第 10 部

利用者認証技術

第 1 章

はじめに

1.1 広域認証技術

インタネットの普及と利用の拡大は、コンピュータネットワークを特定の業務や閉じたコミュニティのネットワークから、多彩なサービス、開かれたネットワークへと大きな変化を生じさせている。それにともない、セキュリティの問題、特に高度なユーザ認証の実現と通信データの機密保護が重要な問題として認識されるようになってきた。

ユーザ認証とは通信相手が本人であることを電子的に確認することである。従来のシステムではパスワードの提示 (入力) で実現されるユーザ認証が一般的であった。

しかし、認証を受けるユーザとシステムがインタネットを介して結ばれている場合には通信路の盗聴を防ぐことが難しいことから、暗号技術を駆使した高度な認証システムが主流となりつつある。

また、WWW(World Wide Web) システムの普及で、研究活動が主体だったインターネットでも、オンライン・ショッピングなどの電子商取引が行われるようになった。電子商取引では、顧客側からはクレジットカード番号や個人の住所などのプライバシ情報を送信する必要があるし、顧客と受注者が相互に身元を確認する必要もある。

WWW システムに限って言えば、SSL(Secure Socket Layer)[66] や SHTTP(Secure Hypertext Transfer Protocol)[67] などのセキュリティ強化機能が考案され、その暗号通信機能によりプライバシの保護はある程度実現されたが、個人単位の認証についてはいまだ実現されていない。[1]

1.2 公開鍵による認証機構と証明書発行局

インターネットのような広域分散環境での認証システムは、RSA 暗号に代表される公開鍵暗号を用いて実現されている。

公開鍵暗号を応用したシステムでは、不特定多数の相手にメッセージの暗号化、認証情報の検証に使う鍵 (公開鍵) を配布し、暗号メッセージの解読、認証情報の生成を使う鍵 (秘

[1](注) S-HTTP も SSL もプロトコル上は個人認証が可能なようになっている。将来的には顧客個人の認証を正しくおこなえるようになるであろう。

密鍵) を所有することを可能とする.

公開鍵暗号システムを認証に使う場合、配布する公開鍵と秘密鍵の所有者の関連づけの確かさがシステムの信頼性となる。この関連づけを実用的に行うための仕組みが公開鍵証明書を用いた認証機構である。

公開鍵証明書は、公開鍵に関連づけられるユーザの身元を保証する第三者機関、証明書発行局 (CA - Certification Authority) によって発行される。証明書は、公開鍵のデータと秘密鍵の所有者情報に対して CA の電子署名を施した電子情報である。証明書による認証機構のユーザは、限られた数の CA の公開鍵を確実に入手することで、CA の発行する証明書を使って多数の証明書 (あるいは公開鍵) の信頼性を検証できる。

公開鍵証明書は、PEM(Privacy Enhanced Mail) や SSL、S-HTTP などすでに多くのシステムの認証機能で利用されているが、CA の運用や証明書利用に必要なサービスの整備など検討すべき問題が多く残されている。

1.3 認証実用化実験の目指すもの

InterAuth-WG では、今後重要視される電子認証技術のなかで、インターネットのような広域分散環境で個人単位の認証をも可能するための技術の開発、広域認証機構の整備を目的とした活動を行っている。

証明書発行局の機能は、インタネット上のサービスの共通基盤として必要なものであり、要となる機能である。ここでは、まず、証明書発行局の概要について解説し, その 10 の技術課題を整理する。

(a) 証明書の検証と検証局 (VA - Verification Authority)

クレジットカードは利用時にオンラインで有効性の確認、いわゆる与信が行なわれる。これに対して、証明書の有効性の検証は、一般的にオフラインの電子署名の検証と、定期的に発行される廃棄された証明書のリスト CRL(Certificate Revocation List) の検索によって行なわれるが、オンラインの検証には及ばない。そこで、証明書の有効性をリアルタイムに確認するサービスを想定し、これを提供する検証局 VA(Verification Authority) の実現を検討している。

なお、認証機構を議論している IETF の PKIX-WG でも同様のサービス (Online-Verification Service) が提案されている [68]。

(b) 証明書

証明書の形式および認証の基本的な枠組は、ITU-T 勧告 X.509 「ディレクトリ— 認証の枠組」にもとづいている。X.509 は、改定が行なわれており現在のバージョンは第 3 版と呼ばれている。これは従来の規約が OSI での利用を想定して設計されており、あまりにも現実の環境にそぐわなかったためである。例えば、バージョン 1 の証明書では、Internet

の名前空間には適切でないという問題があった。こういった反省にたち、証明書のフォーマットに様々な拡張が行なわれて現在バージョン 3 になっている。バージョン 3 で定義されている主な拡張の属性を表 1.1に示す。ところが、今度は属性が多過ぎて運用が非効率的になりかねない。X.509v3 の拡張のうち何を標準として用意すべきか、プロファイルを設定することが検討されている。

表 1.1: X.509v3 の証明書拡張属性

拡張属性	説明
alternative name	メールアドレス、DNS 名など別の識別子
certificate policy	ポリシの識別番号
key attributes	鍵の用途
key usage restriction	鍵の利用制限
basic constraints	発行局になることを制約する
information access	発行局のアドレスと参照
authority key identifier	署名鍵の識別子
CRL distribution point	CRL 配布先

証明書の中には、ユーザだけでなく証明書発行機関などの識別子が必要となってくる。この識別子の割り当て空間の設定、実際の割り当て方法は、証明書の相互運用性を維持するうえでの緊急の課題となっている。

(c) 広域認証実験シナリオ

証明書でどのような実験を行なうべきか。現在、いくつかの提案があがっている。

また、複数の証明書発行局の相互保証という問題がある。基本的には発行局の証明書を発行することによって行なわれる。証明書の発行の方法として、大きく 3 つの方式にわかれる。

1. 自分で自分の証明書を発行する自己保証の方式
2. 複数の証明書発行局がお互いに相互保証を行なう方式
3. 証明書発行局をツリー構造に配置し上位の発行局が下位の発行局の保証を行なう階層的方式

相互認証の各方式は、証明書の目的によって選択されうるもので、公共的な証明書であれば階層的な方式が良いし、特定のサービスのための証明書であれば、自己保証または相互保証による形態になるものと考えられる。

どの方式においても最終的にはいずれかの発行局の証明書を配布する必要あり、証明書の配布方法および、相互保証の方式は、運用上もアプリケーションプログラムにおける鍵

管理の点にも影響を及ぼす。それぞれの方式について技術面はもちろんのこと、運用上の問題、役割とそれに適した方式について検討する。

(d) 更新問題

定期券は切れる前に更新することが出来る。証明書の場合にも同じことはできないだろうか、というポイントから、証明書の更新に関わるいくつかの問題を指摘している。定期券との相違点には、古い証明書もそれを用いて行なった署名が残っている可能性があるので、むやみに消去できないこと、証明書は更新されても鍵はそのままの場合もあること、更新時に一つの鍵に複数の証明書が存在してしまうこと、などがある。

(e) 登録

証明書の発行申請は、オンラインまたは郵便やファクシミリなどの方法を用いて、証明書発行局に送られる。申請方法、申請後の本人確認の手段については、後述するように、基本的には証明書発行局の運用方針による。しかし WWW や電子メイルなどインタネットを用いたオンラインによる申請は、今後のユーザの増加を考えると必須であろう。ユーザの鍵の紛失や盗難など鍵事故のオンラインによる連絡も同様である。ユーザと証明書発行局間での標準的なプロトコルおよびツールの開発が必要となってくる。

本質的な問題は、証明書を発行する際のユーザの認証をどうしたらいいのだろうか。ここに、たまごと鶏問題が生じる。対策としては、

(1) オンラインでパスワードを使う

(2) オフライン

(3) 折衷案、

などがあり、統一的な方法には至っていない。これはむしろ、各 CA に依存して異なる方が自然という意見も出ている。また、秘密鍵をどうやって保管するかという問題も残されたままである。

