マルチメディア通信と分散処理　106－5
コンピュータセキュリティ　16－5
（2002．2．14）

SSLにおけるモバイル向け鍵管理システム

西郷　悟　　三浦　史光　　高橋　修
（株）NTT DoCoMo マルチメディア研究所

あらまし　SSL[1]は公開鍵認証により相手認証を行うため、正規ユーザのみが秘密鍵を利用出来る仕組みが重要である。これには例えばクライアント端末がユーザ認証を行い、秘密鍵を用いた演算の可否を制御する方法等があるが、クライアント端末が秘密鍵を保管している限り、盗難時に保管している秘密鍵を漏洩することを防止することはできない。また、ユーザ認証と公開鍵認証のつなぎ目がセキュリティホールとなる可能性もある。そこで、本稿では、信頼できる第三者機関（以降、検証センタ）を設け、秘密分散法[2]を用いて秘密鍵を管理することにより、上記被害を防止することのできる鍵管理システムを提案する。

キーワード：SSL、ユーザ認証、鍵管理、公開鍵認証、秘密分散

A Study on SSL Key Management System under Mobile situation

Satoru Saigo　　Fumiaki Miura　　Osamu Takahashi
NTT DoCoMo Multimedia Laboratories Co., Ltd.

Abstract　A SSL [1] authentication is based on public key cryptography. So it is required in SSL to allow only an appropriate user to use his secret key. An example of this key management system is that a client terminal authenticates user and decided whether carry out an operation by the secret key or not. But as and when the client terminal has the secret key, it is impossible to protect it against theft completely, and a connection between user authentication and SSL authentication may be a security hole. This paper proposes a new key management system to prevent above problems, by introducing a TTP (such TTP are henceforth referred to as "verification center") and a secret sharing [2].

Keywords：SSL、user authentication、key management、public key authentication、secret sharing

1．はじめに

HTTP[3]を用いたネットビジネスにおける取引情報/顧客情報の保護には、現在SSLが最もよく利用されている。また、最近ではブラウザフォンにもSSLが実装され始めてきており、モバイル環境でもSSLが利用される傾向が強まってきている。ところで、よく知られているようにSSLは公開鍵認証によりクライアント認証/サーバ認証を行うことが可能である。しかし、現状では、サーバ認証が非常によく利用されているのに対してクライアント認証はほとんど利用されていない。例えば、アクセス制御やログイン等クライアント認証が使える場面においても、パスワード入力等あまり信頼性が高くなく、使い勝手がよくない方法が別途用いられることが多い。この要因としては、一般ユーザにとって秘密鍵管理等、公開鍵認証を適切に運用するのが困難であることが挙げられる。

そこで、本研究では公開鍵認証の運用が特に困難なモバイル環境においても、SSLクライアント認証をユーザに負担をかけることなく簡単かつ適切に利用してもらうための鍵管理システムを提案する。

2．課題および検討

本稿では、公開鍵認証を簡単かつ適切に運用するために次の3つの課題に注目した。まず一つ目の課題は、正規ユーザのみが秘密鍵を利用出来る仕組みである。これには運用面におけるユーザ負担を軽減するために簡単な手続きで行える仕組みが求められる。二つ目の課題は、秘密情報の適切な保管方法である。これには、現在耐タンパ性のデバイスを使った方法がよく利用されているが、耐タンパ性のデバイスは内容検知/改竄を困難にはしてくれるものの完全に防止出来るわけではない。そのためモバイル環境においては、耐タンパ性のデイバスだけには依存しない秘密情報の保管方法がより望ましい。三つ目の課題は、詳細については2.3.節で述べるが、安全かつ高速な鍵管理システムが要求される。以降、それぞれの課題に対するアプローチ方法を検討する。

2．1．正規ユーザのみが秘密鍵を利用出来る仕組み

1.で述べたようにSSLは公開鍵認証によりクライアント端末を認証することが可能である。しかし、それを利用しているユーザまで認証することはできない。そこで、公開鍵認証時にクライアント端末が更にユーザ認証を行い、正規ユーザとして認証したときのみ秘密鍵を用いた演算処理を行う方法を用いる。これにより正規ユーザがクライアント端末を利用している場合のみ SSL クライアント認証が成立する仕組みを実現することができる（ただし、この際あらましで述べた秘密鍵の保管方法が心配されるが、これに関しては次節で述べる。）

