IPsec と IKE の管理

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

Part No: 817–3492–10
2003 年 12 月

原典: *IPsec and IKE Administration Guide*

Part No: 817-2694-10

Revision A

Adobe PostScript

031005@6671

目次

はじめに　**7**

1　IP セキュリティアーキテクチャ **(**概要**)　11**
IPsec とは　11
IPsec セキュリティアソシエーション　15
　キー管理　15
保護機構　16
　認証ヘッダー　16
　セキュリティペイロードのカプセル化　17
　認証アルゴリズムと暗号化アルゴリズム　17
保護ポリシー機構と実施機構　19
トランスポートモードとトンネルモード　19
　信頼性の高いトンネル　21
仮想プライベートネットワーク　21
IPsec ユーティリティおよび IPsec ファイル　22
　IPsec ポリシーコマンド　23
　IPsec ポリシーファイル　24
　IPsec のセキュリティアソシエーションデータベース　25
　キーユーティリティ　26
　その他のユーティリティに対する IPsec 拡張機能　27

2　IPsec の管理 **(**手順**)　29**
IPsec の実装 (作業マップ)　29
IPsec 作業　30
　▼ 2 つのシステム間のトラフィックを保護する方法　31

▼ Web サーバーを保護する方法　33
▼ 仮想プライベートネットワーク (VPN) を構築する方法　35
▼ 乱数を生成する方法　42
▼ IPsec セキュリティアソシエーションを手動で生成する方法　43
▼ パケットが保護されていることを確認する方法　47

3　インターネットキー交換 (概要)　49
IKE の概要　50
フェーズ 1 交換　50
フェーズ 2 交換　51
IKE 構成の選択　51
IKE と事前共有鍵　51
IKE と公開鍵証明書　52
IKE とアクセラレータハードウェア　52
IKE とハードウェアストレージ　53
IKE ユーティリティおよび IKE ファイル　53
IKE デーモン　54
IKE ポリシーファイル　55
IKE 管理コマンド　55
事前共有鍵ファイル　56
IKE 公開鍵のデータベースおよびコマンド　56

4　IKE の管理 (手順)　61
IKE の設定 (作業マップ)　61
事前共有鍵による IKE の設定 (作業マップ)　62
▼ 事前共有鍵による IKE の設定方法　63
▼ 既存の事前共有鍵を更新する方法　65
▼ 新しい事前共有鍵を追加する方法　66
▼ 事前共有鍵が同一であることを確認する方法　70
公開鍵証明書による IKE の設定 (作業マップ)　71
▼ 自己署名付き公開鍵証明書による IKE の設定方法　71
▼ CA からの署名付き証明書による IKE の設定方法　75
▼ ハードウェア上で公開鍵証明書を生成、格納する方法　80
▼ 証明書無効リストを処理する方法　83
IKE とハードウェアの使用 (作業マップ)　85
▼ IKE で Sun Crypto Accelerator 1000 ボードを使用する方法　86
▼ IKE で Sun Crypto Accelerator 4000 ボードを使用する方法　87

A 『**IPsec** と **IKE** の管理』の更新情報 **89**
Solaris 9 4/03 の更新情報 89
Solaris 9 12/03 の更新情報 89

用語集 **91**

索引 **95**

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 オペレーティング環境のインストール
- SunOS 5.9 オペレーティング環境の Solaris 9 12/03 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 オペレーティング環境は 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 オペレーティング環境は、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 オペレーティング環境は、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 オペレーティング環境は、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 (IPsec) 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 で使用するインターネットキー交換 (IKE) 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。

付録 A では、Solaris 9 リリースと Solaris 9 12/03 リリースの相違点を一覧表示します。

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

IP セキュリティアーキテクチャ (概要)

IP セキュリティアーキテクチャ (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。

- 11 ページの「IPsec とは」
- 15 ページの「IPsec セキュリティアソシエーション」
- 16 ページの「保護機構」
- 19 ページの「保護ポリシー機構と実施機構」
- 19 ページの「トランスポートモードとトンネルモード」
- 21 ページの「仮想プライベートネットワーク」
- 22 ページの「IPsec ユーティリティおよび IPsec ファイル」

ネットワークに IPsec を実装する手順については、第 2 章を参照してください。

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様子を示します。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。

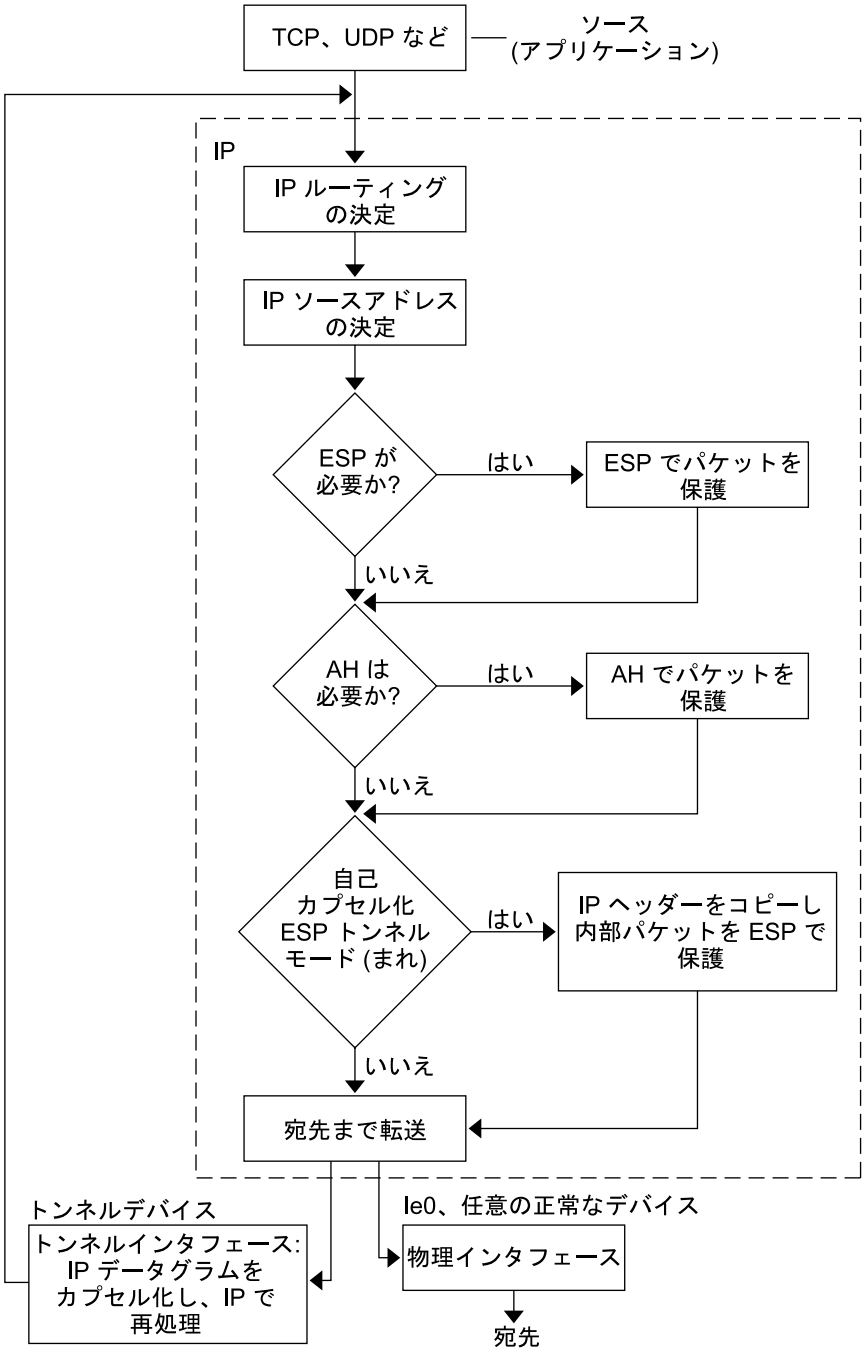


図 1–1 アウトバウンドパケットプロセスに適用された IPsec

図 1–2 は、IPsec インバウンドプロセスを示したものです。

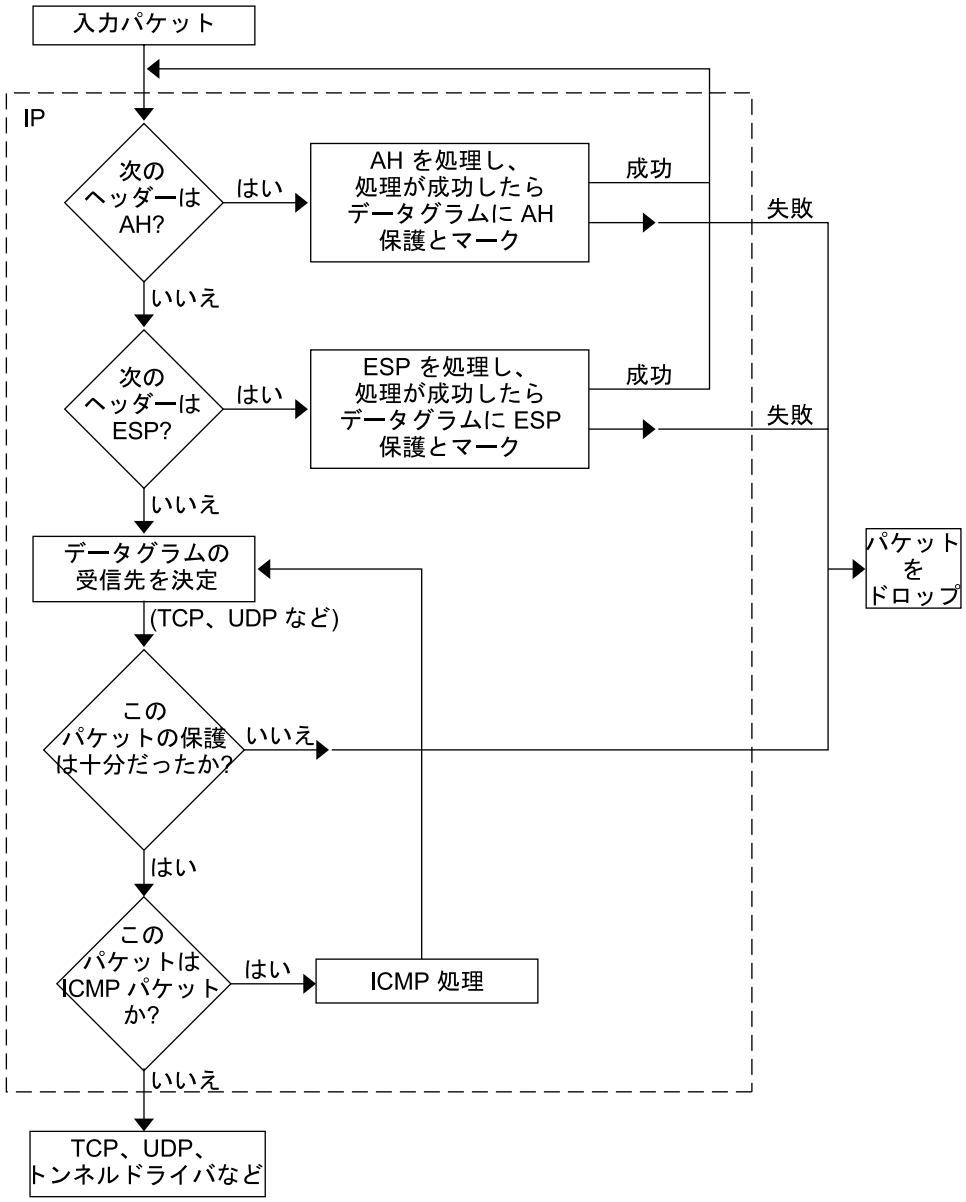


図 **1–2** インバウンドパケットプロセスに適用された IPsec

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 データベースに格納されます。ソケットベースの管理エンジン `pf_key` 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` デーモンにより、自動キー管理が可能になります。`pf_key(7P)` と `in.iked(1M)` のマニュアルページを参照してください。

