認証局パッケージ EasyCert の開発と評価

Development and Evaluation of A Certification Authority Package

奥野 琢人* 若山 公威* 村瀬 晋二** 鈴木 春洋** 岩田 彰*

Takuto Okuno Kimitake Wakayama Shinji Murase Shunyo Suzuki Akira Iwata

あらまし 電子証明書を発行する認証局（Certification Authority：以下CA）の必要性が高まっている。しかし現在では、CAを構築するためには大規模で高価な商用CAを利用するか、あまり設定が簡単ではないオープンなCAを利用するしかない。そこで、インストール、CA構築から証明書発行までユーザに負担をかけないGUI CAアプリケーションをWindows上で構築し、その利用可能性について考察した。

キーワード 電子証明書, 認証局, GUI

1. はじめに

ユーザがインターネット上で安全な通信を行う場合、多くの手段が考えられる。その中でも代表的なものとしてWWWで暗号通信として使われているSSLや、Emailで使われているS/MIME、これらの他にもSSHやPGPなどが挙げられる。特に大手のブラウザやメーラーにはSSLやS/MIMEは標準機能として備わっており、これらの機能の使用率は今後も増加していくと考えられる。

しかし、こうした技術を利用する上で必要不可欠なのが電子証明書の発行や秘密鍵の生成を行うソフトウェアである。秘密鍵の生成や証明書要求の作成などはNetscapeやMicrosoftのWWWブラウザや、メーラーに暗号通信機能を付加する魔法便II[1]といったアプリケーションにより可能であるが、証明書要求に対し署名を行うCAアプリケーションは、まだそれほど普及しているとはいえない。

最近では多くのCAアプリケーションが大手メーカーや、セキュア系ベンチャーから販売されている。また、VeriSign[2]などの証明書発行サービスも多く利用されている。しかしこれらのサービス・アプリケーションは非常に高価なため、簡単に使用できるとは言い難く、また証明書作成や管理といった煩雑な作業を軽減できる本格的なCAはまださほど多くは存在しない。さらに、OpenLDAP[3]やActiveDirectoryといった新しい情報検索システムも電子証明書を保持しており、CAはこれらの仕組みと同期しながら動作する必要がある。すなわち、CAはまだまだ発展途上のアプリケーションであるといえる。

こうした背景がありながら、通信路の安全を確保するため、実験的にCAを導入するオフィスは多数存在する。しかし初めから高価なCAを導入するにはコストがかかり過ぎ、実験段階ではフリーのCAを使用することが多い。ここでよく利用されるのがOpenSSL[4]であり、フリーながら非常に高度な暗号ライブラリをもち証明書の発行も可能である。また、国産のCAでWWWブラウザを通して証明書の発行が行えるICAP[5]等も多く利用されている。

本研究ではCAを試験的に導入したり、上記にあるフリーのCAよりももっと手軽にCAの構築・証明書の生成を行いたいユーザに対して、Windows上で動作するGUIベースのCAアプリケーション、証明書ビューアや証明書コンバーターをセットにしたEasyCertパッケージを開発した。このパッケージは、セットアッププログラムが付属しており、インストール・アンインストールの実行も容易である。また、CA構築ウィザード

*名古屋工業大学 電気情報工学科, 〒466-8555 名古屋市昭和区御器所町

**株式会社シーティーアイ SI事業部, 〒450-0003 名古屋市中村区名駅南1-27-2 日本生命笹島ビル(7F・8F・9F)

によりユーザに負担をかけることなく CA を構築することを可能にしている。

本論文では、初めに EasyCert の CA としての位置付けを行い、次に CA アプリケーション群の使用法等を含めた解説を行う。その後、他の CA との操作感や暗号ライブラリの性能も含めた比較評価を行い、EasyCert の改良や発展について考察を行っていく。

2. CA アプリケーション

2.1 証明書発行形態

ユーザが電子証明書を入手するためには、いくつかの方法を挙げることが出来る。

１つは、ユーザ自身が証明書要求と秘密鍵を作成し、そのうち証明書要求を CA に送信する。そして CA は受け取った証明書要求に対し、自身の持つ発行ポリシーに従いユーザの存在を認知し、証明書を発行、又は発行拒否を行う。そしてユーザは、発行された証明書と秘密鍵を自身の情報として保全管理する。(図 1)

