情報処理学会論文誌 Vol. 52 No. 2 621–632 (Feb. 2011)

証明書検証サービス高速化方式の開発

藤城孝宏[†1,†2] 佐藤茜[†1] 熊谷洋子[†1]
鍛忠司[†1] 羽根慎吾[†1]
手塚悟[†3] 岡田謙一[†2]

近年，電子商取引や，電子政府のための基盤として，国内外で多くの認証局が構築されてきている．日本政府においても電子政府実現に向けて，その基盤となる認証システムである政府認証基盤（GPKI）や公的個人認証サービスの構築を行っている．これらの公的な認証基盤では，各府省の認証局間や民間の認証局との間での連携を行うために，ブリッジ認証局方式を採用している．しかしながら，ブリッジ認証局方式には，利用者にとって電子証明書の検証処理が複雑になるという問題がある．この問題の解決のため，著者らは，従来，利用者側で行っていた検証処理をサーバ側で一括して行うことにより，利用者側の負担軽減と，検証処理の高速化を行えることを提案してきた．本論文では，この検証処理をさらに高速化する手法の提案とその有効性について報告を行う．

Development of a Speed up Method of Certificate Validation Service

Takahiro Fujishiro,[†1,†2] Akane Sato,[†1]
Yoko Kumagai,[†1] Tadashi Kaji,[†1] Shingo Hane,[†1]
Satoru Tezuka[†3] and Kenichi Okada[†2]

Government Public Key Infrastructure (GPKI) has been developed for actualization of Electronic Government. A Bridge Certification Authority model is adopted with GPKI to make communication between Ministry or Agency Certification Authorities and Private Certification Authorities. However, the public key certificate validation process in the Bridge Certification Authority model is complicated for the user. To reduce this problem and make the public key certificate validation process faster, we proposed a certificate validation model. In this paper, we proposes a new speed up method of Certificate Validation Service.

1. はじめに

近年のインターネットの爆発的普及を受けて，電子商取引や，電子申請・申告の実現にむけた取り組みが各所で行われている．対面で行う取引，手続きや，専用線などで行われてきたEDI（Electronic Data Interexchange）と異なり，インターネット上で，これらの商取引，申請・申告を行う場合には，その相手が正当な相手なのかを確認する手段が必要になる．そこで，公開鍵暗号技術に基づく認証基盤（公開鍵認証基盤，PKI）が着目され，多くの認証局が構築されてきている．

日本政府においても，政府認証基盤（GPKI）をはじめとし，地方公共団体組織認証基盤（LGPKI）や，住民に証明書を発行する公的個人認証サービスが構築され，サービスを開始している．このような背景のもと，GPKIのような複数の認証局が連携する認証基盤では，証明書の検証処理において，利用者の負担が増大するという問題があることにいちはやく着目し，利用者の証明書検証処理を代理で行う証明書検証サーバの研究を行ってきた．

本論文は，証明書検証サーバにおける証明書検証処理の高速化を行う手法の提案と，その有効性を報告することを目的とする．2章では，日本における公的な認証基盤の概要を示し，複数の認証局が連携する際に証明書の検証処理が問題となることを示す．また，証明書検証に関する利用者の負担を軽減する証明書検証サーバについて示す．3章では，証明書検証処理を高速化する提案について示す．4章においては，性能実験の結果を示し，5章においてその結果と検討を行い，提案手法の有効性について示す．6章において，証明書検証サーバを利用する際のセキュリティに関して考察し，最後に7章において，本論文のまとめとして結言を述べる．

2. 公的認証基盤の概要

2.1 公的認証基盤の構成

国内の公的認証基盤は**図1**のように構成される[1)–3)]．GPKIでは官職認証局を構築し，各府省の処分権者である官職（官職End Entity（官職EE））に対し証明書を発行している．

†1 日立製作所システム開発研究所
Systems Development Laboratory, Hitachi, Ltd.
†2 慶應義塾大学
Keio University
†3 東京工科大学
Tokyo University of Technology

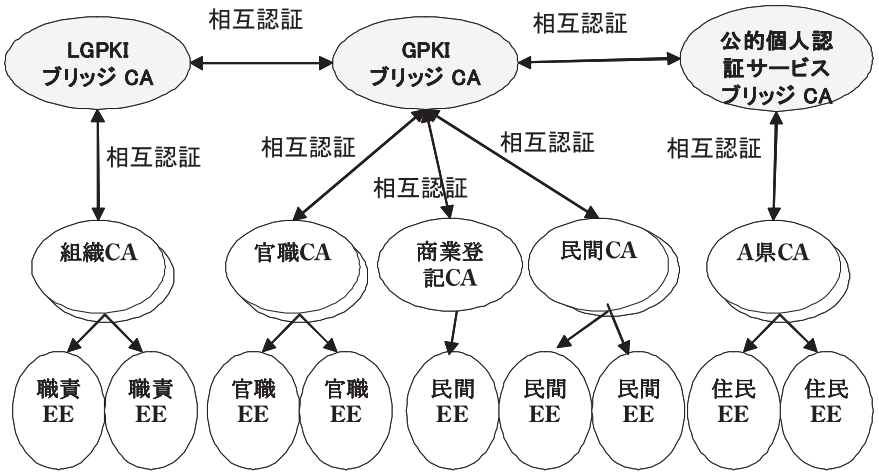


図 1　日本の公的認証基盤の構成
Fig. 1　Structure of Japanese public PKI.

同じく，LGPKI では，地方公共団体の都道府県，市区町村の職責者に証明書を発行している．公的個人認証サービスでは，都道府県単位に認証局が設置され，市区町村を窓口として，住民に証明書を発行している．これらに加え，法務省の運営する商業登記認証局，民間の運営する認証局がある．これらの認証局は，それぞれ GPKI ブリッジ認証局との間で相互接続を行っている．このように，国内において，すでに 50 を超す認証局が相互に接続され運用されている．

2.2　証明書の検証

認証局間の相互接続は，相互認証によって行われており，各認証局間で相互認証証明書を互いに発行している．相互接続が必要な個々の認証局間で相互認証を行う場合，非常に多くの相互認証証明書を発行，管理する必要があるため，GPKI など国内の公的な認証局は，ブリッジ認証局を採用し，相互認証のプロセスの軽減を行っている[4),5)]．

しかしながら，ブリッジ認証局方式を採用する場合の課題として，認証パスの長さが長くなることで，証明書の検証を行う際の処理が複雑になるという問題がある．証明書の検証方法は，ITU-T X.509[6)]，RFC5280[7)] に従っており，その詳細については GPKI における相互運用性仕様書[8)] に記載されている．