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` 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。IPv4 および IPv6 パケットの SA は自動キー管理を使用できます。

暗号キーを IKE で自動的に管理する方法については、50 ページの「IKE の概要」を参照してください。`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、26 ページの「キーユーティリティ」を参照してください。コマンドオプションの詳しい説明については、`ipseckey(1M)` のマニュアルページ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、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` コマンドで 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 `authsha1(7M)` のマニュアルページを参照してください。IP 設定パラメータの調整方法については、`ndd(1M)` のマニュアルページを参照してください。

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、セキュリティペイロードのカプセル化 (ESP) ヘッダー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 による実装では、ESP は IP の先頭にプッシュされるモジュールです。`/dev/ipsecesp` エントリでは、`ndd` コマンドで 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。IP 設定パラメータの調整方法については、`ndd(1M)` のマニュアルページを参照してください。

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 オペレーティング環境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 オペレーティング環境でサポートされる暗号化アルゴリズムを示します。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

表 1–2 サポートされる暗号化アルゴリズム　(続き)

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
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

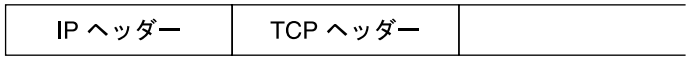
ipsecconf コマンドは、システム規模ポリシーの設定に使用します。ipsecconf(1M) のマニュアルページを参照してください。

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したり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、セキュリティ機能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

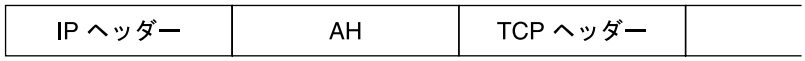
IP ヘッダーの後に、ESP または AH を呼び出してデータグラムを保護するときに、トランスポートモードを使用します。たとえば、パケットが次のヘッダーで始まる場合です。



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` 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。トンネルの設定方法については、`ifconfig(1M)` 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、仮想プライベートネットワーク (VPN) 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。

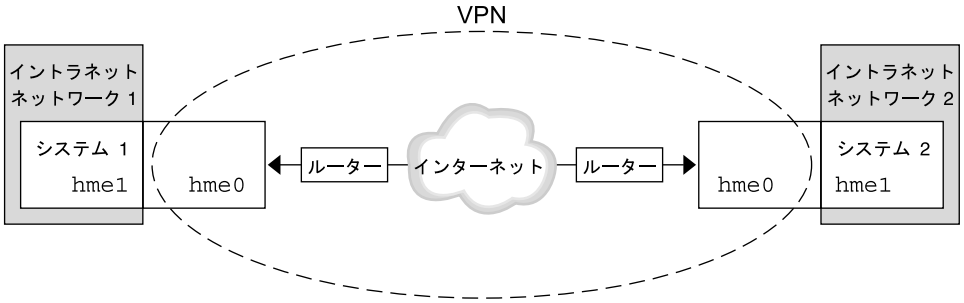


図 **1–3** 仮想プライベートネットワーク

セットアップ手順については、35 ページの「仮想プライベートネットワーク (VPN) 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

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

IPsec ファイルまたは IPsec コマンド	説明
/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 は自動キー管理機能を提供する。/etc/inet/ike/config ファイル内の規則およびグローバルパラメータに基づいて管理が行われる。53 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 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設定すると、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。IPsec ポリシーオプションについては、ipsecconf(1M) のマニュアルページを参照してください。

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 オペレーティング環境を起動したときに 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。

```
#
# For example,
#
#     {rport 23} ipsec {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}
#
# 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.
```

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() または accept() 関数呼び出しが行われた場合、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は、すでにラッチされたソケットを保護しません。connect(3SOCKET) と accept(3SOCKET) のマニュアルページを参照してください。

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 および IPv6 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キーユーティリティには、ipseckey コマンドがあります。ipseckey(1M) のマニュアルページを参照してください。

ipseckey コマンドを使用して、ipsecah と ipsecesp 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。

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 に次のオプションが追加されました。

- auth_algs
- encr_auth_algs
- encr_algs

1 回の呼び出しで、1 つのトンネルにすべての IPsec セキュリティオプションを指定する必要があります。たとえば、ESP だけを使ってトラフィックを保護する場合、次のように両方のセキュリティオプションを指定して、トンネル ip.tun0 を設定します。

```
# ifconfig ip.tun0 ... encr_algs 3DES encr_auth_algs MD5
```

同様に、ipsecinit.conf エントリは、次のように両方のセキュリティオプションを指定して、トンネルを設定します。

```
# WAN トラフィックは 3DES および MD5 とともに ESP を使用します。
{} ipsec {encr_algs 3des encr_auth_algs md5}
```

*auth_algs*セキュリティオプション

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。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 セキュリティオプション

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アルゴリズム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 セキュリティオプション

このオプションを設定すると、指定した暗号化アルゴリズムで、トンネルに IPsec ESP を使用できます。`encr_algs` 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。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。

使用可能な暗号化アルゴリズムの一覧とアルゴリズムのマニュアルページへのポインタについては、`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、このコマンド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 29 ページの「IPsec の実装 (作業マップ)」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)

作業	説明	参照先
2 つのシステム間のトラフィックの保護	次のことを行う: ■ `/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 ■ キー交換の設定 ■ `ipsecinit.conf` ファイルの呼び出し	31 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による、保護トラフィックだけの有効化。ファイルの呼び出しも行う	33 ページの「Web サーバーを保護する方法」

作業	説明	参照先
仮想プライベートネットワーク (VPN) のセットアップ	次のことを行う: ■ IP 転送のオフ ■ IP の厳密宛先マルチホームのオン ■ 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 ■ セキュリティ保護されたトンネルの設定 ■ IP 転送のオン ■ 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	35 ページの「仮想プライベートネットワーク (VPN) を構築する方法」
乱数の生成	手動で SA を作成する場合に、`od` コマンドを使って、キー情報を表す乱数を生成する	42 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 (SA) の作成または置き換え	`ipseckey` コマンドを使って SA を作成する。また、キー情報の格納先となる `ipseckeys` ファイルを作成する	43 ページの「IPsec セキュリティアソシエーションを手動で生成する方法」
IPsec がパケットを保護しているかどうかの検査	`snoop` の出力を調べ、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 (VPN) をセットアップするための手順について説明します。追加手順では、キー情報の提供、SA の提供、IPsec が設定どおり動作しているかどうかの確認を行います。

手順内の例で、システム名 `enigma` および `partym` を使用する場合があります。`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。

- 2 つのシステムが `enigma` および `partym` と名付けられている。この手順を実行する際、実際のシステム名で置き換えること
- 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 (SA) を使用する

  共有 SA 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、セキュリティ上トラフィックが盗聴される可能性があります。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。

   IPv4 アドレスしか持たないシステムに接続する場合は、`/etc/inet/hosts` ファイルに変更を加えます。

   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
   ```

   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` ファイル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** システムで、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** システムで、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 SA** の組を追加します。

   インターネットキー交換 (IKE) を設定すると、SA が自動的に生成されます。SA は手動でも追加できます。

注 – 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 を使用すべきです。IKE キー管理では、手動でのキー管理よりも強力なセキュリティ効果が得られます。

   - IKE を設定するには、61 ページの「IKE の設定 (作業マップ)」 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。
   - SA を手動で追加する場合は、43 ページの「IPsec セキュリティアソシエーションを手動で生成する方法」を参照してください。

6. 各システムをリブートします。

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

7. パケットが保護されていることを確認します。**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— リブートなしでのシステム間のトラフィックの保護

この例では、2 つのシステム間のトラフィックが保護されていることを確認する方法を示します。実際の稼働環境では、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。

31 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 でリブートする代わりに、次のいずれかの作業を行います。

- IKE を使ってキー情報を作成した場合は、in.iked デーモンをいったん停止後、再起動します。

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

- キーを手動で追加した場合は、ipseckey コマンドを使って、データベースに SA を追加します。続いて、ipsecconf コマンドを使用して IPsec ポリシーを有効にします。

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

注意 – ipsecconf 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。詳細については、25 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。

▼ 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 アルゴリズムによる ESP が必要です。その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 SA 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.
    {lport  80 ulp tcp dir both} bypass {}
    {lport 443 ulp tcp dir both} bypass {}
```

```
# Outbound DNS lookups should also be bypassed.
{rport 53 dir both} bypass {}

# 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、手順 2 で説明した検査を省略するトラフィックは例外です。

4. 手順 **3** で作成したファイルを **/etc/inet/ipsecinit.conf** ファイルに読み込みます。

```
# 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** サーバーをセキュリティ保護します。次のオプションのどちらか **1** つを選択します。

   - キー管理に IKE を使用する場合は、in.iked デーモンをいったん停止後、再起動します。

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

   - 手動でキーを管理する場合は、ipseckey および ipsecconf コマンドを実行します。

     ```
     # ipseckey -f /etc/inet/secret/ipseckeys
     # ipsecconf -a /etc/inet/IPsecWebInitFile
     ```

     ipsecinit.conf ファイル内にエントリがある場合は、再度読み込むときエラーが発生します。

注意 – ipsecconf 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。詳細については、25 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。in.iked デーモンの再起動時にも、同じ警告が表示されます。

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 接続で使用されます。

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。

7. **(省略可能)** リモートシステムと非 **Web** トラフィックを持つ **Web** サーバーとの通信を可能にするには、リモートシステムの **`ipsecinit.conf`** ファイルに次のポリシーを追加します。

   IPsec ポリシーが一致した場合にかぎり、リモートシステムは、非 Web トラフィックを持つ Web サーバーと安全に通信できます。

```
# Communicate with web server about nonweb stuff
#
{saddr webserver} permit {encr_algs blowfish encr_auth_algs sha}
{saddr webserver} apply {encr_algs blowfish encr_auth_algs sha sa shared}
```

