社団法人 情報処理学会 研究報告
IPSJ SIG Technical Report

2008－DPS－134 (51)
2008－CSEC－ 40 (51)
2008／3／7

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† 佐々木 良一†

†東京電機大学

〒101-8457 東京都千代田区神田錦町 2-2

kosuda@isl.im.dendai.ac.jp, sasaki@im.dendai.ac.jp

あらまし 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 などの環境に依存しない 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．動作実験により，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．

Proposal and implementation of SSH client system using Ajax

Yusuke KOSUDA† Ryoichi SASAKI†

† Tokyo Denki University

2-2, Kanda-Nishiki-cho, Chiyoda-ku, Tokyo, 101-8457 JAPAN

kosuda@isl.im.dendai.ac.jp, sasaki@im.dendai.ac.jp

Abstract 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 from OS. 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.

1. はじめに

近年，Ajax 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，従来のウェブアプリケーションとは異なり，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．その高機能性から，今後ますます数多くのデスクトップアプリケーションがウェブアプリケーションになることが期待されている．また，OS 等の環境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 は極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とって業

務を遂行する上で必須のツールとなっている．

本研究では，この Ajax を用いて OS 等の環境に依存しないエンドツーエンドの SSH 通信を実現する SSH クライアントシステムの提案と実装を行う．

提案システムは，単にデスクトップアプリケーションの代替としての利用も可能であるが，特にサーバ管理者が出先で緊急に対応を行わなければならない場合の携帯端末からの利用や SSH 通信が禁止されているネットワークから外部のサーバを操作したい場合の利用に力を発揮すると考えられる．

2. 要素技術

2.1 Ajax について

Ajax(Asynchronous JavaScript + XML)とは，JavaScript の XMLHttpRequest オブジェクトによる非同期 HTTP 通信(図 1)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．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るようなイメージである．

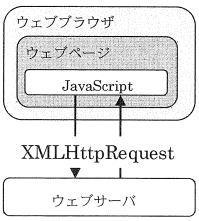

図 1　XMLHttpRequest 通信

Ajax では，バックグラウンドで通信を行うため従来のウェブアプリケーションの様にページ遷移を待たずに作業が継続出来る．また，様々な JavaScript のイベントに応じて処理を行うことにより，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．

Ajax を用いた既存のアプリケーションとして電子メールクライアントやオフィスソフト，オンラインゲーム，デスクトップ環境などがあげられる．また，Ajax ではウェブの標準技術である HTML, JavaScript, css のみを用いるので，標準的なウェブブラウザが利用できれば，OS 等の環境に依存しないことが特徴である．

2.2 Ajax の制限

Ajax の核となる XMLHttpRequest は，セキュリティの制限である同一生成元ポリシー(Same origin policy)によって自身の生成元のサーバ(プロトコル・ポートも同一)としか通信ができない[2]．そのため，他のサーバと通信を行う場合は生成元のサーバが通信を中継する必要がある．また，HTTP(S)以外の通信をする場合も同様に生成元のサーバによる中継の必要がある．

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 等に比べセキュアな仕組みとなっている．

- サーバホスト認証
  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．
- ユーザ認証
  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．
- 通信路の暗号化(図 2)
  クライアントー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，ARCFOUR，その他の共通鍵暗号をサポートする．
- 完全性の保証
  ハッシュ関数を用いて改ざんを検知し，不正なデータの挿入を防止する．

