SSL/TLSの仕組みを知っていますか？

神田 雅透[†]

1. まえがき ～SSL/TLSを使ったことがありますか～

まず皆さんに質問です。

「あなたは、Amazonで買い物したり、オンラインでチケットや楽曲を買ったりしたことがありますか？あるいは、ネットバンキングやネットトレードなどをしたことがありますか？」

この質問の答えが「YES」だった人は、例えSSL/TLS^{*1}のことを知らなかったとしても普通にSSL/TLSを使っているということになります。

SSL/TLSは、クレジットカード番号やパスワードなどの重要な情報をインターネット上で安全にやり取りするために使うセキュリティプロトコルの一つです。ブラウザに標準搭載されており、ブラウザのURLが表示されているあたりに「鍵（南京錠）マーク」が表示されていたら、その通信はSSL/TLSで通信している状態になっていることを表しています。

本稿では、SSL/TLSはどのような仕組みで動いているのかを解説します。

2. SSL/TLSは何をするものですか？

～役割と仕組み～

SSL/TLSには、以下の2つの大きな役割があります。

- (1) 通信相手を確認する：ネットワークを介した通信相手が本当に自分の意図した相手であるかを確認します。例えば、アクセスした先のサイトが、本物に似せたフィッシングサイトなどの危険なサイトではないことを確認することです。
- (2) 通信内容を保護する：通信内容を第三者に盗聴されないように、暗号通信を行います。オンラインショッピングなどで「情報は暗号化されています」といった説明があるとき、この機能が使われているこ

図1 SSL/TLSの動作

とを意味しています。

次に、実際のSSL/TLSの動作を見てみましょう。

SSL/TLSでは暗号通信が始まるまでに、大きく三つのステップが実行されます（図1）。

最初に行われるのが、SSL/TLSを使ってあるサイト（サイトA）にアクセスしたときに、アクセスした先のサイトAが本物であるかどうかの確認です。

その確認のために使われるのがSSL/TLSサーバ証明書（単にサーバ証明書ともいいます）です。具体的に何を行っているかは4章を参照してもらおうとして、概念としては、「サイトAの正しいサーバ証明書を持っているのはサイトAだけである」ということを踏まえ、ブラウザが受け取ったサイトAと称したサーバ証明書が正しいければ、そのサーバ証明書を送ってきたサイトは正しいサイトAである、と判断し、次の暗号通信のための準備に移ります。

もしサーバ証明書が正しいものと確認できなかった場合には、図2のように何らかの警告表示がブラウザに出るようになっています。

サーバ証明書によって自分の意図したサイトAにアクセスしていると確認できると、次のステップとして、暗号通信を行うための秘密鍵をブラウザが自動的に作成し、後述

*1 SSL/TLS: Secure Socket Layer/Transport Layer Security.

[†] NTTセキュアプラットフォーム研究所
"Security Technologies on Image Information (3): How SSL/TLS Works?: Role and mechanism" by Masayuki Kanda (NTT Secure Platform Laboratories, Tokyo)

図2 警告表示の例 (IE 8 (上), Firefox 33 (下))

する公開鍵暗号を使ってサイト A にその秘密鍵を送ります。そして、サイト A がその秘密鍵を取出した時点で、ブラウザとサイト A との間で安全に秘密鍵の共有ができたことになります。

これで、SSL/TLS のもう 1 つの役割である、暗号通信のための準備が終了したことになります。その後、ブラウザとサイト A とで共有した秘密鍵を用いて実際の暗号通信が始まります。この暗号通信では共通鍵暗号が使われます。

3. どんな暗号を使いますか？

～共通鍵暗号・公開鍵暗号～

SSL/TLS では、共通鍵暗号 (図3) と公開鍵暗号 (図4) と呼ばれる二つの方式を使います。

共通鍵暗号は、暗号化と復号で同じ鍵 (秘密鍵と言います) を使う方式で、はるか昔、古代ローマ時代から使われ

図3 共通鍵暗号

図4 公開鍵暗号

てきました。暗号化処理が高速であるという特長を持つため、主に大きなサイズのコンテンツやメッセージの暗号化に使われています。

共通鍵暗号で使う秘密鍵は通信相手同士が互いに秘密に持たなければなりません。そのため、通信相手同士が何らかの手段で事前に秘密鍵を安全に共有する必要があります。この欠点を解決したのが公開鍵暗号です。

