コンピュータセキュリティ　18−13
(2002. 7. 18)

SSL (Secure Socket Layer) のシステムとしての安全性の考察

鬼頭　利之†　　齋藤　孝道††

† 東京理科大学　〒278-8510 千葉県野田市山崎 2641
†† 東京工科大学　〒192-0982 東京都八王子市片倉町 1404-1
E-mail: †Toshiyuki.Kito@jcom.home.ne.jp, ††saito@cc.tec.ac.jp

あらまし　本論文では，SSL (Secure Socket Layer) の設計における安全性の考察を行い，ある状況で考慮すべき危険が存在することを示す．SSL における認証方式は，主に相互認証方式とサーバ認証方式があり，それぞれの方式とも安全な通信路を確立するためのセッション鍵の交換を行う．ここで，相互認証方式では，SSL サーバとユーザ双方が自身の秘密鍵を生成して，保持していなければならない．そのため，利便性よりサーバ認証方式も多用される．秘密鍵を生成して，保持していないユーザを認証する場合，サーバ認証方式を行い，その上で，交換したセッション鍵で暗号化したパスワードなどの認証情報を用いて認証する方式が用いられる．しかし，この方式を用いた場合，その認証情報の送信データの構造によっては，その認証情報が奪取される可能性がある．
キーワード　認証プロトコル，ネットワークセキュリティ

On the Architecture of the SSL System

Toshiyuki KITO† and Takamichi SAITO††

† Tokyo University of Science　Yamazaki 2641, Noda-shi, Chiba, 278-8510, Japan
†† Tokyo University of Technology　Katakura-machi 404-1, Hachioji-shi, Tokyo, 192-0982, Japan
E-mail: †Toshiyuki.Kito@jcom.home.ne.jp, ††saito@cc.tec.ac.jp

Abstract　Some flaws have been found out in the SSL, the Secure Socket Layer. Therefore, in this paper, we consider the SSL's architecuture and its total design of the SSL. There are mainly the way of authenticating between the SSL server and user, and that of only authentication the SSL server. Each authentication exchages session key, which can make establish secure communication between the SSL client and server. In authenticating of the SSL server and user, the SSL server and user must have the own secret key. For convenience, in the case authenticating a uer who does not hold secret key, the way of using password, encrypted by session key is used after server authentication. However, a user can be deprived of its password.
Key words　authentication protocol, network security

1. はじめに

近年，インターネットのコンピュータネットワークの普及にともない，電子メールや電子商取引など様々な応用が実現され，それによるさまざまな利益を享受している．その一方，ネットワークを流れるデータの盗み見や改ざんといった脅威が存在するようになった．そのため，特定の相手との内容を第三者に秘匿した通信を行う技術，つまり，認証プロトコルの技術が必要不可欠である．特定の相手との内容を第三者に秘匿した通信を行うためには，秘密の通信を行うための鍵であるセッション鍵を特定の相手と共有しなければならない．しかし，認証プロトコルの設計の仕方によっては，このセッション鍵が第三者に漏洩するといった問題や MITM (man in the middle) 攻撃を実現することができるプロトコルが存在するものもある [2]．本論文では，SSL (Secure Socket Layer) の安全性について議論する．

SSL [1] は，トランスポート層よりも上位層の HTTP や FTP などの通信相手の認証とデータの暗号化を行うことによって第三者によるデータの盗用や改ざんを防ぐ．SSL はサーバプログラムとクライアントプログラムから構成され，ドラフト [1] で記述されている．また，SSL は，SSL ハンドシェークプロトコル，SSL レコードプロトコル，SSL アラートプロトコルから構成されている．特に，SSL ハンドシェークプロトコルにおいて，セッション鍵の交換方法等の取り決めや通信相手の認証，セッション鍵の共有を行う．

SSL ハンドシェークプロトコルにおける認証方式は，サーバ

とユーザが秘密鍵を生成して，保持していることを確認することによって通信相手を認証する「相互認証方式」とサーバが秘密鍵を保持していることを確認することによってサーバのみを認証する「サーバ認証方式」と「お互いに認証しない方法」の3種類に分けられる．本論文では，相互認証方式とサーバ認証方式を扱う．相互認証方式は，サーバとユーザが自身の秘密鍵を生成して，保持していることが前提で，お互いに通信相手が秘密鍵を保持していることを確認して認証を行うことをいう．一方，サーバ認証方式は，サーバが秘密鍵を生成して，保持していることが前提で，ユーザはサーバが秘密鍵を保持していることを確認することによってサーバ認証をすることをいう．しかし，サーバはユーザ認証を行わないため，ユーザを特定することはできない．そのため，ユーザが秘密鍵を生成して，保持していない場合，サーバ認証方式を行った後，共有したセッション鍵を用いて，サーバはユーザしか知りえないパスワードなどの認証情報を確認することによってユーザを認証する．この方式は，ユーザ認証する前にセッション鍵の交換を行うことに注意する．要するに，SSLサーバはユーザ認証する手続きにおいて，SSLクライアントを識別することができないのである．そのため，状況によっては，SSLサーバは詐欺のSSLクライアントによって騙される可能性がある．

