社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2005－DPS－122 (10)
2005－CSEC－ 28 (10)
2005／3／22

X.509 証明書の高速認証パス検証アルゴリズム

○羽根慎吾 藤城孝宏 橋本洋子 手塚悟
株式会社日立製作所

X.509 証明書の認証パス検証を高速で行うためのアルゴリズムの研究開発を行った。CA 証明書や失効情報だけでなく、認証パス情報もキャッシュすることで高速化した。高速認証パス検証アルゴリズムを搭載したサーバ機の評価をプライベートなテスト環境において行い、高速化の効果を確認した。

Fast Algorism for X.509 Certificate Path Validation

Shingo Hane, Takahiro Fujishiro, Yoko Hashimoto, Satoru Tezuka
Hitachi, Ltd.

An algorism to execute fast path validation of the X.509 Certificate was developed. Not only certificates and certificate revocation information but also certification path information was used to be faster. The effect of it was confirmed by the experiment of the server system that had the fast algorism of certification path verification on the private test environment.

1. はじめに

ITU-T の X.509[1]に代表される PKI(Public Key Infrastructure)応用の一例として、セキュア通信のためのサーバ認証や暗号鍵の交換などに用いられている[2][3]。その他にも、電子署名法[4]の整備のおかげで印鑑の電子版として利用できるようになり、ユーザ認証や契約行為にも用いられるようになってきている。また、PKI にはその他さまざまな応用が考えられ、IEEE802.1X のネットワーク認証や、機器の認証などのユースケースも挙げられる。このように PKI の応用範囲の広まりとともに利用機会が増大しているため、より負荷が小さく高速な PKI の処理が求められている。

PKI を利用する上で発生する処理には、署名や署名検証のように暗号演算が主な処理となるものがある。この他にも、証明書検証のように失効情報を取得するためにネットワークへアクセスする処理なども必要となるものがある。前者に対する高速化としては主に暗号計算のアルゴリズムの改良などが考えられ、後者に対してはキャッシュや検索アルゴリズムなどの改良も有効になる。今回、ネットワークキャッシュや中間処理状態を保管・再利用することで証明書検証の高速化が行えたので報告する。

2. 証明書の検証

PKI を用いて、認証を行う場合、署名の検証と証明書の検証を行う必要がある。署名の検

証では、証明書に記載された公開鍵で署名されたかどうかを確認する。証明書の検証では、証明書が信頼できる認証局から正しく発行されたものか確認する。さらに、正しく発行された証明書でも、証明内容の変更や、秘密鍵の危殆化などの理由で証明書が失効される場合があり、証明書の有効性の確認を行う。もしも、署名検証を行うだけでは、不正な証明書や失効された証明書を検出できず、PKI により担保する真正性や本人性が確保できない。

以上のことから、PKI を利用する上で証明書の検証を署名検証とともに行うことが重要であると理解できる。この証明書の検証方式には、大きく分けてマルチトラストの考え方と、相互認証による考え方がある。以下より、マルチトラスト方式と相互認証方式について記述した後、相互認証方式における証明書の検証方法とそれをサーバで行う方式について述べる。

2.1. マルチトラスト方式

マルチトラスト方式では証明書の検証を行う際、検証に使用する CA 証明書を複数所持する。そして、検証対象の証明書に合わせて検証用の CA 証明書を切り替える方式である。検証用の CA 証明書を複数所持することからマルチトラストと呼ばれている。一般的な Web ブラウザが SSL サーバ証明書の検証に用いている方式である。

2.2. 相互認証方式

相互認証による方式では CA 証明書を 1 つまたはマルチトラスト方式と比較して少数だけ保持しておき、証明書検証に用いる。このとき、相互認証と呼ばれる CA 間での連携を利用して検証対象の証明書の検証を行う。日本の政府認証基盤（GPKI）[5]や、公的個人認証サービス [6]などの公的認証基盤で用いられている方式である。