この証明書検証処理は，認証パスの構築処理と認証パスの検証処理から構成されている．

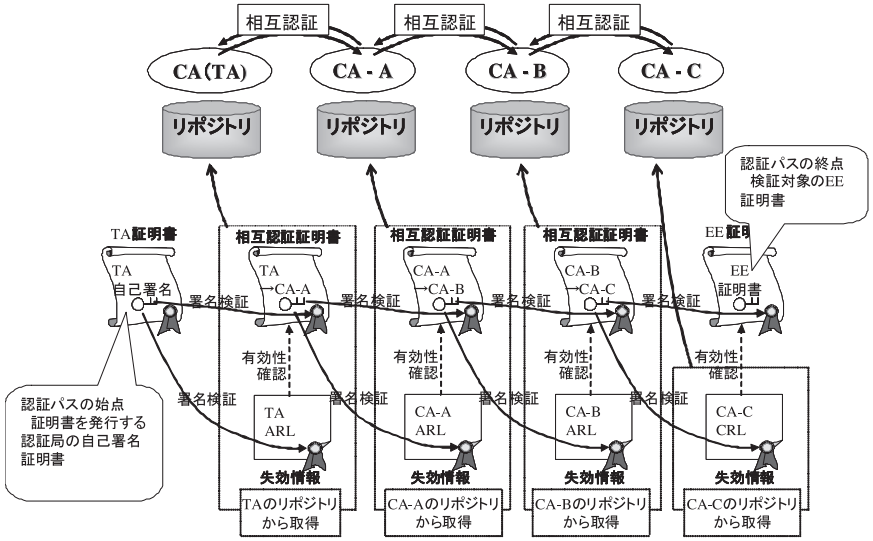


図 2　認証パスの構築と検証
Fig. 2　Certificate path building and verification.

まず，認証パスの構築処理において，利用者は，複数のリポジトリにアクセスし，相互認証証明書やリンク証明書などを取得し，証明書内の主体者名，発行者名，鍵識別子などをもとに，一連の証明書の並びからなる認証パスを構築する必要がある．

図 2 に示すように，たとえば，CA-C 認証局の発行する EE 証明書を，CA（TA）の利用ユーザが検証する場合には，このユーザの信頼点となる CA（TA）の自己署名証明書をはじめとする，一連の証明書からなる認証パスの構築と検証を行わなければならない．この場合は，5 枚の証明書からなる認証パスの構築が必要となる．

ここで，日本国内の公的な認証基盤においては，5 年ごとに認証局秘密鍵の鍵更新が行われている．認証局の鍵更新時には，リンク証明書が発行されることにより，鍵更新前後の連続性を維持するため，認証パス中にはリンク証明書も含まれることになる．このため実際に EE 証明書を検証する際には，認証パスを構成する証明書が 5 枚より増加する場合もあることになる．

また，次に行う認証パスの検証処理においても同様に，複数のリポジトリや OCSP レスポンダにアクセスを行い，証明書失効リスト（CRL，ARL）の取得や，OCSP（Online

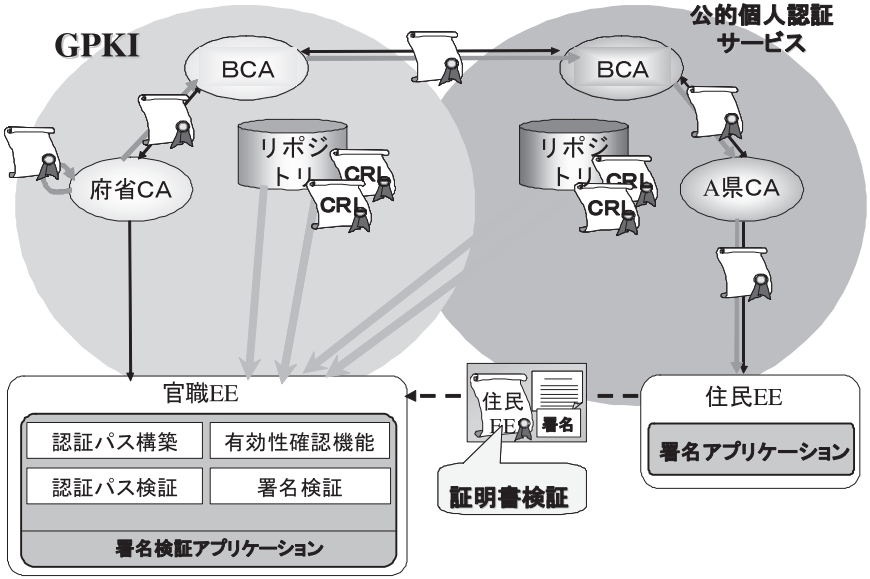


図 3　クライアントによる認証パスの構築と検証
Fig. 3　Certificate path building and verification by client.

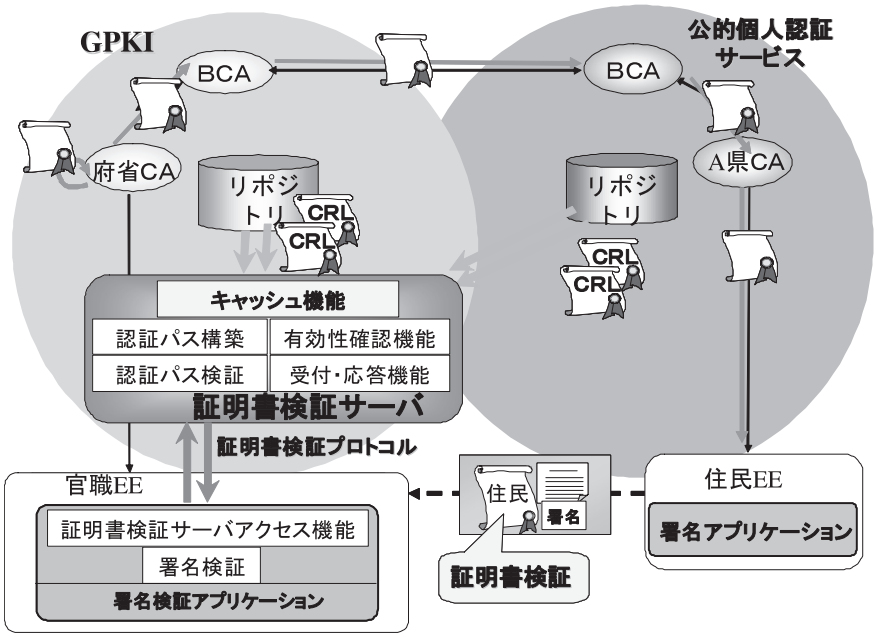


図 4　証明書検証サーバの機能
Fig. 4　Function of certificate validation server.

Certificate Status Protocol)[9] により証明書の失効を確認する必要がある．図 2 のケースにおいては，同じく認証局の鍵更新が行われていない場合においても，4 つの証明書失効リスト（CRL，ARL）の検証を行うことになる．

2.3　証明書検証サーバ

ここまでで述べたように，証明書の検証処理は大変複雑かつ，複数のリポジトリにアクセスし，認証情報を取得する必要がある．**図 3** に，公的個人認証サービスの証明書を保持する住民（住民 EE）が，GPKI の利用者である官職（官職 EE）に，電子署名文書を送付した場合の検証処理を示す．この場合，まず，官職 EE は，自分の CA である府省 CA を TA（Trust Anchor）とし，GPKI の BCA（Bride CA：ブリッジ認証局）と公的個人認証サービスの BCA ならびに A 県 CA の証明書を，GPKI もしくは，公的個人認証サービスのリポジトリを検索することにより取得し，住民 EE 証明書までの認証パスを構築する．次に，一連の証明書の記載事項の確認とともに，それら証明書が失効されていないかを確認するため，各々の証明書の失効リストを，証明書と同様に GPKI，公的個人認証サービスのリポジトリを検索することにより，取得し検証を行う必要がある．したがって，これら証明書の検証処理は，利用者である官職 EE 側のシステムにとっての負担が大きいという問題がある．橋本らは，認証情報を誰が取得すべきかに関する比較検討を行っており，利用者側で行う場合の運用負荷が大きいことを指摘している[10]．

