになっている。

  ```
  ### ike/config file on ada, 192.168.15.7
  ...
  ## The rule to communicate with enigma

  { label "ada-to-Enigma"
    local_addr 192.168.15.7
    remote_addr 192.168.116.16
    p1_xform
    { auth_method preshared oakley_group 5 auth_alg md5 encr_alg blowfish }
    p2_pfs 5
  }
  ```

  注 – auth_method のすべての引数は同じ行になければなりません。

1. システムコンソールから、スーパーユーザーになるか、同等の役割を引き受けます。

   注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

2. **in.iked** デーモンがキー情報の変更を許可するかどうか確認します。

   ```
   # /usr/sbin/ikeadm get priv
   Current privilege level is 0x2, access to keying material enabled
   ```

コマンドから 0x1 または 0x2 の権限レベルが戻された場合には、キー情報を変更できます。レベル 0x0 の場合には、キー情報を操作できません。デフォルトでは、in.iked デーモンは 0x0 の権限レベルで実行されます。

3. **in.iked** デーモンがキー情報の変更を許可しない場合は、デーモンを強制終了します。デーモンを強制終了してから、正しい権限レベルでデーモンを再起動します。

   たとえば、次のように指定します。

```
# pkill in.iked
# /usr/lib/inet/in.iked -p 2
Setting privilege level to 2!
```

4. ランダム鍵を生成し、その出力を結合して **64** から **448** ビットの鍵を作成します。

   Solaris システムでは、od コマンドを使用できます。

```
# od -X -A n /dev/random | head -4
        0fb834c5 8d1fb4ee 500e2bea 071deb2e
        781cb483 74411af5 a9671714 672bb174
        9ad9364d 53574f27 4aacea56 c34861bb
        b4509514 145c1845 f857ff2b 6e5e3766
```

   コマンドの説明については、44 ページの「乱数を生成する方法」と od(1) のマニュアルページを参照してください。

5. このキーを何らかの方法で通信するシステムの管理者に送信します。

   両者は、同じ事前共有鍵を同時に追加する必要があります。

6. **ikeadm** コマンドモードの **add preshared** サブコマンドを使って新しいキー情報を追加します。

```
ikeadm> add preshared { localidtype id-type localid id
remoteidtype id-type remoteid id ike_mode mode key key }
```

id-type	*id* のタイプ
id	IP アドレス (*id-type* が IP の場合)
mode	IKE モード。有効な値は main だけ
key	事前共有鍵 (16 進数形式)

   たとえば、ホスト enigma で新しいインタフェース ada 用のキー 192.168.15.7 を追加します。

```
# ikeadm
ikeadm> add preshared { localidtype ip localid 192.168.116.16
remoteidtype ip remoteid 192.168.15.7 ike_mode main
key 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d }
ikeadm: Successfully created new preshared key.
```

   ホスト ada でも、次のようにして同じキーを追加します。

```
# ikeadm
ikeadm> add preshared { localidtype ip localid 192.168.15.7
remoteidtype ip remoteid 192.168.116.16 ike_mode main
key 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d }
ikeadm: Successfully created new preshared key.
```

注 – Error: invalid preshared key definition というメッセージは、add preshared コマンドの引数が正しくないことを示しています。パラメータに入力ミスがあるかもしれません。パラメータを指定していない可能性もあります。コマンドを正確に入力し直して鍵を追加してください。

---

7. **ikeadm** コマンドモードを終了します。

```
ikeadm> exit
#
```

8. システムごとに、**in.iked** デーモンの権限レベルを低くします。

```
# ikeadm set priv base
```

9. システムごとに、**ipsecinit.conf** ファイルを有効にして、追加したインタフェースを保護します。

```
# ipsecconf -a /etc/inet/ipsecinit.conf
```

注 – このコマンドの実行時には警告を読んでください。ソケットがすでにラッチされている (使用されている) 場合には、システムへ侵入される恐れがあります。

---

10. システムごとに、**ikeadm** コマンドを実行して新しい規則を読み込みます。

    ada と enigma の新しい規則の例がこの手順の始めにあります。規則は /etc/inet/ike/config ファイルに格納されているため、ファイルの名前を指定する必要はありません。

```
# ikeadm read rules
```