これに対して、ユーザの代わりに秘密鍵を持ち、証明書の申請から電子署名までを代理で行なう支援システムが提案されている。これは、通常の Web を利用するもので、利用者側の Web クライアントには手を加えないという利点がある。

(f)Web トランザクション

証明書を用いるアプリケーションとして、Web を想定している。Web の上ではこれまでにも様々なセキュリティの提案が行なわれているが、我々の中では、富士通研究所のグループが Secure HTTP の試験実装を行なっている。これは、Web のセキュリティを強化することが目的ではなく、Web を利用して証明書を効果的に分配する枠組を実現することが狙いである。

(g)CA パッケージ

これまで CA の実装は組織の運用ポリシに依存していたので、共通に利用できる汎用なものが存在していなかった。多くの CA を設けるという目的からも、運用の容易なパッケージを用意することが計画されている。

(h)PET3

仮想端末 telnet ではパスワード認証を用いるため、盗聴となりすましの不正行為の可能性があった。そこで、富士通研究所のグループでは暗号化 telnet を試みる。証明書を用いて利用者認証を実現している。

(i) 鍵生成

暗号システムでは、認証や暗号通信に使う乱数の質がシステムの安全性を決めてしまう。妥当な乱数発生アルゴリズムと、鍵の選択方法を検証する。

(j) ポリシ

業務として運用される証明書発行局には程度の差こそあれ、公平性、信頼性、守秘性、迅速性、可用性の確保等が問われる。証明書の信頼性は、証明書発行時の本人確認の方法にかかっている。さらにユーザや場合によっては証明書発行局そのものの秘密鍵の紛失や盗難などが発生したときの手順にも大きく関わってくる。登録時のユーザの情報プライバシの問題は発行局の社会的な信頼をうるうえでも重要である。

証明書発行業務およびそれに付随する鍵有効性チェックや与信機能など業務について、信頼性や守秘性などいくつくか観点でのクライテリアを設け、運用上のガイドラインを策定し、社会的に信頼される環境を構築していく必要がある。

証明書の利用目的によって、証明書発行時の本人確認の手段や、有効期限、鍵の紛失事故や本人の資格喪失などによる失効前の鍵を廃棄する手順などが異なってくる。たとえば、既存の電子メイルの本人確認だけを目的とするなら、比較的簡単な本人確認を行えばよく、鍵の無効手順も簡単でよい。一方、電子商取引に用いられる証明書の場合は、本人確認の時点で支払能力などが調査されるだろうし、取引き額に応じて証明書の有効性のチェックを、その都度行なうかもしれない。将来、公的な機関が証明書を発行するようになれば、その本人確認は公的機関の既存システムとリンクして行なわれることになる。

このような違いを発行局のポリシ (運営方針) とよぶことがある。逆にいうと証明書の発行ポリシによって、証明書の目的や能力、信頼性といったものが異なってくる。現実のシステムでは、目的に応じて、異なったポリシを持った、証明書発行局が複数でてくる。

X.509 の証明書ではポリシを記述する項目もあり、うまく設計すれば、証明書の形式やプロトコルを大きく変更することなしに、同じ枠組で、異なったポリシの証明書を処理することができるようになる。このことは、発行局だけでなくサーバやクライアントソフトウエアの共通化のメリットがでてくる。

第 2 章

証明書の拡張に関する動向

広域ネットワークでのセキュリティ確保には、公開鍵暗号技術の利用が必須である。公開鍵暗号を利用する場合、通信相手の正しい公開鍵を入手する方法が重要であり、その手段の一つとして X.509 の認証フレームワーク [69] がある。これは Certification Authority (CA) という信頼できる第三者が発行する公開鍵証明書を用いて、ユーザの公開鍵を配布する方法である。本節では、version 3 の公開鍵証明書 [70] のプロファイルと証明書確認サービスを紹介する。

2.1 公開鍵証明書

X.509 Version 3 の公開鍵証明書の ASN.1 定義は次の通りである。

```
Certificate  ::=  SIGNED  {  SEQUENCE  {
  version               [0] INTEGER { v1(0), v2(1), v3(2) } DEFAULT v1,
  serialNumber          INTEGER,
  signature             AlgorithmIdentifier,
  issuer                Name,
  validity              Validity,
  subject               Name,
  subjectPublicKeyInfo SubjectPublicKeyInfo,
  issuerUniqueID        [1] IMPLICIT UniqueIdentifier OPTIONAL,
  subjectUniqueID       [2] IMPLICIT UniqueIdentifier OPTIONAL,
  extensions            [3]  Extensions OPTIONAL } }
```

version、serial number、signature、issuer、validity、subject、subject public key info の意味は version 1 の定義通りである。以下、version 2、version 3 の拡張フィールドで ICAT で検討を進めている部分に限って説明する。

2.2 unique identifier

version 2 で導入した CA、ユーザを識別するための BIT STRING。PKIX のドラフト [68] では使わない方向。ただし、本来の使い方とは異なるが、CA やユーザが複数の鍵を持っていた場合にその鍵を識別するために、このフィールドを使う方法もある。また、鍵の移行期 (更新) に用いることも可能かも知れない。

2.3 issuer / subject alternative name

```
AltNames  ::=  SEQUENCE OF GeneralName

GeneralName  ::=  CHOICE  {
  otherName     [0] INSTANCE OF OTHER-NAME,
  rfc822Name    [1] IA5String,
  dNSName       [2] IA5String,
  x400Address   [3] ORAddress,
  directoryName [4] Name,
  ediPartyName  [5] IA5String,
  url           [6] IA5String }
```

この他、IP アドレスなどネットワークアドレスが必要であり、フォーマットを決めなければならない。他のネットワークアドレスやデータリンク、トランスポートアドレスの必要性も検討する必要もある。

2.4 certificate policy

PKIX ではオブジェクト識別子だけを使っている。ISO のドキュメント [70] ではパラメータも含めている。

```
PolicyInformation ::= SEQUNCE OF SEQUENCE {
  certPolicyId       CERTIFICATE-POLICY.&id,
  qualifier          CERTIFICATE-POLICY.&Qualifier { @certPolicyId } OPTIONAL }

CERTIFICATE-POLICY ::= CLASS {
  &id           OBJECT IDENTIFIER UNIQUE,
  &Qualifier    OPTIONAL
} WITH SYNTAX {
  POLICY-IDENTIFIER     &id
```

```
  [QUALIFIER-TYPE       &Qualifier] }
```

例えば、VeriSign のように証明書クラスを分ける場合 [71]、ポリシ自体を分けるのか、それともパラメタ化するのか検討する必要がある。また、そもそもパラメタ化できるようにポリシ自体に何らかの構造を入れることができるのかどうかの検討も必要。

2.5 key attributes

鍵の識別子、秘密鍵 (private key) の使用期間、鍵の用途を示す。

```
KeyAttributes  ::=  SEQUENCE {
  keyIdentifier         KeyIdentifier OPTIONAL,
  intendedKeyUsage      KeyUsage  OPTIONAL,
  privateKeyUsagePeriod PrivateKeyValidity OPTIONAL }

KeyIdentifier ::= OCTET STRING

PrivateKeyValidity ::= SEQUENCE {
  notBefore [0] GeneralizedTime OPTIONAL,
  notAfter  [1] GeneralizedTime OPTIONAL }

KeyUsage ::= BIT STRING {
  digitalSignature (0), nonRepudiation (1), keyEncipherment (2),
  dataEncipherment (3), keyAgreement (4), keyCertSign (5), offLineCRLSign (6) }
```

2.6 key usage restriction

鍵の用途をポリシと用途で制限する。

```
keyUsageRestriction  ::=  SEQUENCE  {
  certPolicySet      SEQUENCE OF CertPolicyId OPTIONAL, -- OID
  restrictedKeyUsage KeyUsage OPTIONAL }
```

