暗号化する SSL

SECTION 08

「2-01　インターネットを構成するネットワーク」で説明したように、クライアントとサーバとが通信する際には、プロバイダや IXなど、間にたくさんのネットワークを通過します。
もし、経路上に悪意がある者がいると、データが盗み見られる恐れがあります。そこで重要なデータを通信するときは、盗聴されても中身がわからないよう、暗号化します。

安全にデータを届ける SSL

Webのデータを暗号化して安全に届けるのが、「**SSL**（Secure Sockets Layer）」という仕組みです。
通常の（暗号化しない）通信は、「**http://**」で始まるURLで示されます。このときのポート番号は**80番**です。
それに対して暗号化した通信は、「**https://**」で始まるURLで示されます。このときのポート番号は**443番**です。
つまり、暗号化を有効にしたWebサーバは、暗号化していない通信用のポート80番と、暗号化通信用のポート443番の2つで待ち受けしています。

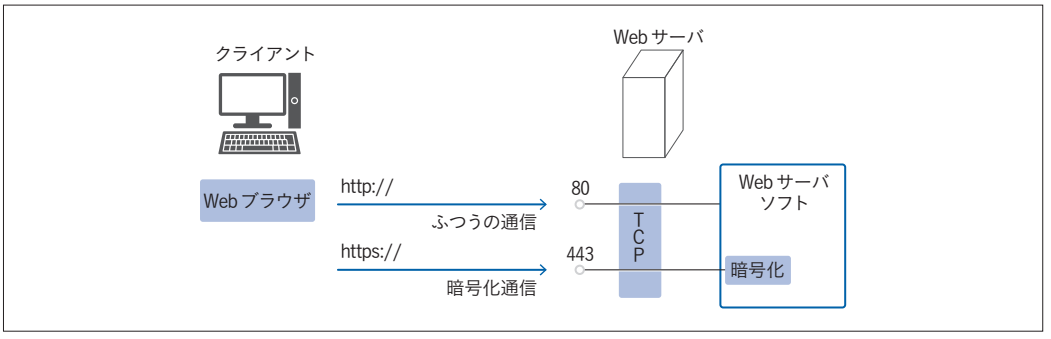

図2-8-1　暗号化が有効であるときのWebサーバ

MEMO

暗号化していないデータのことは、「平文（ひらぶん）」と呼ばれます。

MEMO

本書では知名度の高さゆえ便宜的に「SSL」と呼んでいますが、現在では、それを発展させた「TLS」が使われています。また本書では、主にRSA公開鍵暗号方式を使った通信について解説していますが、SSL/TLSは、それ以外の暗号方式にも対応しています（p.078のコラムを参照）。

「https://」で接続しているときは、ブラウザのアドレス欄に、**鍵のマーク**が表示され、クリックするとセキュリティの状態を見ることができます。

オンラインバンキングやショッピングサイトはもちろん、お問い合わせページなど、少しの個人情報を入力する場所でも、SSL（TLS含む。以下同じ）が導入されています。また最近は、SSL化されていないと検索エンジンの結果で下位に表示されたり、一部のブラウザで警告が表示されるなど、不利になることが多いため、扱うデータの機密性にかかわらず、SSL化しておくのが主流です。

> MEMO
> 全体がSSL化されているサイトでは、「http://」にアクセスしたとき、自動で「https://」にアクセスに接続し直すように設定しているサイトもあります。これは「リダイレクト」という機能で実現しています。

図2-8-2　セキュリティの状態を確認する

公開鍵暗号と署名

SSLにおいて、重要な技術が「公開鍵暗号」と「デジタル署名」です。前者はデータの暗号化に使うもの、後者はデータが改ざんされていないことを確認するのに使うもので、どちらも2つの鍵（key）を用います。「**秘密鍵**（private key）」と「**公開鍵**（public key）」です。