公開鍵暗号は、暗号化鍵と復号鍵が異なる方式で、その暗号化鍵で作った暗号文は対応する復号鍵でしか元に戻せません。しかも、暗号化鍵は公開することができる一方、復号鍵は一人が秘密にして持てるという特長があります。そこで、前者を公開鍵、後者を個別鍵とも言います。

この特長は、事前に鍵共有を行う必要がない、不特定多数の相手と暗号通信を行う場合に適しています。例えば、不特定多数の利用者を相手にするサイト A では、サイト A の公開鍵で暗号化してもらうようにすることで、どのブラウザからもサイト A への暗号通信を事前準備なく始めることができます。

ただし、共通鍵暗号と比較すると、はるかに複雑な計算を必要とするため、暗号化処理が遅く、大きなサイズのコンテンツやメッセージなどの暗号化には向きません。そのため、共通鍵暗号で使う秘密鍵の配送手段として公開鍵暗号を使い、実際のコンテンツやメッセージは共通鍵暗号で行うハイブリッド型 (図5) がよく使われます。SSL/TLS はハイブリッド型の典型的な利用例です。

ところで、公開鍵暗号では復号鍵を一人しか持たないことから、その鍵を逆に利用して暗号文を作るとすると、その鍵 (この場合は、復号鍵が暗号化鍵になりますが) による暗号文は一人しか作れないともいえます。この概念を応用したものがデジタル署名 (図6) です。メッセージに対し、秘密に持つ鍵 (署名鍵) で署名を行い、公開されている鍵 (署名検証鍵) で署名検証することによって、そのメッセージの正当性と署名者を検証できます。

図5 ハイブリッド型

図6 デジタル署名

4. どうやって相手を信じますか？

～トラストモデル～

4.1 公開鍵暗号基盤

サーバ証明書は、①ブラウザに対して、アクセスした先のサイトAが意図する相手（運営組織等）によって管理されるサーバであることを確認する手段を提供すること、②SSL/TLSによる暗号通信を行うために必要なサーバの公開鍵をブラウザに正しく伝えること、の2つの役割を担っています。前者がアクセスした先のサイトが本物であることの確認のために使われ、後者がブラウザの作った秘密鍵を暗号化してサイトAに安全に送るために使う公開鍵として使われます。

ところで、アクセスした先のサイトAが意図した相手であると確信できるためには“サイトAのサーバ証明書が本物”でなければなりません。そこで、サーバ証明書が本物であると確認できる仕組みが必要となります（図7）。

具体的には、信頼できる第三者としての認証局（CA*²）が、サイトA自体の存在とサイトAの公開鍵が確かにAのものであるということの両方を確認した後、サイトAの公開鍵に認証局自らのデジタル署名を付与します。これがサイトAのサーバ証明書となります。

ブラウザでは、サイトAのサーバ証明書に認証局の正当なデジタル署名が付いていることを根拠に、認証局によってサイトAの存在が確認されており、かつサイトAの公開鍵が正しいものと判断します。

サーバ証明書の中でも、サーバの運営組織の法的実在性をCA/Browser Forumが規定した国際的な認定基準に基づいて認証局が確認して発行されるサーバ証明書はEV*³証明書と呼ばれています。ブラウザの「グリーンバー」機能により、有効なEV証明書で運用されているサイトにアクセスするとアドレス表示部分等が緑色になるため（図8の黒枠内）、利用者にとって有効なサーバ証明書の視認性が高いものとなっています。

ところで、どうして認証局が信頼できるといえるのかと疑問に思った方もいるかもしれません。勝手に認証局になることはないのでしょうか。

答えから言えば、サーバ証明書の発行プログラムさえあれば誰もがサーバ証明書を自由に作ることができ、認証局の役割を担うことができます。つまり、どの認証局が発行したサーバ証明書かによって信頼性はまったく異なります。

