情報処理学会論文誌 Vol. 50 No. 1 421–429 (Jan. 2009)

Ajaxを用いた SSHクライアントシステムの提案と実装

小須田 優介[†1] 佐々木 良一[†1]

ウェブアプリケーションにデスクトップアプリケーションに迫る機能や操作性をもたらすAjaxという技術がある．本研究ではこのAjaxを用いて，OSやJavaの実行環境などに依存しないSSHクライアントシステムの提案と実装を行う．JavaScriptにより，ウェブブラウザ上でSSHパケットを生成し，ウェブサーバにSSHサーバへのTCP通信を代理してもらうことで，エンドツーエンドのSSH通信を実現する．プロトタイプを実装し，動作実験を行ったところ，複数のウェブブラウザと携帯電話で動作することを確認した．このようなシステムを用いることで，インターネットカフェのPCや，任意の携帯電話などからもSSH通信が可能となった．また，処理性能の測定を行い，速度的には十分でないケースもあるが，緊急時などには十分使いうることを確認した．

Proposal and Implementation of SSH Client System Using Ajax

Yusuke Kosuda[†1] and Ryoichi Sasaki[†1]

Ajax is used to make web applications behave as if they are desktop applications. In this paper, we propose and implement SSH client system using Ajax which is independent of OS or Java runtime environment. In our system, SSH packets are generated by JavaScript on a web browser, and a web server acts as proxy to send the packets to SSH server. Thus end-to-end SSH communication is realized by achieving above function. The result of experiment shows that the system is working correctly on web browsers in PCs and mobile-phones. Moreover, the performance in PCs and mobile-phones is measured. The results show that the processing speed is sufficient in an emergency, although it is not sufficient to use regularly in some environment.

1. はじめに

近年，Ajax（Asynchronous JavaScript + XML，詳しくは2.1節参照）と呼ばれる技術を用いたウェブアプリケーションが数多く開発されている．Googleマップ[1)]などに代表されるこれらは，「ウェブ2.0」のアプリケーションとも呼ばれ，従来とは異なり，デスクトップアプリケーションに迫る高い使用感や機能を提供する．OSやJavaの実行環境などに依存しないことから携帯電話やPDAなどの携帯端末上からの利用も期待されている．

また，SSH（Secure Shell，詳しくは2.3節参照）はきわめてセキュアな通信でリモートコンピュータにアクセスできることから，サーバ管理者やネットワーク管理者に日常的に利用されており，業務を遂行するうえで必須のツールとなっている．

しかしながら，現状のSSHは，どこからでも自由に利用できるわけではない．インターネットに接続された端末を持っていても，その端末にSSHクライアントのソフトウェアがインストールされている必要があるためである．これでは，緊急にSSHを利用したい場面に遭遇しても，インターネットに接続され，SSHクライアントがインストールされている，もしくは，対応したSSHクライアントが提供され，かつ，インストール可能な状況の端末を持っていなくてはならない．

そこで，本研究では，Ajaxを用いたSSHクライアントシステムの提案と実装を行う．Ajaxを用いることにより，インターネットに接続され，ウェブブラウザを搭載した様々な機器からSSHが利用できるようになる．これを広くサービスとして提供すれば，先に述べた緊急時などに大きな力となることが期待できる．

なお，上記の機能は，Javaによる実装であるMindTermをはじめ，数多く実用化されている[19),20)]が，現実に多く存在するJava実行環境がインストールされていないPCやJava実行環境が存在しないタイプの携帯端末で利用することはできない．

以下，2章では要素技術，3章では従来システム，4章で提案システムについて述べ，5章で実装，6章で安全性と性能についての評価を行う．

†1 東京電機大学
Tokyo Denki University

2. 要素技術

2.1 Ajax について

Ajax（Asynchronous JavaScript + XML）とは，既存のウェブ開発技術を組み合わせたウェブアプリケーションの実装手法のことである．具体的には，JavaScript の XMLHttpRequest オブジェクトによる非同期の HTTP(S) 通信とダイナミック HTML によるウェブページの動的な書き換えを行う．これは，ウェブブラウザに読み込まれたウェブページが通信をし，自身の一部を書き換えていくようなイメージである（**図 1**）．XMLHttpRequest オブジェクトは，主要な PC 用のウェブブラウザでサポートされ，携帯電話などの携帯端末でも続々とサポートされている．また，W3C（World Wide Web Consortium）のワーキングドラフト[2] となっており，標準化の作業が進んでいる．