本論文では，SSLのハンドシェークプロトコルについて，RSA鍵交換方式とDiffie-Hellman鍵交換方式に分けてプロトコルの流れを示し，SSLの構造について考察する．

2. 用語と表記

主　体：

SSLは，SSLサーバとSSLクライアントから構成される．SSLサーバはSで表し，SSLサーバプログラムを主に実行する．また，SSLクライアントはCで表し，SSLクライアントプログラムを主に実行する．さらに，攻撃者はIによって表す．

通　報：

Msgは任意のメッセージを示す．CがMsgをSに送信する場合，以下のようにする：

$$C \rightarrow S \ : \ Msg$$

次に，CとSがMsgをお互いに送信し合うことを以下のように記述する：

$$C \leftrightarrow S \ : \ Msg$$

さらに，他の主体に成りすます攻撃者についての表現を示す．例えば，Sに成りすますIが，Cに対してMsgを送信する場合は，以下のように示す：

$$I(S) \rightarrow C \ : \ Msg$$

暗号化と鍵：

P_SはSの公開鍵を示し，P_CはCの公開鍵を示す．Msgを鍵Yで暗号化したものを$\{Msg\}_Y$と示し，公開鍵Yに対する秘密鍵Y^{-1}で電子署名されたMsgを$\{Msg\}_{Y^{-1}}$と示す．また，KS_{CS}はSとCの通信路間のセッション鍵とする．

Diffie-Hellman鍵交換：

pは大きく安全とされる素数であり，gはGF(p)の部分群に対する生成元である．また，qは部分群の位数である．乱数$y, x(1 \leqq x, y \leqq q)$は，$S$と$C$によってそれぞれ生成される．$Y_S$は$g^y \bmod p$であり，$Y_C$は，$g^x \bmod p$である．その上，$pre_master_secret_{CS}$は，セッション鍵の元となる$g^{xy} \bmod p$である．

その他：

SA_Cは，Cのプロトコルバージョン，Cによってサポートされている圧縮アルゴリズムのリスト，暗号オプションのリスト，鍵の交換方法から構成される．RN_CはCの現在時刻・日付と28バイトの乱数の構造体を示す．SA_Sは，Sのプロトコルバージョン，SA_Cから選択した圧縮アルゴリズム，暗号オプション，鍵の交換方法から構成される．RN_SはSの現在時刻・日付と28バイトの乱数の構造体を示す．S_Cは，Cのセッション識別子を示し，S_Sは，Sのセッション識別子を示す．$cert_list_S, cert_list_C$は，Sの証明書のリスト，Cの証明書のリストをそれぞれ示す．$cert_type$はCの証明書の種類を示すリストであり，$cert_auth$はSが証明書を受理することができる認証局のDistinguishedNameのリストである．

RSA鍵交換方式で用いられる$pre_master_secret_C$は，Cによって新たに生成される48バイトの乱数とCのプロトコルバージョンである．$message_{CS}, message_all_{CS}$は，$C$と$S$の間でこれを通報するまでに送信されたすべてのデータをそれぞれ示す．また，AckはSが必要なネゴシエーションを終了したことを示すメッセージである．

$f(Msg_1)$は，Msg_1から生成されるハッシュ値を表す．同様に，$f(Msg_1, Msg_2, \cdots, Msg_i)$は，$Msg_1, Msg_2, \cdots, Msg_i$を結合したハッシュ値を表す．

最後に，ユーザを認証するためのパスワードなどの認証情報を$password_C$で示す．

3. SSLハンドシェークプロトコル

SSLハンドシェークプロトコルの認証方式には，相互認証方式とサーバ認証方式がある．まず，相互認証方式について説明する．

SSLハンドシェークプロトコルにおける相互認証方式を開始すると，最初に，暗号アルゴリズム，圧縮アルゴリズム，鍵の交換方法をサーバとクライアントは取り決める．その後，サーバはクライアントにサーバの証明書を送信し，クライアントはそれが正しいかどうかを検証する．また，クライアントも同様に，サーバにユーザの証明書を送信し，サーバはそれが正しいものであることを検証する．ただし，本論文では，正しく対応した公開鍵を持つことを前提とする．次に，サーバとクライアント間の情報を暗号化するためのセッション鍵を共有する．この鍵交換方法は，RSA鍵交換方式，Diffie-Hellman鍵交換方式，FORTEZZA鍵交換方式がある．RSA鍵交換方式におけるサーバ認証は，クライアントがサーバの公開鍵で暗号化された情報をサーバが復号できたことを確認することである．また，ユーザ認証はユーザが公開鍵に対する秘密鍵で電子署名を行い，サーバがその電子署名を確認することである．これに対して，Diffie-Hellman鍵交換方式におけるサーバ認証は，サーバが公開鍵に対する秘密鍵で電子署名を行い，クライアント