2.7 basic constraints

cert path、つまり証明書の発行先を制限する。

```
basicConstraints  ::=  SEQUENCE  {
  subjectType       BIT STRING { cA (0), endEntity (1) },
  pathLenConstraint INTEGER OPTIONAL,
  permittedSubtrees SEQUENCE OF GeneralName OPTIONAL,
  excludedSubtrees  SEQUENCE OF GeneralName OPTIONAL }
```

名前とパス長で制限することができる。ISO ではポリシによる制限 (policy constraints フィールド) も定義しているが、複雑でありどこまで実用性があるか疑問である。

2.8 information access

ISO のドキュメントでは未定義。CA のポリシ、配布局、検証局のアドレスを入れるべきフィールドだと思われる。

2.9 authority key identifier

証明書に署名した鍵の識別に使用。

```
AuthorityKeyId ::= SEQUENCE {
  keyIdentifier     [0] KeyIdentifier OPTIONAL,
  certIssuer        [1] Name OPTIONAL,
  certSerialNumber  [2] CertificateSerialNumber OPTIONAL }

KeyIdentifier ::= OCTET STRING
```

CA の署名鍵が複数ある場合、例えば、署名鍵の移行期には必要かもしれない。証明書のクラスが複数の場合、証明書の署名鍵を分ける可能性もある。PKIX のドラフトでは入っていないが、含めるべきだという議論もある。

2.10 subject directory name

ディレクトリ属性を格納。住所、シャチハタ製印影、顔写真、電話番号などの可能性あり。本来はディレクトリから取り出すべきだが、ディレクトリが普及していない現在、証明書に含めても良いかも知れない。しかし、ディレクトリに格納したとしても第三者による認証がないという問題は残る。

```
AttributesSyntax ::= SEQUENCE OF Attribute
```

2.11 CRL distribution point

Certificate Revocation List (CRL) は、鍵が盗難にあうなどして公開鍵が無効になり廃棄された証明書のシリアル番号を集めて CA が署名したものである。

このフィールドを参照することで、CRL の入手先がわかる。

```
CRLDistPointsSyntax ::= SEQUENCE OF DistributionPoint

DistributionPoint ::= SEQUENCE {
  distributionPoint Name,
  reasons           ReasonFlags OPTIONAL }

ReasonFlags ::= BIT STRING {
  unused (0), keyCompromise (1), caCompromise (2), affiliationChanged (3),
  superseded (4), cessationOfOperation (5), certificateHold (6) }
```

第 3 章

検証局 VA

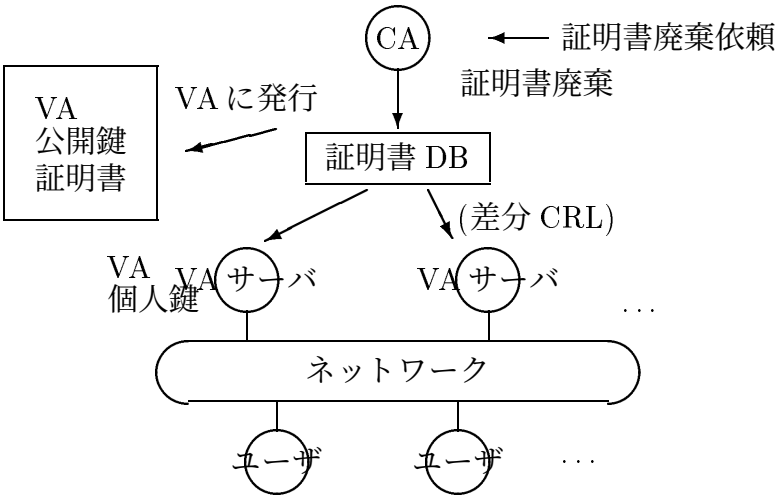
リアルタイムで証明書の検証を行う VA サービスの概念ついては、先に述べたが、ここではサービス機構のアーキテクチャ、ユーザ–VA サーバ間の通信データについて説明する。

3.1 VA アーキテクチャ

全体の構成を下の図に示す。証明書の廃棄依頼が来ると、CA は依頼の正当性を確認してからデータベースに反映し、さらに (例えば CRL を使って)VA サーバに伝える。

ユーザからの検証要求に応じて、VA サーバは証明書の最新の状態、つまり、有効/廃棄済みなどの結果を署名をつけて返す。通信には TCP を用い、BER エンコードした要求/返答データを交換する。

VA は、サーバごとに鍵が異なる、オンライン処理で高い応答性が必要などの理由で、検証を受ける証明書とは異なる署名アルゴリズムや鍵を使う可能性がある。この場合には、primary key attributes フィールドを使い鍵の用途や秘密鍵の使用期間を明示した CA から VA サーバに公開鍵証明書を発行する。



3.2 要求データ構成要素

```
VARequest ::= [APPLICATION 1] OPTIONALLY SIGNED SET {
  version           [0] INTEGER default 0,
  issuer            [1] Name,
                    CHOICE { [2] SEQUENCE {
  subject                        Name,
  issuerUniqueIdentifier  [0] BIT STRING OPTIONAL
  subjectUniqueIdentifier [1] BIT STRING OPTIONAL },
  serialNumber            [3] INTEGER },
  requestVerificationDateAndTime
                    [4] UTCTime OPTIONAL,
  requestOriginator [5] Name OPTIONAL,
  requestIdentifier [6] OCTET STRING OPTIONAL
}
```

主なフィールドの意味は次の通り。

1. version:要求データのバージョンを示す。
2. issuer、subject、issuerUniqueIdentifier、subjectUniqueIdentifier、serialNumber: 有効/廃棄を確認する証明書を識別する。
3. requestVerificationDateAndTime: ユーザが要求する、有効/廃棄を確認する日時。
4. requestOriginator:要求の発信元 (ユーザ) を示す。
5. requestIdentifier:要求データの識別子。

3.3 返答データ構成要素

```
CVSReply ::= [APPLICATION 2] SEQUENCE { SIGNED SET {
  version                      [0] INTEGER default 0,
  issuer                       [1] Name,
                               CHOICE { [2] SEQUENCE {
  subject                        Name,
  issuerUniqueIdentifier         [0] BIT STRING OPTIONAL
  subjectUniqueIdentifier        [1] BIT STRING OPTIONAL },
  serialNumber                   [3] INTEGER },
  requestIdentifier            [4] OCTET STRING OPTIONAL,
```

```
  verificationResult            [5] VerificationResult,
  revokedReason                 [6] RevokedReason OPTIONAL,
  revokedOrHoldDateAndTime      [7] UTCTime OPTIONAL,
  holdExpirationDateAndTime     [8] UTCTime OPTIONAL,
  holdInstructionCode           [9] OBJECT IDENTIFIER OPTIONAL,
  invaliditydateAndTime         [10] UTCTime OPTIONAL,
  requestedDateAndTime          [11] UTCTime OPTIONAL,
  verificationDateAndTime       [12] UTCTime,
  verificationIdentifier        [13] OCTET STRING OPTIONAL },
                               CertificationPath OPTIONAL }

VerificationResult ::= ENUMERATE {
  valid (0), notYetValid (1), finished (2), revoked (3), hold (4) }

RevokedReason ::= ENUMERATE {
  unspecified (0), keyCompromise (1), caCompromise (2),
  affiliationChanged (3), superseded (4), cessationOfOperation (5) }
```

各フィールドの意味は次の通り。

1. version:返答データのバージョン。
2. issuer、subject、issuerUniqueIdentifier subjectUniqueIdentifier、serialNumber: 検証の対象になった証明書を識別する。
3. requestIdentifier:要求データの識別子。
4. VerificationResult: 証明書確認の結果。hold は未確認の廃棄要求があったり、個人鍵が盗まれた可能性があっり、一時的に使用中止状態であることを示す。holdExpirationDate フィールドを使用し使用中止終了予定日時を、holdInstructionCode に証明書の取り扱い方法を示す。
5. revokedReasonFlags: 廃棄した理由をしめす。
6. revokedOrHoldDateAndTime: 当該証明書を廃棄、または一時使用中止となった日時。
7. invaliditydateAndTime: 当該証明書が無効になった日時を示す。鍵が盗難な遭った日時など。
8. requestedDateAndTime: サービスの要求を受け付けた日時。
9. verificationDateAndTime: 有効/廃棄/一時使用中止を確認した日時。

