卒業論文 1999 年度 (平成 11 年度)

公開鍵基盤を利用した IP Security 自動鍵交換機構の実装と評価

慶應義塾大学 環境情報学部

有賀 征爾

say@sfc.wide.ad.jp

指導教員

村井 純
徳田 英幸
楠本 博之
中村 修

1999 年 1 月 31 日

概要

インターネットが日常的に使われるようになり，通信内容の保護 (暗号化)，通信相手の特定 (認証) が必要とされる状況が増えている．既存のアーキテクチャでは通信の安全性を守るために，ユーザが意識的にセキュリティを実現するためのソフトウェアを利用しなければならなかった．そのため，通信の安全を一元的に実現できず，ユーザに負担をかけていた．そこで，透過的にネットワークのセキュリティを実現する技術である IPSec が開発され，徐々に普及してきた．

IPSec には IKE と呼ばれる自動鍵交換のための機構がある．IKE による認証には電子証明書などを使用する．しかし，現在の IKE では電子証明書を自動的に扱う仕組みが整っていない．

そこで本研究では，既存の自動鍵交換機構に改良を加え，電子証明書を用いる際に公開鍵基盤との連係を行い，電子証明書の効率的な利用を促進する．IKE デーモンの一実装である racoon に改変を加え PKI との連係を実現した．その結果，複数の IKE デーモンから電子証明書を自動的に利用できるようになり，IPSec を利用する際の管理コストを低減できることが確認できた．

Abstract

As the Internet is used commonly in everyday life, it is necessary to protect (cipher) the contents and identify (authentication) the person on the other side. Users had to use security software to secure their communications deliberately with existing architecture. Securing communications was not integrated with system software and users had to work hard. A feasible technology for transparent network security called IPSec was developed and spreading steadily.

IPSec has a function of automatic key exchange called IKE. IKE uses electronic certification for authentication. However, ongoing IKE system cannot handle electronic certification automatically.

This research promotes efficient use of electronic certification by improving existing automatic key exchange system and by using public key infrastructure when using electronic certification. Racoon, which is one of the IKE demon implementation was re-implemented, so that electronic certificate was used automatically by plural daemons. The cost when using IPSec was decreased by this implementation.

目 次

第 1 章　序論 **1**
1.1　背景 1
1.2　本研究の目的 2

第 2 章　現在の技術とその問題点 **3**
2.1　Public Key Infrastructure (PKI) 3
2.2　IPSec 5
2.3　IKE 6
2.4　現在の問題点 7

第 3 章　問題解決 **10**
3.1　問題点の整理 10
3.2　解決へのアプローチ 10

第 4 章　設計 **13**
4.1　設計上の前提条件 13
4.2　システム全体の概要 14
4.3　電子証明書を利用するために必要な情報 15
4.4　電子証明書を利用するために必要な機能拡張 15

第 5 章　実装 **20**
5.1　実装の方針 20
5.2　動作環境 20
5.3　racoon に対する改良 20
5.4　OpenSSL に対する改良 24
5.5　リポジトリの構成 25

第 6 章　評価 **26**
6.1　既存技術との比較 26
6.2　まとめ 27

第 7 章　結論 **28**
7.1　まとめ 28
7.2　本研究の成果 28
7.3　今後の課題 28

謝辞 **29**

参考文献 **30**

図 目 次

1.1 IPSec の優位性 2

2.1 CA , エンドノード , リポジトリ 3
2.2 CA の階層構造 4
2.3 電子証明書の検証 4
2.4 トランスポートモード 6
2.5 トンネルモード 6
2.6 事前共有鍵方式 7
2.7 電子証明書方式 7
2.8 公開鍵暗号方式 7

3.1 IKE と PKI の連係 11

4.1 相互認証 13
4.2 IKE の認証手順 14
4.3 必要な証明書 16
4.4 複数の電子証明書 17
4.5 必要な電子証明書の要求 17

5.1 証明書・CRL の取得 22
5.2 証明書のリストを保持する構造体 22
5.3 Certificate Request Payload 23
5.4 openssl.cnf に対する拡張 25

表 目 次

5.1 OS , IPSec スタック , 各ライブラリ 20
5.2 証明書のリストを保持する構造体のメンバ 22
5.3 Certificate Request Payload 23

6.1 実現可能な機能 26
6.2 電子証明書の取り扱い 26
6.3 運用コストの違い 27

第1章　序論

1.1　背景

ここ数年で，広域情報通信の環境は劇的に変化した．例えば企業では，社内の情報を WWW や E-mail を利用して互いにやりとりしている．また，家庭では WWW を通じてクレジットカードで商品を購入する，といったことが当たり前に行われている．しかし，現在のインターネットではほとんどの通信が平文のまま行われているため，通信の安全性が保証されていない．ここで述べる「安全な通信」とは，以下の条件を満たすものである．

- 通信内容を第三者に傍受されない
- 自分が通信をしようと思っている相手を確実に特定できる
- 送信した情報が途中で改変されない

インターネットのユーザが安全な通信を利用する場合，安全性を提供するアプリケーションや追加的なプロトコルを使用しなければならない．これは，通信の基盤技術である Internet Protocol(IP)[1] が，安全な通信路を提供する仕組みを持たないことに起因している．従って，安全なネットワークの構築の際に，様々なアプリケーションやプロトコルを組み合わせて利用することになり，設定のコストが高い．

既存のインターネットでは，いくつかの方法で安全な通信路を提供している．その代表的な手法が，アプリケーション層におけるデータの暗号化である．利用するアプリケーションやプロトコル毎に通信内容を暗号化するため，送信ノードと受信ノード間に安全な通信路を構築できる．この方法の代表的な例には，Secure Shell(ssh)[2] や，Secure Socket Layer(SSL)[3] を用いたアプリケーションが挙げられる．

しかし，この方法では，そのアプリケーションやプロトコルが暗号化の機能に対応していなければならず，必ずしもすべてのユーザにとって有効とは言えない．ユーザが利用するアプリケーションやプロトコルは多岐に渡る．しかし，そのすべてが暗号化の機能に対応しているわけではなく，ユーザが意図的に暗号化の機能を持ったアプリケーションやプロトコルを選択しなければならないからである．

そこで，セキュリティ機能をより統一的に扱うために，IP レベルでセキュリティ機能を実現する IP Security(IPSec) [4] が開発されている．IPSec ではネットワーク層でセキュリティを実現しているため，上位層のプロトコルによらず透過的に通信の安全を実現する．IPSec を利用することで通信の認証・暗号化を行える．図 1.1 は IPSec とその他のセキュリティプロトコルの違いを簡単に示している．図から分かるように，今まではアプリケーションごとに暗号化などを行ない，通信の安全を守っていた．暗号化通信に対応してないアプリケーションは，安全に通信できなかった．しかし，IPSec を使用することにより，すべてのアプリケーションをまとめて暗号化することができ，一元的に通信の安全を実現する．

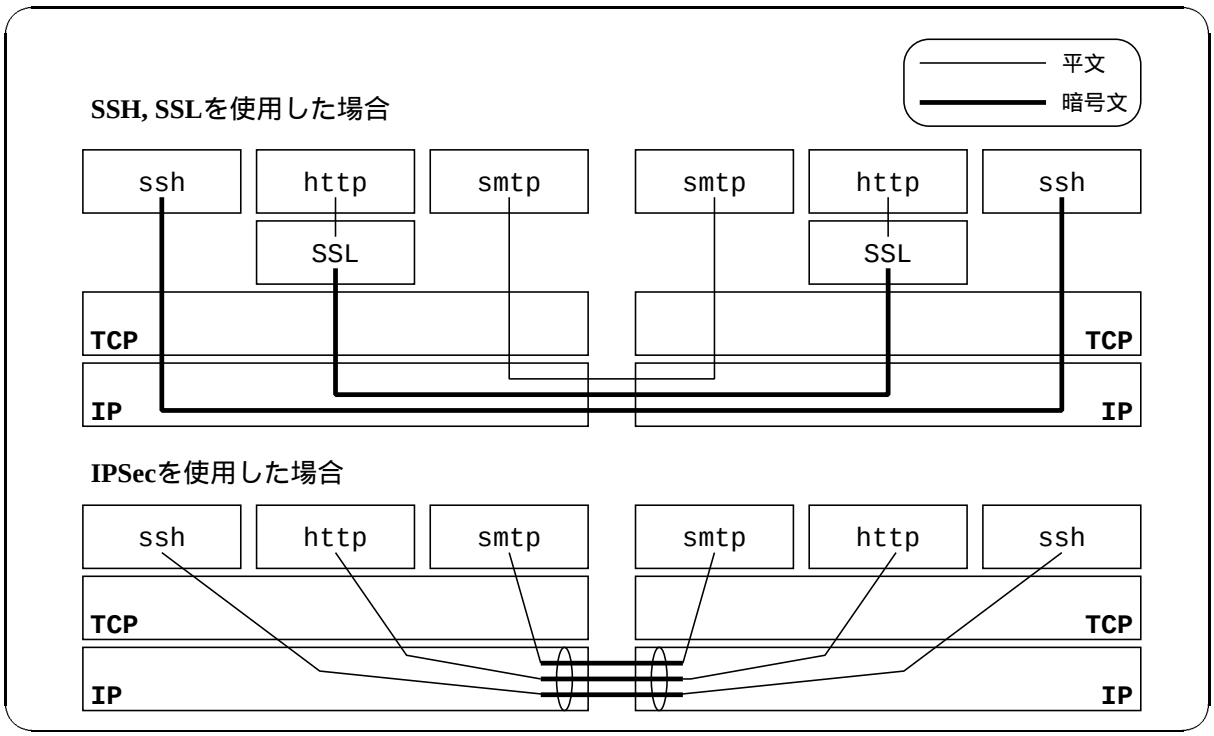

図 1.1: IPSec の優位性

しかし，IPSec を使用する際には通信相手の認証・暗号鍵の設定・その他 IPSec に必要な設定など，様々な設定が事前に必要となる．そこで，これらの設定を自動的に行なうための機構として Internet Key Exchange(IKE) [5] が用意されている．