▼ 仮想プライベートネットワーク (VPN) を構築する方法

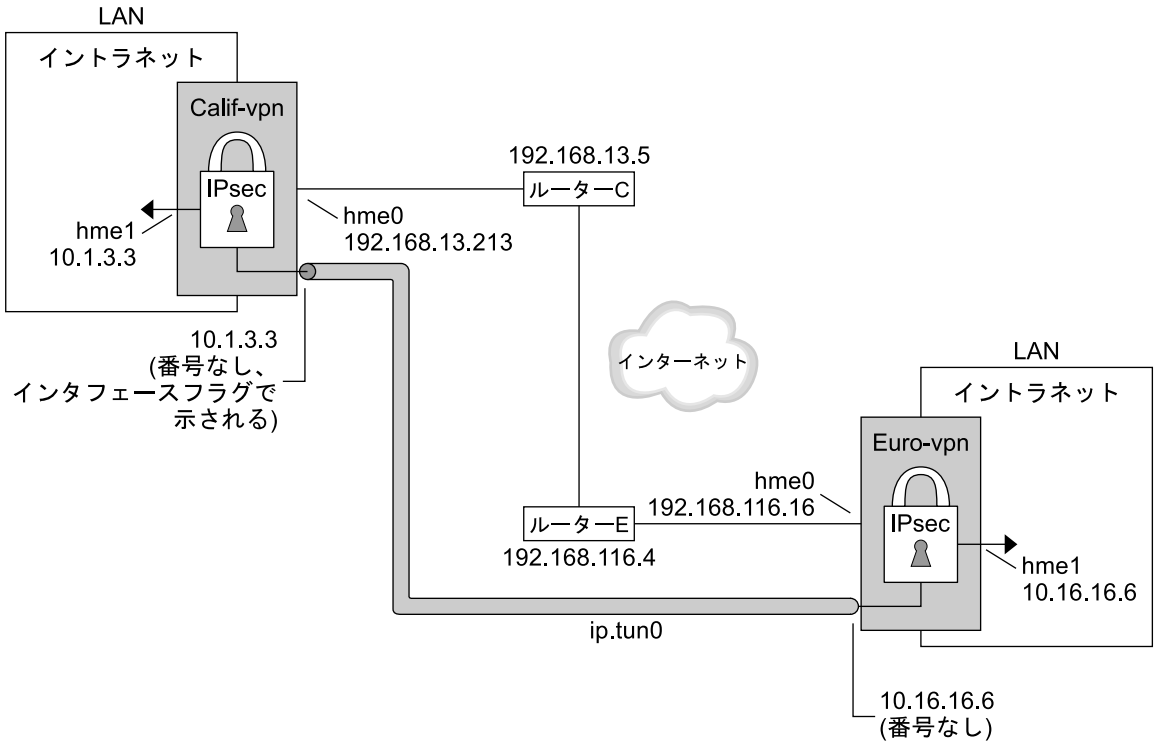
この手順では、インターネットで VPN を構築して組織内の 2 つのネットワークを接続する方法について説明します。また、そのネットワーク間のトラフィックを IPsec で保護する方法について説明します。

この手順は、31 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 つのシステムを接続するだけでなく、これら 2 つのシステムに接続している 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 つのシステム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。この手順は IPv6 アドレスを使用する場合も同じ
- 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ローカルエリアネットワーク (LAN)、すなわちイントラネットに接続する。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 (SA) を使用する

VPN については、21 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
システム名	enigma	partym
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

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. **IP** 転送をオフにします。次のいずれか **1** つを指定してください。

   - IPv4 ネットワークでは、次の構文を使用します。

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

   - IPv6 ネットワークでは、次の構文を使用します。

     ```
     # ndd -set /dev/ip6 ip6_forwarding 0
     ```

   IP 転送をオフにすると、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. **IP** の厳密宛先マルチホームをオンにします。次のいずれか **1** つを指定してください。

   - IPv4 ネットワークでは、次の構文を使用します。

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

   - IPv6 ネットワークでは、次の構文を使用します。

     ```
     # ndd -set /dev/ip6 ip6_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 転送をオフにし、IP 厳密宛先マルチホームをオンにすると、システムを流れるパケットの数は少なくなります。厳密宛先マルチホームが有効な状態では、特定のインタフェースに到着したパケットに、そのインタフェースのいずれかのローカル IP アドレスを指定する必要があります。その他のパ

ケットは、システムのほかのローカルアドレスが指定されているものも含めてすべて捨てられます。

4. 必要に応じて次の手順を行い、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** ファイルを編集し、重要なサービス以外のすべてのサービスを削除します。続いて、**inetd** デーモンを実行し、**inetd.conf** ファイルを再度読み込みます。

      ```
      # 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、33 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** ファイルの編集をしていない場合は、**inetd** デーモンを強制終了します。

      ```
      # pkill inetd
      ```

   c. 適切なコマンドを入力して **SNMP**、**NFS** など他のインターネットサービスを無効にします。たとえば、次のコマンドでは、**NFS** サービスとメールサービスが強制終了します。

      ```
      # /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 接続、rlogin 接続など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 **SA** の組を追加します。

   SA 用のキーを管理するように IKE を設定します。VPN に IKE を設定するには、61 ページの「IKE の設定 (作業マップ)」 のいずれかの手順を実行します。

   キーを手動管理する正当な理由がある場合は、43 ページの「IPsec セキュリティアソシエーションを手動で生成する方法」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** システムで、**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. **partym** システムで、**ipsecinit.conf** ファイルに次のエントリを追加します。

```
# 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 ファイル内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、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。次の 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** システム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. **partym** システムで、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** 転送をオンにします。次のいずれか **1** つを指定してください。

   - IPv4 ネットワークでは、次の構文を使用します。

     ```
     # ndd -set /dev/ip hme1:ip_forwarding 1
     # ndd -set /dev/ip ip.tun0:ip_forwarding 1
     ```

   - IPv6 ネットワークでは、次の構文を使用します。

     ```
     # ndd -set /dev/ip6 hme1:ip6_forwarding 1
     # ndd -set /dev/ip6 ip.tun0:ip6_forwarding 1
     ```

   IP 転送とは、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 転送はまた、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 転送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 転送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 転送はオンにしておきます。

   hme0 インタフェースの IP 転送はオフです。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 転送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。

    デフォルトルートは、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** システムで、次のルートを追加します。

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

    b. **partym** システムで、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** システムで **enigma** のインターネットルーターを **/etc/defaultrouter** ファイルに追加します。

```
# vi /etc/defaultrouter

192.168.116.4 router-E
```

b. **partym** システムのインターネットルーターを **partym** の **/etc/defaultrouter** ファイルに追加します。

```
# 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** システムで、**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** システムで、**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** という名前を付けます。

    各システムで、 hme1 および ip.tun0 インタフェースの IP 転送をオンにします。次のいずれか 1 つを指定してください。

    - IPv4 ネットワーク上で、ファイルに次の行を追加します。

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

    - IPv6 ネットワーク上で、ファイルに次の行を追加します。

      ```
      ndd -set /dev/ip6 hme1:ip6_forwarding 1
      ndd -set /dev/ip6 ip.tun0:ip6_forwarding 1
      ifconfig hme0 private
      in.routed
      ```

    in.routed プロトコル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

キーを手動で入力する場合、キー情報に無作為性を持たせる必要があります。キー情報は 16 進数で表されます。

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. **16** 進数の乱数を生成します。

   % od -x|**-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　乱数のソースとなる

   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. これらの出力を組み合わせて、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 文字キーを作成します。32 文字キーの長さは 128 ビットです。セキュリティパラメータインデックス (SPI) の場合は、8 文字キー 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生成する方法

IPsec セキュリティアソシエーション (SA) は手動でも管理できますが、セキュリティ上の理由からお勧めしません。 31 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」 に加えて、次の手順を実行します。まず、`ipseckey` コマンドで SA を作成します。次に、キー情報を `ipseckeys` ファイルに追加します。

1. **SA** のキー情報を生成します。

   16 進のアウトバウンドトラフィックと、同じく 16 進の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。つまり、1 台のシステムで次の数値を生成する必要があります。

   - `spi` キーワードの値として、2 つの 16 進数の乱数。1 つはアウトバウンドトラフィック用。もう 1 つ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 つは `dst enigma` 用。もう 1 つは `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 つは `dst enigma` 用。もう 1 つは `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、42 ページの「乱数を生成する方法」を参照してください。

2. 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。

---

3. 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。

4. 既存の **SA** を置き換える場合、現在の **SA** をフラッシュします。

```
> flush
>
```

悪意のあるユーザーによる SA の破壊を防ぐには、キー情報を置き換える必要があります。

注 – 管理者は、通信システム上のキーの置き換えを調整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リモートシステムの SA も置き換える必要があります。

---

5. **SA** を作成するには、次のコマンドを実行します。

次の構文で、フラッシュした SA を置き換えることもできます。

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

random-hex-string
16 進数形式の最大 8 桁の乱数を指定する。0x が前置される。セキュリティパラメータインデックス (SPI) が受け取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 を指定する

addr
システムの IP アドレスを指定する

addr2
addr のピアシステムの IP アドレスを指定する

protocol-prefix
encr または auth を指定する。encr 接頭辞は esp プロトコルとともに 使用される。auth 接頭辞は ah プロトコルとともに使用される。encr_auth_alg オプションは esp プロトコルとともに使用される

protocol-algorithm
ESP または AH のアルゴリズムを指定する。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 進数の乱数を指定。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** システムで、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。手順 **1** で生成した乱数を使用します。

```
> add esp spi 0x8bcd1407 \
src 192.168.116.16 dst 192.168.13.213 \
encr_alg 3DES \
encrkey d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d

> add ah spi 0x18907dae \
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** コマンドモードを使って、**enigma** システムで、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。

```
> add esp spi 0x122a43e4 \
src 192.168.13.213 dst 192.168.116.16 \
encr_alg 3des \
encrkey dd325c5c137fb4739a55c9b3a1747baa06359826a5e4358e

> add ah spi 0x91825a77 \
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、SA ごとに変更できます。SA ごとに異なるキーと異なる SPI を割り当てるべきです。

---

6. **ipseckey** コマンドモードを終了するには、**Control-D** キーを押すか、**quit** と入力します。

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

/etc/inet/secret/ipseckeys ファイルの行とコマンド行の言語が同じになるようにします。

a. たとえば、**enigma** システム上の **/etc/inet/secret/ipseckeys** ファイルは次のようになります。

```
# ipseckeys - This file takes the file format documented in
#   ipseckey(1m).
#   Note that naming services might not be available when this file
#   loads, just like ipsecinit.conf.
#
# for outbound packets on enigma
add esp spi 0x8bcd1407 \
   src 192.168.116.16 dst 192.168.13.213  \
   encr_alg 3DES \
   encrkey  d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d
#
add ah spi  0x18907dae \
   src 192.168.116.16 dst 192.168.13.213 \
   auth_alg MD5  \
   authkey  e896f8df7f78d6cab36c94ccf293f031
#
# for inbound packets
add esp spi 0x122a43e4 \
   src 192.168.13.213 dst 192.168.116.16 \
   encr_alg 3DES \
   encrkey dd325c5c137fb4739a55c9b3a1747baa06359826a5e4358e
#
add ah spi  0x91825a77 \
   src 192.168.13.213 dst 192.168.116.16 \
   auth_alg MD5  \
   authkey  ad9ced7ad5f255c9a8605fba5eb4d2fd
```

b. 読み取り専用ファイルを保護します。

```
# chmod 400 /etc/inet/secret/ipseckeys
```

8. **partym** システム上で手順 **2** から 手順 **7** を繰り返します。**enigma** システムの場合と同じキー情報を使用します。

   両システムのキー情報は同じでなければなりません。次の例のように、ipseckeys ファイル内のコメントだけが異なります。コメントが異なるのは、dst enigma が enigma システム上ではインバウンド、partym システム上ではアウトバウンドになるからです。

```
# partym ipseckeys file
#
#for inbound packets
add esp spi 0x8bcd1407 \
   src 192.168.116.16 dst 192.168.13.213  \
   encr_alg 3DES \
   encrkey  d41fb74470271826a8e7a80d343cc5aae9e2a7f05f13730d
#
add ah spi  0x18907dae \
   src 192.168.116.16 dst 192.168.13.213 \
   auth_alg MD5  \
   authkey  e896f8df7f78d6cab36c94ccf293f031
#
```

```
# for outbound packets
add esp spi 0x122a43e4 \
   src 192.168.13.213 dst 192.168.116.16 \
   encr_alg 3DES \
   encrkey dd325c5c137fb4739a55c9b3a1747baa06359826a5e4358e
#
add ah spi  0x91825a77 \
   src 192.168.13.213 dst 192.168.116.16 \
   auth_alg MD5  \
   authkey  ad9ced7ad5f255c9a8605fba5eb4d2fd
```