Ajax を用いたウェブアプリケーションは，通信をバックグラウンドで行い，JavaScript の様々なイベントや機能を用いることにより，デスクトップアプリケーションに迫る高い使用感や機能を提供する．また，ウェブの標準的な技術である HTML，JavaScript，スタイルシートを用いるので，対応ウェブブラウザがあれば，OS や Java などの環境に依存せずに利用できる．

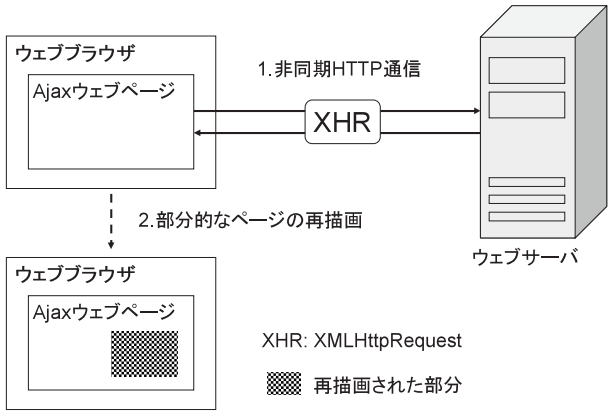

図 1　Ajax ウェブアプリケーションのイメージ
Fig. 1　Image of Ajax web applications.

2.2 Ajax の制限

Ajax の核である XMLHttpRequest は，セキュリティの制限である同一生成元ポリシ[3]（Same origin policy）が適用される．XMLHttpRequest をウェブページに script タグで直接記述している場合は，そのウェブページの生成元サーバ，外部 js ファイルに記述している場合は，その js ファイルを script タグで呼び出しているウェブページの生成元サーバ（プロトコル・ポートも同一）としか通信ができない．

この制限は，生成元のサーバがプロキシとなって中継を行う（プロトコルが異なる場合は相互変換も行う）ことにより，回避することができる．

2.3 SSH について

SSH（Secure Shell）とは，ネットワークを介してリモートコンピュータへのログイン，コマンドの実行を行う通信プロトコル，プログラムである．

SSH では，認証や暗号技術を用いることにより，従来利用されていた TELNET や rlogin などに比べセキュアな仕組みとなっている．以下に SSH の主要なセキュリティ機能をあげる．

（ア） サーバホスト認証により，クライアントが接続先のサーバが正しいかを判断するとともに中間介入攻撃（man-in-the-middle attack）を防ぐ．

（イ） ユーザ認証により，サーバが接続要求をしてきたユーザを受け入れるかを判断する．パスワードや公開鍵認証など多くの認証方式がある．

（ウ） 通信路の暗号化により，クライアント–サーバ間でエンドツーエンドの暗号化を行い，通信の盗聴を防ぐ．TripleDES，RC4，その他の共通鍵暗号をサポートする．

（エ） 完全性の保証により，改ざんや不正なデータの挿入を防止する．

SSH については，文献 4) に詳しく書かれている．

3. 従来システム

3.1 従来システム概要

すでに Ajax を用いて SSH クライアントを利用するシステムが存在する[5),6)]．これらは，**図 2**，**図 3** に示すような構成である．従来システムでは，ウェブブラウザに読み込まれた入出力機能を持つ Ajax ウェブページから XMLHttpRequest を用いた HTTP(S) を通じて，ウェブサーバ内にある SSH クライアントを操作することによって SSH 通信を実現している．そのため，SSH 通信はウェブサーバとリモートコンピュータの間のみである．

3.2 従来システムの問題

従来システムでは，通信路上で中間介入攻撃などによる盗聴・改ざん・なりすましが行わ

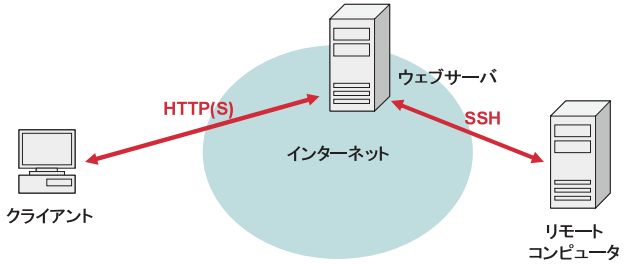

図 2　従来システムのネットワーク構成
Fig. 2　Network configuration of existing systems.