そこで，著者らは，利用者の代理で証明書検証を行い，利用者に対して検証結果を応答する機能を持つ証明書検証サーバを提案している．証明書検証サーバのように利用者の負担を軽減する取り組みとして，たとえば，IETF の PKIX WG での要求仕様がまとめられており[11]，SCVP（Server-Based Certificate Validation Protocol)[12]，DVCS（Data Validation and Certification Server Protocols)[13] などの検討が行われている．しかしながら，SCVP や DVCS では証明書検証をサーバで行うためのプロトコルについて，標準化を行うことを目的としており，実際にサーバ側で検証を行う実装手段に関しての検討は行っていない．その

ため本論文では，サーバ側で証明書検証処理を実装する場合に必要となる，証明書検証処理の高速化の方法に関して提案を行っている．

2.3.1　証明書検証サーバによる検証

証明書検証サーバを用いて証明書の検証を行う方法を**図 4** に示す．証明書検証サーバでは，官職 EE から証明書検証要求を証明書検証プロトコルを用いて受け取り，同じく検証結果を応答する．この際，証明書検証プロトコルで規定される応答メッセージには証明書検証サーバの署名と証明書が貼付され，検証結果の信頼性を確保する．証明書検証サーバを用いる場合，従来利用者となる官職 EE 側のシステムで行っていた検証処理を，証明書検証サーバが行うため，認証パス構築・検証機能などは，証明書検証サーバに備えればよい．さらに，証明書検証サーバに本論文で提案するような認証情報のキャッシュ機能や高速なネットワークを持たせることで，証明書検証処理を高速かつ効率的に行うことができる．

また，利用者からは証明書検証が 1 回のトランザクションで行えること，また，証明書の失効数にともなってサイズの増大する CRL をダウンロードすることなどに比べ，証明書検証サーバと通信するデータ量が少ないという特徴もある．これらは，広帯域ネットワークの利用できない環境，たとえばモバイル PKI などでの利用時には大きなメリットになると考えられる[14),15)]．

3.　証明書検証処理の高速化

本論文では，相互認証する認証局の数に影響を受けにくく，かつ，さらに高速化する方式の提案とその検証結果について述べる．

3.1　証明書・CRL キャッシュ機能

一般に，処理速度を高速化するためには，1 度処理した処理内容を再利用することが行われている．証明書検証処理における再利用の範囲としては，これまで，証明書，CRL/ARL のリポジトリから取得した情報を，ローカルに保持したリポジトリキャッシュという形で再利用してきた．従来の証明書検証サーバは，証明書検証を高速化するために，**図 5** に示すように，複数のディレクトリサーバなどのリポジトリに格納されている証明書ならびに，CRL の情報をリポジトリ格納データの複製という形でキャッシュする機能を有していた．このリポジトリキャッシュ方式を用いることにより，証明書検証を高速化することができることを著者らは示してきた[16)]．

3.1.1　リポジトリキャッシュ方式の課題

リポジトリキャッシュ方式の概要を，**図 6** に示す．リポジトリキャッシュ方式では，ネットワーク越しにあるリポジトリの格納データをローカルに保持し，再利用することにより，検証処理の高速化を行っている．しかしながら，リポジトリとして主に利用される LDAP サーバから証明書や CRL を取得する場合，LDAP ディレクトリ内の認証局の DN の示すエントリから取得する必要があるが，LDAP では，個々の情報をファイル単位で取得できるのではなく，属性ごとのまとまった情報の取得となる．このため，相互認証証明書を取得する際に，このリポジトリキャッシュ方式には次の 2 つの問題があげられる．

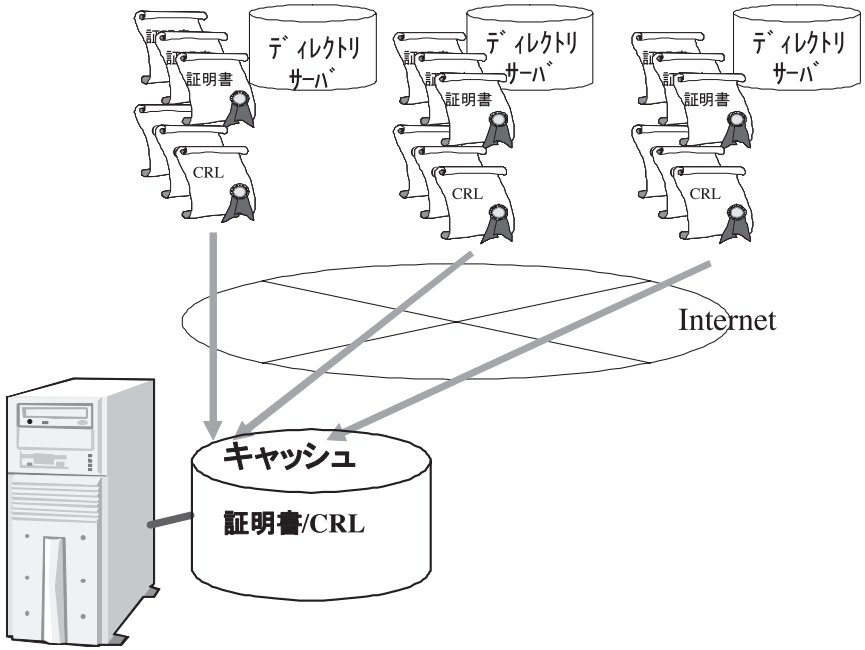


図 5　証明書・CRL キャッシュ機能
Fig. 5　Cache function of certificate and CRL.

1 つ目の問題は，多くの認証局と相互認証している認証局では，複数の相互認証証明書ペアが同じ crossCertificatePair 属性として格納されているため，LDAP で取得する際にすべてのペアを同時に取得することしかできない．そのため，相互認証証明書ペアとして利用する際には，個々のペアへの分割が必要になる．多くの認証局と相互認証を行っているブリッジ認証局に関する相互認証証明書ペアを取得する際には，この取得，分割処理が課題となる．

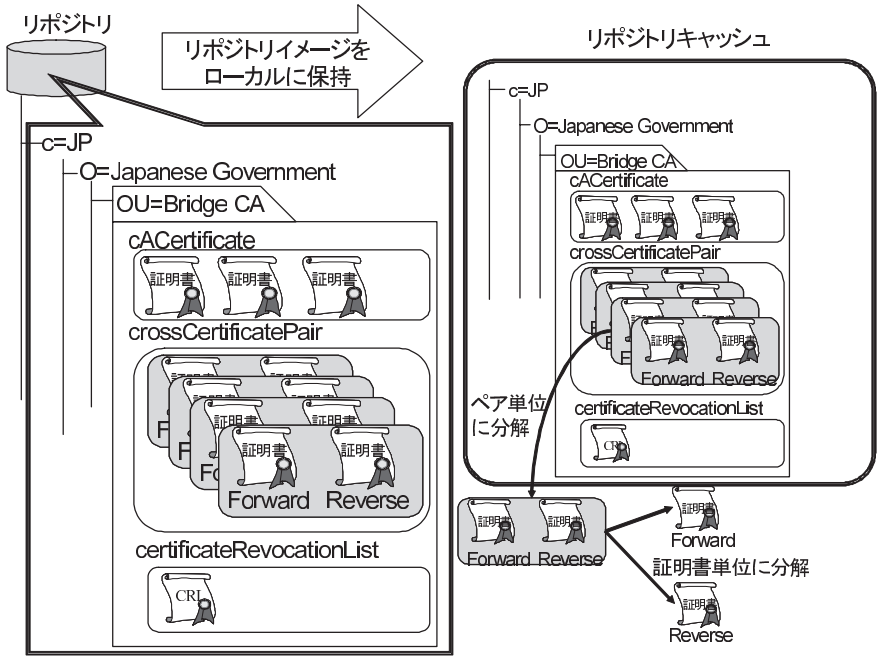


図 6　リポジトリキャッシュ方式の課題
Fig. 6　Problems of repository cache method.