がその電子署名を確認することである．ユーザ認証も同様に，サーバがユーザの電子署名を確認することによって認証を行う．FORTEZZA 鍵交換方式は，本論文では扱わない．最後に，サーバとクライアントがセッション鍵を共有していることをそれぞれ確認し，ハンドシェークプロトコルを終了する．

次に，SSL ハンドシェークプロトコルにおけるサーバ認証方式について説明する．相互認証方式と同様に，サーバとクライアントは，暗号アルゴリズム，圧縮アルゴリズム，鍵交換方法について取り決める．この後，サーバはクライアントにサーバの証明書を送信し，クライアントはそれが正しいかどうかを検証する．次に，サーバとクライアントは，セッション鍵を共有する．サーバ認証方式においては，RSA 鍵交換方式と Diffie-Hellman 鍵交換方式がある．RSA 鍵交換方式におけるサーバ認証は，クライアントがサーバの公開鍵で暗号化された情報をサーバが復号できたことを確認することである．また，Diffie-Hellman 鍵交換方式におけるサーバ認証は，サーバが公開鍵に対する秘密鍵で電子署名を行い，クライアントがその電子署名を確認することである．最後に，サーバとクライアントがセッション鍵を共有していることをそれぞれ確認し，ハンドシェークプロトコルを終了する．

上記に説明したとおり，ユーザの認証を行うためには，相互認証方式を使用しなければならない．しかし，相互認証方式を利用するためには，ユーザが秘密鍵を保持しなければならない．そのため，ユーザが秘密鍵を保持していない状況において，ユーザ認証を行う場合は，サーバ認証方式を行った後，ユーザ認証を行うことが多い．ユーザ認証の方法は，ユーザしか知りえないパスワードなどの認証情報をサーバ認証方式において共有したセッション鍵で暗号化し，これをサーバが確認することによって行われる．

3.1 RSA 鍵交換方式

M1) $C \rightarrow S : SA_C, S_C, RN_C$
M2) $S \rightarrow C : SA_S, S_S, RN_S$
M3) $S \rightarrow C : cert\_list_S$
M4) $S \rightarrow C : cert\_type, cert\_auth$
M5) $S \rightarrow C : Ack$
M6) $C \rightarrow S : cert\_list_C$
M7) $C \rightarrow S : \{pre\_master\_secret_C\}_{P_S}$
M8) $C \rightarrow S : \{h(KS_{CS}, h(message_{CS}, KS_{CS}))\}_{P_C^{-1}}$
M9) $C \leftrightarrow S : \{h(KS_{CS}, h(message\_all_{CS}, KS_{CS}))\}_{KCS}$

上記のプロトコルは，RSA 鍵交換方式を用いた SSL ハンドシェークプロトコルである．また，サーバ認証方式では，**M4**, **M6**, **M8** の通報を行わない．

最初に，クライアントがサーバに接続した時，クライアントは SA_C, S_C, RN_C を最初に送信する．S_C は，C が以前の接続で用いた暗号アルゴリズムとセッション鍵を再利用したい場合，以前のセッション識別子となる．新規にセッションパラメータの取り決めを行う場合，このフィールドは空となる（**M1**）．この後，S は C に SA_S, S_S, RN_S を送信する．S_S は S_C が空でない場合，S のキャッシュの中から一致するものを探す．一致するセッション識別子が存在し，以前の接続で用いた暗号アルゴリズムとセッション鍵を再利用したい場合，S_S は S_C と同じ値となる．しかし，S のキャッシュに存在しない場合，もしくは，新規のセッションを確立する場合には，S_S は別の値，または，空が入る（**M2**）．次に，S は $cert\_list_S$ を C に送信する（**M3**）．また，S は，C に対して証明書を要求するため，証明書が有効であるかどうかを確認することができる証明書の種類 $cert\_type$ と認証局のリスト $cert\_auth$ を送信する（**M4**）．この後，S は必要なネゴシエーションが終了したことを C に送信する（**M5**）．次に，C は S が示した $cert\_auth, cert\_type$ に基づいて C の $cert\_list_C$ を送信する（**M6**）．C は $pre\_master\_secret_C$ を生成し，それを S の公開鍵で暗号化し，S に送信する（**M7**）．この暗号化されたメッセージは S のみ復号できる．また，C は $pre\_master\_secret_C$ から KS_{CS} を生成する．さらに，C は $h(KS_{CS}, h(message_{CS}, KS_{CS}))$ を計算し，これに C の秘密鍵 P_C^{-1} を用いて電子署名をして，S に送信する（**M8**）．この通報を受け取った S は，$pre\_master\_secret_C$ からセッション鍵を生成し，C の署名が正しいものであることを確認する．最後に，S と C は鍵交換と認証処理が成功したことを確認することを示すメッセージをお互いに送信する（**M9**）．この通報が正しいことを C が確認できた場合，S の公開鍵で暗号化された $pre\_master\_secret_C$ を復号することで，C は S がサーバ証明書の公開鍵に対応する秘密鍵を保持していることを確認することができる．また，S と C のセッション鍵は $pre\_master\_secret_C$ から生成されるが，C が暗号化するセッション鍵と S が暗号化するセッション鍵は厳密にいうと異なっている．このため，**M9** において C から送信されるものと S から送信されるものは異なっている．

