SSL/TLS 実践入門

Webの安全性を支える暗号化技術の設計思想

ICHIHARA Hajime
市原 創
ITAKURA Hiroaki
板倉広明
［著］

技術評論社

本書は、小社刊の以下の刊行物をもとに、
大幅に加筆と修正を行い書籍化したものです。

・『WEB+DB PRESS』Vol.110特集3「最新TLS 1.3徹底解剖――通信を覗いてHTTPSの裏側を知る」

サンプルコードのダウンロード

本書で利用しているサンプルコードはWebで公開しています。詳細は本書サポートページを参照してください。補足情報や正誤情報なども掲載しています。

https://gihyo.jp/book/2024/978-4-297-14178-3/support

はじめに

インターネットが世に登場してから数十年が経過しましたが、その通信ネットワークは、またたく間に世界の隅々まで拡大し、現在では1ヵ月に数百エクサバイト[注1]という膨大な情報がやりとりされるまでに急成長しました。端末は無線通信の普及により個人の手もとまで行き渡るようになり、もはや道路や水道のようにいつも当たり前にそこにあるインフラとして利用されています。インターネットのオープンな設計思想はアプリケーションの爆発的な発展を促し、Web 2.0やクラウドコンピューティング、モバイル、IoT（*Internet of Things*）などの変革期を経て進化し続けています。インターネットは単なる通信インフラにとどまらず、政治／経済／文化の変革の舞台として重要な役割を担いつつあるようです。

筆者らは学生のころからPCとインターネットの世界に直接触れて、遊び、学び、働きながら、この進化を見てきました。黎明期のインターネットはアメリカの西部開拓時代さながらの自由で無法な空間でしたが、World Wide Webが登場し一般や商用での利用が増えてからは情報セキュリティの必要性が唱えられるようになり、SSL（*Secure Socket Layer*）とその後のTLS（*Transport Layer Security*）へとつながる暗号化通信技術の進化が始まります。当時あまり意識していませんでしたが、振り返ればそれはインターネットにおける大きな変革期でした。今ではSSL/TLSに代表される暗号化通信は、誰もが使う日常的なサービスとなりました。しかし、その裏に専門家とエンジニアによる改良の積み重ね、脆弱性や脅威との格闘の歴史があったことはあまり知られていません。

一方で暗号化の技術がどんなに進歩しても、それを使う人間の意識やアプリケーションの実装が脆弱であれば、巧みな攻撃により簡単に安全が破られることに変わりはなく、開発者に求められるリテラシはむしろ複雑化／高度化しています。筆者らは通信パケットの中身を解析することも行ってきましたが、暗号化されていない通信からアプリケーションレベルの脆弱性を発見することもありました。また、通信が暗号化されていたとしても、暗号化技術の設計思想を理解せずに実装されたアプリケーションが脆弱性を抱えているケースもありました。過去の膨大な知識の積み重ねによって支えられている暗号化技術は紛れもなく世の中に必須のインフラとなりましたが、それを

注1　テラバイトのおよそ100万倍。

正しく使いこなすためには時代に合わせた新しい技術／知識／スキルを習得することも必須な時代となっているのです。

暗号関連の業務を始めたときに筆者らがまず直面したのは、暗号をエンジニアリングの現場で活用するために必要な情報収集の難しさでした。当時も暗号理論やSSL/TLSの仕組みの解説は書籍として存在しましたが、いざ実装やものづくりを行うとなると、背景となる標準規格やRFC、その変遷に関する情報、実装しているライブラリの脆弱性や性能に関する情報、データの扱い方に関する注意事項など多岐にわたり収集しなければならず、たいへん苦労しました。そのような情報は個人の力だけで集めるのはなかなか難しく、組織的活動によって収集しなければならない場合もあります。

そこで筆者らは自らの経験に基づき、SSL/TLSの初学者が実務で最初に直面する「何から手を付ければよいのかわからない」「どんな情報を収集すべきかがわからない」「どうやって使えばよいのかわからない」といった壁を突破するための道具とスキルをエンジニアに提供することを目的として本書を執筆しました。対象読者は暗号技術やSSL/TLSを実際のサービスやアプリケーションに活用することを求められるエンジニアやプログラマーを想定しています。前述の課題意識を踏まえ、SSL/TLSを構成する暗号技術の設計思想に加え、SSL/TLSを活用する上で必要となるソフトウェアの操作／実装の手法や性能測定といった実践的知識を幅広く豊富に盛り込むことで、SSL/TLSを最終的に実務で適切に使えるようになるまで導く道しるべとなるように配慮しました。

特にUNIX/Linuxを中心とした暗号化ライブラリのデファクトスタンダードであるOpenSSLは、その圧倒的なシェアにもかかわらずAPIマニュアルが不明瞭であり、実装方法の習得にあたっては今も高い壁が存在します。本書は実際に手を動かしながら理解を深めてもらえるように、OpenSSLの利用とAPIの実装について解説します。また、SSL/TLSの最新RFCとなるTLS 1.3は仕様が整理されたことで、トレードオフの関係にある安全性と性能のバランスを柔軟に選択できるようになりました。これらを上手に使いこなすために必要となる前提知識も盛り込みました。

暗号技術や実装ライブラリ、ハードウェア、攻撃手法、脆弱性に関する情報は日々更新されており、SSL/TLSを取り巻く状況は常に変化しています。エンジニアは様々なニュースに触れながら、セキュリティに関する正解のない課題について臨機応変に判断することが求められます。未来に起きるであろう問題の本質を読み解き、どう対処すべきか適切に判断するには、過去の

知識を正しく身に付け、自ら手を動かし検証／納得しながら、その判断力を養う必要があります。本書を通して、読者の皆さんが将来の環境変化にも柔軟に対応できる実践的スキルを身に付け、今後の活躍に役立てていただければうれしく思います。

謝辞

本書の執筆にあたり、多くの方のご支援とご協力を賜りました。技術評論社の池田大樹さんには、「WEB+DB PRESS」記事書籍化の機会と、本書の構成についても多くの助言をいただきました。また、執筆が大幅に遅延したことでご心配、ご負担をおかけしたと思います。

中安淳也さんには記事執筆から本書の企画まで精力的にコーディネート・仲介していただきました。加藤晶啓さんを始めとするプロジェクト関係者の方々は多忙な中でも執筆活動を快く見守り応援・サポートしてくださりました。また、長谷川智久さんには本書のレビューをお願いし、きめ細かいフィードバックをいただいています。

皆様の物心両面でのサポートに心より感謝申し上げます。

章構成と執筆担当

本書は以下のように知識の解説と実践的な解説を交えた構成となっています。知りたい内容や試してみたいことに応じて読み分けることも可能です。

章	タイトル	著者名	内容
第1章	SSL/TLSの世界へようこそ	市原	歴史・全体像
第2章	暗号アルゴリズムと鍵	市原	知識
第3章	SSL/TLSの各プロトコル詳細——Wiresharkによる解析	板倉	知識・実践
第4章	SSL/TLSの標準規格とPKI	市原	知識・実践
第5章	OpenSSLによるSSL/TLSプログラミング入門	板倉	実践
第6章	脅威・脆弱性	市原	知識
第7章	性能の測定	板倉	実践
第8章	SSL/TLSが抱える課題と展望	市原	課題・展望

目次

第1章

SSL/TLSの世界へようこそ

情報を交換する生き物である人間は、高度な知的能力を駆使し集団や個人の間で秘密を共有できるようになりました。誰かに秘密の情報を伝えるときに、第三者が分からないように伝える手法の一つが暗号です。暗号は大昔から利用されており、その目的・動機も様々ですが、SSL/TLSもその延長線上にあります。暗号の本質的な役割を理解することはSSL/TLSの世界を紐解く手掛かりとなります。本章ではSSL/TLSを理解する上で避けて通れない重要なキーワードを集め、暗号技術とSSL/TLSの歴史、構成要素、関連知識について俯瞰的に概説します。SSL/TLSの世界は広大ですので、足を踏み入れる前に世界地図を手に入れましょう。

暗号化の役割と重要性

SSL/TLS（*Secure Socket Layer/Transport Layer Security*）を一言で表現すると「インターネット通信を暗号化する技術」です。言葉にするとシンプルですが、その実像は複雑で日頃それほど意識することもありません。ですがSSL/TLSはインターネットを使う上で必須の技術となっています。表現を変えるなら、インターネットは暗号化されていないのが前提の通信ネットワークであるためSSL/TLSによる暗号化を必要としています。昨今では当たり前のように通信の暗号化が使用されており、Webの通信にSSL/TLSが使われていることは、通信の秘密が「安全に守られている」証（あかし）である、と一般的に考えられています。多くの場合「安全である」とか「守られている」という感覚は主観的、観念的なものですが、理論的、技術的にはどのような状態を指しているのか考えてみましょう。

