

## 第 XIV 部

# IP パケットの暗号化と認証



## 第 14 部

### IP パケットの暗号化と認証

#### 第 1 章 はじめに

IPsec のための鍵交換プロトコルとしては標準の IKE があるが、IETF の IPsec WG では、IKE の改定版である Internet Key Exchange version 2 (IKEv2) の仕様策定が進んでいる。また、IPsec の鍵交換の認証に Kerberos を用いた Kerberized Internet Negotiation Keys (KINK) の策定が KINK WG で行われている。

WIDE Project IPsec WG では、昨年度に引き続き、これら複数の鍵交換プロトコルを、1 つのプラットフォーム上で同時にサポートするためのアーキテクチャの研究を行い、その実装として racoon2 の開発を中心にした活動を行った。

WIDE 2004 年 12 月研究会では、その成果として KINK および IKEv2 の鍵交換のデモを行った。

#### 第 2 章 racoon2

racoon2 は、複数の鍵交換プロトコルを BSD 系 OS と Linux で動作させるためのアーキテクチャである。サポートしているプロトコルは IKEv2、KINK であり、今後 Internet Key Exchange (IKEv1) もサポートする予定である。対象とするプラットフォームは、FreeBSD、NetBSD、Linux である。

##### 2.1 アーキテクチャ

racoon2 は、以下の 3 つのデーモンから構成される。

- spmd            policy management daemon
- iked            IKE daemon
- kinkd           KINK daemon

これら 3 つのデーモンの中で特徴的なのは spmd である。spmd はセキュリティポリシーを管理すると

同時にリゾルバの役割を果たす。これは、FQDN から IP アドレスへの変換と、その結果をキャッシュし逆変換に用いるためである。この機能は KINK をサポートするために必要となる。

KINK は、認証に Kerberos を使い、そのためには Kerberos のプリンシパル名を必要とする。それは以下のように FQDN から導出される。

```
kink/fqdn@REALM
```

一方、カーネルは FQDN に関する情報を持たず、IP アドレスによって鍵交換を始動するため、IP アドレスから FQDN のマッピングを行う必要があり、それは、アプリケーションが使用した FQDN であるべきである。このために IP アドレスと FQDN のマッピングを行う spmd を導入した。

racoon2 の実装は、libracoon を用いた以下のようなモジュールからなる。それぞれのデーモンに共通する機能は、libracoon に集約されている。

##### 2.2 設定モデル

racoon2 の設定は以下の 5 つの情報をリンクしたものを基本とする。それらは

- selector\_info
- policy\_info
- ipsec\_info
- sa\_info
- remote\_info

である。

selector\_info は、Security Policy Database (SPD) の selector に相当する。selector としてアドレスを指定する部分には、FQDN または特定の IP アドレスを示す特殊な文字列を用いることができる。

policy\_info は、selector\_info からポイントされ selector\_info にマッチしたパケットの処理に関する基本的な情報が格納される。ポリシーが IPsec であった場合、policy\_info は remote\_info と ipsec\_info をポイントする。

ipsec\_info は、IPsec SA の有効期限やモードを保持する。トンネルモードの場合には IPsec SA のエンドポイントといった IPsec SA に関する基本的な情報を保持し、sa\_info をポイントする。



タを交換する。これによって IKE\_SA を確立し、その IKE\_SA 上において相互の認証と IPsec SA の確立を行う。

<i>Initiator</i>	<i>Responder</i>
IKE_SA_INIT(request)	IKE_SA_INIT(response)
IKE_AUTH(request)	IKE_AUTH(response)

## 2.6 kinkd

kinkd は racoon2 フレームワーク内で KINK プロトコルを提供する、鍵管理デーモンである。

KINK とはセキュリティプロトコルで用いるパラメータを管理するための、自動鍵管理プロトコルである。現在は IPsec の鍵管理に用いることが想定されているが、IKE と同様、DOI を定義することで IPsec 以外のセキュリティプロトコルにも適用可能となるよう設計されている。