3.2 Diffie-Hellman 鍵交換方式

M1) $C \rightarrow S : SA_C, S_C, RN_C$
M2) $S \rightarrow C : SA_S, S_S, RN_S$
M3) $S \rightarrow C : cert\_list_S$
M4) $S \rightarrow C : p, g, Y_S, \{h(RN_C, RN_S, p, g, Y_S)\}_{P_S^{-1}}$
M5) $S \rightarrow C : cert\_type, cert\_auth$
M6) $S \rightarrow C : Ack$
M7) $C \rightarrow S : cert\_list_C$
M8) $C \rightarrow S : Y_C$
M9) $C \rightarrow S : \{h(KS_{CS}, h(message_{CS}, KS_{CS}))\}_{P_C^{-1}}$
M10) $C \leftrightarrow S : \{h(KS_{CS}, h(message\_all_{CS}, KS_{CS}))\}_{KCS}$

上記のプロトコルは，Diffie-Hellman 鍵交換方式を用いた SSL ハンドシェークプロトコルである．また，サーバ認証方式では，**M5**, **M7**, **M9** の通報を行わない．

まず始めに，RSA 鍵交換方式における **M1**, **M2** と同様に，C は SA_C, S_C, RN_C を S に送信し（**M1**），S は C に SA_S, S_S, RN_S を送信する（**M2**）．次に，S は C に $cert\_list_S$ を送信する（**M3**）．S は x を生成し，$Y_S (= g^x \bmod p)$ を計算し，p, g, Y_S とともに RN_C, RN_S, p, g, Y_S を連結したハッシュ値に S の秘密鍵 P_S^{-1} で電子署名をしたものを C に送信する（**M4**）．この通報は，**M3** において p, g, Y_S が S の証明書に記

載されていない場合に送信される．RSA 鍵交換方式と同様に，相互認証方式の場合，S は C に対して，クライアント証明書を要求する（**M5**）．この要求の後，S は必要なネゴシエーションが終了したことを C に送信する（**M6**）．次に，C は S が示した $cert\_type$, $cert\_authe$ に基づいて C の $cert\_list_C$ を送信する（**M7**）．この後，C は y を生成し，$Y_C(g^y \bmod p)$ を計算し，$pre\_master\_secret_{CS}(g^{xy} \bmod p)$ を生成する．さらに，C は $pre\_master\_secret_{CS}$ から KS_{CS} を生成する．その上で，C は $Y_S(=g^y \bmod p)$ を S に送信する．これは，C の証明書に Y_C が記載されていない場合に送信される（**M8**）．この通報を受け取った S は，同様に $pre\_master\_secret_{CS}(=g^{xy} \bmod p)$ を計算し，KS_{CS} を計算する．相互認証方式では，C は $h(KS_{CS}, h(message_{CS}, KS_{CS}))$ を計算し，これに C の秘密鍵 P_C^{-1} で電子署名を施し，S に送信される **(M9)**．S は，$h(KA_{CS}, h(message_{CS}, KS_{CS}))$ を計算し，**M7** で受け取った証明書に記載されている公開鍵で電子署名を確認する．最後に，S と C は，$h(KS_{CS}, h(message\_all_{CS}, KS_{CS}))$ を計算し，これを KS_{CS} で暗号化し，鍵交換と認証処理が成功したことを確認することを示すメッセージをお互いに送信する **(M10)**．また，S と C のセッション鍵は $pre\_master\_secret_{CS}$ から生成されるが，C が暗号化するセッション鍵と S が暗号化するセッション鍵は厳密にいうと異なっている．このため，**M10** において C から送信されるものと S から送信されるものは異なっている．

