Computer Security Symposium 2017
23 - 25 October 2017

エッジコンピューティングを利用した公開鍵証明書検証の提案

北島　祥伍[1]　満保　雅浩[1]

概要：利用者に近いエッジで運用されるエッジコンピューティングは，クラウドを補完する形で，クラウドのみよりも迅速に各種サービスを提供できると期待されている．一方，公開鍵証明書の正当性を検証するために使用される OCSP などの手法は効率と安全を両立することが課題として残されており，改善が望まれている．そこで本稿では，エッジコンピューティングを導入し，公開鍵証明書の検証を効率的かつ安全に行う方式を提案し，実装評価する．

キーワード：PKI,　公開鍵証明書,　エッジコンピューティング,　Web セキュリティ

Verifiying the Validity of Public Key Certificates Using Edge Computing

Shogo Kitajima[1]　Masahiro Mambo[1]

Abstract: Edge Computing, which is created near client, is expected to provide services faster than using only Cloud Computing. Meanwhile, methods to verify the validity of public key certificate such as OCSP has some problem for achieving both efficiency and safety. In this paper, we propose a certificate verification method using Edge Computing, and we implement and evaluate it.

Keywords: PKI,　Public Key Certificate,　Edge Computing,　Web Security

1. はじめに

公開鍵基盤（PKI; Public Key Infrastructure）は，現在 SSL 通信などで広く普及した技術である．PKI に於いては認証局が公開鍵と所有者の対応関係に関する情報を記述した電子証明書を発行する．証明書の正当性は信頼される認証局によるデジタル署名により保証される．しかし，証明書の中には秘密鍵の漏洩や発行者に関する情報の変更などの理由で有効期限を待たずに証明書が失効することがある．そのため，有効期限内であっても証明書の失効確認を行う必要がある．失効確認を行う方式には CRL（Certificate Revocation List），OCSP（Online Certificate Status Protocol）などの方法があるがそれぞれ応答速度やプライバシ保護などの課題がある．

エッジコンピューティング [1] はユーザの近くにエッジサーバを設置し，距離による通信遅延の改善を試みる技術である．認証局が設置したサーバは分散されていないことが多く，遠距離の認証局に対する確認は時間がかかることがあるという問題が存在する．そこで，本論文では公開鍵証明書の失効確認のためのサーバをエッジに設置する手法を提案し，その有効性を実装により検討する．

2. 準備

2.1 公開鍵証明書

公開鍵証明書とは，公開鍵の所有者に関する情報を記述した証明書である．公開鍵暗号を用いる際に入手した公開鍵の正当性を検証することに用いる．現在，公開鍵証明書には RFC5280[2] で定義されている X.509 証明書が広く用いられている．X.509 証明書には公開鍵そのもの，署名ア

[1] 金沢大学 Kanazawa University

ルゴリズム，発行者，有効期間などの情報が含まれている．

公開鍵証明書を受信したクライアントは証明書を検証する必要がある．このためにクライアントは証明書チェーンの検証，署名の整合性の検証，失効確認などを行う．証明書の中には秘密鍵の漏洩，発行者に関する情報が変更になるなどの理由で有効期限を待たずに証明書が失効することがある．そのため，有効期限内であっても証明書の失効確認を行う必要がある．失効確認を行う方式には CRL，OCSP などの方法がある．

3. 既存手法

3.1 CRL

証明書失効リスト（CRL，Certificate Revocation List）は失効した公開鍵証明書のリストである．CRL の URL は公開鍵証明書に記載されており，認証局により公開されている．サーバが提示した証明書が失効リスト内に含まれていないか確認することで検証できる．証明書失効リストには有効期限が存在し，認証局は定期的に失効リストを更新する．

証明書失効リストはその認証局が発行した証明書全体に対する失効情報を含むため，そのサイズが大きくなることがある．不要な情報を含んでいるため帯域幅を浪費するなどの問題点がある．通信毎に証明書失効リストを取得するのは帯域幅を多く専有するため，通常，証明書失効リストはクライアント側でキャッシュされるが，必ずしも最新の状態に保たれないことも多い．認証局ごとに証明書失効リストを保存し，それを定期的に更新する必要がある．

3.2 OCSP

オンライン証明書状態プロトコル（OCSP:Online Certificate Status Protocol）は RFC2560，RFC6960 などによって規定されている証明書の失効状態を取得するためのプロトコルであり，その処理プロトコルは**図 2** のように示される．クライアントは証明書を受け取った時点で認証局が用意した OCSP レスポンダに OCSP 要求を行う．OCSP 要求は認証局情報と証明書のシリアル番号によって構成される．OCSP レスポンダはその要求に記述されている証明書の失効状態を調べ応答する．