KINK の主な特徴としては以下のものが挙げられる。

- 認証機構として Kerberos を利用する。
- プロトコルは基本的に two-way であり、メッセージ交換中の状態遷移を比較的簡単にできる。  
( responder が initiator の提案に同意しなかった場合のみ、3-way のハンドシェイクが行われる )

これらによって、公開鍵暗号などのコストのかかる処理や、通信のレイテンシを最小限にとどめ、効率的かつ安全に鍵管理を行うことを目標としている。

KINK は、交換するパラメータを表現する部分では、ISAKMP で定義されているペイロードのフォーマットをそのまま利用している。

kinkd は KINK で定義されるメッセージタイプ、ペイロードタイプをすべてサポートし、ISAKMP で定義されているペイロードタイプのうち、以下のものをサポートする予定である。

- Security Association Payload
- Proposal Payload
- Transform Payload
- Identification Payload
- Nonce Payload
- Notification Payload
- Delete Payload

## 第3章 実装

### 3.1 spmd

後述する libracoon の API が決まる以前に UDP のみであるが DNS proxy としての機能は動作しており、設定した FQDN と実 IP アドレスを内部にキャッシュする機能はできていたが、今回 libracoon の API ができあがったことにより libracoon を利用するように改造を行った。

改造を行った点としては spmd 起動時に racoon2 設定ファイルを読み込み初期 IPsec セキュリティポリシーとして値を設定する。

このとき設定ファイル中に FQDN が記述されていたら /etc/nsswitch.conf ファイルなど OS の名前解決設定ファイルを参照し /etc/hosts ファイルなどから静的な FQDN ↔ IP アドレス設定をキャッシュしかつ DNS の問い合わせパケットを DNS サーバに投げ名前解決できた IPsec セキュリティポリシーを設定する。

ただし、現在 libracoon によって定義されているマクロ ( MY\_IP など ) への対応が不十分であり、これらをきちんと実装する必要がある。

また、現在主に kinkd のみが利用している UNIX ドメインソケットもしくは inet ソケットにより各鍵交換デーモンは spmd を接続し IP アドレスから FQDN 問い合わせ、IPsec セキュリティポリシーの設定依頼などを行うことができる ( spmif )。

### 3.2 iked

#### 3.2.1 処理フロー

iked の内部処理フローの概略は以下の通りである。

1. コマンドラインオプション処理
2. コンフィギュレーションファイルを読む  
( libracoon/rcf\_read() )
3. PF\_KEY インタフェースライブラリ初期化  
( libracoon/rcpfk\_init() )
4. ISAKMP の UDP ソケット作成
5. 無限ループで、作成したソケットに対して select(2) する  
PF\_KEY ソケットから SADB\_ACQUIRE を受信

1. IKE\_SA が存在しなければ initiator として作成し、
  2. IKE\_SA から CHILD\_SA を initiator として作成する。その後の処理は状態遷移を参照のこと。ISAKMP の UDP ソケットにパケットを受信
1. プロトコルバージョン番号により振り分ける(以下は IKEv2 に関して述べる)
  2. IKE\_SA が存在するか調べ、
  3. 存在する場合は IKE\_SA の状態処理ルーチンへ
  4. 存在しない場合、パケットが新規リクエストであれば新しい IKE\_SA を responder として作成
- IKE\_SA と CHILD\_SA の状態遷移の概略は以下の通りである。

**IKE\_SA initiator**

IKEV2\_STATE\_IDLE

- ・ - (通常のイニシエートの場合はここで) initiator の CHILD\_SA を作成。
- ・ CHILD\_SA が GETSPI\_DONE に遷移するのを待つ

- ・ responder へ IKE\_SA\_INIT を送信

IKEV2\_STATE\_IN\_IKE\_SA\_INIT\_SENT

- ・ responder から IKE\_SA\_INIT を受信
- ・ responder へ IKE\_AUTH パケットを送信

IKEV2\_STATE\_IN\_IKE\_AUTH\_SENT

- ・ responder から IKE\_AUTH を受信

IKEV2\_STATE\_ESTABLISHED

- ・ ライフタイム経過、またはエラー発生

IKEV2\_STATE\_DYING

- ・ すべての CHILD\_SA が回収される

(消滅)