この2つの鍵は対をなしていて、「**キーペア**（key pair）」と呼ばれます。キーペアは、ツールを使って作ります。

> MEMO
> 具体的には、たとえばLinuxなどのサーバでSSLを用いる場合、「openssl コマンド」を使います。

秘密鍵は、その名の通り、誰にも見せてはいけない自分だけの鍵です。それに対して公開鍵は、皆に配る鍵です。

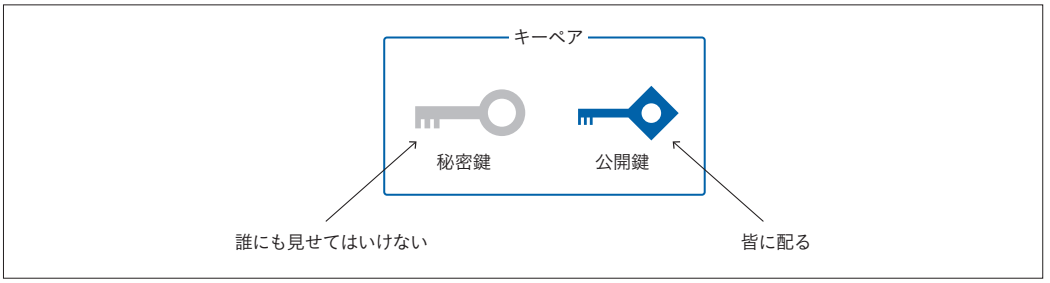

図2-8-3　秘密鍵と公開鍵

公開鍵暗号

2つの鍵のひとつめの用途は**暗号化**（encrypt）や、その**解読**です。
次の特徴があります。

- **公開鍵を使って暗号化したデータは、秘密鍵で解読できる**
- **公開鍵がわかっても、暗号化したデータは解読できない**
- **公開鍵がわかっても、秘密鍵がわかることはない**

MEMO

暗号化されたデータを、元のデータ（平文）に戻す（解読する）ことを、「**復号化**（decrypt）」と言います。

この特徴を使って、次のように暗号化通信します。

1　公開鍵を広く配布しておく

公開鍵をあらかじめ配布しておきます。

2　公開鍵を使って暗号化する

暗号化したいときは、1の公開鍵を使って暗号化します。

3　秘密鍵で元に戻す

2のデータを、秘密鍵を使って元に戻します。
秘密鍵は、秘密にしておく鍵ですから、漏洩しない限り、自分以外の人によって解読される恐れがありません。

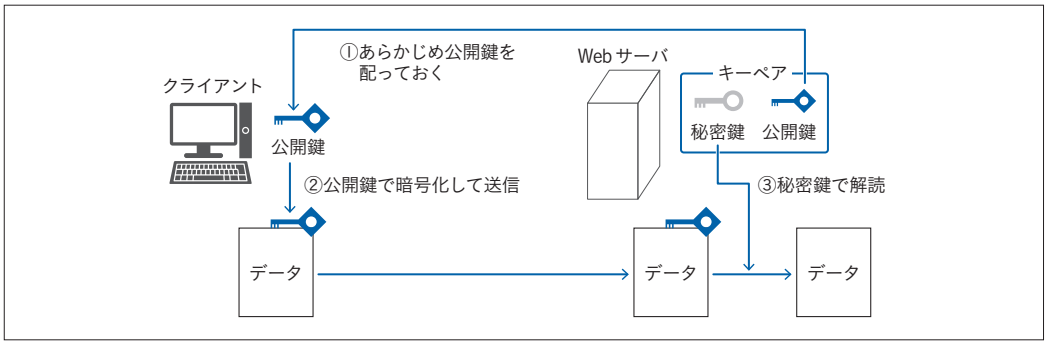

図2-8-4　暗号化通信の原理

デジタル署名

もうひとつの用途は、デジタル署名です。デジタル署名は、本人が作ったものであるかを確認する証です。この証は、ハッシュ値と呼ばれる値を使って作ります。

ハッシュ値とは、元のデータの特徴を捉えて短くしたデータ列です。同じデータから同じ方法で計算したハッシュ値は同一ですが、少しでもデータが違えばハッシュ値が大きく変わる特徴があり、改ざん検知に使えます。

デジタル署名は、こうした性質を持つハッシュ値を秘密鍵で暗号化したものです。データを受け取った側は、その人の公開鍵を使ってハッシュ値を取り出します。それとは別に、受信したデータからハッシュ値を再計算し、2つのハッシュ値が同一であれば、①改ざんされていないこと、②秘密鍵の保有者が作ったこと、を確認できます。

COLUMN　暗号化の方式と強度

データの暗号化の実体は、算術的な計算です。公開鍵暗号方式で肝となるのは、「公開鍵から秘密鍵を求めることができない」という点です。

これには、「AからBを求めるのはたやすいが、BからAを求めるのには膨大な時間がかかる」という、逆算の難しさを利用しています。

現在使われている公開鍵暗号方式は、「素因数分解を使った方法（RSA暗号方式）」と「楕円曲線暗号（EC-DSA方式）」が主流です。SSLは、どちらの暗号化方式にも対応しており、接続時に、クライアントとサーバの双方で対応している方式をやりとりして、どの暗号化方式を使うのかが決まります。