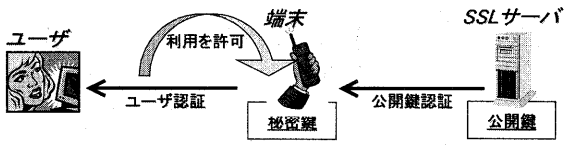

図1:ユーザ認証の公開鍵認証の併用

2．2．秘密情報の適切な保管方法

バイオ認証は誤差のある測定値に基づく認証方式であるため、現段階では公開鍵認証のような演算による認証を行うことができない。そのため、バイオ認証は照合による相手認証を行っており、ユーザ認証を行う側が事前に生体情報の登録値を保有しておく必要がある。したがって、バイオ認証とSSL公開鍵認証を用いた場合、正規ユーザ以外の第三者に漏洩してはいけない秘密情報としてユーザの秘密鍵と生体情報の登録値の二つが挙げられる。それぞれの保管方法に関する検討結果を以下に示す。

2．2．1．秘密鍵の適切な保管方法

公開鍵認証を適切に運用するには、秘密鍵の保管方法が重要な課題の一つである。これには、例えば入退室管理等物理的に端末へのアクセスを制限する方法などがよく用いられているが、ブラウザフォンのように持ち運ぶことを前提としている端末にはこの方法を用いることができない。したがって、モバイル環境においては秘密鍵の保管方法が特に難しくなる。そこで本研究では、これに対して秘密分散法を利用した秘密鍵の保管方法を提案する。秘密分散法を用いることにより、秘密鍵を 2 つに分散し、単独では元の秘密鍵を復元することのできない分散鍵に分散する。ただし、秘密鍵を用いて演算した結果と分散鍵を用いて順次演算した結果とが等しくなるように鍵を分散する。これらを端末及び新たに設けた検証センタにそれぞれ保管することにより、もしどちらか一方の鍵が盗まれた場合においても、素の秘密鍵を復元するためのいかなる情報も分散鍵からは得られないため秘密鍵が漏洩する危険性を防止することができる。

2．2．2．生体情報の適切な保管方法

一方、生体情報の保管方法に関しては、2.2.で述べたように、バイオ認証は演算を用いた認証を行うことができないため、秘密鍵の保管方法のように秘密分散法を用いることができない。そのため、分散鍵を保管するクライアント端末/検証センタのどちらかが生体情報を保管する必要がある。そこで、本研究では盗まれる危険性の低い検証センタに生体情報を保管し、検証センタ・ユーザ間のバイオ認証の結果により、検証センタ内の分散鍵を用いた演算処理の可否を制御することとした。

2．3．安全かつ高速なプロトコル

以上の条件を基に導出したアーキテクチャの概要図を図 2 に示す。検証センタ・ユーザ間でバイオ認証を実施するため、両者の間では生の生体情報がやり取りされる。そのため、検証センタ・クライアント端末間の通信路には安全性が求められることになる。また、SSL 相互認証を実施する間に、バイオ認証及び分散鍵を用いた演算処理を実施しなければならないため、検証センタ・端末間のプロトコルには高速性も要求される。本研究では、現行のブラウザフォンにおけるSSL サーバ認証に要する時間と同程度（～10 秒）で実行できる検証センタ・クライアント端末間のプロトコルを考案することを目標とした。