▼ 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:接頭辞は、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: 接頭辞は、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、スーパーユーザーであるか、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**) でスーパーユーザー になります。

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

2. 端末ウィンドウで、リモートシステム (たとえば **enigma** システム) 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インします。パスワードを入力します。次に、スーパーユーザーになり、**enigma** システムからのパケットを **partym** システムに送信します。

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

4. **partym** システムの **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 セキュリティアソシエーション (SA) 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 オペレーティング環境では、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 パケットと IPv6 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 ボードや Sun Crypto Accelerator 4000 ボ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ボードで行われます。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。Sun Crypto Accelerator 4000 ボードを搭載したシステムで IKE を使用する場合、証明書、公開鍵、および非公開鍵はこのボードに格納されます。鍵の格納先をシステム以外の場所にすることで、保護機能が強化されます。

この章では、以下の内容について説明します。

- 50 ページの「IKE の概要」
- 51 ページの「IKE 構成の選択」
- 52 ページの「IKE とアクセラレータハードウェア」
- 53 ページの「IKE とハードウェアストレージ」
- 53 ページの「IKE ユーティリティおよび IKE ファイル」

IKE の実装方法については、第 4 章を参照してください。

IKE の概要

IKE デーモン `in.iked` では、保護された方法で SA のキー情報のネゴシエーションと認証を行います。Solaris オペレーティング環境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。`in.iked(1M)` のマニュアルページを参照してください。

IKE デーモンによってリモートシステム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、リモートシステム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、このリモートシステムだけです。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 エンティティで 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 (SA) 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 SA 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。次の設定情報があります。

- グローバルパラメータ (公開鍵証明書の名前など)
- PFS (Perfect Forward Secrecy) を使用する場合
- 影響を受けるインタフェース
- 使用するアルゴリズム
- 認証方式

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、Baltimore Technologies、Entrust、GeoTrust、RSA Security、Sun Open Net Environment (Sun ONE) Certificate Server、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システム間の IPsec SA を作成および管理します。また、フェーズ 1 交換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 デバイス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 ipsecinit.conf 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

/etc/inet/ike/config 構成ファイルには、IKE ポリシーエントリが含まれています。2 つの IKE デーモンを相互に認証するためには、この構成ファイル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。構成ファイルのエントリは、フェーズ 1 交換を認証するためのキー情報の使用方法を決定します。選択肢は、事前共有鍵か公開鍵証明書のどちらかです。

エントリ 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。ike.config(4) のマニュアルページを参照してください。

IKE と事前共有鍵

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、リモートシステムの管理者とアウトオブバンドで共有します。使用する場合は、大量のランダムキーの作成、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 ファイルとなります。ike.preshared ファイルにある鍵に何らかの問題があると、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、そのリモートシステムの鍵と同一にする必要があります。これらの鍵は、特定の IP アドレスに関連付けられています。あるシステムの管理者が通信先のシステムを制御する場合、これらの鍵は最も安全です。ike.preshared(4) のマニュアルページを参照してください。

IKE と公開鍵証明書

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、キーの認証とネゴシエーションに Diffie-Hellman プロトコルを使用します。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、リモートシステムから X.509 形式で自己署名付き証明書の出力を取得します。リモート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、通信先システムの `/etc/inet/ike/publickeys` ディレクトリに保存されます。 証明書をシステムに接続されているハードウェアに保存したい場合は、`-T` オプションを指定します。

自己署名付き証明書は、事前共有鍵 と CA 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。固定番号を使用しないで、システムの証明書に自己署名するには、`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、`/etc/inet/ike/publickeys` ディレクトリに格納されます。 `-T` オプションを指定すると、証明書はシステムに接続されたハードウェアに保存されます。 また、ベンダーは証明書無効リスト (CRL、Certificate Renovation List) も発行します。使用する場合は鍵と 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 とアクセラレータハードウェア

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。大量の交換処理を行うシステムでは、Sun Crypto Accelerator 1000 ボードを使って公開鍵の操作を処理することができます。Sun Crypto Accelerator 4000 ボードを使って、計算量の多いフェーズ 1 交換を処理することもできます。

IKE の計算負荷をアクセラレータボードに移すための IKE の設定方法については、86 ページの「IKE で Sun Crypto Accelerator 1000 ボードを使用する方法」を参照してください。鍵の格納方法については、87 ページの「IKE で Sun Crypto Accelerator 4000 ボードを使用する方法」を参照してください。

IKE とハードウェアストレージ

公開鍵証明書、非公開鍵、および公開鍵は、Sun Crypto Accelerator 4000 ボードに格納されます。RSA の場合、このボードは最大 2048 ビットの鍵をサポートします。DSA の場合は、最大 1024 ビットになります。

ボードにアクセスするための IKE の設定方法については、86 ページの「IKE で Sun Crypto Accelerator 1000 ボードを使用する方法」を参照してください。ボードに証明書と公開鍵を追加する方法については、80 ページの「ハードウェア上で公開鍵証明書を生成、格納する方法」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。 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 の設定 (作業マップ)」を参照してください。

表 **3–1** IKE の構成ファイルおよびコマンド

ファイルまたはコマンド	説明
`in.iked` 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` コマンド	IKE ポリシーの表示および変更用 IKE 管理コマンド
`ikecert` コマンド	ローカルの公開鍵証明書データベースを操作する証明書データベース管理コマンド。 データベースも接続された Sun Crypto Accelerator 4000 ボードに格納できる
`/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
`/etc/inet/secret/ike.preshared` ファイル	事前共有鍵のファイル。フェーズ 1 交換での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

表 **3–1** IKE の構成ファイルおよびコマンド　(続き)

ファイルまたはコマンド	説明
`/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	公開鍵および証明書ファイルの無効リストを保存するディレクトリ
Sun Crypto Accelerator 1000 ボード	オペレーティングシステムの処理を少なくすることで公開鍵操作を高速化するハードウェア
Sun Crypto Accelerator 4000 ボード	オペレーティングシステムの処理を少なくすることで公開鍵操作を高速化するハードウェア。公開鍵、非公開鍵、および公開鍵証明書も格納する

IKE デーモン

`in.iked` デーモンを実行すると、Solaris システム上の IPsec 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システム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システムで実行する必要があります。

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 交換では、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 交換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 つのコマンドで IKE デーモンをサポートします。`ikeadm` コマンドを実行すると、IKE ポリシーを表示および変更できます。`ikecert` コマンドを実行すると、公開鍵データベースを表示および変更できます。このコマンドは、ローカルデータベース `ike.privatekeys` および `publickeys` を管理します。公開鍵の操作とハードウェア上の公開鍵の記憶領域も管理します。

IKE ポリシーファイル

IKE ポリシーの設定ファイル /etc/inet/ike/config により、IKE デーモン自体とこの構成ファイルで管理する IPsec SA に、キー操作の規則とグローバルパラメータが提供されます。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、キー情報を使用して保護されるシステム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「事前共有鍵による IKE の設定 (作業マップ)」を参照してください。そのエントリの例と説明は、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 ポリシーの設定ファイル /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 ファイルには、RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki) に従って実装されるライブラリのパスを含めることができます。IKE は、この PKCS #11 ライブラリを使って、アクセラレータおよびキーストレージハードウェアにアクセスします。

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、25 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
- フェーズ 1 交換時の SA 作成に関する統計情報の表示
- IKE プロセスのデバッグ

例とこのコマンドの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、キー情報を表示または変更できない。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、26 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。事前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.preshared ファイルは、事前共有鍵を必要とする ike/config ファイルの設定時に作成します。IKE 認証用として、ike.preshared ファイルに ISAKMP SA のキー情報を入力します。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。ファイルを手動で管理する例については、43 ページの「IPsec セキュリティアソシエーションを手動で生成する方法」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。

注 – 事前共有鍵は、ハードウェア記憶領域を利用できません。事前共有鍵は、生成後、システムに格納されます。

IKE 公開鍵のデータベースおよびコマンド

ikecert コマンドを実行すると、ローカルシステム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 コマンドを実行すると、Sun Crypto Accelerator 4000 ボード上のキーストレージも操作できます。ikecert コマンドの tokens 引数により、ボード上の使用可能なトークン ID を一覧表示できます。このコマンドは、/etc/inet/ike/config ファイルに指定された PKCS #11 ライブラリからボードを検索します。PKCS #11 エントリが必要です。このエントリが存在しない場合、ikecert コマンドの -T オプションは機能しません。エントリは次のようになります。

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

詳細については、ikecert(1M) のマニュアルページを参照してください。

ikecert tokens コマンド

tokens 引数は、Sun Crypto Accelerator 4000 ボード上の使用可能なトークン ID を一覧表示します。トークン ID により、ikecert certlocal コマンドと ikecert certdb コマンドは、公開鍵証明書を生成し、ボード上の要求を認証します。

ikecert certlocal コマンド

certlocal サブコマンドを実行して、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ます。-kc オプションを選択すると、証明書要求が作成されます。鍵はシステムの /etc/inet/secret/ike.privatekeys ディレクトリに格納されます。-T オプションを指定した場合は、システムに接続されたハードウェアに格納されます。

非公開鍵を作成する場合、ikecert コマンドには、ike/config ファイル内のエントリが必要です。ikecert 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** ikecert オプションと ike/config エントリの対応表

ikecert 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 よりもわずかに遅い認証方式。特許は登録されていない
-t rsa-md5 および -t rsa-sha1	auth_method rsa_sig	DSA よりもわずかに速い認証方式。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。通常、X.509 識別名などの ID ペイロードが最大ペイロードになる

表 **3–2** ikecert オプションと ike/config エントリの対応表　(続き)

ikecert オプション	**ike/config** エントリ	注
-t rsa-md5 および -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。
-T	pkcs11_path	PKCS #11 ライブラリは、Sun Crypto Accelerator 1000 ボードおよび Sun Crypto Accelerator 4000 ボード上で鍵操作の高速化処理を行います。このライブラリは、Sun Crypto Accelerator 4000 ボード上のキーストレージを操作するトークンも提供します。

ikecert certlocal -kc コマンドを指定して証明書要求を実行する場合、そのコマンド出力を PKI 機関または認証局 (CA) 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。PKI 機関、CA、または PKI の管理者はこれに基づいて証明書を作成します。PKI または CA から返された証明書は、certdb サブコマンドに渡されます。PKI から返された CRL は、certrldb サブコマンドに渡されます。

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、リモートシステムで 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、71 ページの「自己署名付き公開鍵証明書による 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、75 ページの「CA からの署名付き証明書による IKE の設定方法」を参照してください。

証明書と公開鍵は、システムの /etc/inet/ike/publickeys ディレクトリに格納されます。-T オプションを指定した場合、証明書、非公開鍵、公開鍵は、システムに接続されたハードウェアに格納されます。

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) データベース /etc/inet/ike/CRL 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、83 ページの「証明書無効リストを処理する方法」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。 -T オプションを指定すると、鍵は publickeys ディレクトリではなく Sun Crypto Accelerator 4000 ボード上に格納されます。

「スロット」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格納されます。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、リモート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。PKI に送信した情報に基づいて証明書を格納します。PKI から CA も格納します。