そこで、今度は認証局をどうやって信頼するのかという課題が出てきます。この課題を解決する仕組みが公開鍵認証基盤（PKI*⁴）です（図9）。PKIを簡単に言えば、下位の認証局の信頼性を上位の認証局が保証し、その上位の認証局の信頼性をさらに上位の認証局が保証する、といった階

図7 サーバ証明書の使い方

図8 EV証明書を扱うサイトでのアドレス表示

図9 公開鍵認証基盤

層型の信頼スキームのことです。そして、最上位に位置する認証局のことをルート認証局といいます。

Web Trust for CAなどの一定の基準を満たしていることを第三者の監査機関によって審査された代表的なルート認証局は、「信頼されたルート証明機関」や「認証局証明書」として予めブラウザに登録されています。これにより、そのルート認証局が発行するルートCA証明書を信頼起点（トラストアンカ）として、順次検証を行い、最終的にサーバ証明書の正当性を自動的に検証する仕組みになっています（図10）。

4.2 公開鍵暗号基盤の信頼性が揺らいだ事件

とは言え、PKIの信頼性が大きく揺らいだ事件が2009年～2012年に相次いで発覚しています。特に衝撃的だったのは、2008年末に発表されたRapidSSL事件¹⁾、2011年に起きたDigiNotar事件²⁾、2012年に発覚したFlame事件³⁾です。

いずれの場合も、ブラウザベンダーがルートCA証明書を

*2 Certification Authority.
*3 Extended Validation.
*4 Public Key Infrastructure.

図10 ブラウザでの証明書の自動検証の様子

図11 RapidSSLでの偽証明書の検証

失効させるなどのセキュリティパッチを緊急リリースするなどの対策に迫られました。

(1) RapidSSL 事件

安全性が低下したハッシュ関数MD5の脆弱性を悪用して、偽認証局のCA証明書が偽造された事件で、2008年末のChaos Communication Congressで発表されました¹⁾。

この偽認証局のCA証明書は、ルート認証局(RapidSSL)が発行した正規のCA証明書であるとブラウザが判断するように作られていたため、偽認証局が発行した偽サーバ証明書でも、ブラウザは最終的に正しいサーバ証明書であると誤認証してしまうことを実際に示しました(図11)。

この事件は、PKIの中で有効に機能してしまう偽CA証明書が実際に偽造できることを証明するために行われた実験であったため、実害が生じることはありませんでした。ただし、当時はCA証明書やサーバ証明書にMD5を使っているものが少なくありませんでしたが、この事件を契機に多くの認証局がMD5を使うCA証明書やサーバ証明書の発行を取りやめました。

(2) DigiNotar 事件

2011年夏に、オランダに本拠を置くルート認証局DigiNotarがComodo hackerと名乗る人物に不正侵入され、少なくとも531枚ものサーバ証明書が不正発行されたことが判明した事件です。そのうち、少なくとも344枚にはGoogle、マイクロソフト、Mozilla、Skypeなど有名なドメイン名が使われ、またイスラエル諜報特務局、英国MI6、米国CIAといった諜報機関のドメインも含まれていました。

図12 Flameでの検証

この事件は単なる不正侵入事件にとどまらず、①Comodo hackerは約半年前に別の認証局Comodoでも偽サーバ証明書を不正発行させる攻撃に成功していたこと(この事件が発端でComodo hackerとの名がついた)、②Comodo hackerはイラン在住の人物であることを名乗っており、実際にDigiNotarから不正発行されたサーバ証明書がイラン政府機関(体制側)等による盗聴行為に利用された可能性があったこと、③マイクロソフト、Mozilla、Google、Appleなどの主要ブラウザベンダがDigiNotarのルート証明書を失効させるセキュリティパッチを緊急発行したこと、④DigiNotarが信用を失って破産したこと、といった実害が発生しました。