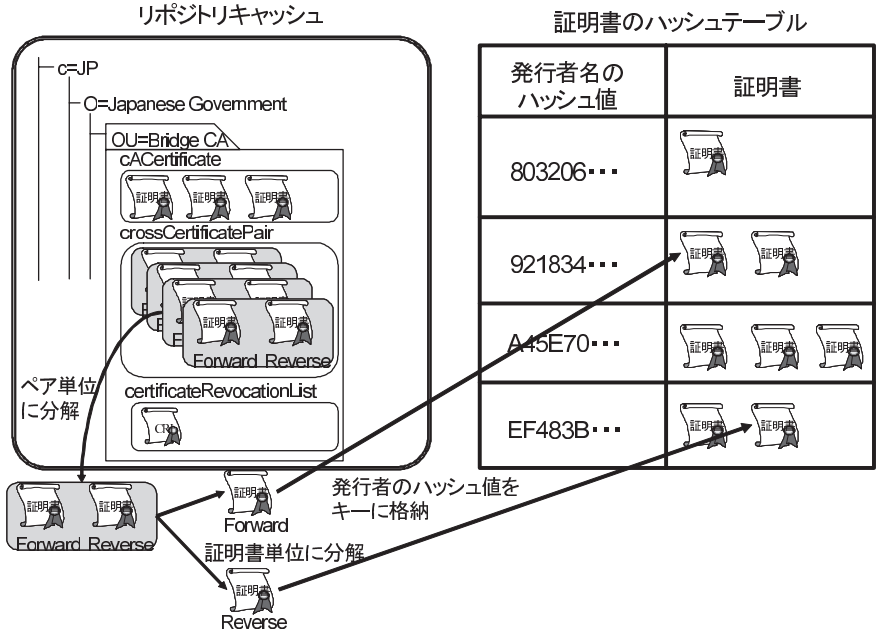


図 7　ハッシュテーブル格納方式
Fig. 7　Certificate store method with using hash table.

2 つ目の問題は，相互認証証明書は，crossCertificatePair 属性として格納されていることである．crossCertificatePair の構成は，発行者と主体者が相対する issuedToThisCA（Forward），issuedByThisCA（Reverse）と呼ばれる 2 対の証明書がペアとなり，連結された形でとなっている．そのため，必要な証明書を取得するためには，これを分解し，選択する必要がある．

つまり，単に LDAP サーバから得られた情報をそのままキャッシュしているリポジトリキャッシュ方式の場合，キャッシュとしての再利用時にも，保持しているキャッシュ情報の相互認証ペアへの分割と，相互認証証明書の抽出処理が必要になる．したがって，このリポジトリキャッシュ方式には，相互認証を行っている認証局の増加に応じて，これらの処理量が増えるため処理時間が増大するという課題がある．

3.1.2　証明書ファイルのハッシュテーブル格納方式

提案する証明書ファイルのハッシュテーブル格納方式を**図 7** に示す．ハッシュテーブル格納方式では，LDAP サーバから得られる相互認証証明書を，リポジトリキャッシュに格納後さらに，LDAP に格納されている状態そのままではなく，まず，相互認証証明書ペアへ分解した後に，さらにペアを分割し，相互認証証明書ごとに 1 つの公開鍵証明書としてファイル化する．

次に，これら公開鍵証明書をファイルとして管理するが，この際に，発行者名のハッシュ値をキーとしてハッシュテーブルに格納している．公開鍵証明書は，発行者名とシリアル番号で一意性が担保されることになっているため，ファイル化した証明書は，発行者名での検索が行えることが証明書検証処理を行ううえで望ましい．しかしながら，発行者名をそのままファイル名にした場合，発行者名にはファイル名として利用できない文字を含む可能性が

あること，ならびに，発行者名は可変長のデータであり，ファイル名の制限長を超えてしまう可能性があるなどの問題がある．そのため，分割した証明書は，証明書の発行者名のハッシュ値をキーとするハッシュテーブルを構築し，証明書ファイルを格納することとした．

ここで，相互認証証明書ペアは，2 つの認証局の DN の示す LDAP サーバ上のエントリに，issuedToThisCA（Forward），issuedByThisCA（Reverse）が入れ替わる形に存在するため，この提案する証明書ファイルのハッシュテーブル格納方式では，同じ相互認証証明書を 2 カ所から取得することとなる．一般に，相互認証を行う 2 つの認証局の LDAP サーバは，各々の認証局でそれぞれ運用され，発行した相互認証証明書を登録するタイミングが異なることが想定される．本手法では，両者から取得するため，異なっていたものが格納されていた場合に，新規の証明書を採用することにより，最新の情報を利用できるという利点もある．

3.2　認証パス情報キャッシュ機能

証明書・CRL キャッシュ機能による高速化に加えて，証明書検証サーバをさらに高速化するためには，再利用する範囲を拡大する方法が考えられる．

証明書検証処理は，認証パスの構築処理と検証処理に分けることができる．このうち，認証パス構築処理においては，ネットワークルーティングで行われているような，経路情報の交換を行う機能がなく，また，相互認証証明書内に，主体者名，発行者名が記載されているだけであるため，直接相互認証を行っていない認証局との間の認証パスを構築するには，グラフ理論における経路問題を解くことと同様の処理を行う必要がある．このため，相互認証を行う認証局の数に応じて，処理時間が増大してしまうという課題があり，この課題の解決が必要である．したがって，認証書検証処理の高速化に向けて，認証パス構築処理における処理結果の再利用について，検討を行った．また，証明書検証処理には，証明書検証時の初期入力値によって，有効な認証パスが切り替わってしまうことがあるなどの要件があり，最短経路だけでなく，構築できる複数の認証パスを経路としてキャッシュすることが可能な方式を検討した．

本論文では，構築した認証パスのうち，パスを構成する証明書が CRL などに失効されるなどして，無効なことが明確な認証パス以外の認証パスを認証パス情報キャッシュとして保持しておき，このキャッシュ情報をもとに，パス構築処理を高速化する手法の提案を行う．ここで，キャッシュサイズとそのヒット率に鑑みて，パス検証処理まで行った後，検証に成功した認証パスのみをキャッシュの対象とした．すでに示したように，構築を行う認証パスは信頼点となる認証局の自己署名証明書から検証対象の証明書までの一連の証明書群から