IKE では，予め設定されているセキュリティポリシーに従い 通信相手の認証，IPSec の設定，暗号鍵の交換などを行なう．IKE の認証には，事前に設定した鍵を使用する方法，電子証明書を使用する方法，通信相手の公開鍵を利用する方法などが用意されている．

1.2 本研究の目的

本研究では以上のような現状を背景とし，IPSec をより簡便に利用するための機構を提案・実装する．具体的には，現在の IKE デーモンに改変を加え，電子証明書を使用する際に，Public Key Infrastracture(PKI) [6] を利用できるようにする．

電子証明書を利用する場合には，証明書が確かに正しく，かつ有効かどうかの確認が必要となる．それを確認するためには Certificate Revocation List (CRL) [6] が必要である．また，通信相手の証明書を IKE の枠組みの中で入手できなかった場合には，別途，証明書を入手する動作も必要となる．

このように電子証明書を利用するには，電子証明書に関するさまざまな処理を行なわなければならない．そのため IKE デーモンが個別に電子証明書の管理を行なっていると，IKE を複数のノードで使用し始めた時に管理コストの増大を招く．

そこで，本研究では IKE から電子証明書の統一的な管理基盤である PKI を利用することで電子証明書の管理コストを低減し，IPSec の導入を容易にする．

第2章　現在の技術とその問題点

ここでは，本研究の基盤となっている既存の技術とその問題点について述べる．

2.1　Public Key Infrastructure (PKI)

PKI とは公開鍵認証基盤である．これは，公開鍵を使った認証のために，CA による電子署名を使って，公開鍵の登録，参照，配布のプロトコルやフォーマットを定めたものである．広く認知された公開鍵のためのインターネット上でのインフラストラクチャを目指しており，IETF の Public Key Infrastructure (X.509) WG で標準化が進められている．図 2.1 は PKI の基本的な構成を示している．

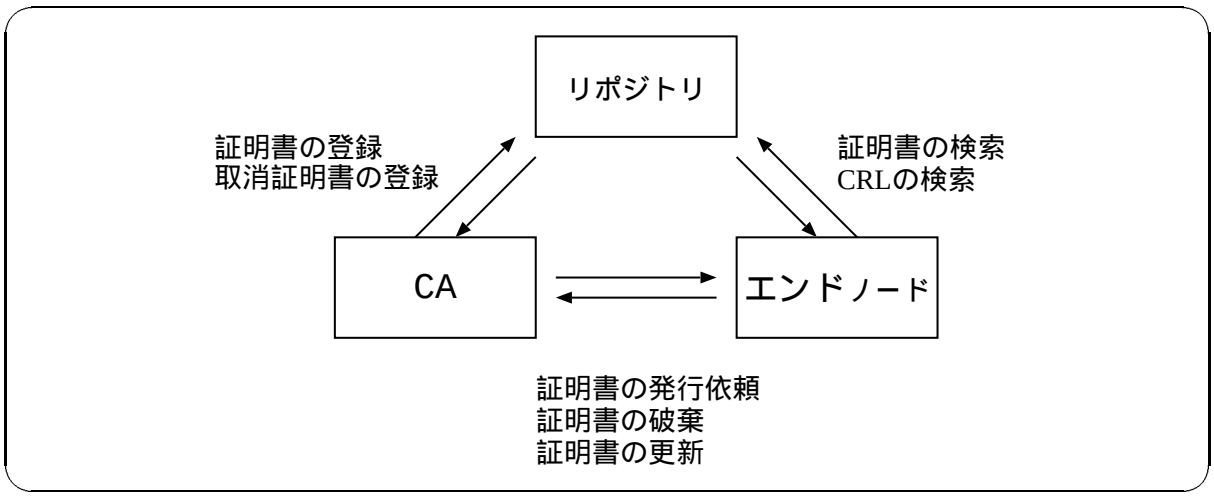

図 2.1: CA，エンドノード，リポジトリ

PKI の基本的な要素について，以下で説明する．

2.1.1　Certification Authority (CA)

CA とは個人や組織を確認して，それらの公開鍵と関連情報を一定のフォーマット (X.509[7] など) で登録する機関またはサーバである．また CA は，第 3 者の依頼に対して，個人や組織の公開鍵を含む情報に CA の電子署名をつけた証明書を提供する．

CA は上位の CA によって認証される階層的な認証関係をとる．階層のトップレベルの CA は自分自身を認証することになり，とくに root CA と呼ばれる．階層構造の一例を図 2.2 に示す．

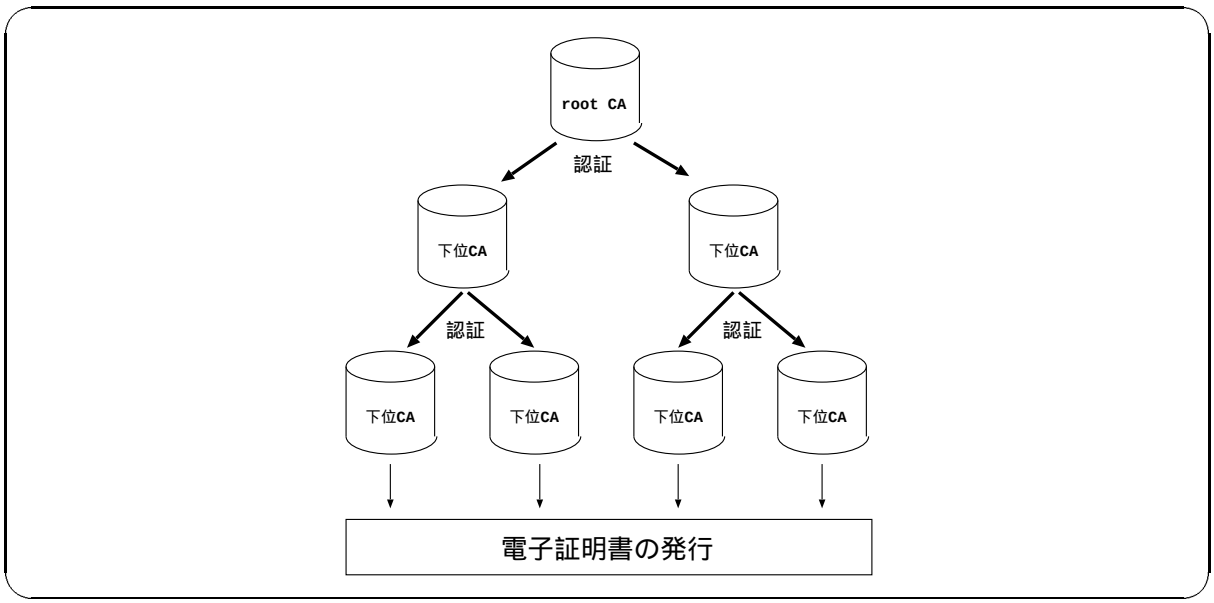

図 2.2: CA の階層構造

2.1.2 電子証明書

電子証明書とは，CA がインターネット上のメールアドレス，サーバ，個人，組織などについて，その存在と内容を証明するために発行する，電子的な情報である．内容を改変から守るために，CA の秘密鍵で署名がされている．したがって，改変のチェックには CA の公開鍵を用いる．

電子証明書の検証の流れを，図 2.3 に示す．検証を行なう電子証明書から，検証者が持っている電子証明書 (通常 root CA の電子証明書) までの CA の連鎖をすべて検証する必要がある．

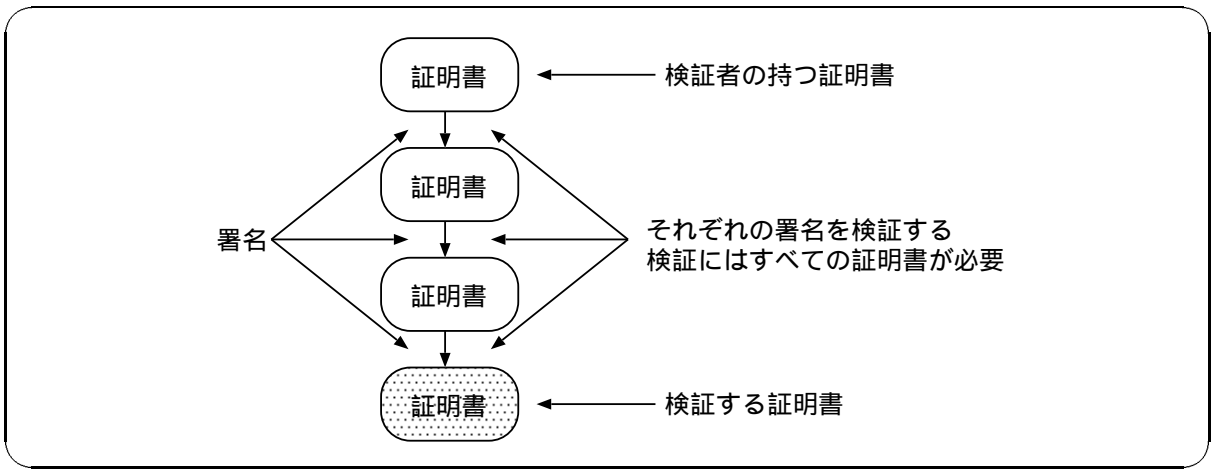

図 2.3: 電子証明書の検証

CA の利用者が証明書の発行を要求すると，CA はその内容を審査し，確かに正しいものであると確認した上で，証明書を発行する．

証明書には有効期間，暗号方式，シリアル番号，バージョン，拡張フィールドなどが含まれている．

2.1.3 Certificate Revocation List(CRL)

破棄証明書リストと呼ばれ，公開鍵の証明を取り消すためのリストである．有効期限の切れたものや，紛失した鍵を取り消すために使われる．X.509v2 に CRL のフォーマットが定められている [6]．

CRL は CA によって発行される．PKI を使うアプリケーションでは，取り扱う証明書の効力を CRL にもとづいて検査できる．

2.1.4 リポジトリ

単体のシステムか分散した複数のシステムからなる．電子証明書や CRL を保管し，PKI を利用するエンドノードにこれらの電子証明書や CRL を配布する役割を担っている．