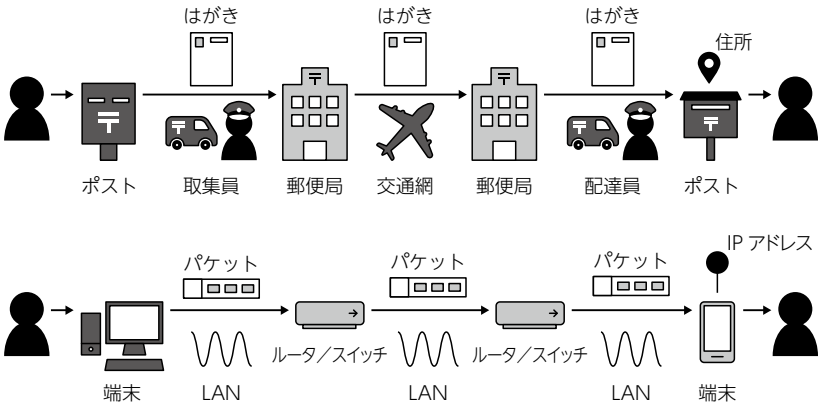
インターネットに暗号化が必要な理由

ある人が情報を遠くにいる誰かに手紙で伝えるケースを考えます。伝えたい情報を手紙に書いて送る場合、通常は郵便という社会インフラを使って輸送します。手紙は封筒に入れて封をして送るので、輸送される途中で中身を誰かに見られる心配はありません。では郵便で送れる郵便物が「はがき」だけに限定されているとしたらどうでしょうか？ はがきは1枚の紙の表面に文面が記されているだけなので、誰かに見られる可能性があります。自宅のポストに投函された後も他の家族や同居人に見られる可能性があります。確率と

しては低いですが最悪のケースを考えるなら、はがきを情報もろとも悪意のある第三者に盗まれて、届けたい相手にすら届かないこともあるかもしれません。

インターネットは、この「はがき郵便」に似ています（**図1.1**）。インターネットを構成する基本的な通信プロトコル（規約）であるIP（*Internet Protocol*）は、パケットと呼ばれるデータのまとまりを、ある機器からある機器に届けるための仕組みです。インターネットにつながる機器は全てIPアドレスという固有の情報を持っており、ネットワークはパケットをバケツリレーのように受け渡していくことで正確に届けます。このリレーはルーティングという仕組みで実現されており、その役割を担うのがルーターやスイッチといった中継機器[注1]です。パケットをはがき、ルーターやスイッチを郵便局や取集員、配達員と考えるなら、インターネットにもはがき郵便と同じようなリスクがあることは容易に想像できます。

図1.1　はがき郵便とインターネット



インターネットを流れる情報はデジタルデータですので、いとも簡単にコピーできます。どこかの記憶媒体に保存することもできますし、書き換えることもできるでしょう。インターネットで情報が届けられる間に経由するルーターやスイッチは無数にありますが、その中のどれか一つでも悪意のある人物のコントロール下にあれば、情報を覗かれたり、書き換えられたり、

注1　ネットワークの用語では「ミドルボックス」とも呼ばれます。

保存されたりするリスクがあります。実際、初期のインターネットはこういったネットワークだったのです。それでも活発に使われていたのは、ネットワーク上に悪意を持った機器が接続されていることは非常に少ないという性善説と、そもそも誰かに読まれて困る情報は扱っていない、というオープン性に支えられていたのかもしれません。

ご存じのように現在のインターネットを飛び交う情報は、取引の決済情報や購買履歴、閲覧履歴といった個人情報、企業や組織の機密情報、人命が関わるような「守るべき」データとなり、その量も昔とは比較にならないくらい爆発的に増えました。「はがき郵便」のように比較的オープンでハイリスクな仕組みのインターネットで、そのように重要な情報が流通できるようになったのは、SSL/TLSに代表される暗号技術のイノベーションがあったからに他なりません。暗号技術なくして現在のインターネットは成り立たないと言っても過言ではないでしょう。

SSL/TLSの持つ機能は主に次の3つの性質に代表されます。第一は「機密性」です。はがき郵便でたとえるなら、文面を「暗号」に書き換えて第三者には簡単に読めないようにして情報を秘匿することです。第二は「真正性」です。これは、情報のやり取りをしている相手が本当に正しい相手なのかを互いに認証することを意味します。第三は「完全性」です。デジタルデータであるインターネットのパケットはいとも簡単に書き換えることができるので、情報が書き換えられずにありのまま伝わっていることを証明する必要があります。これについてはハッシュ関数という技術を応用しますが、次章で詳しく解説します。本章では第一と第二の性質についてもう少し紐解いてみます。

情報の秘匿

情報量

暗号により「情報を秘匿する」ことは理論的に説明できるのでしょうか？それを最初に試みたのが「情報理論」の始祖と言われるクロード・シャノン（*Claude Elwood Shannon*）です[注2]。暗号について考えるにはまず「情報」というものを定量的に定義して捉える必要があります。ここで、情報理論における「情報量」という概念に少し触れます。ある人が誰かに情報を伝えたときのことを考えます。情報の送り手は「送信者」、受け手は「受信者」とします。送信者が受信者に「今日の天気は晴れです。」という情報を伝えたとすると、

注2　秘匿システムの通信理論（Communication Theory of Secrecy Systems）。

受信者の受けた情報の量はどれくらいなのでしょうか。

受信者が送信者と同じ場所にいて受信者が外の天気の様子を既に知っていた場合、何も新しい情報は得られていませんから情報量はゼロになります。逆に受信者が外の様子が見えない部屋にずっと閉じ込められていて外の天気について一切知らなかった場合、情報量は多いでしょう。では「部屋に閉じ込められているものの、今の季節を知っている受信者」に同じ情報が伝えられたとしたらどうでしょうか？ この場合は真夏より梅雨に晴れであることを伝えられた方が「情報量が多い」と感じます。情報理論ではこういった感覚を数学的に表現しており、確率が低い事象であればあるほど情報量が多いと考えます。

通信における情報量は、通信前の受信者の知識の「不確かさ」が前提として存在し、それを通信によって**どれだけ減らしたのか**を、通信の正確さを測る客観的尺度と考えることができます[注3]。通信前の状態を「不確かさ」として情報量の数値で表現し、通信後に全ての情報を知っている状態は「不確かさ」を意味する情報量がゼロとなるのです。ある事象の情報量は事象が出現する確率から表現され、複数の事象については次の式によって表現します。これを「平均情報量」（エントロピー）と呼びます。

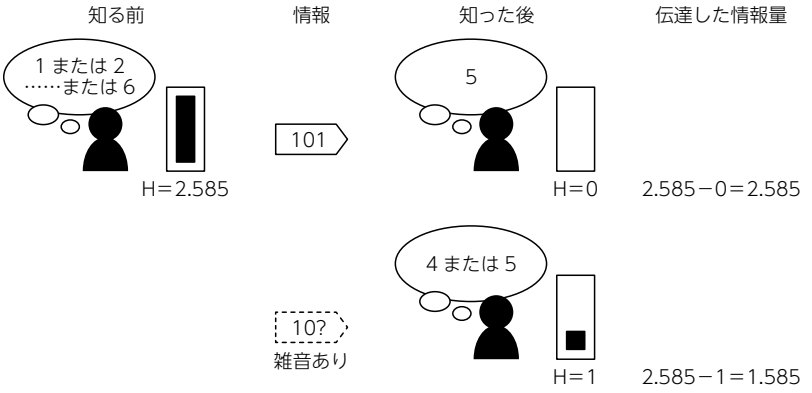
$$H = -\sum_{i=1}^{n} p_i \log p_i$$

p_iは事象の確率を意味する0〜1の実数であり、確率の低い事象は情報量が多いことが分かります。例えば通信で伝える情報がサイコロを振って出た数（1〜6）だけであり、2進数で符号化されているという場合を考えてみましょう。2進数の情報は、対数の底を2とすることで情報量を計算します。サイコロのそれぞれの数は6分の1の確率で出現するので、その平均情報量Hは2.585ビットとなります。これは、情報の受信者は情報を受信する前に2.585ビットのエントロピーすなわち「不確かさ」を持っている、と言うことができます。ここで通信により5（101）という数字が伝わると、受信者のエントロピーは0ビットとなり、正確に伝えられたことになります。では、なんらかの通信路の雑音のせいで、受信者は最後のビットが0か1か分からなくなったとしたらどうでしょうか？ 取りうる値は4（100）か5（101）のいず

注3 あくまで客観的な数値化を試みたものであり、主観的な情報の価値を意味しません。

れかであり、確率は1/2です。エントロピーは1ビットに下がりますが「不確かさ」が残ってしまいました（**図1.2**）。

図1.2　サイコロの目と情報量



情報が正確に伝わると受信者の「不確かさ」すなわちエントロピーが減りますが、もし通信路を流れる情報に雑音が入れば受信者のエントロピーは十分に下がらず、情報は正確に伝わりません。シャノンは暗号の機能を通信における雑音（ノイズ）に近いものとして捉え、情報を暗号化することは情報に意図的な雑音を入れてエントロピーを維持するようなものだと考えました。もちろん、ただ雑音を入れるだけでは、完全に情報が失われて誰にも伝わりません。情報を伝えたい相手はこの雑音を取り除いて理解できることが、暗号として成立する条件になります。

自然言語と暗号

そのような雑音を作ることが暗号の機能の本質ですが、もっと簡単に考えることもできます。ある人にとっては雑音にしか見えない情報が、ある人にとっては規則性のある符号に見えるならどうでしょうか。この最も卑近な例は自然言語でしょう。

情報の送信者と受信者が共通の言葉を共有していて、盗聴者がその十分な知識を持っていない場合を考えてみましょう。「晴れ」「雨」といった単語は日本語ですが、もし盗聴者が英語圏の人でこれらの日本語を知らなければ、この情報を覗いたとしてもただの雑音に見えるかもしれません。「晴れ」とい

う言葉が「*Sunny*」と同じ意味であることの確率は日本語と英語の両方を知る受信者にとってはほぼ100%ですが、日本語を知らない盗聴者にとっては「あらゆる言葉の中の一つ」であることしか分からず、数万分の1くらいに小さい確率となります。これは、言語という共有知識を利用して盗聴者から情報を秘匿していることに近いのです。実際、歴史上の古代言語は現代人にとってはほぼ暗号に近く、解読には多くの労力を必要とします。戦時中には日本が薩摩弁、米国がナバホ語の難解さを利用して、実際に暗号として使っています[注4]。