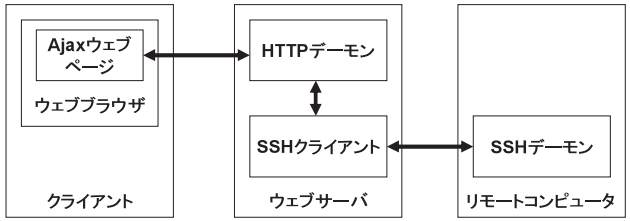

図 3　従来システムのソフトウェア構成
Fig. 3　Software configuration of existing systems.

れる可能性がある．この攻撃により，本来セキュアである SSH の機能が損なわれるという問題がある．

ウェブブラウザとウェブサーバの間を HTTP で通信する場合，通信データは暗号化されないため，ネットワーク上の攻撃者によって攻撃を受ける可能性がある．

ネットワーク上での攻撃を防ぐためにウェブブラウザとウェブサーバの間を HTTPS で通信する場合，ネットワーク上での攻撃は防ぐことができる．しかし，この場合でも，ウェブサーバ内の処理がユーザから見ればブラックボックスとなっていることが問題となる．ウェブブラウザから入力されたコマンド，あるいは，リモートコンピュータからの出力に対して攻撃が行われている可能性を否定できない．また，ユーザがそれを検証することもできない．

従来システムは，ウェブサーバが完全に信頼できる場合，たとえば，自宅や所属組織のサーバに HTTPS を用いて運用するには適している．しかし，不特定多数に向けたサービスを考えた場合には，先に述べた理由への不安から，ユーザは利用しにくいと考えられる．

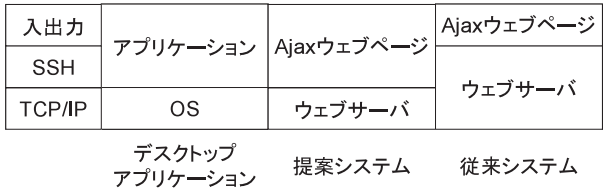

図 4　各実装の機能構成
Fig. 4　Functions configuration of implementations.

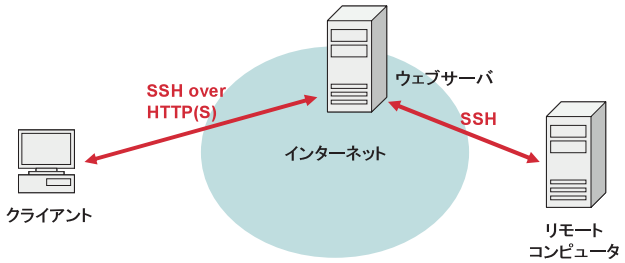

図 5　提案システムのネットワーク構成
Fig. 5　Network configuration of proposed system.

4. 提案システム

4.1 コンセプト

提案システムは，ウェブアプリケーションでありながら，デスクトップアプリケーションの SSH クライアントを利用する場合と同様の安全性を確保するため，エンドツーエンドの SSH 通信を目指す．従来システムの Ajax ウェブページが入出力インタフェースを提供していたのに対し，提案システムでは，Ajax ウェブページが SSH クライアントの機能をすべて持つ．

図 4 はデスクトップアプリケーション，提案システム，従来システムの機能構成を示している．

4.2 システム構成

図 5 に提案システムのネットワーク構成を示す．ネットワーク構成は，従来システムと同じである．

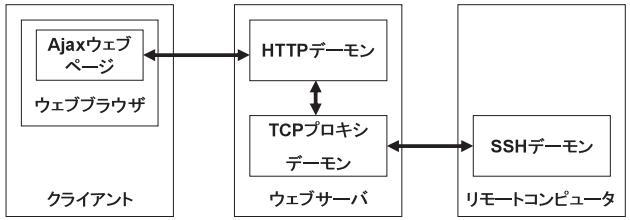

図 6　提案システムのソフトウェア構成
Fig. 6　Software configuration of proposed system.

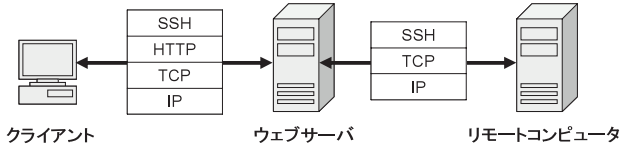

図 7　提案システムのプロトコルスタック
Fig. 7　Protocols stack of proposed system.

