ACME プロトコルを利用した UPKI サーバ証明書の自動更新システムの試験的な実装

宇田川 暢

東海国立大学機構 名古屋大学

udagawa@icts.nagoya-u.ac.jp

Experimental implementation of automatic UPKI server certificate renewal system using ACME Protocol

UDAGAWA Mitsuru

Nagoya Univ., Tokai National Higher Education and Research System

概要

近年、サーバ証明書の有効期限が短縮され続ける傾向にあり、その更新作業に対応する負荷が大きな問題になる懸念がある。学内でサーバ証明書の発行申請に対応する「登録担当者」、および、実際にサーバ証明書の発行申請や設定作業を行う「利用管理者」の負荷軽減を目的とし、ACME プロトコルを用いた自動更新システムの実現可能性の検討を行った。

1 はじめに

国立情報学研究所より、UPKI 電子証明書発行サービスが提供され、加入機関はサーバ証明書やクライアント証明書などを利用することができるようになっている。

現在は同サービスで提供されるサーバ証明書（以下、UPKI サーバ証明書）の有効期限は 396 とおよそ 13 ヵ月である。これは「2020 年 10 月 1 日以降は 398 日を超える有効期限を持つサーバ証明書の発行を許可しない」というように規定が CA/Browser Forum の Baseline Requirements for the Issuance and Management of Publicly-Trusted Certificates（以下、BR） Version 1.7.5 で改定され、UPKI サーバ証明書もそれに準拠する必要があるためである。

サーバ証明書の有効期限について、以前は最長 5 年であったが、3 年、2 年と短縮されてきた経緯がある。今後さらに有効期限が短縮されることが予想されるが、有効期限が短くなるにつれて、サーバ管理者がサーバ証明書の更新作業を行う頻度が増し、負荷が増大する。登録担当者として対応する学内の IT 部門についても同様となる。

そこで、サーバ管理者、登録担当者の双方の負担を軽減できる自動化されたシステムの実現可能性を検討してみる。

2 サーバ証明書発行の自動化

2.1 従来のサーバ証明書の発行申請

ホストの所有証明のみを目的とした DV(Domain Validation)サーバ証明書を取得する場合には、サーバ側で作成した CSR(Certificate Signing Request)とともに所定のフォームに情報を記載の上、ドメインの Whois 情報に登録されたメールアドレスやドメイン名に関連した指定のメールアドレスに送られたメール中のリンク等から当該ホストの所有の事実を確認し、サーバ証明書を発行するという方式の有償サービスが一般的であった。また、2010 年代前半ではその費用として年間数千円（単一 FQDN）から数万円（ワイルドカード）をサーバ証明書の取得および維持に支払う必要があった。

2.2 Let’s Encrypt の登場

2015 年に Internet Security Research Group[1] (ISRG)により、サーバ証明書の提供サービスが開始された。Let’s Encrypt と呼ばれるこのサービスは、無償で自動化されたサーバ証明書発行手続きによ

[1] https://www.abetterinternet.org/

り、サーバ証明書の導入を容易にした。

Let's Encrypt でのサーバ証明書の取得は、クライアントと Boulder[2]と名付けられた CA(Certificate Authority) サーバとの間を、ACME(Automatic Certificate Management Environment)プロトコルと呼ばれる HTTPS 通信により行われる。クライアントは主として Certbot[3]が利用されていると思われる。

3 ACME プロトコル

3.1 ACME プロトコルの概要

ACME プロトコルは RFC 8555 に規定されている。RFC の Draft で存在した ACMEv1 は現在では廃止されており、ACMEv2 が利用されているため、本稿では ACMEv2 について解説する。

ACME プロトコルは前述の通り ACME クライアントから ACME サーバに対するサーバ証明書発行の要求からホストの所有の確認、サーバ証明書のダウンロードまでのやりとりについて定めていて、サーバ証明書を取得するまでの大まかな流れは以下のようになる。

3.2 Directory 情報の取得

ACME クライアントはまず ACME サーバの /directory エンドポイントにアクセスし、今後の処理に必要ないくつかのエンドポイントの URL などを得る。

3.3 nonce の利用

ACME プロトコルでは基本的にリクエストごとに nonce(Number Used Once)と呼ばれる使い捨ての値を利用する。これはリプレイ攻撃を防ぐために利用され、最初の nonce は Directory 情報に含まれる newNonce エンドポイントから取得し、次回以降は ACME サーバのレスポンスに含まれるものを利用することになっている。