10. `verificationIdentifier`: 返答データの識別子。

11. `CertificationPath`: 返答データの署名を検証するのに必要な公開鍵証明書のリスト。

3.4 まとめ

X.509 Version 3 の公開鍵証明書のプロファイルと証明書確認サービスについて簡単に説明した。今後、実験に参加する CA のポリシを検討した上で、CA 間の相互認証・協調を証明書にどのように反映するかを検討していく。

第 4 章

証明書変更を考慮した管理方式の設計案

証明書を利用するようなアプリケーションを長期運用することを想定すると、以下のタイミングで鍵を変更し証明書を発行しなおす場合が出てくる。

1. 鍵を紛失した時
2. 鍵を人に知られた、あるいは鍵を盗まれた時
3. 証明書の有効期限が切れた時

そこで、本章では、証明書の変更によってアプリケーションにどのような影響や問題点があるかを整理し解決策について考察することにする。

4.1 証明書変更時のプロトコルの問題

ユーザが鍵を変更する際のプロトコルは、ユーザから CA に対して証明書の変更要求を出す部分と、CA で証明書の変更処理をした結果を本人および他のユーザに通知する部分に分かれる。

(1) 要求時

ユーザが CA に対して証明書変更の要求を行う場合、第三者が不正に CA に対して証明書変更の要求を行う可能性が考えられる。そこで、CA では何らかの方法で証明書変更の要求を行ってきたユーザの確認を行う必要がある。

(2) 通知時

証明書変更の通知が遅れた場合には、本人に通知が届くまでに有効期限の切れた古い鍵が不正に利用される可能性がある。また、ユーザの通信相手がそのユーザの証明書をキャッシュしていて、証明書の変更に気づかない場合がある。古い鍵が不正利用された時の被害が深刻となるようなアプリケーションでは、ユーザが毎回 VA に証明書の有効性確認を行う方がよい。

4.2 証明書の有効期限切れ時のサービス中断問題

送信者が鍵を利用する時には有効期限内であったが、受信者が鍵を利用する時には有効期限が切れていて、署名確認やデータの復号に失敗して誤判定する可能性がある。

この問題に対する対策方針としては、以下の方法が考えられる。

1. 発信者であるユーザ A に再度送信を要求する
2. 証明書の有効期限前後のある期間は 2 つの証明書を利用できるようにする
3. 証明書の有効期限以前に早めに鍵を変更する
4. 発信時刻を認証する

発信者がデータを送信した時間のタイムスタンプをデータ中に記録しておき、有効期限内に鍵が利用されたことを確認する。

しかし、第三者を介さない場合では、送信者と受信者双方にとって時刻を偽ることは容易であり、時刻の保証を行うのは困難である。第三者を介す方法では、DTS(digital time-stamping service) 等が実際に有料サービスとして存在する [72]。

4.3 過去の蓄積データの管理問題

証明書が変更されると、古い鍵を利用した暗号化データが読めなくなったり、認証情報 (署名データ) が確認できなくなるという問題が発生する。

暗号化データと署名データの場合に分けて、問題点および対策について考察する。

(1) 暗号化データ

過去の暗号化データに対する要件としては以下の 2 つにまとめられる。

1. 読めるようにする
2. できれば安全に保存する

前者のためには、暗号化データを復号して管理するという方法でもよい。後者のためには、暗号化した状態でデータを管理する必要があり、復号鍵として古い秘密鍵を履歴を管理する方法や、別の暗号化鍵で暗号化して保存する方法が考えられる。

(2) 署名データ

過去の署名データに対する要件としては、署名が正しかった、かつ改ざんされなかったことの保証があげられる。このためには、古い公開鍵を管理することや署名データを確認できたという印を残すなどの方法が考えられる。

しかし、鍵が無効になった後では、署名を第三者への証拠とすることは不可能である。

4.4　CA の証明書変更に関する問題

ユーザの他、CA の鍵の対についても変更する可能性がある。CA の証明書を変更すると、結果としてその CA が今までに発行した証明書は無効になるという問題がある。

また、CA の秘密鍵が漏洩したという理由による証明書の変更はさらに深刻で、証明書が変更されるまでに、架空の証明書が偽造されたり、CA 自身の証明書が変更される可能性があり、発行局の信頼性が失われるという問題がある。

CA の証明書を変更した場合は、必ずオフライン (例えば新聞等) やオンラインの複数の経路を使って、証明書を変更したという通知を全ユーザに知らせなければならない。

4.5　証明書変更方式の設計

前節で述べたように、証明書を利用するアプリケーションにおいて証明書の変更が生じた場合には様々な問題があり、アプリケーションの機能自体に大きな影響を与える可能性がある。ここでは、特にユーザの証明書に変更が生じた場合に必要なユーザ認証の方法、証明書の履歴管理方法の設計について述べる。まず、本設計を行う上での前提条件についてまとめ、用語の定義を行う。

4.6　前提

CA の証明書の配布: CA の証明書自体の有効期限切れについては考慮しない。また、CA の階層の最上位の CA の証明書については、あらかじめ正しく配布されている (例えばアプリケーションのパッケージの中に最初から含まれている等)。

CA の構成: CA の階層における最上位の CA が複数ある状態を許す。すなわち、CA の木構造が独立に複数ある状態を許し、あるユーザは独立した階層下の CA からそれぞれ証明書の発行を受けられるものとする。

CA の証明書とユーザの関係: ある CA が同一のユーザに対して証明書を複数発行しても構わない。

VA と証明書データベースの関係: VA は証明書の有効性を確認して結果を返すサーバで、有効性を確認するために必要なデータを CA が登録する証明書データベースから入手するものとする。このとき、VA と CA は同一の証明書データベースの内容を共有する形となるが、VA と CA の間での証明書データベース内容の同期の取り方やその場合のデータ転送形式等についてはここでは解決されているものと仮定する。

また、ある証明書を変更した場合、証明書データベースには、その証明書を変更したという情報が (例えば CRL の形式によって) 登録されるものとする。

4.7 用語の定義

CA および認証局: CA のユーザ認証の機能と証明書発行機能を分離して、それぞれ認証局および CA と再定義する。認証局と CA は各々鍵の対を持ち、CA と認証局の認証は署名の確認によって行うものとする。

証明書 ID: ユーザは、同一の、または複数の CA から 1 つ以上の証明書の発行を受けることが可能である。あるユーザの複数の証明書から証明書を一意に識別するために、証明書 ID を、以下のように定義する。

