第 10 部

セキュリティ

第 1 章

はじめに

近年のインターネットの急速な拡大にともなって、ネットワークの利用形態や構成は大きく変化し続けている。特に商用インターネットの登場によって、従来の学術研究目的でのネットワーク利用だけでなく、さらに商業目的でのネットワーク利用が必然的に増加している。また、インターネットへの参加も、インターネットサービスを提供する通信サービス会社よりサービスを購入することで実現できるようになった。これにより、インターネットは、従来と比較してより多くの組織、人々を相互接続するネットワークとして発展を続けている。このような発展にともない、セキュリティの問題も大きくなってきている。特に、インターネットが商業活動に利用された場合、利用者の認証、アクセス制御、通信内容の保護といった機能を持ったアプリケーションが求められるようになる。このようなことを背景として、WIDE Project Security Working Group では、インターネット環境で求められるセキュリティ機能に注目し、それらを実現することを具体的な方法として、研究を進めている。

1993 年度における Security Working Group の活動では、特にインターネット環境における電子メールでのセキュリティ機能の実現に注目して活動を進めてきた。インターネットにおける電子メールは、基本的に内容を保護するような機能を有していない。現在の電子メールシステムでは、インターネット内を経由して配送される時、電子メールの本文を暗号化したりすることはなく、全てが平文のままで配送されている。このため、電子メールの配送中にメールの中身が漏洩する可能性が問題点として以前より指摘されていた。この問題を解決すべく、FJPEM と呼ぶシステムを本 WG では開発し、現在国内インターネットでの運用実験を進めている。FJPEM は、従来より提案されている PEM (Privacy Enhanced Mail) の機能を拡張し、暗号技術を用いて電子メールの内容の保護、発信者の電子署名などの機能を提供するだけでなく、暗号鍵の分散管理と効率的な鍵配送を行なうシステムである。この FJPEM の実用化により、インターネット環境での電子メールの内容の保護が可能となり、例えば電子メールショッピングなどといった商業活動に安心して電子メールを使用することができるようになる。

本節では、まず FJPEM で用いられているモデル、具体的なシステムの設計、さらに開発したシステムの性能について述べる。さらに現在 GNU Emacs 上で利用可能な、PEM 化された電子メールの作成と参照を支援するユーザインタフェースパッケージ mhpem.el について述べる。

1.1 暗号メール PEM

PEM(Privacy Enhanced Mail) は、インターネットの標準化文書 RFC1421-1424 [132, 133, 134, 135] で定められている電子メールの暗号形式である。PEM で対象としている電子メールのセキュリティ上の脅威には、次の 3 つがある。

- **盗聴**···通常、電子メールは、メッセージが平文 (何も暗号化されていない状態) のまま配送される。このため、ネットワークが盗聴されて、メールが読まれてしまう危険性がある。
- **改竄**···電子メールは幾つもの中継ノードを経由して、蓄積交換を繰り返しながら配送される。このため、中継ノードにおいて、内容を改竄される危険性がある。
- **なりすまし**···インターネットにおける電子メールでは、発信者を特定する情報が保護されていない。このため、悪意を持つ第三者が発信者情報を偽造して他人になりすます危険性がある。

PEM では、上記セキュリティ上の脅威に対して、それぞれ暗号化、電子署名、公開鍵証明書を利用することによって対処している。

1.2 PEM の原理

PEM の原理について説明する。

1.2.1 DES と RSA によるハイブリッドな暗号化方式

秘密鍵暗号は簡易で高速であるが、すべてのユーザの間について秘密鍵を用意する必要があり大規模な運用は不可能である。一方、公開鍵暗号は、各ユーザ一人一人について公開鍵と秘密鍵を用意するだけで良いのでスケーラビリティは高いが、計算に多くの処理時間が必要である。そこで、PEM では、これらをハイブリッドに組み合わせた高速でしかも広域性にも優れた暗号化方式を採用している。

図 1.1は、送信者 A が受信者 B に宛ててメールを暗号化する処理を示している。まず、A は、秘密鍵暗号 DES の鍵 DEK(Deta Encryption Key) をランダムに作り、本文を DEK で暗号化する。更に、DEK を B の公開鍵について公開鍵暗号 RSA で暗号化して送信する。受信者 B は、この処理を逆に行ない、B の秘密鍵で復号化して DEK を取り出し、それを用いて暗号文を復号する。

複数人の受信者にメールを発信する場合には、DEK を各受信者の公開鍵で暗号化してメールに加える。受信者が自分の秘密鍵で復号化する情報を特定できるように、各 DEK の暗号情報には識別用のヘッダーが前置きされる。また、発信者本人が暗号文を復号化出来るように、発信者 A の公開鍵についても DEK を暗号化してメールに加える。

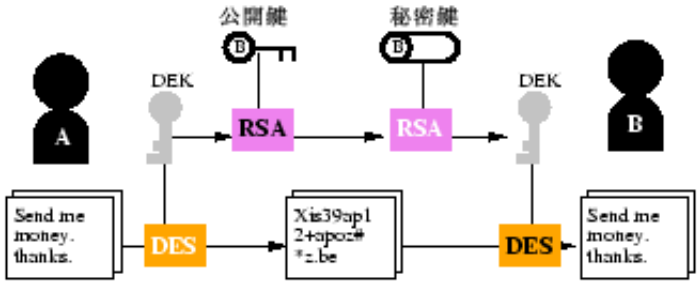

図 1.1: 暗号化方式

1.2.2　改竄を防止する電子署名

電子署名は、発信者の認証と本文内容に改竄が行なわれていないことを検証する技術である。 PEM では、図 1.2に示すように、RSA 公開鍵暗号を用いて電子署名を実現している。

まず、署名者 A は、適切なハッシュ関数により本文のチェックサムにあたるメッセージダイジェストを計算し、その値を自分の秘密鍵を用いて暗号化する。これが署名となる。ハッシュ関数としては、MD2, MD4, MD5[136, 137, 138] などが用いられる。検査者 B は、A の公開鍵を用いて署名を復号化し、その値が原本のメールのメッセージダイジェストと等しいかどうかを検査する。もしも、本文の一部が書き換えられていれば、この検査によりたちまち改竄が検出される。たとえ、本文を書き換えて署名を偽造しようにも、真の署名者の秘密鍵を知らない限り不可能である。

PEM には、暗号化と署名を同時に行なうモード “ENCRYPTED” と、本文は平文のままで署名だけ行なうモード “MIC-CLEAR” がある。

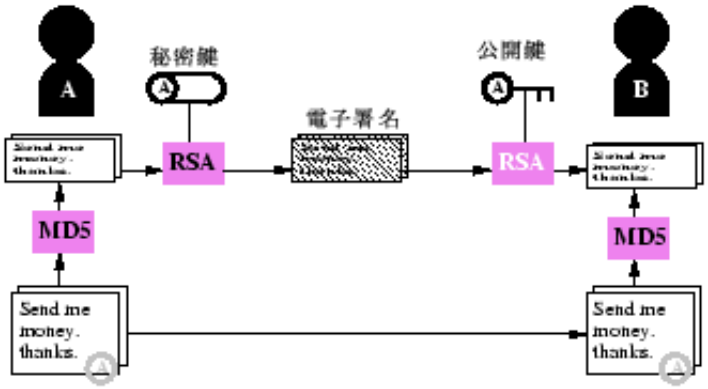

図 1.2: 電子署名

1.2.3 認証手段を提供する公開鍵証明書

盗聴と改竄については前記の二つの機構で防止できるが、公開鍵の管理を疎かにするとなりすましの危険性が生じてくる。つまり、本来自由に公開されている公開鍵を、不正ユーザが偽の公開鍵に置き換えることで、そのユーザになりすことが出来てしまう。そこで、PEM では、この不正行為を防止するため、正規の公開鍵であることを示す公開鍵証明書 (Public key Certificate, 以後単に証明書と呼ぶ) を導入している。

証明書による鍵管理の方法を図 1.3に示す。ユーザ A は、まず自分で公開鍵と秘密鍵を作り、公開鍵だけを信頼できる発行局 (Certification Authority) へ登録する。発行局 C は、この公開鍵とユーザ A の個人情報に秘密鍵で署名し、 A に返送する。これが証明書である。 A が PEM を発信する際には、暗号文と共に証明書を添付して送信し、受信者 Bが信頼している発行局 Cの公開鍵で署名を検証出来るようにする。この処理により、Bは常に正規の公開鍵を検証して、使用することができる。

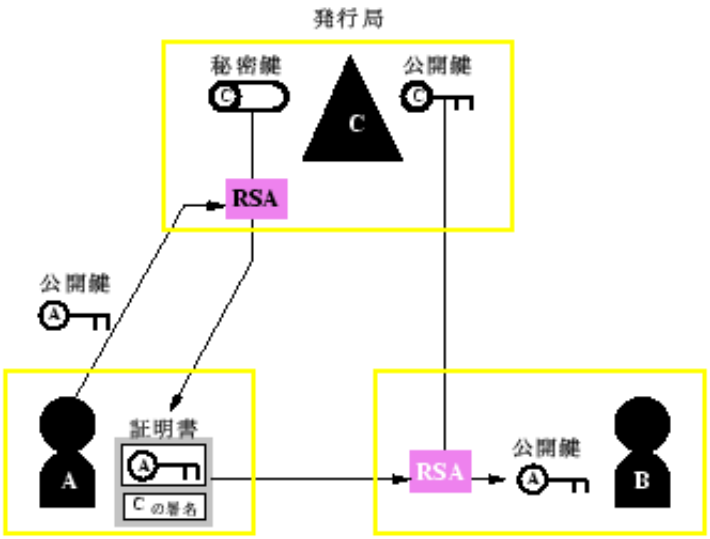

図 1.3: 公開鍵証明書による鍵管理

1.2.4 証明書発行局の階層構造

発行局という信頼点を設けることで、公開鍵を安全に配布することが可能となるが、一つの発行局で管理できるユーザの数には限度がある。そこで、発行局の公開鍵をまた別の発行局に発行してもらうことで、発行局同士を階層的に構成する方法が導入されている。