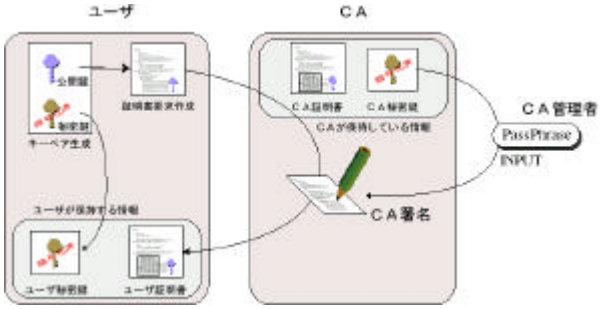

図 1. 証明書発行手順

この方法により証明書を入手する場合、ユーザ自身で多くの手順を踏む必要がある。幸い、広く普及している Netscape Navigator や Internet Explorer では、これらの手順を軽減できるよう多くの機能が含まれているが、商用の証明書発行サイトでは料金の支払いや、ユーザの特定を厳密にするためにメールを使用するなど、結果的に煩雑であり利用しづらい。

これに対して、もう一つの方法は自身で証明書を生成するのではなく、RA（Registration Authority）に証明書の発行を依頼し、RA がユーザ本人に代わって秘密鍵の生成や証明書の取得を行う方法である。(図 2)

この方法ではユーザに負担をかけずに証明書を作成することができ、Windows 環境においては「証明書＋秘密鍵」のファイル（PKCS#12）を比較的容易にシステムにインストールし、メーラーやブラウザで使用することが可能である。また、RA が秘密鍵の保守管理を行えるため、万一ユーザが秘密鍵を無くした場合でも鍵のリカバリーが行える。

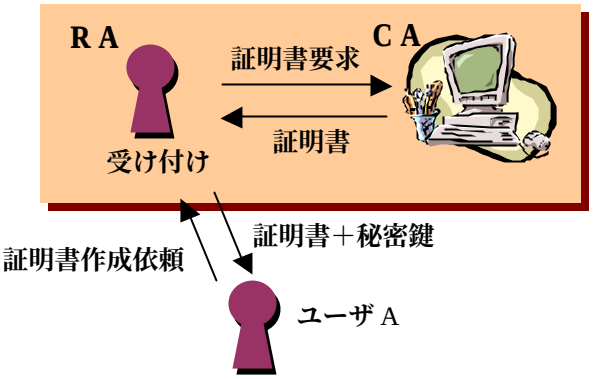

図 2. RA による証明書発行手順

こうした利用形態において唯一の欠点は、作成した証明書と秘密鍵を安全・確実に本人に渡すことが困難である事である。しかし、この場合はある組織が組織内で CA を構築し内部の人間に対して証明書発行を行うことが多いと考えられ、物理的な手段や閉じたネットワーク内で安全にファイルの受け渡しが可能だと考えられる。

これら双方の発行形態を比べると、1 つの大きな違いが浮かび上がってくる。この違いとは、証明書の信頼性の問題である。現在では、最初の手段が多く使われているが、この方式では証明書の信用度があまり高いとはいえない。すなわち、CA に対して電子メールや WWW 経由で証明書要求が送信され、証明書を受け取る本人の身元を確認できないまま証明書の発行を行っているため、証明書それ自体の信用度が低いといえる。

しかし、将来的には IC カード等に証明書が含まれ社員証や運転免許のように身分証明として証明書が使われる事が考えられる。さらには、判子の代わりに署名が行われる事も考えられ、単純に証明書要求に対して証明書を発行するような運用形態は考えにくい。すなわち、2 番目に述べている RA によって決まっただけの数の証明書を発行・一括管理し、それを本人に対面しながら手渡すのが理想的な証明書の発行形態だと考えられる。

2.2 CA 運用形態

現在一般には多くのアプリケーションが出回っているが、これらの運用形態は大きく 2 つに分類される。すなわち、スタンドアロン型とサーバ・クライアント型である。

CA アプリケーションもこれらの形態のどちらか 1 つをとり得る事になるが、多くの商用の CA は CA サーバとして世に出ており、WWW ブラウザにサーバの処理結果を表示するような形態が多くとられている。しかし、本格的な CA が望まれる中でこうした利用形態は不十分であると思われる。すなわち、CA が証明書発行だけではなく証明書や秘密鍵の管理といった RA の仕事を包括すること、また LDAP や電子公証システムなどと連携が必須になること等により、強固な CA サーバと CA クライアントの確立が必要であるといえる。

また、実際の CA の運用を考えたとき、1 つの企業が 1 つの CA ですべての従業員の証明書を発行するのではなく、それぞれの部署が独立に CA をもった上で特定の部署がそれらの CA を管理する運用形態が自然で有るといえる。