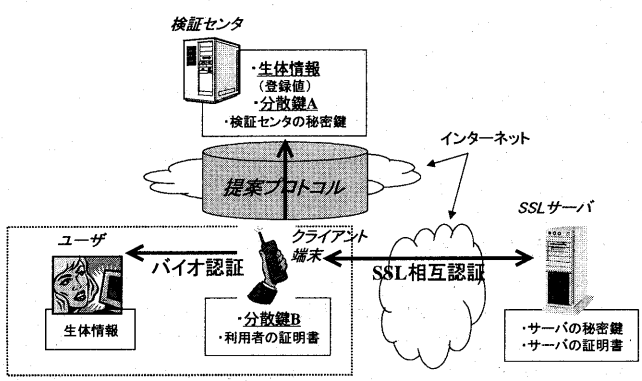

図2:提案アーキテクチャ

3．提案方式

3．1．提案方式におけるユーザ認証手順

図3に上記アーキテクチャを用いたユーザ認証手順を示す。秘密鍵は秘密分散法により検証センタ・端末に分散して保管する。そのため、SSLサーバからのチャレンジに対しては、両者が協力して正しレスポンスを演算しなければならない。クライアント端末は、検証センタとの間に安全な通信路を確保した後にユーザの生体情報の測定値及びSSLサーバからのチャレンジを検証センタに送信する。安全な通信路の確保については次節で述べる。検証センタは、受信した生体情報の測定値を基にバイオ認証を実施し、クライアント端末を使用しているユーザの認証を行う。このとき正規ユーザとして認証されたときのみ、検証センタは保管している分散鍵を用いてチャレンジに対する演算を行い、演算結果をクライアント端末に送信する。クライアント端末は、受信した演算結果を更にクライアント端末内の分散鍵で演算することによりチャレンジに対する正しいレスポンスを計算することができる。つまり、この認証手順を用いることにより正規ユーザがクライアント端末を利用している場合のみSSLクライアント認証が成立する仕組みを実現することが可能となる。

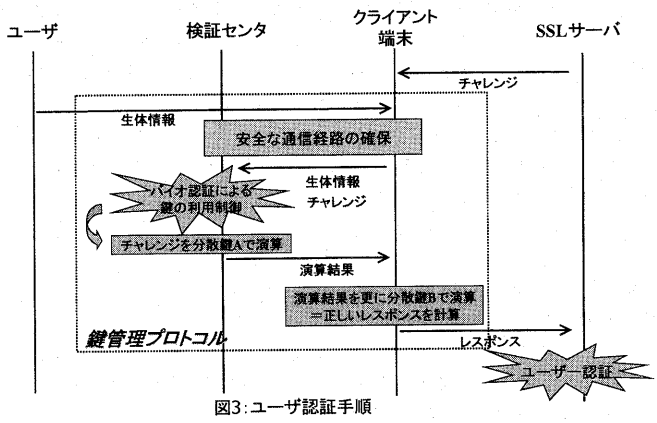

図3:ユーザ認証手順

3．2．鍵管理プロトコル（安全性・高速性）

上記ユーザ認証手順には検証センタ・クライアント端末間の安全な通信路の確保及び高速性が求められる。そこで、本研究ではこれらの条件を満たす検証センタ・クライアント端末間のプロトコルを新たに提案した。以降、これを鍵管理プロトコル1、2と呼ぶことにする。比較対象として検証センタ・クライアント端末間にもSSLを使用した場合を挙げる。これらのシーケンスを図4に示す。まず、SSLを用いた場合、検証センタ・クライアント端末

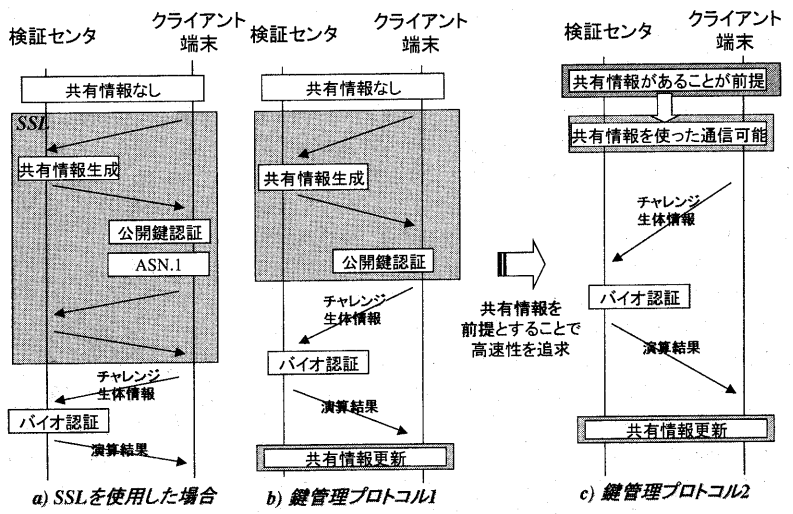

図4:提案アーキテクチャ

間に事前に共通鍵、MAC鍵、IV等の共有情報が存在しないため、両者の間でこれらの共有情報を生成する。この際、公開鍵認証により端末は検証センタを認証する。両者の間でネゴシエーションした内容に対する同意確認を行った後、これらを使用して前節で述べたユーザ認証手順を実施する。SSLを始めとし公開鍵認証を用いて不特定