このように情報理論で考えると、暗号による情報の秘匿性は確率的な事象として説明することが可能です。このことから、必ずしも完全な秘匿性がなくともエントロピーを最大化して解読される確率を十分に小さくできる方法があれば、それは暗号として機能することが分かります。例えば送信者と受信者しか知らない全く新しい信号（情報の表現方法・読み解き方）や雑音の入れ方、取り除き方を定義して事前に共有すれば、盗聴者に対して情報を秘匿できる確率は比較的高いと言えます。また、そこで定義した事前知識は暗号を解くための「鍵」と捉えることもできます。

現代では暗号の理論的研究が進み、解読するための方法論や計算リソースも大変豊富なため、解読されない暗号を作るのは至難の業です。素人が思い付きで考えた暗号はすぐに解読される可能性が高く、まず使い物になりません。暗号に必要とされる強度は盗聴者の技術レベルとのいたちごっこなので、より高度な計算でしか実現できない時代になったのです。

認証

通信先の真正性

次にSSL/TLSの際立った特徴である認証の機能について、その基本的な概念を整理します。先ほどのはがき郵便の例で考えてみましょう。はがきは住所と宛名と郵便番号により、その送付先が決まります。はがきを受け取った人は差出人欄を見れば送り主が誰であるかが分かります。しかし、それは本当に正しいでしょうか？ もしかすると全く関係ない第三者がなりすまして差出人を名乗っている可能性も考えられます。差出人が本当に正しい相手であることを確認するために必要な性質が「真正性」です。はがきの場合は必要に

注4 少数言語を話す暗号兵は「コードトーカー」と呼ばれており、専門部隊が結成されていました。

応じて筆跡鑑定や本人しか持っていない印章の印影などの証拠をもって真正性を確認することになります。

インターネットの情報はデジタルデータですからさらに厄介です。デジタルデータには筆跡鑑定や印影とは違って個人を特定する固有性がないので、いくらでも他人になりすますことが可能です。しかも、世界中の顔の見えない会ったこともない相手と通信をするのです。あなたがAmazonで買い物をするときにアクセスしているWebサイトはAmazon.comが提供しているサービスで間違いない、ということはどうすれば確認できるのでしょうか？

真正性を確認する方法

真正性は実は暗号でも確認できます。まず秘密の暗号メッセージを送信したい送信者が、メッセージの受信者と事前に暗号解読に必要な鍵の情報を共有している場合を考えます。その鍵は事前に鍵を渡した受信者だけが知っているとするなら、その暗号メッセージは受信者だけが解読できるため、正しい相手だけに「伝わる」と考えられます。受信者の視点から見た送信者の真正性も同様です。送信者と受信者以外に知りえない情報を使っていることで結果的に真正性の確認ができるのです[注5]。

ただし、この考え方が通用するのは、送信者と受信者がお互いのことを信頼していて、あらかじめ共有された暗号や鍵の情報が漏れない、相手が嘘をつかないという前提に基づいています。実際には、暗号や鍵の情報を共有し漏らさないように管理するのは容易ではありません。備忘のために情報の記録が残されてしまうと、何かのきっかけで第三者がその記録を覗いてしまう恐れがあります。送信者と受信者の間で交わされた守秘の約束も本当に守られるのか、いつまで守られるのかといった不安が残ります。通信した結果について相手が「受け取っていない」と嘘をつく「否認」のリスクもあります。インターネットはコンピュータ端末同士の相互通信であり、通信相手の素性も分からないことは日常茶飯事です。通信中の当事者が互いの真正性を確認する以前に、暗号や鍵の情報共有をするためには素性の分からない相手を信頼しないといけないという根本的な問題があるのです。

第三者による真正性の保証

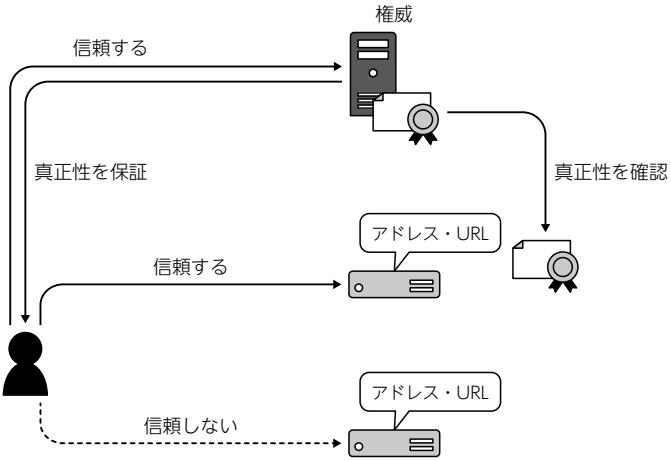
私達が不特定多数のWebサーバにアクセスし個人情報を入力するのは、そ

注5　二者間で共有する情報に基づく相互認証は英語では*Authentication*と言います。

の通信相手が「何者か」を理解し、その通りの正しい相手であると信頼しているからに他なりません。インターネット上で初めて通信する相手やサーバはアドレスやURLを持っており、当然のように自分が何者であるかを表明しているはずです。しかし、それを鵜呑みにするわけにはいきません。なぜなら大手Webサイトの名をかたる偽者である可能性もあるからです。こういった場合はなんらかの「権威」によって信頼できる第三者から通信相手の真正性を保証してもらうことが素直な解決策となるでしょう[注6][注7]。

SSL/TLSは**図1.3**のように権威ある第三者により真正性を保証する仕組みを採用しています。具体的にはPKI（*Public Key Infrastructure*）と呼ばれる信頼のネットワークを使って、通信当事者以外の「信頼できる第三者」が互いの真正性を保証するのです。インターネットにおける「認証」の実現に必要な要素を並べると、次の3つになることが分かります。正しい本人であることを証明する「真正性」、第三者が真正性の保証を与える「権威」、そしてその権威と真正性に対する「信頼」です。SSL/TLSはこれらを暗号と数学的計算を駆使して技術的に実現しています。

図1.3 権威による認証



注6 第三者により通信相手の真正性を証明してもらう認証は英語では*Certification*と言います。

注7 ブロックチェーンのように第三者の権威ではなく「仕組み」を信頼するという考え方・手法もあります。これについては第8章で簡単に紹介します。

SSL/TLSの歴史

暗号は非常に長い歴史を持っており、現代で使われている暗号技術も大昔の手法や理論の上に構築され、発展してきました。SSL/TLSはそういった暗号技術の塊であり、インターネット時代に花咲いた暗号技術の集大成と言ってもよいでしょう。暗号史を通してSSL/TLSがどのようにして誕生したのかを振り返ってみましょう。

暗号技術の成り立ち

通信文を秘匿する技術

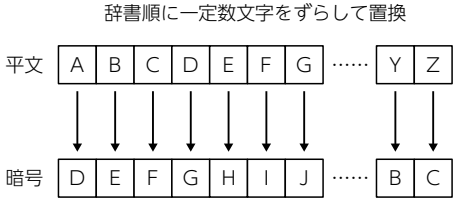
通信文を秘匿するという技術的課題の歴史は古く、これまでに数多くの技術が開発されてきました。古代にはメッセージが存在すること自体を巧みに隠す「ステガノグラフィ」がよく使われていました。ステガノグラフィは、あるテキストや絵画、画像などの中に、本当に伝えたいメッセージを見つからないように紛れさせて隠蔽する手法です。果物の搾り汁で紙に書いた文字を火であぶると見えるようになる「あぶりだし」を子供の頃にやったことがあるかもしれませんが、これも典型的なステガノグラフィです。ステガノグラフィは現代でも電子透かしやGPSなどの技術で活用されており、技術的にも進化しています。ステガノグラフィはその存在が気付かれてしまうとメッセージが全てバレてしまうという根本的な弱点があるため、現代では秘密を守る目的のために単体で使われるケースは少ないようです。

ステガノグラフィはメッセージの存在自体を隠すことが目的ですが、それに対しメッセージの意味を隠すことを目的に使われる技術が暗号（クリプトグラフィ）です。前述のように世の中に全く知られていない言語体系を用いればそれ自体が暗号となりえますが、一から新しい言語を作って受信者と共有することは大変な労力を必要とするので現実的ではありません。そのため常用言語のテキストをあるルールに基づいて解読困難な文に変換する暗号技術が多く用いられてきました。暗号の世界では暗号に変換する前のメッセージを「平文」、平文を暗号文に変換する処理のことを「暗号化」、暗号文を再び平文に戻す処理のことを「復号」と呼びます。暗号はたとえ暗号文が第三者の手に渡ったとしても、そのメッセージの意味を読み取れない、また簡単には復号できないようにすることで秘密を守るのが目的です。

シーザー暗号

暗号もステガノグラフィと同様に古くから使われてきました。史実上の最も有名な古典暗号と言えば「シーザー暗号」ではないでしょうか。シーザー暗号は紀元前100年頃の共和制ローマの政治家、ユリウス・カエサルが使ったとされる暗号です[注8]。シーザー暗号の手法は、文字を辞書の順番に数文字分（前または後ろに）ずらして書き直すという極めてシンプルなものです（**図1.4**）。これは容易に解読できるため現代では全く使用されませんが、暗号化の手法・規則（アルゴリズム）と何文字分ずらすかという「鍵」に相当する情報を受信者との間で共有している点において、現代の暗号の原型とも呼べる要素が含まれています。