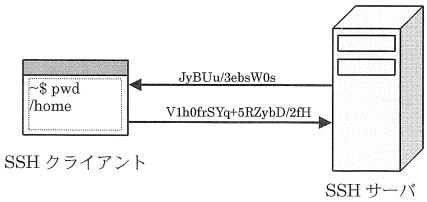

図 2　SSH の暗号化通信

SSH についてさらに詳しく知るには，参考文献[3]を参照してほしい．

3. 従来システム

3.1 システム構成

既に Ajax で SSH クライアントを実現するシステムが存在する[4]．それらのシステムは図 3 に示すような構成である．

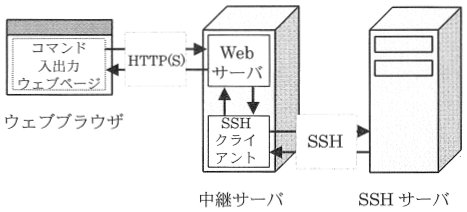

図 3　従来システムの構成

従来システムは，Ajax を用いてウェブブラウザからコマンドの入出力を行い，中継サーバ内の SSH クライアントを利用する仕組みとなっている．

3.2 従来システムの利点

従来システムは Ajax を用いることによる以下の利点がある．

- ウェブブラウザが利用できれば，OS 等の環境に依存しない
- 利用者が SSH クライアントをインストールする必要がない
- SSH 通信が禁止されている場合でも利用できる

3.3 従来システムの問題点

従来システムにはセキュリティ上の問題点がある．

まず，ウェブブラウザと中継サーバの間を HTTP で通信する場合，通信データは暗号化されていないため(図 4)，ネットワーク上の攻撃者による盗聴・改ざん・成りすましなどの攻撃を受ける可能性がある．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問題がある．

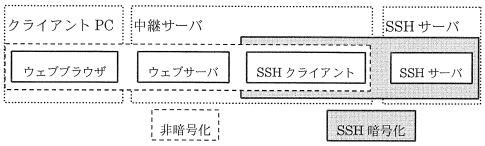

図 4　HTTP 通信を用いた場合の暗号化(従来システム)

そして，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中継サーバの間を HTTPS で通信する場合においても中継サーバ内に非暗号化区間が存在する(図 5)ため，中継サーバ内において攻撃が行われる可能性がある．これにより，HTTP 通信の場合と同様に SSH の機能が損なわれる問題がある．

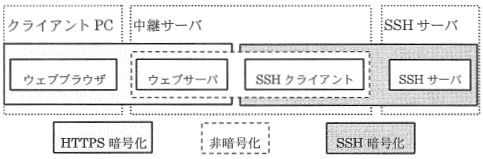

図 5　HTTPS を用いた場合の暗号化(従来システム)

これらの問題は SSH 通信が部分的であることに起因するため，解決にはエンドツーエンドの SSH 通信を行うことが必要である．

4. 提案システム

4.1 システム構成

提案システムは，図 6 に示す構成である．従来システムと同様の機器構成になっているが，その仕組みは異なる．

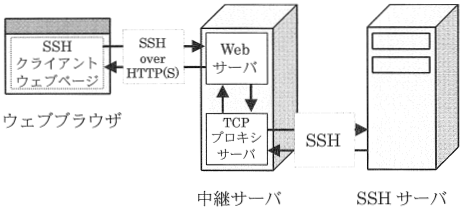

図 6　提案システムの構成

提案システムでは，Ajax を用いることにより，ウェブブラウザが SSH クライアントになる．

ウェブブラウザはユーザからコマンドの入力を受け付けると，JavaScript によって SSH パケットを生成する．そして，生成した SSH パ

ケットを SSH サーバへ送信する．しかし，2.2 節で述べたように，Ajax ではウェブページの生成元サーバと HTTP(S)通信しか行うことが出来ない．そのため，ウェブサーバが動作している中継サーバに SSH サーバへの TCP 通信を代理してもらうことにより擬似的にエンドツーエンドの SSH 通信を行う仕組みとした(図 7)．

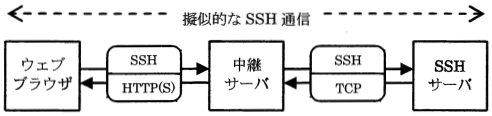

図 7　提案システムの SSH 通信の仕組み

4.2　システムの動作概要

シェルが実行されコマンド入力が可能になると，バックグラウンドで図 8 に示す SSH パケット受信処理が繰り返される．コマンド入力を行うと，図 9 に示す SSH パケット送信処理が行われる．

