
# CSS SSL の概要

Secure Sockets Layer（SSL）は、e コマース Web サイトでのクレジット カード番号の送信など、インターネットで安全なトランザクションを確保するための暗号化技術を提供するアプリケーションレベルのプロトコルです。SSL は、プライバシーと認証、およびデータの完全性を組み合わせ、クライアントとサーバ間のデータ トランザクションのセキュリティを確保します。SSL の高度なセキュリティは、証明書、秘密 / 公開キー交換ペア、および Diffie-Hellman キー合意パラメータを通じて実現されます。

この章の主な内容は次のとおりです。

- SSL 暗号化の概要
- CSS での SSL モジュールの機能の概要

# SSL 暗号化の概要

CSS は、SSL アクセラレーション モジュールと専用の SSL コマンド群を使用して、クライアントとサーバ間で SSL 暗号化処理を実行します。SSL の機能には、クライアントとサーバの認証、秘密キーと公開キーの生成、証明書の管理、およびデータ パケットの暗号化と復号化（暗号解除）があります。

SSL モジュールは、SSL バージョン 3.0 と Transport Layer Security （TLS）バージョン 1.0 をサポートしています。また、SSL モジュールは SSL バージョン 2.0 の ClientHello メッセージを受け入れ、正しく解釈できます。したがって、SSL 2.0 と SSL 3.0 の両方をサポートするクライアントは、SSL モジュールを介して CSS と通信できます。この場合、クライアントは SSL 2.0 の ClientHello 内で、SSL バージョンとして 3.0 を明示し、それによって SSL 3.0 に対応していることを SSL モジュールに通知します。SSL モジュールは、SSL 3.0 の ServerHello メッセージを返送します。

**（注）** 現在、SSL バージョン 2.0 だけをサポートするクライアントはほとんどありませんが、クライアントがバージョン 2.0 だけをサポートする場合、SSL モジュールはネットワーク トラフィックをやりとりできません。

SSL モジュールを使用する一般的な SSL セッションでは、接続を安全に確立および管理するために暗号化が必要です。SSL モジュールがキー交換、認証、および Message Authentication Code （MAC; メッセージ認証コード）を実行するために必要な暗号化アルゴリズムは、暗号スイートによって提供されます。サポートされる暗号スイートの詳細については、第 3 章「SSL 証明書とキーの設定」の「暗号スイートの指定」を参照してください。

ここでは、CSS の SSL モジュールを通じて実装される SSL 暗号化の概要について説明します。次の内容について説明します。

- SSL 公開キー基盤の概要
- SSL モジュールの暗号化機能

# SSL 公開キー基盤の概要

SSL は、Public Key Infrastructure （PKI; 公開キー基盤）で認証、暗号化、およびデータの完全性を実現します。PKI は、デバイス間で情報を安全にやりとりするためのポリシーと手順のセットです。非対称暗号化で使用される PKI には 3 つの基本要素があり、これらの要素によって e コマースを実装するための安全なシステムと、企業のイントラネットからインターネット上 e ビジネス アプリケーションに至る、あるゆる種類の電子トランザクションを支える信頼性の高い環境が提供されます。

この 3 つの要素は次のとおりです。

- 機密性
- 認証
- メッセージの完全性

## 機密性

*機密性*とは、意図されたユーザ以外、データを参照できないことを意味します。PKI では、さまざまな方式によるデータの暗号化によって機密性が実現されます。SSL では特に、2 つのエンドポイントだけで認識される 1 つ以上の対称キーを使用して、大量のデータが暗号化されます。通常、対称キーは 2 つのエンドポイントのいずれか一方で生成されるため、生成したキーを他方のエンドポイントに安全に送信する必要があります。対称キーを安全に送信する手段として、通常は*キー交換*、*キー合意*という 2 種類のメカニズムを使用します。