4. SSL ハンドシェークプロトコルの構造の欠陥

本節では，SSL のハンドシェークプロトコル構造の安全性について考察する．

4.1 安全性の考察

SSL ハンドシェークプロトコルにおける相互認証方式においては，ユーザ認証とサーバ認証を行った後，セッション鍵の交換を行っている．相互認証方式を行う場合は，サーバとユーザが自身の公開鍵とそれに対応した秘密鍵を生成して，保持していなければならない．そのため，ユーザが公開鍵と秘密鍵を生成して，保持していない場合，サーバ認証方式を行った後，パスワード等のユーザとサーバしか知らない認証情報を用いて認証を行うことが多い．しかし，この認証方式を用いた場合，パスワードをサーバ認証において共有したセッション鍵で暗号化しただけではサーバはユーザを識別することができない．

SSL サーバとクライアントはサーバ認証を行った後，パスワード等の認証情報を用いてユーザ認証を行った場合，安全な方法でセッション鍵を共有することの合意を得ることができていない．このため，文献[2]の安全な認証プロトコルの用件を満たしていないため，サーバとクライアント間で安全な通信路を確立することができない．そのため，サーバは MITM 攻撃に騙される可能性がある．これは，SSL ハンドシェークプロトコルにおけるサーバ認証方式では，サーバ認証をしているのでクライアント側からの通報を秘密にサーバに通報することができるが，サーバはユーザ認証をしていないため，誰から通報されたかを特定することができないためである．この攻撃方法は，例えば，クライアントが正規のサーバのミラーサーバであるとして偽っている攻撃者に接続する際の「正規のサーバを騙す MITM 攻撃」である．これは，G.Lowe の論文[3]で示されている．

サーバを騙す MITM 攻撃には，次に示す仮定が必要となる：クライアントは最初に正規のサーバのように振る舞っている攻撃者に接続をする．このクライアントの行動をおかしいと思うかもしれないが，たとえば，正規のサーバのミラーサーバであるとメールなどで告知された場合，ユーザはそれを正規のサーバと信じ，接続するユーザが存在するかもしれない．そのため，攻撃者を正規のサーバであると信じたユーザは，SSL ハンドシェークプロトコルにおけるサーバ認証を行い，それ以後の通報も正しく行ってしまう．ここで，最も重要なことは，攻撃者は正規のサーバが保持している資源をまったく保持していない，つまり，攻撃者がサーバのふりをしても，ユーザのホームにアクセスさせることはできない，ということであり，また，ユーザの秘密鍵を生成して，保持していないということである．以後，この仮定に基づいて議論する．

4.2 サーバ認証方式に対する攻撃

4.2.1 RSA 鍵交換方式を用いたサーバ認証方式

M1) $C \to I$: SA_C, S_C, RN_C
M1′) $I(C) \to S$: SA_I, S_I, RN_I
M2) $S \to I(C)$: SA_S, S_S, RN_S
M2′) $I \to C$: SA_I, S_I, RN_I
M3) $S \to I(C)$: $cert\_list_S$
M3′) $I \to C$: $cert\_list_I$
M4) $S \to I(C)$: Ack
M4′) $I \to C$: Ack
M5) $C \to I$: $\{pre\_master\_secret_C\}_{P_I}$
M5′) $I(C) \to S$: $\{pre\_master\_secret_I\}_{P_S}$
M6) $S \leftrightarrow I(C)$: $\{h(KS_{IS}, h(message\_all_{IS}, KS_{IS}))\}_{KS_{IS}}$
M6′) $C \leftrightarrow I$: $\{h(KS_{CI}, h(message\_all_{CI}, KS_{CI}))\}_{KS_{CI}}$

クライアント C と攻撃者 I，攻撃者 I とサーバ S は暗号アルゴリズム，圧縮アルゴリズム，鍵の交換方法を取り決める（**M1**～**M2′**）．この後，S は I に証明書のリスト $cert\_list_S$ を送信し（**M3**），同様に，I も C に証明書のリスト $cert\_list_I$ を送信する（**M3′**）．この後，S は送信すべき情報を送信し終えたことを送信する（**M4**）．I も同様に，C に送信すべき情報を送信し終えたことを送信する（**M4′**）．次に，C は $pre\_master\_secret_C$ を生成し，I の公開鍵 P_I でそれを暗号化して，I に送信する（**M5**）．この通報を受け取った I は，I の秘密鍵 P_I^{-1} でそれを復号し，$pre\_master\_secret_C$ を得る．同様に，I は $pre\_master\_secret_I$ を P_S で暗号化し，S に送信する（**M5′**）．また，$pre\_master\_secret_C, pre\_master\_secret_I$ を保持したそれぞれの主体は，それからセッション鍵を生成する．