11. **IKE** 事前共有鍵がリブート時に確実に使用できるように、**/etc/inet/secret/ike.preshared** ファイルを編集します。

    システムごとに、add preshared コマンドに対する引数をファイルに入力します。

    a. たとえば、**enigma** システムで、次のキー情報を **ike.preshared** ファイルに追加します。

```
# ike.preshared on enigma for the ada interface
#...
{ localidtype IP
  localid 192.168.116.16
  remoteidtype IP
  remoteid 192.168.15.7
```

```
  # enigma and ada's shared key in hex (32 - 448 bits required)
  key 04413a3e68854b732742024d19995f7972136a2f33e5d302bdd7b2624e4c6429
    }
```

b. ada システムで、次のキー情報を **ike.preshared** ファイルに追加します。

```
# ike.preshared for the ada interface, 192.168.15.7
#...
{ localidtype IP
  localid 192.168.15.7
  remoteidtype IP
  remoteid 192.168.116.16
  # ada and enigma's shared key in hex (32 - 448 bits required)
  key 04413a3e68854b732742024d19995f7972136a2f33e5d302bdd7b2624e4c6429
    }
```

▼ 自己署名付き公開証明書による IKE の設定方法

1. システムコンソールから、スーパーユーザーになるか、同等の役割を引き受けます。

注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

---

2. **ikecert certlocal -ks** コマンドを使用して、自己署名付き証明書を **ike.privatekeys** データベースに追加します。

```
# ikecert certlocal -ks -m keysize -t keytype \
-D dname -A altname[ ... ] [-f output]
```

`-ks`	自己署名付き証明書を作成する
keysize	キーのサイズ。*keysize* は、512、1024、2048、3072、4096 のいずれか
keytype	使用するアルゴリズムのタイプ。*keytype* は、rsa-sha1、rsa-md5、dsa-sha1 のいずれか
dname	証明書主体の X.509 識別名。*dname* の一般的な形式は次のとおり : C = country (国)、O = organization (組織)、OU = organizational unit (組織単位)、CN = common name (共通名)。有効なタグは、`C`、`O`、`OU`、`CN`
altname	証明書の代替名。*altname* の形式は `tag=value` でなければならない。有効なタグは、`IP`、`DNS`、`EMAIL`、`URI`、`DN`、`RID`
output	エンコーディング出力の形式。指定できる値は `pem` (PEM Base64) と `ber` (ASN.1 BER)。`-f` を指定しないと、 `pem` が使用される

たとえば、次のように指定します。

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
> -D "C=US, O=ExampleCompany, OU=US-Example, CN=Example" \
> -A IP=192.168.116.16
Generating, please wait...
Certificate:
Certificate generated.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

3. その証明書を、通信するシステムの管理者に送信します。
   その証明書は、次のようにして電子メールにカット&ペーストできます。

```
To: root@us.example.com
From: root@un.example.com
Message: -----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

4. 送信側のシステムで、**/etc/inet/ike/config** ファイルを編集して、通信するシステムからの公開鍵を認識します。たとえば、次のように指定します。

```
# Explicitly trust the following self-signed certs
# Use the Subject Alternate Name to identify the cert

cert_trust "192.168.116.16"
cert_trust "192.168.13.213"

## Parameters that may also show up in rules.

p1_xform
  { auth_method preshared oakley_group 5 auth_alg sha encr_alg des }
p2_pfs 5

{
 label "UN-Example to US-Example"
 local_id_type dn
 local_id "C=US, O=ExampleCompany, OU=UN-Example, CN=Example"
 remote_id "C=US, O=ExampleCompany, OU=US-Example, CN=Example"

 local_addr 192.168.116.16
 remote_addr 192.168.13.213

 p1_xform
  { auth_method rsa_encrypt oakley_group 2 auth_alg md5 encr_alg 3des }
}
```

5. 次の手順を実行して、通信するシステムの公開鍵を追加します。

   a. 管理者の電子メールから公開鍵をコピーします。

b. 次のように **ikecert certdb -a** コマンドと **<Return>** を入力します。
<Return> キーを押してもプロンプトは表示されません。

```
# ikecert certdb -a <Return キーを押す>
```

c. 公開鍵を貼り付けます。次に **<Return>** を入力します。