(3) Flame 事件

サーバ証明書での不正事件とは違いますが、PKIの信頼性を失わせた事件という意味で2012年に発覚したFlame事件も紹介しておきます。

サーバ証明書の代わりにドキュメントやソースコードに認証局の署名をつけるとそのドキュメントやソースコードを認証局が保証したことになります。

Flame事件では、Windowsが上位の認証局が保証したCA証明書と区別できないような偽認証局のCA証明書(図12の偽CA証明書MS)が偽造され、その偽認証局がマルウェアであるFlameを保証していました。しかも、そのときのルートCA証明書がMicrosoft Root Authorityであったため、マルウェアであるFlameがWindows updateの正規対象とWindowsが誤認することとなりました。

この事件が衝撃的なのは、①イランを中心にFlameの感染が拡大しており、発覚する5年以上前から存在していた可能性があること、②米国政府機関がFlameに関与していたと報道されたこと、③イランの核開発を中断させたstuxnetと兄弟的な関係にあるマルウェアであったこと、など、Flameがサイバー兵器としての様相も見せていたことです。しかも、Windows updateの正規対象と見えていたことから、利用者がWindowsのセキュリティパッチを適用したら気付かないうちにFlameに感染した可能性さえあります。

なおFlame事件を受けて、マイクロソフトはWindows updateで使うルートCA証明書とそれ以外の目的で使う

ルートCA証明書を分離しました。この対策により、現在では、非正規な更新プログラムをWindows updateの対象とWindowsが誤認するような事態はなくなりました。

5. どうやって情報を守りますか？

～暗号との関係～

5.1 プロトコルバージョンと暗号スイート

SSL/TLSは鍵交換アルゴリズム、署名アルゴリズム、暗号化アルゴリズム、ハッシュ関数等のアルゴリズム（広い意味での暗号アルゴリズム）が複数組合されたプロトコルです。20年以上使われているため、その間に何度か仕様の見直しが行われ、現在までに5つのバージョンが作られています。また、2000年以前は米国の暗号輸出規制のためにあえて弱い（＝解読しやすい）暗号アルゴリズムしか使えなかったこともあり、仕様上は強い（＝解読が事実上できない）暗号アルゴリズムから弱い暗号アルゴリズムまでさまざまな暗号アルゴリズムが使えるようになっています。

基本的に、プロトコルのバージョンが後になるほど、以前の攻撃に対する対策が盛り込まれ、また強力な暗号アルゴリズムも利用できるようになるなど、より安全性が高くなっています（表1）。

実際にどのプロトコルバージョンと暗号アルゴリズムを暗号通信で使うかは、サイトの確認や秘密鍵の交換をするときに、サーバとブラウザの両方が実装しているプロトコルバージョンと暗号アルゴリズムのなかから決めています。なお、利用する暗号アルゴリズムは「鍵交換_署名_暗号化_ハッシュ関数」の組で構成された暗号スイートの形で決まっています。選択された暗号スイートに記載された鍵交換、署名、暗号化、ハッシュ関数の各暗号アルゴリズムによりSSL/TLSにおける各種処理が行われますので、どの暗号スイートが選ばれるかによって使われる暗号アルゴリズムが異なり、SSL/TLSにおける安全性が変わります。

ちなみに、SSL/TLSの仕様上、どのような優先順位でプロトコルバージョンや暗号スイートを選択するかは明確には決められておらず、サーバやブラウザの設定によって異なります。このため、同じサイトにアクセスしても使うブラウザによって異なるプロトコルバージョンや暗号アルゴリズムが使われることがあります。

5.2 最近のSSL/TLSへの攻撃

ここ数年、SSL/TLSへの攻撃が相次いでいます。ここでは、有名な攻撃の概要を紹介します。

(1) BEAST 攻撃やPOODLE 攻撃など

ID/パスワードの入力を要求するサイトにおいて、ブラウザを閉じた直後にもう一度そのサイトにアクセスするとIDやパスワードを入力していないのにブラウザを閉じる前のページがそのまま表示された経験がないでしょうか。