**IKE\_SA responder**

(IKE\_SA が存在しない状態から)

- ・ initiator から IKE\_SA\_INIT パケットを受信

IKEV2\_STATE\_IDLE

- ・ initiator へ IKE\_SA\_INIT を送信

IKEV2\_STATE\_RES\_IKE\_SA\_INIT\_SENT

- ・ initiator から IKE\_AUTH を受信
- ・ initiator へ IKE\_AUTH を送信

IKEV2\_STATE\_ESTABLISHED

- ・ ライフタイム経過、またはエラー発生

IKEV2\_STATE\_DYING

- ・ すべての CHILD\_SA が回収される

(消滅)

**CHILD\_SA initiator**

IKEV2\_CHILD\_STATE\_IDLE

- ・ PF\_KEY へ SADB\_GETSPI リクエスト送出

IKEV2\_CHILD\_STATE\_GETSPI

- ・ PF\_KEY から SADB\_GETSPI リプライ受信

IKEV2\_CHILD\_STATE\_GETSPI\_DONE

- ・ IKE\_SA が IDLING 状態ならば IKE\_SA ネゴシエーションをスタート、ESTABLISHED 状態ならば CREATE\_CHILD\_SA ネゴシエーションをスタート

IKEV2\_CHILD\_STATE\_WAIT\_RESPONSE

- ・ ネゴシエーション終了
- ・ PF\_KEY へ SADB\_ACQUIRE 送出

IKEV2\_CHILD\_STATE\_MATURE

- ・ ライフタイム経過

IKEV2\_CHILD\_STATE\_EXPIRED

(消滅)

**CHILD\_SA responder**

IKEV2\_CHILD\_STATE\_IDLE

- ・ PF\_KEY へ SADB\_GETSPI 送出

## IKEV2\_CHILD\_STATE\_GETSPI

- PF\_KEY から SADB\_GETSPI 受信
- IKE\_SA が RES\_IKE\_SA\_INIT\_SENT 状態ならば IKE\_SA ネゴシエーションの続き、ESTABLISHED 状態ならば CREATE\_CHILD\_SA ネゴシエーションの続き

## IKEV2\_CHILD\_STATE\_MATURE

- ライフタイム経過

## IKEV2\_CHILD\_STATE\_EXPIRED

(消滅)

## 3.2.2 IKEv1 プロトコル実装状況

現在のところ実装されているのは IKEv2 のみであり、IKE(v1) プロトコルは対応していない。

## 3.2.3 IKEv2 プロトコル実装状況

IKEv2 仕様に対して以下のような制約・未実装部分がある。

- CERT、CERTREQ ペイロード取り扱いの未実装 (受信すると無視またはエラー)
- certificate は x509\_subject( ID\_DER\_ASN1\_DN ) でしか使えない
- HTTP\_CERT\_LOOKUP\_SUPPORTED、ESP\_TFC\_PADDING\_NOT\_SUPPORTED、NON\_FIRST\_FRAGMENTS\_ALSO への対応なし (受信すると無視)
- DSS ( Digital Signature Standard ) はなし
- ENCR\_AES\_CTR はなし
- プロトコルのオプションな部分でサポートしていない、または最低限サポートのもの
  - NAT traversal なし
  - CONFIG ペイロードなし (受信するとエラー)
  - EAP なし (同)
  - window は minimum ( 1 )
  - rekey なし
- informational exchange の扱いは minimum
- DELETE ペイロードの処理はなし
- peer polling なし
- 相手が NO\_ADDITIONAL\_SAS を返してきた

時の処理がない

- cookie の secret は固定
- トランスフォームネゴシエーション、ID = 0 の対応がない
- IPCOMP\_SUPPORTED の取り扱いはなし

## 3.2.4 racoon2 仕様に対する未実装部分

racoon2 仕様に対して以下のような制約・未実装部分がある。

- SA ライフタイムは iked が管理せずカーネルの SADB\_EXPIRE メッセージで処理する。
- コンフィギュレーション項目の proposal\_check が未実装 (指定にかかわらず obey )。
- コンフィギュレーション項目の selector\_check が ID の選択に反映されない。TS の選択は exact のみ (これは仕様どおり)。
- コンフィギュレーションに指定する IP アドレスは数値形式でなければならない。
- コンフィギュレーションにマクロが使用できない。
- コンフィギュレーションに矛盾・不足があっても、使われるまで検出できない。
- MobileIP 連携などはまだ無い