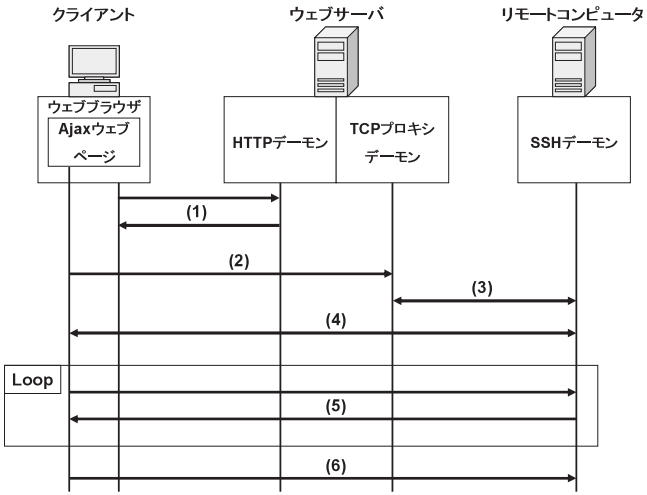

図 8　提案システムの動作シーケンス
Fig. 8　Communication sequence of proposed system.

図 6 にソフトウェア構成を示す．TCP プロキシデーモンは，XMLHttpRequest を用いたリクエストによって TCP 通信を実現するプログラムである．

図 7 にネットワークのプロトコルスタックを示す．

4.3　動 作 概 要

図 8 に提案システムの通信シーケンスを示す．

提案システムの動作は (1)〜(6) の手順で行われる．

(1) ウェブページ取得

ユーザは，まず，ウェブブラウザを用いてウェブサーバの HTTP デーモンにアクセスし，Ajax ウェブページを取得する．これにより，図 6 に示したソフトウェア構成となる．

(2) TCP 接続リクエスト

ユーザが，Ajax ウェブページにリモートコンピュータ名・ポート番号・ユーザ名などの情報を入力してログインすると，Ajax ウェブページは，リモートコンピュータ名・ポート番号を引数にした TCP 接続リクエストを XMLHttpRequest で HTTP デーモンを介して TCP プロキシデーモンに送信する．

(3) TCP 接続の確立

TCP プロキシデーモンは，(2) のリクエストを受け，SSH デーモンと TCP 接続を確立する．これにより，Ajax ウェブページから SSH デーモンへの仮想的な TCP 接続を確立する．

(4) SSH 接続の確立

Ajax ウェブページは，ウェブサーバを介して (3) で確立した仮想的な TCP 接続上で SSH デーモンと SSH 接続を確立する（SSH over HTTP）．

(5) SSH 受信処理

Ajax ウェブページは，ウェブサーバを介して，SSH デーモンから SSH パケットを受信する処理を繰り返す．これには，ウェブブラウザからのリクエストをサーバで一定時間保留し，受信データが発生したときにレスポンスを返す Comet [7] を用いる．SSH パケットを受信した場合には，JavaScript で復号・チェックバイトの演算をし，画面への出力を行う．

(6) SSH 送信処理

ユーザの Ajax ウェブページへのコマンド入力を JavaScript のイベントで取得し，JavaScript で暗号化・チェックバイトの演算を行い，SSH のパケットを生成し，ウェブサーバを介して SSH デーモンへ送信する．

5. 実　　装

プロトタイプとして，提案システムの実装（図 6 の Ajax ウェブページと TCP プロキシデーモン）を行った．

表 1 のとおりの機能を実装した．SSH プロトコルには，SSH1[8] と SSH2[9)–16)] がある．SSH1 はすでに多くの脆弱性が指摘されており，SSH2 の利用が推奨されているが，実装では，開発の容易さから SSH1 を用いることとした．

5.1 動 作 確 認

図 9，**図 10** に実装したプロトタイプのスクリーンショットを示す．**表 2** は動作確認で用いた各マシンの IP アドレス構成である．

図 11 は 4.3 節の動作概要の (2) 以降の通信をウェブサーバ上でキャプチャしたものである．クライアントとウェブサーバの通信は HTTP であり，ウェブサーバとリモートコンピュータの通信は SSH（SSH1）であることが分かる．**図 12**，**図 13** は図 11 の No.176, 177 のパケットの詳細である．