図1.4　シーザー暗号



置換表による換字式暗号

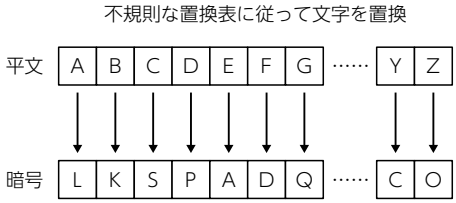
シーザー暗号は高々アルファベットの26文字分の鍵パターンしか存在しないため、解読者は26回分のパターンを試すだけで極めて容易に解読できます。そのため「辞書順」のような前提を置かずに不規則に文字同士の関係を定義する置換表を使って変換する換字式暗号が使われるようになります（**図1.5**）。この手法であれば、換字のパターンはアルファベットなら26の階乗[注9]分だけ存在することになり、途端に解読が困難になります。この手法は何世紀もの間、解読は不可能と考えられていましたが、これは後に、置き換えた文字が文章の中でどれくらいの頻度で現れるかを分析する「頻度分析」という手法で破られることになります。アルファベットを使う言語では、文章の中に特定の文字が現れる頻度に大きな偏りがあるため、一定以上の長さのある暗号文の中に現れる文字の頻度を統計的に分析すると非常に高い確率で平文の文字

注8　「シーザー」は「*Caesar*（カエサル）」の英語読みで、シーザー暗号は「カエサル式暗号」と呼ばれることもあります。

注9　403291461126605635584000000通り。

を推定することができるのです。

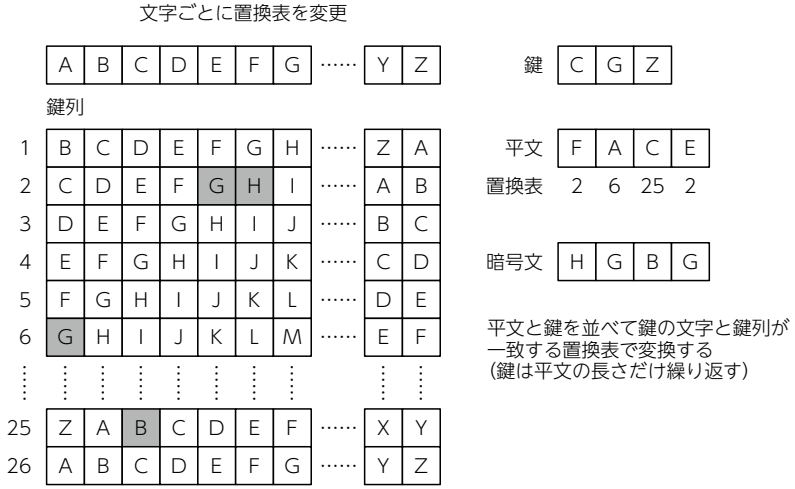
図1.5　換字式暗号



ヴィジュネル暗号

その後、頻度分析による解読に対抗するために発明された「ヴィジュネル暗号」と呼ばれる暗号では、置換するアルファベットの一覧（暗号アルファベット）をたくさん用意して、1文字ずつ暗号化する度に使用する一覧を変更するという手法を使います（**図1.6**）。この方式であれば暗号文のある1文字に対応するアルファベットは毎回異なるため、出現頻度が攪乱されて統計的な分析が困難になります。この暗号は解読不能と言われるほど強力でしたが、暗号化と復号に必要な手間が多く煩雑であったため、しばらく活用されることはありませんでした。近代になり電信などの通信技術が発達し、国民国家や企業間の情報戦が激しくなってくると、より安全な暗号としてヴィジュネル暗号が見直されるようになります。ヴィジュネル暗号はあるキーワードを鍵として共有することで暗号アルファベットの切り替えを行いますが、このキーワード自体は変化しないので、暗号アルファベットの一覧表とキーワードの長さを比べて分解すると従来の頻度分析による解読が可能であることが19世紀に発見されてしまいます。それまで解読不能とされた暗号が破られたことでさらに強力な暗号が求められることになりました。

図1.6 ヴィジュネル暗号

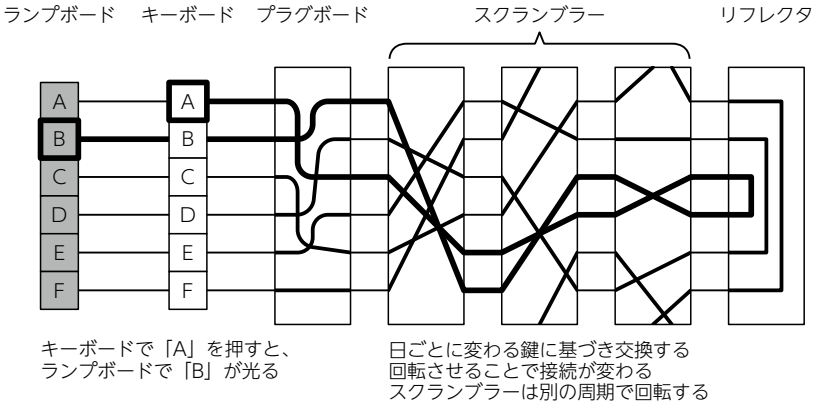


エニグマ暗号

近代以降は暗号の作成と解読にスピードが求められるようになり、主に機械が用いられるようになります。歴史上の機械式暗号装置としては第二次世界大戦中にドイツ軍が使用したエニグマ暗号機がよく知られています。エニグマ暗号機は換字式暗号ですが、ヴィジュネル暗号の考え方を引き継ぎ、アルファベットの対応関係を定義する換字表が機械的に1文字単位で変更されます。換字表は電気的な結線の変更という仕組みで実現されており、スクランブラーと呼ばれる円盤が回転することで換字の対応関係が1文字単位で変更されます。スクランブラーは3枚連結しており、別の周期で回転します。さらに入力文字とスクランブラーの間の結線を決定するプラグボードと呼ばれる部品もありました（**図1.7**）。

重要なのはプラグボードの配線とスクランブラーは変更可能であることです。これはドイツ軍で共有する鍵表（コードブック）に従って毎日変更されます。これによりヴィジュネル暗号の弱点であった鍵の周期性の問題が克服されており、暗号解読者は1京にも及ぶ鍵パターンを検証しなければなりませんでした。しかも仮に日が暮れた頃に解読に成功したとしても、情報の鮮度は低く、翌日には全く新しい暗号が現れるのです。

図1.7　エニグマ暗号機の仕組み



当時、イギリスは一堂に集めた優秀な数学者や暗号解読者により毎日新しい暗号の解読に取り組んでいました。その中の一人だった天才数学者アラン・チューリングはエニグマ暗号の弱点を発見し、解読の効率化に成功しますが、それでも解読に膨大な人手を必要としていました。そこでチューリングは、膨大な組み合わせの中から電気的に試行錯誤を繰り返し、自動で鍵を発見する機械（*Bombe*）を実用化し、解読に必要な時間を劇的に短縮することに成功します。チューリングはかねてよりプログラム可能で自動的な計算機である「チューリングマシン」という現代のコンピュータの原型とも言える仮想機械を提唱していました。当時最強と言われた機械式暗号が、数学者であり計算機の始祖であるチューリングの手で解読されたことは象徴的な出来事でした。その後、イギリスは*Colossus*と呼ばれる電子的な暗号解読機を開発します。*Bombe*も*Colossus*も極秘扱いだったため1970年代まで詳細が明らかにされることはありませんでした[注10]。戦後は電子計算機と数学理論が暗号の主戦場となっていきます。

暗号の理論化とデジタル化

暗号に関する理論的分析が進み、数学的に議論されるようになったのは、20世紀になってからです。暗号の数学理論としての体系を論じ、その基礎を

注10　一般的に「世界初のコンピュータ」は米国のENIACであると言われていましたが、現在は該当する計算機がいくつもあるとされており、Colossusもその中の一つとして数えられています。

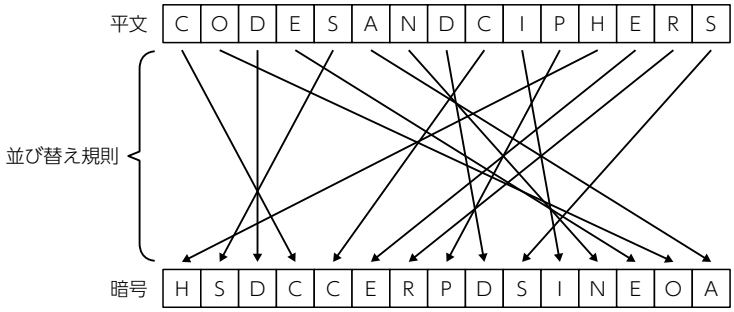
築いたのは前述のシャノンと言われています。シャノンは情報理論や符号理論を駆使し、暗号の理論的な定義を試みました。暗号は送信者と受信者で鍵情報を共有し、盗聴者から見た情報のエントロピーを増大させることで暗号解読を難しくします。暗号解読の難しさは、暗号の強度、あるいは安全性とほぼ同義です。暗号の理論的安全性については主に2つの考え方があります。情報理論的安全性と計算量的安全性です。情報理論的安全性とは、完全秘匿系と呼ばれる仕組みで実現される暗号で、理論的に解読不可能とする暗号の考え方です[注11]。情報理論的安全性を実現した暗号技術は実用化されていますが、インターネットのような通信に使うのは技術的ハードルが高く、普及はこれからという状況です。それに対し計算量的安全性とは、暗号を解読する計算機の計算資源は有限であるという前提のもとで、数学的モデルにより暗号のエントロピーを増大させ、情報を守る考え方です。SSL/TLSで使われている暗号は数学理論に基づいており、計算量的安全性をその理論的根拠としています。