2.2 IPSec

IPSec には大きく分けて，通信の認証，通信の暗号化，通信の完全性という三つ機能が用意されている．

通信の認証
: 受信ノードがそのパケットが正しく送られてきたかどうかを認証できるように，送信ノードは送信するパケットに Authentication Header(AH) [8] と呼ばれるヘッダを付加し署名する．パケットを受けとった受信ノードは，その署名を検査し，そのパケットの正当性を判断する．もし署名が正当であればそのパケットを次の処理に渡し，もし署名が不当であればそのパケットを破棄する．これにより，正しい通信相手からのパケットのみ受け取ることができる．

通信の暗号化
: 送信ノードと受信ノードの間の第三者によって通信内容を傍受，あるいは改竄されないように，送信ノードは，あらかじめ交換しておいた鍵を使ってそのパケットの内容を暗号化し，Encapsulating Security Payload(ESP) [9] と呼ばれるパケットにカプセル化する．これにより，鍵を持っていない第三者の不正な操作を防げる．

通信の完全性
: 通信の完全性は，認証と暗号化によって実現されている．

また，IPSec はトランスポートモードとトンネルモードと呼ばれる 2 つのモードを持っている．

トランスポートモード
: IPSec を利用するノードが通信相手と直接 IPSec による通信を行なう．具体的な利用形態を図 2.4 に示す．トランスポートモードでは通信を行なうホスト同士が，直接 IPSec により通信の安全を確保する．

トンネルモード
: ホストからルータまでは平文で通信するが，ルータ間で IPSec を利用した通信を行なう．具体的な利用形態を図 2.5 に示す．トンネルモードでは，ホストは直接は IPSec は使用せず，間に存在するルータが IPSec を使用し通信の安全を確保する．

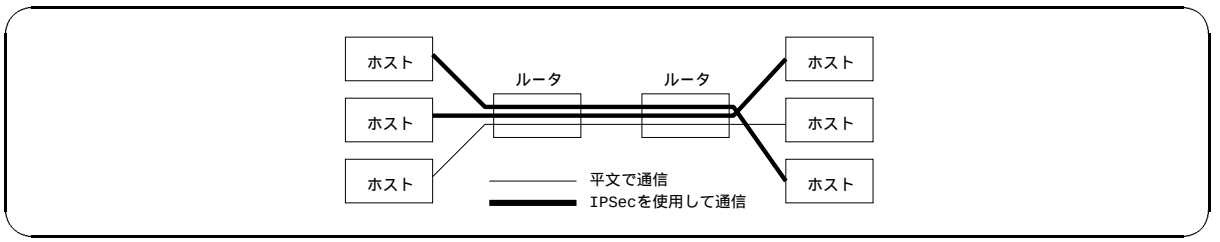

図 2.4: トランスポートモード

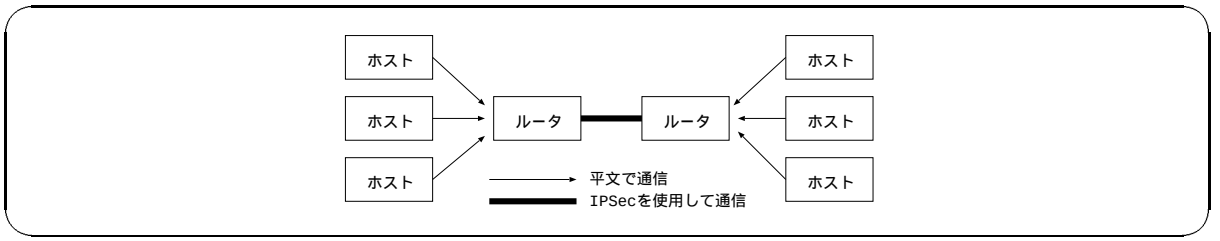

図 2.5: トンネルモード

2.3 IKE

IPSec を使用するには，通信を開始する前には，次の設定が必要となる．

- 通信を認証・暗号化するための暗号方式
- 通信を認証・暗号化するための鍵
- 使用する IPSec のモード

管理者がこれらすべての設定を，通信毎に行なうことは現実的ではない．

そこで，IPSec には IKE[5] と呼ばれる自動鍵交換のための機構が用意されている．IKE を用いることにより上記の設定を自動的に行なえる．IKE を使うためには，IKE 同士の認証が必要になる．認証方法には，以下の方法が用意されている．

事前共有鍵方式
: IKE を利用して通信を開始する前に，お互いのパスワードを設定しておく方式．

  通信を行なう際に，それぞれのパスワードを使用し相手から送られてきた nonce(一度だけ使われる数字) を暗号化し，お互いの nonce を復号化することで認証を行なう．認証の手順を図 2.6 に示す．

電子証明書方式
: X.509 形式の電子証明書を用いて相手の認証を行なう方式．

  電子証明書方式では，事前共有鍵方式と異なり，通信前の鍵の設定などは必要ない．認証はお互いの電子証明書を確認することにより行なわれる．認証の手順を図 2.7 に示す．

公開鍵暗号方式
: お互いの公開鍵を利用して，認証を行なう．この方式では，通信前に通信相手の公開鍵を入手しておく必要がある．

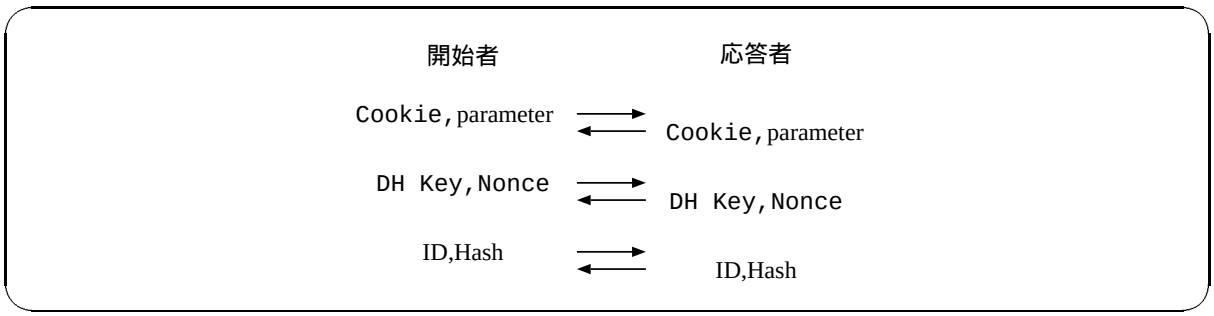

図 2.6: 事前共有鍵方式

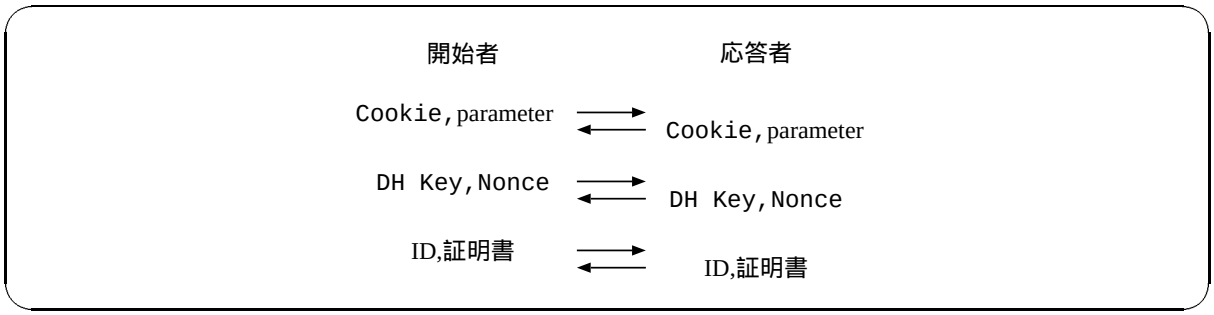

図 2.7: 電子証明書方式

実際の認証は通信相手が nonce を復号化できることを証明することによって行なわれる．認証の手順を図 2.8 に示す．

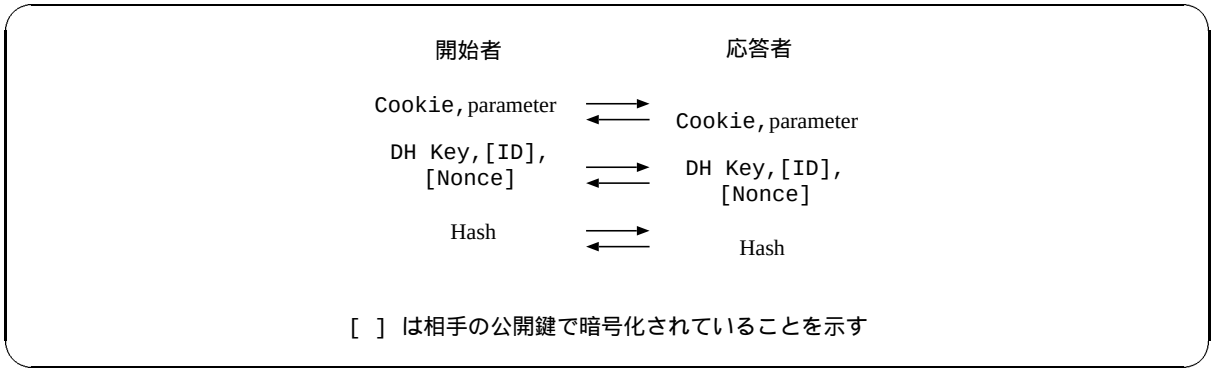

図 2.8: 公開鍵暗号方式

図中の Cookie と DH は IKE の鍵交換をすすめる上で必要となるパラメータである．

2.4 現在の問題点

ここでは，IKE の認証方法における問題点について考察する．

2.3 の説明から分かるように，事前共有鍵方式や公開鍵暗号方式では通信をする前にパスワードの設定や，公開鍵の取得などの作業が必要となる．しかし，電子証明書方式を利用する場合でも，