すなわち、多くの社員を抱える会社では全ての部署の全ての社員に証明書を発行することは容易なことではなく、証明書の発行・管理を集中管理するよりも支社や部署別に分散管理する方が、発行者にとって迅速な対応が行え、作業効率が良いと考えられる。

この場合は、最上位に社内用 CA が存在し、その下に各支社や部署毎に下位 CA が立てられ、その CA により各社員に証明書が配られる形となる。さらにそれを管理する人員は各支社の複数部署の CA を管理する事も考えられる。

このことから、1 つの CA クライアントが同時に多くの CA の状況を掴んだり、操作を行ったり出来ることが有効的な CA の管理方法だといえる。こうした CA サーバ・クライアントの登場が望まれている。

もう 1 つの運用形態であるスタンドアロン型であるが、こちらの方はそれ自体が完成した物であり、コンパクトで利用しやすいという利点がある。小規模なオフィスで巨大なサーバ・クライアント型のシステムを導入することは管理運用面でも導入面でもコストがかかり過ぎ、更にはクラッキング対策等、ネットワークセキュリティの面でもあまり好ましくない。

単体の CA を構築し管理運営にあまり手間をかけずに証明書を利用するためには、スタンドアロンで簡単に利用できる CA アプリケーションの存在が必須であるといえる。

2.3 EasyCert の位置付け

前節では証明書の発行形態や CA の運用形態から CA 又は RA の大別を行った。以下の表にこれらをまとめた。

表 1. 証明書発行形態

	単純 CA 型	RA 型
利点	システム構築が容易である ユーザ尊重であり、CA は必要最低限の情報をもつ	ユーザに負担をかけない ユーザの情報損失に対応できる
欠点	ユーザが多くの操作をしなければない 鍵の損失等に対応できない 証明書自体の信頼性の低下	管理運用に手間がかかる 安全な通信路があるなど組織内でのみ運用が可能

表 2. CA 運用形態

	サーバ・クライアント型	スタンドアロン型
利点	他のシステムと同期が容易である 大規模なシステムに対応できる	安価である 運用が簡単である
欠点	高価である 運用が複雑である セキュリティ面に不安	大規模なシステムに対応しづらい 他のシステムとの連携が困難

この中で EasyCert は前節で言うところの、RA 内包型のスタンドアロン CA アプリケーションである。すなわち、証明書や秘密鍵の生成・発行から管理まで GUI により見やすく簡単に操作が行えるよう設計した。また、一括大量発行にも対応しており、中規模の CA 運用ならば十分対応する事が出来る。

3. EasyCert パッケージ

この章では EasyCert パッケージが含んでいる、証明書ビューア、証明書コンバータ、CA アプリケーションのそれぞれについて、その役割や使用法を述べていく。なお、これらのアプリケーションは、Windows95/98, NT/2000 上で動作するよう作成されている。

3.1 証明書ビューア CertView

CertView は従来、PEM 形式[6]や DER 形式によって保存されている電子証明書をわかり易くテキスト表示するアプリケーションである。

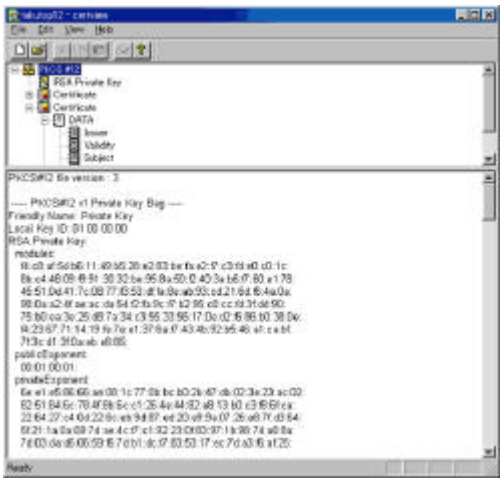

図 3. 証明書ビューア

図 3 に示される通り、一つのウィンドウを上下に分割しており、上部は TreeView を利用して証明書や各 PKCS ファイルの構造をツリー状に表示する。そして、ツリーのアイテム、すなわち証明書構造の一部を選択した場合、その部分だけの情報を下部にテキスト表示することも可能である。