たとえば、素因数分解を使ったRSA暗号方式では、2つの素数AとBがあるとき、その積（かけ算の結果）は簡単に求められるけれども、逆に積から、元のAとBを求めるのは、とても時間がかかるという理論を利用しています。

「とても時間がかかる」というのがポイントで、計算できないわけではありません。何十年、何百年もの歳月をかければ、公開鍵に対応する秘密鍵を見つけられます。

「強い暗号強度をもつ鍵」というのは、「解読に時間を要する鍵」です。

鍵の長さは、「ビット（bit）」という単位で示され、長いほど、解読に時間がかかります。

鍵の長さは長いほど解読の恐れがなく安心ですが、計算量が多くなり負荷も高くなります。本書の執筆時点においては、2048ビットの鍵がよく使われます。

公開鍵が本物であるかどうかを確認するための証明書と認証局

公開鍵暗号だけでも、データの秘匿はできますが、安全ではありません。

もし、途中で誰かに、偽物の公開鍵にすり替えられてしまうと、本当の相手ではなく、偽物の相手に解読されてしまうからです。

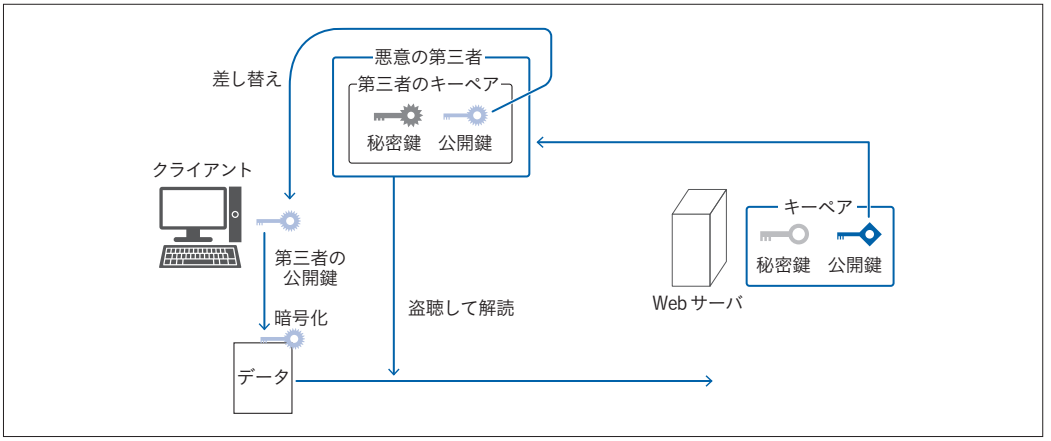

図2-8-5　第三者によって公開鍵がすり替えられる恐れがある

この問題を解決するには、通信相手が正しいかどうかを確認する仕組みが必要です。SSLでは、**PKI（Public Key Infrastructure：公開鍵基盤）**という枠組みで、この問題を解決します。

PKIでは、通信する際に「証明書」というデータを提示することで、正当な通信先であることを示します。証明書の実体は、「自分の公開鍵」に加えて、「運用するサーバ名や企業名、ときには所在地などの身分情報」を、第三者の証明書を使ってデジタル署名してもらったデータです。言い換えると、第三者によって「この公開鍵を使っているのは、この企業やサーバである」というお墨付きが付いたものです。ここで言うデジタル署名は、前ページのハッシュ関数で計算したものです。

PKIにおいては、「信頼できる組織が署名した証明書は信頼できる」という考え方をします。証明書を実際に発行するのは、信頼できる企業や団体です。これを**「認証局（CA。Certification Authority）」**と言います。OSやブラウザには、こうした信頼できる企業や団体が保有する証明書がインストールされていて、それを**「ルート証明書」**と呼びます。SSLでは、こうしたルート証明書がインストールされていることを前提とし、安全な通信を実現します。ルート証明書を含め、すべての証明書には有効期限が設定されています。また、秘密鍵が漏洩したなど安全が保てなくなったときに備え、証明書の失効情報を管理する「CRL（Certificate Revocation List）」という仕組みもあります。