証明書 ID
= ユーザ ID + CA 名 + シリアル番号

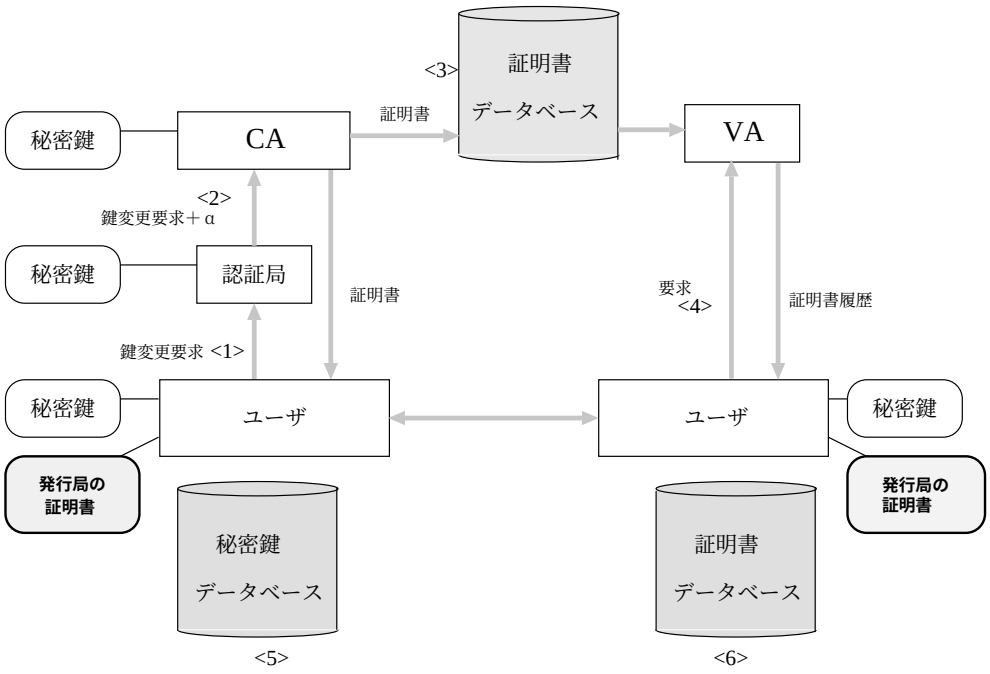
4.8 全体の流れ

ユーザの証明書を変更する際の一連の情報の流れを示すと図 4.1のようになる。つまり、認証局でユーザ認証を行い、CA で証明書を発行し、証明書データベースに変更済みの証明書自身や証明書が変更したという情報を登録する。

図 4.1中の$< 1 > \sim < 6 >$は、証明書変更時に技術的に検討すべきポイントを示している。証明書や秘密鍵の履歴管理 (古い秘密鍵や古い証明書そのものをどう管理すべきかという点、図 4.1中$< 3 > \sim < 6 >$) を除けば証明書の作成時の流れと同じである。本設計では、ユーザ認証、証明書や秘密鍵の履歴管理、履歴入手のインターフェイスに分類して検討する。

4.9 設計方針

ユーザの証明書を変更する場合、認証局において変更を要求してきたユーザをどのように認証するかは、アプリケーションの性質やポリシによって変わる。また、ユーザの古い証明書を利用する必要があるかどうかに関しても、アプリケーションによって異なる。そ



<1> 証明書変更時の、ユーザと認証局間の認証
<2> CAと認証局間の認証
<3> VAでの証明書の履歴管理
<4> 証明書の履歴を入手するためのインターフェイス
<5> ユーザ毎の秘密鍵の管理
<6> ユーザ毎の相手の証明書の履歴管理

図 4.1: 証明書変更時のシステム全体の流れ

こで、証明書変更方式の設計に当たっては、具体的な機能をアプリケーションが選択できるようにするために、個々の機能に対して複数の方式を用意する。

4.10　証明書変更時のユーザの認証方法

(1) アプリケーションと認証レベル

アプリケーションがサービスを提供する場合、証明書作成や変更時にどのような認証レベルを要求するかについて、アプリケーションのタイプ (表 4.1) 別の分類例を表 4.2に示す。

(2) オフラインによる認証方法

表 4.1: アプリケーションのタイプ分類

タイプ	アプリケーションの特徴	不正な証明書発行によるサーバ側の不利益
A	金銭的取り引き	被害額負担、信用低下
B	個人情報提供	(間接的) 被害額負担、信用低下
C	組織内共有情報提供	競合他社との競争力低下、信用低下

表 4.2: アプリケーションのタイプ別認証レベルの分類 (例)

レベル	タイプ A	タイプ B	タイプ C
高	高額取り引き	機密データ (医療カルテ、秘話等)	機密データ (新技術情報、人事情報等)
中	中額取り引き (100 万円程度まで)	評価データ等 (成績等)	組織データ (電話帳、組織図、顧客名簿等)
低	少額取り引き (1 万円程度まで)	低機密データ (所属、プロフィール等)	低機密データ (書類テンプレート等)

不正な変更要求を受け入れた場合の被害が大きいという理由で、高い認証レベルを必要とするアプリケーションについては、電話や郵便等オフラインの方法を利用するのが現実的である。この場合、処理が終わるまでは証明書の利用を停止する必要がある。以下にいくつかの方法を列挙する。

(2.1) 調査会社 (第三者) を利用した認証

調査会社 (第三者) がユーザについてのブラックリストを管理し、これをもとにして認証を行いその結果を認証局に伝える。システムに依存しない認証が可能であるが、調査会社の信頼性をどのようにして確立するかが課題である。

(2.2) 郵便を利用した認証

変更要求を行うと認証局がユーザに対して葉書を郵送し、ユーザは受け取った葉書を持って認証局に行き証明書の発行を受ける。

(2.3) 電話を利用した認証

変更要求時には認証を行わずに、変更結果を本人に通知する際に認証を行うために電話で直接ユーザに確認を求める。電話番号は証明書を最初に発行するときに登録しておく必要がある。

(3) オンラインによる認証方法

アプリケーションを利用するユーザ数が非常に多い場合や、証明書の変更時の処理時間がサービス提供に重大な影響をもたらす場合、オンラインの方法を利用して迅速かつ大量に証明書変更処理を行う機能を必要とする。

証明書の変更をオンラインで行うためには、証明書変更要求フォームは認証局への配送途中で書き換えられないようにする (完全性を保証する) 必要がある。

(3.1) ユーザの証明書を用いた認証

証明書の変更要求フォームをユーザがその証明書の対となる秘密鍵を使って署名し、認証局がユーザの公開鍵で確認する。

この方法では、認証が自動化できるという利点があるが、4.1節で述べたように、ユーザの秘密鍵が盗まれた場合、第三者に悪用される可能性がある。

(3.2) アプリケーションと独立した他の証明書を用いた認証

方法は (3.1) と同様である。ユーザは少なくとも 2 つの独立した階層の証明書を持たなければならない。

(3.3) 証明書作成時の登録情報を用いた認証

証明書作成時に証明書変更に必要な情報を認証局に登録しておき、その情報を利用して認証を行う方法が考えられる。この方法ではユーザにとって秘密に管理すべき情報が新たに発生するため、アプリケーションで利用する秘密鍵とは独立に管理しておくことが大前提となる。

a) Kerberos[73] 等の他の認証システムを利用する

ユーザと認証局が同一組織内に存在しているような場合、同一組織内にある Kerberos サーバ等の第三者による仲介を利用して認証を行う方法が挙げられる [74]。

証明書変更要求フォームの配送途中での改ざん防止のために、フォームを Kerberos のセッションキーで暗号化する。

b) パスワードを登録して利用する

証明書作成時に認証局にユーザのパスワードを登録し、証明書変更時にパスワードを利用してユーザの認証を行う。登録時には、パスワードの盗聴を防ぐためにパスワードを認証局の公開鍵で暗号化する等の工夫が必要である。パスワード利用時には、リプレイアタックを防ぐためにパスワードを含む情報が毎回異なっている必要がある。これに対しては、変更要求フォームとパスワードを認証局の公開鍵で暗号化して送信する等の方法が考えられる。また、変更要求フォームを一時的な鍵で暗号化し、この鍵とパスワードを認証局の公開鍵で暗号化する方法もある。

c) 公開鍵を登録して利用する

ユーザは証明書作成時に公開鍵と秘密鍵の組を 2 組作成する。図 4.2のように、証明書作成時にはパスワードの代わりにユーザのもう一つの公開鍵 (Sp2) を登録する。証明書変更時には、ユーザは証明書変更要求フォームを Sp2 に対応する秘密鍵 (Ss2) で暗号化して送信する。認証局では登録しておいた Sp2 を利用して証明書変更要求フォームを復号できるかどうかによってユーザ認証を行う。