多数相手を認証可能なプロトコルでは、相手を特定するために電子証明書の中身を解読するデコード処理が必要となる。これには ASN.1[4]が用いられているが、この処理作業は計算量を大幅に増加させる[5]。しかし、求められているのは特定の検証センタとの通信のみに使用するプロトコルであるため、ASN.1 の処理を事前に行っておくことができ、鍵管理プロトコルからは省くことが可能である。また、SSL ではネゴシエーション内容の同意確認を行っているが、実際に再度ネゴシエーションのやり直しが要求されるケースはほとんど発生しない。そこで、ネゴシエーション内容に問題があったときのみ、ネゴシエーションの再要求を行うことで、高速性を向上させることができる。以上のことを踏まえて安全性を強化したのが図 4 に示した鍵管理プロトコル 1 である。暗号化の対象を生体情報/乱数等の第三者が知りえない情報のみに限定することにより既知平文攻撃に対する耐性を向上させた（詳細については付録 A を参照）。また、鍵管理プロトコル 1 に、次回のセッションで使用する共有情報を更新するための乱数を検証センタ・クライアント端末間で共有する手順を追加することにより、更に高速性を追求した鍵管理プロトコル 2 を次回以降のセッションで用いることができる。検証センタ・クライアント端末は、乱数をハッシュ鍵として前回使用した共有情報のハッシュ値をとることにより、次に使用する共有情報を更新する。つまり、事前に共有情報が整っている状態から通信が始められるので、検証センタ・クライアント端末間でいきなりユーザ認証手順から開始することができる。これにより、計算量、RTT 共に大幅に削減することが可能となる。以上のことから、検証センタ・クライアント端末間の共有情報が使えない場合には鍵管理プロトコル 1 を、使える場合には鍵管理プロトコル 2 を用いるといった使い分けを行うことにより安全性/高速性の両方の条件を満たすことが可能である。

4．性能評価

本章では、上記で紹介した鍵管理プロトコルの高速性及び安全性の評価を行う。比較対象として、検証センタ・クライアント端末間で SSL を用いた場合を挙げる。

4．1．高速性の評価

高速性に関しては、SSL が実装されているブラウザフォン（F503i：NTT DoCoMo）を用いた測定値と、CRYPTO4.0++の計算量比較[5]を用いて行った。まず HTTP サーバまでのアクセス時間から RTT を概算し、SSL 認証時間から計算のみに要する時間を求めた。ここで、サーバの CPU の演算能力はクライアント端末の CPU の演算能力より十分大きいものと仮定した。この結果から、計算量と計算時間の対応関係を導き、CRYPTO4.0++の計算量比較を用いて鍵管理プロトコルに要する計算時間を求め、必要な RTT 数を加算することにより鍵管理プロトコルに要する認証時間を導出した。計算式を以下に示す。

鍵管理プロトコルの認証時間（*F503i*換算）
= 鍵管理プロトコルの計算量(*Crypto*換算)×SSL認証の計算時間（*F503i*換算）/SSL認証の計算量（*Crypto*換算）+ *nRTT*

計算結果を図 5 に示す。縦軸に認証時間、横軸に CPU の演算能力を示す。認証時間の 8～9 割を計算時間が占有していることから CPU の演算能力の向上により認証時間を大幅に削減できることが分かる。しかしながら、現行のブラウザフォンの演算能力では、バイオ認証及び分散鍵演算を除く SSL 相互認証単体でも時間がかかり、“現行の SSL サーバ認証時間と同程度”という目標レベルに達するには、これだけでも 2 倍以上 CPU の演算能力が必要となる。鍵管理プロトコルが目標レベルに達するには、これから更に 2 倍以上の CPU の向上が求められるが、今後もムーアの法則に従い CPU の演算能力が向上すれば 3 年後には鍵管理プロトコルが目標レベルに達すると考えられる。

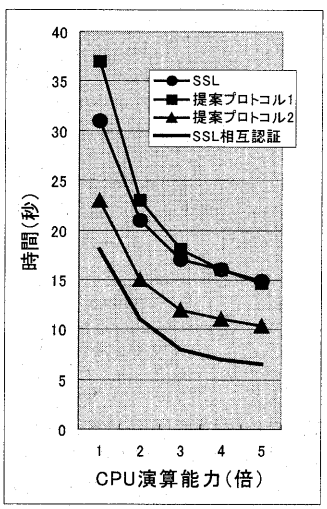

図5：高速性評価

4．2．安全性の評価

安全性に関する評価だが、本稿ではプロトコルに対する主流な攻撃方法である DoS 攻撃/リプレイ攻撃、及び暗号アルゴリズムに対する攻撃方法である選択平文攻撃/既知平文攻撃に着目しそれぞれに対する耐性を表に示した。まずリプレイ攻撃に対しては、SSL 同様本鍵管理プロトコルにおいても、シーケンス番号を MAC に組み込むことで防止する。また DoS 攻撃に関しては、DoS 攻撃を検出するまでの時間が攻撃への耐性の