1　Webサーバの証明書を申請してインストールする（事前準備）

Webサーバで公開鍵と秘密鍵のペアを作ります。このうちの公開鍵に、自分の企業名やサーバ名、所在地等の情報を追加したデータ（これを**CSR**／Certificate Signing Requestと言います）を認証局に送り、デジタル署名をしてもらって、証明書をもらいます。その証明書と秘密鍵の両方をWebサーバに、事前にインストールしておきます（認証局によっては、中間証明書と呼ばれる、ルート証明書から認証局に至るまでの証明書を追加でインストールしなければならないことがあります）。

2　証明書で通信相手の正当性を確認する

通信の際に、証明書をやりとりして通信相手の正当性を確認します。証明書には認証局のデジタル署名が付いているので、OSやブラウザにインストールされているルート証明書を使って、その正当性を確認することで、通信相手が正しいかを確認します。証明書は、サーバの公開鍵と運用するサーバ名や企業名などが含まれたものです。RSA公開鍵暗号方式を使っている場合は、この公開鍵を暗号化通信に用います。「証明書に含まれているサーバ名とアクセスしているサーバ名が違う」「署名の確認ができない」「有効期限が切れている」「失効している」など証明書の検証に失敗したときは、ブラウザによって警告が表示されます。

MEMO

この説明からわかるように、万一、悪意のある第三者に認証局を乗っ取られると、偽物の証明書が作られ、すり替えが自在になってしまいます。実際、2011年にオランダのDigiNotarという認証局が乗っ取られ、約500枚の不正な証明書が発行されたことがあります。このときは、CRLを更新したり、ブラウザやOSに内蔵されている信頼されるルート証明書をアップデートしたりすることで、証明書を失効する対応がされました。

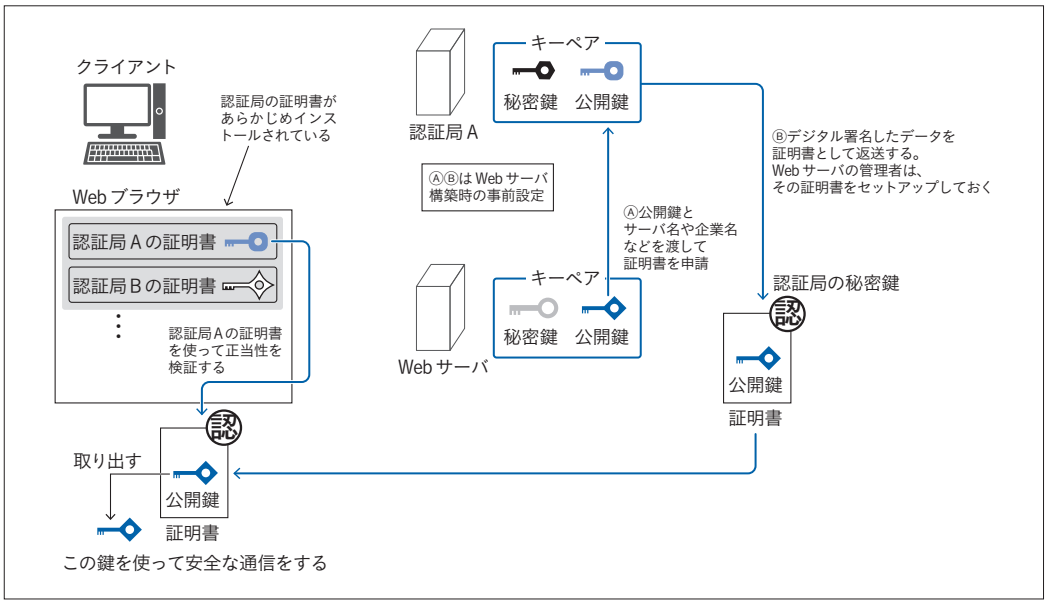

図2-8-6　証明書による偽装の防止

SSLでは共通鍵暗号方式と公開鍵暗号方式を併用する

このようにSSLでは、公開鍵暗号方式を使って安全性を保ちますが、実際のデータの暗号化には、公開鍵暗号方式を使いません。なぜなら、公開鍵暗号方式は、速度が少し遅いからです。

実際のデータは、「**共通鍵暗号方式**」と呼ばれる、「暗号化と復号化とで同じ鍵を使う方式」を使います。

具体的には、通信を始めようとするときに、クライアント側で、共通鍵暗号方式で用いる鍵をランダムに作成します。そして、そのランダムな鍵をサーバに送信するときにだけ公開鍵暗号化方式を使い、以降は、送信した鍵を使って、共通鍵暗号方式で暗号化します。