ところで、古典的暗号手法としては「換字式」の他に「転置式」という方式があります。転置式暗号では、**図1.8**のように暗号化前の平文の文字をある規則に沿って並び替えることで暗号文を作成します。これは文字の順番をあみだくじで入れ替えるのをイメージすると分かりやすいです。文字の並び替え規則とその組み合わせパターンが文字数の増加に従い指数関数的な増え方をするため、長い平文ほどより複雑で安全な暗号を作ることができます。ただ、一定以上の長さの文章になると、その転置パターンは天文学的な数になり、送信者の暗号文作成と受信者の解読に必要な労力も平文が長くなればなるほど膨大になるという欠点があります。暗号手法に求められる要件として「解読が難しいこと」は重要ですが、「暗号作成と解読に必要な労力や時間がその目的に見合うものであるかどうか」も重要です（これは現代の計算機による暗号であれば性能問題と考えることができます）。そのため、転置式を使う場合は平文を一定の文字数の「ブロック」に区切り、その範囲内で転置する「ブロック暗号」という手法を用いることが現実的です。この手法は文字の脱落や重複が一つでもあると途端に解読できなくなるという欠点があり、昔はあまり使われていませんでした。しかし、計算機が登場し、デジタル信号による厳密な情報伝達が可能になると転置式は見直され、換字式と組み合わされることで強力な暗号を生み出し、SSL/TLSの中核的暗号技術の役割を担う

注11 情報理論的安全性の考え方は第2章でも触れます。

ようになります。これらは「暗号アルゴリズム」と呼ばれ、現在に至るまでにいくつもの手法が開発され、SSL/TLSに採用されています。

図1.8　転置式暗号



暗号技術の進化と新たな問題

ここまでを振り返ると、歴史上の暗号技術はその系譜を引き継ぎながら発展してきたため、基本的な仕組みに共通点があることが分かります。すなわち、鍵に相当する秘密の情報を送信者と受信者の間で共有し、ある決まった規則（暗号アルゴリズム）に基づいて暗号化、復号を行うという考え方です。このアイデアのメリットは2つあります。1つ目は二者間で共有すべき知識を最小限に抑えることができることです。変換の規則である「暗号アルゴリズム」に従うだけで簡単に暗号に変換できますし、第三者に知られぬように秘匿すべき情報は鍵に集約され、二者間で守るべき秘密の管理コストを抑えることができるのです。2つ目は暗号アルゴリズムを広く公開できることです。アルゴリズムを公開すれば、数多くの専門家がアルゴリズムに問題がないかを検証し、問題が見つかればさらに改善されることも期待できます。より多くの専門家の検証に耐え抜いたアルゴリズムこそが最も強力な暗号アルゴリズムなのです。

この考え方は現代では数学理論や計算機技術と融合し、「共通鍵暗号」として実を結びました。しかし、その後ARPANET[注12]などの全く新しい通信技術が登場すると、暗号化通信を共通鍵暗号だけで運用するのは限界があるこ

注12　米国防総省のARPA（高等研究計画局）が開発した軍事用コンピュータの通信ネットワークです。その後、政府機関や学術機関と接続され、その技術は改良されてインターネットに引き継がれます。

とが強く意識されるようになります。まず、不特定多数の送信者と受信者の間でどうやって相手を信頼するのかという「認証」の問題がありました。そしてもう一つがネットワーク上で初めてつながる相手とどうやって鍵を共有するのかという「鍵配送」の問題です。特に後者の問題はピアツーピア[注13]で自由に接続し通信できるARPANETの通信プロトコル（現在の*Internet Protocol*の起源に相当）において致命的な問題だったと言えます。SSL/TLSが世に現れるためには、これらの問題の解決を待たなければなりませんでした。

SSLの誕生

公開鍵暗号の発明

20世紀後半にコンピュータが小型化し、通信で活躍するようになると、それに呼応するように暗号技術も飛躍的な進歩を遂げます。その中で暗号技術がSSL/TLSへと進化を遂げるために欠かせない革命的な暗号が発明されました。それが「公開鍵暗号」です。公開鍵暗号は歴史上の暗号学者を悩ませ続けた「鍵配送」の問題を画期的な方法で見事に解決します。また、公開鍵暗号を応用した「デジタル署名」[注14]という派生技術も生み出し、それをPKIに応用したことで「認証」の問題も解決します。公開鍵暗号の発明がなければ、その後のSSL/TLSの誕生もなかったことでしょう。公開鍵暗号の概念を最初に提唱したのは、米スタンフォード大学のホイットフィールド・ディフィー（*Diffie*）とマーティン・ヘルマン（*Hellman*）だと言われています[注15]。彼らが後に考案した鍵配送のためのプロトコルは「ディフィー・ヘルマン（DH）鍵交換（鍵共有）」と呼ばれています。

共通鍵暗号は通信したい二者が1つの秘密の鍵を共有します。暗号化と復号の処理が対称となっているので「対称鍵暗号方式」とも呼ばれます。これに対しDH鍵交換が画期的だったのは、鍵に相当する情報を「秘密値」と「公開値」に分離し、「公開値」だけを交換することで、互いに同じ値を計算できる

注13 通信ネットワークに接続する機器同士が特定のサーバを介さずに対等につながることのできるアーキテクチャ。

注14 本書では文脈により誤解のない場合は「デジタル署名」を省略形の「署名」と表現する場合があります。

注15 ディフィーとヘルマンの発表より数年前には英国政府通信本部（GCHQ）のジェイムズ・エリス（*James H. Ellis*）が公開鍵暗号の概念を考案していましたが、極秘扱いだったためその事実は1997年まで公開されませんでした。

ことです。ここで秘密値と公開値は異なる使い方をする「非対称な鍵」と考えることができます。

DH鍵交換以後に発明された公開鍵暗号技術は、暗号化・復号、署名・検証といった機能も生み出しました。その機能は非対称な鍵を使うことで実現されるため「非対称鍵暗号方式」とも呼ばれます。これらのアルゴリズムは、非対称な仕組みを実現するために「一方向性関数」という数学的な性質を持つ計算を利用します。一方向性関数とは関数そのものの計算は容易ですが、逆関数の計算が非常に難しいという性質を持つ関数です。暗号に活用できる一方向性関数は現在ではいくつか発見されていますが、代表的な関数としてあげられるのが素数の因数分解の問題を応用した関数です。これを実用化したのがマサチューセッツ工科大学のロナルド・リヴェスト（*Ronald Rivest*）、アディ・シャミア（*Adi Shamir*）、レオナルド・エーデルマン（*Leonard Adleman*）の3名でした[注16]。この関数を応用した公開鍵暗号は3名の頭文字を取って「RSA」と名付けられます。

RSAでは、不特定多数に配ることができる鍵である「公開鍵」と、所有者以外の誰にも知られてはならない鍵である「秘密鍵」のペアを作ります。RSAの仕組みを使った暗号である「RSA暗号」では、送信者は受信者の公開鍵を使ってメッセージを暗号化し、受信者に送信します。この暗号を復号するためには受信者が所有する秘密鍵を必要とします。公開鍵で復号することも理論上不可能ではありませんが、一方向性関数の性質により、膨大で非現実的な計算をこなさなければ復号することはできません。これは「計算量的安全性」により秘密が守られていることを意味します。

COLUMN

公開鍵暗号のたとえ

公開鍵暗号の仕組みには高度な数学が使われており、直観的に理解することが難しかったため、一般的な知識と符合しやすい「たとえ」で表現されることもありました。南京錠の付いた箱のたとえもその一つです。

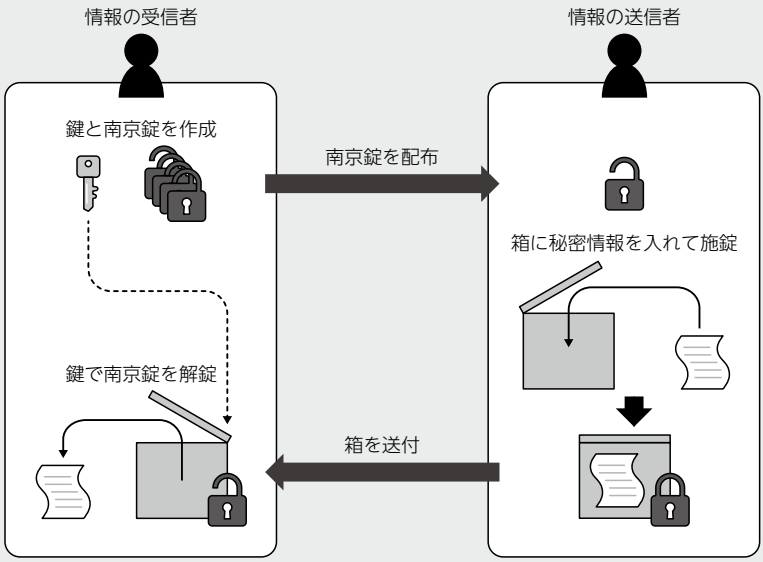
秘密の文章を書いた紙を南京錠で施錠できる箱に入れて相手に送るケースを考えます（**図1.9**）。まず受信者は自分の鍵で開く大量の南京錠を作成

注16 この点においてもGCHQのクリフォード・コックスとマルコム・ウィリアムソンの2人がRSAに先立ち素数の因数分解の問題を応用することを発見していましたが、やはり公開されませんでした。

し、不特定多数の人に配ります。秘密のメッセージを受信者に送りたいある送信者は、メッセージを入れた箱の蓋を閉め、受信者から受け取った南京錠で施錠し、箱を受信者に送ります。このときこの南京錠の施錠は簡単にできますが、一度施錠したら送信者にも開くことはできなくなりますし、受信者以外の第三者が解錠して中身を見ることもできなくなります。受信者だけが自分の鍵を使って秘密のメッセージを読むことができるのです。このたとえは「施錠するのは誰でも簡単にできるが、解錠は鍵を持っている人にしかできない」という性質を利用することにより、秘密の鍵を互いに共有しなくても秘密の情報をやり取りできることを示唆しています。