図 1.4に示される階層構造の例を示す。ここでは、ビジネス主体の A 社 (A corp.) から、学術目的の B 大学 (B univ.) まで各々ポリシーの異なるいくつかの組織が、より上位のポリシー発行局 PCA1 (Policy Certification Authority) により相互接続性を確立している。例えば、A 社のユーザ A_1と B 大学のユーザ B_2が通信する場合には、 A_1は自分の証明書の他に、 A 社発行局の発行局証明書を添付する必要がある。 PCA1 と PCA2 は、更に上位の IPRA (Internet Policy Registration Authority) により相互に接続している。

このように発行局の管理する単位をより小さなものに分割するので、管理の目が行き届いて安全性は高くなり、かつ世界規模のユーザにも十分対応が出来るようになっている。

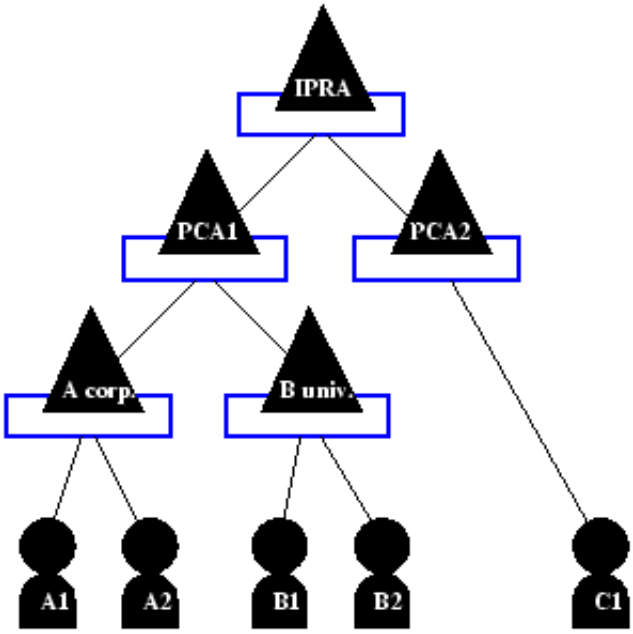

図 1.4: 証明書発行局の階層構造

1.3　世界的な PEM の現状

現在、世界各国でこの RFC 標準案に従った PEM の実装と、相互接続実験が始まっている。現在確認されている PEM の実装を表 1.1に示す。しかしながら、これらの実装の多くはまだ実験レベルにとどまっており、現実のネットワーク環境での実用性は不確かなままである。各システムの性能評価も報告されていない。

1.4　FJPEM の実装

RFC 標準案の有効性を評価し、運用上の課題を検討するために、暗号メールシステム FJPEM をサンプル実装した。 FJPEM システムは、C 言語および perl, shell スクリプト

表 1.1: 現在確認されている PEM の実装

名前	国名	開発元
TIPEM	米国	RSADS 社で開発、製品化.
RIPEM	米国	ミシガン大学
		暗号化を除いてパブリックドメイン.
TIS/PEM	米国	Trusted Information Systems 社、
		ARPA がスポンサー
		米、カナダ限定フリーソフト
UCL/PEM	英国	ロンドン大学 Password Project
		フリーソフト
Cambridge/PEM	英国	ケンブリッジ大学
GMD/PEM	独	
COST/PEM	スゥエーデン	COST 社、商用
松下 PEM	日本	松下電器産業、楕円暗号を使用.
FJPEM	日本	富士通研究所 (WIDE との共同開発)
		フリーソフト

により記述されており、一般的な UNIX ワークステーション上での動作が確認されている。現在動作が確認されている計算機を表 1.2に示す。

また FJPEM は、OSISEC RSA ライブラリ, GNU DES ライブラリ、メッセージダイジェスト MD2,MD5 ライブラリ、Printable 符号化 (6 ビット符号化) ライブラリのフリーソフトを用いている。基本仕様を表 1.3に示す。

1.5 RFC からのの拡張点

FJPEM は、RFC1421-1424 に従って実装されているが、以下の点を RFC の標準案から拡張している。

1.5.1 日本語に対応した電子署名

日本語を表現するコード体系として、EUC, JIS, Shift-JIS の三種類が広く用いられている。日本の Internet の電子メールに限っては、 7 ビット JIS コードが採用されているが、それにも表 1.4に示されるいく通りものエスケープシーケンスが存在し、それらが混在して用いられている。例えば、漢字を開始するには (1) または (2)、英数文字には (3) または (4) のエスケープシーケンスが利用されているので、計 4 通りの表現があり得る。

メール転送システム (MTA) やユーザ支援メールシステム (UA) の中には、これらを自

表 1.2: 動作を確認しているマシン一覧

マシン名	OS
SUN SparcStation 1/2	Sun OS 4.1.*
SONY RISC-News	NEWS-OS 4.2R
SONY CISC-News	NEWS-OS 4.2C
Gateway 2000	BSD386
LUNA 2	Mach 2.5
NEC EWS4800	EWS-UX/V(Rel4.2) R7.1, EWS-UX/V(Rel4.0) R6.2
	EWS-UX/V WSOS20 R9.1
NEXT	NEXT STEP
IBM RS6000	AIX 3.2.5
FUJITSU DS/90	UXP-DS

表 1.3: 暗号処理仕様

秘密鍵	p, q(256bit の素数), d(512bit 素数)
公開鍵	$n = pq$ (512bit)
共通鍵	$e = 65537$ (17bit)
符号化	RFC-1421 Printable Encoding [132]
ハッシング	Message-Digest MD2,MD5[136, 138]
データ暗号方式	DES in cipher block chaining (CBC) mode[139]
公開鍵暗号方式	RSA
復号方式	Quisquarter(中国人剰余定理利用)
証明書形式	ITU-T 勧告 X.509 [140]
メールシステム	mail, mh, mh-e, VM, rmail

動的に変換するものがある。ところが、本文はそのままで電子署名だけを行なうモード (MIC-CLEAR) を利用している時には、それらのコード変換によってメッセージダイジェストの値が変わるため、改竄が行なわれたものとして検出されてしまう。

そこで、メッセージダイジェストを計算する際に、表 1.4のエスケープシーケンスの (1) を (2) に、 (3) を (4) に標準化する前処理を加えた。 (2) は (1) より新しく、また (3) にあって (4) では表せない “¥” 文字は漢字の一部として表現可能と考えたためである。なお、メール自体はそのまま転送するため、ユーザから見た表示には違いは生じない。

RFC 標準案では、標準化の処理として、 EBCDIC などから ASCII へと変換することだけが定められており、その意味ではこの標準化は RFC に反している事となる。しかし、他のコード体系を用いる場合にも同様の問題が生じることが予想されるため、むしろ RFC の枠組を拡張することが必要と考えられる。

表 1.4: JIS コードに関するエスケープシーケンス

(1)	ESC $ @	JIS C 6226-1978 の G0 への指示
(2)	ESC $ B	JIS X 0208-83 の G0 への指示
(3)	ESC (J	JIS X 0201 ローマ字の G0 への指示
(4)	ESC (B	ASCII の G0 への指示

1.5.2 メールアドレスを証明書の中に包含

RFC では、ITU-T 勧告 X.509[140] の証明書の形式を使っている。これは、OSI ディレクトリシステムによってユーザを識別するためである。しかし、現在のインターネットでは、OSI ディレクトリが十分に相互接続されていないため、X.509 の証明書の形式ではユーザを十分に識別できない。

そこで、FJPEM では、メールアドレスを証明書の中に包含することを試みた。これにより、メールの受信者は、電子署名と証明書の中に含まれるメールアドレスによって送信者を確認することが可能となる。

1.6 配布局

証明書を公開する手段として、本実装では、配布局と呼ぶ証明書を配布するサーバを用意した。配布局は、RFC-954 で標準化されている WHOIS プロトコルを用いてユーザと通信する。WHOIS を採用したのは、 Internet で広く利用され、実際に多くのシステムに組み込まれているためである。配布局は、発行局と連携して証明書データベースを管理している。証明書の偽造は、発行局の秘密鍵が盗まれない限り困難であるので、発行局ほど高いセキュリティを必要としない。複数の配布局を用意して、負荷を分散することも可能である。

また、ユーザからの要求に答えて証明書データベースの検索などのサービスも提供する。用意されている検索の種類を表 1.5に示す。

表 1.5: 配布局の機能

検索の種類	内容
キーワード	組織名や名前の一部などから該当ユーザのリストを出力
シリアル番号	該当するシリアル番号の証明書の情報を出力
メールアドレス	メールアドレスに一致する証明書を出力
廃止証明書	指定する日付以降に廃止された証明書の一覧を出力
新規証明書	指定する日付以降に発行された証明書の一覧を出力

配布の効率を上げるため、一度受信した証明書を格納しておく証明書キャッシュの機能を加えている。従って、一度 MIC-CLEAR のメールを交換し、互いの証明書を相手の証明書キャッシュへ格納しておくことで、 WHOIS による配布局のサービスを受けられな

いネットワークにおいても、暗号メールを利用することが可能となる。一般には、初対面の相手との通信で暗号化が必要な情報を送ることは稀なので、証明書キャッシュはかなり有効と考えられる。