この 2 種類の送信メカニズムのうち、*キー交換*は特に広く普及しています。キー交換では、1 つのデバイスが対称キーを生成し、非対称暗号化スキームで暗号化を行った後、そのキーを他方のエンドポイントに送信します。非対称暗号化では、両方のデバイスが公開キーと秘密キーをそれぞれ 1 つずつ持っていることが必要です。この 2 つのキーは数字的に関連付けられています。つまり、公開キーで暗号化できるデータは秘密キーで復号化できます。同様に、秘密キーで暗号化できるデータは公開キーで復号化できます。最も一般的に使用されているキー交換アルゴリズムは、Rivest Shamir Adelman （RSA）アルゴリズムです。

SSL では、送信者が受信者の公開キーを使用して対称キーを暗号化します。これによって、受信者の秘密キーは、送信データを復号化できる唯一のキーになることが保証されます。非対称暗号化のセキュリティは、所有者だけが秘密キーを認識可能で、第三者は認識できないという事実が大前提になります。このキーの機密性が何らかの理由で侵害されると、第三者の Web ユーザ（または Web サイト）が対称キーを含むストリームを不正に復号化し、送信データ全体を復号化するおそれがあります。

*キー合意*では、データ交換を行う双方の側が共同で対称（共有）キーを生成します。最も一般的なキー合意アルゴリズムは Diffie-Hellman アルゴリズムです。Diffie-Hellman では、計算されクライアントとサーバの間で交換される共通キーの生成は、特定のパラメータによって左右されます。

## 認証

*認証*は、交換を行う 1 つ以上のデバイスで、対話している相手が、実際に意図した相手かどうかを検証するために必要です。たとえば、クライアントが e コマースの Web サイトに接続しているとします。クレジット カード番号などの機密情報を送信する前に、クライアントは接続先のサーバが e コマースの Web サイトであることを確認します。トランザクションを開始する前に、クライアントとサーバの両方で、相手方の認証が必要な場合もあります。銀行間の金融トランザクションでは、クライアントとサーバの両方で、相手方が本当に意図した取引先であるかどうかを確認する必要があります。SSL では、デジタル証明書を使用することでこの認証を容易に行うことができます。

デジタル証明書は、クライアントの正当性をサーバに証明するための一種のデジタル ID です。Certificate Authority （CA; 認証局）が PKI に基づいてデジタル証明書を発行します。PKI の公開キーと秘密キーの暗号化により、セキュリティが確保されます。CA は、証明書に「署名」することによって、その証明書の正当性を保証する権限を持ちます。CSS に接続されているクライアントやサーバには、同じ CA またはその CA と信頼関係の連鎖で結ばれている（A が B を信頼し、B が C を信頼する場合の A と C など）別の CA から発行された正当な証明書が必要です。

証明書は、識別情報が正しいこと、および公開キーが実際にそのクライアントまたはサーバに属することを保証します。デジタル証明書には、所有者に関する詳細、証明書発行者に関する詳細、所有者の公開キー、有効日付と失効日付、関連する特権などの情報が記載されています。

証明書を受領したクライアントは、証明書発行者に接続し、発行者の公開キーを使用して証明書の有効性を確認できます。これによって、証明書が実際に CA で発行および署名されたものであることが確認できます。証明書は、有効期限が切れるまで、または終了するまで有効です。

## メッセージの完全性

*メッセージの完全性*により、メッセージの受信者は、メッセージの内容が送信中に改ざんされなかったことを確認できます。SSL では、メッセージ ダイジェストをデータに適用してから送信することでメッセージの完全性を実現します。メッセージ ダイジェストは、任意の長さのメッセージから、メッセージの特徴を表す固定長の文字列を出力する機能です。

メッセージ ダイジェストの重要な特徴に、元に戻すのが非常に難しいという点があります。メッセージのダイジェストをメッセージ自体に付加するだけでは、メッセージの完全性を保証する手段として十分ではありません。攻撃者はメッセージを変更してから、それに合わせてダイジェストを変更する可能性があります。送信者の秘密キーを使用してメッセージ ダイジェストを符号化すると、メッセージ認証コード（MAC）と呼ばれるメッセージ完全性アルゴリズムが生成されます。受信者は、送信者の公開キーを使用して MAC を復号化できます。SSL は、MAC の 2 つのアルゴリズム、Message Digest 5（MD5）と Secure Hash Algorithm（SHA）をサポートしています。