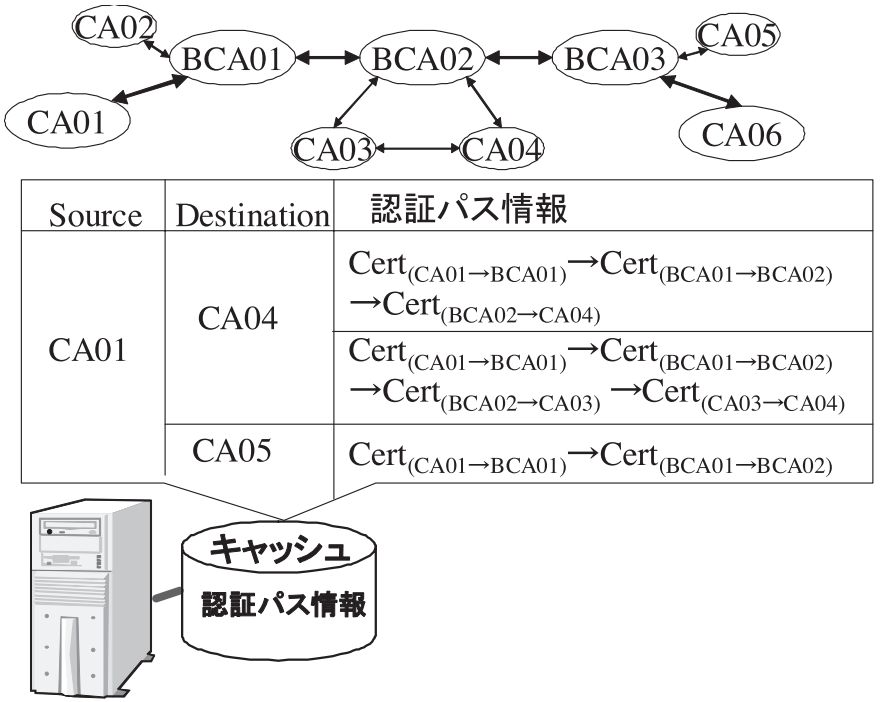


図 8　認証パス情報キャッシュ機能
Fig. 8　Cache function of certificate path.

構成されている．このうち，検証対象の証明書の発行者の証明書は，検証対象証明書に記載されている発行者名や鍵識別子の情報から，容易に検索が可能なため，再利用の範囲としては，信頼点となる証明書から検証対象証明書までの範囲と，信頼点となる証明書から検証対象証明書の発行者の証明書までの範囲が考えられる．ここで，一般に発行者は複数の証明書を発行しており，それら証明書の検証時には，同じ認証パス情報を用いることが可能であるため，検証対象証明書まで含む場合に比べ，その発行者までの場合のほうが，再利用性に優れていると考えられる．したがって，後者の範囲を認証パス情報のキャッシュとして再利用することとした．

認証パス情報キャッシュでは，**図 8** に示すように，認証パスの開始点である認証局の自己署名証明書を Source とし，検証対象の証明書の発行者を Destination とするときの，そ

の経路となる認証局の証明書の情報をパス情報として保持している．図 8 では，CA01 を信頼点とする CA04 の発行する証明書の検証を行う際の，認証パス情報を示している．この例での認証パスは，BCA02 から CA04 へ直接つながる認証パスと，間に CA03 が介在する認証パスの 2 種類存在することになり，そのどちらもがパス検証の対象となる．また，認証局が鍵更新を行っていた場合などは，新しい鍵に対する証明書と古い鍵に対する証明書を結びつけるリンク証明書が必要になり，同じ認証パスを構成する場合においても必要となる証明書群が異なるケースが存在する．

認証パス情報キャッシュを用いたパス構築処理においては，まず，信頼点の情報と，検証対象証明書内の発行者の情報をもとに，キャッシュから，検証に必要な認証パス情報を取得する．次に，得られた認証パス情報をもとに，証明書ファイルキャッシュから証明書ファイルを取得し，一連の証明書群を得る．提案する手法では，この得られた証明書群が認証パスとなる．この後，パス検証処理に構築した認証パスを渡すことにより，証明書の検証を行う．本手法では，負荷の大きいパス構築処理を認証パス情報キャッシュの検索処理に置き換えることで，認証パスの構築処理の高速化を行っている．

また，複数の認証パスが構築可能な場合においては，構築したパスのうち，認証パスを構成する証明書が CRL などにより失効され，その認証パスが無効なことが明らかなケース以外に，証明書検証時の初期入力値によって，ある初期入力値の場合は有効であるが，別の初期入力値では無効と判定される認証パスが存在することが考えられる．認証パス情報キャッシュでは，CRL などにより認証パスが無効であることが明確な場合には，検証処理の失敗を契機として，該当する認証パスをキャッシュから除外しているが，それ以外の原因による検証失敗の場合は，キャッシュとして保持し続けるようにしている．

4. 性 能 実 験

4.1 キャッシュ機能測定実験の構成

提案する証明書ファイルのハッシュテーブル格納方式と認証パス情報キャッシュ機能の有効性を検証するために，証明書検証サーバの応答時間ではなく，その内部で行っている認証パス構築処理の処理時間を，すでにパス構築に必要な証明書が，リポジトリキャッシュとして登録されている状態で測定した．また，処理結果の分析を行うため，**図 9** に示すような 3 種のパターンの構成で実験を行った．キャッシュ機能測定実験は，すべての機器（証明書検証サーバ，クライアント，リポジトリなど）を同一 LAN 上に設置して行っている．

パターン A は，ブリッジ認証局を介在した基本形であり，パターン B では，相互認証を

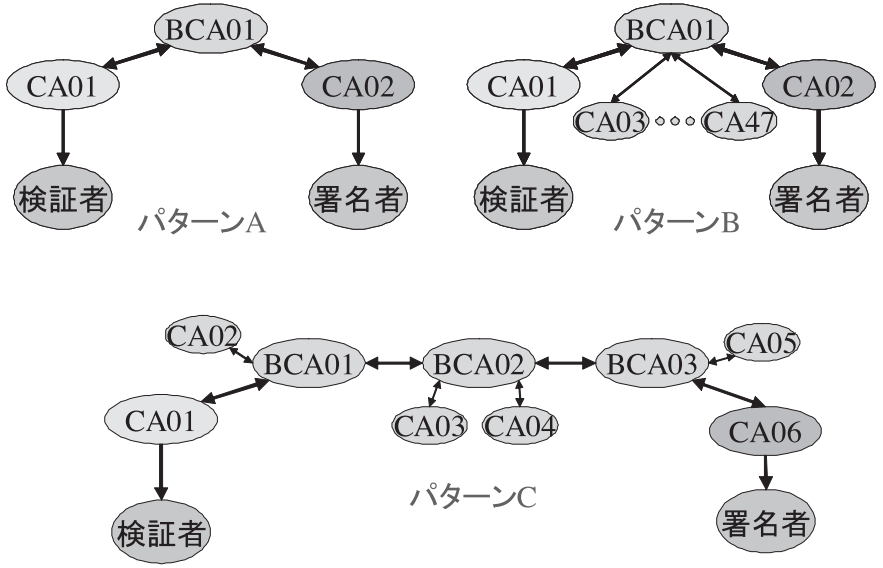


図 9　キャッシュ機能測定実験環境の構成
Fig. 9　Model of cache function experiment environment.