3.4 Account の作成または認証

ACME クライアントは公開鍵暗号の鍵ペアを生成し、公開鍵を newAccount エンドポイントに送信する。ACME サーバは公開鍵を登録し、利用者の識別に利用する。また、ACME クライアントはリクエスト改竄防止のため、以降のリクエストで公開鍵と、秘密鍵で行った署名を付与して送信することになっている。

3.5 Order の作成

ACME クライアントはサーバ証明書の発行を要求する全ての FQDN のリスト(以下、identifiers)を newOrder エンドポイントに送信する。ACME サーバは Order として保存する。

3.6 Authorization の作成

ACME サーバは Order の identifiers 中の FQDN 毎に、それぞれの認証結果を記録する Authorization を作成し、それらへのエンドポイントを含めて newOrder へのレスポンスとして返す。つまり、一つの Order に対して FQDN の数だけ Authorization が作成される。また、各 Authorization には（複数の）Challenge が含まれている。

3.7 Authorization へのアクセス

ACME クライアントがレスポンスに含まれる任意の Authorization のエンドポイントにアクセスすると、その FQDN の所有を証明するための Challenge の情報が返される。

3.8 Challenge によるチェック

ACME クライアントは Authorization に含まれる任意の手段（現在は http-01 および dns-01 のみ規定）で FQDN の所有を証明する。http-01 では ACME プロトコルで規定された場所(URL)から指定されたトークンを ACME サーバから読み取れること、dns-01 では DNS の規定されたレコードから指定されたトークンを ACME サーバが名前解決可能なことをもって FQDN の所有証明としている。

ACME クライアント側で準備が完了した際に Challenge 用エンドポイントにアクセスすると、ACME サーバは指定された手段でチェックを行う。

チェックの結果として ACME サーバが FQDN の所有を確認できた場合は、Challenge を valid とする。一つでも Challenge が valid となった Authorization は valid となる。

3.9 CSR の送信

全ての Authorization が valid となった後、ACME クライアントは Order のレスポンスに含まれていた finalize 用エンドポイントに CSR を送信する。CSR に問題が無ければサーバ証明書の作成を行う。この時点ではサーバ証明書そのものや、ダウンロード方法に関する情報は提供されない。

3.10 サーバ証明書のダウンロード

ACME クライアントは CSR の送信後、Order のエンドポイントにアクセスを続け、certificate エイドポイントの URL がレスポンスに含まれた場合に、certificate エンドポイントにアクセスし、サー

[2] https://github.com/letsencrypt/boulder
[3] https://github.com/certbot/certbot

バ証明書の取得を行う。

4 ACME プロトコルを利用した UPKI サーバ証明書取得の検討

4.1 前提条件

UPKI サーバ証明書の発行を自前の専用サーバと ACME プロトコルにより自動化することを目指し、まず技術的に問題になる可能性となる点について洗い出しを行った。

4.2 ACME クライアント

ACME クライアントを自作すると開発の手間がかかるだけでなく、サポートの面からも現実的ではないため、特に致命的な問題が無い場合は、前述の Certbot を利用する前提とした。なお、本稿では検討および検証に”CertbotACMEClient/1.22.0”を利用している。

4.3 ACME サーバのエンドポイント URL

Certbot ではデフォルトで Let’s Encrypt の ACME サーバに接続するようになっている。これについて独自のエンドポイントを指定することが可能であるかが問題になるが、**--server** オプションで任意に指定できることが分かった。

4.4 CSR の作成

Certbot では与えられた FQDN から CSR を作成するが、DV 証明書用の CSR となるため、UPKI で必要な他の DN の属性値を実行時に指定することができない。また、UPKI では SANs(Subject Alternative Names)の値を CSR に含めることも許可されていない。ACME プロトコルでは ACME サーバ側から送られた秘密鍵をクライアント側が受け取ることを明示的に禁止しているため、ACME サーバ側で CSR を作成することも不可能である。

これらについては**--csr** オプションで別途作成した CSR を利用することで解決することとした。

4.5 SANs の指定

CSR に含めることができない以上、なんらかの方法で SANs を ACME サーバに送信する必要がある。これについては**--user-agent-comment** オプションで実行時に指定するしか他に手段が無いと考えた。

4.6 OU(Organizational Unit Name)の判定

部署名を示す OU については事前に UPKI によって許可されたリストに含まれている必要があり、また、新規 OU の場合は無関係な組織や他者のサービス等を誤認させるようなもので無いことが求められる。