```
-----BEGIN X509 CERTIFICATE-----
MIICL...
...
KgDid/nxWPlWQU5vMAiwJXfa0sw/A12w448JVkVmEWaf
-----END X509 CERTIFICATE-----<Return キーを押す>
```

d. **<Control-D>** を入力して入力を終了します。

```
<Control-D>
```

6. 通信するシステムの管理者と一緒に、キーが改ざんされていないことを確認します。
たとえば、その管理者に電話で連絡して公開鍵ハッシュの値を比較できます。一方のシステムの公開鍵ハッシュは、通信するシステムの公開鍵ハッシュと同一でなければなりません。

a. たとえば、**enigma** で **ikecert certdb -l** コマンドを実行します。

```
enigma # ikecert certdb -l
        Certificate Slot Name: 0   Type: if-modn
        Subject Name: <C=US, O=ExampleCo, OU=UN-Example, CN=Example>
        Key Size: 1024
        Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

b. **partym** で **ikecert certlocal -l** コマンドを実行します。

```
partym # ikecert certlocal -l
Local ID Slot Name: 1   Type: if-modn
        Key Size: 1024
        Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

注 – この例の公開鍵ハッシュは、使用しているシステムで生成される公開鍵ハッシュとは異なります。

▼ 認証局による署名付き公開鍵による IKE の設定方法

1. システムコンソールから、スーパーユーザーになるか、同等の役割を引き受けます。

注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

2. **ikecert certlocal -kc** コマンドを実行して証明書要求を作成します。

   このコマンドに対する引数は、 ikecert certlocal -ks コマンドで使用する引数と同じでなければなりません。

```
# ikecert certlocal -kc -m keysize -t keytype \
-D dname -A altname[ ... ] [-f output]
```

-kc	公開鍵と非公開鍵のペアを作成する。さらに、-kc は、生成された公開鍵に基づいて証明書要求を作成する
keysize	キーのサイズ。 *keysize* は、512、1024、2048、3072、4096 のいずれか
keytype	使用するアルゴリズムのタイプ。*keytype* は、rsa-sha1、rsa-md5、dsa-sha1 のいずれか
dname	証明書主体の X.509 識別名。*dname* の一般的な形式は次のとおり : C = country (国)、O = organization (組織)、OU = organizational unit (組織単位)、CN = common name (共通名)。有効なタグは、C、O、OU、CN
altname	証明書の代替名。*altname* の形式は tag=value でなければならない。有効なタグは、IP、DNS、EMAIL、URI、DN、RID
output	エンコーディング出力の形式。指定できる値は pem (PEM Base64) と ber (ASN.1 BER)。-f を指定しないと、 pem が使用される

   たとえば、次のコマンドを実行して証明書要求を作成します。

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=US, O=ExampleCompany\, Inc., OU=US-Example, CN=Example" \
> -A "DN=C=US, O=ExampleCompany\, Inc., OU=US-Example"
Generating, please wait...
Certificate request generated.
-----BEGIN CERTIFICATE REQUEST-----
MIIByjCCATMCAQAwUzELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFEV4YW1wbGVDb21w
...
lcM+tw0ThRrfuJX9t/Qa1R/KxRlMA3zckO80mO9X
-----END CERTIFICATE REQUEST-----
```

3. この証明書要求を、**PKI** を処理する機関に送信します。

   この機関は、外部の認証局や PKI の場合があります。また、会社によっては PKI を独自に運営する場合もあります。証明書要求の送信方法については PKI に問い合わせてください。ほとんどの機関は、Web サイトに送信フォームを掲載しています。フォームの記入に当たっては、その送信が正当なものであることを証明する必要があります。通常は、証明書要求をフォームに貼り付けます。要求を受け取った機関は、それ

をチェックしてから 2 つまたは 3 つの証明書オブジェクトを発行します。

- 公開鍵証明書 – この証明書は機関に送信した要求に基づいて作成される。送信した証明書要求も、公開鍵証明書の一部として含まれる。この証明書によって一意に識別される
- 認証局 – 機関の署名。CA によって公開鍵証明書が正規のものであることが確認される
- 証明書無効リスト – 機関が無効にした証明書の最新リスト。CRL へのアクセスが公開鍵証明書に組み込まれている場合には、CRL が別個の証明書オブジェクトとして送信されることはない