これは、公開鍵暗号技術で実現できる「暗号化・復号」の手法について身近な例で分かりやすくした「たとえ」であるため、公開鍵暗号技術全般を理解できるものではありません。過去には鍵の共有に「暗号化・復号」を使用するのが一般的であったため、多くの解説でこのたとえが普及した背景がありますが、現在のSSL/TLSにおいては「暗号化・復号」よりも「鍵交換」や「署名・検証」などの方が主流であり、「暗号化・復号」はほとんど利用されていません。技術によって実現できる手法の一部をあえて単純化して表現したアナロジーとして理解しましょう。

図1.9 公開鍵暗号のたとえ



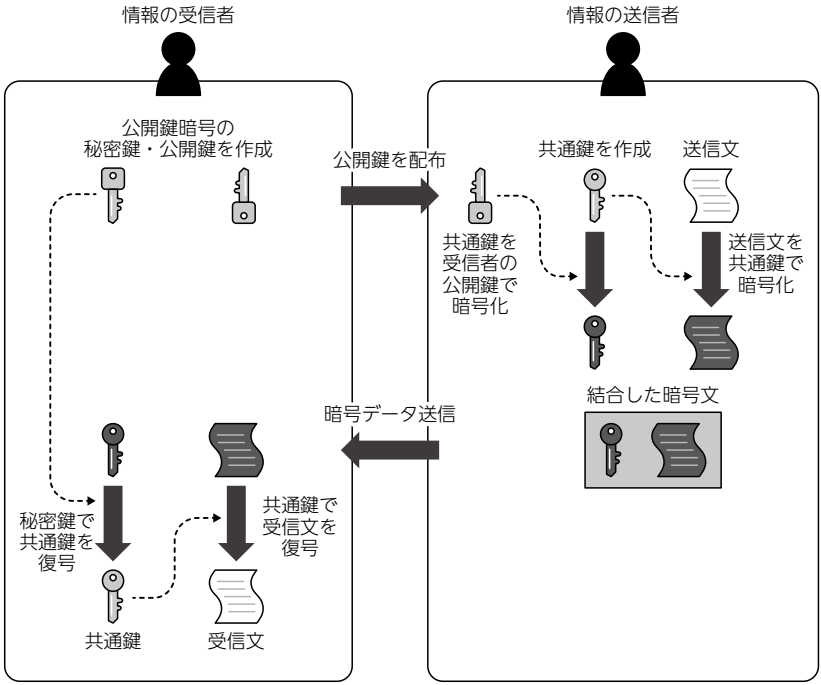
ハイブリッド暗号システム

公開鍵暗号の発明は画期的でしたが、共通鍵暗号に比べると多大な計算資源を必要とする暗号でした。したがって、全ての平文を公開鍵暗号で暗号化することは現実的ではありません。そこで考え出されたのが共通鍵暗号と公開鍵暗号を組み合わせた「ハイブリッド暗号システム」です（**図1.10**）。簡単に言えば、秘密にしたい長い平文の暗号化は従来通りの共通鍵暗号で行い、その鍵を相手と共有するために公開鍵暗号を使って暗号化すればよいのです。送信者は共通鍵暗号で共有すべき鍵を「平文」とみなし、別途手に入れた受信者の公開鍵を使いRSAで暗号化した上で受信者に送信します。受信者はこの暗号を自分の秘密鍵で復号すれば共通の鍵を手に入れることができるので、その後は通常の共通鍵暗号による暗号化通信を行うことができます。公開鍵暗号は積年の課題であった共通鍵暗号の鍵配送問題を解決したのです。

この仕組みを最初に実用化したのはPGP（*Pretty Good Privacy*）です。PGPは米国のシステムエンジニア、フィル・ジマーマン（*Phil Zimmermann*）が開発した暗号化ソフトウェアです。ハイブリッド暗号の考え方はその後の一時期の暗号化通信のスタンダードとなり、SSL/TLSの設計思想にも引き継がれることになります[注17]。なお、PGPは公開鍵暗号を使った「署名」技術とPKIの仕組みを取り入れることで認証の問題も解決しています。公開鍵暗号や署名については第2章、PKIについては第4章で詳しく解説します。

注17 狭義の「ハイブリッド暗号」は現在のSSL/TLSではほとんど利用されておらず、鍵交換が主流となっています。

図1.10 ハイブリッド暗号システム



Web通信の暗号化

さて、ARPANETがインターネットとなって世界中に広がるきっかけとなったのが、HTTP（*Hyper Text Transfer Protocol*）通信により実現したWorld Wide Webでした（本書では省略形の「Web」と表記します）。20世紀末からWebの通信が爆発的に拡大すると、必然的に商取引情報や個人情報などのセンシティブな情報が流通するケースが増えていきました。それに伴い悪意あるユーザによるサーバの偽装リスクや、通信データが盗み見られるリスクも徐々に増大します。Webの健全な発展を促すためには、インターネットの暗号化通信を早期に実現することが、避けて通れない喫緊の課題となったのです。この課題に真っ先に取り組んだのが当時*Netscape Navigator*というブラウザで一世を風靡したNetscape社[注18]であったことは自然な流れでした。

注18 *Netscape Communications Corp.*

Netscape社は*Netscape Navigator*用の暗号化通信機能を想定したSSL（*Secure Socket Layer*）を設計します。SSLはPGPや先行して米NSA（*National Security Agency*）により研究されていたSDNS（*Secure Data Network System*）の設計思想を引き継いで開発されました。最初に設計されたSSL 1.0はリプレイ攻撃や改ざんに対する脆弱性があったため公表されることはありませんでした。その後すぐに再設計を行い、SSL 2.0としてバージョンアップして1995年に公表され、初めてNetscape Navigatorに実装し配布されます。ここからSSLはインターネットの暗号化プロトコルのデファクトスタンダードの道を歩み始めます。

TLSへの進化

表1.1はSSL/TLSの歩みを簡単な年表にしたものです。ここでSSLからTLSの最新バージョンに至るまでの道のりを振り返ります。

表1.1　SSL/TLSの年表

年	プロトコル	詳細
1994	SSL 1.0	Netscape社が暗号化通信プロトコルとして開発（リリースなし）
1995	SSL 2.0	Netscape社がブラウザ用暗号化通信プロトコルとして公開
1996	SSL 3.0	脆弱性が指摘され設計の見直し 証明書チェーンの対応 IETFへ開発を移管
1999	TLS 1.0	RFC 2246（SSL 3.0の仕様を引き継ぎIETFが標準化） FIPS認証 HMACによる擬似乱数生成、パディング強化
2006	TLS 1.1	RFC 4346 AES、楕円曲線暗号の追加、CBC脆弱性対策、IV脆弱性改善
2008	TLS 1.2	RFC 5246 新規アルゴリズムの追加、DES廃止、ハッシュ強化、認証付き暗号・前方秘匿性サポート
2018	TLS 1.3	RFC 8446 RTT改善、アルゴリズムの選別、前方秘匿性徹底

SSL 2.0

SSL 2.0は初のインターネット通信の暗号化プロトコルとして公開されましたが、Netscape社の中だけで開発され、外部の専門家による検証が十分になされていませんでした。その結果、ほどなくしていくつかの脆弱性や機能

の不足が指摘されることになります[注19]。その後もいくつかの脆弱性が発見されており、現在では使用することを推奨されていません。Netscape社は暗号学者[注20]のリーダーシップのもとでSSLの次バージョンの開発を加速し、SSL 2.0が公開された翌年の1996年には指摘の脆弱性に対処したバージョンSSL 3.0を公開します。SSL 3.0は根本的な設計の見直しが行われており、基本的な設計思想はその後のTLSへと引き継がれていきました。

SSL 3.0

SSL 3.0はその後大きな成功を収め、数年でインターネットの標準的な暗号化プロトコルとして世界中に普及します。1996年にはインターネット関連技術の標準化を進める非営利組織であるIETF（*Internet Engineering Task Force*）への移管を目指し、SSLの標準化を検討するワーキンググループが発足します。当時「ブラウザ戦争」と呼ばれる市場シェアの争奪戦が巻き起こり、暗号化通信プロトコルにおいても独自技術の開発競争が激しさを増していました。インターネットの分断を防ぐためには第三者の専門家が共通の標準プロトコルを開発する必要性があったのです。IETFが策定した標準プロトコルはTLS（*Transport Layer Security*）と名付けられ、RFC 2246[注21][注22]として1999年に公開されます。これが現在TLS 1.0と呼ばれる最初のTLSとなります。

TLS 1.0

TLS 1.0はSSL 3.0の設計の大部分を引き継いで開発されました。SSL 3.0とTLS 1.0の間に互換性はありませんが、次のような基本的設計は今現在使用されているTLSにおいても踏襲されています。

注19 米国の暗号輸出規制の影響で暗号鍵が実質40ビットの強度に抑えられていたこと、ダウングレード攻撃（弱いアルゴリズムの強制）ができるなど致命的な脆弱性がありました。また証明書チェーンが使えず、ルートCAの発行した証明書しか検証できませんでした。

注20 タヘル・エルガマル（*Taher A.Elgamal*）とポール・コッヘル（*Paul Kocher*）。両者とも暗号学者で、タヘル・エルガマルはエルガマル暗号の発明者。2019年にはSSL/TLSの開発と通信のセキュリティへの貢献が称えられ、マルコーニ賞が授与されました。https://www.businesswire.com/news/home/20190313005174/en/Cryptographers-Paul-Kocher-Taher-Elgamal-Awarded-2019