図 12 の HTTP パケットの「Session=」から始まる行の「Data=」の後の「AAAAEd5PMOp（以下省略）」というデータは Base64 にエンコードされており，図 13 の SSH の暗号化パケット（すなわち，TCP セグメントのペイロード）の「00 00 00 11 de 4f（以下省略）」に対応している．これにより，クライアントからリモートコンピュータまでエンドツーエンドの SSH 通信が行われていることが分かる．

実装システムは下記の主要なウェブブラウザで動作を確認している（いずれも Windows XP Home Edition SP2 で確認）．

- Internet Explorer 6.0
- Mozilla Firefox 2.0
- Opera 9.26
- Safari 3.1

表 1　実装機能
Table 1　Functions of implement system.

SSH プロトコル	SSH1
ユーザ認証	パスワード認証
共通鍵暗号	Triple DES
通信	キー押下ごとに通信

図 9　プロトタイプのログイン画面
Fig. 9　Screenshot of login dialog.

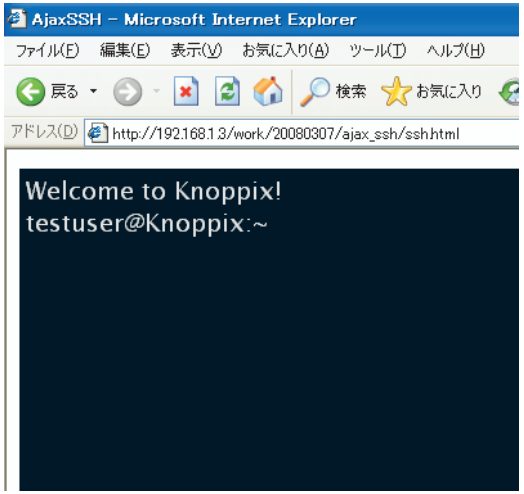

図 10　プロトタイプの端末画面
Fig. 10　Screenshot of terminal window.

表 2　各マシンの IP アドレス
Table 2　IP address cofigurations.

クライアント	192.168.1.5
ウェブサーバ	192.168.1.3
リモートコンピュータ	192.168.1.2

No.	Source	Destination	Protocol	Info
149	192.168.1.5	192.168.1.3	HTTP	POST /work/20080307/ajax_ssh/interface.php
150	192.168.1.3	192.168.1.5	TCP	80 > 1065 [ACK] Seq=1 Ack=539 Win=64997 Le
151	192.168.1.3	192.168.1.2	TCP	1220 > 22 [SYN] Seq=0 Win=65535 Len=0 MSS=
152	192.168.1.2	192.168.1.3	TCP	22 > 1220 [SYN, ACK] Seq=0 Ack=1 Win=5840
153	192.168.1.3	192.168.1.2	TCP	1220 > 22 [ACK] Seq=1 Ack=1 Win=65535 Len=
154	192.168.1.3	192.168.1.5	HTTP	HTTP/1.1 200 OK
155	192.168.1.5	192.168.1.3	HTTP	POST /work/20080307/ajax_ssh/interface.php
158	192.168.1.2	192.168.1.3	SSHv1	Server Protocol: SSH-1.99-OpenSSH_4.3p2 Del
159	192.168.1.3	192.168.1.5	HTTP	HTTP/1.1 200 OK (text/html)
160	192.168.1.5	192.168.1.3	HTTP	POST /work/20080307/ajax_ssh/interface.php
161	192.168.1.3	192.168.1.2	SSHv1	Client Protocol: SSH-1.5-AJAXSSH0.2
162	192.168.1.2	192.168.1.3	TCP	22 > 1220 [ACK] Seq=33 Ack=20 Win=5840 Len=
163	192.168.1.3	192.168.1.5	HTTP	HTTP/1.1 200 OK
164	192.168.1.5	192.168.1.3	HTTP	POST /work/20080307/ajax_ssh/interface.php
165	192.168.1.2	192.168.1.3	SSHv1	Server: Public Key
166	192.168.1.3	192.168.1.5	HTTP	HTTP/1.1 200 OK (text/html)
167	192.168.1.3	192.168.1.2	TCP	1220 > 22 [ACK] Seq=20 Ack=437 Win=65099 Le
168	192.168.1.5	192.168.1.3	TCP	1065 > 80 [ACK] Seq=2127 Ack=2297 Win=6553
169	192.168.1.5	192.168.1.3	HTTP	POST /work/20080307/ajax_ssh/interface.php
170	192.168.1.3	192.168.1.2	SSHv1	Client: Session Key
171	192.168.1.3	192.168.1.5	HTTP	HTTP/1.1 200 OK
172	192.168.1.5	192.168.1.3	HTTP	POST /work/20080307/ajax_ssh/interface.php
173	192.168.1.2	192.168.1.3	TCP	22 > 1220 [ACK] Seq=437 Ack=304 Win=6432 Le
174	192.168.1.2	192.168.1.3	SSHv1	Server: Encrypted packet len=5
175	192.168.1.3	192.168.1.5	HTTP	HTTP/1.1 200 OK (text/html)
176	192.168.1.5	192.168.1.3	HTTP	POST /work/20080307/ajax_ssh/interface.php
177	192.168.1.3	192.168.1.2	SSHv1	Client: Encrypted packet len=17
178	192.168.1.2	192.168.1.3	TCP	22 > 1220 [ACK] Seq=449 Ack=332 Win=6432 Le
179	192.168.1.3	192.168.1.5	HTTP	HTTP/1.1 200 OK