また、ドラッグ&ドロップ等により選択された証明書・秘密鍵等のファイルを自動的に判別し表示することが可能である。

表 3. 表示可能ファイル形式

証明書	X509 証明書 (PEM,DER) PKCS#7 (Signed-Data Type)[7]
秘密鍵	RSA 秘密鍵 (PEM) PKCS#12 (証明書も含む)[8]
CRL	X509 CRL (PEM, DER)
証明書要求	PKCS#10 (PEM, DER)

3.2 証明書コンバータ CertConv

CertConv (図 4) は証明書・秘密鍵等に使われている各種ファイル形式を相互に変換するアプリケーションである。実際、使用する CA によって出力される証明書が PEM 形式、DER 形式、又は CA 証明書とユーザの証明書を保持する PKCS#7 形式であったりするなど多くの形式が氾濫している。しかし、CertConv を利用することでファイルを別の形式に変換し利用することができる。

CertConv では、ドラッグ&ドロップによって選

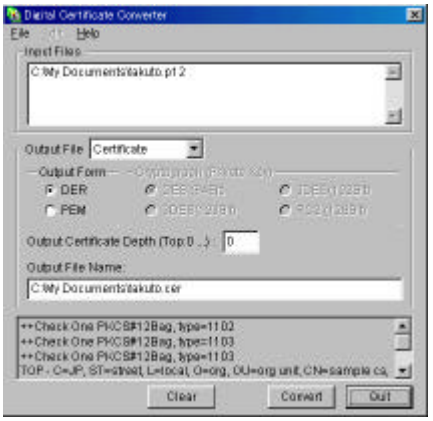

図 4. 証明書コンバータ

択したファイルに対し、CA ツリー構造や証明書と秘密鍵の対応を自動的に行い、必要なファイルだけを保持する。この方法により複数の証明書と 1 つの秘密鍵から、正しい PKCS#12 ファイルの生成を行えるようにしている。また、以下に変換可能なファイル形式の一覧を示す。

- 証明書, 証明書要求, CRL
  PEM ⇔ DER 変換
- チェイン有り複数証明書 ⇒ PKCS#7
- PKCS#7 ⇒ 任意の深さの証明書
- チェイン有り複数証明書＋秘密鍵
  ⇒ PKCS#12
- PKCS#12 ⇒ 任意の深さの証明書
  or 秘密鍵

3.3 CA アプリケーション EasyCert

EasyCert (図 5) は証明書要求や秘密鍵の生成、証明書、CRL の発行が行えるスタンドアロン型の CA アプリケーションである。Windows で一般的に使用されている対話的なウィザードやプロパティダイアログ等によって CA の構築から設定まで簡単に行えるよう設計してある。

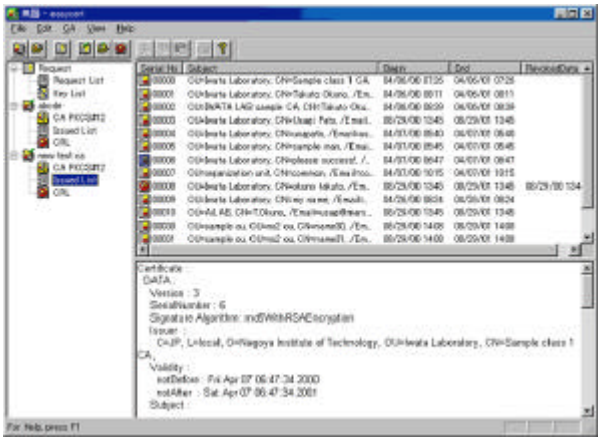

図 5. CA アプリケーション

EasyCert の実行画面は大きく 3 つに分割してある。左側に Explorer 等と同様のデザインを使用し CA や証明書要求、秘密鍵のリストを TreeView で表示する。また図 5 にある通り、複数の CA を同時に表示することが可能であり、クリック 1 つで CA の切り替えを行うことが出来る。

右上部には証明書や証明書要求のリストが表示される。ここには、証明書の基本的な情報であるシリアルナンバやサブジェクト、有効期限などが表示され、左端のアイコンによって証明書の状態（正常・破棄・期限切れ）が示される。さらに、リスト上でクリックされた証明書や証明書要求の全内容が右下部のテキスト領域に表示される。