電子証明書の正当性の検証など，電子証明書を扱う上での様々な作業が発生する．

そこで，以下ではそれぞれの認証方法の問題点を考察する．

2.4.1 事前共有鍵方式

事前共有鍵方式を利用する場合，IKE の運用を開始する前にそれぞれのノードでパスワード (鍵) を設定しなければならない．たとえば，安全性を保つためにすべてのホスト間で異なるパスワードを用いることとする．すると，IPSec を利用するノードが 10 台あった場合，IKE の通信路は双方向であるため，必要なパスワードは，

$$_{10}\mathrm{C}_2 = (10 \times 9)/(2 \times 1) = 45 \text{ 通り}$$

となる．さらにこのパスワードをそれぞれのホストに設定すると，

$$45 \times 10 = 450 \text{ 回}$$

の作業が必要になる．たった，10 台しか IKE を利用するノードがなくても，それぞれ異なるパスワードを利用しようとすると，多数のパスワードが必要となる．したがって，ノードの数が増えれば管理しなければならないパスワードは飛躍的に増えてしまう．その結果，パスワードの変更，同期などの管理の負担が膨大なものになってしまう．

したがって，大規模なネットワークにおいて IKE の認証に事前共有鍵方式を利用することは非現実的である．

2.4.2 公開鍵暗号方式

公開鍵暗号方式を利用する場合には先にも書いたように，通信を開始する前に通信相手の公開鍵が必要となる．これは，信頼できない通信路を用いた場合，正しい公開鍵を任意の時点で手に入れる方法がないからである．

つまり，公開鍵暗号方式を利用して IKE を運用しようと考える場合，通信を行なうすべてのノード間で公開鍵の交換を，事前に，そして安全で確実な方法 (例としては，管理者がオフラインで公開鍵をすべてのホストにコピーするという方法が挙げられる) で行なわなければならない．

また，秘密鍵が外部に洩れてしまった場合の，公開鍵の無効化という作業も必要となる．これも同じように，安全で確実な方法ですべてのノードに対して行なわなければならない．

このように，公開鍵暗号方式を用いた認証は，事前共有鍵方式と同じように，膨大な管理コストがかかるため，大規模なネットワークにおいて利用することは現実的ではないと考える．

2.4.3 電子証明書方式

電子証明書方式を利用する場合，通信相手の電子証明書を取得する必要がある．ただし，一度証明書を取得すればその証明書の有効期限内ならば，再び証明書を取得する必要はない．

パスワードや公開鍵を利用する場合と異なり，IKE の運用を始める前に電子証明書を取得しておく必要はない．なぜなら，電子証明書は CA の証明書で保護されているため，どのようなチャネルで入手してもよいからである．つまり，IKE による通信を開始した後，なんらかの手段によって，相手の証明書を入手すればよいのである．

しかし，電子証明書を利用するには，以下の 2 つの手順を経なければならない．

1. 電子証明書の取得
   何らかの方法で通信相手の電子証明書を入手する．IKE では通信相手に電子証明書を送るよう要求することもできる．

2. 電子証明書の確認
   証明書の正当性の確認を行なう．実際には，

   (a) 有効期限の確認

   (b) 証明書の改変の確認

   (c) CA の連鎖の確認

   (d) CRL の確認

   などの作業を行なう．

このうち，「電子証明書の確認」の部分は，必要な証明書が揃っていれば，現在でも行なうことができる．

第3章　問題解決

3.1　問題点の整理

2.4.1，2.4.2，2.4.3 において，IKE の認証方法の問題点について考察した．
以下に認証方法の違いによる問題点を整理する．

- 事前共有鍵方式，公開鍵暗号方式
  この二方式では，どちらも IKE の運用を開始する前に認証するための設定，または認証鍵の入手を管理者が手動で行なう必要がある．

- 電子証明書方式
  上の二方式と異なり，IKE の運用前に必要な作業は特別にはない．しかし，実際に電子証明書を利用して認証を行なう際の作業が多い．

事前共有鍵方式や公開鍵暗号方式は，管理者が手動で管理する必要があり，それぞれの管理コストも高いため，大規模なネットワークを管理・運用していく上では不適切であると考える．したがって，大規模なネットワークを自動的に管理・運用していくには，IKE の認証方法として，電子証明書を用いた方法が最も適している．しかし，電子証明書を利用する場合でも，電子証明書の利用には管理者の手動の作業を含む，さまざまな作業が必要となる．現在の IKE では電子証明書を使って，通信相手を認証する手順は規定されているが，入手した電子証明書をいかに検証するか，またその検証のための情報をいかに入手するか，といったことは規定されていない．
電子証明書を利用する上で生じる問題を以下に示す．

- 入手した電子証明書をいかに検証するか
  電子証明書の検証にはその電子証明書を発行した CA の電子証明書が必要となるが，どこからどのようにその証明書を入手するかが問題となる．

- CRL をいかに取得するか
  電子証明書は有効期限が切れる前でも，CRL によって無効にされる．したがって，電子証明書の検証は，その証明書の有効期限だけでなく CRL による検証も必要となる．その CRL をどこから，どうやって取得するかが問題となる．

現実的に IKE を電子証明書を用いて運用するには，これらの問題を解決する必要がある．

3.2　解決へのアプローチ

ここでは 2.4 と 3.1 で述べた問題を解決するために IKE から PKI を利用することを提案する．
2.4.3 でも述べたように電子証明書を利用するためには，電子証明書の確認，電子証明書の取得という作業が必要となる．

そこで本研究では，電子証明書による認証を行ない，同時に IKE から PKI を利用することで，IKE の運用コストを下げるという方法を選択した．IKE の運用が容易になれば，IPSec を簡単に利用できるようになり，透過的で安全なネットワークの普及が促進される．

PKI は電子証明書の管理基盤であり，CA による電子署名を使って，証明書の登録，参照，配布などのプロトコルやフォーマットが決まっている．IKE ではこの，電子証明書の参照・配布の部分の機能が欠けており，PKI を利用することによって，その不足部分を補うことができると考える．

IKE から PKI を利用する時の概念図は，図 3.1 の通りである．IKE は CA やリポジトリを使用し，他の IKE と簡単に通信ができるようになる．

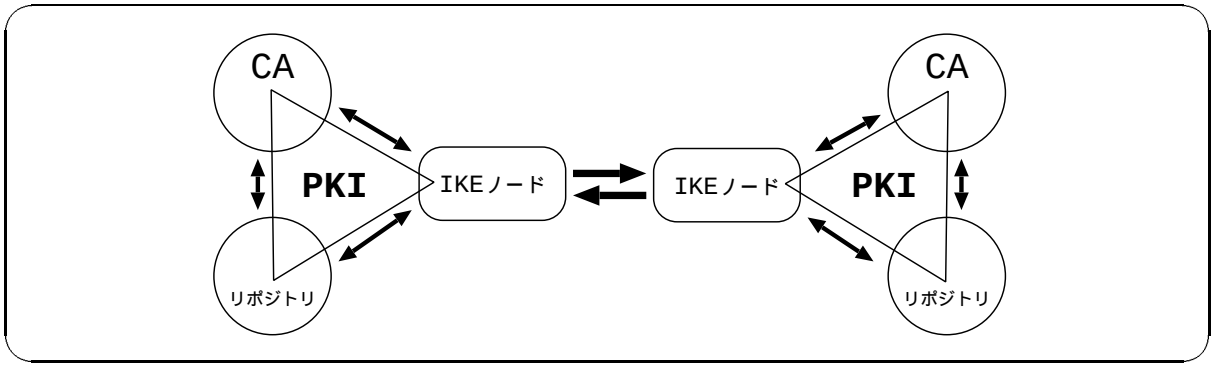

図 3.1: IKE と PKI の連係

3.2.1 PKI を IKE と併用することによる利点

PKI を利用することで，各 IKE ノードが電子証明書を管理する必要がなくなり，管理者の IKE 運用の手間を減らせる．IKE の運用コストが軽減されれば，IPSec をより簡単に利用できるようになる．

また，電子証明書の管理をそれぞれの IKE ノードで行なっている場合，IKE ノードが増えるに従って，加速度的に管理の負担が増る．しかし，PKI を利用することで，大規模なネットワークにおいても，IKE の運用が容易になる．つまり，本研究によって，大規模ネットワークにおいても IKE の実用的な運用が可能となる．

さらに，電子証明書と同時に CRL も利用できるため，より確実な電子証明書の検証を行える．このことは，ネットワークの全体的なセキュリティの向上に役立つ．

3.2.2 PKI を IKE と併用することによる欠点

現在の IKE のモデルでは，通信相手や CA の電子証明書は IKE を実行している同一ホスト内に保存されている．しかし，本研究では，電子証明書 (や CRL) は，PKI を利用して取得する．これは，電子証明書の送受信をネットワークを介して行なうことになり，トラヒックの増加をまねく．

しかし，電子証明書は，一つの証明書で約 1k バイト程度の大きさしかなく，また，IKE の認証は IPSec による通信の開始時に一度だけ行なわれるため，それほどおおきなトラヒックの増加にはつながらない．

また，階層化された CA を認証するには，複数の CA の証明書と CRL が必要となり，認証が終了するまでの時間が長くなる．

現在の IKE では CRL の確認や，CA の連鎖の確認を行なえないので，証明書の確認は簡単なものになる．しかし，確実に有効性を確認するためには，CRL の確認や CA の連鎖の確認まで行なう必要がある．これは求められるセキュリティのレベルにもよるが，やはり電子証明書を安全に使うためには，十分な検証が必要である．

第4章　設計

本研究では，通常単独で動作している IKE を PKI を併用する．これを実現するため，IKE が CA やリポジトリと通信を行なうための機能を設計する．

4.1　設計上の前提条件

システムを設計するにあたり，現実的な制約からの次のことを前提条件とする．

- 各 IKE ノードは，その IKE ノードが電子証明書を取得した CA の上位にある root CA の電子証明書は IKE の運用前に持っている．その取得はオフラインでの受け渡しなどの安全な方法を用いて行なう．

これは，root CA の電子証明書は自己署名されており，安全でない手段で取得した場合，その正当性を確認できないからである．

また，異なる root CA の下にあるホスト間の信頼関係は root CA の相互認証によって実現する．図 4.1 のホスト A がホスト B と通信をする際，相互に認証を行なうためには，CA 'A' と CA 'B' が相互に認証を行なう必要がある．