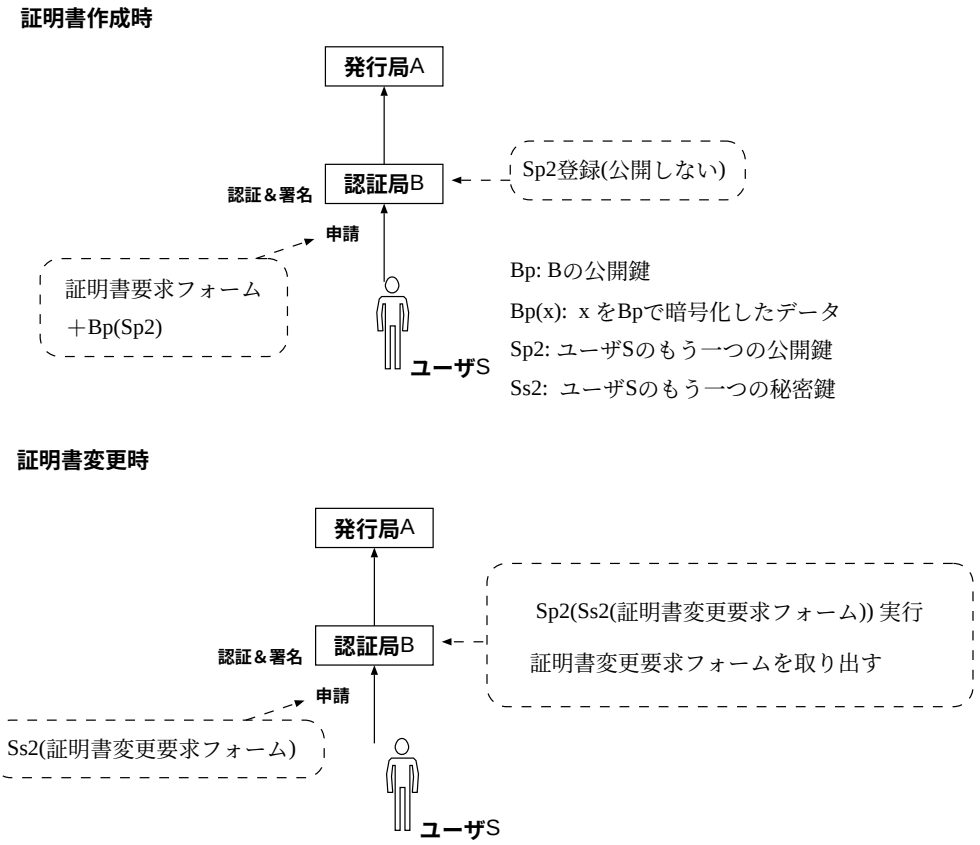


図 4.2: ユーザのもう一組の公開鍵と秘密鍵を利用した認証

この方法では、Sp2 が認証局以外に知られても、Ss2 を持つユーザ以外は証明書の変更ができないため、認証局での Sp2 をそのままの形で管理できる。また、認証局自身の鍵管理と Sp2 の管理を独立に行うことができる。

4.11　証明書や秘密鍵の履歴管理

認証情報や暗号データの利用期間の観点でアプリケーションを分類すると、表 4.3のようなレベルが得られる。前提によれば、証明書を変更し古い証明書が無効になったという状況確認は VA によって確認することができる。しかし、証明書の有効性確認だけではなく、表 4.3のように変更して古くなった証明書 (に含まれる公開鍵) 自身が必要となる場合があるため、証明書の履歴管理について考慮する必要がある。

表 4.3: アプリケーションと証明書利用期間の関係

レベル	証明書利用期間	証明書の履歴管理
リモートログイン	サービス開始時のみ	不要
短い有効期限つき文書	文書の有効期間 但し、証明書の有効期間 $\supset$ 文書の有効期間	不要
長い有効期限つき文書	文書の有効期間 但し、証明書の有効期間 $\subset$ 文書の有効期間	必要
電子メール、電子承認	∞	必要

(1) 証明書の履歴管理

証明書の履歴管理は、VA で行うか、ユーザで行うか等の選択肢がある。

(1.1)VA で全ての証明書の履歴管理を行う場合

VA で証明書の履歴管理を行う利点は、ユーザが個別に証明書を保存する必要がないということである。

VA で扱う証明書の数が多く頻繁に証明書の入手が行われる場合、保管能力と処理能力が問題となる。保管能力については、証明書のデータが 1Kbyte 程度と小さいことから現実に管理可能な容量であると考えられる。処理能力については、VA のキャッシュ技術や負荷分散技術を利用して処理能力の低下を防ぐ必要がある。

他の問題点として、全ての証明書の履歴を長期間公開することによって、公開鍵から秘密鍵が求められる可能性がある。この問題点に対しては、有効性確認時と同様に、履歴入手の際のオーソライゼーション機能についての考慮を行う必要がある。

(1.2) ユーザが通信相手の証明書の履歴管理を行う場合

自分の通信相手の証明書のみを履歴管理すればよいため、保管能力については問題にならない。しかし、他のユーザの証明書が変更されたタイミングを正確に知ることが難しいという問題がある。VA へのアクセスを頻繁に行う必要のないアプリケーションでは、このような履歴管理で十分な場合がある。

(2) 秘密鍵の履歴管理

秘密鍵の履歴は必要な時以外はオンラインでアクセスできないように管理する必要がある。自分の複数の秘密鍵を管理する場合、その識別を行うためにはその秘密鍵がどの公開鍵 (すなわち証明書) の対であるかという情報が必要である。秘密鍵の履歴管理を行う場合、証明書と同様に証明書 ID から履歴を入手する方法を提供するのがよい。

4.12 証明書や秘密鍵の履歴入手のためのインターフェイス

証明書や秘密鍵の履歴を入手しようとする場合、特定の証明書を取り出したり、特定の条件を満たす証明書リストを取り出すためのインターフェイスについて、表 4.4のように定義する。もちろん、アプリケーションにとって必要なインターフェイスを選択して提供することが重要である。

今回提案した履歴入手のためのインターフェイスは VA に証明書の有効性確認の問い合わせを行うためのインターフェイスとは異なっている点がある。今後は、2 つのインターフェイスの整合性について検討を行うことが課題である。

表 4.4: 履歴入手のためのインターフェイス例

要求	指定すべきインターフェイス
特定の証明書	証明書 ID (ユーザ ID + CA 名 + シリアル番号)
あるユーザについて 　現在有効な証明書のリスト 　現在無効な証明書のリスト 　1993 〜 1994 年に有効だった証明書のリスト	 ユーザ ID ユーザ ID + invalid ユーザ ID + 1993–1994
ある CA の発行した証明書のリスト	CA 名
特定の証明書に対応する秘密鍵	証明書 ID + secret

ユーザ ID　[発行局名]　[シリアル番号]　[日時]　[invalid]　[secret]

第 5 章

証明書を用いたプライバシ強化通信 PET

本節では、PET (Privacy Enhanced Telnet) について述べる。クライアント/サーバの形態を持つアプリケーションの基本的なプライバシ強化プロトコルとそのプロトコルを応用した PET について解説する.