これは、Cookieと呼ばれるデータにログイン状態などを記録してサイトとブラウザとの間で一定時間共有することにより、そのCookieを使ってアクセスしてきたブラウザに対してサイトが自動的に元のログイン状態に戻しているために起きている現象です。多くの場合、セッションIDと呼ばれる情報でログイン状態を管理しており、暗号化された状態でサイトとブラウザに記録されます。

ここ数年、相次いで公表されたBEAST 攻撃⁴⁾やPOODLE 攻撃⁵⁾などは、暗号化されたCookie（セッションID）の中身を攻撃者が知ろうとする攻撃方法です。上記のCookieの仕組みから、これらの攻撃に成功すると、ユーザUのCookie（セッションID）を攻撃者が自由に使うことができるようになり、ユーザUのID/パスワードを知らなくてもそのサイトAにユーザUに成りすまして不正アクセスできることになるからです。

これらの攻撃方法の詳細な説明は省略しますが、技術的には共通鍵暗号のなかのブロック暗号で長いメッセージの暗号化を行うモードであるCBCモードで利用したときの脆弱性を利用した攻撃とされています。攻撃方法の原理は、以下の攻撃条件がそろった時に、暗号化されたCookieについて1バイトずつ攻撃者が中身を知ることができるような攻撃方法を構築でき、それを繰り返すことでCookie全体の情報がわかるようになるというものです。ここでのポイントは、攻撃者が秘密鍵を知らなくても暗号化されたCookieの中身を知ることができるということです。

- ・攻撃対象のブラウザUにマルウェアを感染させる等により、攻撃者がブラウザUからサイトAに任意のメッセージを同じ秘密鍵で暗号化させたうえで大量に送信できる状態にできる
- ・攻撃者は、ブラウザUからサイトAに送るメッセー

表1 プロトコルバージョンの違い

		TLS1.2	TLS1.1	TLS1.0	SSL3.0	SSL2.0
	開発年数	2008	2006	1999	1995	1994
SSL/TLSへの攻撃方法に対する耐性	ダウングレード攻撃（最弱の暗号アルゴリズムを強制的に使用することができる）	安全	安全	安全	安全	脆弱
	バージョンロールバック攻撃（SSL2.0を強制的に使用することができる）	安全	安全	安全	安全	脆弱
	ブロック暗号のCBCモード利用時の脆弱性を利用した攻撃（BEAST/POODLE攻撃など）	安全	安全	パッチ適用要	脆弱	脆弱
利用できるより安全な暗号アルゴリズム	128ビットブロック暗号（AES, Camellia）	可	可	可	不可	不可
	認証付暗号利用モード（GCM, CCM）	可	不可	不可	不可	不可
	楕円曲線暗号	可	可	可	不可	不可
	SHA-2ハッシュ関数（SHA-256, SHA-384）	可	不可	不可	不可	不可

ジの中身を自由に加工できる

- ・攻撃者は、ブラウザUからサイトAに送る暗号化されたパケットをすべて傍受できる

これらの攻撃を防ぐために、CBCモードを利用したときの脆弱性が使われないようにする対策^{*5}が考案されており、これらの対策はセキュリティパッチ等の形ですでに広く配布されています。したがって、セキュリティパッチを適用することが対策となります。

また、これらの攻撃方法が成功するためには、攻撃者がブラウザUを自由にコントロールできることが必要ですので、マルウェアの感染を防止するなどの対策も有効です。

(2) RC4の脆弱性を利用した攻撃

RC4は1987年に開発された共通鍵暗号の1つです。RC4ではキーストリームという秘密の情報を生成して暗号化していきますが、そのキーストリームの出力分布に微妙な偏りがあり、大量の異なる秘密鍵で統計解析をするとある程度キーストリームの出力分布を識別できるという脆弱性があります。

この脆弱性を使った攻撃方法⁶⁾が提案されており、詳細な説明は省略しますが、攻撃原理は、異なる秘密鍵で同じ内容のメッセージを大量に送信した時に得られる暗号文を集めて統計解析すると、RC4のキーストリームの出力分布に依存する形になることを用います。そこから、キーストリームの影響を取り除くと、もともとのメッセージに復元できます。例えば、 2^{32} 個の暗号文があれば最初の257バイト分のメッセージを高確率で求めることができます。