2.3. 証明書検証方法

証明書で作られた認証のチェーンを認証パスと呼び、相互認証方式においては認証のチェーンの検証を行うことで証明書の検証が行われる。この認証パスを構築して証明書の検証を行う例を図 1 に示す。証明書検証機能は、TA から EE 証明書までの認証パスを構築し、検証を行っている。この例では、検証のために扱われる認証情報は相互認証証明書 3 枚と、CRL/ARL4 枚を含めて全部で 9 つある。これらの証明書と CRL/ARL はそれぞれリポジトリと呼ばれるディレクトリサーバを検索して収集する。

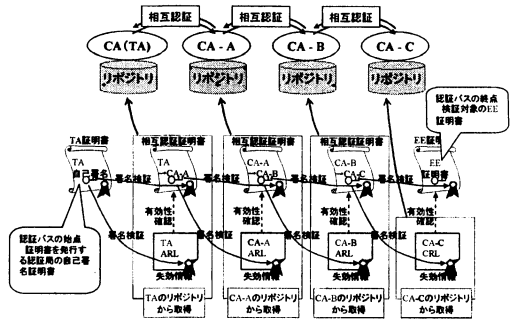

図 1　認証パスの検証の例

2.4. サーバ方式

上記、相互認証方式における証明書の検証は処理が重く、クライアントが証明書を受け取るたびにすべての検証作業を行うことは負担になる。そのため、証明書の検証処理をサーバで行う方法が有効であると考えられ、プロトコルや検証方法 [7], [8]の検討が行われている。このような証明書を検証するためのプロトコルには、日本政府における OCSP[9]の拡張を使ったプロトコル [5]や、IETF で検討されている SCVP[10]などがある。

本報告において、証明書の検証処理を行うサーバを証明書検証サーバと呼び、研究の対象とした。サーバで処理する場合、クライアントから繰り返し同じ証明書の検証要求があるため、

この検証処理で必要な証明書類を保存・管理することにより、高速に証明書の検証を行えると予想された。このキャッシュを用いたいくつかの高速化の手法を開発し、これらの効果を測定したので報告する。

3. 証明書検証の高速化

今回、以下の3つの手法により高速化を行った。

3.1. 証明書のキャッシュによる高速化

証明書検証処理において、一度取得した証明書と失効情報をキャッシュすることで高速化する。従来の手法では、検証のたびにそれに必要な証明書と失効情報をディレクトリサーバにアクセスして取得していた。高速化した検証アルゴリズムでは、図1のように認証局の証明書や失効情報をキャッシュすることで、ディレクトリサーバへのアクセスを減らし、高速化を図る。

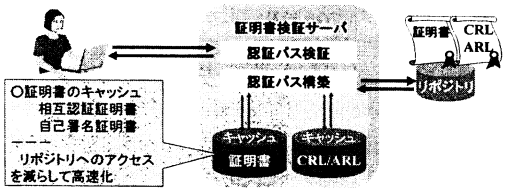

図2　証明書のキャッシュによる高速化

3.2. パスキャッシュによる高速化

一度構築した認証パスをキャッシュすることで高速化する。認証パスの構築は、証明書の記載内容をもとに行うが、最適化された手法がない。現状では、考えられるすべての証明書を収集し、それを用いて認証パスを構築する必要がある。そのため、認証パスの構築処理はサーバに負荷が大きいと予想され、図2のように一度構築した認証パスをサーバ上でキャッシュする。前記の証明書と失効情報のキャッシュに加え、認証パスをキャッシュすることでリクエストのたびに毎回行っていた認証パスの構築作業を簡略化し、高速化を図る。

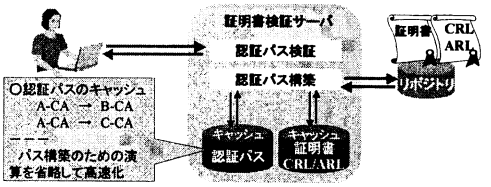

図3　認証パスのキャッシュによる高速化

3.3. 失効テーブル作成による高速化

証明書検証サーバにおいて、図4のように失効情報から失効テーブルを作成することで高速化を行う。ある証明書の失効状態を確認するためには証明書のシリアルナンバが失効情報に記述されていないか確認をする必要がある。図4の左側のように一般にCRLやARLで提供される失効情報には失効した証明書のシリアルナンバが失効した順番に記述されている。そのため、失効リストに載っていない証明書の失効確認をするためには、失効リストの先頭から最後まで確認して失効リストにシリアルナンバが記載されていないことをチェックする必要がある。失効リストの中身が少ないうちは処理時間への影響が少ないが、公的個人認証サービスなど1万件を越す失効情報が発行される場合は、処理時間への影響が顕著になると考えられる。