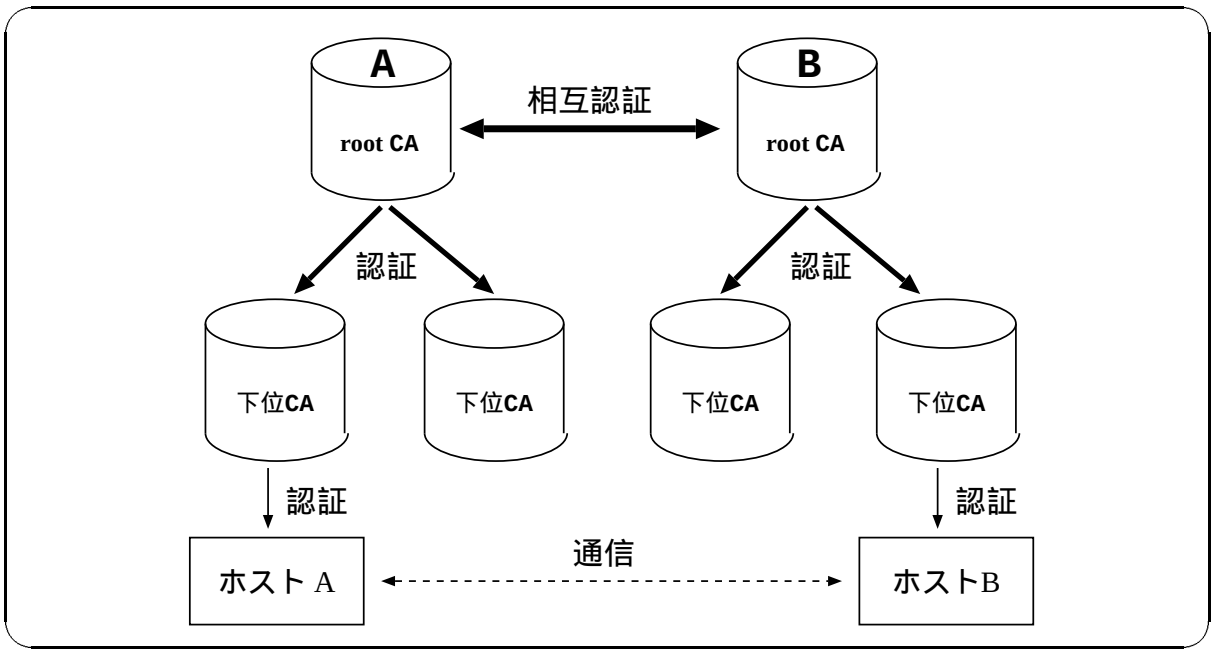

図 4.1: 相互認証

相互認証が行なわれている場合，認証の手順は，同じ CA の木構造内で通信を行なう場合と同じである．従って，以降では特に断わりのない場合，通信は同じ CA の木構造の中で行なわれる

ものとする．

4.2 システム全体の概要

IKE の認証手順を図 4.2 に示す．

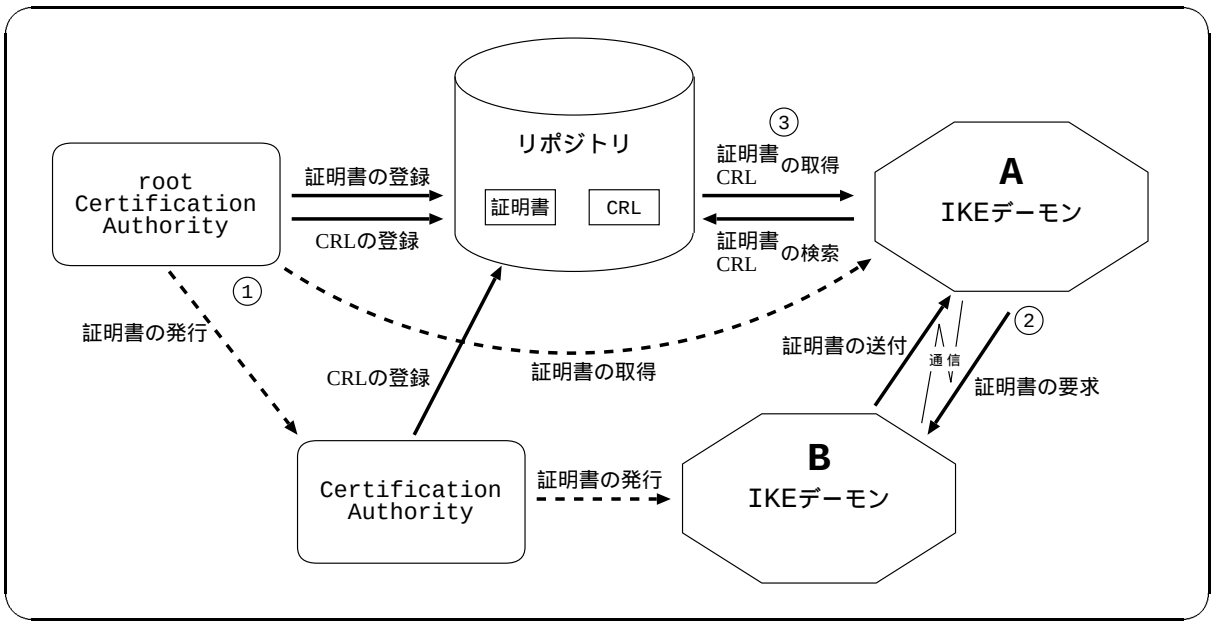

図 4.2: IKE の認証手順

それぞれの動作は以下のように行なわれる．この図の例では，IKE デーモン A が IKE デーモン B を認証することを目的とする．

既存のアーキテクチャでは，IKE デーモン B の電子証明書を確認するために，IKE デーモン A が，IKE デーモン B の電子証明書を認証しているすべての CA の電子証明書を取得していなければならない．本研究では，IKE を使用する際の手間を軽減するため，必要な証明書をリポジトリから取得するように IKE を拡張する．

1. まず IKE の運用を始める前に，破線で示されている，電子証明書の取得・発行を行う．電子証明書の発行は必須であり，root CA の電子証明書の取得の必要性も先に述べた通りである．また，CA は CA に対して電子証明書を発行した場合は，リポジトリにその証明書を登録しなければならない．

2. 次に IKE デーモンが通信を開始する．IKE の通信を開始した際に，IKE デーモン B がホスト B の電子証明書を送ってこなかった場合，IKE デーモン A は電子証明書の送付を要求できる．その際，IKE デーモン A は自分が持つ root CA の証明書をリクエストに含める．これは，IKE デーモン B が自分の持っている証明書のうち，どの証明書を送信すればよいかということを判断する基準に使われる．

3. IKE デーモン B の電子証明書を取得した IKE デーモン A は，次にその証明書の確認を行なう．

4. IKE デーモン B の電子証明書を取得した IKE デーモン A は，その証明書の発行者をキーとして，リポジトリを LDAPv3 [10] を用いて検索する．

5. IKE デーモン B の電子証明書の発行者の電子証明書と，発行者が作成した CRL を取得する．

6. 取得した発行者の電子証明書の発行者が，IKE デーモン A があらかじめ持っていた root CA でない場合は，もう一度，その証明書の発行者をキーとしてリポジトリを検索する．この例の場合は，取得した電子証明書の発行者と事前に持っていた root CA は一致する．

7. リポジトリから取得した証明書と CRL をもとに，IKE デーモン B の証明書の確認を行なう．

8. 確認が成功した場合，認証は完了したことになり，IKE デーモン B との IKE のメッセージの交換を続行する．確認が失敗した場合，IKE デーモン B との通信は終了する．

以上の手順を経ることによって図 4.2 に示した条件下で，IKE デーモン A は IKE デーモン B の認証を行える．

4.3 電子証明書を利用するために必要な情報

電子証明書を確認するために必要となるデータを以下に示す．

- 通信相手の電子証明書
- 通信相手の電子証明書に署名をしている CA の証明書
- 通信相手の電子証明書に署名をしている CA の発行する CRL

また，CA が階層化されている場合には，図 4.3 に示すように，自分が信頼している CA までの CA の電子証明書，もしくは自分が信頼している CA の上位にいる root CA の電子証明書までの一連の証明書がそれぞれ必要となる．

具体的には以下のようになる．

まず，図 4.3 における，例 (1) の場合，自分が信頼している CA の電子証明書が必要となる．必ずしも，root CA の電子証明書は必要ではない．

また，例 (2) の場合は，通信者は自分が信頼している CA の電子証明書と，その CA を認証している root CA の電子証明書が必要となる．

どちらの場合も，自分の信頼する CA の電子証明書，もしくは root CA の電子証明書は，IKE の運用前に取得しておく必要がある．

しかし電子証明書において，root CA の電子証明書だけを PKI の利用前に取得しておけば，その root CA 以下で発行された電子証明書はすべて確認することができる．したがって，本研究では IKE の運用以前に必要な証明書は root CA の電子証明書のみであるとする．

4.4 電子証明書を利用するために必要な機能拡張

次に本システムにおいて必要とされる，各部分にたいする機能拡張を説明する．

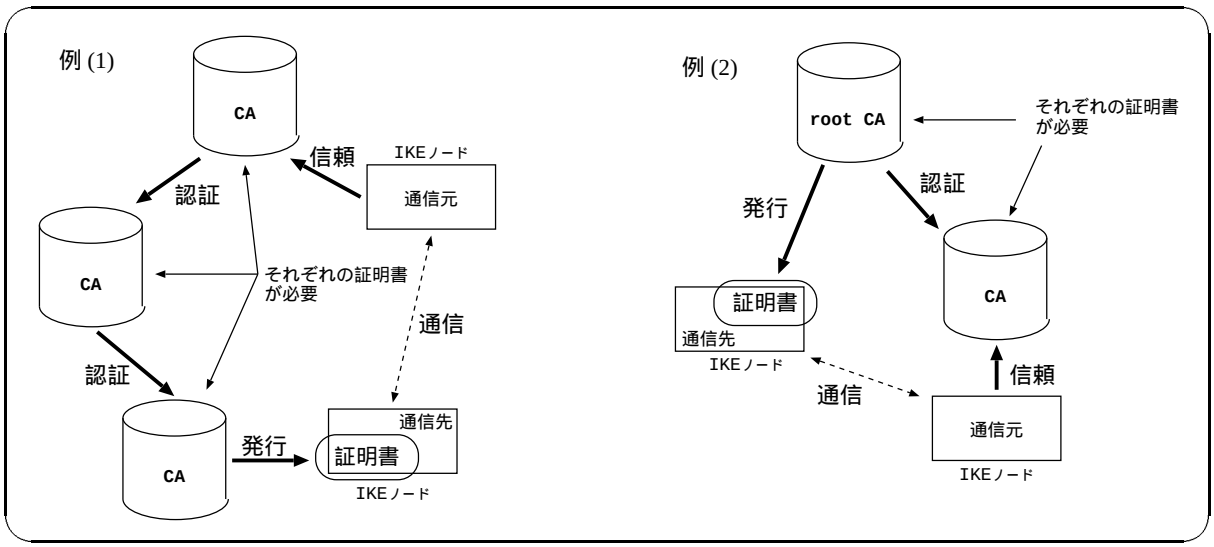

図 4.3: 必要な証明書

4.4.1 リポジトリ

リポジトリは，電子証明書・CRL の所在地を登録してある LDAP サーバと，電子証明書・CRL の実体が置いてあるファイルサーバから構成される．

LDAP サーバには X.509 のサブジェクト (電子証明書の鍵を持っている実体の X.500 の名前) を属性として，電子証明書・CRL のデータベースを構築する．それぞれのエントリの持つデータとして，電子証明書・CRL の実際を示す Uniform Resource Identifier(URI) [11] を格納する．