注21 Request for Comments。IETFがインターネット関連標準技術に関する情報を管理・公開するための文書形式。

注22 RFC 2246 - The TLS Protocol Version 1.0 : https://datatracker.ietf.org/doc/html/rfc2246

1. クライアントがサーバに接続する際、「ハンドシェイクプロトコル」により使用する暗号アルゴリズムについて合意する
2. サーバは外部の権威ある機関により署名された公開鍵証明書を持っており、クライアントに配布する
3. クライアントは公開鍵証明書の真正性を検証することでサーバを認証する
4. クライアントとサーバは公開鍵暗号による鍵交換（鍵共有）アルゴリズムを使ってセッションに使用する共通鍵暗号の鍵の材料を共有する
5. 共通鍵暗号により暗号化されたセッションデータはMACと呼ばれるアルゴリズムで完全性の検証を行う

その他にもTLS 1.0は鍵交換や認証に使用できる暗号アルゴリズムが追加され、鍵生成に擬似乱数関数を使うようになりました。

標準プロトコルとなったTLS 1.0は、2000年代に入るとWebと共に急速に普及していきます。この頃になると米国の暗号輸出規制の緩和などの措置もあり、インターネットにTLSは必須であるとの共通認識が形成され、普及を後押しします。オープンソースによるSSL/TLS実装ライブラリであるOpenSSLもオープン化の波に乗り、メジャーな暗号ライブラリへと成長しました。環境が整ったことで2010年代までには主要な商用サーバにおける普及率は100%近くにまで達します。

反面、TLSが幅広く使われるようになるにつれ、暗号解読の攻撃に晒される頻度も増えていくことになります。TLS 1.0は長い間安全なプロトコルであると信じられてきましたが、特定条件下における脆弱性があることも徐々に分かってきます。TLS 1.0ではCBC（*Cipher Block Chaining*）という暗号利用モードが広く使われていましたが、CBCモードにはIV（*Initialization Vector*）やパディングといった補完的な機能があり、それがプロトコルやアプリケーションと組み合わされた際に脆弱性につながりやすいことが分かってきます。2001年には「パディングオラクル攻撃」と呼ばれる攻撃手法が発見され、後にOpenSSLへの攻撃も成功しました。パディングオラクルは致命傷とはなりませんでしたが、TLS 1.0はのちに「BEAST」と呼ばれる攻撃手法も発見されています。今ではTLSといえども絶対安全とは言えないことが広く認識されるようになりました。

TLS 1.1

2006年になると、RFC 4346[注23]としてTLS 1.1がリリースされます。TLS 1.1では、脆弱性が指摘されていたCBCモードに対する攻撃[注24]への対策が施されました。また、新しい共通鍵暗号のアルゴリズムとしてAES（*Advanced Encryption Standard*）が使えるようになりました。AESはその後広く浸透し、十数年が経過した現在でも最もよく使われている暗号アルゴリズムです。

TLS 1.2

その2年後の2008年には、さらに機能と安全性が強化されたTLS 1.2（RFC 5246[注25]）がリリースされます。TLS 1.2は前バージョンの「守り」に近い更新に比べると「攻め」の更新が多いのが特徴です。暗号アルゴリズムとしては認証機能で安全性が強化されたGCMモード、完全性を検証するためのハッシュ関数は計算量的安全性の高いSHA-256というアルゴリズムが採用されます。さらにハンドシェイクにおける暗号アルゴリズムの選択の柔軟性が向上し、使い勝手も改善されました。TLS 1.2はTLS 1.1と共に時間をかけてゆっくりと、しかし着実に普及・浸透し、リリースから十数年が経過した本書執筆時点において主要サーバの99.9%の普及率を誇るプロトコルとなっています。SSL/TLSはここで一つの完成形に到達したと言ってよいかもしれません。

しかしながら、**表1.2**に列挙した通り、ここまでの道のりの裏には数多くの脆弱性の問題があったことも事実です。特にTLS 1.2がリリースされたあたりから10年くらいは脆弱性の発見ラッシュとなり、SSL/TLSにとっては苦難とも言うべき状況が続きました。たとえ新しいバージョンの強化されたTLSが公開されても、実際にそれが実装されて普及・浸透するには多くの時間を必要とします。また、古いバージョンであっても致命的な脆弱性がなければ差し迫った脅威としては認識されず、実用上問題ないと判断されてしまうこともあります。通信は相手とつながらなければ意味がありませんので、そのことが古いバージョンのSSL/TLSの利用をやめることを躊躇させ、プロ

注23 RFC 4346 - The Transport Layer Security (TLS) Protocol Version 1.1 : https://datatracker.ietf.org/doc/html/rfc4346

注24 ブロック暗号の利用モードの一つであるCBC（*Cipher Block Chaining*）モードのIV（*Initialization Vector*）の採用方法に関する脆弱性を利用した攻撃。CBCモードについては第2章でも解説します。

注25 RFC 5246 - The Transport Layer Security (TLS) Protocol Version 1.2 : https://datatracker.ietf.org/doc/html/rfc5246

トコルの世代交代がなかなか進まない現実がありました。また、TLS 1.2にも安全性が十分とは言えないアルゴリズムが含まれていることが指摘されていました。しかも脆弱性が発見される度に機能の無効化や追加などの暫定的な対策を繰り返してきたため、そのことが運用をさらに難しく複雑にしていました。

表1.2　SSL/TLSに対する主な攻撃や脆弱性

発見された年	名称	特徴
1994	擬似乱数予測攻撃	乱数生成アルゴリズムの脆弱性の発見
1996	ダウングレード攻撃/ バージョンロールバック攻撃	ハンドシェイクで最弱アルゴリズムを選択させたりSSL 2.0の使用を強制する攻撃
1996～2000	パディングオラクル攻撃	パディングの改ざんによる平文の推測
2006	選択平文攻撃	大量の任意の平文を暗号化することで暗号化鍵を推測する攻撃
2009	再ネゴシエーション攻撃	再ネゴシエーションの脆弱性に対する中間者攻撃
2011	BEAST	CBCモードIV脆弱性を悪用した平文の推測
2011	Comodo事件 DigiNotar事件	認証局（登録局）がハッキングされ不正な偽造証明書が発行された
2012	CRIME	圧縮オラクルを悪用したサイドチャネル攻撃
2013	Lucky 13	CBCの完全性検証の脆弱性を悪用したパディングオラクル攻撃
2013	BREACH	HTTPレスポンスの圧縮サイドチャネル攻撃
2014	Heartbleed	OpenSSLのHeart Beat脆弱性に対する攻撃
2014	POODLE	SSL 3.0のCBCモードの脆弱性を悪用したパディングオラクル攻撃
2015	FREAK	輸出用暗号の脆弱性を悪用した攻撃
2016	SLOTH	MD5やSHA-1などのハッシュ関数の脆弱性に対する攻撃
2020	Raccoon Attack	DH鍵交換への中間者攻撃

コンピュータの能力が向上すれば、今まで現実的ではなかった攻撃もいずれは可能となります。どんなに盤石なプロトコル、アルゴリズムも時間が経てば、それだけたくさんの攻撃に晒されることになり、小さな綻びが発見されると、それがダムの決壊のように大きな脆弱性に発展します。暗号アルゴリズムの世界では、時間の経過と共にアルゴリズムの計算量的安全性が低下し、攻撃手法の発見で安全性が脅かされることを「危殆化する」と言います。セキュリティの研究者は危殆化の現状を警告するために、継続的に古いアル

ゴリズムやプロトコルへの攻撃を試み、成功させてきました[注26]。その効果もあり、近年は主要なブラウザやサーバで古いプロトコルやアルゴリズムの利用を積極的に中止する動きが見られています[注27]。

TLS 1.3の登場

技術的負債の解消

TLS 1.2は長い期間使用されましたが、近年では脆弱性の発見とその対策が積み上がり、多くの技術的負債を抱えていました。またTLSを取り巻く環境も大きく変化しています。SSL/TLSの登場初期はインターネット接続のトラフィックのほとんどが有線ネットワーク上のPCでしたが、近年はスマートフォンなどのモバイル、無線通信からの接続が有線を凌駕するようになりました。IoTやFA機器などWeb以外の利用も着実に増加傾向にあります。扱うデータの質・量だけでなく重要性も飛躍的に増大しました。その結果、TLSには安全性だけでなく性能面の配慮や、様々な環境に適した選択ができる柔軟性も求められるようになります。

2013年には過去の負債の払拭と、新しいニーズに対応することを目指して、TLS 1.3の検討が本格的に始まります。TLS 1.3はTLS 2.0と言ってもよいほど根本的な設計思想の見直しが行われ、多くの大胆な変更が盛り込まれました。異例とも言える5年以上の月日をかけて議論を積み重ね、2018年8月にRFC 8446[注28]としてTLS 1.3がリリースされます。

TLS 1.3では、危殆化により安全性が十分ではなくなった暗号アルゴリズムや暗号利用モードが軒並み廃止されました。これらのアルゴリズムはTLS 1.2までは下位バージョンとの互換性を考慮してサポートが継続されていましたが、既に脆弱性として重荷になっていたのです。標的となりやすかったブロック暗号のCBCモードも廃止となり、GCMモード[注29]のような認証付き暗号[注30]し

注26 本書執筆時点でSSL 2.0からTLS 1.1は利用すべきでない非推奨プロトコルとなっています。攻撃手法については本書の第6章「脅威・脆弱性」で解説します。

注27 Edge、Chrome、Firefox、Safariなどの主要ブラウザは、2020年前半までにTLS 1.0/TLS 1.1を無効化する計画を発表していました。その後、新型コロナウイルス感染症（COVID-19）のパンデミックの影響で無効化スケジュールが変更され、2022年に無効化されています。