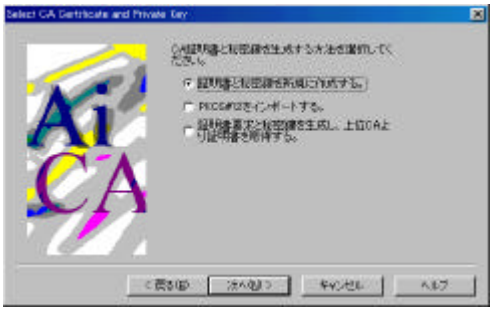

図 6. CA 構築ウィザード

次に CA 構築ウィザード (図 6) について述べる。EasyCert パッケージをインストール後に初めて EasyCert を起動すると自動的に CA 構築ウィザードが立ち上がり、明示的にユーザに CA を構築してもらうことが出来る。このウィザードでは図 7 の手順に従って CA を構築する。

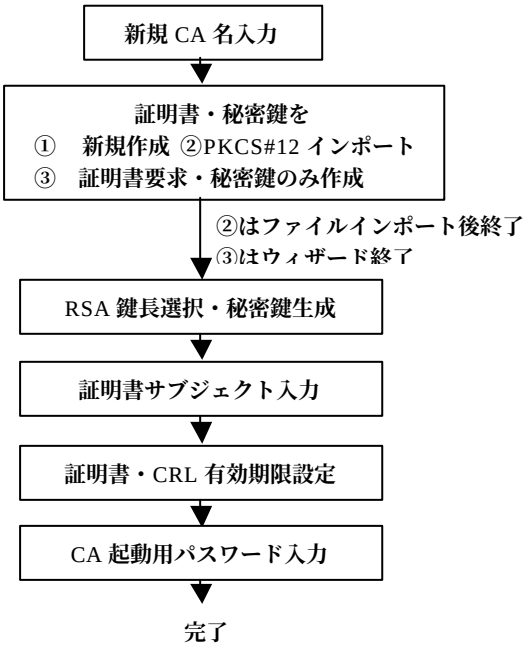

図 7. CA 構築ウィザードの流れ

また、図 8 に示す通り、発行する証明書や CRL の有効日数の設定、CA の証明書発行ポリシーの設定等、CA の設定を行うプロパティシートを用意して簡単に設定を行うことが出来る。同様に、発行する証明書の拡張フィールドには CA 毎に独立した拡張情報を付け加える可能性があるため、各 CA 毎に別々の設定が行える。なお、発行する全ての証明書（CA 証明書も含む）に対し、標準的に X.509 Basic Constraints を追加するようにしている。

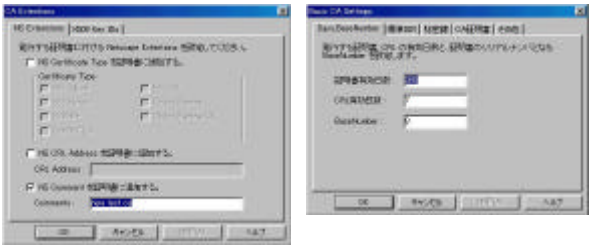

図 8. CA・拡張フィールド設定

CA のメインの機能となる証明書発行について、EasyCert では 3 通りの方法により証明書発行を行うことが出来る。1 つは、別のアプリケーションにより作成された証明書要求ファイルを開きこれに署名を行う方法である。署名されたファイルは CA 以下の証明書リストに追加されるため、その証明書の上で右クリックを行い、表示されたポップアップメニューの中から”Save As...”を選択することで証明書ファイルのエクスポートが出来る。ただしこの場合、秘密鍵の情報を EasyCert が保持していないため、PKCS#12 ファイルのエクスポートは不可能である。

２つ目の方法は、先に使用する CA を選択した後、秘密鍵の生成と証明書の作成を行う方法である。この方法では、EasyCert が秘密鍵を保持しており、証明書又は証明書要求と 1 対１で対応しているため PKCS#12 ファイルのエクスポートや秘密鍵のリカバリーといった操作が容易に行える。さらに、証明書のサブジェクト入力時に”CN”タグより上位のタグが CA のものと同様になるよう自動的に挿入されるため、煩雑なサブジェクト入力が軽減されるよう工夫されている。

最後の方法は、シリアルナンバやサブジェクトといった証明書発行に必要な情報をカンマで区切り、リストで構成する CSV 形式のファイルを入力する事で証明書の大量発行を行う方法である。大きな組織の RA で、大量の証明書を１枚づつ発行すること非現実的であり、データベースや表計算

ソフトにより CSV リストを生成し、それをもとに証明書の大量発行を行う事で、発行作業を大幅に軽減する事が出来る。実際に表 6 の環境で大量発行テストを行った場合、1 万枚の証明書を約５時間で発行を終え、1 枚あたり 1.68 秒で証明書発行を行う事が出来た。