証明書キャッシュと配布局を併用した場合の、暗号メール (ENCRYPTED モード) を送信する手続きを示す。

1. 証明書キャッシュから受信者の証明書を検索する。
2. キャッシュになければ、配布局を検索する。
3. 配布局になければ、MIC-CLEAR(署名のみ) のモードに変更する。

これらの処理は、メールシステム (UA) が自動的に実行されるため、ユーザが意識する必要はない。

1.7　メールインターフェイスへの対応

メールの配送システムは、一般に (sendmail などの) リモートへの配送サーバ、(popd などの) ローカルへの配送サーバ、および、(mh-e などの) メールのインターフェイスに分けることができる。PEM を設計する際に考慮されているように、リモートへの配送サーバがメールを暗号／復号する方法は、現在の配送システムを根本から見直すことになり、PEM に対応してないリモートへの配送サーバとの互換性を考えると現実的ではない。そこで、以下ではリモートへの配送サーバには変更を加えず、ローカルへの配送サーバ、あるいは、メールのインターフェイスで処理を施すことについて議論する。

(1) ローカルへの配送サーバで暗号／復号する場合
既存のメールのインターフェイスを全く改良する必要がない。しかしながら、メールのインターフェイスへは復号されたメールが渡されるので、メールを平文で保存することになる。ローカルへの配送サーバを配置していない組織では、この方法は適応できない。また、ユーザはメールが実際に暗号化されたことを確かめることができない。

(2) メールのインターフェイスで暗号／復号する場合
メールを暗号化された状態で保存できるため、方法 (1) に比べセキュリティレベルが高い。また、ユーザは、メールが実際に暗号化されて配送されることを確かめることができる。しかしながら、メールのインターフェイスは数多く存在するので、すべてのアプリケーションを PEM に対応させることは困難である。

ローカルへの配送サーバが、メールのインターフェイスへ、メールを復号せずにメールを転送機能を提供すれば、方法 (1) と方法 (2) は共存できる。つまり、PEM に対応しているメールのインターフェイスは、暗号化されたままのメールの配送を、対応してないインターフェイスは、復号化されたメールの配送を要求すればよい。

ローカルへの配送サーバを PEM に対応させることは今後の課題とし、以下ではメールのインターフェイスを PEM に対応させることについて論じる。

1994 年 3 月末現在、FJPEM Version 1.1 に対応しているメールのインターフェイスは次の通りである。

- コマンドラインからのインターフェイス
  - /bin/mail
  - MH
- エディタ Emacs 上のインターフェイス
  - VM
  - Rmail
  - mh-e

これらのメールのインターフェイスのうち、PEM へ対応するように改良を重ねてきた mh-e の場合について、拡張方針、拡張点、課題について述べる。

mh-e は、MH を Emacs から利用するためのメールのインターフェースである。我々は、このインターフェースから FJPEM を利用できるようにするための目標を、「現在の mh-e の利用から混乱なく PEM 形式のメールの読み書きをできるようにすること」に設定した。

拡張方針は以下の通りである。

- mh-e が呼び出す MH 自体には変更を加えない。
- ユーザの行う操作方法は従来の mh-e と極力変わらないようにする。
- メールは暗号化されたまま保存し、読む時に随時復号する。
- メールの送信の際には、ユーザは FJPEM が提供している暗号化タイプ MIC-CLEAR か ENCRYPTED を選択できるようにする。また、平文のままでメールを送信することを選択できるようにする。
- 通常は、やりとりするメールのほとんどが暗号化されたメールであることを想定する。従って、メールを読む際には自動的に復号し、PEM ヘッダを取り除く。
- 加えて、暗号化されたメールを扱うことが頻繁でない状態を想定し、必要な時だけ復号して表示する方法を提供する。

これらの設計方針に基づき、以下のような機能を mh-e に拡張した。

- 配送時におけるメールの暗号化
  通常のメールの送信操作(C-c C-c) 直後に、暗号化形式MIC-CLEAR、ENCRYPTED、あるいは PEM 形式にしない CLEAR を選択してから送信できるようにした。

  また、頻繁にはメールを暗号化して送らない場合のために、通常はメールを暗号化して送らないことを選択できるようにした。この場合、(C-u C-c C-c とタイプして) 明示的に暗号化すれば、メールを暗号化することができる。

- 暗号化されたメールの自動復号
  平文のメールを読む時と全く同じ操作 (“.” と入力) で、自動的に PEM ヘッダの解析や復号を行なう復号モードを提供した。加えて、明示的な操作があった場合のみに復号する非復号モードを提供した。また、動的に復号モードと非復号モードを切替える方法を提供した。

- 暗号化されたメールの引用
  暗号化されたままのメールを引用するのは、意味をなさない場合が多い。よって、暗号化されたメールの内容を引用する際には、メールを復号するようにした。

- 暗号化されたメールの転送
  暗号化されたメールは他人には読めないので、転送の際は復号する必要がある。よって、暗号化されたメールを自動的に復号し転送する機能を実現した。

これらの拡張部分は、mhpem.el というライブラリとして提供した。mh-e のバージョン 3.6、3.7、および、3.8 に対応している。

mh-e の拡張に関する課題として、次のような点が挙げられる。

- alias 機能への対応
  mh-e (MH) ではメールの宛先の入力を簡略するために、alias を設定することができ、これは送信操作後に自動的に展開される。mhpem.el の現在のバージョンでは、alias を展開する前に PEM 形式に直すため、宛先アドレスの公開鍵を取得する場合に問題が生じる可能性がある。

  例えば、

```
sato:  sato@remoto.xx.jp
```

  のように alias を設定し、ローカルのドメイン名が local.yy.jp であるとする。この時、sato 宛先アドレスに sato とだけ書くと、MH はこれを sato@remoto.xx.jp に展開し配送する。しかしながら、PEM では、ローカルのドメイン名 local.yy.jp が省略されているとみなし、ローカルのドメイン名を補った sato@local.yy.jp の公開鍵を取得することを試みる。

- mh-e 自体のバージョンアップへの対応
  mhpem.el では mh-e の関数を書き換えている。そこで、mh-e のバージョンが上がった時に、mhpem.el をバージョンアップに追従させる必要が生じる。今後は、mh-e のバージョンに依存しない手法が必要になってくると考えられる。

- MIME と PEM の統合
  MIME(RFC1521, 1522)、および、PEM は、共に RFC822 に対する独立した拡張である。現在の mhpem.el も、MIME の mh-e からのインターフェースも、ともに MIME と PEM の混在した状態を想定していない。

  MIME と PEM の関係については、以下の 2 つの方式がインタネットドラフトで提案されているが、まだ RFC とはなっていない。

  (a) MIME-PEM Interaction
  PEM を MIME の multi-part として実現する。既存の PEM の実装とは互換性のない仕様となっている。

  (b) An Alternative PEM MIME Integration
  PEM を MIME のアプリケーションと考え、新しい PEM のための Content-Type を定義する。また、MIME を暗号化できるように、PEM の新しい Content-Domain として MIME を定義する。この 2 つの定義により、MIME と PEM を自由に組み合わせることができる。また、FJPEM の実装をそのまま流用できる。

  我々は (b) の方法を採用し、実装を行なう予定である。

1.8 公証人

公証人とは、新規ユーザ (申請者) を認証する正規 PEM ユーザである。公証人申請方式では、公証人が申請者に代わって PEM の電子署名機能を使って証明書の申請を行なう。従って、発行局は、公証人の電子署名を検証することにより、申請者を機械的に認証することが出来る。ユーザの申請内容は、公証人が責任を持つこととなる。それゆえ、組織毎に公証人を用意することにより、各々の組織のポリシに従った証明書発行が実現する。

証明書が発行されたユーザは、他の新規ユーザの公証人となることを許している。従って、信頼できるユーザの証明書を連鎖的に発行していくことにより、大規模環境でのユーザを信頼の鎖で結ぶことが出来る。こうして図 1.5に示す様なユーザの集合の上に公証人の木が構成される。ここで、ユーザ A はユーザ Bの公証人であることを表している。ユーザ Eと Dが異なる組織 (ネットワーク) に属していても、共通に信頼できる公証人 A により互いの認証をすることができる。

公証人の関係は発行局にのみ記録されていて、ユーザの証明書は共通の発行局によって電子署名されている。従って、ユーザは発行局の署名を検証するだけで互いを認証で

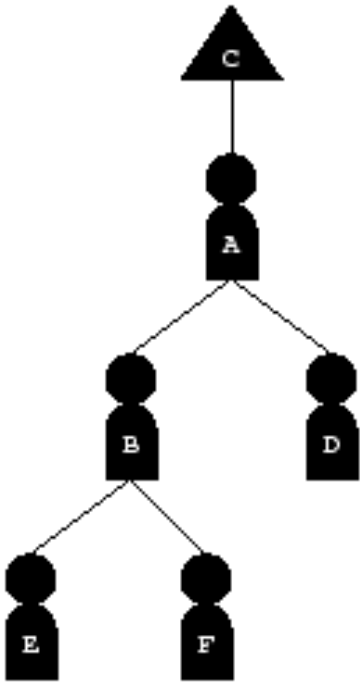

図 1.5: 公証人の木

き、検証処理が簡略化する。また、メールに添付しなくてはならない発行局の証明書は一つだけとなり、伝送効率もよい。更に、ユーザの証明書は発行局で一元管理されるため、証明書の検索も容易である。

RSA Data Security 低信頼発行局と比べると、発行局への申請に PEM を利用しているため、申請書を送信中の改竄やなりすましを防止している点で信頼性が高い。また、申請のための署名は通常の PEM の形式であり、自己署名をするための特別な処理をする必要がない。また、公証人が信用できるという仮定の下で、偽りの申請をすることを防止している。