電子証明書・CRL の実体の取得には File Transfer Protocol(FTP) [12] を利用する．

4.4.2 IKE デーモン に対する機能追加

4.2 で述べた機能を実現するために，既存の IKE に次の機能を追加する必要がある．

1. 必要な電子証明書を決定する機能

   実際のネットワーク環境では，一つの IKE ノードが複数の CA から認証されている場合がある．その場合，IKE ノードは複数の電子証明書を持つ．

   図 4.4 を見るとわかるように，認証の際に必要となるのは，同じ CA の木構造内で発行された電子証明書である．

   電子証明書の有効性は，最終的に root CA まで電子証明書をたどることで確認している．root CA が異なる場合には，これが不可能となり，電子証明書の有効性を確認できない．そこで IKE に，同じ root CA の元で発行された電子証明書を通信相手に要求する機能を追加する必要がある．

   そのため，事前に保持している root CA の電子証明書のリストを作成し，そこから発行者の DN (Distiguished Name) のリストを作成する機能を追加する．証明書を要求する際には，RFC 2408[13] で規定されている，" Certificate Request Payload" の"Certificate Authority" フィールドにこのリストを挿入し，相手に送付する．

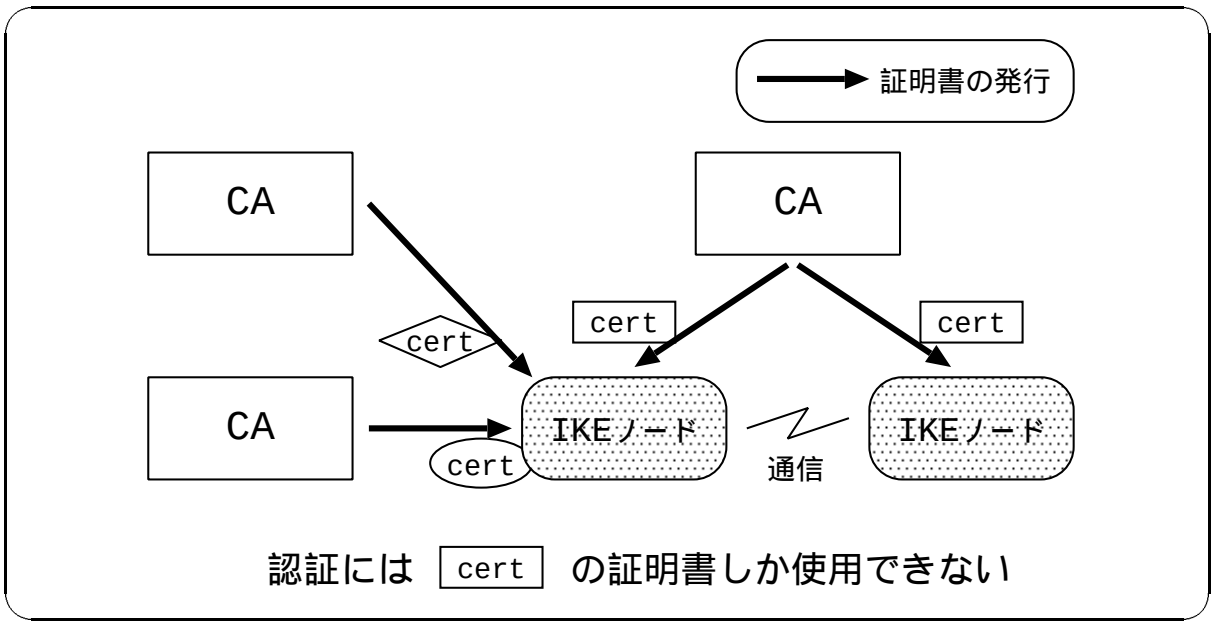

図 4.4: 複数の電子証明書

また，電子証明書の要求を受けた側では，事前に持っている root CA の電子証明書と自らの電子証明書の対応表を持ち，要求された root CA に対応する自らの電子証明書を選択する機能も追加する．

図 4.5 に以上の手順を図説する．

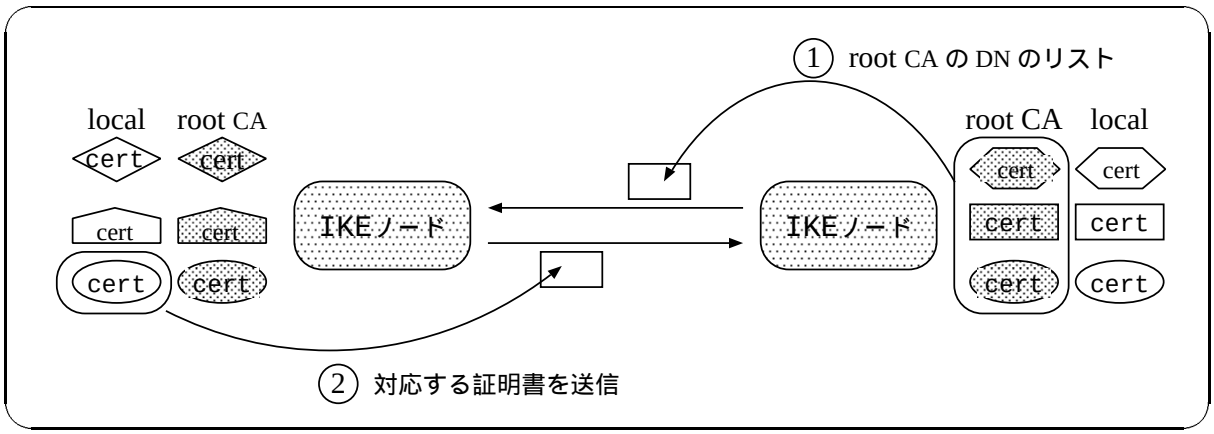

図 4.5: 必要な電子証明書の要求

もし，対応する電子証明書を持たない場合，共有する root CA が存在しないので，IKE における電子証明書を用いた認証は不可能となる．

2. 検索するリポジトリを決定する機能

共有する root CA があり，電子証明書の返送を受けることができた場合，その電子証明書を発行した CA から root CA までの，すべての CA の電子証明書が必要となる．

そのためにまず，IKE デーモンは電子証明書を検索するリポジトリを決定しなければなら

ない．そこで，IKE デーモンに電子証明書の”Authority Information Access” を利用するための機能を追加した．”Authority Information Access” は RFC 2459[6] で規定されており，その証明書を発行した CA の情報の所在地とアクセス方法が書かれている．本研究ではここに CA の電子証明書の URI を記録し，CA の電子証明書の取得に利用する．

3. 電子証明書，CRL をリポジトリから取得する機能

   通信相手の電子証明書を管理している CA の発行した CRL と，1 で取得したそれぞれの CA の CRL が必要である．同じ RFC 2459 において，”CRL Distribution Point” が規定されており，CRL の URI が書かれている．そこで，IKE からこの”CRL Distribution Point” を利用して，CRL を取得する機能を追加する．

   次に，実際に CA の電子証明書，CRL をリポジトリから取得する機能について説明する．

   まず初めに，通信相手から取得した電子証明書の発行者である CA の電子証明書を，リポジトリから取得する．検索すべきリポジトリの選択については，2 において述べた．リポジトリへのアクセスには LDAP v3 を使用する．リポジトリでの検索は，電子証明書のサブジェクトに含まれる値をキーとして検索を行なう．LDAP は X.500 モデルをサポートするディレクトリに対するアクセスを提供するために設計されており，X.509 にもとづく電子証明書と親和性が高いと考えたため，リポジトリへのアクセスプロトコルとして選択した．

   検索が成功した場合，その検索結果には，電子証明書または CRL の実体の所在地を示す URI が含まれる．IKE デーモンはその URI に従って，電子証明書・CRL を取得する．

必要な電子証明書・CRL が集まったら，それらをもとに通信相手の電子証明書の検証を行なう．

4.4.3 CA に対する機能追加

既存の OpenSSL[14] を用いた CA に対する機能追加について述べる．

- 証明書・CRL をリポジトリに登録する機能

  現在の CA では，発行した電子証明書は，そのコピーをローカルなディスクにコピーしておくだけである．したがって，CA の外からその証明書にアクセスすることは難しい．

  そこで，電子証明書や CRL を発行する際に，事前に設定をしておいたリポジトリに，電子証明書を登録する機能を追加する．

  この機能は，以下の 2 つの機能から成り立っている．

  - LDAP を用いて，登録する電子証明書や CRL の属性情報をリポジトリの LDAP サーバに登録する機能
  - FTP を用いて，電子証明書や CRL の実体を，リポジトリのファイルサーバに登録する機能

  このとき登録するリポジトリの所在地は，管理者が設定ファイルに記述することで指定する．

- 発行する証明書に必要な属性をつける機能

  4.4.2 で述べたように，本研究では IKE デーモンが電子証明書を検索するリポジトリを決定するため，電子証明書の”CRL Distribution Point” と”Authority Information Access” を利