  CRL の URI が公開鍵証明書に組み込まれている場合には、IKE は CRL を自動的に取り出すことができる。同様に、DN エントリが公開鍵証明書に組み込まれている場合には、IKE は、指定された LDAP サーバーから CRL を自動的に取り出すことができる

  公開鍵証明書に URI や DN エントリを組み込んだ例については、77 ページの「証明書無効リストにアクセスする方法」を参照

4. 各証明書を **3** つの **ikecert** コマンドのどれかに対する引数として指定します。

   これらのコマンドでは `-a` オプションを使用します。`-a` オプションを指定すると、貼り付けたオブジェクトが、システム内の適切な証明書データベースに追加されます。詳細は、53 ページの「公開鍵証明書の使用」を参照してください。

   a. システムコンソールから、スーパーユーザーになるか、同等の役割を引き受けます。

   b. 次のように **ikecert certdb -a** コマンドと **<Return>** を入力します。

      ```
      # ikecert certdb -a <Return キーを押す>
      ```

   c. 機関から受け取った公開鍵証明書を貼り付け、**<Return>** を入力します。

      ```
      -----BEGIN X509 CERTIFICATE-----
      ...
      -----END X509 CERTIFICATE----<Return キーを押す>
      ```

   d. **<Control-D>** を入力して入力を終了します。

      ```
      <Control-D>
      ```

   e. 次のように **ikecert certdb -a** コマンドと **<Return>** を入力します。

      ```
      # ikecert certdb -a <Return キーを押す>
      ```

   f. 機関の **CA** を貼り付けます。**<Return>** を入力します。次に **<Control-D>** を入力して入力を終了します。

      ```
      -----BEGIN X509 CERTIFICATE-----
      ...
      -----END X509 CERTIFICATE-----<Return キーを押す>
      <Control-D>
      ```

g. 機関から **CRL** オブジェクトを受け取っている場合は、**ikecert certrldb -a** コマンドと **<Return>** を入力します。

```
# ikecert certrldb -a <Return キーを押す>
```

h. 機関の **CRL** を貼り付けます。**<Return>** を入力します。次に **<Control-D>** を入力して入力を終了します。

5. **/etc/inet/ike/config** ファイルを編集して、機関を認識します。

機関から利用するように通知された名前を使用します。たとえば、次のように指定します。

```
# Trusted root cert
# This certificate is from Example PKI
# This is the X.509 distinguished name for the CA that it issues.

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"

## Parameters that may also show up in rules.

p1_xform { auth_method rsa_sig oakley_group 1 auth_alg sha1 encr_alg des }

p2_pfs 2

{
 label "US-Example to UN-Example - Example PKI"
 local_id_type dn
 local_id  "C=US, O=ExampleCompany, OU=US-Example, CN=Example"
 remote_id "C=US, O=ExampleCompany, OU=UN-Example, CN=Example"

 local_addr 192.168.116.16
 remote_addr 192.168.13.213

 p1_xform
  { auth_method rsa_encrypt oakley_group 2  auth_alg md5  encr_alg 3des }
}
```

注 – auth_method のすべての引数は同じ行になければなりません。

---

6. **PKI** 機関から **CRL** を受け取っていない場合は、キーワード **ignore_crls** を **ike/config** ファイルに追加します。

ignore_crls を指定すると、IKE は CRL を検索しません。たとえば、次のように指定します。

```
# Trusted root cert
...
cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"
ignore_crls

...
```

7. **PKI** 機関から **CRL** の一元的なディストリビューションポイントを知らされている場合は、**ike/config** ファイルを変更してこの場所を指定することができます。

   例については、77 ページの「証明書無効リストにアクセスする方法」 を参照してください。

8. 通信するシステムで手順 **1** から手順 **7** を実行します。

   この例に従うなら、前に行ったように、"…OU=UN-Example…" システムで ikecert コマンドを実行します。"…OU=UN-Example…" システムの /etc/inet/ike/config ファイルでは、そのローカル情報をローカルパラメータとして使用します。さらに、システムでは、システムの情報をリモートパラメータとして使用します。