1.9　公証人による証明書申請

公証人による申請方式を図 1.6の上で説明する。ここで、公証人を A, 証明書を申請するユーザを B、発行局を Cとする。

1. **鍵生成** 新規ユーザ A は、乱数を基に RSA 公開鍵暗号の秘密鍵と公開鍵を自分で生成する。

2. **依頼** Aは、ユーザ名、所属、メールアドレスなどの個人情報に公開鍵を加えた申請書を用意し、信頼できる公証人 Bに申請を依頼する。申請書には、公開鍵と証明書に記載するのに必要な個人情報を記載する。公証人との通信にはメールなどを用いる。

3. **署名** 公証人 Bは、申請書の内容を確認し、虚偽がないことを認めれば、その内容に自分の秘密鍵で電子署名をする。PEM MIC-CLEAR モード [132] で署名された申請書の例を図 1.7に示す。ここで、Originator-Certificate:以下の数行が発信者 (公証人) の証明書、MIC-Info:以下の二行が電子署名である。

4. **代理申請** Bは、A の代理で申請書を発行局へ PEM を使って送信する。必要ならば、発行局の公開鍵で暗号化をしてもよい。

5. **検査** 発行局 Cは、申請書を復号化してその署名を検証する。公証人が正規のユーザであり、改竄も行なわれていないことを確認できれば、その申請書は有効である。証明書データベースを照会し、ユーザ名の一意性も確認する。

6. **再署名** Cは、申請書の情報に発行局の秘密鍵で署名して、証明書を発行する。

7. **登録** Cは、証明書を内部のデータベースへ登録する。

8. **発行** Cから証明書をユーザ A に返送する。

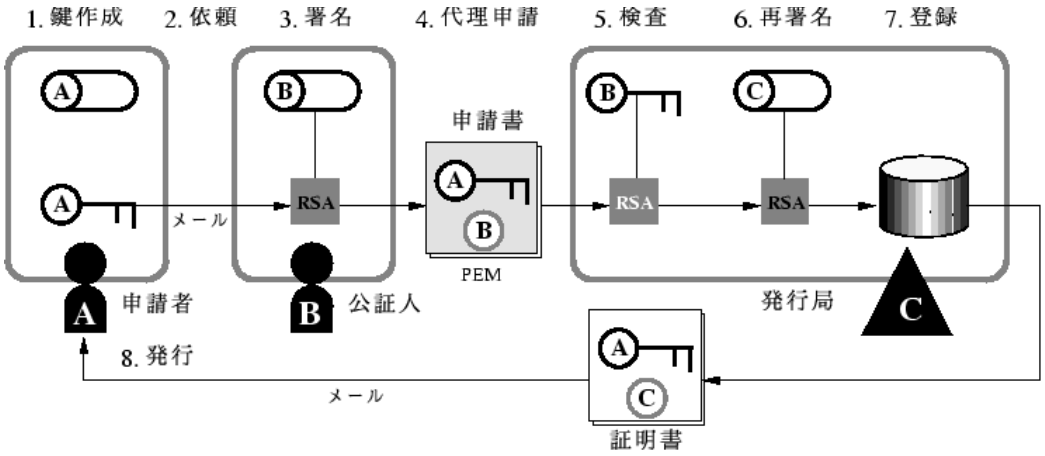

図 1.6: 公証人による発行処理

1.9.1 公証局

組織内申請者の身元をチェックして、証明書発行の代理申請を行なう公証局がある。

1994 年 3 月現在立ち上がっている公証局を表 1.6に示す。申請者の身元のチェックには、正規ユーザのアドレスリストなどが一般的に使われている。

```
From: kikn@abclab.co.jp
To: ca@wide.ad.jp
Subject: certificate request

-----BEGIN PRIVACY-ENHANCED MESSAGE-----
Proc-Type:4,MIC-CLEAR
Content-Domain:RFC822
Originator-Certificate:
 MIIBsDCCAU4CAgFiMA0GCSqGSIb3DQEBAgUAMD4xCzAJBgNVBAYTAkpQMQ0wCwYD
 VQQKEwRXSURFMSAwHgYDVQQLExdDZXJ0aWZpY2F0aW9uIEF1dGhvcml0eTAeFw05
 NDAzMTUxMjQ1MDhaFw05NTAzMTUxMjQ1MDhaMHsxCzAJBgNVBAYTAkpQMRswGQYD
 VQQKExJGdWppdHN1IExhYnMuIEx0ZC4xTzAbBgNVBAMTFEhpcm9ha2kgT2RlbiBL
 SUtVQ0hJMDAGCSqGSIb3DQEJARYja2lrbkBvZGVuLmNlbnRlci5mbGFiLmZ1aml0
 c3UuY28uanAwXDANBgkqhkiG9w0BAQEFAANLADBIAkEAt0bjPwnVKSy05W7pJTt+
 kgK+E9D5q9egX+0szlXR32N6VMCPTTxrnLmzdVnuaZS5AEkSbNLzv3TCm3sZLElT
 3QIDAQABMA0GCSqGSIb3DQEBAgUAA00ACpiuYS+daR2kWOMvRdEOdfBmgoQ6WXAf
 agXk/1hVBt1zB4DTV4Cttrt/zxAWMTctlV6lx75q1nG84i+TMKtvPPA3W3P8/RUr
 XEBvRl==
Issuer-Certificate:
 MIIBfjCCARwCATwwDQYJKoZIhvcNAQECBQAwPjELMAkGA1UEBhMCSlAxDTALBgNV
 BAoTBFdJREUxIDAeBgNVBAsTF0NlcnRpZmljYXRpb24gQXV0aG9yaXR5MB4XDTkz
 MTEwOTAwMDAwMFoXDTk2MTEwOTAwMDAwMFowPjELMAkGA1UEBhMCSlAxDTALBgNV
 BAoTBFdJREUxIDAeBgNVBAsTF0NlcnRpZmljYXRpb24gQXV0aG9yaXR5MGgwDQYJ
 KoZIhvcNAQEBBQADVwAwVAJNAG9zTm8/Lpu3bvxog7283OdT0oSPL2v7Vf9bqQQ+
 INVW5pZyTqQnvH8hEbTSCln/Vdjdph6j10k43WcPtSthSB4ggysVGXdHW3yQTAkC
 AwEAATANBgkqhkiG9w0BAQIFAANNADuEnfRO5r342YWhnYyn6ETxIU6dzd4JMbKg
 TzOrp0hQQtgF/9HxkhjE49NUP00VerZPKQUu7ppt2+v/ZFahBlOIhSQl3J8gUWLj
 Ie2=
MIC-Info: RSA-MD5,RSA,
 k2YAJQQZ6+WmnVh9QY9uAiaR5IGNoTYSu7Eqk6ibGYxuveViPQ6x6dH5oo3Rbgf7
 eKDonJqFIAy09/KVKTxKLi==

CN=     Yasutsugu Kuroda
C=      JP
N= 7e29b196fd3ecead935ceccabe9076af68671a88312f565ff4431adb93260b
624c10b690e78adc816d79b
MAIL=   kuroda@abclab.co.jp
O=      ABC Laboratories Ltd.
TITLE=  researcher
-----END PRIVACY-ENHANCED MESSAGE-----
```

図 1.7: PEM MIC-CLEAR で署名された申請書の例

表 1.6: 現在立ち上がっている公証局

公証局名	公証局のアドレス
NEC 公証局	notary@ccs.mt.nec.co.jp
京都大学 公証局	notary@ca.imel.kyoto-u.ac.jp
日立ソフト公証局	notary@hitachi-sk.co.jp
電気通信大学 公証局	notary@hood.cs.uec.ac.jp
東芝 公証局	notary@isl.rdc.toshiba.co.jp
日本アイビーエム 公証局	notary@imazine.trl.ibm.co.jp
富士ゼロックス 公証局	notary@fujixerox.co.jp
九州大学 公証局	notary@kyushu-u.ac.jp
IPA 公証局	notary@stc.ipa.go.jp
富士通研究所 公証局	notary@ca.fujitsu.co.jp

1.10 性能評価

FJPEM の処理速度、受信者数に対する影響、発行局の階層化に対する影響、証明書キャッシュの効果について評価を行なった。評価は、Sun SPARC Station 2(30MIPS) 上で、標準の C コンパイラ (cc) によりコンパイルした FJPEM システムを用いた。処理時間は、計算機の内部時計 (tcsh 組み込み time コマンド, time() システムコールなど) により計測し、誤差を小さくするため、複数回測定したユーザ処理時間の平均を取っている。

処理速度

1k バイトから 100k バイトまでの英文のファイルについて、各処理時間を図 1.8のグラフに、各処理速度を表 1.7に示す。また、512 ビットの RSA 暗号化と復号化にかかる処理時間は、各々 0.15[s], 1.28[s] であった。

表 1.7: 各部処理速度

処理	処理速度 [kbps]
RSA 暗号化	3.414
RSA 復号化	0.400
DES 暗号化	1974
DES 復号化	1917
MD5	3165
Printable 符号化	7385
Printable 復号化	8689