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

/etc/inet/secret/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。ikecert certlocal コマンドを実行して、ike.privatekeys 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 が格納されてから有効になります。ペアとなる公開鍵は /etc/inet/ike/publickeys ディレクトリまたは Sun Crypto Accelerator 4000 ボードに格納されます。

/etc/inet/ike/crls ディレクトリ

/etc/inet/ike/CRL 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

IKE の管理 (手順)

この章では、使用するシステムにあわせて IKE を設定する方法について説明します。IKE の設定が完了すると、そのネットワークにおける IPsec のキー情報が自動的に生成されます。61 ページの「IKE の設定 (作業マップ)」に、この章で説明する手順を一覧表示します。

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 の設定 (作業マップ)

作業	説明	参照先
事前共有鍵による IKE の設定	2 つのシステムに秘密鍵を共有させることにより、その通信を保護する	62 ページの「事前共有鍵による IKE の設定 (作業マップ)」
公開鍵証明書による IKE の設定	公開鍵証明書を使って通信を保護する。証明書は、自己署名付き、または PKI 機関の保証付き	71 ページの「公開鍵証明書による IKE の設定 (作業マップ)」

作業	説明	参照先
公開鍵証明書を生成し、接続されたハードウェアに格納する設定	Sun Crypto Accelerator 1000 ボードまたは Sun Crypto Accelerator 4000 ボードを使って IKE 操作を高速化する。Sun Crypto Accelerator 4000 ボードには、公開鍵証明書も格納できる	85 ページの「IKE とハードウェアの使用 (作業マップ)」

役割を使って IKE を設定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。

事前共有鍵による IKE の設定 (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	63 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 `ipseckeys` ファイルに最新のキー情報を追加する	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」
事前共有鍵が同一であることのチェック	両方のシステムの事前共有鍵のダンプを行う	70 ページの「事前共有鍵が同一であることを確認する方法」

▼ 事前共有鍵による IKE の設定方法

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。キーの長さは、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。

これらの手順には、システム名 enigma および partym を使用します。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。

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、31 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"      ラベルは一意でなくてはなりません
  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 パラメータ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 **/etc/inet/ike/config** 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"     ラベルは一意でなくてはなりません
  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
}
```

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. キー情報として使用する乱数を生成します。

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。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
```

od コマンドの説明については、42 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. 手順 **5** の出力から、キーを **1** つ作成します。

```
f47cb0f432e14480951095f82b735ba80a9467d08f92c88068b6a40e
```

この手順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。この例の鍵は、推奨されている最小文字数より長い 56 文字です。

7. 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。

   a. たとえば、**enigma** システムの **ike.preshared** ファイル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** ファイル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 – 両システムの事前共有鍵は同一にする必要があります。

▼ 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. 乱数を生成し、適切な長さのキーを作成します。
   詳細については、42 ページの「乱数を生成する方法」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、現在のキーを新しいキーに変更します。
   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。

```
# 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。

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

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

▼ 新しい事前共有鍵を追加する方法

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鍵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いるものとします。

- 2 つのシステム enigma および ada(実際に使用するシステムの名前で置き換え)
- 両システムで 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、両システムの /etc/hosts ファイルのエントリとして存在する。次に例を示す

```
192.168.15.7 ada
```

  /etc/inet/ipnodes ファイル内の Ipv6 アドレスにも同じ手順を適用

- 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 システムの新しいエントリは次のようになる

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

ada システムのエントリは次のようになる

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

- enigma システムと ada システムが安全に通信できるようにするための規則を、両システムの /etc/inet/ike/config ファイルに記述している。たとえば、enigma システム上の規則は次のようになる

```
### ike/config file on enigma, 192.168.116.16
...
## The rule to communicate with ada

{ label "enigma-to-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
    }
```

ada システムの規則は次のようになる

```
### 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
}
```

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 0x0, base privileges 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。次に、正しい権限レベルでデーモンを再起動します。

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

4. 乱数を生成し、**64** ～ **448** ビットのキーを作成します。
   詳細については、42 ページの「乱数を生成する方法」を参照してください。

5. このキーを何らかの方法でリモート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* のタイプを指定する

   id　*id-type* が IP のとき IP アドレスを指定する

   mode　IKE モードを指定する。有効な値は `main` だけ

   key　16 進数の事前共有鍵を指定する

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

   ```
   # 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.
   ```

   b. ホスト **ada** でも、同じキーを追加します。

   ```
   # 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.
   ```

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
   ```

注意 – ipsecconf コマンドの実行時には警告を読んでください。in.iked デーモンの再起動時にも、同じ警告が表示されます。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。詳細については、25 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。

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。

```
# ikeadm read rules
```

ada および enigma システム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、ikeadm コマンドでファイル名を指定する必要はありません。

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、この鍵を **/etc/inet/secret/ike.preshared** ファイルに追加します。

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 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d
}
```

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 on ada for the enigma interface
#...
{ 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 8d1fb4ee500e2bea071deb2e781cb48374411af5a9671714672bb1749ad9364d
}
```

12. 両システムが通信できることを確認します。**70** ページの「事前共有鍵が同一であることを確認する方法」を参照してください。

▼ 事前共有鍵が同一であることを確認する方法

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
```

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していることを確認します。

```
# /usr/sbin/ikeadm get priv
Current privilege level is 0x0, base privileges enabled
```

権限レベル 0x2 が返される場合は、キー情報を表示できます。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表示を許可しない場合は、このデーモンを強制終了します。次に、正しい権限レベルでデーモンを再起動します。

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

4. システムごとに、事前共有鍵情報を表示します。

```
# ikeadm dump preshared
PSKEY: Preshared 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).
```

5. 両方のダンプを比較します。

事前共有鍵が同一でない場合は、/etc/inet/secret/ike.preshared ファイルで、一方のキーを他方のキーで置き換えます。

6. 確認が終わったら、**in.iked** デーモンの権限レベルを低くします。

```
# ikeadm set priv base
```

公開鍵証明書による IKE の設定 (作業マップ)

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リモートシステムからの公開鍵を追加する	71 ページの「自己署名付き公開鍵証明書による IKE の設定方法」
PKI 認証局による IKE の設定	`ikecert certlocal -kc` コマンドの出力を PKI 機関に送信し、その機関からの公開鍵、CA、および CRL を追加する	75 ページの「CA からの署名付き証明書による IKE の設定方法」
ローカルハードウェア上での公開鍵証明書の設定	次のいずれかの作業を行う ■ ローカルハードウェア上で自己署名付き証明書を生成し、リモートシステムの公開鍵をハードウェアに追加する ■ ローカルハードウェア上で証明書要求を生成し、PKI 機関から取得した公開鍵証明書をハードウェアに追加する	80 ページの「ハードウェア上で公開鍵証明書を生成、格納する方法」
PKI 機関から取得した証明書無効リスト (CRL) の更新	中央の配布ポイントから CRL にアクセスする	83 ページの「証明書無効リストを処理する方法」

▼ 自己署名付き公開鍵証明書による 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. 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|-kc -m keysize -t keytype \
-D dname -A altname
```

-ks　自己署名付き証明書を作成する

-kc　証明書要求を作成する。手順は 75 ページの「CA からの署名付き証明書による IKE の設定方法」を参照

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。有効なタグは、IP、DNS、EMAIL、URI、DN、RID

a. たとえば、**partym** システムでは、コマンドは次のようになります。

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
> -A IP=192.168.13.213
Creating software private keys.
  Writing private key to file /etc/inet/secret/ike.privatekeys/0.
Enabling external key providers - done.
Acquiring private keys for signing - done.
Certificate:
 Proceeding with the signing operation.
 Certificate generated successfully (.../publickeys/0)
Finished successfully.
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

b. **enigma** システムでは、コマンドは次のようになります。

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
> -D "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax" \
> -A IP=192.168.116.16
Creating software private keys.
  ...
Certificate added to database.
-----BEGIN X509 CERTIFICATE-----
MIICKDCCAZGgAwIBAgIBATANBgkqhkiG9w0BAQQFADBJMQswCQYDVQQGEwJVUzEV
...
jpxfLM98xyFVyLCbkr3dZ3Tvxvi732BXePKF2A==
-----END X509 CERTIFICATE-----
```

3. 証明書を保存し、リモートシステムに送信します。

証明書は、電子メールに貼り付けることもできます。

a. たとえば、次の **partym** 証明書を **enigma** の管理者に送信します。

```
To: admin@ja.enigmaexample.com
From: admin@us.partyexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MIICLTCCAZagAwIBAgIBATANBgkqhkiG9w0BAQQFADBNMQswCQYDVQQGEwJVUzEX
...
6sKTxpg4GP3GkQGcd0r1rhW/3yaWBkDwOdFCqEUyffzU
-----END X509 CERTIFICATE-----
```

b. **enigma** の管理者は、次の **enigma** 証明書を送信してきます。

```
To: admin@us.partyexample.com
From: admin@ja.enigmaexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MIICKDCCAZGgAwIBAgIBATANBgkqhkiG9w0BAQQFADBJMQswCQYDVQQGEwJVUzEV
...
jpxfLM98xyFVyLCbkr3dZ3Tvxvi732BXePKF2A==
-----END X509 CERTIFICATE-----
```

4. システムごとに、証明書が認識されるように **/etc/inet/ike/config** ファイルを編集します。

リモートシステムの管理者は、 cert_trust、remote_addr、および remote_id パラメータの値を提供します。

a. たとえば、**partym** システム上の **ike/config** ファイルは次のようになります。

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

cert_trust "192.168.13.213"
cert_trust "192.168.116.16"

## 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 "US-partym to JA-enigmax"
 local_id_type dn
 local_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"
 remote_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"

 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}
}
```

b. **enigma** システムで、**ike/config** ファイルにローカルパラメータの **enigma** 値を追加します。

リモートパラメータには、partym 値を使用します。キーワード label の値が一意であることを確認します。この値は、リモートシステムの label 値とは異なる値でなければなりません。

```
...
{
 label "JA-enigmax to US-partym"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213
...
```

5. システムごとに、受け取った証明書を追加します。

a. 管理者の電子メールから公開鍵をコピーします。

b. **ikecert certdb -a** コマンドを入力後、**Return** キーを押します。

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。続いて **Return** キーを押します。**Control-D** キーを押して入力を終了します。

```
-----BEGIN X509 CERTIFICATE-----
MIIC...
...
----END X509 CERTIFICATE-----
    Return キーを押す
<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。共有の証明書の公開鍵ハッシュは、両システムで同一でなければなりません。

a. たとえば、**partym** システムで、格納されている証明書を一覧表示します。

```
partym # ikecert certdb -l
Certificate Slot Name: 0   Type: rsa-md5
    Subject Name: <C=US, O=PartyCompany, OU=US-Partym, CN=Partym>
    Key Size: 1024
    Public key hash: B2BD13FCE95FD27ECE6D2DCD0DE760E2

Certificate Slot Name: 1   Type: rsa-md5
    (Private key in certlocal slot 0)
    Subject Name: <C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax>
    Key Size: 1024
```

```
    Public key hash: 2239A6A127F88EE0CB40F7C24A65B818
```

b. **enigma** システムで、格納されている証明書を一覧表示します。

```
enigma # ikecert certdb -l
Certificate Slot Name: 4   Type: rsa-md5
    Subject Name: <C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax>
    Key Size: 1024
    Public key hash: DF3F108F6AC669C88C6BD026B0FCE3A0