ここで，C と I のセッション鍵は $pre\_master\_secret_C$ から生成され，I と S のセッション鍵は $pre\_master\_secret_I$ から生成されるため，各主体間で異なったセッション鍵となる．次に，C になりすました I と S はセッション鍵 KS_{IS} を共有したことを確認する（**M6**）．同様に，C と I はセッション鍵 KS_{CI} を共有したことを確認する（**M6′**）．よって，S は C に成りすましている I と正しくサーバ認証方式を終え，同様に，C は I と正しくサーバ認証方式を終えている．そのため，サーバ認証方式を終了した後，ユーザのパスワードなどの認証情報を用いて認証する場合，C によってその認証情報がセッション鍵 KS_{CI} で暗号化されただけでは，I はセッション鍵 KS_{CI} で復号することでその認証情報を手に入れることができる．その上で，C から受け取った認証情報をセッション鍵 KS_{IS} で暗号化し，S に送信することで，S は接続してきたのは I であるにもかかわらず，C であると認識してしまう．その例を以下の **M7**，**M7′** において示す．

M7） $C \rightarrow I$: $\{password_C\}_{KS_{CI}}$
M7′） $I(C) \rightarrow S$: $\{password_C\}_{KS_{IS}}$

4.2.2 Diffie-Helman 鍵交換方式を用いたサーバ認証方式

M1） $C \rightarrow I$: SA_C, S_C, RN_C
M1′） $I(C) \rightarrow S$: SA_I, S_I, RN_I
M2） $S \rightarrow I(C)$: SA_S, S_S, RN_S
M2′） $I \rightarrow C$: SA_I, S_I, RN_I
M3） $S \rightarrow I(C)$: $cert\_list_S$
M3′） $I \rightarrow C$: $cert\_list_I$
M4） $S \rightarrow I(C)$: $p, g, Y_S, \{h(RN_I, RN_S, p, g, Y_S)\}_{P_S^{-1}}$
M4′） $I \rightarrow C$: $p, g, Y_I, \{h(RN_C, RN_I, p, g, Y_I)\}_{P_I^{-1}}$
M5） $S \rightarrow I(C)$: Ack
M5′） $I \rightarrow C$: Ack
M6） $C \rightarrow I$: Y_C
M6′） $I(C) \rightarrow S$: Y_I
M7） $C \leftrightarrow I$: $\{h(KS_{CI}, h(message\_all_{CI}, KS_{CI}))\}_{K_{CI}}$
M7′） $S \leftrightarrow I(C)$: $\{h(KS_{IS}, h(message\_all_{IS}, KS_{IS}))\}_{K_{IS}}$

RSA 鍵交換方式の **M1**～**M2′** と同様に，クライアント C と攻撃者 I，攻撃者 I とサーバ S は暗号アルゴリズム，圧縮アルゴリズム，鍵の交換方法を取り決める（**M1**～**M2′**）．この後，S は I に S の証明書のリスト $cert\_list_S$ を送信し（**M3**），同様に，I も C に I の証明書のリスト $cert\_list_I$ を送信する（**M3′**）．次に，S は Y_S を計算し，p, g, Y_S と P_S^{-1} で電子署名をした乱数 RN_I, RN_S, p, g, Y_S のハッシュ値を I に送信する（**M4**）．また，I も同様に Y_I を計算し，p, g, Y_I と P_I^{-1} で電子署名をした乱数 RN_C, RN_C, p, g, Y_I のハッシュ値を C に送信する（**M4′**）．S は送信すべき情報を送信し終えたことを送信する（**M5**）．I も同様に，C に送信すべき情報を送信し終えたことを送信する（**M5′**）．次に，C は Y_C を計算し，それを I に送信する（**M6**）．同様に，I も Y_I を計算し，それを S に送信する（**M6′**）．この後，C と I はセッション鍵 KS_{CI} を共有したことを確認する（**M7**）．また，同様に，C に成りすました I と S もセッション鍵 KS_{IS} を共有したことを確認する（**M7′**）．ここで，C と I のセッション鍵は $pre\_master\_secret_{CI}$ から生成され，I と S のセッション鍵は $pre\_master\_secret_{IS}$ から生成されるため，各主体間で異なったセッション鍵となる．よって，S は C に成りすましている I と正しくサーバ認証方式を終え，同様に，C は I と正しくサーバ認証方式を終えている．そのため，サーバ認証方式を終了した後，ユーザのパスワードなどの認証情報を用いて認証する場合，C によってその認証情報がセッション鍵 KS_{CI} で暗号化されただけでは，I はセッション鍵 KS_{CI} で復号することでその認証情報を手に入れることができる．この後，C から受け取った認証情報をセッション鍵 KS_{IS} で暗号化し，S に送信することで，S は接続してきたのは I であるにもかかわらず，C であると認識してしまう．その例を以下の **M8**，**M8′** において示す．