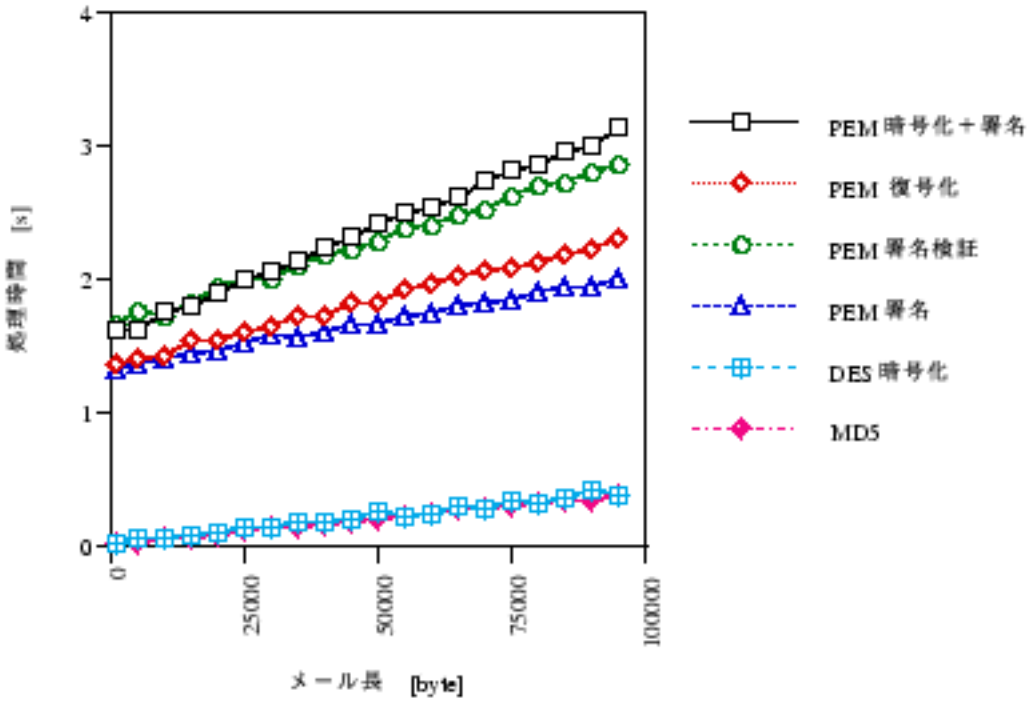

図 1.8: メール長と処理時間の関係

RSA の暗復号化はその他の処理に比べ極端に多くの処理時間を必要としているが、DEK の暗号化などの毎回固定の処理にだけ使われているため、メール長に依存しない。グラフでは、固定の処理が、メール長が 0 の時のオフセットとなって表れている。

一般的なメール長と言われる 2k バイトの暗号化に 1.63[s]、復号化に 1.35[s] の処理時間がかかることがわかった。メールシステムでの、暗号化復号化処理の自動化と組み合わせると、十分利用できる利用感が得られた。

1.10.1　受信者数に対する影響

一般的なメール長と言われる 2k バイトの英文ファイルについて測定した、暗号化の処理時間と暗号文の大きさを図 1.9に示す。

処理時間と暗号文の大きさは受信者数に比例している。メールシステムによっては、一度に転送できるメールの大きさに制約があり、多くの受信者へ同報するような目的に利用するには、問題となることがわかった。ただし、処理時間は、メールは厳密な即応性を要さないアプリケーションであるため、致命的ではないと考えられる。なお、復号化の処理時間は、受信者数にほとんど影響しない。

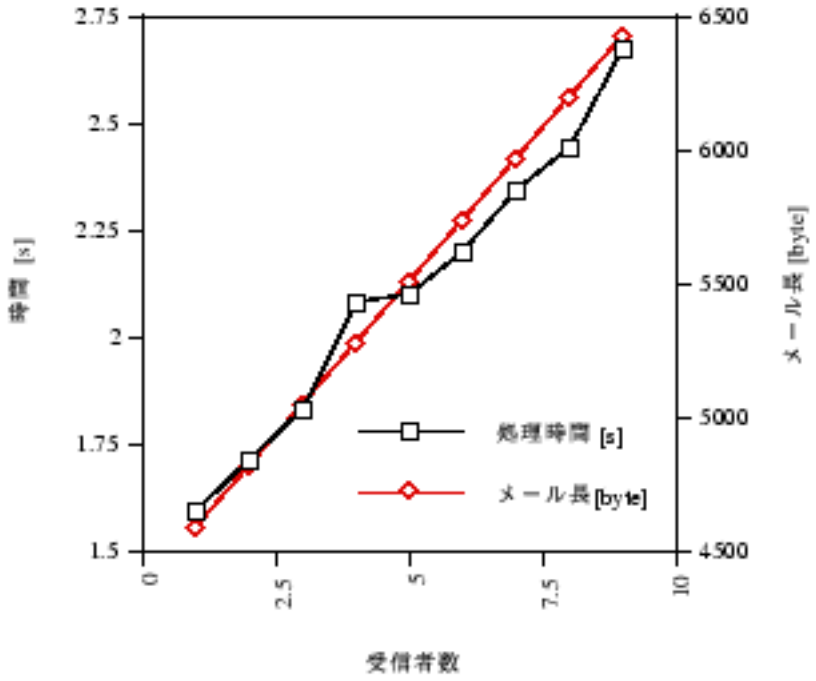

図 1.9: 受信者数の影響

1.10.2 多重暗号化、署名に対する影響

暗号化と署名を行なうモード (ENCRYPTED) と本文はそのままで、署名と証明書だけを追加するモード (MIC-CLEAR) の両方について、暗号化の処理時間と暗号文の大きさを測定した結果を図 1.10に示す。メールには、2k バイトの英文ファイルを用いた。

ENCRYPTED の場合には、処理時間も暗号文も勾配が急になっていくことがわかる。これは、Printable 符号化の際に、8 ビットから 6 ビットへの変換が行なわれるため、n 多重の暗号化で、暗号文が $(4/3)^n$ 倍になるためである。一方、MIC-CLEAR では、多重署名に対する影響は線形である。ただし、機密性は損なわれてしまう。また、多重に繰り返し署名されたメールを、再帰的に検証することは可能だが、署名された順序を確実に守る必要がある。以上より、PEM をそのまま一般のビジネス文書の承認印として用いるためには、まだ問題が多いことが明らかとなった。

1.10.3 発行局の階層化に対する影響

発行局の数は、受信者側の署名の検証処理と、メールに添付する証明書の数に影響する。そこで、証明書の検証処理だけの処理時間を測定し、発行局が増えた場合の仮想的な検証の処理時間とメール長を図 1.11に見積もった。ここでメールには、2k バイトの英文ファイルを用いた。

グラフは、発行局の数が受信者側の負担を増やすことを示している。しかし、証明書

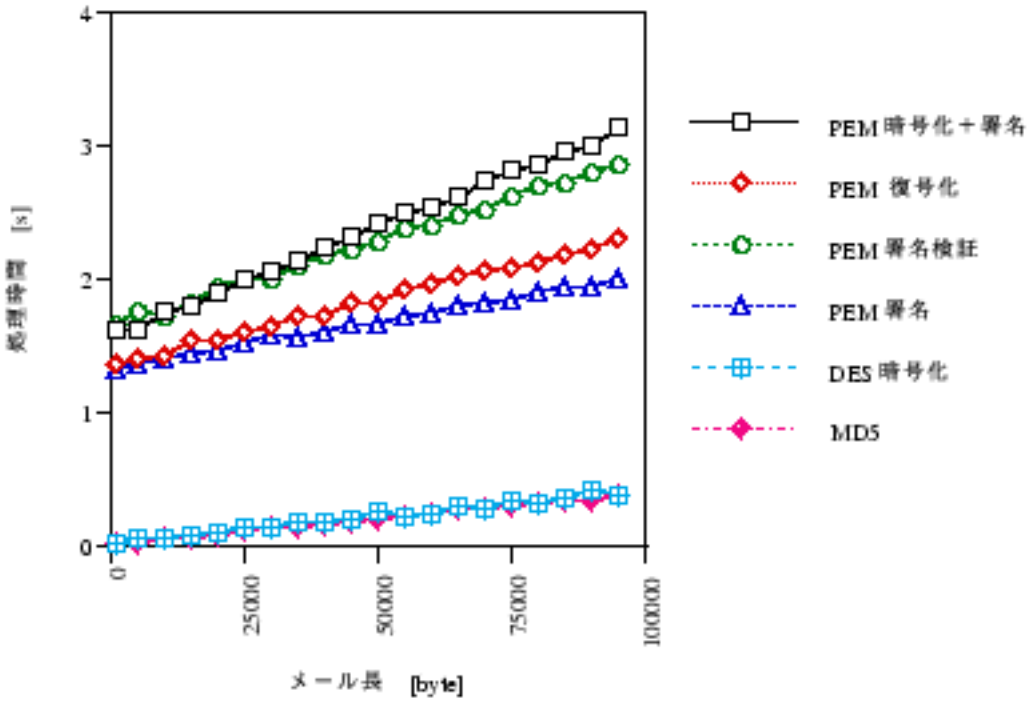

図 1.10: 多重暗号化の影響

のキャッシュが効果的に働けば、実際の検証時間にはそれほど影響しないことが予測される。

メール長に関しては、上位の発行局の証明書を全て添付することが RFC では定められている。実際に、3 つの発行局証明書を添付して運用している実装例もある。しかし、証明書一つの大きさは数百バイトあり、毎回同一の情報を送信するのは効率が悪い。発行局の構成と同時に、検討していかねばならない問題の一つである。

1.10.4　証明書キャッシュの効果

キャッシュの効果を見積もるため、94 年 2 月 24 日より一週間、富士通研究所の共用マシンから発信したメールの統計を調べた。約 1000 人のユーザについての次の二種類の平均値を、図 1.12のグラフに表す。

$$M(x) = x \text{ 日間に発信したメールの総数}$$
$$R(x) = x \text{ 日間に発信した通信者の総数}$$

ここで、$M(x)$ は一通のメールでも複数人に宛てている場合は、複数のメールとして数え上げている。逆に、$R(x)$ は、重複を削除している。従って、もしも全てのメールに PEM を用いたとすると、$M(x)$ の値が x 日間に一人が必要とする平均的な証明書の数を表す。一方、$R(x)$ は証明書がキャッシングされていた場合の必要な証明書の数を表す。x は証明書キャッシュをクリアする間隔のパラメータとなる。