目一つの目安となるが、SSL/鍵管理プロトコル1では検出までに7秒程度かかるのに対して、鍵管理プロトコル2では2秒程度であることが分かった。ただし、この値はCPUの演算能力が現行の4倍向上した場合の時間である。ところで、SSLはHTTPの通信内容全てを暗号化するために、Webページやコンテンツ等といった公開されている情報にも暗号化が利用される。そこで、SSLではIVを用いることにより既知平文攻撃を困難にしている。一方、鍵管理プロトコル1,2においてもIVを用いるが、暗号化の対象をそもそも第三者が知りえない情報に限定することにより既知平文攻撃を完全に防止する。最後に選択平文攻撃に関しては、SSL及び鍵管理プロトコルでは、チャレンジ/レスポンスに使用するチャレンジを検証センタ・クライアント端末間双方で生成することにより、選択平文攻撃を防止する。また、鍵管理プロトコル2に関しても、プリシェアード情報により認証を行うため選択平文攻撃を防止することができる。以上のことから、鍵管理プロトコル2を用いるのが高速性/安全性ともに望ましいことが分かる。ただし、鍵管理プロトコル2は前回利用した共有情報を更新できることを前提とするため、実際にはまず鍵管理プロトコル1を使って共有情報を生成し、以降鍵管理プロトコル2を用いるという使い分けを行うことで高速性/安全性の要求条件を満たすと考えられる。

表1．安全性の評価

	プロトコルに対する攻撃		暗号アルゴリズムに対する攻撃	
	リプレイ攻撃	DoS攻撃	既知平文攻撃	選択平文攻撃
SSLサーバ認証・バイオ認証	○（シーケンスにより検出可能）	△（7秒で検出）	○（IVのみに依存）	○（選択メッセージ自体存在しない）
鍵管理プロトコル1	○（シーケンスにより検出可能）	△（7秒で検出）	◎（既知メッセージ自体がない）	○（選択メッセージ自体存在しない）
鍵管理プロトコル2	○（シーケンスにより検出可能）	○（2秒で検出）	◎（既知メッセージ自体がない）	○（選択メッセージ自体が存在しない）

5．まとめ

本研究は、モバイル環境においてもユーザにとって簡単かつ適切にSSL公開鍵認証を運用することのできるシステムを実現できた。秘密鍵に関しては秘密分散法を用いて検証センタ・クライアント端末に保管し、生体情報に関しては盗まれる危険性の低い検証センタに保管することにより、クライアント端末盗難時においても秘密鍵/生体情報が漏洩しないアーキテクチャを提案した。また、その際に問題となる、検証センタ・クライアント端末間の安全かつ高速な通信路の確保においても、鍵管理プロトコルを用いることで解決することができた。

6．今後の課題

本稿で提案したアーキテクチャは、公開鍵認証を用いるシステム全般で使用することができる。そこで、今後の課題として、検証センタ・クライアント端末間の鍵管理プロトコルをSSL以外でも使用できるように拡張する。この場合、次の二つの課題が問題となると考えられる。まず、一つ目の課題は検証センタに演算を必要とするメッセージを送信する際、検証センタにメッセージの内容を隠匿する必要がある場合にはブラインド署名等の適用を検討する必要がある。また、検証センタが分散した演算結果をクライアント端末で検証する必要がある場合には、計算時間が大きく増加するためこれに対する解決策が必要となる。以上の課題はSSLの仕様では問題とならないが、公開鍵認証一般に拡張する場合には検討の必要がある。

参考文献

[1] Alan O. Freier et.al., "The SSL Protocol Version 3.0", Internet Draft, November 18, 1996
[2] 岡本龍明、山本博資、"現代暗号"、産業図書
[3] R. Fielding et.al.,"Hypertext Transfer Protocol –HTTP/1.1" RFC2616, June 1999
[4] 森野和好、戸部美香著、"プロトコル構文規定言語－ASN.1"
[5] Eric Rescorla, "SSL and TLS -Designing and Building Secure Systems"
[6] Crypto++4.0 Benchmarks

付録A

図に RSA を用いて共通鍵、MAC 鍵、IV 等を共有した場合のシーケンスを示す。検証センタ/クライアント端末の保有している分散鍵をそれぞれ SK_A/SK_B とする。