また、リスト表示アイテムの複数選択に対しては一括破棄・更新・保存が行え、リストのタイトル部分をクリックする事で、タイトルに属するアイテムの順にソートする事が出来る。こういった対応によって、大幅に操作感が向上しユーザの負担を軽減する事が出来た。残る課題として、ドラッグ&ドロップに対する動作、さらにはリスト内部のアイテムに対する検索などを行えるよう機能追加を行っていく予定である。

4 . 性能評価

この章では、EasyCert におけるその操作性や扱えるファイル形式の評価、さらには著者が独自に作成した暗号ライブラリである AiCrypto の速度評価を行い、Free で高度な性能をもつ OpenSSL の CA と対比を行っていく。

なお、操作性における評価については、絶対的な評価方法が存在するわけではないので、著者による独自の判断で「◎, ○, △, ×」の四段階によって点数付けを行った。

4.1 暗号・ファイル形式の評価

EasyCert パッケージの各アプリケーションが使用している AiCrypto では表 4 のような公開鍵暗号・共通鍵暗号・ハッシュ関数が使用できる。また、利用可能なファイル形式については表３と同様なのでここでは記述しないものとする。

表 4. 使用可能な暗号・ハッシュ関数

公開鍵暗号	RSA（キー生成可能） ECDSA（パラメータ生成可）
共通鍵暗号	DES（ECB,CBC,CFB） 3DES（ECB,CBC） RC2（ECB,CBC）
ハッシュ関数	MD2, MD5, SHA1, HMAC

OpenSSL ではこれらの他にも幾らかの公開鍵・共通鍵暗号や暗号モードが利用できるが、これらの機能は証明書利用に関する限り一般的に利用されるものではない。そのため、使用できる暗号の種類やファイル形式について、OpenSSL と AiCrypto での差は感じられない。更には、AiCrypto では既に楕円曲線暗号を利用した署名方式 ECDSA に対応しているため、OpenSSL に一歩先んじているといえる。

また、利用可能なファイル形式についてもほぼ同様であり、差はあまりないと考える。しかし、この先楕円曲線暗号が市場に多く普及すると考えられ、そのデファクトスタンダードとして PKCS#13 が公開・定着すると予測される。この場合、証明書や秘密鍵のフォーマット、楕円曲線のパラメータの扱いなどが定義されるとみられ、こうした新しい規格にも迅速に対応していく予定である。

4.2 AiCrypto 速度評価

CA や関連するアプリケーションにとってエンジンとなる暗号ライブラリの速度は非常に重要なものであり、その速度によって使用感が大きく左右される。

そこで今回、OpenSSL と AiCrypto に対し同一環境下 (表 6) でのコンパイル、速度調査を行いその結果を表 5 にまとめた。どちらも C のソースコードからコンパイル生成したライブラリを使用して速度の計測を行った。OpenSSL には Solaris(x86)用のアセンブリコードも付属しており、こちらを使用した場合は表 5(1) の結果より更に倍の速度で動作することが確かめられたが、ここでの数値の表示は行わない。

この比較測定により AiCrypto は速度的に十分実用範囲である事が確かめられた。すなわち、掛け算や 2 乗演算では、Karatsuba-Ofman アルゴリズムにより掛け算を高速化しており、指数演算では 5bit 固定の Window-Method によって高速化を図っている。さらに、除算では速度的に遅い除算命令の使用を減らし、引き算と掛け算による処理に置き換える事で全体的な処理時間を大幅に縮小し、OpenSSL に対して 10 倍以上の高速化を達成する事ができた。

しかし、幾らかの問題点も明らかになった。1 つは、逆元演算の速度が遅いという点である。現在 AiCrypto では、逆元演算に通常の拡張ユークリッド法を使用しており、これを拡張 General Binary 法等に置き換えて演算の高速化を図る必要

表 5. (1)巨大数演算の速度測定

<巨大数演算>

	OpenSSL	AiCrypto
足し算(192bit)	0.2μs	0.3μs
引き算(192bit)	0.3μs	0.3μs
掛け算(192bit)	1.5μs	1.6μs
割り算(192bit)	79.0μs	6.5μs
指数演算(192bit)	0.84ms	2.1ms
逆元演算(192bit)	0.25ms	0.53ms
足し算(1024bit)	1.2μs	1.2μs
引き算(1024bit)	1.2μs	1.2μs
掛け算(1024bit)	25.2μs	33.5μs
割り算(1024bit)	10.34ms	84.0μs
指数演算(1024bit)	71.86ms	136.6ms
逆元演算(1024bit)	2.85ms	4.97ms