Certificate Slot Name: 5   Type: rsa-md5
    (Private key in certlocal slot 4)
    Subject Name: <C=US, O=PartyCompany, OU=US-Partym, CN=Partym>
    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。

▼ CA からの署名付き証明書による 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。

コマンド引数の詳細については、71 ページの「自己署名付き公開鍵証明書による IKE の設定方法」 の手順 2 を参照してください。

```
# ikecert certlocal -kc -m keysize -t keytype \
-D dname -A altname
```

a. たとえば、次のコマンドでは、**partym** システム上に証明書要求が作成されます。

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany\, Inc., OU=US-Partym, CN=Partym" \
> -A "DN=C=US, O=PartyCompany\, Inc., OU=US-Partym"
Creating software private keys.
  Writing private key to file /etc/inet/secret/ike.privatekeys/2.
Enabling external key providers - done.
Certificate Request:
```

```
  Proceeding with the signing operation.
  Certificate request generated successfully (.../publickeys/0)
Finished successfully.
-----BEGIN CERTIFICATE REQUEST-----
MIIByjCCATMCAQAwUzELMAkGA1UEBhMCVVMxHTAbBgNVBAoTFEV4YW1wbGVDb21w
...
lcM+tw0ThRrfuJX9t/Qa1R/KxRlMA3zckO80mO9X
-----END CERTIFICATE REQUEST-----
```

b. 次のコマンドでは、**enigma** システム上に証明書要求が作成されます。

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=JA, O=EnigmaCo\, Inc., OU=JA-Enigmax, CN=Enigmax" \
> -A "DN=C=JA, O=EnigmaCo\, Inc., OU=JA-Enigmax"
Creating software private keys.
...
Finished successfully.
-----BEGIN CERTIFICATE REQUEST-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...
8qlqdjaStLGfhDOO
-----END CERTIFICATE REQUEST-----
```

3. この証明書要求を **PKI** 機関に送信します。

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、次の 2 つの証明書オブジェクトと、証明書無効リストを発行します。

- 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 (LDAP サーバー上のディレクトリ名) 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 を取得し、キャッシュすることができる

  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、83 ページの「証明書無効リストを処理する方法」を参照

4. **ikecert certdb -a** コマンドを使って、各証明書をシステムに追加します。

`-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、52 ページの「IKE と公開鍵証明書」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. **PKI** 機関から受け取った公開鍵証明書を追加します。

```
# ikecert certdb -a
    Return キーを押す
    証明書を貼り付ける
-----BEGIN X509 CERTIFICATE-----
...
-----END X509 CERTIFICATE----
    Return キーを押す
<Control>-D
```

c. **PKI** 機関の **CA** を追加します。

```
# ikecert certdb -a
    Return キーを押す
    CA を貼り付ける
-----BEGIN X509 CERTIFICATE-----
...
-----END X509 CERTIFICATE----
    Return キーを押す
<Control>-D
```

d. **PKI** 機関が証明書無効リスト **(CRL)** を送信してきている場合は、これを **certrldb** データベースに追加します。

```
# ikecert certrldb -a
    Return キーを押す
    CRL を貼り付ける
-----BEGIN CRL-----
...
-----END CRL----
    Return キーを押す
<Control>-D
```

5. **/etc/inet/ike/config** ファイルを編集して、**PKI** 機関が認識されるようにします。

PKI 機関が提供する名前を使用します。

a. たとえば、**partym** システムの **ike/config** ファイルは次のようになります。

```
# 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-partym to JA-enigmax - Example PKI"
 local_id_type dn
 local_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"
 remote_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"

 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}
}
```

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

---

b. **enigma** システムで、ローカルパラメータに **enigma** 値、リモートパラメータに **partym** 値を使用します。

キーワード label の値が一意であることを確認します。この値は、リモートシステムの label 値とは異なる値でなくてはなりません。

```
...
{
 label "JA-enigmax to US-partym - Example PKI"
 local_id_type dn
 local_id   "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 local_addr  192.168.116.16
 remote_addr 192.168.13.213
...
```

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。

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

ignore_crls キーワードにより、IKE は CRL を検索しなくなります。

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、83 ページの「証明書無効リストを処理する方法」 を参照してください。

注 – 次の手順は、/etc/inet/ike/config ファイル内の auth_method に rsa_encrypt が設定されている場合にのみ必要です。

8. **auth_method** パラメータが **rsa_encrypt** に設定されているので、ピアの証明書を **publickeys** データベースに追加します。

   a. その証明書を、リモートシステムの管理者に送信します。
   証明書は、電子メールに貼り付けることもできます。

      i. たとえば、**partym** の管理者は次のような電子メールを送信します。

```
To: admin@ja.enigmaexample.com
From: admin@us.partyexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MII...
----END X509 CERTIFICATE-----
```

      ii. **enigma** の管理者は次のような電子メールを送信します。

```
To: admin@us.partyexample.com
From: admin@ja.enigmaexample.com
Message: -----BEGIN X509 CERTIFICATE-----
MII
...
-----END X509 CERTIFICATE-----
```

   b. システムごとに、電子メールで送信された証明書をローカルの **publickeys** データベースに追加します。

```
# ikecert certdb -a
    Return キーを押す
-----BEGIN X509 CERTIFICATE-----
MII...
-----END X509 CERTIFICATE-----
    Return キーを押す
<Control>-D
```

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 に rsa_encrypt を指定する場合には、ピアの証明書を publickeys データベースに追加する必要があります。この結果、publickeys 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。

- ユーザーの公開鍵証明書
- CA 証明書
- ピアの公開鍵証明書

▼ ハードウェア上で公開鍵証明書を生成、格納する方法

ハードウェア上で公開鍵および公開鍵証明書を生成、格納するための要件は、次のとおりです。

- ハードウェアの設定が完了していること
- `/etc/inet/ike/config` ファイルが、RSA Security Inc. PKCS #11 Cryptographic Token Interface (Cryptoki) に準拠して実装されているライブラリ、すなわち PKCS #11 ライブラリを指していること

  設定手順については、87 ページの「IKE で Sun Crypto Accelerator 4000 ボードを使用する方法」を参照してください。

ハードウェア上で公開鍵証明書を生成、格納する方法は、システム上で公開鍵証明書を生成、格納する方法とほぼ同じです。違いは次の 2 点です。

- `ikecert certlocal` および `ikecert certdb` コマンドがハードウェアを識別しなければならない。トークン ID に `-T` オプションを指定すると、コマンドがハードウェアを識別するようになる
- `/etc/inet/ike/config` ファイルが `pkcs11_path` キーワードでハードウェアを指していなければならない

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. 自己署名付き証明書か証明書要求を生成し、トークン **ID** を指定します。次のオプションのどれか **1** つを選択します。

注 – Sun Crypto Accelerator 4000 ボードは、RSA で最大 2048 ビットのキーをサポートします。DSA の場合は最大 1024 ビットになります。

- 自己署名付き証明書の場合、次の構文を使用する

```
# ikecert certlocal -ks -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
> -a -T SUN-4000-stor IP=192.168.116.16
Creating hardware private keys.
Enter PIN for PKCS#11 token:      Type user:password
-----BEGIN X509 CERTIFICATE-----
MIIBwjCCASsCBD9bz5swDQYJKoZIhvcNAQEEBQAwKDELMAkGA1UEBhMCVVMxGTAX
...
```

```
PiktCuvURc1TXswaFyftzmLKWafUOQ==
-----END X509 CERTIFICATE-----
```

-T オプションの引数は、Sun Crypto Accelerator 4000 ボードのトークン ID

- 証明書要求の場合、次の構文を使用する

```
# ikecert certlocal -kc -m 1024 -t rsa-md5 \
> -D "C=US, O=PartyCompany, OU=US-Partym, CN=Partym" \
> -a -T SUN-4000-stor IP=192.168.116.16
Creating hardware private keys.
Enter PIN for PKCS#11 token:      Type user:password
-----BEGIN X509 CERTIFICATE-----
MIIBuDCCASECAQAwSTELMAkGA1UEBhMCVVMxFTATBgNVBAoTDFBhcnR5Q29tcGFu
...
oKUDBbZ9O/pLWYGr
-----END X509 CERTIFICATE-----
```

ikecert コマンドの引数の詳細については、ikecert(1M) のマニュアルページを参照してください。

3. **PIN** のプロンプトに、**Sun Crypto Accelerator 4000** ユーザー、コロン、ユーザーのパスワードを入力します。

   Sun Crypto Accelerator 4000 ボードのユーザー ikemgr のパスワードが rgm4tigt の場合、次のように入力します。

```
Enter PIN for PKCS#11 token: ikemgr:rgm4tigt
```

注 – PIN の応答は、ディスク上にクリアテキストとして格納されます。

---

4. 通信先に証明書を送信します。次のオプションのどれか **1** つを選択します。

   - リモートシステムに自己署名付き証明書を送信します。証明書は、電子メールに貼り付けることもできます。
   - PKI を処理する機関に証明書要求を送信します。証明書要求は、PKI 機関の指示に従って送信します。詳細については、75 ページの「CA からの署名付き証明書による IKE の設定方法」の手順 3 を参照してください。

5. システム上で、**/etc/inet/ike/config** ファイルを編集して、証明書が認識されるようにします。次のオプションのどれか **1** つを選択します。

   - 自己署名付き証明書の場合は、リモートシステムの管理者がパラメータ cert_trust、remote_id、および remote_addr 用に提供する値を使用します。

     たとえば、enigma システムの ike/config ファイルは次のようになります。

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

cert_trust "192.168.116.16"      ローカルシステムの証明書
```

```
cert_trust "192.168.13.213"       リモートシステムの証明書

pkcs11_path "/opt/SUNWconn/lib/libpkcs11.so"      ハードウェア接続
...
{
 label "JA-enigmax to US-partym"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 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}
}
```

- 証明書要求の場合は、PKI 機関が cert_root キーワードの値として提供する名前を入力します。

  たとえば、enigma システムの ike/config ファイルは次のようになります。

```
# 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"

pkcs11_path "/opt/SUNWconn/lib/libpkcs11.so"      ハードウェア接続
...
{
 label "JA-enigmax to US-partym - Example PKI"
 local_id_type dn
 local_id "C=JA, O=EnigmaCo, OU=JA-Enigmax, CN=Enigmax"
 remote_id  "C=US, O=PartyCompany, OU=US-Partym, CN=Partym"

 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}
}
```

6. 通信先から受け取った証明書をハードウェアに格納します。

   手順 3 の場合と同様に、PIN 要求に応答します。

注 – 公開鍵証明書は、公開鍵を生成したハードウェアに追加する必要があります。

- 自己署名付き証明書の場合、リモートシステムの自己署名付き証明書を追加します。

```
# ikecert certdb -a -T SUN-4000-stor
    Return キーを押す
```

```
    自己署名付き証明書を貼り付ける
<Control>-D
Enter PIN for PKCS#11 token:      ユーザー名とパスワードを入力する
```

自己署名付き証明書の auth_method パラメータの値として rsa_encrypt を使用した場合、ハードウェアストアにピアの証明書を追加します。

```
# ikecert certdb -a -T SUN-4000-stor
    Return キーを押す
    ピアの証明書を貼り付ける
<Control>-D
Enter PIN for PKCS#11 token:      ユーザー名とパスワードを入力する
```

- PKI 機関の証明書の場合、その機関が証明書要求に応じて発行した証明書と、認証局 (CA) を追加します。

```
# ikecert certdb -a -T SUN-4000-stor
    Return キーを押す
    PKI が発行した証明書を貼り付ける
<Control>-D
Enter PIN for PKCS#11 token:      ユーザー名とパスワードを入力する