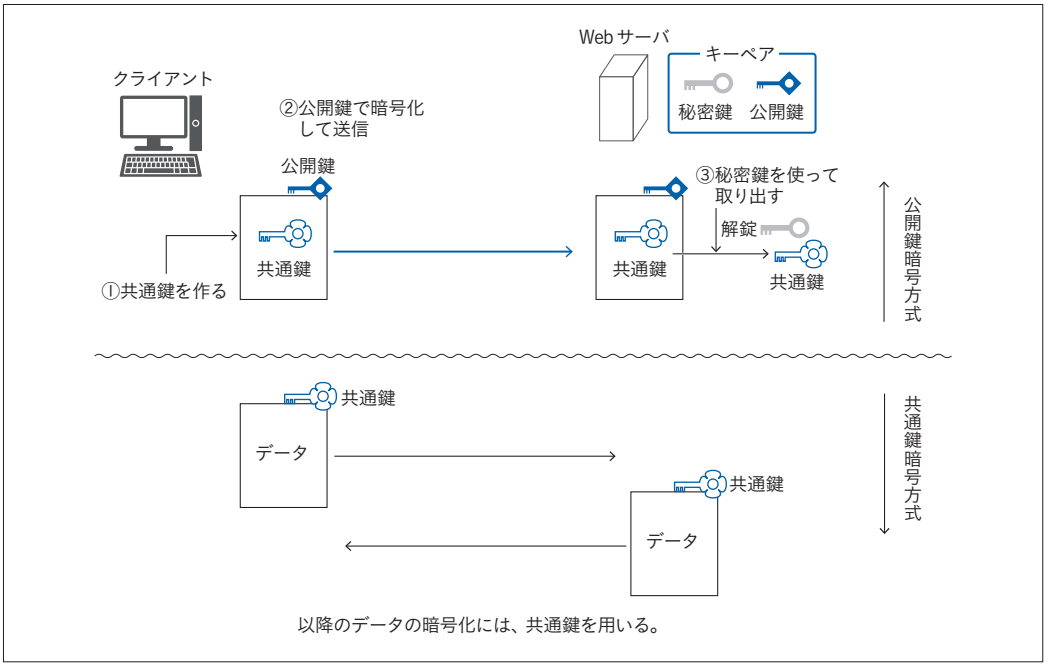

図2-8-7　SSLでは鍵の受け渡しのときだけ公開鍵暗号化方式を使う

COLUMN

SSLからTLSへ

2014年10月、Googleの技術者らによって、SSLにセキュリティ上の問題（脆弱性）が発見されました。この問題は、「POODLE（Padding Oracle On Downgraded Legacy Encryption）」と呼ばれます。

POODLEは、通信規約（プロトコル）の問題であり、根本的なデータのやりとりの方法が原因です。そのため、プログラムを更新することでは直せません。そこでセキュリティ業界では、SSLの利用を無効にして、TLS（Transport Layer Security）という方式に変えることが勧告されました。

TLSは、SSLの仕様を整理して練り直した後継版です。基本的な構造は変わりません。つまり、認証局や証明書を使うなど、そのやりとりに変わりありません。

現在では、すべてのWebサーバは、勧告に則り、SSLからTLSへの移行が完了しています（最新のブラウザがTLSでしか接続できないようにアップデートされたからです）。つまり、「https://」で接続したときは、SSLではなくTLSが使われます。

しかしSSLという名称とTLSという名称の知名度を比べると、SSLのほうが圧倒的に幅広く知れ渡っているため、相当しばらくの間は、本当はTLSで接続しているのだけれど、「SSLで通信する」とか「SSL用の証明書を作る」という言い回しが使われ続けることでしょう。

なお、最新のTLS1.3では、TLS1.2までで使われていた、RSA公開鍵暗号方式を使って共通鍵を交換する方式（図2-8-7に示したもの）は廃止され、共通鍵を通信網に流さずに、計算によって求めるDHE鍵交換やECDHE鍵交換と呼ばれる方式が採用されています。現行では、TLS1.2もTLS1.3も規格として有効なので、どちらの方式もありますが、将来的にTLS1.2が廃止されたならば、RSA公開鍵暗号方式を使った鍵の交換は使われなくなります。