※1. 割り算は (384 bit) /(192 bit)の形式
※2. 指数演算は a^e mod n (同一 bit 数)の形式
※3. 逆元演算は a^{-1} mod n (同一 bit 数)の形式
※4. それぞれ小数点以下 3 桁目を四捨五入

表 6. 評価環境

OS:	Solaris2.6 (x86 版)
CPU:	Intel PentiumII 700MHz
コンパイラ:	gcc 2.8.1

表 5. (2)共通鍵･公開鍵暗号の速度測定

<AiCrypto RSA 演算>

秘密鍵生成(1024bit)	平均 1.4sec
署名速度(1024bit)	136.6ms
検証速度(1024bit)	2.26ms

※署名・検証速度は巨大数演算速度より計算

<AiCrypto ECC Point 演算>

倍算(192bit)	95μs
足し算(192bit)	180μs
引き算(192bit)	175μs
掛け算(192bit)	26.25ms

<AiCrypto ECDSA 演算>

秘密鍵生成(192bit)	28.0ms
署名速度(192bit)	8.8ms
検証速度(192bit)	54.7ms

<AiCrypto 共通鍵暗号速度>

	暗号化	復号化
DES ECB (56bit)	3298	3265
DES CBC (56bit)	3265	3232
3DES ECB (168bit)	1103	1095
3DES CBC (168bit)	1095	1088
RC2 ECB (64bit)	3555	4155
RC2 CBC (64bit)	3478	4102

※ 単位は KBvte/sec

がある。また先に記した通り、OpenSSL にはアセンブリコードが付属しており、そのコードによる実行速度は C によるコードの速度の倍の速度であるため、AiCrypto でも Windows や Solaris 等個別に各演算のアセンブラでの記述を行う必要がある。

表５(2) には、RSA の各種演算と楕円曲線暗号の各種演算の実行速度が現されている。RSA の署名速度や検証速度は巨大数演算から計算することができ、その結果、AiCrypto の署名・検証速度は OpenSSL と比べて２倍程度の遅延がある事がわかった。しかし、鍵生成、すなわち素数生成の速度は1024bitで平均 1.4秒程度でOpenSSL と大差ないことが判明した。また、楕円曲線上の演算については E. De Win による論文[9]と比較することができる。この論文では PentiumPro200MHz による速度測定が行われているが、CPU の差を考慮すると、掛け算、署名速度などに関して、2 倍程度の速度差があることが確認された。この差はやはりアセンブリコードによる記述の差によると考えられる。

1 枚づつ順に証明書を作成する場合、2 倍程度の速度差であればあまり問題がないといえる。しかし、証明書の大量発行を行う場合などでは、全体の処理時間に大きな差が出る事になり、アセンブリコードによる記述も含めて高速化を図っていく必要がある。

4.3 EasyCert 操作性評価

ここでは、EasyCert の操作性に関して、コマンドラインから操作する OpenSSL の CA と比較評価を行う。なお、著者が測定の基準としているのは次のような項目である。

・操作の分かり易さ（覚え易さ）
・操作にかかる手間（キーボード入力等）
・設定の行いやすさ
・表示の見易さ

これらの項目を基準として、各操作・設定に対し著者が独自に判断をし評価を行ったものを表 7 に示す。

Windows における GUI はかなり洗練されたものであり、多くの操作においてコマンドラインとは比べ物にならない程の操作性の違いがある。

EasyCert では TreeView や ListView といった Windows 特有の GUI を利用することで、コマンドラインで操作する OpenSSL の CA とは操作性において大きな差を得ることができた。

また、CA や証明書拡張情報の設定において EasyCert ではプロパティーシートを使用して、チェックボックスやラジオボタンをクリックするだけで多くの設定が可能である。この手法により わかり易く、簡単に設定を行うことができる。

さらに、EasyCert の大きな特徴である CA 構築 Wizard を使用することで、証明書技術に関して知識の乏しいユーザであっても無駄なく確実に CA の構築を行うことができる。また、証明書の大量発行 Wizard により作業状況を確認しながら簡単に大量発行を行える。こうした機能に加え、Windows で一般的に行われるインストール・アンインストールもセットアッププログラムが付属しているので多くのユーザが簡単にインストールすることができる。