5.1 提案プロトコル

サーバ、クライアント間におけるプライバシ強化に対して、基本的なプロトコルを提案している。提案プロコトルの概要を以下に示す。

- 暗号化、改竄検出、なりすましの防止などの様々なセキュリティに対する要求を反映するセキュリティレベルの概念の導入.
- サーバ、クライアント双方のサービスに対する要求（以下ポリシーと表す。例えば、「自分は認証されたくない」など。）を反映するセキュリティレベルの折衝の導入。
- リプレイアタックと秘密情報の最小化を考慮した暗号化通信路確立方式 [75]。
- 証明書をベースとする認証と管理。

5.2 セキュリティレベル

ネットワークセキュリティ上の問題は、盗聴、改竄、なりすまし、リプレイアタックなど様々である。サービス提供者は、サービスの内容によって盗聴防止、改竄検出、なりすまし防止などの手段を考慮する。その手段の組合せは、サービスの内容によって多種多様である。セキュリティレベルは、ネットワークセキュリティ上の問題を解決する手段の組合せを表すものである。例えば、`` セキュリティレベル３は、暗号化と証明書をベースとするクライアント認証の組合せ”、などと表す。実際に PET で定義したセキュリティレベルを表 5.1に示す。

これらのセキュリティレベルは、サーバ、クライアント双方で設定することが可能である。

表 5.1: セキュリティレベル

レベル	内容
1	通常の Telnet 通信
2	暗号化
3	暗号化＋証明書によるクライアント認証
4	暗号化＋証明書によるサーバ認証
5	暗号化＋証明書によるクライアント、サーバ認証

5.3 セキュリティレベルの折衝

サーバ、クライアント間のプライバシ強化に対するポリシーは、同一サービスでも一致するものとは限らない。そのため、提案プロトコルでは、セキュリティレベル変換テーブルを用いたセキュリティレベルの折衝を行なう。PET でのセキュリティレベルの折衝の様子を表 2 に示す。表で、数字はセキュリティレベル、“×” は折衝に失敗して、通信が切断されることを表している。

表 5.2: セキュリティレベルの折衝

client(↓)/server(→)	1	2	3	4	5
1	1	2	×	4	×
2	2	2	×	4	×
3	×	2	3	×	5
4	×	×	×	4	×
5	×	×	×	4	5

5.4 暗号通信の確立

暗号通信を考慮する時、問題となる点を以下に示す。

1. セッション鍵の共有
2. マスター鍵の廃止、更新
3. スケーラビリティ

クライアント	サーバ
	1. PK_Sを送信
2. PK_Sを受信	
3. DEK_1を生成	
4. $PK_S(DEK_1)$ を送信	
	4. $PK_S(DEK_1)$ を受信
	5. $PK_S(DEK_1)$ を SK_Sで復号化し、DEK_1を入手
	6. DEK_2を生成
	8. $DEK_1(DEK_2)$ を送信
9. $DEK_1(DEK_2)$ を受信、復号化し、DEK_2を入手	

図 5.1: 暗号通信路確立のまでの流れ

4. リプレイアタック

5. マスター鍵の管理

6. 暗号方式の強度

1, 2, 3 については、公開鍵暗号方式を採り入れたこと、4 については、セッション鍵の共有方法に工夫を加えたこと、5 については、マスター鍵を秘密鍵で暗号化（パスワード入力方式をとり、そのパスワードが秘密鍵を暗号化したものの鍵となる。スマートカードについては考慮中）することで解決している。提案プロトコルの暗号通信路が確立するまでの流れを図 5.1に示す。図で、PKは公開鍵、SKは秘密鍵、添字の s, c はサーバ、クライアント、DEKはセッション鍵 (Data Encryption Key) を表している。

図からわかるように、サーバのみが秘密情報を持っている。そのため、すべてのユーザにマスタ鍵の廃止や更新を通達する必要がなく、マスター鍵の廃止、更新が容易である、また、そのことで、スケーラビリティに十分対応していると言える。ここで、図の 1 で公開鍵を証明書の形式でを送信しないのは、暗号通信のみを要求するポリシーを考慮したためである。さらに、クライアントから送られてくるセッション鍵をそのまま暗号通信に使うのではなく、最終的にサーバーが生成するセッション鍵で暗号通信を行なう。そのため、リプレイアタックを防ぐことが出来る。この時、サーバからクライアントにセッション鍵を送る暗号方式に慣用暗号方式を採り入れることで、高速化を計っている。なお、6 についてPET では、公開鍵暗号方式に RSA、慣用暗号方式に DES CFB(Cipher Feed Back) モードを採用している。

クライアント	サーバ
1. $CERT_S$を送信	
	2. $CERT_S$を受信 3. $CERT_S$を $CERT_I$ の中の PK_Iで検証 4. ACK or NACK を送信
5. ACK の時継続、 NACK の時切断	
	6. 乱数 rを発生、送信
7. rを受信 8. $SK_C(r)$ を送信	
	9. $SK_C(r)$ を受信、$CERT_C$ の中の PK_Cで r'を取り出す 13. r=r'の時 ACK $r \neq r'$ NACK(切断) を送信

図 5.2: 証明書をベースとする認証

5.5 認証

認証方式は、証明書をベースとする認証方式を採用する。これは、インターネットで一般に利用されているパスワード認証は、辞書攻撃などの問題があることと、ゼロ知識証明など他の認証方式は、インターネットのような広域な環境ではスケーラビリティを得られないためである。もちろん提案プロトコルでは、セキュリティレベルの記述によって、他の認証方式を取り込むことも可能である。PET で採用した証明書をベースとする認証方式を図 2 に示す。図で、$CERT$は証明書、添字の Iは発行局、rは乱数を表す。

クライアントがサーバを認証する時は、図 5.2の手順が逆となる。

5.6 管理

提案プロトコルでは、証明書の情報によってユーザまたはサーバの管理する。また、証明書の形式は ITU-TS 勧告 X.509[69] を採用する。証明書の情報を表 3 に示す。

証明書による認証は、基本的に証明書情報の中の、発行局が独自につけるシリアル番号

表 5.3: 証明書の情報

内容	例
シリアル番号	754
有効期限	1996 年 1 月 1 日〜 19981 月 1 日
発行局の名前	/c=JP/o=ICAT/ou=Certification Authority
ユーザ氏名	/c=JP/o=ABC/ou=Dep./cn=Yamada
暗号方式	RSA
公開鍵	7e29b196fd3ecead9 ···
メールアドレス	taro@abc.co.jp
署名方式	md2 with RSA
署名	86d7bc4d24ea ···

を参照する。しかし、個人対組織、組織対組織の通信を考慮した時 [76]、シリアル番号のみの情報では不十分である。例えば、「A 社にはアクセスを許すが、他の組織には許さない」というポリシーで運用しているサービスがあるとすれば、管理情報には、A 社のユーザまたはサーバすべてを認識する仕組みが必要である。また同様に、証明書発行局は単一であるとは限らない。現実の世界でも「A カード会社は許すが B カード会社は許さない」「ゴールドカードは使えるが、普通のカードは使えない」などの要求があるように、発行局を識別する仕組みも必要である。PET では、ユーザ管理ファイルにワイルドカードを示す “*” 記号を用いることにより、この仕組みを実現している。

管理情報ファイルの例：

754:/c=JP/o=Certification Authority:/c=JP/o=ABC/ou=Dep./*

この例では、“:” がデリミターとなっていて、第 1 カラムがシリアル番号、第 2 カラムが発行局、第 3 カラムがユーザ名となっている。“*” は、発行局に指定することも可能である。なお、この例の意味は、「ABC 社の開発部のユーザすべてにアクセスを許す」である。

5.7　提案プロトコルのまとめ

提案プロトコルのまとめとして、プロトコルの簡単な流れを説明する。提案プロトコルでは、セキュリティレベル５の場合、以下の手順をふむ。

1. 低レベル層の通信の確立
2. セキュリティレベルの交換と折衝

3. セッション鍵の共有

4. 暗号通信開始

5. クライアント認証

6. サーバ認証

7. サービスの開始

この流れは、2 のセキュリティレベルの交換と折衝によって決定したセキュリティレベルによって変化する。PET で定義したセキュリティレベルによる流れの変化を以下に示す。

レベル 1 1 → 2→ 7

レベル 2 1 → 2 → 3 → 4 → 7

レベル 3 1 → 2 → 3 → 4 → 5 → 7

レベル 4 1 → 2 → 3 → 4 → 6 → 7

レベル 5 1 → 2 → 3 → 4 → 5 → 6 → 7

5.8 応用：Privacy Enhanced Telnet

5.8.1 仕様と処理速度

今回、提案プロトコルを遠隔計算機ログインプロトコルである “telnet” に応用した (PET)。PET の仕様を以下に示す。RSA は OSISEC RSA ライブラリ、DES は Eric DES ライブラリを利用している。

