OpenSSL における SSL_SESSION オブジェクトのマイグレーションについての考察

中澤 昌史† 島崎 聡史† 金田 健太郎†† 黒羽 秀一†† 齋藤 孝道†
†明治大学 ††明治大学大学院

1 はじめに

インターネット通信の安全性を確保するために，SSL (Secure Socket Layer)[1] やその後継にあたる TLS (Transport Layer Security)[2] が広く利用されている (以降，これらを併せて SSL と呼ぶ)．SSL を用いた通信は，複雑な暗号処理を施すため，TCP 通信と比べてその処理コストは高くなる．また，サーバに処理が集中すると，処理遅延及びサーバ停止に陥る可能性がある．それ解決する方法として，複数のサーバを協調動作させ，負荷を分散させる方法がある．

しかし，SSL 通信では，バルク暗号化通信に先立ち，クライアントとサーバで暗号パラメータの作成時に必要なマスターシークレットなどの秘密情報を共有 (以降 SSL Handshake と呼ぶ) する必要がある．よって，複数のサーバを協調動作させる場合，サーバ間でこれらの秘密情報を共有する必要がある．

そこで，本論文では，SSL を利用した通信を行う際に，暗号 API として広く利用されている OpenSSL[3] を用いて，複数のサーバ間で秘密情報を共有するために，SSL_SESSION オブジェクトのマイグレーションについて検討する．

2 OpenSSL

OpenSSL は，SSL の実装だけでなく，証明書の発行といった PKI 関連の処理や公開鍵暗号化方式や共通鍵暗号化方式などを容易なインターフェースで利用可能とした API を含むツールキットである．OpenSSL の API は libcrypto と libssl に分類され，前者は暗号技術を，後者は SSL 通信を提供する．libcrypto は，共通鍵暗号化方式として DES，3DES，AES など，公開鍵暗号化方式として，RSA，DH (Deffie-Hellman) など，ハッシュ関数として，SHA-1 や MD5 といった主要なアルゴリズムを提供する．本論文では，OpenSSL-0.9.8b を対象とする．

2.1 OpenSSL を使ったプログラミングの概要

OpenSSL を用いて SSL 通信を行う場合の手順は以下の通りである [4]:

(1) TCP 通信を行う場合と同様に，ソケットを開く

(2) SSL_CTX_new 関数を用いて，SSL_CTX オブジェクトを生成する

(3) SSL_new 関数を用いて，SSL オブジェクトを生成する

(4) SSL_connect 関数や SSL_accept 関数を用いて SSL Handshake を行い，SSL_write 関数や SSL_read 関数などを用いてデータの送受信を行う

(5) アプリケーションを終了する際には，SSL 通信を切断し，手順 (1)〜(3) と逆の順番で各オブジェクトを開放し，ソケットを閉じる

3 OpenSSL のオブジェクト群

OpenSSL で定義されているオブジェクト群の中で，SSL_CTX オブジェクトや SSL オブジェクト，SSL_SESSION オブジェクトが主なものとしてあげられる．各オブジェクトでは，SSL 通信の際に必要なパラメータを保持している．OpenSSL では，SSL_SESSION オブジェクトを用いて，SSL のセッション情報を管理しており，このセッション情報を含む SSL オブジェクトにより，SSL 通信を行っている．また，SSL_CTX オブジェクトにより，複数ある SSL_SESSION オブジェクトを管理している．以下で，各オブジェクトについて詳しく述べる．

3.1 SSL_CTX オブジェクト

SSL_CTX オブジェクトは，SSL 通信に必要な共通の設定値を保持している構造体である．

OpenSSL を用いたアプリケーションは，SSL 通信に先立ち，SSL_CTX オブジェクトを 1 つ生成する必要がある．SSL_CTX オブジェクトは，自身の証明書や秘密鍵，SSL の機能を実現するメソッド群，信頼するルート CA，選択可能な暗号スイートを格納する．また，キャッシュできるセッションのサイズやその有効期限及び，作成した SSL_SESSION オブジェクトのリストといった情報も格納し，54 のメンバーにより構成される構造体として定義されている．

SSL_SESSION オブジェクトのリストは，セッション再開を行う際に利用される．既に確立された全ての SSL_SESSION オブジェクトを，SSL_CTX オブジェクトに保存することで，SSL_CTX オブジェクトからセッション再開に必要となる情報を参照できる．

これらの情報は SSL 通信をセットアップする際に参照されるため，SSL オブジェクトを作成する前に，SSL_CTX オブジェクトを用意する．

† Masashi NAKAZAWA, Satoshi SHIMAZAKI, Takamichi SAITO
†† Kentaro KANEDA, Shuichi KUROBA
Meiji University (†)
Graduate School of Meiji University (††)