注28 RFC 8446 - The Transport Layer Security (TLS) Protocol Version 1.3 : https://datatracker.ietf.org/doc/html/rfc8446

注29 Galois/Counter Mode。暗号利用モードの一つ。

注30 データの秘匿性だけでなく認証や完全性検証の機能も同時に実現した暗号で、認証付きでない暗号に比べて攻撃が難しいとされています。

か認められなくなりました。高い安全性が認められた比較的新しいアルゴリズムのみに絞り、危殆化リスクを徹底的に排除する方針が示されます。

前方秘匿性（Forward Secrecy）

前方秘匿性の重要性を世に知らしめたと言われているのが、2013年に世間を騒がせたスノーデン事件[注31]です。これは米国政府の機密文書がエドワード・スノーデン氏により暴露された事件です。その機密文書によれば、米NSAがテロ対策の一環でPRISMと呼ばれる大規模な通信監視計画により多くの米国人の通信記録を収集していたとされています。IETFはこの事件を受けて、「広範囲の通信の監視は攻撃である」と捉え、このような脅威に対抗することを宣言しました[注32]。

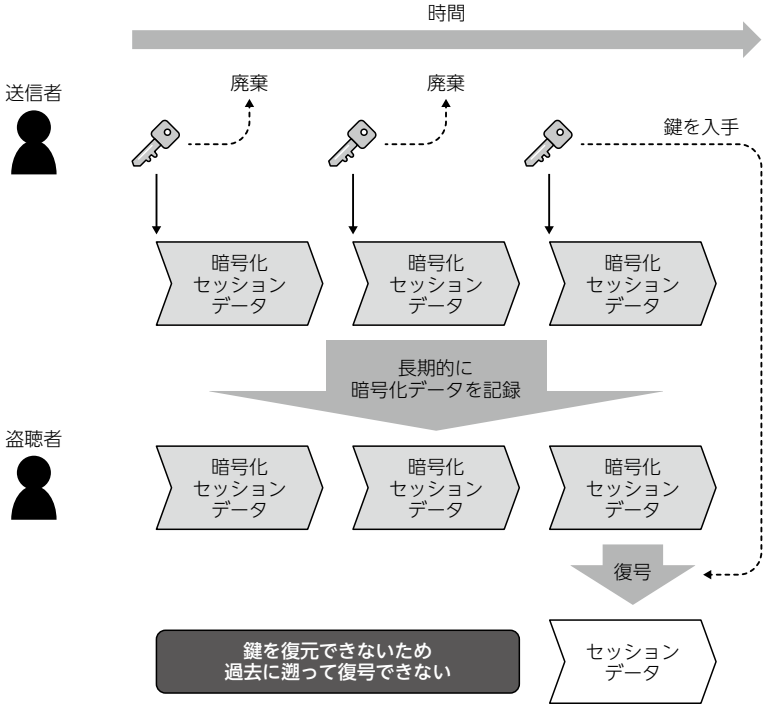
PRISMの通信記録の中にはインターネット上の暗号化されたデータも含まれていたと言われています。もちろん暗号化されたデータは記録してもすぐに読むことはできません。しかし、協力している事業者などに要請して秘密鍵を入手できれば、後から解読できるかもしれません。そのような攻撃から暗号を守るプロトコルの考え方が前方秘匿性[注33]（Forward Secrecy）と呼ばれる性質です。前方秘匿性を実装したプロトコルは通信のセッション（特定のアプリケーションやユーザによって維持される通信接続）ごとに鍵交換に必要な秘密の鍵を生成し、使い終えるとその都度廃棄します。そのため、一度セッションを終えた暗号データは鍵が残らず、通信記録を後から解読されるリスクが非常に小さくなります（**図1.11**）。

注31 元NSA職員だったエドワード・スノーデンが米NSAの通信記録収集プログラム（PRISM）に関する機密文書をマスコミにリークした事件。

注32 IETFは2014年5月にRFC 7258 - Pervasive Monitoring Is an Attack（https://datatracker.ietf.org/doc/html/rfc7258）を公開、同年11月にIAB（Internet Architecture Board）は「インターネットの信頼性に関する宣言（IAB Statement on Internet Confidentiality）」を公開しました（https://www.iab.org/2014/11/14/iab-statement-on-internet-confidentiality/）。

注33 前方秘匿性をより厳格にしたのがPFS（*Perfect Forward Secrecy*）という考え方です。PFSは鍵交換で使用した鍵が全て独立しており、仮にいずれかが漏洩しても他の鍵交換に影響しないことを前提としています。

図1.11 前方秘匿性



前方秘匿性を守るにはDHEやECDHEと言われる鍵交換アルゴリズムを使用します。このアルゴリズムはTLS 1.2から使えるようになっていましたが、TLS 1.3では前方秘匿性のあるアルゴリズムを必須とすることで暗号データの記録・保存に対抗することを明確に打ち出したのです。

RTTの削減

TLS 1.3には他にも重要な変更があります。特筆すべきは接続時のハンドシェイクに必要なメッセージの効率化を実現したことです。通信ネットワークにおいて、伝送遅延によってメッセージの往復に必要な時間を「RTT」（*Round Trip Time*）と呼びます。SSL/TLSでは、暗号データを送る前のハンドシェイクに必要な時間の目安として2-RTTや1-RTTといった表現をします。2-RTTは2往復分の遅延でハンドシェイクが完了することを意味します。従来のTLSではハンドシェイクに最低2-RTTを必要としていましたが、TLS

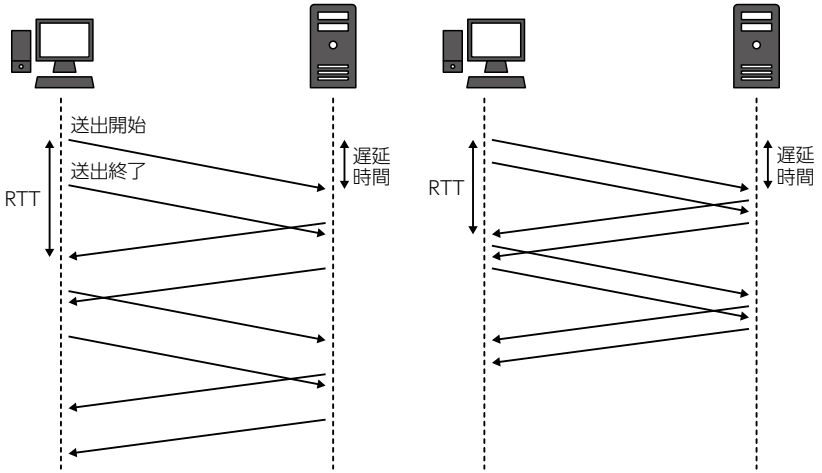
1.3ではこれを1-RTTに短縮しました。さらにセッションの再開に限定した場合は0-RTT、つまり最初から暗号データを送ることも可能となりました。

この変更は、TLSを様々なネットワーク環境で運用する上で大きな意味があります。単位時間内の通信でどれだけのデータ量を転送できるかを通信用語で「スループット」と言います。これは物理的な伝送路の通信速度とコネクション数により増やすことができ、インターネットにおけるスループットは通信技術の進歩と共に伸び続けています。しかし、TLSのハンドシェイクのようにメッセージの往復によるネゴシエーションを必要とする通信では、RTTが実際のアプリケーションの応答性能に与える影響を無視できません。モバイルや無線通信は多重化や輻輳制御の効率化により近年通信性能が著しく向上していますが、物理的な要因に制約される遅延時間の短縮は容易な仕事ではありません。そのため、無線通信は有線通信に比べてRTTによる遅延がより顕著になります。RTTを伝送路の通信速度の向上だけで改善するには限界があり、より速いレスポンスを実現するには往復メッセージを削減することが効果的です。

RTT削減の効果

このことはデータ通信量をトラックの積み荷に置き換えて考えてみると分かりやすいかもしれません。軽トラと10トントラックでは一度に運搬できる積み荷の量が明らかに異なりますが、両方とも同じ距離を同じ制限速度で移動する場合、往復に必要な時間は変わりません。荷台を広くしたり、台数を増やすことで一度に運べる量を増やすことはできますが、そのことは往復時間の短縮には貢献しません。通信データ量は積み荷の積載量、RTTは移動時間と考えることができます。このたとえは全データの伝送時間に対してRTTの遅延時間が極端に短い場合は適切ではありませんが、TLSのプロトコルのように少量のデータの往復でハンドシェイクが成立するケースや遅延の大きい長距離の低速無線通信などでは往復回数の削減の効果が大きいことが理解できます（**図1.12**）。

図1.12 RTTによる遅延



0-RTTはTLS接続の高速化を実現しましたが、実は前方秘匿性が保証されない仕組みのため、通常のハンドシェイクに比べて安全性が落ちるという問題があります。あくまで安全性と性能のトレードオフを前提として用途に応じて活用することを想定しているのです。このようにTLS 1.3は、IoTや無線通信など多様化する通信環境でも活用できるように配慮され、より柔軟な設計になったと言ってよいでしょう。

TLS 1.3が公開されてから本書執筆時点で5年近く経過していますが、主要なサーバにおけるサポート率は60%以上に達しており、順調に普及しているようです。無線デバイスの増加、テレワークの普及などの要因も後押しし、今後も安全で高速な暗号化通信の需要は伸び続けると考えられます。安全性と性能の両立を目指して設計を大きく変更したTLS 1.3は、SSL/TLSの歴史の中でも大きな転換点と捉えることができるのではないでしょうか。