   たとえば、次のように指定します。

```
# Trusted root cert
# This certificate is from Example PKI

cert_root "C=US, O=ExamplePKI\, Inc., OU=PKI-Example, CN=Example PKI"
ignore_crls

## Parameters that may also show up in rules.

p1_xform { auth_method rsa_sig oakley_group 1 auth_alg sha1 encr_alg des }
p2_pfs 2

{
 label "UN-Example to US-Example - Example PKI"
 local_id_type dn
 local_id "C=US, O=ExampleCompany, OU=UN-Example, CN=Example"
 remote_id "C=US, O=ExampleCompany, OU=US-Example, CN=Example"

 local_addr 192.168.13.213
 remote_addr 192.168.116.16

 p1_xform
  { auth_method rsa_encrypt oakley_group 2  auth_alg md5  encr_alg 3des }
}
```

9. **auth_method** は **rsa_encrypt** であるため、ピアの証明書を公開鍵データベースに追加します。

注 – 以下のサブステップは、/etc/inet/ike/config ファイルの p1_xform で rsa_encrypt 認証方式を使用する場合にのみ必要です。

---

   a. その証明書を、通信するシステムの管理者に送信します。

      その証明書は、次のようにして電子メールにカット&ペーストできます。

```
To: root@un.example.com
From: root@us.example.com
Message: -----BEGIN CERTIFICATE REQUEST-----
```

```
MIIByjCCATMCAQAwUzELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFEV4YW1wbGVDb21w
...
lcM+tw0ThRrfuJX9t/Qa1R/KxRlMA3zckO80mO9X
-----END CERTIFICATE REQUEST-----
```

b. システムごとに、ピアの証明書をローカルの公開鍵データベースに追加します。

たとえば、`un.example.com` システムで次のように入力してから、電子メールの内容を貼り付けます。

```
# ikecert certdb -a <Type the Return key>
-----BEGIN CERTIFICATE REQUEST-----
MIIByjCCATMCAQAwUzELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFEV4YW1wbGVDb21w
...
lcM+tw0ThRrfuJX9t/Qa1R/KxRlMA3zckO80mO9X
-----END CERTIFICATE REQUEST-----
```

RSA 暗号化認証方式を使用すると、IKE の ID が盗聴者から保護されます。`rsa_encrypt` 方式では ID が隠されるため、IKE はピアを知りません。そのため、ピアの証明書を取り出すことはできません。したがって、その方式では、IKE ピアが互いの公開鍵を認識することが必要になります。よって、`/etc/inet/ike/config` ファイルの `auth_method rsa_encrypt` を使用する場合には、ピアの証明書を公開鍵データベースに追加する必要があります。この結果、公開鍵データベースには、通信するシステムペアごとに 3 つの証明書が存在することになります。公開鍵データベースには、公開鍵証明書、CA 証明書、およびピアの証明書が存在します。これに対して CRL データベースには、無効化された証明書が格納されています。

IKE デーモンは、次の処理が完了すると、公開鍵と CA を使って自身を認証します。

a. 各システムの `/etc/hosts` ファイルに、保護されたインタフェースが追加されている

b. 各システムの `/etc/inet/ipsecinit.conf` ファイルに、保護されたインタフェースが追加されている

c. 両方のシステムがリブートされている

▼ 証明書無効リストにアクセスする方法

証明書無効リスト (CRL) には、認証局が発行した証明書のうち、期限切れになったりセキュリティが低下したりした証明書が記載されます。CRL を処理する方法には、次の 4 つがあります。

- CA 機関から CRL が発行されない場合は、`/etc/inet/ike/config` ファイルにある CRL を無視するように IKE に指定できる。このオプションについては、72 ページの「認証局による署名付き公開鍵による IKE の設定方法」を参照
- CA から受け取った公開鍵証明書に URI のアドレスが組み込まれている場合は、IKE は URI から CRL にアクセスする
- CA から受け取った公開鍵証明書に LDAP サーバーの DN エントリが組み込まれている場合は、IKE は LDAP サーバーから CRL にアクセスする。その場合には、`/etc/inet/ike/config` ファイルの `ldap-list` キーワードに対する引数として LDAP サーバーを指定する