ただし、この完全性スキームは、送信者の秘密キーが漏洩していると有効性を失います。その場合、攻撃者は送信者の MAC を偽造できるからです。メッセージの完全性も、秘密キーの保護に左右されます。このプロセスは、デジタル署名と呼ばれます。

RSA キー ペアは、MAC の署名に有効です。ただし、キー交換と署名の機能を分離した方が効果的な場合もあります。SSL アルゴリズムの 1 つである Digital Signature Algorithm（DSA; デジタル署名アルゴリズム）はデジタル署名には厳格に適用されますが、キー交換については厳格ではありません。

DSA は FIPS-186、すなわち Digital Signature Standard （DSS; デジタル署名規格）として標準化されています。DSA と DSS が互いに区別せずに使用される場合もあります。DSS は Diffie-Hellman と同じ暗号化ロジックを使用し、キーを生成するには Diffie-Hellman と同様のパラメータが必要です。また、DSS の用途は、Secure Hash Algorithm 1（SHA-1）メッセージ ダイジェストだけに限定されます。

## SSL モジュールの暗号化機能

表 1-1 に、SSL モジュールの SSL 暗号化機能について説明します。

**表 1-1　SSL モジュールの SSL 暗号化機能**

| SSL 暗号化機能 | SSL モジュールでサポートされる機能 |
|---|---|
| SSL のバージョン | SSL バージョン 3.0 と Transport Layer Security （TLS）バージョン 1.0 |
| 公開キー交換とキー合意アルゴリズム | • **RSA** : 512 ビット、768 ビット、1024 ビット、および 2048 ビット（キー交換およびキー合意アルゴリズム）<br>• **DSA** : 512 ビット、768 ビット、および 1024 ビット（証明書署名アルゴリズム）<br>• **Diffie-Hellman** : 512 ビット、768 ビット、1024 ビット、および 2048 ビット（キー合意アルゴリズム） |
| 暗号化タイプ | • Data Encryption Standard （DES; データ暗号規格）<br>• Triple-Strength Data Encryption Standard（3DES; トリプル DES）<br>• RC4<br>サポートされている暗号スイートおよびキー暗号化タイプのリストについては、第 4 章「SSL 終了の設定」の表 4-1 を参照してください。 |
| ハッシュ タイプ | • SSL MAC-MD5<br>• SSL MAC-SHA1<br>サポートされている暗号スイートおよびハッシュ タイプのリストについては、第 4 章「SSL 終了の設定」の表 4-1 を参照してください。 |

表 1-1　SSL モジュールの SSL 暗号化機能（続き）

| SSL 暗号化機能 | SSL モジュールでサポートされる機能 |
|---|---|
| デジタル証明書 | SSL モジュールは、次にあげるような認証局（CA）が発行する主要なデジタル証明書をすべてサポートします。<br>• VeriSign<br>• Entrust<br>• Netscape iPlanet<br>• Windows 2000 Certificate Server<br>• Thawte<br>• Equifax<br>• Genuity |

# CSS での SSL モジュールの機能の概要

CSS 11503 と CSS 11506 は複数の SSL モジュール（CSS 11503 では最大 2 つ、CSS 11506 では最大 4 つ）を使用できます。CSS 11501 は統合型の SSL モジュールを 1 つサポートします。

SSL モジュールはクライアントとサーバ間で、すべてのユーザの認証、公開キーや秘密キーの生成、管理の許可、およびパケットの暗号化と復号化を行います。SSL モジュールは、スイッチ モジュールに依存してネットワーク トラフィックの処理を行うインターフェイスを提供し、Switch Control Module （SCM; スイッチ コントロール モジュール）に依存して設定情報を送受信します。

CSS は、すべての証明書とキーを SCM ディスクに格納します。CSS は、SSL モジュールごとに最大で 256 の証明書と 256 のキー ペアをサポートします。これは、ディスクのおよそ 3 MB の記憶域に相当します。CSS は、すべての証明書とキー関連のファイルを、ディスクの安全な場所に格納します。格納済みの証明書とキーは、接続を処理する際に CSS によって SSL モジュールの揮発性メモリにロードされ、処理の高速化が図られます。