行っている認証局の数を JPKI を想定し，その影響を検証する．また，パターン C では，認証パスが長くなった場合の影響の検証を行うことを目的としている．

4.2 検証性能測定実験の構成

証明書検証サーバの有効性を検証するために，性能面における検討が重要である．証明書検証サーバの性能を測定するにあたり，どのような要因が検証時間に影響を与えるかを確かめるために，証明書検証における条件を変化させて実験を行った．実験の基本構成を**図 10** に示す．

キャッシュ機能測定実験と同様に，すべての機器（証明書検証サーバ，クライアント，リポジトリなど）を同一 LAN 上に設置して行った．性能測定においては，ハッシュテーブル格納方式を用いた証明書・CRL キャッシュ機能と認証パス情報キャッシュ機能をそれぞれ，有効化/無効化させながら，サーバからの応答時間，ならびに，クライアントから多重にアクセスを行った際の処理件数を測定した．また，キャッシュを有効化しているケースでは，すでに有効なキャッシュが登録されている状態で測定を行った．実験に用いた機器の諸元を

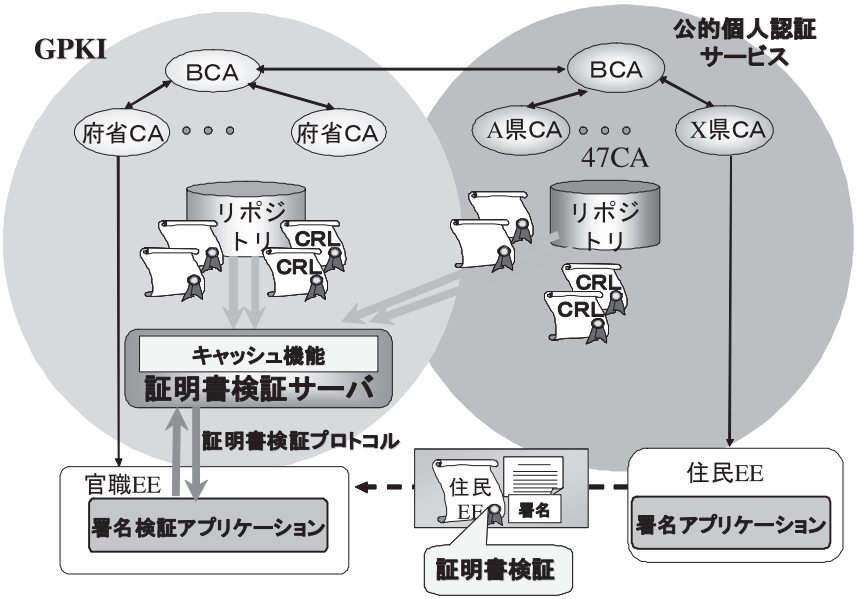


図 10　実験環境の構成
Fig. 10　Model of experiment environment.

表 1 に示す.

4.3　実環境での性能評価実験

また，実際に稼動している環境での，性能測定実験を行った．本物の公的個人認証サービスは，認められた署名検証者以外には，CRL を提供していないため，パス検証の向きを検証性能測定実験環境とは逆にし，公的個人認証サービスのある都道府県認証局をトラストアンカとし，GPKI の官職証明書を検証対象証明書として，性能評価を行った．この場合，認証パスは，TA となる都道府県 CA ⇒ 公的個人認証サービス BCA ⇒ GPKI BCA ⇒ 官職認証局 ⇒ 官職証明書となりパス長は 4 である．

性能測定時には，性能測定用の CVS を光ファイバ回線を通じてインターネットに接続することにより，実環境の LDAP サーバにアクセスし，各キャッシュ機能の有効/無効を切り替えながら検証速度の測定を実施した．また，検証クライアントは CVS と同一のネットワークから直接アクセスした場合と別の回線によりインターネットに接続し，インターネット経由でアクセスした場合に分けて検証要求を行った．本実験でも，同じくキャッシュを有効化しているケースでは，すでに有効なキャッシュが登録されている状態で測定を行った．

表 1　実験機器諸元
Table 1　Experiment equipiment spec.

項番	機器	諸元
1	証明書検証サーバ	OS RedHat Enterprise Linux 4 CPU Xeon 3 GHz Memory 2 GByte
2	クライアント	OS Windows®XP CPU Pentium®M 1.2 GHz Memory 1 GByte
3	CA，LDAP サーバ	OS Redhad Enterprise Linux 4 CPU Xeon 3 GHz Memory 2 GByte
4	LAN	100BaseTX switching HUB

5.　結果と検討

5.1　キャッシュ機能測定結果

4.1 節で示した構成で行ったキャッシュ機能測定実験の測定結果を**表 2** に示す．認証パス構築処理を 100 回繰り返したときの平均処理時間と標準偏差を示している．

5.2　検証時間測定結果

4.2 節で示した構成で行った性能測定実験の測定結果を**表 3** に示す．結果は，クライアントからの証明書検証要求を 100 回行い，そのレスポンスタイムの平均値と標準偏差ならびに処理件数を示している．

5.3　実環境での性能評価結果

4.3 節で示した実環境で行った性能評価実験の結果を**表 4** に示す．結果は，クライアントからの証明書検証要求を 100 回行い，同じくそのレスポンスタイムの平均値と標準偏差を示している．

5.4　検　　討

5.4.1　キャッシュ機能に関して

性能測定実験結果ならびに，キャッシュ機能測定実験結果より，ハッシュテーブル格納方式での証明書・CRL キャッシュ機能により，認証パス構築処理の高速化が行われてることが確認できる．キャッシュ機能測定実験により，多くの相互認証証明書を取得し処理する必

表 2　キャッシュ機能測定実験結果
Table 2　Result of cache function experiment.

キャッシュ機能の種類	パターン A		パターン B		パターン C	
	平均 (msec)	標準偏差	平均 (msec)	標準偏差	平均 (msec)	標準偏差
リポジトリキャッシュ（従来方式のキャッシュ）	23.2	1.8	60.1	1.2	25.3	2.5
ハッシュテーブル格納方式	4.5	0.2	4.0	0.8	4.3	0.8
リポジトリキャッシュ＋認証パス情報キャッシュ	22.3	0.7	56.7	1.9	25.0	2.1
ハッシュテーブル格納方式＋認証パス情報キャッシュ	3.6	0.2	3.9	0.7	4.0	0.6

表 3　性能測定実験結果
Table 3　Experiment result.