図 11　ウェブサーバでキャプチャしたパケット
Fig. 11　Packets captured on the web server.

```
⊞ Transmission Control Protocol, Src Port: 1065 (1065), Dst Port: 80 (80), Seq: 3550,
⊞ Hypertext Transfer Protocol
⊟ Line-based text data: application/x-www-form-urlencoded
    Session=1213012265687&Method=SEND&Data=AAAAEd5PMOp!sbG8!UIGyPfJd7ziqv5scFNqlQ==
```

図 12　HTTP パケットの詳細（No.176）
Fig. 12　Detail of the HTTP packet (No.176).

```
⊞ Transmission Control Protocol, Src Port: 1220 (1220), Dst Port: 22 (22), Seq: 304,
⊟ SSH Protocol
0000  00 e0 18 22 cb 10 00 13  a9 2a 67 df 08 00 45 00   ..."..... .*g...E.
0010  00 44 0c 33 40 00 80 06  6b 2b c0 a8 01 03 c0 a8   .D.3@... k+......
0020  01 02 04 c4 00 16 4f 7d  28 ce 6f e7 0f ed 50 18   ......O} (.o...P.
0030  fe 3f c5 d5 00 00 00 00  00 11 de 4f 30 ea 7e b1   .?...... ...O0.~.
0040  b1 bc f9 42 06 c8 f7 c9  77 bc e2 aa fe 6c 70 53   ...B.... w....lpS
0050  6a 95                                             j.
```

図 13　SSH 暗号化パケットの詳細（No.177）
Fig. 13　Detail of the SSH encrypted packet (No.177).

5.2　携帯端末への対応

携帯端末で動作する Ajax ウェブページの実装も別途行った．au の携帯電話 W53S に付属する PC サイトビューア（Opera Mini 3.1）で動作を確認した．通信が HTML のボタン押下ごとに行われるようになっている以外は PC 用と同じである．

6.　提案システムの評価

6.1　安全性の評価

(1) SSH クライアント機能を提供する Ajax ウェブページへの不正に対する安全性

提案システムは，「ウェブブラウザに SSH クライアント機能を提供する Ajax ウェブページが正規の状態で読み込まれる」という前提が有効である必要があり，次の 2 つに分解することができる．

（ア）　ウェブサーバに置かれている当該ページが正規の状態である．

（イ）　当該ページがウェブサーバからウェブブラウザにダウンロードされる通信路上で改ざんされない．

（イ）については，HTTPS を用いることで実現可能である．

（ア）については，完全に防止するのは容易ではないが，当該ページの構成要素（HTML・JavaScript・スタイルシート）はクライアント上でソースコードが閲覧可能であるため，それらの知識を持った人が解析することが可能である（現在の実装では合計 2.5K ステップ程度）．また，ホワイトリスト方式のプログラムによって自動判別することも可能になると考えられる．これらにより，サーバ提供者は心理的に不正をしにくいと考えられる．なお，Trusted Third Party がコンテンツの正当性を保証する，著者らが提案したインターネット・マーク[17] のような仕組みが普及すれば，この防止は容易となる．