この 2 つの処理を組み合わせることにより，SSH 通信を行う．

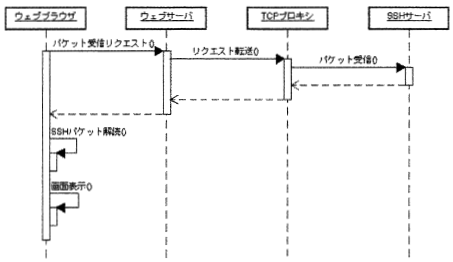

図 8　SSH パケット受信のシーケンス

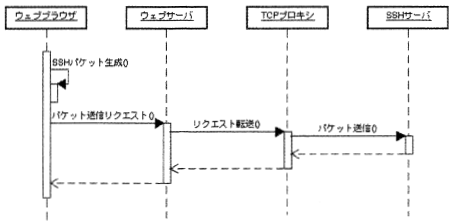

図 9　SSH パケット送信のシーケンス

4.3　提案システムの利点

従来システムは，ウェブブラウザとウェブサーバ間を HTTPS で通信する場合でも非暗号化区間が存在した．しかし，提案システムではエンドツーエンドの SSH 通信を実現するため，非暗号化区間は存在しない(図 10)．これにより，本来の SSH 通信に迫る安全性が実現できる．

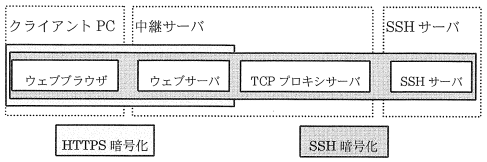

図 10　HTTPS を用いた場合の暗号化(提案システム)

また，提案システムにおいては，SSH クライアントの機能を JavaScript で実装するため，利用者がそのソースコードを閲覧することにより，処理の正当性を検証することが可能である．

4.4　提案システムの安全性

提案システム自体の安全性について検討を行う．提案システムは中継サーバがクライアントの要求によって TCP 通信の代理を行うため，攻撃者が不正な通信の要求を行うことにより踏み台にされる可能性がある．これには 2 つの対策が考えられる．

A) 22 番ポート以外への接続を禁止する
　この対策はほとんどの場合において有効であると考えられる．しかし， SSH サーバが 22 番ポート以外で待機している場合に接続が出来なくなる問題がある．

B) SSH 通信かを判別する
　SSH プロトコルでは，通信開始時に文字列によるプロトコルバージョンの交換を行うため，これが見られない場合に通信を切断する．この方式であれば A)における問題を解決することができる．

5.　提案システムの実装

5.1　実装プログラム

提案システムを実現するために次の 3 つのプログラムを実装する．

A) TCP プロキシサーバ
　TCP 通信を代理で行うプログラム．

B) Ajax インタフェース
　ウェブサーバと TCP プロキシサーバのプロセス間通信を行うプログラム.
C) SSH クライアントウェブページ
　SSH クライアント機能を持つ Ajax ウェブページ. 機能は下表の通りである.

表 1　SSH クライアントウェブページ実装機能

SSH プロトコル	バージョン 1.5[5]
ユーザ認証	パスワード認証
暗号アルゴリズム	TripleDES
キー入力	透過モード

5.2 開発環境

表 2〜4 の環境を用いて実装を行った.

表 2　TCP プロキシサーバ開発環境

OS	Windows XP Home Edition SP2
開発言語	C#

表 3　Ajax インタフェース開発環境

OS	Windows XP Home Edition SP2
ウェブサーバ系	XAMPP Lite 1.6.1 Beta 1
開発言語	PHP

表 4　SSH クライアントウェブページ開発環境

OS	Windows XP Home Edition SP2
ウェブサーバ系	XAMPP Lite 1.6.1 Beta 1
開発言語	HTML, JavaScript, CSS

6. 実装システムの動作実験

6.1 実験環境

図 11 の環境で実装システムの動作実験を行った. 各マシンの仕様は表 4〜6 の通りである.