キャッシュ機能の有無		レスポンスタイム		処理件数
認証パス情報キャッシュ	ハッシュテーブル格納方式	平均 (msec)	標準偏差	(件/秒)
-	-	360	122.2	3.5
-	○	222	108.6	8.2
○	-	94	10.7	17.3
○	○	76	11.49	32.0

要があるパターン B において，パス構築時間が約 1/15 に短縮されているという結果が得られ，その効果が顕著であることから，リポジトリから取得する証明書郡を分離して格納している本提案の証明書ファイルのハッシュテーブル格納方式による高速化への貢献が大きいことがいえる．

実験環境での性能測定結果では，LDAP サーバと同一ネットワーク上にあるため相対的に，ハッシュテーブル格納方式による証明書・CRL キャッシュ機能の効果が約 60%程度のレスポンスタイムの短縮にとどまり，低く現れてしまっている．しかしながら，実際の環境を用いたテストでは，インターネット越しに証明書，CRL を取得するため，ハッシュテーブル格納方式による証明書・CRL キャッシュ機能の効果が，レスポンスタイムが平均して約 1/7〜約 1/8 に短縮され，また，標準偏差の値が小さく安定しているなど，その効果が顕著に現れている．

また，認証パス情報キャッシュ機能の効果に関しては，両方のキャッシュ機能を有効にした際に，レスポンスタイムが約 1/30〜約 1/100 に短縮されるなど，ハッシュテーブル格納方式での証明書・CRL キャッシュ機能と併用したときの効果が大きいことが分かる．

表 4　実環境での性能測定結果
Table 4　Experiment result in real environment.

キャッシュ機能の有無		クライアントが同一ネットワークの場合のレスポンスタイム		クライアントが Internet 経由でアクセスする場合のレスポンスタイム	
認証パス情報キャッシュ	ハッシュテーブル格納方式	平均 (msec)	標準偏差	平均 (msec)	標準偏差
-	-	5,066	1,259.3	5,814	1,444.3
-	○	623	72.8	794	78.2
○	-	3,585	1,410.8	2,966	309.7
○	○	54	1.8	210	45.0

5.4.2　相互認証の数による影響に関して

著者らは，従来のリポジトリキャッシュによる高速化では，相互認証の数の増加が性能に影響を及ぼすことを示していた．今回のキャッシュ機能測定実験におけるリポジトリキャッシュを利用した際にも，パターン A と B での比較において，相互認証の数が 2 から 49 に増加していることから，パス構築処理時間が 23.2 msec から 60.1 msec に増加している．本論文で提案する証明書ファイルのハッシュテーブル格納方式での証明書 CRL キャッシュ機能ならびに，認証パス情報キャッシュ機能の双方を有効にして行った実験結果では，すべての構成において，認証パス構築処理時間が 4 msec 以下の処理時間に収束した．これは，本論文で提案している手法により本来のパス構築処理が，キャッシュ検索処理に置き換えられたためと考えられ，相互認証の数による検証性能への影響を減少させることができていることが分かる．ただし，認証パスの構築処理に比べ，検証処理にかかる時間の比率が高まっており，今後，認証パスが長くなるのに合わせて，パス検証処理時間が長くなることへの対策をしていく必要があると考えられる．

5.4.3　証明書検証サーバ性能評価

検証性能測定実験の結果より，環境によっては，サーバ 1 台あたり，毎秒 30 件以上の検証処理が実行可能なことが示されている．ここで，2008 年の統計で国，地方に対する全申請・届出件数である年間約 11.5 億件のうち，電子申請が行われているのは約 22 パーセント（約 2.5 億件）であることから[17]，2.5 億件の申請・届出のすべてが，電子署名されていたことを想定したとしても，24 時間，365 日で平均すると，秒間約 8 件である．したがって，ピーク時での 10 倍の件数を想定したとしても，数台のサーバで処理できる性能であり，現

時点での性能としては，十分なものであると考えられる．

しかしながら，近年の電子申請/申告件数の著しい増加傾向に鑑みた場合，多重アクセス数の増大による性能劣化も懸念される．それに対し，証明書検証サーバは，検証要求・応答のプロトコルを，HTTP 上に実装しているため，複数台のサーバを並列に設置し，市販の負荷分散装置などで負荷分散することにより，容易にシステム拡張を行うことも可能である．しかし，システムコストの低減などを考慮した場合，将来的には，より検証処理を高速化する手法に関して，検討する必要性が発生してくることも考えられる．

6. 証明書検証サーバのセキュリティの考察

証明書検証サーバを利用する際のセキュリティ上の問題について考察する．証明書検証サーバの応答の正当性の担保，証明書検証サーバの可用性，キャッシュの更新方式の 3 点について，考察を行う．

6.1 証明書検証サーバの応答の正当性

証明書検証サーバ方式における脆弱性としては，まず，検証サーバの応答の改ざん，もしくは，検証サーバへのなりすましが考えられる．それに対し，証明書検証サーバでは，証明書検証用のプロトコルとして OCSP の拡張や，SCVP を利用している．OCSP や SCVP において，その応答にはサーバ側の署名がなされるるため，利用者側のクライアントにおいては，その署名を検証することにより，サーバからの正当な応答であることを確認することにより，この問題に対処している．

6.2 証明書検証サーバの可用性

また，証明書検証サーバに対して，多くの不正なリクエストが行われることにより，サービス不能状態に陥れられることが考えられる．これに対しては，OCSP や SCVP においては，オプショナルな項目であるが，証明書検証サーバでは NONCE を必須とし，利用者側のクライアントから送られる証明書検証要求電文中にランダムに生成される NONCE の値を入れることとしている．通常では，NONCE はリプレイ攻撃に対する対策であるが，証明書検証サーバ側では，この NONCE の値を監視し，短期間に同じ NONCE の値を持つ検証要求が送られてきた場合，その要求を拒絶することで，サービス不能攻撃にも対処している．また，リクエストデータに署名を行うことも可能であり，署名を必須にすることにより，署名付きリクエスト以外を拒絶すること，ならびに，署名により要求元を明確化することにより，サービス不能攻撃を抑止することも可能である．さらに，OCSP や SCVP を HTTP 上に実装しているため，サーバを複数台設置し，市販の負荷分散装置などを利用することで，リクエスト数の増加にともなう，サーバ増強や，障害時の閉塞運転が容易に行えるなど，可用性を高めることが可能である．

6.3 キャッシュ更新方式

本来，PKI においては，認証局の運用により緊急的に，新しい証明書や，CRL が発行されることがあるため，証明書検証時には可能な限り最新の情報を入手し検証を行う必要がある．つまり，証明書検証時に，これら情報が公開されているリポジトリにアクセスして，つねに最新の公開情報を用いて証明書の検証を行うことが，セキュリティ上は求められる．しかしながら，証明書検証サーバにおける証明書検証処理の高速化には，本論文で示すとおり，証明書・CRL キャッシュ機能の効果によるところが大きい．キャッシュ情報の利用は，最新の情報の利用と相反する面もあり，証明書検証サーバとしては，キャッシュ内の情報をいかに最新の情報として維持管理するのかが重要な課題となる．

このため，証明書検証サーバでは，クライアントからの検証要求処理とは非同期に，バックグラウンド処理として，定期的に自身のキャッシュしている証明書，CRL の更新をチェックする処理を行っている．そして，証明書，CRL に変更があった場合には，キャッシュ内容を更新することとしている．このことにより，キャッシュによる検証処理の高速化の恩恵を受けつつも，最新の情報による証明書検証結果を得ることができるようにしている．また，証明書，CRL の有効期限のチェックも同時に行っており，キャッシュ内に有効期限切れなど不必要な証明書，CRL が残り続けるのを防止するようにしている．