用している．通常，これらの属性は，発行される電子証明書や CRL に付加されないため，常にこれらの属性を付加する機能が必要となる．本研究では電子証明書や CRL の発行に OpenSSL のコマンドを用いている．これを拡張し 4.4.3 で述べた設定ファイルに含まれるリポジトリの所在地を，常に電子証明書・CRL に付加するようにした．

第5章　実装

本章では，4章で述べた設計にもとづいて実装したIKEとPKIを連係する機能を説明する．

なお今回はracoonを改良する形で実装を行なったため，IKEの仕様書では決まっているがracoonで実装されていない機能のうち，必要な機能の実装も合わせて行なった．

以下に，IKEの仕様書にあるがracoonでは実装されていない機能のうち，本研究に必要な機能を示す．

- 電子証明書送付の要求
- 電子証明書の送付
- 取得したCRLを利用しての電子証明書の確認

5.1　実装の方針

IKEに対する機能追加は，KAMEに含まれるIKEデーモンであるracoonを改良する形で行なった．また，CAに対する機能追加は，OpenSSLに含まれるCAの管理機能に，必要な機能を追加する形で行なった．LDAPサーバは，OpenLDAP Foundation[15]の，実装を使用した．

5.2　動作環境

実装を行なった環境を表5.1に示す．

表5.1: OS，IPSecスタック，各ライブラリ

OS	FreeBSD 3.4
IPSecスタック	KAME-20000110-snap
OpenSSL	OpenSSL 0.9.4
OpenLDAP	OpenLDAP 1.2.9

5.3　racoonに対する改良

以下ではracoonに追加した機能ごとに説明を行なう．

5.3.1 検索するリポジトリの決定

通信相手から取得した電子証明書の特定の属性から，CA や CRL のリポジトリの所在を特定する．

- 電子証明書のリポジトリの特定
  RFC 2459 に規定されている電子証明書のフォーマットには，X.509 v3 用の拡張が示されている．そこには，PKI で使用するための拡張項目が含まれる．その中に Authority Information Access と呼ばれる項目がある．さらに，この項目には，AuthorityInfoAccessSyntax と呼ばれるものが含まれており，AuthorityInfoAccessSyntax には，accessMethod と accessLocation という属性が含まれている．ここで本実装では，accessLocation を利用し，その証明書を発行した CA の所在を URI を用いて保持する．
  具体的には，LDAP を用いて証明書の所在地を配布するための例を以下に示す．

```
accessLocation = ldap://ldap.sfc.wide.ad.jp:389/C=JP,
ST=Kanagawa,O=Keio%20Univ,OU=EnvInfo,CN=ARIGA%20Seiji,
SN=0?Certificate
```

  この URI にアクセスすることで，上記の識別名をもつ CA の証明書を手に入れる．同じ識別名で複数の証明書を発行することがあるため，識別名の最後に Serial Number (SN) を付加している．また，フィルタを適用し，電子証明書を選択している．

- CRL のリポジトリの特定
  RFC 2459 で規定されている．CRL Distribution Points には，CRL の配布場所を指示する属性がある．
  この属性は，cRLDistributionPoints と呼ばれ，CRL の配布場所を URI で指定する．例を以下にあげる．

```
cRLDistributionPoints = ldap://ldap.sfc.wide.ad.jp:389/
C=JP,ST=Kanagawa,O=Keio%20Univ,OU=EnvInfo,CN=ARIGA%20
Seiji,SN=0?CRL
```

  電子証明書と同じように，識別名の最後に，SN を付加している．また，フィルタを適用し CRL を選択している．

5.3.2 電子証明書・CRL の取得

5.3.1 で決定されたリポジトリから，実際に電子証明書を取得する．

まず初めに，リポジトリから電子証明書または CRL の所在を示す URI を取得する．その URI から電子証明書または CRL を取得する．

ファイルの取得には FTP を用いる．取得したファイルは，racoon の設定ファイルに従い，ローカルのストレージに保存する．

手順を図 5.1 に図示する．

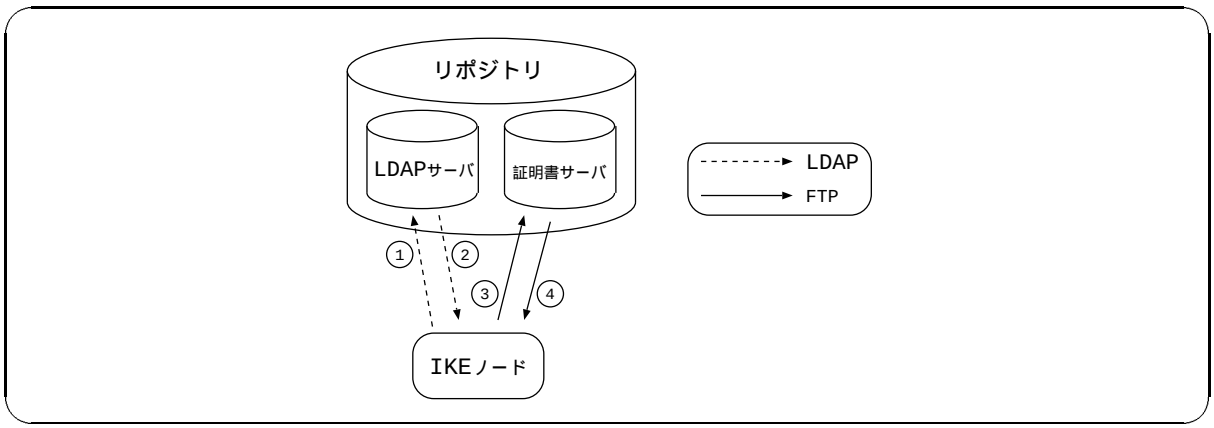

図 5.1: 証明書・CRL の取得

5.3.3 電子証明書のリストの作成

本実装では，IKE デーモンが保持している root CA の証明書のリストを作成した．リストは以下の手順によって作成した．

1. 証明書を保存しているディレクトリにあるファイルをすべてリストアップする．
2. それぞれのファイルの識別名をリストに保持する．

このリストを保持する root_cert_list 構造体の説明を表 5.2 に示す．このリストは，racoon が起動したときに作られ，racoon が終了するまで保持される．

```
struct root_cert_list{
    char *dn_ptr;
    struct root_cert_list *next_cert_ptr;
};
```

図 5.2: 証明書のリストを保持する構造体

表 5.2: 証明書のリストを保持する構造体のメンバ

メンバ名	変数型	説明
dn_ptr	char *	root CA の識別名
next_cert_ptr	struct root_cert_list *	次の構造体へのポインタ

5.3.4 Certificate Request Payload

この機能は IKE の仕様で規定されているが，racoon には実装されていない．しかし，本研究で必要であるため，実装を行なった．

Certificate Request Payload のフォーマットを図 5.3 に示す．

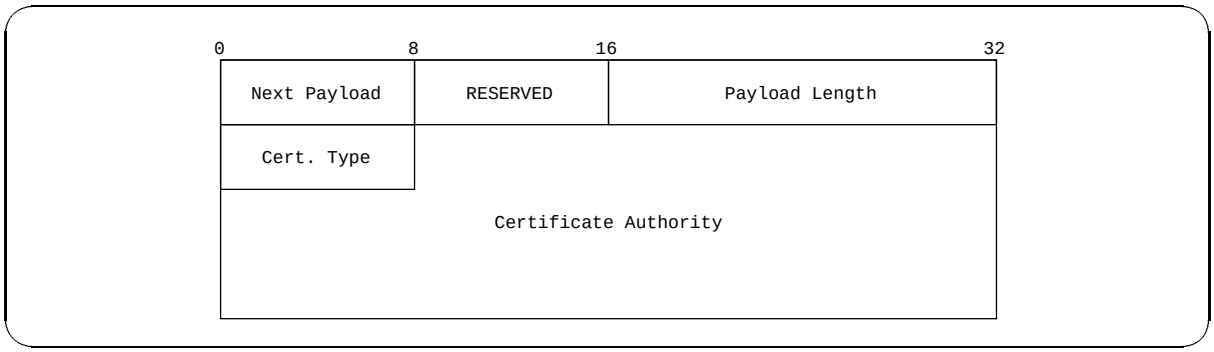

図 5.3: Certificate Request Payload

それぞれのフィールドの内容は，表 5.3 に示す．

表 5.3: Certificate Request Payload

フィールド名	説明
Next Payload	次にくるペイロードを示す
Payload Length	ペイロードヘッダも含めた現在のペイロードの大きさを示す
Cert. Type	証明書のエンコーディングタイプを示す
Certificate Authority	受信者はここに含まれる CA が発行した証明書を返すことが期待される．本実装では，5.3.3 で作成した，CA の識別子のリストを挿入する．

5.3.5 適切な証明書の返送

Certificate Request Payload を受けとった時に，適切な電子証明書を返す機能を実現する．

5.3.3 で作成したリストを検索し，要求された識別子を持つ証明書を Certificate Payload として送信する．Certificate Payload は RFC 2408 で規定されている．要求された識別子をもつ証明書がローカルになかった場合は，リストにあるすべての証明書を返す．また，Certificate Request Payload の Certificate Authority フィールドが NONE だった場合も，リストにあるすべての証明書を送信する．

これは以下の考えにもとづく．

- どの証明書を送ればよいかの判断基準がないため，最も処理を継続できる可能性の高い「すべての証明書を送信する」という方法を選んだ．

- 電子証明書はファイルサイズが小さい，また，一つの IKE ノードが持つ証明書は一般的に少なく，ネットワークのオーバーヘッドにならないと考えられる．

5.3.6 CRL の検証機能

現在の racoon では CRL の検証を行なっていないため，CRL を確認する機能を実装した．
CRL の確認には，OpenSSL ライブラリに含まれる，X509_CRL_verify という関数を用いた．

5.3.7 CA の連鎖の検証

現在の racoon ではただ一段上の CA の電子証明書でしか，証明書の確認ができない．これを改善するため，OpenSSL ライブラリに含まれる，X509_verify_cert_chain という関数を用いて，CA の連鎖の確認も行なうようにした．