そこで証明書検証サーバでは、図4の右側のようにあらかじめ登録した失効リストのレコードをシリアルナンバでソートし、二分木検索でシリアルナンバの検索を行う。二分検索は全検索に比べ検索のための処理が少なく、特にレコードが増えた際は顕著に差異が現れる。この失効リストをソートし、失効テーブルを作成する方法で高速化を図る。

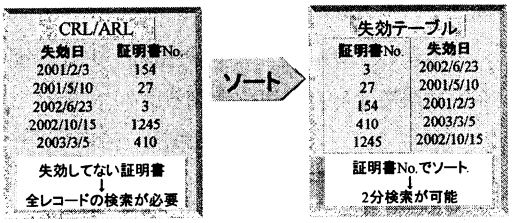

図 4　失効テーブル作成による高速化

にクライアント、証明書検証サーバ、ディレクトリサーバと CA で構築した。また、ネットワークは 100Mbps で接続した。

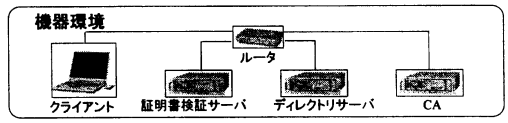

図 5　評価環境

4. 高速化の評価方法

4.1. 評価方法

3 章で述べた処理能力の向上手法を評価するために、証明書検証サーバにアルゴリズムを組み込み、処理能力の測定をおこなった。測定項目は、証明書検証サーバの時間当たりの平均処理件数と、平均応答時間の測定であり、下に示す条件で行った。

4.1.1. 証明書、CRL、認証パスキャッシュの効果

おのおの同時処理数が 5 で 500 回試行を行い、中間の約 300 回をサンプルとして抽出して平均をとった。認証パスの長さは 4 で行った。

A-1 キャッシュ機能を用いずに証明書検証を行う

A-2 証明書と CRL のキャッシュを行う

A-3 証明書、CRL、認証パスのキャッシュを行う

4.1.2. 失効テーブルの効果

おのおの同時処理数が 10 で 500 回試行を行い、中間の約 300 回をサンプルとして抽出して平均をとった。認証パスの長さは 2 で行った。利用した CRL のレコード件数は 2 万件で行った。

B-1 失効テーブルなし

B-2 失効テーブルあり

4.2. 実験環境

評価環境は図 5 のように、同じルータの配下

実験で用いた機器は以下のとおりである。

クライアント

CPU	Pentium® III　850MHz
メモリ	762MB
OS	Windows® XP

証明書検証サーバ

CPU	Ultra SPARC™ II　500MHz
メモリ	4GB
OS	Solaris® 8

ディレクトリサーバ

CPU	Pentium® III　700MHz
メモリ	762MB
OS	Windows® 2000

5. 結果・検討

5.1. 証明書、CRL、認証パスキャッシュの効果

A1～A3 の条件で測定を行った結果を表 1 に示す。

表 1　キャッシュの効果

	平均応答時間 (sec)	平均処理能力 (処理件数/sec)
A-1 キャッシュなし	1.54	3.17
A-2 証明書 CRL キャッシュ	0.86	5.57
A-3 証明書、CRL、認証パスキャッシュ	0.75	6.56

A-1 と A-2 の値から証明書・CRL キャッシュの効果により、平均応答時間が 1/1.8 倍、平均処理能力が約 1.8 倍になったことがわかる。一方、A-2 と A-3 から認証パスキャッシュの効果により、平均応答時間が 1/1.1 倍、平均処理能力が約 1.2 倍になった。2 種類のキャッシュにより全体として、平均応答時間が 1/2.1 倍、平均処理能力が 2.1 倍になったことがわかった。