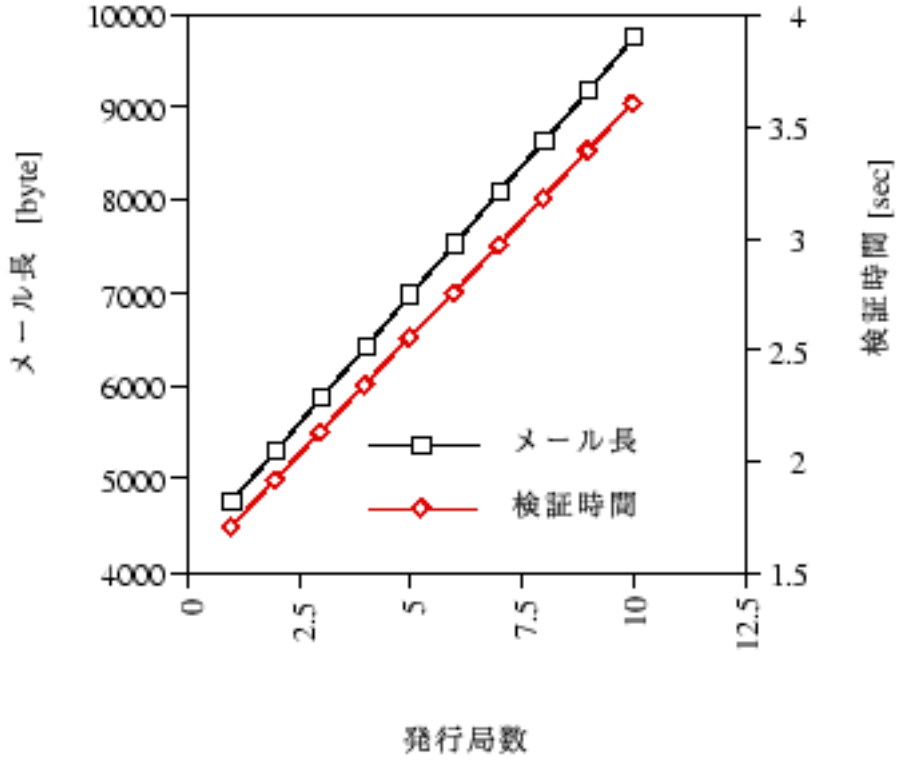

図 1.11: 発行局の影響

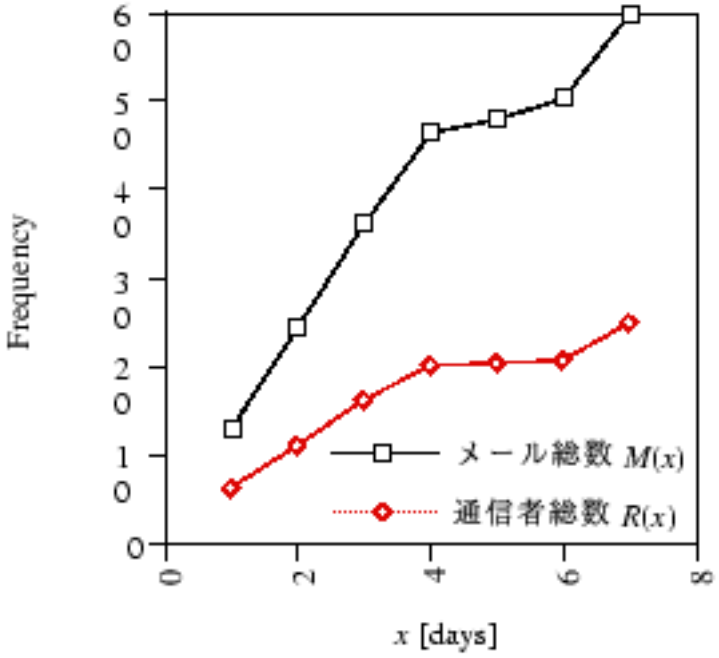

図 1.12: 電子メール送信回数

グラフより、メール総数は日数に比例して増加しているが、通信者の数の増加は緩やかである。従って、証明書をキャッシュしておけば、配布局へのアクセスがかなり押えられることがわかる。

これを、具体的に示すために、メールの発信をランダム、配布局での処理時間を負の指数分布に従うもの (M/M/1 モデル) としてシュミレーションを行なった。

実測による配布局での平均処理時間 1.5 [s] より、配布局での検索にかかる平均サービス率を $\mu = 1/1.5$ とする。 n 人のユーザが一日当たり必要とする証明書数を c とおくと、配布局への検索要求の平均到着率 λは、

$$\lambda = \frac{nc}{60 \times 60 \times 24}$$

と表せる。c の値は、1000 人の統計値より、表 1.8に従うものと仮定する。すると、配布局での平均応答時間 Tは次式で与えられる。

$$T = \frac{1}{\mu - \lambda}$$

この結果を図 1.13に示す。キャッシュがない場合には、一つの配布局で 5,000 人程度までしか運用できないが、7 日分のキャッシュを導入することで、三倍の 15,000 人のユーザに対応できることが示された。実際には、証明書が更新されることは稀なので、より長い間隔でキャッシュしていてもよく、配布局への負荷は更に小さくなることが予想される。

表 1.8: 一日に必要とする証明書数 (c)

キャッシュなし	$M(1) = 8.53$
1 日おきキャッシュをクリア	$R(1) = 6.26$
2 日おきキャッシュをクリア	$R(2)/2 = 5.55$
3 日おきキャッシュをクリア	$R(3)/3 = 5.30$
4 日おきキャッシュをクリア	$R(4)/4 = 5.00$
5 日おきキャッシュをクリア	$R(5)/5 = 4.05$
6 日おきキャッシュをクリア	$R(6)/6 = 3.41$
7 日おきキャッシュをクリア	$R(7)/7 = 3.52$

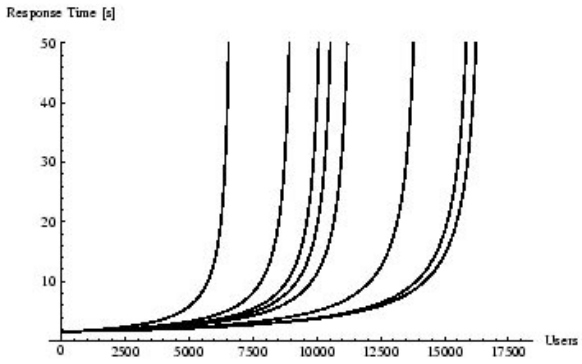

図 1.13: 配布局の応答時間

1.11　公開実験について

提案方式の有効性を実証するため、発行局を実装し、WIDE Internet 上で運用実験を行なった。より多くの環境から参加できるように、実装システムのソースプログラムを Anonymous FTP サービスやネットワークニュース (電子掲示版) などにより参加を呼びかけた。公証人となる初期ユーザには、 WIDE プロジェクトセキュリティワーキンググループのメンバーを登録した。ここでは、実験の目的、実装、および結果を述べる。

1.11.1　目的

1. 現実の大規模ネットワークで PEM の有効性を実証する
   これまでに行なわれてきた暗号メールの実験は、単一の組織内で閉じたものがほとんどであった。これに対して、本実験では、組織間での通信も含む大規模な環境での運用を試みる。 WIDE Internet は、異なるポリシーを持つ組織が接続した実験ネットワークであり、ここで運用上の課題を明らかにすることは、より本格的なネットワーク社会において有用であると考える。この目的のために、発行局の運用上の議論や情報交換などに積極的に PEM を活用することを心がける。

2. 発行局の運用限界を明らかにする
   Kerberos[141] などのチケットに基づく認証システムと比較して、PEM の証明書は有効期間が長い。チケットや証明書の有効期間は、暗号方式の安全性に基づいて決められており、例えば、秘密鍵暗号による Kerberos のチケットは数時間、公開鍵暗号を用いた証明書は 2 年が有効と言われている。認証手段の有効期限が長いほど、信頼出来る第三者 (認証サーバや発行局) の負荷は小さくなり、それだけ大規模な運用が可能となる。本実験では、発行局の負荷を基にこの効果を明らかにし、発行局の運用規模の限界を見積もる。

3. 公証人方式の効果を明らかにする
   公証人方式は、発行局を階層化にすることに比べて、発行局証明書の数が少なく、ユーザの検証処理が少ない。実験で構成される公証人の木を基に両方式の処理量を比較し、この効果を明らかにする。