7. おわりに

今後，公開鍵認証基盤（PKI）の広がりに従って，証明書の検証時の処理がますます複雑になる．このことが PKI 利用普及の妨げにつながるという問題が発生するということに，いちはやく着目して，証明書検証処理の高速化手法に関する研究を行い，証明書検証サーバの開発を行った．また，性能実験の結果により，証明書検証サーバの有効性の確認が行えた．しかしながら，今後は，SHA-1 から SHA-2 や RSA から楕円曲線暗号など複数の暗号アルゴリズムの利用などが想定され，相互認証関係の複雑化により，証明書検証処理速度の劣化が想定されるため，複数の暗号アルゴリズムが混在した場合でも，高速化可能な証明書検証方式の検討を行っていきたい．

参 考 文 献

1) 総務省：政府認証基盤（GPKI）について (2001). http://www.gpki.go.jp/documents/gpki.html
2) 総合行政ネットワーク運営協議会：地方公共団体における組織認証基盤（LGPKI）(2004). http://www.lgpki.jp/
3) 公的個人認証サービス都道府県協議会：公的個人認証サービス ポータルサイト (2004). http://www.jpki.go.jp/
4) カーライル・アダムズ，スティーブ・ロイド：PKI 公開鍵インフラストラクチャの概念，標準，展開，（株）ピアソン・エデュケーション (2000).
5) 小松文子：PKI ハンドブック，（株）ソフト・リサーチ・センター (2000).
6) International Telecommunication Union: Information technology – Open systems interconnection – The Directory: Public-key and attribute certificate frameworks, ITU-T Recommendation X.509 (2008).
7) Cooper, D., Santesson, S., Farrell, S., Boeten, S., Housley, R. and Polk, W.: Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, RFC5280 (2008).
8) 総務省：政府認証基盤（GPKI）政府認証基盤相互運用性仕様書 (2001). http://www.gpki.go.jp/session/CompatibilitySpecifications.pdf
9) Myers, M., Ankney, R., Malpani, A., Galperin, S. and Adams, C.: Online Certificate Status Protocol – OCSP, RFC2560 (1999).
10) 橋本正一，政本廣志：署名検証サーバにおける経路情報管理，情報処理学会 コンピュータセキュリティシンポジウム 2001 論文集，pp.149–154 (2001).
11) Pinkas, D. and Housley, R.: Delegated Path Validation and Delegated Path Discovery Protocol Requirements, RFC3379 (2002).
12) Freeman, T., Housley, R., Malpani, A., Cooper, D. and Polk, T.: Server-Based Certificate Validation Protocol (SCVP), RFC5055 (2007).
13) Adams, C., Sylvester, P., Solotarev, M. and Zuccherato, R.: Internet X.509 Public Key Infrastructure Data Validation and Certification Server Protocols, RFC3029 (2001).
14) 梅澤克之，笈川光浩，洲崎誠一，手塚　悟，平澤茂一：モバイル向け証明書検証システムの開発，情報処理学会論文誌，Vol.48, No.2, pp.625–634 (2007).
15) 梅澤克之，笈川光浩，洲崎誠一，手塚　悟，平澤茂一：モバイル環境での証明書検証方式の評価，電子情報通信学会論文誌（D），Vol.J90-D, No.2, pp.384–398 (2007).
16) 藤城孝宏，鍛　忠司，羽根慎吾，熊谷洋子，手塚　悟：証明書検証サービスの開発，電子情報通信学会論文誌 D-I，Vol.J87-D-I, No.8, pp.833–840 (2004).
17) 電子政府評価委員会：電子政府評価委員会 平成 20 年度報告書 (2009). http://www.kantei.go.jp/jp/singi/it2/ithyouka/houkoku/2008/den_huzoku2.pdf

付　　録

A.1　商標，登録商標

Windows®の正式名称は，Microsoft®Windows® operating system です．Microsoft®，Windows®は，米国 Microsoft Corporation の米国およびその他の国における登録商標です．

Pentium®は，米国およびその他の国における Intel Corp.（or Intel Corporation）の商標または登録商標です．

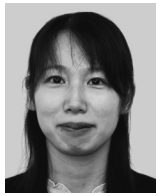
RedHat は，米国およびその他の国における RedHad, Inc. の商標です．

Linux は，米国およびその他の国における Linux Torwalds の商標です．

(平成 22 年 5 月 28 日受付)
(平成 22 年 10 月 4 日採録)

藤城　孝宏

1993 年慶應義塾大学大学院理工学研究科電気工学専攻修士課程修了．同年（株）日立製作所入社，システム開発研究所第 7 部に勤務．2009 年より慶應義塾大学理工学研究科開放環境科学専攻後期博士課程在籍．ネットワーク管理システム等の研究開発を経て，現在，セキュリティシステム，特に電子認証の研究開発に従事．電子情報通信学会員．



佐藤　　茜

2004 年東京理科大学理学部応用物理学科卒業，2006 年東京工業大学大学院総合理工学研究科物理情報システム創造専攻修士課程修了．同年（株）日立製作所入社，システム開発研究所第 7 部に勤務．セキュリティシステム，特に電子認証の研究開発に従事．

熊谷　洋子

2000 年東京農工大学大学院生物システム応用科学研究科修士課程修了．同年（株）日立製作所入所，システム開発研究所第 7 部に勤務．入所以来，セキュリティシステム，特に電子認証の研究開発に従事．

鍛　忠司

1996 年大阪大学大学院基礎工学研究科情報工学分野博士前期課程修了．同年（株）日立製作所勤務．企業情報システム，分散オブジェクトシステム，PKI 等のセキュリティに関する研究開発に従事．IEEE 会員．

羽根　慎吾

1998 年慶應義塾大学理工学部物理学科卒業，2001 年東京大学大学院理学系研究科物理学専攻博士課程修了．同年（株）日立製作所入社，システム開発研究所第 7 部に勤務．セキュリティシステム，特に電子認証の研究開発に従事．博士（理学）．日本物理学会員．

手塚　　悟（正会員）

東京工科大学コンピュータサイエンス学部教授，工学博士．1984 年慶応義塾大学工学部数理工学科卒業．同年（株）日立製作所入社．PC の OS，デバイスドライバ，LAN システムの構築・運用管理の研究開発を経て，セキュリティシステムの研究開発に従事．2009 年 4 月より現職．情報処理学会論文賞（2004 年，2008 年），IEEE-IIHMSP2006 Best Paper Award．著書に『Inside CORBA』（共訳，アスキー出版，1998 年），『インターネットコマース　新動向と技術』（共著，共立出版，2000 年），『インターネット時代の情報セキュリティー　暗号と電子透かし』（共立出版）．

岡田　謙一（フェロー）

慶應義塾大学理工学部情報工学科教授，工学博士．専門は，CSCW，グループウェア，ヒューマン・コンピュータ・インタラクション．『ヒューマンコンピュータインタラクション』（オーム社），『コラボレーションとコミュニケーション』（共立出版）をはじめ著書多数．情報処理学会誌編集主査，論文誌編集主査，GW 研究会主査等を歴任．現在，日本 VR 学会理事，情報処理学会 IE 領域委員長．情報処理学会論文賞（1996 年，2001 年，2008 年），情報処理学会 40 周年記念論文賞，日本 VR 学会サイバースペース研究賞，IEEE SAINT'04，ICAT'07 最優秀論文賞を受賞．情報処理学会フェロー，IEEE，ACM，電子情報通信学会，人工知能学会各会員．