(2) 従来システムと同様の中間介入攻撃に対する安全性

提案システムは，4.1 節で述べたとおり，Ajax ウェブページが SSH クライアント機能を

持ち，直接 SSH パケットを扱う．そのため，リモートコンピュータへの SSH ログイン時，その後のコマンドのやりとりにおいて，従来システムのようにクライアントからリモートコンピュータ間で暗号化がエンドツーエンドでないことに起因する攻撃を受けることはない．

(3) 一般的な SSH クライアントとしての安全性

一般に SSH では，中間介入攻撃を防ぐためにクライアントが接続先のサーバの公開鍵の検証を行う．SSH のデスクトップアプリケーションでは，この検証は known_hosts ファイルに登録されている公開鍵のリストと比較することによって行っている．現在，提案システムにこの機能は実装されていないため，この点についてはデスクトップアプリケーションに劣る．しかし，SSH クライアント機能を提供する Ajax ウェブページにパスワードベースの暗号化の機能（文献 18) 参照）を用い，ウェブサーバ上に暗号化した状態で known_hosts ファイルを保存するような仕組みを追加することで，解決可能であると考えている．

なお，既存の SSH が対処できない DDoS などの SSH よりも下位のレイヤへの攻撃については，提案システムにおいても解決することはできない．

6.2 性 能 評 価

実装システムについて，ユーザの観点からログインに要する時間，入出力の処理能力の計測を行った．

クライアントのスペックは**表 3**，**表 4** に示すとおりである．それぞれ 10 回計測した平均を結果として**表 5** に示す．（ア）〜（ウ）にそれぞれの計測方法，各計測においてクライアント上で行われる処理の概要を示す．

（ア）ログインの計測

ユーザが図 9 に示したログイン画面の login ボタンを押下してから図 10 に示した画面が表示され，コマンド入力が可能になるまでの所要時間を計測した．

ログイン処理では，サーバホスト認証の RSA 暗号化演算，セッション ID 生成の MD5 ハッシュ演算などが行われる．

（イ）入力の計測

キー入力イベント発生時に呼ばれる関数（引数はキーコード）を計測関数から for ループで同期的に 100 回呼び出し，その for ループを抜け，エコーバック（入力した文字をサーバが出力として送り返すこと）の表示が終了するまでの時間を計測した．

入力処理では，Triple DES 暗号化演算，CRC32 チェックバイト演算などが行われる．

（ウ）出力の計測

リモートコンピュータ内に用意した，ランダムな文字列を 100 行（1 行は 80 文字）出力するプログラムを呼び出し，クライアントでコマンドを入力してから表示が完了するまでの時間を計測した．

出力処理では，Triple DES 復号演算，CRC32 チェックバイト演算などが行われる．

PC においては，結果から 1 秒あたりに 24.7 文字程度が入力可能であり，1 行の表示には 0.01 秒程度を要すると算出できる．これらはいずれも十分実用に値する結果であると考えられる．

携帯電話においては，どの処理についても常用するには厳しい結果となった．ログインについては，2 回の RSA 暗号化演算（1,024，768 ビット）に 20 秒程度要することが大きな

表 3　性能評価に用いたクライアント PC のスペック
Table 3　Specification of client PC.

OS	Windows XP Home Edition SP2
CPU	AMD Mobile Sempron 3100+ 1.80GHz
メモリ	512MB
ウェブブラウザ	Internet Explorer 6.0

表 4　性能評価に用いたクライアント携帯電話のスペック
Table 4　Specification of client mobile phone.

OS	REX OS ＋ KCP
CPU	ARM9E
メモリ	—
ウェブブラウザ	Opera mini 3.1 (8.60)

表 5　計測の結果
Table 5　Result of measurement.

	処理時間（秒）	
	PC	携帯電話
ログイン	0.82	46.38
入力（100 文字）	4.05	286.04
出力（100 行）	0.93	73.73

要因である．また，PC ではほとんどなかった通信におけるオーバヘッドが大きいことも携帯電話における性能の低下を招いている．しかしながら，緊急を要する場合においては性能が低くても多くの端末から利用可能であることを考えれば，利用価値はあると考える．

なお，ここでウェブブラウザに Ajax ウェブページをロードする時間の測定は行わなかったが，Ajax ウェブページのコードセットのサイズは 100 KB 程度と非常に小さく，64 Kbps の回線でも計算上は 12 秒程度でダウンロード可能であり，かつ，動作中に再度これらのファイルをダウンロードすることはないためである．