- `ikecert certrldb` コマンドの引数として CRL を指定する

次の手順は、一元的なディトリビューションポイントの CRL を使用するように IKE に指示する方法を示しています。

1. **`ikecert certdb -lv`** *certspec* コマンドを実行して、**PKI** 機関から受け取った証明書を表示します。

`-l`	IKE 証明書データベースにある証明書を一覧表示する
`-v`	証明書を冗長モードで一覧表示する。このオプションは慎重に使用すること
certspec	*certspec* パターンと証明書のパターンを照合する

たとえば、次の証明書は Sun Microsystems から発行されたものです。詳細は変更されています。

```
# ikecert certdb -lv example-protect.sun.com
Certificate Slot Name: 0   Type: if-modn
   (Private key in certlocal slot 0)
 Subject Name: <O=Sun Microsystems Inc, CN=example-protect.sun.com>
 Issuer Name: <CN=Sun Microsystems Inc CA (Class B), O=Sun Microsystems Inc>
 SerialNumber: 14000D93
   Validity:
      Not Valid Before: 2002 Jul 19th, 21:11:11 GMT
      Not Valid After:  2005 Jul 18th, 21:11:11 GMT
   Public Key Info:
      Public Modulus  (n) (2048 bits): C575A...A5
      Public Exponent (e) (  24 bits): 010001
   Extensions:
      Subject Alternative Names:
              DNS = example-protect.sun.com
      Key Usage: DigitalSignature KeyEncipherment
      [CRITICAL]
   CRL Distribution Points:
      Full Name:
         URI = #Ihttp://www.sun.com/pki/pkismica.crl#i
         DN = <CN=Sun Microsystems Inc CA (Class B), O=Sun Microsystems Inc>
      CRL Issuer:
      Authority Key ID:
      Key ID:              4F ... 6B
      SubjectKeyID:        A5 ... FD
      Certificate Policies
      Authority Information Access
```

`CRL Distribution Points` のデータに注目してください。URI エントリは、この機関の CRL が Web 上にあることを示しています。DN エントリは、CRL が LDAP サーバー上にもあることを示しています。ユーザーはこれら 2 つのうちのどちらかを使用できます。

2. **URI** を使用する場合は、ホストの **`/etc/inet/ike/config`** ファイルにキーワード **`use_http`** を指定します。

たとえば、ike/config ファイルの内容は次のようになります。

```
# Use CRL from organization's URI
use_http
...
```

IKE は CRL を取り出し、証明書の期限が切れるまで CRL を保持します。

さらに、ike/config ファイルにキーワード proxy を指定することによって Web プロキシを使用することもできます。proxy では、次のように、引数として URL を指定します。

```
proxy "http://proxy1:8080"
```

3. **LDAP** を使用する場合は、ホストの **/etc/inet/ike/config** ファイルの **ldap-list** キーワードへの引数として **LDAP** サーバーを指定します。

   LDAP サーバーの名前は、使用する機関にたずねてください。ike/config ファイルのエントリは、次のようになります。

```
# Use CRL from organization's LDAP
ldap-list "ldap1.sun.com:389,ldap2.sun.com"
...
```

   IKE は CRL を取り出し、証明書の期限が切れるまで CRL を保持します。

例 — CRL をローカルの certrldb データベースに貼り付ける

この例は、一元的なディストリビューションポイントから得ることができない CRL を使用する方法を示しています。

使用する機関の証明書に一元的なディストリビューションポイントが含まれていない場合は、機関の CRL を手動でローカルの crls データベースに追加できます。その場合は、機関の説明に従って CRL を抽出し、それを ikecert certrldb -a コマンドでデータベースに追加します。

```
# ikecert certrldb -a<Return キーを押す>
<PKI 機関からの CRL を貼り付ける>

<Return キーを押す>
<<Control-D> を押して CRL をデータベースに追加する>
```

▼ IKE で Sun Crypto Accelerator 1000 カードを使用する方法

注 – 次の手順では、Sun Crypto Accelerator 1000 カードがすでにシステムに取り付けられているものとします。さらに、カードに必要なソフトウェアがすでにインストールされ、構成されているものとします。詳細については、『*Sun Crypto Accelerator 1000 Board Version 1.1 Installation and User's Guide*』を参照してください。

---

1. システムコンソールから、スーパーユーザーになるか、同等の役割を引き受けます。

注 – リモートログインすると、セキュリティ上重要なトラフィックが盗聴される恐れがあります。何らかの方法でリモートログインを保護していても、システム全体のセキュリティがリモートログインセッションレベルに低下します。