表 7．操作性評価

	OpenSSL	EasyCert
インストール アンインストール	○	◎
新規 CA 構築	○	◎
証明書要求 秘密鍵作成	△	○
証明書(一括)破棄 CRL 発行	△	◎
証明書(一括)更新	△	◎
証明書 一括大量発行	△	◎
CA 別　証明書,CRL 有効日数設定	△	◎
CA 別 CA ポリシー設定	○	◎
CA 別　証明書 拡張情報設定	△	○
作成済　証明書要求 秘密鍵一覧	×	○
CA 別　発行済み 証明書一覧	△	○
ファイル出力 PKCS#12 等	△	○

4.4　総合評価と考察

これまで、ファイル形式、暗号ライブラリ、操作性のそれぞれについて評価を行ってきた。その結果、ライブラリ性能における多少の弱点が見受けられたが、扱えるファイル形式はほぼ同等であり、操作性に関しては EasyCert が多くの部分で優れていると判断された。また、ICAP や OpenSSL でも Web に対応した CA アプリケーションが存在するが、操作性の面では専用のアプリケーションには及ばない。

総合的な評価として、EasyCert にはほぼ CA・RA を運営するのに必要な機能は備わっており、その利用しやすさ、操作性といった面では十分評価できる。さらに、扱えるファイル形式や暗号性能の面から見ても、実用的な CA・RA としての機能を保持していると考えられる。

5．おわりに

本論文では Windows 上で動作する GUI ベースの CA アプリケーションの開発に関して報告を行った。また、現存の CA と操作性や暗号ライブラリの性能に関する比較評価を行うことで、開発した EasyCert パッケージの有用性を示した。

今後の課題として、大きく 2 つの事が挙げられる。1 つは暗号ライブラリの性能向上であり、もう 1 つは EasyCert の機能向上である。暗号ライブラリの性能向上については、逆元演算アルゴリズムの向上とアセンブラによるコード記述によって多くの速度向上が期待できるため、引き続きコードの改善を行っていく。

また、EasyCert の機能向上についてはスタンドアロン型の CA アプリケーションとして完成度を高めると共に、CA クライアントとしての機能拡張を図っていきたい。すなわち、多くのベンダによって様々な CA サーバが販売されているが、別の種類の複数 CA サーバに対して 1 つの CA クライアントで操作が行えれば、管理運営が軽減できると考えている。このため、現在 UNIX 上で動作している AiCA[10]に対しサーバ機能を持たせ、ネットワーク上での証明書検証を行う OCSP や CA サーバ機能を定義している RFC2510 等に対応したい。さらには、Windows 上で動作する EasyCert を CA クライアントとして機能の差別化を図る予定である。

参考文献

[1] NTT Electronics 魔法便 II
http://www.nel.co.jp/security/index.html
[2] VeriSign
http://www.verisign.co.jp/
[3] OpenLDAP
http://www.openldap.org/
[4] OpenSSL
http://www.openssl.org/
[5] 服部 弘之,櫻井 三子,小林 良至,菊池 浩明, "オンライン認証局パッケージ(ICAP)の実装と評価," SCIS97 (1997)
[6] Privacy Enhancement for Internet Electronic Mail (PEM), RFC1421(1993)
[7] PKCS#7, RSA Laboratory
http://www.rsasecurity.com/rsalabs/pkcs/
[8] PKCS#12, RSA Laboratory
http://www.rsasecurity.com/rsalabs/pkcs/
[9] E. De Win, S. Mister, B. Preneel, M. Wiener, ``On the performance of signature schemes based on elliptic curves,'' *Algorithmic Number Theory Symposium III, LNCS 1423*, J.P. Buhler, Ed., Springer-Verlag, 1998, pp. 252-266.
[10] 若山公威, 奥野琢人, 岩田彰, 村瀬晋二, 鈴木春洋, "暗号ライブラリと認証局パッケージの開発," 第 59 回(平成 11 年後期)情報処理学会全国大会, 情報処理学会

A パッケージ入手経路

本論文で用いられた暗号ライブラリや、認証局パッケージの紙媒体（ライブラリコードを紙に印刷した物）や光・磁気記憶媒体を入手したい場合は、以下の住所に問い合わせを行なってください。

〒466-8555
名古屋市昭和区御器所町
名古屋工業大学 電気情報工学科 岩田研究室

また、本論文の著作者である奥野に対し電子メールを送る事で問い合わせる事も可能です。

okuno@mars.elcom.nitech.ac.jp

さらに、何らかの媒体ではなく電磁気的な記録だけを入手したい場合は、以下の Web サイト
http://mars.elcom.nitech.ac.jp/security/
にアクセスする事によりダウンロードを行なう事が可能です。