CRL では証明書リスト全体をダウンロードするが，OCSP では必要なレコードについての情報のみ取得するため無駄が少ない．クライアントがキャッシュした CRL を最新の状態に保ち続けるのは労力がかかるのに対して，クライアントは OCSP を用いて最新の情報を中間認証局から取得することにより少ない労力で確認できる．しかし認証局にとっては OCSP レスポンダの専有する帯域は多いため，必ずしも良いことばかりではない [4]．また，OCSP レスポンダに障害などにより接続できなかった場合は失効情報を得ることができないという問題や，OCSP サーバが海外など離れた場所にある場合，通信遅延により時間がかかると言う問題がある．加えて，証明書ごとに複数の OCSP 要求が必要であるという問題や，本来サーバのみ知るべきクライアントのアクセス情報を OCSP レスポンダも知ることができるというプライバシ上の懸念もある．

3.3 OCSP stapling

RFC6066[3] において規定された TLS 拡張には，証明書状態リクエストに関するものが含まれている．クライアント証明書情報を TLS ハンドシェイクにおいて送る拡張で，一般に OCSP Stapling と呼称されている．OCSP stapling はクライアント Hello 中に証明書状態要求の拡張（status_request，status_request_v2）を含む場合に，証明書を用いるサーバが OCSP レスポンダに予め問い合わせを行い，得られた OCSP 応答をクライアントに公開鍵証明書と共に提供する TLS 拡張である．RFC6066 においては一つの証明書状態しか添付できなかったが，RFC6961[4] において複数の証明書状態を添付できる標準が規定され，一般に OCSP Multi-Stapling と呼称されている．

証明書を用いるサーバが予め OCSP レスポンダに問い合わせを行った上でクライアントに配信するため，認証局への帯域幅は削減される．OCSP 応答には認証局による署名がなされているため，発信者によらず OCSP 応答は信頼することができる．

しかし，OCSP Stapling の実装状況を調査したところ普及率は 24 %程度（121/500）と，あまり普及が進んでいない．

3.4 Google Chrome による実装

Google Chrome においては CRLSet と呼ばれる機能を利用している．これは失効情報を Google がブラウザに対して配信するものである．Google は厳格な実装をしなければ OCSP は効果的ではないと主張しており，Google Chrome では標準では OCSP は無効化されている [5]．しかし，CRLSet のファイルサイズの上限は 250KB と規定されており，大量の証明書が失効すると適切に働かなくなる懸念がある．

3.5 Mozilla Firefox による実装

Mozilla Firefox においては OneCRL と呼ばれる独自の機能を用いている．これは中間認証局の失効情報を Mozilla がブラウザに対して配信するものである．よってブラウザは中間認証局の失効情報を毎回取得する必要がなく，証明書チェーン末端の公開鍵証明書一つのみ OCSP で確認をすればよい．