上記のように、証明書・CRL のキャッシュ、認証パスのキャッシュともに証明書検証処理の高速化の効果を得ることができた。また、今回の評価環境においては、証明書・CRL のキャッシュは認証パスのキャッシュと比較して、効果が大きいことがわかった。このことから、評価環境において証明書検証処理においてはネットワーク経由で CA 証明書や失効情報を取得するのに多くのリソースを費やしていたことが予想される。

一般的にも、このようなキャッシュの効果が現れるのは、一度利用され、キャッシュとして保存された CA 証明書や CRL を再度利用した場合である。そのため、クライアントのように同じ証明書を何度も検証しない環境では効果は限定的であると予想される。一方、同じ CA 証明書や CRL を利用して大量の検証を行うサーバにおいては、キャッシュのヒット率が高まり、上記のようなキャッシュが効果を発揮すると考えられる。

5.2. 失効テーブルの効果

B-1 と B-2 の条件で測定を行った結果を表 2 に示す。

B-1 と B-2 の結果から、失効テーブル作成の効果により、平均応答時間が 1/32 倍、平均処理能力が約 30 倍になったことがわかった。なお、A-3 と比較して、大規模 CRL を用いている B-2 の方が高速に処理を行っているのは、認証パスの長さが短いためである。

表 2　失効テーブルの効果

	平均応答時間 (sec)	平均処理能力 (処理件数/sec)
B-1 失効テーブルなし	19.7	0.50
B-2 失効テーブルあり	0.61	15

上記のように、失効テーブルの作成が平均処理時間の減少と、平均処理能力を向上させ、証明書検証処理を高速化させた。これは、あらかじめ情報をソートすることで、CRL 中のレコードを検索するための処理が大幅に削減されているからであると考えられる。

このことから、大きな CRL が発行されるような大規模な認証環境においても、失効テーブルを用いることで証明書検証処理能力の向上が行えると考えられる

6. まとめ

証明書・CRL のキャッシュと認証パスのキャッシュを行うことで証明書検証の高速化が実現された。これらのキャッシュ機能は特に大量の検証処理を行うサーバにおいて、上記のようなキャッシュ機能が処理速度向上の効果を示すと考えられる。

また、2 万件の大規模 CRL で証明書の検証を行う際に、CRL のレコードをソートして保持することで、大きな検証速度の向上を実現した。このことより、大きな CRL が発行されるような大規模な認証環境において、CRL レコードを作製して検証処理に用いることで処理能力の向上が行えると考えられる。

参考文献

1) ITU-T Recommendation X.509、 "Information technology - Open Systems Interconnection - The

Directory: Public-key and attribute certificate frameworks "(2000)

2) A. Frier、 P. Karlton、 and P. Kocher、 "The SSL 3.0 Protocol" "http://wp.netscape.com/eng/ssl3/ssl-toc.html" (1996)

3) T. Dierks and C. Allen: RFC 2246:"The TLS Protocol Version 1.0" (1999)

4) 電子署名及び認証業務に関する法律(平成十二年法律第百二号)

5) 政府認証基盤相互運用性仕様書 http://www.gpki.go.jp/session/CompatibilitySpecifications.pdf (2001)

6) 電子署名に係る地方公共団体の認証業務に関する法律(平成十四年法律第百五十三号)

7)藤城孝宏、鍛忠司、羽根慎吾、熊谷洋子、手塚悟: 電子情報通信学会論文誌 D-I Vol. J87-D-I No.8: "証明書検証サービスの開発" (2004)

8) R. Housley、 W. Ford、 W Polk and D. Solo: RFC 3280: "Internet X.509 public key infrastructure certificate and Certificate Revocation List (CRL) profile" (2002)

9) A. Malpani、 S. Galperin、 M. Myers、 R. Ankney and C. Adams: RFC 2560: "X.509 Internet public key infrastructure online certificate status protocol - ocsp" (1999)

10) A. Malpani、 R. Housley and T. Freeman: IETF Internet drafts: "Simple Certificate Validation Protocol (SCVP)"、 "draft-ietf-pkix-scvp-16.txt"、 2004(work in progress)