## 3.3 kinkd

2004 年 12 月末時点での kinkd の実装状況を示す。

## 3.3.1 メッセージ交換パターン

kinkd の行うメッセージ交換は、大きく 3 つに分けることができる。このうち、KINK プロトコルが規定しているものは、基本的に鍵管理の部分である。

- TGT 取得  
Kerberos の KDC ( Key Distribution Center ) から TGT ( Ticket-Granting Ticket ) を取得する。
- サービスチケット取得  
KDC から通信相手に提出するためのチケットを取得する。  
基本的に Kerberos の KRB\_TGS\_REQ メッセージのみで取得するが、user-to-user モードを用いる場合は KINK の GETTGT メッセージも用いる。
- 鍵管理  
通信相手と折衝を行い、SA を設定する。  
CREATE メッセージで SA を生成し、DELETE

メッセージで SA を削除する。  
以下に、各機能の実装状況を示す。

表 3.1. TGT 取得 (Kerberos)

Message flow	
KRB_AS_REQ — KRB_AS_REP	OK

表 3.2. サービスチケット取得 (Kerberos (+KINK))

Message flow	
KRB_TGS_REQ — KRB_TGS_REP	OK
GETTGT — REPLY — KRB_TGS_REQ — KRB_TGS_REP	NG
CREATE — REPLY (KRB_AP_ERR_USER_TO_USER_REQUIRED) — GETTGT — REPLY — KRB_TGS_REQ — KRB_TGS_REP	NG

表 3.3. 鍵管理 (KINK)

Message flow	
CREATE — REPLY	OK
CREATE — REPLY — ACK (i.e 3-way hand shake)	OK
DELETE — REPLY	OK
STATUS — REPLY	OK

### 3.3.2 REPLY メッセージで送受信するエラー

「送」は responder がそのエラーを返信するかどうか、「受」は initiator がそのエラーを受信して処理できるかどうかを示す。

KINK\_KRB\_ERROR では以下の処理が可能である。

表 3.4. KINK\_KRB\_ERROR

ERROR	送	受
KRB_AP_ERR_BAD_INTEGRITY	OK	OK
KRB_AP_ERR_TKT_EXPIRED	OK	OK
KRB_AP_ERR_SKEW	OK	OK
KRB_AP_ERR_NOKEY	OK	OK
KRB_AP_ERR_BADKEYVER	OK	OK

表 3.5. KINK\_ERROR

ERROR	送	受
KINK_OK	never	OK
KINK_PROTOERR	OK	OK
KINK_INVDOI	OK	OK
KINK_INVMAJ	OK	OK
KINK_INVMIN	OK	OK
KINK_INTERR	OK	OK
KINK_BADQMVERS	OK	OK

- TKT\_EXPIRED 時のトランザクションやり直し
- ERR\_SKEW 時の処理 [kink-06 5.1.4 (page 18) 7.1 (page 27)]
- responder は ctime に時刻を入れて返す
- 次回 initiator は ctime で受けとった時刻を元にオフセットする

### 3.3.3 SAD 管理

「送」は kinkd がカーネルにそのメッセージを送信するかどうか、「受」は kinkd がカーネルからそのメッセージを受信して処理できるかどうかを示す。

表 3.6. SAD 管理

PF_KEYv2 message	送	受
SADB_GETSPI	OK	OK
SADB_UPDATE	OK	NG(無視)
SADB_ADD	OK	NG(無視)
SADB_DELETE	OK	OK
SADB_ACQUIRE	NG	OK
SADB_REGISTER	OK	NG(無視)
SADB_EXPIRE	never	OK