この攻撃は大量の暗号文を必要とする点で直ちに現実的な脅威になるというわけではありませんが、防御策がないという点で、現在ではRC4を使わなくなってきたいます。

(3) Heartbleed：実装の脆弱性を利用した攻撃

2014年4月に発覚したHeartbleedは、TLS1.2の拡張機能として組込まれた、サーバの動静を簡易に確認するHeartbeatの実装上の脆弱性を利用した攻撃⁷⁾です(図13)。

SSL/TLSそのものに対する攻撃ではありませんが、

図13 Heartbleed

*5 POODLE攻撃については、強制的にSSL3.0にダウングレードさせたうえで攻撃する方法があり、SSL3.0の仕組み上、対策が困難とされています。

SSL/TLSの機能をサーバに実装するためのオープンソースライブラリーであるOpenSSL（もしくは暗号化ソフト）の致命的な脆弱性としてニュースでも大きく取り上げられたため、ご存知の方も多いかと思えます。

具体的にはOpenSSLでのHeartbeatの実装において、メモリーサイズのチェックを十分にしない脆弱性があったために、OpenSSLを使って構築したSSL/TLSサーバでは返信すべきデータに隣接するメモリー領域のデータまでも返信してしまいました。その中にパスワードやクレジットカード番号などがたまたま含まれていると、それらが漏えいしたことになります。

問題は、①OpenSSLを使っているサーバが多数あること、②この脆弱性が2年以上も対策されずにいたこと、③意図せず偶然持っていかれたデータなので漏えいした内容の特定が困難であること、④サーバ側に攻撃の痕跡が残らないため攻撃されているかどうかの判断さえも困難であること、といった点にあります。そのため、被害の全容を把握することができず、重要なデータが漏えいしたとの前提で対策が必要となりました。現在では、この脆弱性を修正したバージョンのOpenSSLがあり、対策されています。

6. むすび

SSL/TLSは、電子商取引での利用をはじめとして、インターネット社会においてなくてはならないセキュリティプロトコルです。ただ、20年以上も前に開発されたプロトコルであるうえ、相互接続性を優先させてきた背景から、さまざまな点で脆弱性があることも事実です。

今回、SSL/TLSがどのような仕組みで動いているのかを知ることで、皆さんがより安全にSSL/TLSを使えるようになっていただければ幸いです。

(2014年11月27日受付)

〔文 献〕

- 1) MD5 considered harmful today, <http://www.win.tue.nl/hashclash/rogue-ca/>
- 2) Black Tulip Update, <http://www.rijksoverheid.nl/documenten-en-publicaties/rapporten/2012/08/13/black-tulip-update.html>
- 3) Microsoft releases Security Advisory 2718704, <http://blogs.technet.com/b/msrc/archive/2012/06/03/microsoft-releases-security-advisory-2718704.aspx?Redirected=true>
- 4) BEAST, <http://vnhacker.blogspot.jp/2011/09/beast.html>
- 5) B. Möller, T. Duong, and K. Kotowicz: "This POODLE Bites: Exploiting the SSL 3.0 Fallback", <https://www.openssl.org/~bodo/ssl-poodle.pdf>
- 6) T. Isobe, T. Ohigashi, Y. Watanabe, and M. Morii: "Full Plaintext Recovery Attack on Broadcast RC4", Proc. of FSE '13, pp.179-202 (2013)
- 7) The Heartbleed Bug, <http://heartbleed.com/>

神田 雅透 1993年、東京工業大学大学院修了。同年、NTT入社。NTT研究所にて、共通鍵暗号の研究に従事し、Camelliaの設計に関わる。現在、NTTセキュリティプラットフォーム研究所主任研究員。2009年より、IPA非常勤研究員として暗号政策の調査等にも関与する。平成26年度文部科学大臣表彰、第53回前島賞等受賞。