# ikecert certdb -a -T SUN-4000-stor
    Return キーを押す
    CA 証明書を貼り付ける
<Control>-D
Enter PIN for PKCS#11 token:      ユーザー名とパスワードを入力する
```

PKI 機関から取得した証明書無効リスト (CRL) を追加する方法については、83 ページの「証明書無効リストを処理する方法」を参照してください。

▼ 証明書無効リストを処理する方法

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、75 ページの「CA からの署名付き証明書による IKE の設定方法」の手順 6 を参照
- CA から受け取った公開鍵証明書に URI (Uniform Resource Indicator) 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 (ディレクトリ名) 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
- 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. **CA** から受け取った証明書を表示します。

   # **ikecert certdb -lv** *certspec*

   -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　　IKE 証明書データベース内の証明書と一致するパターン

   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: dsa-sha1
   (Private key in certlocal slot 0)
 Subject Name: <O=Sun Microsystems Inc, CN=example-protect.sun.com>
 Issuer Name: <CN=Sun Microsystems Inc CA (Cl 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 (Cl 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** を追加します。

   たとえば、ike/config ファイルは次のようになります。

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

キーワード proxy を ike/config ファイルに追加して、Web プロキシを使用することもできます。キーワード proxy は、次のように引数として URL を取ります。

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

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。

3. **LDAP** を使用する場合は、ホストの **/etc/inet/ike/config** ファイルのキーワード **ldap-list** 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

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 certrldb 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 とハードウェアの使用 (作業マップ)

作業	説明	参照先
IKE キーの操作を Sun Crypto Accelerator 1000 ボードで行う	PKCS#11 ライブラリのパス設定を行う	86 ページの「IKE で Sun Crypto Accelerator 1000 ボードを使用する方法」
IKE キーの操作とキーの格納を Sun Crypto Accelerator 4000 ボードで行う	PKCS#11 ライブラリのパス設定と、使用可能なトークン ID の一覧表示を行う	87 ページの「IKE で Sun Crypto Accelerator 4000 ボードを使用する方法」

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

注 – 次の手順では、Sun Crypto Accelerator 1000 ボ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、ボ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 Accelerator 1000 ボード上の IKE 公開鍵操作を高速化します。ボ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 交換のデータを保持するからです。PKCS #11 によって処理が高速化されたキーは、フェーズ 1 データの一部です。

▼ IKE で Sun Crypto Accelerator 4000 ボードを使用する方法

注 – 次の手順では、Sun Crypto Accelerator 4000 ボ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、ボ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 4000 Board Installation and User's Guide*』を参照してください。このマニュアルには、Sun Hardware Documentation の Web サイトの「Network and Security Products」の下からアクセスできます。

---

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 Accelerator 4000 ボード上でキー生成および格納処理を行います。

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
   ...
   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。

   注 – Sun Crypto Accelerator 4000 ボードは、RSA で最大 2048 ビットのキーをサポートします。DSA の場合は最大 1024 ビットになります。

   ---

5. 接続されている **Sun Crypto Accelerator 4000** ボードのトークン **ID** を検索します。

```
$ ikecert tokens
Available tokens with library "/opt/SUNWconn/lib/libpkcs11.so":