### 3.3.4 ISAKMP ペイロード

kinkd が各 ISAKMP ペイロードを送受信できるかどうかを示す。

[] で囲まれているものは、KINK で optional と規定されているものである。

表 3.7. CREATE/REPLY-to-CREATE

ISAKMP ペイロード	
SA	OK
Nonce( Ni )	OK
Nonce( Nr, 3-way の場合 )	OK
[KE]	NG
[ID]	OK
[Notification]	NG

表 3.8. DELETE/REPLY-to-DELETE

ISAKMP ペイロード	
Delete	OK
[Notification]	inprogress

表 3.9. STATUS/REPLY-to-STATUS

ISAKMP ペイロード	
[Notification]	OK

## 3.3.5 各種パラメータ

表 3.10. IPsec プロトコル

<i>IPsec protocol</i>	
AH	OK
ESP	OK

表 3.11. IPsec SA のモード

<i>Mode of SA</i>	
Transport	OK
Tunnel	NG

表 3.12. 認証アルゴリズム

<i>Authentication Algorithm</i>	
MD5	OK
SHA-1	OK

表 3.13. 暗号アルゴリズム

<i>Encryption Algorithm</i>	
DES	OK
3DES	OK
AES-128	OK
AES-192	OK
AES-256	OK

## 3.3.6 libracoon 化

kink は最初 racoon1 をベースに書かれたので、racoon2 の共通ライブラリである libracoon を利用していない部分があった。ここではそれらの部分の libracoon 化の状況を示す。

表 3.14. libracoon 対応

機能	
SADB インタフェース (rcpfb)	OK
設定ファイル操作 (rcf)	inprogress
共通タイプ (RCT)	inprogress
spmd インタフェース (spmif)	OK
ログ出力 (plog)	NG
バッファ (wmbuf, rbuf)	OK
ネットワークユーティリティ	inprogress

## 3.3.7 その他

表 3.15. その他機能

機能・仕様	
epoch change の検出・処理 (DPD) [kink-06 4.4.2 (page 10)]	OK
複数の KINK_ISAKMP ペイロード [kink-06 5.1.7 (page 21)]	NG
randomized rekeying time [kink-06 4.4.1 (page 9)]	NG
replay protection (by Kerberos)	OK
DELETE grace timer	OK

## 3.4 libracoon

本節では racoon2 の共通ライブラリ libracoon.a について記述する。

## 3.4.1 共通ライブラリ概要

共通ライブラリ libracoon は、鍵管理デーモンの実装を容易にするためにカーネルとのインタフェースや設定ファイルの解読などの共通処理をライブラリ関数として提供する。

共通ライブラリは、主に以下のモジュールに分けられる。

- SAD/SPD 管理インタフェース
- 設定ファイル操作
- spmd インタフェース
- ログ出力
- 汎用バッファ rbuf に関するユーティリティ
- ネットワークに関するユーティリティ
- その他ユーティリティ

libracoon では、暗号アルゴリズムや各タイプの定数の実装による違いを吸収するために中間形式 (RCT 形式) を定義している。ライブラリでの定数はすべて RCT 形式で扱い、カーネルとの入出力や各鍵管理プロトコルにしたがって定数を変換する。

以下は RCT 形式のうち、特別な数字が予約されているリストである。

固定な値

RCT\_BOOL\_OFF = 0

RCT\_BOOL\_ON = 1

RCT\_ADDR\_INET = 0x1000

RCT\_ADDR\_FQDN = 0x2000

RCT\_ADDR\_MACRO = 0x4000

RCT\_ADDR\_FILE = 0x8000

以降は各モジュールについて説明する。詳細な

記述や関数定義は配布予定のキットに含まれる `libracon.txt` を参照されたい。

### 3.4.2 SAD/SPD 管理インタフェース `rcpfbk`

`libracon` は、SAD や SPD を管理する API を提供している。これらの関数はカーネルとのインタフェースとして `PF_KEYv2` を使っている。カーネルからの戻り値は、それぞれのメッセージに対応したコールバック関数を定義することで、アプリケーションがカーネルの戻り値を利用できるようにしている。

また、アプリケーションとパラメータの受渡しは構造体 `rcpfbk_cont` を使い、パラメータの管理を容易にしている。現在の所、これらの API は非同期処理を想定していない。