また、RFC からの拡張点である以下の点の有効性を実証するのも目的である。

- 日本語に対応した電子署名
- メールアドレスを包含した証明書

1.11.2　実験環境

実験に用いる環境を図 1.14 に示す。 PEM 環境は、ユーザ A, B, 公証人 C, 発行局 W, 配布局 Sから成る。

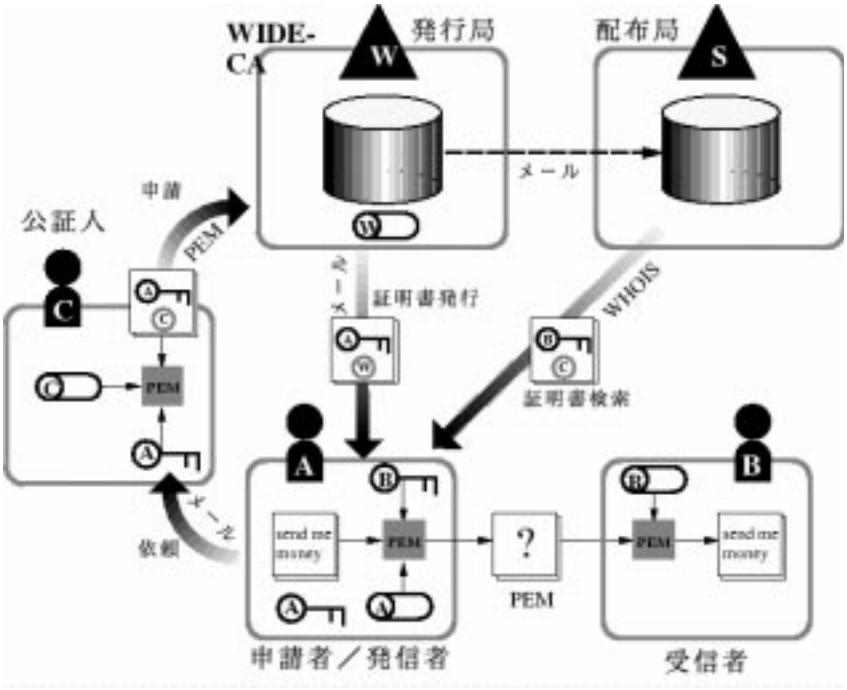

図 1.14: PEM 環境の構成

1.11.3 発行局

発行局 WIDE-CA は、証明書の発行を行なうシステムである。 C 言語,Perl スクリプトで Sun4/110 上に実装されている。申請書の検証部、証明書発行部、証明書データベースより構成されている。発行局は、通常のメールインターフェースにより申請書や証明書を送受信する。従って、ネットワークと疎に結合して、侵入者からの攻撃を防止することが可能である。本人かあるいはその公証人からの要求に応じて、証明書の廃止や修正も用意している。

1.11.4 公証局

公証人の仕組みを自動化したシステムを公証局と呼ぶ。公証局では、正規ユーザのデータベースなどを基に、組織に応じたユーザ認証が行なわれる。

1.11.5 配布局

配布局 Keyserv は、発行局と連携しており、ユーザからの要求に答えて証明書データベースの検索や参照のサービスを提供する。 C 言語,Perl スクリプトで Sun4/110 上に実装されている。ユーザとのインターフェースには、 RFC954 WHOIS プロトコル [108] を用いる。キーワード、シリアル番号、メールアドレス、廃止証明書、新規証明書の 5 種類

の検索が機能している。負荷に応じて分散が可能で、本実験では二台の配布局を用いる。

1.12　実験結果

1993 年 10 月 25 日から 1994 年 3 月 3 日までの、各種ログデータを表 1.9 に示す。平均公証人階層数は、発行局までに経由している公証人の数の平均値である。例えば、図 1.5 のユーザ Eの場合は 2 となる。公証人総数は、ユーザの中で公証人をした人の数である。証明書や配布局での処理時間は、システムログ情報より算出した値であり、ユーザまでのメールの遅延は含んでいない。

表 1.9: 試験運用のログデータ

運用期間 [日]	128	
ユーザ数	126	
平均公証人階層数	1.84	(最大 4)
参加組織 (ドメイン) 数	31	
公証人総数	33	
公証局数	11	
証明書発行平均処理時間 [s]	34.7	
証明書発行回数 [回/日]	1.82	(最大 19)
証明書申請回数 [回/日]	11.38	(最大 75)
証明書修正破棄回数 [回/日]	1.3	(最大 7)
配布局平均処理時間 [s]	2.93	
配布局 1 検索回数 [回/日]	25.88	(最大 130)
配布局 2 検索回数 [回/日]	19.41	(最大 140)

ユーザ数の推移と一日当たりの証明書発行回数を図 1.15のグラフに示す。ユーザ数が小さく減っている時があるのは、証明書の廃止による。

ユーザ数に対する発行局への申請の内訳を図 1.16に示す。申請の種類には、発行、修正、廃止の三種類がある。

二つの配布局の負荷を図 1.17に示す。グラフでは、5 種類の検索サービスの合計と、FJPEM システムによって自動的に行なわれる検索の回数 (自動) を一週間毎に平均したものを、各々の配布局について示している。検索の種類は、ユーザからのキーワード検索の要求がほとんどである。

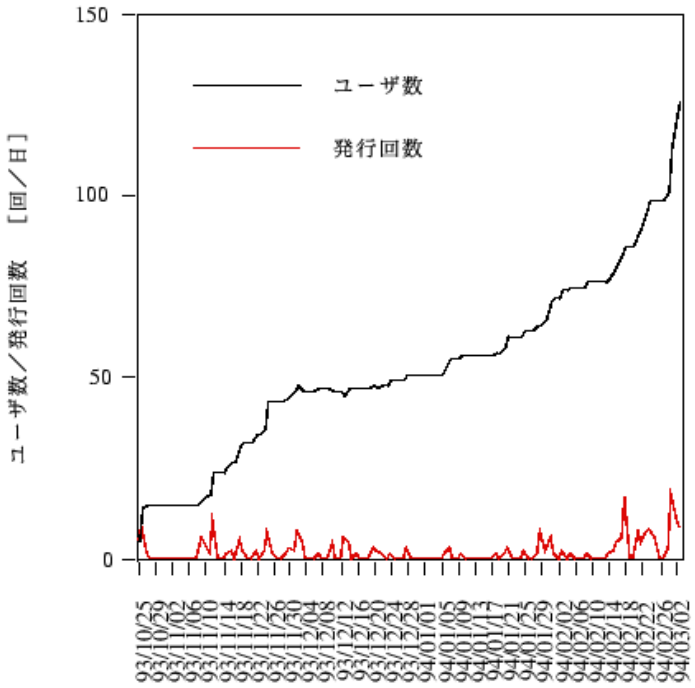

図 1.15: PEM ユーザ数の推移

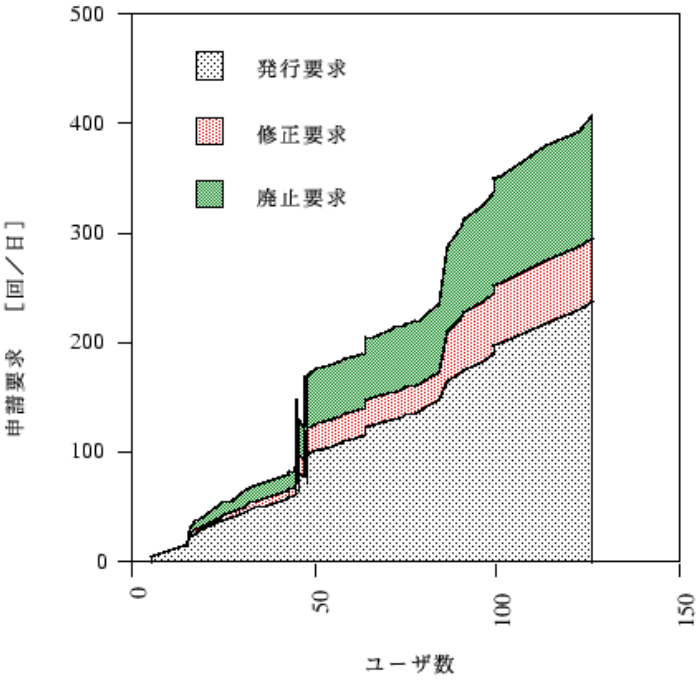

図 1.16: 申請の内訳

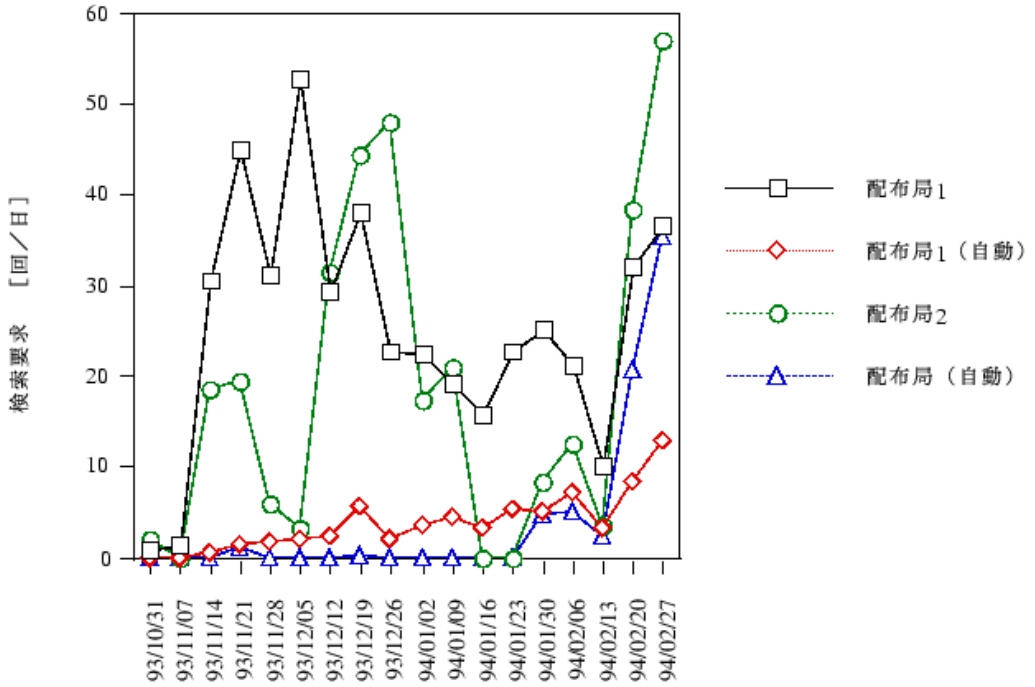

図 1.17: 配布局の負荷

1.13　評価

1.13.1　発行局の運用限界

図 1.15のグラフを見ると、ユーザ数の増加に対して、発行局での証明書発行回数はほとんど一定であることが観察できる。すなわち、ユーザ数に対して発行局の負荷はそれほど増加していない。これをより厳密に示すため、ユーザ数に対する一日の発行回数を図 1.18のグラフで表した。発行回数はユーザ数に対して線形に増加するものと仮定して、最小自乗法を適用すると、 n 人のユーザの時の一日の申請回数 $I(n)$ は次式となる (図 1.18)。