3.2 SSL オブジェクト

SSL オブジェクトは，個々の SSL 通信に，SSL_CTX オブジェクトよりも詳細な設定値を保持している構造体である．このオブジェクトは，SSL_CTX オブジェクトを SSL_new 関数に渡すことで生成することができ，これにより作成された SSL オブジェクトは，SSL_CTX オブジェクトに設定されたパラメータを継承することになる．

SSL オブジェクトは，SSL_SESSION オブジェクト，暗号パラメータ (MAC 鍵や暗号化鍵，IV)，マスターシークレットや暗号化パラメータ作成時のシード (クライアントランダムやサーバランダム) を格納している．その他，シーケンス番号，クライアントなのかサーバなのかを示すフラグ値，及び SSL Handshake を行うメソッドやセッション ID を生成するメソッドなどを格納する 58 個のメンバーにより構成される構造体として定義されている．

データの送受信では，SSL オブジェクトに格納されたパラメータを用いて，SSL 通信を行う．

3.3 SSL_SESSION オブジェクト

SSL_SESSION オブジェクトとは，セッション再開などのセッション管理に必要な情報を格納する構造体である．このオブジェクトは，SSL Handshake 中の client hello メッセージの送受信の際に生成され，SSL Handshake を通して，各パラメータが設定される．

SSL_SESSION オブジェクトは，セッション ID やマスターシークレット及びその長さ，通信相手の証明書，圧縮アルゴリズム，暗号化アルゴリズムを格納している．その他，セッションの保存期間やセッションの再開が可能かを示すフラグ値などといった情報が格納する 25 のメンバーにより構成される構造体として定義されている．

複数 SSL_SESSION オブジェクト扱うために，双方向に参照可能なリストで管理しており，このリストの先頭と末尾のノードは，SSL_CTX オブジェクトから参照されることで，SSL_SESSION オブジェクトは管理されている (図 1 を参照)．

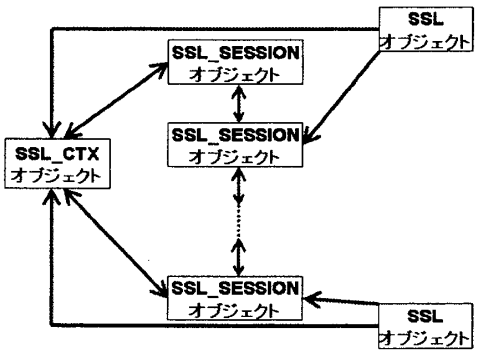

図 1: 各オブジェクトの参照関係

4 SSL におけるセッションの管理方法

サーバは，複数のクライアントからのリクエストを処理する必要があり，それぞれのクライアントからの接続に対して，マルチプロセス (もしくはマルチスレッド) で対処することとなる．すなわち，あるプロセスで，SSL Handshake を行いセッションを確立し，別のプロセスで処理を引き継ぐ場合，セッションの管理には特別な方法が必要になる．

一般に，マルチプロセスを用いるなどして複数のプロセスを利用する場合は，それぞれのプロセスはメモリを共有しないので，プロセス間でセッション情報を共有する機能が必要になる．これを実現する方法として，例えば，DBM (DataBase Manager) や共有メモリを用いることができる [5]．これらの方法は，mod_ssl[6] で用いられている方法である．

DBM は，検索キーをその値に対応付け，データはファイルとしてディスク上に保存され，それぞれのプロセス間で共有する．

一方，共有メモリでは，対応付けた検索キーと値を共有メモリ内に保存し，それぞれのプロセスからアクセスができるように，共有メモリセグメントに置く．この方法では，メモリを用いているため，ディスクを用いている DBM よりも高速なアクセスが可能となる．

どちらの方法もマルチプロセス処理なので，保存しているデータを書き換える際には，データをロックし，書き換え処理が終了したらロックを解除する必要がある．

5 まとめ

本論文では，OpenSSL における SSL_SESSION オブジェクトをどのように管理しているかについて調査した．これにより，SSL_SESSION オブジェクトをマイグレーションすることでセッションが再利用できることが確認できた．

参考文献

[1] Alan O.Freier, Philip Kocher, and Paul C.Kaltorn, "The SSL Protocol Version 3.0 draft", March 1996

[2] T.Dierks, C.Allen, The TLS Protocol Version 1.0, RFC2246, January 1999

[3] OpenSSL: http://www.openssl.org/

[4] John Viega, Matt Messier, Pravir Chandra 共著
齋藤　孝道 監訳
OpenSSL 暗号・PKI・SSL/TLS ライブラリの詳細

[5] Eric Rescorla 著
齋藤　孝道，鬼頭　利之，古森　貞 監訳
マスタリング TCP/IP SSL/TLS 編

[6] mod_ssl: http://www.modssl.org/