M8） $C \rightarrow I$: $\{password_C\}_{KS_{CI}}$
M8′） $I(C) \rightarrow S$: $\{password_C\}_{KS_{IS}}$

4.3 相互認証方式に対する攻撃

4.3.1 RSA 鍵交換方式を用いた相互認証方式

M1） $C \rightarrow I$: SA_C, S_C, RN_C
M1′） $I(C) \rightarrow S$: SA_I, S_I, RN_I
M2） $S \rightarrow I(C)$: SA_S, S_S, RN_S
M2′） $I \rightarrow C$: SA_I, S_I, RN_I
M3） $S \rightarrow I(C)$: $cert\_list_S$
M3′） $I \rightarrow C$: $cert\_list_I$
M4） $S \rightarrow I(C)$: $cert\_type, cert\_auth$
M4′） $I \rightarrow C$: $cert\_type, cert\_auth$
M5） $S \rightarrow I(C)$: Ack
M5′） $I \rightarrow C$: Ack
M6） $C \rightarrow I$: $cert\_list_C$
M6′） $I(C) \rightarrow S$: $cert\_list_C$
M7） $C \rightarrow I$: $\{pre\_master\_secret_C\}_{P_I}$
M7′） $I(C) \rightarrow S$: $\{pre\_master\_secret_I\}_{P_S}$
M8） $C \rightarrow I$: $\{h(KS_{CI}, h(message_{CI}, KS_{CI}))\}_{P_C^{-1}}$

クライアント C と攻撃者 I，攻撃者 I とサーバ S は暗号アルゴリズム，圧縮アルゴリズム，鍵の交換方法を取り決める（**M1**～**M2′**）．この後，S は I に S の証明書のリスト $cert\_list_S$ を送信し（**M3**），同様に，I も C に証明書のリスト $cert\_list_I$ を送信する（**M3′**）．次に，S は I に対して $cert\_type, cert\_auth$ で示した証明書を要求する（**M4**）．I は C に成りすますため，S に要求された $cert\_type, cert\_auth$ を C に要求する（**M4′**）．この後，S は I にネゴシエーションが終了したことを送信し（**M5**），I は同様に，C にネゴシエーションが終了したことを送信する（**M5′**）．次に，C は I に $cert\_list_C$ を送信する

(**M6**)．この通報を受け取った I は，$cert\_list_C$ を S に送信する（**M6′**）．次に，C は，$pre\_master\_secret_C$ を I の公開鍵で暗号化し，これを I に送信し，$pre\_maser\_secret_C$ からセッション鍵 KS_{CI} を生成する（**M7**）．この通報を受け取った I は，秘密鍵 P_I^{-1} で復号し，$pre\_master\_secret_C$ を得る．その上で，I は $pre\_master\_secret_C$ から KS_{CI} を生成する．I も同様に，S に $pre\_master\_secret_I$ を S の公開鍵で暗号化し，S に送信し，$pre\_master\_secret_I$ から KS_{IS} を生成する（**M7′**）．これを受け取った S は $pre\_master\_secret_I$ から KS_{IS} を生成する．次に，C は $\{h(KS_{IS}, h(message_{IS}, KS_{IS}))\}$ に C の秘密鍵 P_C^{-1} で電子署名したものを I に送信する（**M8**）．ここで，I は C に成りすますために $\{h(KS_{IS}, h(message_{IS}, KS_{IS}))\}_{P_C^{-1}}$ を作成しなければならない．しかし，I は C の P_C^{-1} を保持していないため，その通報を作成できない．また，送信すべき通報と C の通報とでは，$message_{CI}$ と $message_{IS}$ が異なっている．そのため，RSA 鍵交換方式を用いた相互認証方式は攻撃者の攻撃を防ぐことができるが，この通報まで攻撃者の存在を発見することはできない．