7. おわりに

本研究では，Ajax を用いて動作環境に依存しない SSH クライアントシステムの提案を行った．提案システムは従来システムとは異なり，デスクトップアプリケーションと同様にエンドツーエンドの SSH 通信を行うことにより，SSH 本来の安全性に大きく近づけることができた．

また，提案システムの実装を行い，PC においては常用できる速度で動作することが分かった．携帯電話においては現状として常用は厳しいが，緊急時に他に代替案がない場合には何とか使える速度であると考えることができる．

今後は，SSH プロトコル 2 や多バイト文字への対応などを行い，実務で利用可能なシステムの構築を目指すとともに，より提案システムの特徴を生かせるような利用法の検討も行っていきたい．

参考文献

1) Google マップ．http://maps.google.co.jp/maps
2) W3C Working Draft, The XMLHttpRequest Object. http://www.w3.org/TR/XMLHttpRequest/
3) Mozilla Japan: 同一生成元ポリシー. http://www.mozilla-japan.org/projects/security/components/same-origin.html
4) Barrett, D.J., Silverman, R.E. and Byrne, R.G.: 実用 SSH：セキュアシェル徹底活用ガイド，オライリー・ジャパン (2006).
5) ベイエリア情報局：AJAX SSH 作ってみた. http://blog.bz2.jp/archives/2005/09/ajax_ssh.html
6) Ajaxterm. http://antony.lesuisse.org/qweb/trac/wiki/AjaxTerm
7) ITpro：第 2 回 Comet—プッシュ型の Web アプリケーションを作る. http://itpro.nikkeibp.co.jp/article/COLUMN/20080220/294242/
8) Ylonen, T.: The SSH (Secure Shell) Remote Login Protocol. http://www.graco.c.u-tokyo.ac.jp/~nishi/security/ssh/RFC
9) The Secure Shell (SSH) Protocol Assigned Numbers. http://www.ietf.org/rfc/rfc4250.txt
10) The Secure Shell (SSH) Protocol Architecture. http://www.ietf.org/rfc/rfc4251.txt
11) The Secure Shell (SSH) Authentication Protocol. http://www.ietf.org/rfc/rfc4252.txt
12) The Secure Shell (SSH) Transport Layer Protocol. http://www.ietf.org/rfc/rfc4253.txt
13) The Secure Shell (SSH) Connection Protocol. http://www.ietf.org/rfc/rfc4254.txt
14) Using DNS to Securely Publish Secure Shell (SSH) Key Fingerprints. http://www.ietf.org/rfc/rfc4255.txt
15) Generic Message Exchange Authentication for the Secure Shell Protocol (SSH). http://www.ietf.org/rfc/rfc4256.txt
16) The Secure Shell (SSH) Transport Layer Encryption Modes. http://www.ietf.org/rfc/rfc4344.txt
17) 洲崎誠一，吉浦　裕，永井康彦，豊島　久，佐々木良一，手塚　悟：Web サイトの真正性を確認可能とするインターネット・マークの提案，情報処理学会論文誌，Vol.41, No.8, pp.2198–2207 (2000).
18) Burnett, S. and Paine, S.: RSA セキュリティオフィシャルガイド—暗号化，翔泳社 (2002).
19) MindTerm. http://www.appgate.com/products/80_MindTerm/
20) JTA – Telnet/SSH for the JAVA(tm) platform. http://www.javassh.org/

(平成 20 年 3 月 31 日受付)
(平成 20 年 10 月 7 日採録)

小須田優介（正会員）

1985 年 8 月 21 日生．2008 年 3 月東京電機大学工学部第二部情報通信工学科卒業．同年 4 月 NEC ソフト株式会社入社．

佐々木良一（フェロー）

1971 年 3 月東京大学卒業．同年 4 月日立製作所入社．システム開発研究所にてシステム高信頼化技術，セキュリティ技術，ネットワーク管理システム等の研究開発に従事．2001 年 4 月より東京電機大学工学部教授，2007 年 4 月より未来科学部教授．工学博士（東京大学）．1998 年電気学会著作賞受賞．2002 年情報処理学会論文賞受賞．2007 年総務大臣表彰等．著書に，『IT リスクの考え方』（岩波新書，2008 年）等．情報処理学会フェロー．情報処理学会コンピュータセキュリティ研究会顧問．日本セキュリティ・マネージメント学会会長，情報ネットワーク法学会理事長，日本学術会議連携会員，日本ネットワークセキュリティ協会会長．