---

2. **PKCS #11** ライブラリのパスを **/etc/inet/ike/config** ファイルに追加します。

```
pkcs11_path "/opt/SUNWconn/lib/libpkcs11.so"
```

パス名は 32 ビット PKCS #11 ライブラリを指していなければなりません。ライブラリが存在していれば、IKE は、ライブラリのルーチンを使用して、Sun Crypto 1000 カード上の IKE 公開鍵操作を高速化します。カードがこのような重い操作を行っている間、オペレーティングシステムのリソースは他の操作に使用できます。

3. ファイルを閉じてからリブートします。

4. リブートしたら、ライブラリがリンクされていることを確認します。**PKCS #11** ライブラリがリンクされていることを確認するには、次のコマンドを実行します。

```
# ikeadm get stats
Phase 1 SA counts:
Current:   initiator:          0   responder:          0

Total:     initiator:          0   responder:          0
Attempted: initiator:          0   responder:          0
Failed:    initiator:          0   responder:          0
           initiator fails include 0 time-out(s)
PKCS#11 library linked in from /opt/SUNWconn/lib/libpkcs11.so
#
```

`/etc/inet/ike/config` ファイルの他のパラメータとは異なり、`pkcs11_path` キーワードは IKE の起動時にだけ読み込まれます。`ikeadm` コマンドを使って新しい `/etc/inet/ike/config` ファイルを追加したり再読み込みしたりしても、`pkcs11_path` は持続します。パスが持続するのは、IKE デーモンがフェーズ 1 のデータを保持するためです。PKCS #11 によって処理が加速されたキーは、フェーズ 1 データの一部です。

用語集

この用語集は、ネットワークセキュリティの用語を定義します。

AES Advanced Encryption Standard。対称 128 ビットブロックのデータ暗号技術。米国政府は、2000 年の 10 月に暗号化標準として Rijndael 方式を採用した。DES に代わる米国政府の標準として、AES が採用されている。

Blowfish 32 ビットから 448 ビットまでの可変長キーの対称ブロックの暗号化アルゴリズム。その作成者である Bruce Schneier 氏は、鍵を頻繁に変更しないアプリケーションに効果的であると述べている。

DES Data Encryption Standard。1975 年に開発され、1981 年に ANSI X.3.92 として ANSI であ化された対称鍵の暗号化方式。DES では 56 ビットの鍵を使用する。

DSA デジタル署名アルゴリズム。512 ビットから 4096 ビットまでの可変長キーの公開鍵アルゴリズム。米国政府標準である DSS は最大 1024 ビットである。DSA は入力に SHA-1 を使用する。

Diffie-Hellman プロトコル 公開鍵暗号化としても知られている。1976 年に Diffie 氏と Hellman 氏が開発した非対称暗号鍵協定プロトコル。このプロトコルを使用して、セキュリティ保護されていない媒体で事前に秘密鍵を用意しなくても 2 人のユーザーが秘密鍵を交換できます。Diffie-Hellman は、IKE プロトコルで使用される。

HMAC メッセージ認証を行うためのキー付きハッシュ方法。HMAC は秘密共有鍵と併用して、MD5 、SHA-1 などの繰り返し暗号化のハッシュ関数で使用する。HMAC の暗号の強さは、基底ハッシュ関数のプロパティによって異なる。