4.3.2 Diffie-Helman 鍵交換方式を用いた相互認証方式

M1）	$C \rightarrow I$	: SA_C, S_C, RN_C
M1′）	$I(C) \rightarrow S$	: SA_I, S_I, RN_I
M2）	$S \rightarrow I(C)$	: SA_S, S_S, RN_S
M2′）	$I \rightarrow C$	: SA_I, S_I, RN_I
M3）	$S \rightarrow I(C)$	: $cert\_list_S$
M3′）	$I \rightarrow C$	: $cert\_list_I$
M4）	$S \rightarrow I(C)$	: $p, g, Y_S, \{h(RN_I, RN_S, p, g, Y_S)\}_{P_S^{-1}}$
M4′）	$I \rightarrow C$	: $p, g, Y_I, \{h(RN_C, RN_I, p, g, Y_I)\}_{P_I^{-1}}$
M5）	$S \rightarrow I(C)$	: $cert\_type, cert\_auth$
M5′）	$I \rightarrow C$	: $cert\_type, cert\_auth$
M6）	$S \rightarrow I(C)$	: Ack
M6′）	$I \rightarrow C$	: Ack
M7）	$C \rightarrow I$	: $cert\_list_C$
M7′）	$I(C) \rightarrow S$	: $cert\_list_C$
M8）	$C \rightarrow I$	: Y_C
M8′）	$I(C) \rightarrow S$	: Y_I
M9）	$C \rightarrow I$	: $\{h(KS_{CI}, h(message_{CI}, KS_{CI}))\}_{P_C^{-1}}$

RSA 鍵交換方式の **M1**～**M2′** と同様に，クライアント C と攻撃者 I，攻撃者 I とサーバ S は暗号アルゴリズム，圧縮アルゴリズム，鍵の交換方法を取り決める（**M1**～**M2′**）．この後，S は I に S の証明書のリスト $cert\_list_S$ を送信し（**M3**），同様に，I も C に証明書のリスト $cert\_list_I$ を送信する（**M3′**）．次に，C に成りすました I と S は **M4** と **M8′** においてセッション鍵 KS_{IS} を共有する．C と I は **M4′** と **M8** においてセッション鍵 KS_{CI} を共有する．さらに，**M4** において I は S を認証するが，I は意図的に S に接続をするのでこの認証は成立する．また，**M4′** において C は I を認証するが，C は I に接続しているのでこの認証は成立する．次に，**M9** において C は $\{h(KS_{CI}, h(message_{CI}, KS_{CI}))\}$ に C の秘密鍵 P_C^{-1} で電子署名したものを送信する．I は C に成りすますために $\{h(KS_{IS}, h(message_{IS}, KS_{IS}))\}_{P_C^{-1}}$ を作成しなければならない．しかし，I は C の秘密鍵 P_C^{-1} を保持していないため，その通報を作成することができない．また，送信すべき通報と C の通報とでは，$message_{CI}$ と $message_{IS}$ が異なっている．そのため，Deffie-Hellman 鍵交換方式を用いた相互認証方式は攻撃者の攻撃を防ぐが，この通報まで攻撃者の存在を発見できない．

5. ま と め

本論文では，SSL ハンドシェークプロトコルを利用してネットワークシステムを設計する際の安全性について考察を行った．その結果，SSL ハンドシェークプロトコルのサーバ認証方式を行った後，ユーザのパスワードなどの認証情報を用いてユーザの認証を行った場合，その認証情報をセッション鍵で暗号化しただけでは，攻撃者にその認証情報を奪取され，成りすましをされる可能性があることを示した．すなわち，セッション鍵で暗号化されているから安全であるということはいえない．そのため，このような設計を行う際には，十分注意する必要がある．また，相互認証方式を用いた場合は，サーバを騙す MITM 攻撃を防ぐことはできるが，ユーザを認証するまで攻撃者を特定できないことも示した．サーバを騙す MITM 攻撃を行った際に生じるこれらの問題は，プロトコルの欠陥が原因で生じるため，SSL の設計上で避けることができる．

ユーザ認証におけるサーバを騙す MITM 攻撃を防ぐためには，以下のような修正がユーザ認証において必要となる：

（1） 平文のパスワードを通報しない

（2） サーバがチャレンジレスポンスを利用する

(1) は，セッション鍵が信用できないからである．(2) は，サーバがパスワードで暗号化されたランダムなチャレンジを送信した後，適切に修正されたレスポンスを受け取る場合，サーバはクライアントを識別することができるからである．

文　献

[1] A.Freier,P.Kocher,and P.Kaltorn:The SSL Protocol Version 3.0, http://home.netscape.com/eng/ssl3/draft302.txt

[2] 斎藤孝道，萩谷昌巳，溝口文雄：公開鍵を用いた認証プロトコルについて，情報処理学会　論文誌，Vol.42, No8, pp2040-2048.

[3] G.Lowe:Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR, In T.Margaria and B.steffen, editors , *Tools and Algorithms for the Construction and analysis of Systems. Second International Workshop, TACAS* '96, Margaria, T.and Steffen, B.(Eds.), LNCS, Vol.1055, pp.147-166, 1996.