$$I(n) = 0.043n - 0.467$$

従って、ユーザ数 n の時の申請書平均到着率$\lambda_c(n)$ は次式で求まる。

$$\lambda_c(n) = \frac{I(n)}{60 \times 60 \times 24}$$

厳密には、証明書の有効期限より 2 年に一回の更新処理があるが、$I(n)$ に対して十分小さいため無視する。

表 1.9の証明書発行平均処理時間より、発行局の平均サービス率は $\mu_c = 1/34.7$ と定まる。ここで、申請書の到着間隔をランダムとみなし、発行局処理時間の分布を負の指数分布に従うもの (M/M/1 モデル) とする。すると、発行局での平均応答時間 $T_c(n)$ は次

式で求まる (図 1.19)。

$$T_c(n) = \frac{1}{\mu_c - \lambda_c(n)} = \frac{1}{0.029 - 5.0 \times 10^{-7} n}$$

例えば、$n = 10000$ の時の平均応答時間は 42 秒, 平均待ち行列数は 0.21 である。ただし、発行局までのメールの転送時間は考慮していない。また、$\lambda_c(n^*) = \mu_c$ となる n^*の値は、$n^* = 57915.8$ である。従って、本実装の発行局が処理できる最大ユーザ数は約 5 万人であると言える。最大ユーザ数は平均サービス率に比例しているので、発行局が実装されている計算機の処理能力を上げれば、より多くのユーザについて運用できる。例えば、発行局を Sun4/110 (7MIPS) から Sparc Station 2 (28MIPS) に置き換えると、証明書発行の平均処理時間は約 1/6 となり、約 34 万人までの証明書発行が行なえることとなる。

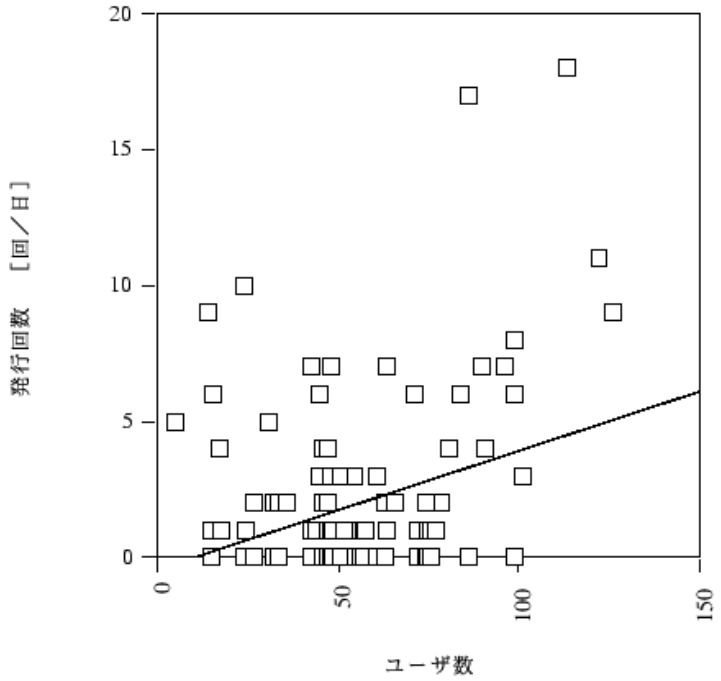

図 1.18: 発行回数の分布

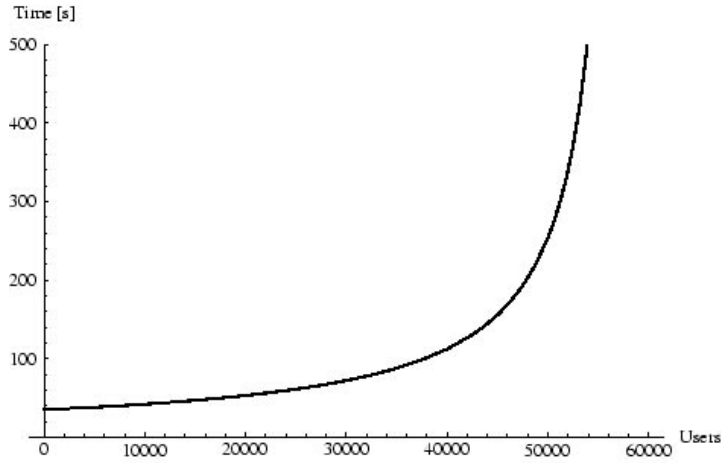

図 1.19: 発行局の平均応答時間

1.13.2 配布局の運用限界

配布局についても発行局と同様の仮定を行ない、配布局 1 のデータから検索要求の平均到着率$\lambda_w(n)$ を求めると次式のようになる。

$$\lambda_w(n) = \frac{0.15n + 18.0}{60 \times 60 \times 24}$$

表 1.9の証明書発行処理時間より、配布局の平均サービス率は$\mu_w = 1/2.93$ と定まる。以上より、平均応答時間 T_wが次式で求まる。

$$T_w(n) = \frac{1}{\mu_w - \lambda_w(n)} = \frac{1}{0.34 - 1.7 \times 10^{-6} n}$$

この結果を、図 1.20のグラフに示す。同様にして、最大ユーザ数は、約 19 万人と見積もられる。ただし、証明書の信頼性は RSA 公開暗号に基づいているため、配布局のセキュリティレベルは比較的低い。従って、配布局を各地へ分散させることによって、運用限界を更に引き上げることも可能である。

1.13.3 公証人方式の効果

表 1.9より、平均公証人階層数は 1.84 である。公証人階層数は、ポリシの違いなどにより分けなければならない発行局の数を意味している。従って、もしも本実験を複数の発行局で運用したならば、 WIDE-CA を含めて 2.84 階層が必要となることがわかる。

Sun SparcStation 2 での平均的な 2k バイトのメールを暗号化した時の処理速度とメール長を表 1.13.3に示す。発行局を 3 階層で運用した時を、公証人方式の場合の 1 階層と比べると、メール長で約 1.2 倍、検証処理時間で約 1.2 倍に増加している。メールは即応性を要さないので処理時間は無視できるが、メール長には最大転送サイズの制約があり、平文 2k バイトに対して約 3 倍の大きさに増加している点は問題である。なお、ここで示した受信者の証明書検証にかかる時間は、証明書のキャッシュの効果がない最悪の場合である。

表 1.10: 発行局数による処理時間

CA 階層数	1	2	3	4	5
検証処理時間 [s]	1.7	1.91	2.12	2.33	2.54
メール長 [byte]	4757	5311	5865	6419	6973

1.14 おわりに

本節では、WIDE Project Security Workging Group が中心となって開発した FJPEM について述べた。現在 FJPEM は、国内のメジャーな Anonymous FTP サイトから配布

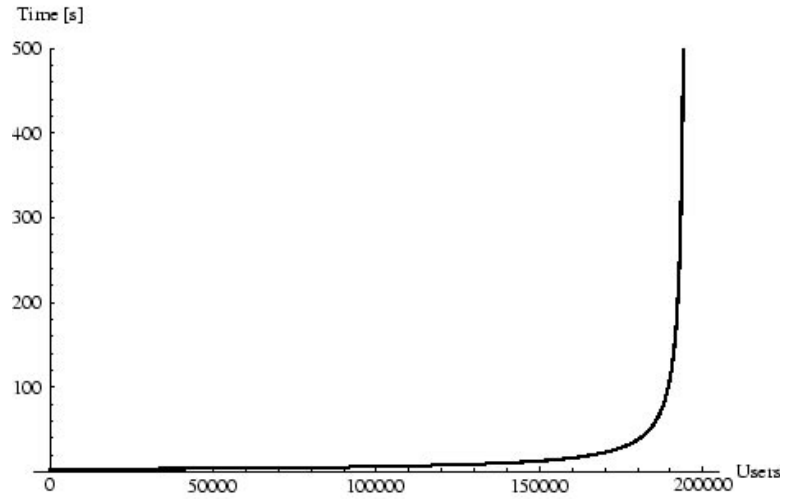

図 1.20: 配布局の平均応答時間

を行なっている。また、FJPEM がサポートするプラットフォームも順次増えつつある。一方 FJPEM における未解決の大きな問題として、Macintosh や PC (Windows システム) といったプラットフォームでのサポートが完了していないことがあげられる。現在のインターネット環境では、電子メールを UNIX ワークステーションに限らず、さまざまなプラットフォームで利用している。しかし、現在 FJPEM は UNIX ワークステーションでの利用のみを対象としている。これをどのように PC や 他の UNIX に拡張していくかは，今後とも検討が必要である．また、発行局が複数になった場合の認証のメカニズムや、システムの運用方法などについても検討が必要であろう。WIDE Project Security Working Group では、これらの問題についても 1994 年度で引続き検討・実装を進めていく予定である。