この API の戻り値として、`sa_src`、`sa_dst`、`sp_src`、`sp_dst` に値が返される場合、これらのポインタは `rcpfbk_cont` 内のバッファを指しているため、値を利用したい場合は呼び出し側でコピーしておくことが望ましい。

### 3.4.3 設定ファイル操作 `rcf`

`racoon2` では設定の煩わしさを軽減させるために設定ファイルを共通化して各デーモンが同じファイルを読み込む。このために `libracon` は、各デーモンが設定ファイルを操作し、設定を参照するための API を提供している。各設定はいくつかの構造体に納められ直接参照できる。また個別の関数でも参照することができる。

### 3.4.4 `spmd` インタフェース `spmif`

`racoon2` では、各デーモンはセキュリティポリシーの設定や削除などを `spmd` に依頼する。`libracon` は、この API を提供している。

コールバック関数を定義することにより、`spmd` からの戻り値をアプリケーションが利用できるようにしている。

### 3.4.5 ログ出力 `plog`

`libracon` はログ出力のための共通関数群 `plog` を提供している。

`plog` は指定したメッセージはメモリダンプを、`syslog()` を使用して出力したり、ファイルや標準出力へ出力したりできる。またインラインで出力先を変更できる。メッセージタグとして以下が用意され

ている。

`RC_LOG_INFO`

[INFO] と表示される。

一般的な情報

`RC_LOG_PROTO_ERR`

[PROTO\_ERR] と表示される。

IKE や KINK などのワイヤに流れるプロトコルのエラー

フォーマットがおかしい

チェックサムがおかしい

認証に失敗

`RC_LOG_PROTO_WARN`

[PROTO\_WARN] と表示される。

ワイヤに流れたプロトコルのエラーで処理を中断しない時

`RC_LOG_INTERNAL_ERR`

[INTERNAL\_ERR] と表示される。

`PF_KEY` や `system call` などの内部処理に関するエラー

`PF_KEY` のエラー

`malloc` に失敗

DB にエントリーがない

`RC_LOG_INTERNAL_WARN`

[INTERNAL\_WARN] と表示される。

内部処理に関するエラーで処理を中断しない時

`RC_LOG_DEBUG`

[DEBUG] と表示される。

デバッグ

`RC_LOG_CRITICAL`

[CRITICAL] と表示される。

`exit(-1)` する状態

### 3.4.6 汎用バッファ `rbuf`

`libracon` は、固定で割り当てられたバッファ領域を順次使用する関数 `rbuf` を提供している。

`rbuf` は、2 つの固定サイズバッファと 1 つの可変長バッファから構成され、初期化時にそのサイズと個数を設定する。

使用する時に必要なバッファサイズに応じた関数を呼び出し、使用後は `free()` する必要はなく順次再利用される。したがって同時に使用される個数をあらかじめ見積もっておく必要がある。

### 3.4.7 ネットワークに関するユーティリティ

libracoon は、ネットワークに関する関数を提供している。racoon2 では、設定ファイルでインタフェースについての IP アドレスを表現するために以下に示す特別な文字列を用意している。以下のプレフィックスに MY\_ が付く文字列はすべて自インタフェースに対して作用する。

表 3.16. アドレス設定マクロ

文字列	アドレス
MY_IP	自インタフェースすべてについている IP アドレス
MY_IPV6	自インタフェースすべてについている IPv6 アドレス
MY_IPV6_GLOBAL	自インタフェースすべてについているグローバルアドレス
MY_IPV6_LINKLOCAL	自インタフェースすべてについているリンクローカルアドレス
MY_IPV4	自インタフェースすべてについている IPv4 アドレス
IP_ANY	:: と 0.0.0.0 が返る

IP\_ANY を除く上記の文字列の直後に文字列 ‘%’ を付加し続けてインタフェース名を指定することで、展開する IP アドレスの範囲を絞ることができる。

libracoon では、この文字列や IP アドレス、FQDN を一元管理するために rc\_addrlist 構造体を使っている。

以下は rc\_addrlist 構造体である。

```
struct rc_addrlist {
    struct rc_addrlist *next;
    rc_type type;
    int port;
    int prefixlen;
    union {
        struct sockaddr *ipaddr;
        vchar_t *vstr;
    } a;
};
type
RCT_ADDR_INET
    IP アドレスまたはネットワークアドレス
```