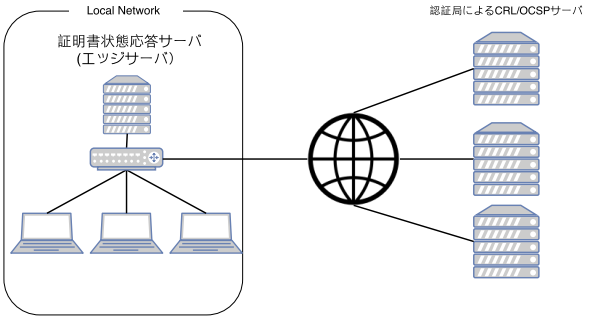

図 1　提案手法におけるネットワーク図

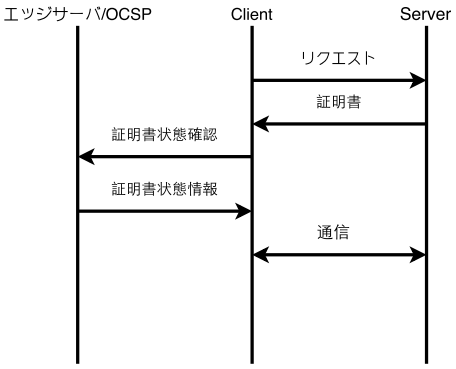

図 2　OCSP・提案手法における処理プロトコル

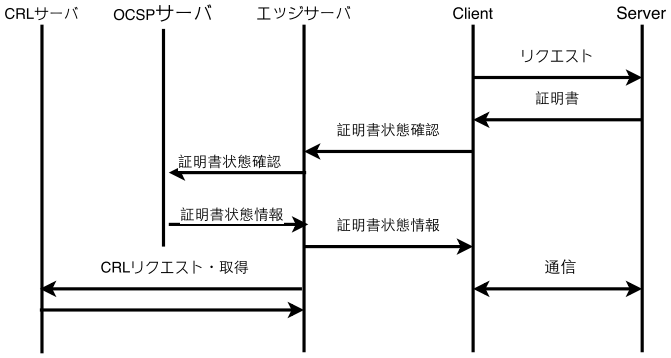

図 3　提案手法における処理プロトコル（エッジサーバが把握しない認証局の場合）

4. 提案手法

提案プロトコル

提案手法のネットワークを**図 1** に，その処理プロトコルを図 2，**図 3** に示す．図 1 のように，クライアントの近くに証明書状態を確認するための証明書状態応答サーバを設置し，クライアントはその証明書状態応答サーバに問い合わせを行うことで状態確認を行う．証明書状態応答サーバは定期的に多くの認証局の CRL を取得し，図 2 のようにその CRL を用いて OCSP サーバの代わりにクライアントの問い合わせに応じ失効状態を応答する．応答サーバが把握していない認証局の問い合わせに対しては，図 3 のように応答サーバは OCSP で認証局に問い合わせを行いその結果をクライアントに応答し，その後に次回以降の問い合わせに備えて CRL を取得する．

特長

OCSP Stapling はサーバ側による対応が必要であるが，提案手法ではサーバ側の特別な対応は必要ない．またクライアントの近くに設置することで，応答速度が早くなる．クライアントが自ら CRL を保管する場合，クライアントごとに認証局に問い合わせて CRL を取得する必要があるが，提案手法では同一の応答サーバを使うクライアントの間で証明書状態情報を共有することができるため，認証局への負荷も減少する．また，一箇所に CRL を集積しているため，一度の要求で複数の証明書の認証情報を取得することができる．

5. 性能評価

性能評価のため，提案手法を用いて失効状態を確認し HTTPS リクエストを送信するクライアントと OCSP を用いた同等のもの，及び提案手法で用いる証明書状態応答サーバを Kotlin（JVM 言語）で証明書状態の検証に着目して実装した．この OCSP を用いるクライアントは OCSP stapling に対応せず，全ての証明書に対して OCSP で確認を行う．あるユーザのコンピュータの Web ブラウザの履歴からランダムに選択した 500 の HTTPS サーバに対しリクエストを送信した．

表 1 証明書状態確認に掛かる時間 [ms]

パーセンタイル	5 %	25 %	50 %	75 %	95 %
OCSP	11	16	67	214	300
提案手法	0.74	0.88	1.01	1.19	2.1

表 2 HTTPS 通信に掛かる時間 [ms]

パーセンタイル	5 %	25 %	50 %	75 %	95 %
OCSP	232	484	922	1621	3650
提案手法	116	233	485	1119	2555

表 1 は証明書状態確認の通信に掛かる時間をパケットキャプチャより求めたものである．OCSP を用いて確認を行う際は，証明書チェーンの全ての証明書に対して証明書状態確認を行うため，一度の SSL 通信に対し複数の OCSP リクエストがなされる．表 1 の OCSP は，一つの OCSP リクエストにかかった時間を示している．表 1 より OCSP より 10〜100 倍程度高速に証明書状態確認を行うことができることが分かる．また，HTTPS 通信全体の高速化にも寄与していることが**表 2** よりわかる．

6. おわりに

本論文では，公開鍵証明書の検証にエッジコンピューティングを利用する方式を提案し，OCSP よりも検証の高速化を行えることを示した．複数のクライアントが証明書

状態応答サーバの役割を果たす同一のエッジサーバを利用するため，効率的なシステムを構築できる．その他にも，提案手法では証明書状態応答サーバの役割を果たすエッジサーバが失効リストの更新頻度を上げることにより，クライアント側で Web ブラウザによる実装より安全性の高い通信が可能となる．

今回の実装は JSON により実装したため，より効率的なデータ形式で実装することで高速化の余地がある．これらの課題に今後取り組んでいく予定である．

参考文献

[1] A. Davis, J. Parikh, and W. Weihl, EdgeComputing: Extending Enterprise Applications to the Edge of the Internet, WWW 2004.

[2] RFC5280 Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile, 入手先 ⟨https://www.ietf.org/rfc/rfc5280.txt⟩

[3] RFC6066 TLS 拡張：拡張定義 (Transport Layer Security (TLS) Extensions: Extension Definitions),2011 入手先 ⟨https://www.ipa.go.jp/security/rfc/RFC6066JA.html⟩

[4] RFC6961 The Transport Layer Security (TLS) Multiple Certificate Status Request Extension, 2013 入手先 ⟨https://www.ietf.org/rfc/rfc6961.txt⟩

[5] Chrome Security FAQ 入手先 ⟨http://bit.ly/2wbU925⟩