IKE インターネットキー交換。IKE は、IPsec セキュリティアソシエーションの認証されたキー情報を自動的に提供する。

IPsec IP データグラムを保護するためのセキュリティアーキテクチャ。

IPv4 インターネットプロトコルバージョン 4。IP とも呼ばれる。このバージョンは 32 ビットのアドレス空間を提供する。

IPv6	インターネットプロトコルバージョン 6。このバージョンは 128 ビットのアドレス空間を提供する。
IP 内 **IP** カプセル化	IP パケット内で IP パケットをトンネリングするための機構。
MD5	デジタル署名などのメッセージ認証に使用する繰り返し暗号化のハッシュ関数。1991 年に Rivest 氏によって開発された。
PKI	Public Key Infrastructure。インターネットトランザクションに関係する各関係者の有効性を確認および承認する、デジタル署名、認証局、他の登録機関のシステム。
RSA	デジタル署名と公開鍵暗号化システムを取得するための方法。1978 年に最初に公開され、 Rivest氏、Shamir 氏、Adleman 氏によって開発された。
SADB	セキュリティアソシエーションデータベース。暗号化鍵と暗号化アルゴリズムを指定するテーブル。鍵とアルゴリズムは、安全なデータ転送で使用される。
SHA-1	セキュリティ保護されたハッシュアルゴリズム。メッセージ要約を作成するために 2^{64} 文字以下の長さを入力するときに操作する。これは DSA への入力となる。
SPI	セキュリティパラメータインデックス。受信したパケットの暗号解除に受信側が使用する SADB の行を指定する整数。
Triple-DES	Triple-Data Encryption Standard。168 ビットの鍵を提供する対称鍵暗号化方法。
鍵管理	セキュリティアソシエーションを管理するための手法。
仮想プライベートネットワーク **(VPN)**	インターネットのような公共ネットワーク内でトンネルを利用する、単独の、安全で論理的なネットワーク。
カプセル化	ヘッダーとペイロードを 1 番目のパケット内に配置し、そのパケットを 2 番目のパケットのペイロード内に配置すること。
カプセル化セキュリティヘッダー	データグラムに対して認証と完全性を提供する拡張ヘッダー。
公開鍵暗号化	2 つの鍵を使用する暗号化システム。公開鍵はだれでも知ることができる。非公開鍵はメッセージの受信側だけが知っている。IKE により、IPsec の公開鍵が提供される。
セキュリティアソシエーション **(SA)**	1 つのホストから 2 番目のホストに、セキュリティ属性を指定するアソシエーション。
セキュリティパラメータインデックス **(SPI)**	受信したパケットを復号化するために使用する、SADB (セキュリティアソシエーションデータベース) 内の行を特定する整数値。
専用アドレス	インターネット経由で経路指定ができない IP アドレス。
双方向トンネル	双方向にデータグラムを送信するトンネル。

対称鍵暗号化	メッセージの暗号解除に、メッセージの送受信側が単一の共通鍵を使用する暗号化システム。対称鍵は、IPsec での大量データ転送の暗号化に使用する。対称鍵システムの一例として DES がある。
デジタル署名	送信側を一意に識別する、電子的に転送されたメッセージに添付されるデジタルコード。
トンネル	カプセル化される間データグラムが通過するパス。
認証局 **(CA)**	デジタル署名および公開鍵と非公開鍵のペアの作成に使用するデジタル証明書を発行する、公証された第三者機関または企業。CA は、一意の証明書を付与された個人が当該の人物であることを保証する。
認証ヘッダー	IP データグラムに対し認証と完全性を提供する拡張ヘッダー。機密性は提供されない。
ネットワークインタフェースカード **(NIC)**	リンクとのインタフェースになる、内部ネットワークアダプタまたは独立したネットワークアダプタカード。
ノード	ホストまたはルーター。
パケット	通信回線上で、1 単位として送られる情報の集合。ヘッダーとペイロードで構成される。
ハッシュ値	テキストの文字列から生成される数値。ハッシュ関数は、転送されるメッセージが改ざんされないようにするために使用する。1 方向のハッシュ関数の一例としては、MD5 および SHA-1 がある。
非対称鍵暗号化	メッセージの送受信側で異なる鍵を使用してメッセージの暗号化および暗号解除を行う暗号化システム。非対称鍵を使用して、対称鍵暗号に対するセキュリティ保護されたチャネルを作成する。非対称鍵プロトコルの一例には、Diffie–Hellman がある。対称鍵暗号化と対比。
ファイアウォール	組織内の私的ネットワークまたはイントラネットを、インターネットなどの外部ネットワークからの侵入に対して保護する装置またはソフトウェア。
物理インタフェース	リンクに対するノードの接続。この接続は通常、デバイスドライバとネットワークアダプタとして実装される。ネットワークアダプタによっては、qfe のように複数の接続点を持つ場合もある。このマニュアルでは、「ネットワークアダプタ」は「単一接続点」を示す。
マルチキャストアドレス	特定の方法でインタフェースのグループを特定する IP アドレス。マルチキャストアドレスに送信されるパケットは、グループにあるすべてのインタフェースに配信される。

索引