RCT\_ADDR\_FQDN

文字列

RCT\_ADDR\_MACRO

IP アドレスを示す文字列 (前述)

RCT\_ADDR\_FILE

UNIX ドメインのファイル名

port

type == RCT\_ADDR\_INET の時に a.ipaddr のポート番号がコピーされる。

RCT\_ADDR\_INET の時は a.ipaddr で参照できる。

type がそれ以外の時は a.vstr で参照できる。

### 3.4.8 その他ユーティリティ

その他のユーティリティとして、バージョン番号を返す関数や起動時に表示するメッセージを返す関数が用意されている。

---

## 第 4 章 まとめ

---

racoon2 は、IKEv2 と KINK を用いた基本的な鍵交換が可能である。今後の課題としては、まず IKE を含む未実装となっている機能の実装が挙げられる。また、既存の racoon の設定と racoon2 の設定が大きく異なるために、racoon2 を使いやすくするためには設定ファイルの変換ツールなどを用意する必要がある。

IPsec WG 全体では、アプリケーションから IPsec の状態を参照できる API に関する研究などが考えられるが、当面は引き続き racoon2 の開発を中心に活動を行う。

---

## 付録 Glossary

---

- Authentication Header (AH)  
RFC2402 で定義され、パケットの認証と完全性を保障するための情報を格納したヘッダである。

- Internet Security Association and Key Management Protocol (ISAKMP)  
RFC2408 で定義されるインターネット上で SA と鍵を交換するためのフレームワーク
- Internet Key Exchange version 2 (IKEv2)  
draft-ietf-ipsec-ikev2-xx.txt で定義される。IKE の鍵交換プロトコルの後継プロトコルに必要なメッセージ数などの改善がなされている。
- IP Encapsulation Security Payload (ESP)  
RFC2406 で定義され、ペイロードに対して機密性を提供する。認証と完全性を提供することもできる。
- IP Payload Compression Protocol (IPComp)  
IP レイヤにおいてペイロードを圧縮するためのプロトコル
- Kerberos  
RFC1510 に定義されている認証を行うプロトコル。三者認証を行うために Key Distribution Center (KDC) を導入している。
- Kerberized Internet Negotiation of Keys (KINK)  
ノードの認証に Kerberos を用いた鍵交換プロトコルで、ペイロードに ISAKMP と IPsec DOI を利用している。draft-ietf-kink-kink-xx.txt に定義されている。
- PF\_KEY Key management API, version2 (PF\_KEYv2、PF\_KEY)  
RFC2367 で定義され、socket を持つアーキテクチャにおいて SA を管理するインタフェースを定義している。インタフェースは拡張可能で BSD 系 OS や Linux では、SP も管理できるように拡張されている。
- Security Architecture for Internet Protocol (IPsec)  
RFC2401 で定義され、IP レイヤにおいて認証、機密性、完全性、否認防止などのセキュリティサービスを提供する。
- Security Association (SA)  
単方向の論理的なコネクションでセキュリティサービスを提供するための情報の単位
- Security Association Database (SAD)  
IPsec SA を格納したデータベースで、IP アドレスやセキュリティプロトコル (AH、ESP) から SA を検索する。
- Security Policy (SP)  
Security Policy は、IP アドレスやプロトコルなどに基づいてパケットをフィルタし、そのパケットに対してどのような処理を行うかを定義する。
- Security Policy Database (SPD)  
Security Policy を格納したデータベース
- The Internet IP Security Domain of Interpretation for ISAKMP (IPsec DOI)  
RFC2407 で定義され IPsec の SA と鍵を交換するために ISAKMP を使う際のパラメータの値と意味を定義する。
- The Internet Key Exchange (IKE、IKEv1)  
RFC2409 で定義される RFC2407、RFC2408 を用い 2 つのノード間で IPsec SA を動的に設定するためのプロトコル