SS2(30MIPS) で測定した処理速度を以下に示す。なお実際の暗号通信では、8byte の CFB モードを利用しているため、セッション時の実際の速度は、約 1Mbps にとどまっている。

5.9 PET の動作

1. PET クライアント

   PET クライアントは、p(pet の p) オプションに何も指定しなければサーバのセキュリティレベルが反映される。例えば namachu ホストがセキュリティレベル 5 で立ち上がっている場合、以下のようになる。

表 5.4: PET の仕様

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)
乱数発生方式	srandam(8) + 時間情報
通信路暗号方式	DES in Cipher Feed Back mode (CFB)
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
ハッシュ関数	Message-Digest MD2
証明書の署名	RSA

表 5.5: 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

```
% pet namachu
Trying 133.160.39.17...
Connected to namachu.
Escape character is '^]'.
You don't set your security level.
Now, I set your security level to remote server's
 security level 5.
Remote server is requesting security level 5.
We are going to talk on level 5.
Now, exchanging a data encryption key. Please wait.
Your name is
          C=JP
          O=Fujitsu Laboratories Ltd.
          OU=Information Processing Network Center
          CN=Yasutsugu Kuroda
          emailAddress=kuroda@flab.fujitsu.co.jp
Your certificate is valid.
Your authentication is valid.
Remote server name is
          C=JP
          O=Fujitsu Laboratories Ltd.
          OU=IPNC
          CN=namachu.center.flab.fujitsu.co.jp
          emailAddress=kuroda@namachu.center.flab.fuj
itsu.co.jp
Server's certificate is valid
Server's authentication is Valid

SunOS UNIX 4.1 (namachu) (ttypb)

login:
```

p オプションに、数字でセキュリティレベルを指定すると、以下のようになる。

```
% pet -p 3 namachu
Trying 133.160.39.17...
Connected to namachu.
Escape character is '^]'.
You are requesting security level 3.
Remote server is requesting security level 5.
We are going to do security level 5.
Exchanging data encryption key.....
Remote server name is
         C=JP
         O=Fujitsu Laboratories Ltd.
         OU=IPNC
         CN=namachu.center.flab.fujitsu.co.jp
         emailAddress=kuroda@namachu.center.flab.fuji
tsu.co.jp
Server's certificate is valid
Server's authentication is Valid
Your name is
         C=JP
         O=Fujitsu Laboratories Ltd.
         OU=Information Processing Network Center
         CN=Yasutsugu Kuroda
         emailAddress=kuroda@flab.fujitsu.co.jp
Your certificate is valid.
Your authentication is valid.

SunOS UNIX 4.1 (namachu) (ttyp8)

login:
```

サーバとクライアントのセキュリティの折衝が噛み合わず、通信が切断される時は以下のようになる。

```
% pet -p 4 namachu
Trying 133.160.39.17...
Connected to namachu.
Escape character is '^]'.
You are requesting security level 4.
Remote server is requesting security level 5.
Security level negotiation is fault.
```

通常の telnet コマンドでアクセスして、かつ、PET サーバがル 1 以外のセキュリティレベルを要求しいる時は以下のようになる。


```
% pet -p 4 namachu
Trying 133.160.39.17...
Connected to namachu.
Escape character is '^]'.
You are requesting security level 4.
Remote server is requesting security level 5.
Security level negotiation is fault.
```

通常の telnet コマンドでアクセスして、かつ、PET サーバがル 1 以外のセキュリティレベルを要求しいる時は以下のようになる。


```
% telnet namachu
Trying 133.160.39.17 ...
Connected to namachu.
Escape character is '^]'.
This server is a Privacy Enhanced Telnet.
It is requesting security level 5 communication.
Please access it using Privacy Enhanced Telnet at
the next time. Bye.
Connection closed by foreign host.
```

pet で通常の telnetd にアクセスできる。

```
% pet flab
Trying 133.160.32.1...
Connected to flab.
Escape character is '^]'.

SunOS UNIX (azalea)
```

2. 公開鍵証明書によるユーザ管理

PET3 のセキュリティレベル３、５では、公開鍵証明書によるユーザ管理を行なう。ユーザ管理ファイル (petd の Makefile.generic で定義した SERVERMANAGEMENTFILE) は、一行に一ユーザの公開鍵証明書のシリアル番号となっている。

```
例
----------------------
36F
234
587
458
・
・
・
---------------------
```

このファイルにユーザを登録するためには、peek コマンドを用いる。

```
% peek -s < CERTFILE >> petuser
```

これで、petuser ファイルに CERTFILE の中の証明書のシリアル番号が追加される。ユーザを削除する時は、peek コマンドで証明書のシリアル番号を確認し、その番号を petuser ファイルから削除する。

```
% peek -c < .cert
Serial No = 36F <---- この番号が、証明書のシリアル番号。
Validity        from 950317025240Z
                to   970317025240Z
issuer:
        C=JP
        O=WIDE
        OU=Certification Authority
subject:
        C=JP
        O=Fujitsu Laboratories Ltd.
        OU=Information Processing Network Center
        CN=Yasutsugu Kuroda
        emailAddress=kuroda@flab.fujitsu.co.jp
signature:
        md2WithRSAEncryption
publickey:
        alg = rsaEncryption
```

3. ファイル転送

   telnetx の機能を追加したことにより、zmodem, xmodem, bplus 等を使ってファイル転送を行なうことが出来ます。詳しくは、TELNETX のマニュアルを読んで下さい。

   注意）セキュリティレベル 2 以上（暗号通信）の時、zmodem で text file を 6150byte 以上アップロードしようとすると、ハングする。

   binary データについては問題ない。

5.10 まとめ

本節では、証明書発行局を中心とした環境でのサーバ、クライアント間のセキュリティ強化に対する基本的なプロトコルを提案した。また、提案プロトコルを応用した PET について紹介した。今後の課題を以下に示す。

- 複数発行局への対応
  カード会社が複数存在するように、将来発行局が複数存在する環境が予想できる。このような環境でのプライバシ強化の枠組が問題となっている。

- 証明書発行申請方式
  ネットワーク社会での認証の枠組はネットワーク社会の中で閉じるべきであると考える。しかし現実には、証明書を発行する際のユーザ認証は、書面や運転免許証などを信頼点にする他ない。ネットワーク上で、個人を初期認識する方法が必要である。

- 証明書検証局
  証明書検証局は、証明書の状態（有効/無効/サスペンドなど）を検証するものである。これは、証明書の期限が有効でもユーザの都合によって無効になる場合があるので、発行局を中心とするシステムでは必須である。

- 証明書の更新
  更新の手続きをどのように行なうか。また、無効になった鍵の管理や、その鍵で暗号化されたデータをどのように扱うかが問題となる。

- 証明書の廃止通知
  分散された環境で、すべてのユーザに、どのように同期して廃止を通知するかが鍵となる。現在では有効な方法がないのが現状である。

- ポリシーの記述方式
  管理者やユーザのポリシーをどのようにシステムに反映させるかが問題となっている。X.509 version 3 の動向が注目される。

- 改竄検出機能追加の検討
  RSA による署名や keyed MD5[77] などの改竄検出機能の追加を検討する。

- セキュリティレベルのオーソライゼーション
  インターネットのような広域な環境では、セキュリティレベルの定義はオーソライズされたものでなければならない。今後 IETF などに提案していく必要がある。

- 提案プロトコルの他のシステムへの応用。
  PET は、発行局を中心とする環境でのアプリケーションのプロトタイプである。今後 FJPEM[78, 79, 80, 81] などの経験をもとに、電子決済システムなどのシステムに応用していく。

今後、インターネットの発展による急速なネットワーク社会の実現は避けられない。そこで用いられるプライバシ強化の手段は発行局を中心としたシステムであると推察する。現在活発に活動している IETF PKIX WG(Public Key Interface (X.509) WG) や ISPP WG(Internet Secure Payment Protocol WG) などの活動に注目したい。