"SUN-1000-accel                "
"SUN-4000-stor                 "
```

ライブラリは 32 文字のトークン ID (キーストア名) を返します。この例では、`ikecert` コマンドに `SUN-4000-stor` トークンを指定して IKE キーを格納します。

トークンの使用方法については、80 ページの「ハードウェア上で公開鍵証明書を生成、格納する方法」を参照してください。

`ikecert` コマンドにより、後続スペースが自動的に付加されます。

付録 A

『IPsec と IKE の管理』の更新情報

このマニュアルに記載されている Solaris 9 オペレーティング環境の機能の更新情報を一覧します。

Solaris 9 4/03 の更新情報

IKE 暗号化のハードウェアによる高速化が可能になりました。86 ページの「IKE で Sun Crypto Accelerator 1000 ボードを使用する方法」を参照してください。

Solaris 9 12/03 の更新情報

- IPv4 ネットワークに加えて IPv6 ネットワークでも IKE を実行できるようになりました。
- IKE 公開鍵、非公開鍵、および証明書を Sun Crypto Accelerator 4000 ボードに格納できるようになりました。このボードで IKE 暗号化を高速化することができます。詳細については、次の各節を参照してください。
  - 53 ページの「IKE とハードウェアストレージ」
  - 80 ページの「ハードウェア上で公開鍵証明書を生成、格納する方法」
  - 87 ページの「IKE で Sun Crypto Accelerator 4000 ボードを使用する方法」

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。

3DES 「Triple-DES」を参照。

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。

CA 「認証局 (CA)」を参照。

DES Data Encryption Standard。1975 年に開発され、1981 年に ANSI X.3.92 として ANSI で標準化された対称鍵の暗号化方式。DES では 56 ビットのキーを使用する。

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る。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 は秘密鍵認証アルゴリズムの 1 つである。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。IPsec セキュリティアソシエーション (SA) の認証されたキー情報を自動的に提供する。

IP データグラム	IP 経由で転送される情報パケット。IP データグラムはヘッダーとデータを持つ。ヘッダーにはデータグラムのソースと宛先のアドレスが含まれる。ヘッダーのその他のフィールドには、複数のデータグラムを宛先で識別し、再結合するための情報が含まれる。
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。
IPsec	IP データグラムを保護するための IP 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。
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。
SA	「セキュリティアソシエーション (SA)」 を参照
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。SHA-1 アルゴリズムは DSA に入力される。
SPI	「セキュリティパラメータインデックス (SPI)」を参照。
Triple-DES	Triple-Data Encryption Standard。対称鍵暗号化システムの 1 つ。Triple-DES で使用する鍵の長さは 168 ビット。 Triple-DES を「3DES」と表記することもある。
鍵管理	セキュリティアソシエーション (SA) 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。
キーストア名	管理者がネットワークインタフェースカード (NIC) 上の記憶領域やキーストアに付与する名前。キーストア名は、「トークン」、「トークン ID」とも呼ばれる。

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、メッセージの受信者だけが知っている。IKE により、IPsec の公開鍵が提供される。
証明書無効リスト **(CRL)**	CA が無効とした公開鍵証明書のリスト。
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。
セキュリティペイロードのカプセル化 **(ESP)**	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。
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。たとえば、Sun Crypto Accelerator 4000 ボードは NIC の 1 つである。
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を持つ。
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。ハッシュ関数として、MD5、SHA-1 などがある。
非対称鍵暗号化	メッセージの送受信側で異なる鍵 (キー) 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。
ペイロード	パケットで伝送されるデータ。ペイロードには、パケットを宛先に送るために必要なヘッダー情報は含まれない。
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。
メッセージ認証コード **(MAC)**	データの整合性を保証し、データの出所を明らかにするコード。MAC は盗聴行為には対応できない。

索引

数字・記号

3DES 暗号化アルゴリズム
  キー長, 45
  と IPsec, 18
> プロンプト, ikeadm コマンドモード, 68
> プロンプト, ipseckey コマンドモード, 44

A

-A オプション, ikecert コマンド, 57
AES 暗号化アルゴリズム, と IPsec, 18
AH, 認証ヘッダー (AH)を参照
auth_algsセキュリティオプション,
  ifconfig コマンド, 27
-a オプション
  ikecert certdb コマンド, 76
  ikecert certrldb コマンド, 85
  ikecert コマンド, 80
  ipsecconf コマンド, 33, 68

B

Blowfish 暗号化アルゴリズム, と IPsec, 18

C

cert_root キーワード, 77
cert_trust キーワード, 73
CRL
  CRL データベース, 59
CRL (続き)
  ikecert certrldb コマンド, 58
  一元的な場所からアクセス, 83
  無視, 78
  リスト, 84
-c オプション, in.iked デーモン, 64

D

DES 暗号化アルゴリズム, と IPsec, 18
/dev/ipsecah ファイル, 16
/dev/ipsecesp ファイル, 17
/dev/random デバイス, 42
DSS 認証アルゴリズム, 57
-D オプション, ikecert コマンド, 57

E

encr_algs セキュリティオプション,
  ifconfig コマンド, 28
encr_auth_algs セキュリティオプション,
  ifconfig コマンド, 28
ESP, セキュリティペイロードのカプセル化 (ESP)を参照
/etc/inet/hosts ファイル, 31
/etc/inet/ike/config ファイル
  cert_root キーワード, 77
  cert_trust キーワード, 73
  ignore_crls キーワード, 78
  ikecert コマンド, 57
  ldap-list キーワード, 85

/etc/inet/ike/config ファイル (続き)
PKCS #11 ライブラリエントリ, 56
pkcs11_path キーワード, 56, 80, 86, 87
proxy キーワード, 85
rsa_encrypt 認証方式, 79
use_http キーワード, 84
概要, 53
公開鍵証明書, 77
サンプル, 63
自己署名付き証明書, 73
セキュリティについて, 55
説明, 51, 55
と CRL, 83
と事前共有鍵, 63
ハードウェアに証明書を格納, 81
/etc/inet/ike/CRL ディレクトリ, 59
/etc/inet/ike/publickeys ディレクトリ, 59
/etc/inet/ipnodes ファイル, 31
/etc/inet/ipsecinit.conf ファイル, 24, 32, 34
/etc/inet/ipsecpolicy.conf ファイル, 23
/etc/inet/secret/ike.privatekeys ディレクトリ, 59
/etc/init.d/inetinit スクリプト, 24

F

-f オプション, ipseckey コマンド, 33

H

HMAC-MD5 認証アルゴリズム, と IPsec, 18
HMAC-SHA 認証アルゴリズム, と IPsec, 18
hosts ファイル, 31

I

ifconfig コマンド
auth_algs セキュリティオプション, 27
encr_algs セキュリティオプション, 28
encr_auth_algs セキュリティオプション, 28
IPsec セキュリティオプション, 27
ifconfig コマンド (続き)
トンネルの設定, 20
ignore_crls キーワード, 78
IKE
CRL データベース, 59
CRL の処理, 83
/etc/inet/ike/config ファイル, 86, 87
ike.preshared ファイル, 56
ike.privatekeys データベース, 59
ikeadm コマンド, 55, 65
ikecert certdb コマンド, 76
ikecert certlocal コマンド, 75
ikecert certrldb コマンド, 85
ikecert tokens コマンド, 88
ikecert コマンド, 56
in.iked デーモン, 54
ISAKMP SA, 50
Perfect Forward Secrecy, 50
PKCS #11 ライブラリ, 58, 87
publickeys データベース, 59
RSA 暗号化アルゴリズム, 79
インターネットキー交換, 50
概要, 49
キーのハードウェアストレージ, 53
権限レベルのチェック, 66
権限レベルの変更, 68
事前共有鍵の更新, 65, 66
実装, 61, 71
セキュリティアソシエーション, 50, 54
設定, 53, 62
と事前共有鍵, 62
と証明書, 52
とハードウェア, 85
ハードウェアによる高速化, 52
フェーズ 1 交換, 50
フェーズ 2 交換, 51
ポリシーの妥当性検査, 64
ike/config ファイル,
/etc/inet/ike/config ファイルを参照
ike_mode キーワード, 68
ike.preshared ファイル, 56, 65
サンプル, 69
ike.privatekeys データベース, 59
ikeadm コマンド
権限レベルのチェック, 66
権限レベルの変更, 68
説明, 54, 55
対話モード, 68

ikecert certdb コマンド, 74
ikecert certlocal コマンド, 71
ikecert certrldb コマンド, 85
ikecert tokens コマンド, 88
ikecert コマンド
  説明, 54, 56
in.iked デーモン
  起動, 54
  権限レベルのチェック, 66
  権限レベルの変更, 68
  説明, 50
  停止と起動, 32, 66, 70
inetd.conf ファイル, IPsec, 38
inetinit スクリプト, 24
ipnodes ファイル, 31
IPsec
  /dev/ipsecah ファイル, 16
  /dev/ipsecesp ファイル, 17
  /etc/hosts ファイル, 31
  /etc/inet/ipnodes ファイル, 31
  /etc/inet/ipsecinit.conf ファイル, 32, 34
  /etc/inet/ipsecpolicy.conf ファイル, 23
  /etc/init.d/inetinit スクリプト, 24
  ifconfig コマンド, 39
    VPN の設定, 39
    セキュリティオプション, 27
    ポリシーの設定, 23
  in.iked デーモン, 15
  inetd.conf ファイル, 38
  ipsecconf コマンド, 19, 23
  ipsecinit.conf ファイル, 24
  ipseckey コマンド, 15, 26
  ndd コマンド, 16
  route コマンド, 41
  snoop コマンド, 28
  Web サーバーの保護, 33
  アウトバウンドパケットプロセス, 12
  暗号化アルゴリズム, 17, 18
  暗号化アルゴリズムの指定, 27
  一時的にポリシーを設定, 23
  インバウンドパケットプロセス, 14
  永続的にポリシーを設定, 24
  概要, 11
  仮想プライベートネットワーク (VPN), 21
  キー管理, 15
IPsec (続き)
  キーユーティリティ
    IKE, 50
    ipseckey コマンド, 26
  起動, 22
  実施機構, 19
  実装, 29
  省略, 19
  セキュリティアソシエーション, 15
  セキュリティアソシエーションデータベース, 25
  セキュリティアソシエーションの置き換え, 44
  セキュリティアソシエーションの作成, 43
  セキュリティアソシエーションの追加, 32
  セキュリティパラメータインデックス (SPI), 15
  セキュリティプロトコル, 15
  セキュリティペイロードのカプセル化, 16, 17
  設定, 19, 22, 23
  データのカプセル化, 17
  トラフィックの保護, 31
  トランスポートモード, 19
  トンネル, 21
  トンネルモード, 19
  認証アルゴリズム, 18
  認証アルゴリズムの指定, 27
  認証ヘッダー, 16
  バイパス, 33
  パケットの保護, 11
  保護機構, 16
  保護ポリシー, 19
  ポリシーコマンド, 23
  ポリシーファイル, 24
  ユーティリティの拡張
    ifconfig コマンド, 27
    snoop コマンド, 28
ipsecconf コマンド
  -a オプション, 33, 68
  IPsec の起動, 22
  IPsec ポリシーの設定, 19, 23
  セキュリティについて, 33
ipsecconf コマンド, セキュリティについて, 25
ipsecinit.conf ファイル
  サンプル, 24
  セキュリティについて, 25

ipseckeys ファイル, IPsec キーの格納, 23
ipseckey コマンド, 44
  IPsec キーの管理, 22
  セキュリティについて, 26
  説明, 15, 26
ipsecpolicy.conf ファイル, 23
IP セキュリティアーキテクチャ, IPsecを参照
IP データグラム, IPsec による保護, 11
IP 転送
  VPN での, 37, 40
  VPN における, 21
ISAKMP SA, 50

K

-kc オプション
  ikecert certlocal コマンド, 57, 75
-ks オプション, ikecert certlocal コマンド, 57

L

ldap-list キーワード, ike/config ファイル, 85

M

MD5 認証アルゴリズム
  キー長, 45
  と IPsec, 18

N

ndd コマンド
  IPsec の調整, 17
  IP 転送, 37
  VPN の設定, 42

O

od コマンド, 42, 64
/opt/SUNWconn/lib/libpkcs11.so エントリ, ike/config ファイル内, 56

P

Perfect Forward Secrecy, IKE, 50
PF_KEY ソケットインタフェース
  IPsec, 15, 22
PKCS #11 ライブラリ, 58, 87
  ike/config ファイル内, 56
pkcs11_path キーワード, 56, 80, 86, 87
proxy キーワード, ike/config ファイル, 85
publickeys データベース, 59
-p オプション, in.iked デーモン, 67

R

route コマンド, IPsec, 41
rsa_encrypt 認証方式, ike/config ファイル, 79
RSA 暗号化アルゴリズム, 58
RSA 認証アルゴリズム, 79

S

Sun Crypto Accelerator 1000 ボード, 52, 86
Sun Crypto Accelerator 4000 ボード, 87
  IKE キーの格納, 53
  IKE の処理の高速化, 52
SHA 認証アルゴリズム, と IPsec, 18
snoop コマンド
  保護されたパケットの表示, 28, 47

T

tokens 引数, ikecert コマンド, 57
Triple-DES 暗号化アルゴリズム, と IPsec, 18
-T オプション
  ikecert コマンド, 58, 80
-t オプション, ikecert コマンド, 57

U

URI (Uniform Resource Indicator), CRL にアクセスするための, 83
use_http キーワード, ike/config ファイル, 84

V

-v オプション, snoop コマンド，28

W

Web サーバー, IPsec による保護，33

あ

暗号化アルゴリズム
  IPsec，17
    3DES，18
    AES，18
    Blowfish，18
    DES，18
  IPsec 用に指定，27

か

回避, LAN 上の IPsec，39
格納
  IKE キーをディスクに，76
  ディスク上の IKE キー，58, 59
  ハードウェア上の IKE キー，53, 87
仮想プライベートネットワーク (VPN)
  IPsec で構築，21
  ndd コマンドによる設定，37, 42
  設定，35
  例，35

き

キー
  ike.privatekeys データベース，59
  ike/publickeys データベース，59
  IPsec の管理，15
  事前共有，51
  自動管理，50
  手動管理，26
  ハードウェアに格納，53
  乱数の生成，42
キー管理
  IKE，50
  IPsec，15
キー管理 (続き)
  自動，50
  手動，26
キーストア名, トークン IDを参照
キーユーティリティ
  IKE プロトコル，49
  ipseckey コマンド，15

け

権限レベル
  IKE に設定，67
  IKE のチェック，66

こ

公開鍵証明書, 証明書を参照
高速化
  IKE の処理，52, 86
コマンド
  IKE
    ikeadm コマンド，53, 54, 55, 68
    ikecert コマンド，53, 54, 56
    in.iked デーモン，54
  IPsec
    ipsecconf コマンド，19, 22, 23, 33
    ipseckey コマンド，15, 22, 26, 44
    snoop コマンド，28, 47
    セキュリティについて，26
    リスト，22

さ

作業マップ
  IKE，61
  IKE キーのハードウェアによる高速化，85
  IKE キーをハードウェアに格納，85
  IKE と公開鍵証明書，71
  IKE と事前共有鍵，62
  IKE とハードウェア，85
  IPsec，29

し
システム, 通信保護, 31
事前共有鍵, 作業マップ, 62
証明書
  CA から, 76
  CA による署名付き, 75
  CRL を無視, 78
  ike/config ファイル内, 81
  自己署名付き, 71
  説明, 52, 76
  データベースに追加, 76
  ハードウェア上の CA の, 83
  ハードウェア上の自己署名付き, 80
  ハードウェア上の要求, 81
  ハードウェアストレージ, 80
  ハードウェアに格納, 53, 80
  要求, 58, 75
  リスト, 74
証明書無効リスト, CRLを参照
省略, IPsec ポリシー, 19
処理
  IKE のハードウェアによる高速化, 52, 86, 87

す
スロット, ハードウェア内, 59

せ
セキュリティ
  IKE, 54
  IPsec, 11
セキュリティアソシエーション (SA)
  IKE, 54
  IPsec, 15, 32
  IPsec SA の置き換え, 44
  IPsec SA の作成, 43
  IPsec SA のフラッシュ, 44
  IPsec データベース, 25
  IPsec の追加, 32
  ISAKMP, 50
  ISAKMP SA の置き換え, 65
  乱数発生, 51
セキュリティアソシエーションデータベース (SADB), 25
セキュリティについて
  ike/config ファイル, 55
  IKE の設定, 63
  ipsecconf コマンド, 25
  ipsecinit.conf ファイル, 25
  ipseckeys ファイル, 46
  ipseckey コマンド, 26
  IPsec の設定, 31
  カプセル化されたセキュリティペイロード, 17
  キー長, 63
  事前共有鍵, 51
  認証ヘッダー, 16
  ラッチされたソケット, 25
セキュリティパラメータインデックス (SPI)
  キーサイズ, 43
  説明, 15
セキュリティペイロードのカプセル化 (ESP)
  IPsec 保護機構, 16
  IP パケットの保護, 11
  ndd コマンドによる調整, 17
  説明, 17
設定
  IKE, 61
  ike/config ファイル, 55
  IPsec, 23
  ipsecinit.conf ファイル, 24

そ
ソケット
  IPsec セキュリティ, 25
  セキュリティについて, 33

た
対話モード
  ikeadm コマンド, 68
  ipseckey コマンド, 44

て
ディレクトリ名 (DN), CRL にアクセスするための, 83
データグラム, IP, 11

デーモン
in.iked デーモン, 50, 53, 54
デジタル署名
DSA, 57
RSA, 58, 79

と
トークン ID, ハードウェア内, 59
トランスポートモード, IPsec, 19
トンネル
ifconfig セキュリティオプション, 27
IPsec, 21
パケットの保護, 21
トンネルモード, IPsec, 19

に
認証アルゴリズム
IKE, 57
IPsec
MD5, 18
SHA, 18
IPsec 用に指定, 27
認証ヘッダー (AH)
IPsec のモジュール, 16
IPsec 保護機構, 16
IP データグラムの保護, 16
IP パケットの保護, 11

は
ハードウェア
IKE キーの格納, 53, 87
IKE の処理の高速化, 52, 86, 87
パケット
IKE による保護, 50
IPsec による保護, 16
アウトバウンド, 12
インバウンド, 14
IPsec 保護の確認, 47

ふ
ファイル
IKE
CRL ディレクトリ, 54, 59
ike/config ファイル, 23, 51, 53, 55
ike.preshared ファイル, 53, 56, 66
ike.privatekeys ディレクトリ, 59
ike.privatekeys ファイル, 54
publickeys ディレクトリ, 54, 59
IPsec
/etc/inet/ipsecpolicy.conf ファイル, 23
/etc/init.d/inetinit ファイル, 24
ipsecinit.conf ファイル, 22, 24
ipseckeys ファイル, 23
ipsecinit.conf, 22

ほ
保護
2 つのイントラネット間のパケット, 35
Web サーバーと IPsec, 33
システム間のパケット, 31
保護機構, IPsec, 16
ポリシーファイル
ike/config ファイル, 23, 53, 55
ipsecinit.conf ファイル, 24
ipsecpolicy.conf 一時ファイル, 23
セキュリティについて, 25

ま
マシン, 通信保護, 31

ら
ライブラリ
PKCS #11, 58, 87
乱数
/dev/random デバイス, 42
od コマンドで生成, 42, 64

ろ
ローカルファイル名サービス
  /etc/inet/hosts ファイル, 31
  /etc/inet/ipnodes ファイル, 31