5.3.8 証明書のキャッシュ

通信相手の電子証明書を認証した CA の電子証明書を持っていなかった場合，リポジトリからその証明書を取得する．本実装では，その証明書はローカルのストレージに保存しておく．これは，ローカルのストレージに証明書をキャッシュしておくことで，リポジトリへのアクセスなど余計なトランザクションを減らすことができ，IKE 全体でのパフォーマンスの向上が見込めるからである．電子証明書には有効期限があり，また，本実装では CRL も確認しているため，電子証明書をキャッシュすることによる，セキュリティホールなどは発生しないと考える．

5.4 OpenSSL に対する改良

5.4.1 発行した証明書・CRL の登録

本研究では CA は発行した証明書や CRL をリポジトリに登録する必要がある．

- ファイルサーバに CA の電子証明書や CRL のファイルを登録する
- LDAP サーバ，登録したファイルの URI を，自分の識別子をキーとして登録する．

そこで，URI を指定すると，その CA の識別子をキーとして，証明書・CRL を設定ファイルにあるリポジトリに登録する LDAP クライアントを実装した．
証明書や CRL を発行すると，発行スクリプトから FTP が呼ばれ，その証明書をリポジトリのファイルサーバに登録し，同時に LDAP クライアントを実行し，そのファイルの URI を LDAP サーバに登録するようにした．
また，5.3.1 で X.509 の拡張を使っているため，証明書・CRL の発行時に必ずその拡張を含めるように，OpenSSL のツールに改良を行なった．

5.4.2 設定ファイルへの記述の追加

OpenSSL の設定ファイルである openssl.cnf に拡張を行ない，IKE デーモンが使用するリポジトリを指定できるようにした．

```
LDAPServer = ldap://ldap.sfc.wide.ad.jp:389/
CertServer = ftp://ldap.sfc.wide.ad.jp/pub/cert/
```

図 5.4: openssl.cnf に対する拡張

5.5 リポジトリの構成

リポジトリは , LDAP サーバとファイルサーバ (証明書サーバ) からなり , LDAP と FTP でアクセスされる . それぞれ , 通常の LDAP サーバと FTP を実行しているだけである .

第6章　評価

6.1　既存技術との比較

本研究による安全なネットワークの実現・運用のためのコストを評価するために，既存の技術との比較を行なう．

1. 既存の技術と本研究によって実現されるネットワーク環境を比較する．

   表 6.1 に，それぞれのセキュリティ技術で実現可能な機能についてまとめる．

表 6.1: 実現可能な機能

	ユーザ認証	ホスト認証
SSH	○	×
SSL	○	○
IPSec+IKE	○	○
IPSec+IKE+PKI	○	○

これから分かるように，アプリケーションでセキュリティを実現している SSH 以外は，すべてホスト単位の認証を行える．これは，アプリケーションがユーザレベルで実行されており，ホストのアクセス制御には関与しないためである．

また，ホスト単位の認証を行なっている，それぞれの技術は，どれも電子証明書を用いて，アクセス制御を行なっている．

そこで表 6.2 に，電子証明書を利用する上で必要な機能に，それぞれのセキュリティ技術が対応しているかどうかをまとめた．

表 6.2: 電子証明書の取り扱い

	証明書を用いた認証	CA の証明書の取得	CRL の取得
SSL	○	×	×
IPSec+IKE	○	×	×
IPSec+IKE+PKI	○	○	○

これまで本論文で述べてきたように，電子証明書の正当性を検査するには，CA の証明書や CRL の利用が必要となる．したがって，電子証明書を利用するノードはこれらの情報をなんらかの手段しなければならない．しかし，表 6.2 から分かるように，既存のセキュリティ

技術では CA の連鎖や CRL などは考慮していない．電子証明書の正当性の検査は，証明書を利用する上で最も基礎的で重要な手順であるが，かならずしも厳密には行なわれていないことがわかる．

2. 運用のコストについて考える．

   例として，電子証明書を利用しているネットワークに新しく，CA を追加することを考える．

表 6.3: 運用コストの違い

既存の技術を利用した場合	その CA が証明書を発行したホストと通信をするホストにすべて，手動でその CA の証明書を置かなければならない	管理コストが高い
本研究を利用した場合	その CA の証明書を一度リポジトリに登録すれば，証明書が必要なホストはリポジトリにアクセスするだけでよい．	管理コストが低い

大規模なネットワークにおいては，新規 CA の設立や，CA の廃棄などがしばしば発生する．そのたびに，その CA の電子証明書の配布・破棄がすべてのホストについて必要となると，膨大な管理コストが発生する．

本研究では，証明書の登録・配布をリポジトリで一元的に管理することにより，この問題を解決した．

6.2 まとめ

以上より，以下の結果が得られた．

- 電子証明書を一元的に管理することができた．
- リポジトリとのトランザクションは，大きいものではない．

したがって，本研究によって IKE の管理コストが低減され，安全なネットワークをより簡単に利用できることが実証された．

第7章　結論

7.1　まとめ

本研究では，IKE の認証に電子証明書を利用する際，IKE から PKI を利用するための機構の実装・評価を行ない，IKE の運用コストの軽減を実現した．

これからネットワークに対するセキュリティの要求はますます増えていく．IPSec は通信の安全を実現する機構としてその重要性が高くなっていく．IPSec の自動鍵交換の仕組みとして IKE が用意されている．IKE の認証方法にはいくつかあるが，大規模ネットワークにおける運用コストの面から，電子証明書を利用した方法が有効である．電子証明書を利用するには，証明書の正当性の検証が必要となる．検証には CA の証明書や CRL などの情報が必要となる．

そこで，本研究では，IKE デーモンを改良し，電子証明書を扱う際に PKI を利用するようにした．PKI を利用することで，電子証明書の検証に必要な情報を自動的に取得することが可能になる．その結果，IKE の運用コストを下げることができ，IPSec を簡単に利用できるようになった．

また PKI を利用することで，電子証明書の検証に十分な情報を取得できた．そのため，電子証明書の正当性を確実に検証できた．その結果，電子証明書の信頼性が向上し，ネットワークの安全性を高められた．

7.2　本研究の成果

本研究の成果としては，以下のものが挙げられる．

- IKE の管理コストの軽減電子証明書取得を自動化することにより，IKE の管理コストを下げることができた．
- より安全な電子証明書の運用 CA の連鎖や，CRL の確認を行なうことで，より確実に電子証明書の正当性を検証できた．

7.3　今後の課題

本研究における今後の課題には，以下のものがあげられる．

- Certificate Management Protocol(RFC2510[16]) への対応
  IETF の PXIX WG で議論されている，電子証明書の管理プロトコルである，Certificate Management Protocol への対応
- セキュリティポリシー管理の統合
  現在の実装では，LDAP サーバを電子証明書・CRL の所在を示すものでしかないが，その他にセキュリティポリシー情報なども同時に扱うことにより，より一元的な IKE の管理を行なえる．

謝辞

本研究を進めるにあたり，暖かい御指導を頂いた，慶應義塾大学環境情報学部の村井純氏，徳田秀幸氏，中村修氏，楠本博之氏に感謝します．

また，本研究に対して，細部にわたる指導と助言を頂いた株式会社東芝研究開発センターのみなみまさき氏，慶應義塾大学政策・メディア研究科の関谷勇司氏，土井裕介氏，楠根雄志氏，増田康人氏，小原泰弘，中前周氏，三谷洋司氏に氏感謝します．

論文執筆に際して，多大なる支援をいただいた，慶應義塾大学総合政策学部の三川荘子氏，慶應義塾大学環境情報学部の三屋光史朗氏，服部亜紀子氏に感謝します．

そして，様々な激励と助言を頂いた，WIDE プロジェクト IPv6 WG の皆様，慶應義塾大学村井・徳田・楠本・中村研究会の SING の皆様と研究会の諸氏に感謝します．

快適な論文執筆環境を IPv6 で提供してくれた fm.v6.sfc.wide.ad.jp に感謝します．

最後になるが，最愛なる表現者，椎名 “サディスティック” 林檎姫に感謝と敬意の念を表する．

参考文献

[1] J. Postel. *Internet Protocol.* RFC 791, September, 1981.

[2] T. Ylonen, T. Kivinen, M. Saarinen, T. Rinne, and S. Lehtinen. *SSH Protocol Architecture.* Internet-Draft, June, 1999.

[3] Paul C. Kocher Alan O. Freier, Philip Karlton. *The SSL Protocol Version 3.0.* Internet-Draft, November, 1996.

[4] R. Atkinson S. Kent. *Security Architecture for the Internet Protocol.* RFC 2401, November, 1998.

[5] D. Carrel D. Harkins. *The Internet Key Exchange (IKE).* RFC 2409, November, 1998.

[6] R. Housley, W. Ford, W. Polk, and D. Solo. *Internet X.509 Public Key Infrastructure Certificate and CRL Profile.* RFC 2459, January, 1999.

[7] *Information Technology - Open Systems Interconnection - The Directory: Authentication Framework.* ITU-T Recommendation X.509, June, 1997.

[8] R. Atkinson S. Kent. *IP Authentication Header.* RFC 2402, November, 1998.

[9] R. Atkinson S. Kent. *IP Encapsulating Security Payload (ESP).* RFC 2406, November, 1998.

[10] M. Wahl, T. Howe, and S. Kille. *Lightweight Directory Access Protocol (v3).* RFC 2251, December, 1997.

[11] T. Berners-Lee, R. Fielding, and L. Masinter. *Uniform Resource Identifiers (URI): Generic Syntax.* RFC 2396, August, 1998.

[12] A.K.Bhushan. *File Transfer Protocol (FTP).* RFC 414, December, 1972.

[13] D. Maughan, M. Schertler, M. Schneider, and J. Turner. *Internet Security Association and Key Management Protocol.* RFC 2408, November, 1998.

[14] OpenSSL. *http://www.OpenSSL.org/.* 20 January, 2000.

[15] OpenLDAP. *http://www.OpenLDAP.org/.* 5 January, 2000.

[16] C. Adams and S. Farrell. *Internet X.509 Public Key Infrastructure Certificate Management Protocols.* RFC 2510, March, 1999.