この新規 OU の扱いに苦慮していたが、本システムの検討中に BR 1.7.9 より OU が廃止となったため、OU については考慮する必要がなくなった。

4.7 サーバ証明書作成にかかる時間

経験上、UPKI サーバ証明書は「電子証明書自動発行支援システム」（以下、支援システム）に申請用 TSV を送信してからサーバ証明書がダウンロード可能になるまで、およそ数分から 10 分程度の時間が必要になる。Certbot がタイムアウトにならずに発行完了まで耐えられるかについては、**--issuance-timeout** オプションで 900 秒など十分に大きい値を与えることで解決できると考えた。

4.8 支援システム上での状態フラグ

UPKI サーバ証明書は支援システムでの申請後にダウンロード用 URL がメール通知され、通常はそのリンク先からダウンロードすることになる。しかしながら、今回検討のシステムではそのような処理を行うことが困難なため、支援システムから「サーバ証明書証明書情報一括ダウンロード TSV」（以下、証明書一括 TSV）を取得し、そこからサーバ証明書を取得することになる。

その場合に、支援システム上で”状態”フラグが「発行案内メール送信済み」から変更されないため、サーバ証明書の失効申請を行うことが不可能になってしまう。

これについては対応の仕様が無いため、自動化したとしても通知メールのリンク先から証明書を取得する作業を行うように利用者に依頼することが妥当と思われる。

4.9 証明書一括 TSV のダウンロード頻度

支援システムから証明書一括 TSV をダウンロードする際にレートリミットが存在するため、支援システムへの申請用 TSV 送信の後、証明書一括 TSV をダウンロードして処理状況を確認する場合は、あまり頻繁に支援システムにリクエストが送られることが無いように考慮する必要がある。

4.10 検討結果

以上の結果から、Certbot を利用するサーバ証明書申請システムを作成することが可能だろうという結論を得た。

5 ACME サーバの実装

実際に UPKI サーバ証明書の申請を行う ACME サーバ（以下、本サーバ）を開発するにあたり、支援システムとのやり取り部分を作成する必要性

などから、以前に作成した「UPKI サーバ証明書管理用ツール」[1]を拡張する形で実装することとした。

本サーバでは ACME プロトコルの全てを実装するのではなく、実現可能性の検証に必要な部分に限定した。そのため、前掲のサーバ証明書取得までの流れのうち、Account についてはリクエストに対して HTTP_CREATED を返すだけのスタブとなっている。また、実際に支援システムに TSV を送るのではなく、TSV 作成に必要情報を取得するのみに留め、ダウンロードさせるサーバ証明書も以前に発行されたサーバ証明書を返すようにしている。前者は申請のサービスデザインやシステム連携が必要なためで、後者は支援システムへの負荷や実際には不要なサーバ証明書が多数発行されてしまうことを避けるためである。

結果として期待した通りに動作することが確認できた。

6　本サーバの実装により得られた知見

6.1　認証と Account の問題

ACME プロトコルには今のところ実行途中にプロンプトを表示して認証を要求するような仕組みは存在しない。しかし、externalAccountBinding と呼ばれる仕組みを利用し、Certbot の実行時に引数として識別子と MAC キーと呼ばれる値を渡すことが可能となっている。

この仕組みを利用して、事前に別途用意する申請用ページで認証後に必要な値を取得し、利用管理者の認証を行うことが可能になる。

あるいは、初回申請だけ従来の方法で行うこととして、更新時には本サーバでの自動更新を可能にするといった対応も考えられる。この場合は externalAccountBinding を利用する必要はない。

6.2　処理時間の問題

通常であれば certificate エンドポイントに断続的にアクセスを継続したのち、90 秒後にタイムアウトとして打ち切られるところ、見込み通りオプションによりタイムアウトを延長することで 10 分以上も対応することが可能であった。ただし、その間 Certbot の画面出力は変化しないため、利用管理者が処理が進んでいないと判断して Certbot を停止する事態も考えられる。やはり長くても TSV 送信後 30 秒以内にサーバ証明書のダウンロードが完了できることが望ましいであろう。

6.3　中間証明書について

通常は UPKI サーバ証明書用の中間証明書を指定されたサイトからダウンロードして適用する必要があるが、本サーバからのサーバ証明書ダウンロード時に中間証明書を結合してからダウンロードさせることで、利用管理者の手間を減らすことが可能となる。

7　さいごに

いくつか解決すべき課題があるものの、UPKI サーバ証明書の自動更新システムは実現可能であると言える。

ただし、各大学がシステムを導入するよりは、支援システム側で ACME プロトコルに対応することが自然であるように感じる。その場合は、externalAccountBinding を利用した認証をそれぞれの大学が求める認証方法とうまく擦り合わせることが重要であると思われる。

参考文献

[1]　宇田川、UPKI サーバ証明書管理用ツールの開発、大学 ICT 推進協議会、2019