SSL コンテンツ ルールをアクティブ化して次の各項目を決定すると、SCM から SSL モジュールにネットワーク トラフィックが送信されます。

- コンテンツが物理的に存在する場所
- コンテンツの要求を送信する場所（サービス）
- 使用するロード バランシング方式

SSL プロキシ リストによって、SSL モジュールが送受信する情報の流れが決まります。プロキシ リストでは、クライアントから SSL モジュールへのフローを定義します。また、SSL モジュールからバックエンド SSL サーバへのフローも定義します。SSL モジュールが SSL コンテンツ要求を処理する方法を定義するために、SSL プロキシ リストを SSL サービスに追加します。SSL モジュールの機能の詳細については、第 8 章「CSS SSL 設定の例」の「SSL モジュールによる SSL フローの処理」を参照してください。

SSL モジュールの主な機能は次のとおりです。

- SSL 終了
- クライアント認証
- バックエンド SSL

- SSL 開始

## SSL 終了

SSL モジュールとクライアントの間のフローを定義するためにプロキシ リストのエントリを定義するときに、Web ブラウザ（クライアント）と HTTP 接続（サーバ）の間のセキュリティ サービスを追加することで、SSL モジュールは仮想 SSL サーバとして動作します。クライアントから着信するすべての SSL フローは、CSS の SSL モジュールで終端します。

いったん接続を終端すると、SSL モジュールはデータを復号化し、それをクリア テキストとして CSS に送信します。そして CSS で、ロード バランシングの決定が行われます。CSS はこのデータをクリア テキストで HTTP サーバに送信します。CSS の SSL 終了の詳細については、第 4 章「SSL 終了の設定」を参照してください。

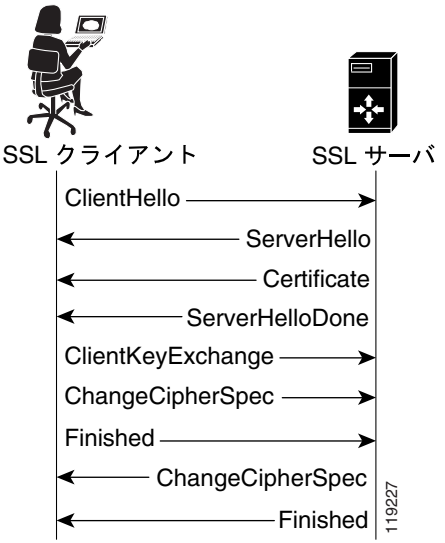
# クライアント認証

CSS によるクライアント認証では、次の点が確認されます。

- 証明書を送信したクライアントが、対応する秘密キーを持っていること
- 既知の CA がクライアント証明書に署名していること
- 証明書が失効していないこと
- シグニチャが有効であること
- Certificate Revocation List （CRL; 証明書失効リスト）が CSS に設定されている場合、発行元 CA が証明書を取り消していないこと

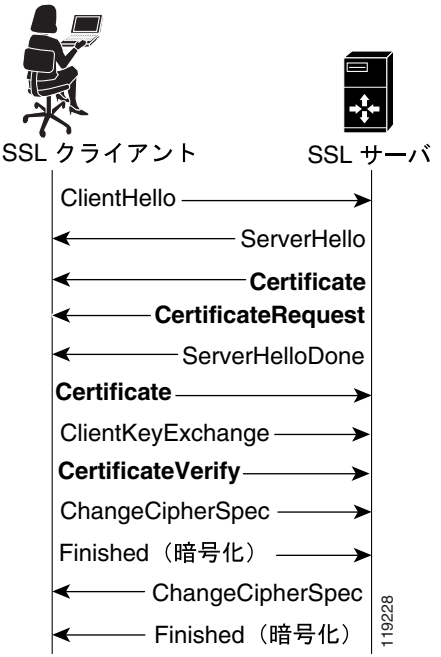
クライアントとサーバ間の典型的な SSL ハンドシェイクでは、図 1-1 に示すように、クライアントは証明書を送信しません。

**図 1-1　クライアント認証のない SSL ハンドシェイク**



クライアントに証明書を送信させるには、図 1-2 に示すように、サーバ側でハンドシェイクに証明書要求（CertificateRequest）メッセージを含める必要があります。この要求メッセージには、サーバが許可する証明書の種類が記載されています。ただし、このメッセージには認証局が示されていません。

**図 1-2　クライアント認証のある SSL ハンドシェイク**




サーバが ServerHelloDone メッセージを送信した後、クライアントは証明書（Certificate）とキー交換を応答します。次にクライアントは、サーバからのすべてのハンドシェイク メッセージのダイジェストを含み、クライアントの公開キーを使用して署名した CertificateVerify メッセージを送信します。サーバはクライアントの公開キーを使ってメッセージを復号化します。これによって、クライアントが正しい秘密キーを持っていることが検証されます。

CertificateVerify メッセージでは、証明書の正当性は検証されません。クライアントの秘密キーの公開部分が、証明書に含まれるものと一致するかどうかが検証されます。これにより分かるのは、クライアントがその証明書の作成に使用したキーペアを所有しており、他人の証明書を送信したのではないということです。一方、CSS では発行元のシグニチャが正当なものかどうかを検証することができます。X.509 証明書には、CA の秘密キーを使って証明書オブジェクト全体のメッセージ ダイジェストに署名することによって生成されるシグニチャが含まれます。CA 証明書には、クライアント証明書のデジタル署名を検証する CA 公開キーが含まれます。サーバが CA 証明書を所有し、したがって CA の公開キーを所有していれば、クライアント証明書が CA によって署名されたものであることが確認できます。CSS では、仮想 SSL サーバごとに最大 4 つの CA 証明書を設定できます。

CA はクライアントの証明書を取り消すと、その証明書を Certificate Revocation List（CRL; 証明書失効リスト）と呼ばれる公開リストに加えます。CA はこのリストを公表し、定期的に更新します。クライアントとサーバは HTTP 経由でこのリストにアクセスし、証明書が有効かどうかを検証できます。CSS では、CRL を CSS に取り込んだ方法と日時を定義する CRL レコードを設定できます。CSS が CRL をダウンロードした後、仮想 SSL サーバはその CRL を使用して、すべてのクライアント証明書の有効性を確認できます。

CSS の仮想 SSL サーバ上でのクライアント認証の設定（クライアント認証の有効化、CA 証明書の正当性の確認、CRL レコードの設定、仮想 SSL サーバへの CRL レコードの割り当てなど）については、第 4 章「SSL 終了の設定」を参照してください。

## バックエンド SSL

SSL プロキシ リストのバックエンド SSL サーバのエントリでは、SSL モジュールからバックエンド SSL サーバへのフローを定義します。クライアントから暗号化データを受け取った後、SSL モジュールはそのクライアントの IP アドレスを保つことで仮想クライアントとして動作し、フローのロード バランシングに使用したクリア テキスト データを再度暗号化して、バックエンド サーバへの SSL 接続を開始します。

CSS からの送信フローでは、SSL モジュールはこれと反対方向に応答し、サーバからの暗号化データをクライアントに送信します。CSS のバックエンド SSL の詳細については、第 5 章「バックエンド SSL の設定」を参照してください。

## SSL 開始

SSL 開始を設定した CSS は、クライアントからクリア テキストを受信し、SSL サーバとの SSL セッションを開始して、クライアント接続を SSL バックエンド接続に結合できます。SSL 終了用に設定した CSS（仮想 SSL サーバ）と、実際のバックエンド SSL サーバ（Web サーバ）のどちらも、SSL サーバとして使用することができます。

送信フローでは、CSS はサーバからの SSL データを復号化し、クリア テキストをクライアントに戻します。CSS の SSL 開始の詳細については、第 6 章「SSL 開始の設定」を参照してください。

SSL モジュールの機能の詳細については、第 8 章「CSS SSL 設定の例」の「SSL モジュールによる SSL フローの処理」を参照してください。