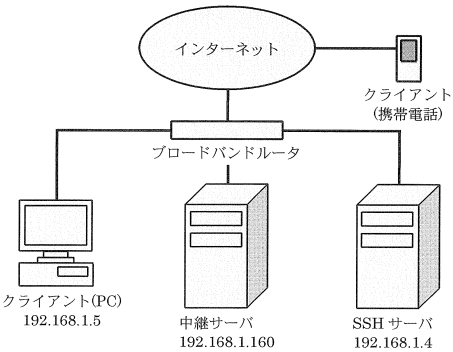

図 11　動作実験のネットワーク構成

表 5　クライアント(PC)仕様

OS	Windows XP Media Center Edition SP2
CPU	Intel Pentium D 820 (2.8GHz)
メモリ	1024MB

表 6　クライアント(携帯電話)仕様

端末	au W53S

表 7　中継サーバ仕様

OS	Windows XP Home Edition SP2
CPU	Intel Celeron M 383 (1GHz)
メモリ	512MB
コンポーネント	.NET Framework 2.0
ウェブサーバ系	XAMPP Lite 1.6.1 Beta 1

表 8　SSH サーバ仕様

OS	KNOPPIX 日本語版 5.1.1CD
CPU	AMD Sempron 3100+ (1.8GHz)
メモリ	512MB
SSH サーバ	OpenSSH_4.3p2 Debian-8

6.2 動作確認

6.2.1. PC における動作確認

下記のウェブブラウザにおいて動作することを確認した.

- Mozilla Firefox 2.0
- Internet Explorer 6.0
- Opera 9.25
- Safari 3.0

図 12, 13 に動作時のクライアントの画面をキャプチャしたものを示す.

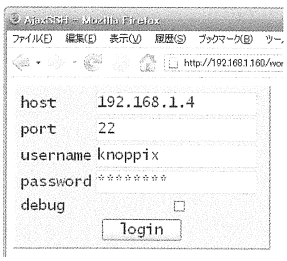

図 12　ログイン画面

図 13　端末画面

6.2.2. 携帯端末での動作確認

携帯端末で動作する SSH クライアントウェブページの実装も別途行い，携帯電話に搭載されている PC サイトビューアー(Opera Mini 3.1)で動作を確認した．しかし，サーバホスト認証時に行われる 2 回の RSA 暗号化にかなりの時間を要するため，計測を行った．結果を表 9 に示す．参考に PC の Mozilla Firefox 2.0 で計測した結果を表 10 に示す．演算時間はいずれも 3 回計測した平均である．

表 9　PC サイトビューアーのサーバホスト認証 RSA 暗号化演算時間

ホスト鍵長 (ビット)	サーバ鍵長 (ビット)	合計演算時間 (ミリ秒)
1024	768	19018.67
2048	768	77090.33

表 10　Mozilla Firefox 2.0 のサーバホスト認証 RSA 暗号化演算時間

ホスト鍵長 (ビット)	サーバ鍵長 (ビット)	合計演算時間 (ミリ秒)
1024	768	359.67
2048	768	1036.33

また，携帯端末では，HTTP 通信の速度が遅く，ログインしてからコマンド入力可能になるまでに，さらに他の処理に要する時間が 20 秒程度加えられ(ホスト鍵長 1024 ビット・サーバ鍵長 768 ビットの場合，計 42 秒程度)，日常的な利用は難しい．しかしながら，緊急を要する場合においては，現状でも利用する価値はあると考えている．

7. まとめと今後の課題

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と実装を行った．従来システムの利点を引き継ぎ，エンドツーエンドの SSH 通信を実現することにより，SSH 本来の安全性に近づけることができた．

今後，システムについてはクライアントの SSH プロトコルバージョン 2 への対応と携帯端末でのパフォーマンスの向上，サーバ側の同時接続数の確保行いたい．また，他のシステムとの比較と評価に努めていくと同時に様々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/, (参照 2008.01.31)

[2] “同一生成元ポリシー”. Mozilla Japan. http://www.mozilla-japan.org/projects/security/components/same-origin.html, (参照 2008.01.20)

[3] Barrett, Daniel J.; Silverman, Richard E.; Byrne, Robert G. 実用 SSH:セキュアシェル徹底活用ガイド. 小島肇監訳. 第 2 版, オライリー・ジャパン, (2006.11)

[4] “AJAX SSH 作ってみた”. ベイエリア情報局. http://blog.bz2.jp/archives/2005/09/ajax_ssh.html, (参照 2008.01.23)

[5] Ylonen, T. “The SSH (Secure Shell) Remote Login Protocol”. http://www.graco.c.u-tokyo.ac.jp/~nishi/security/ssh/RFC, (参照 2008.01.20)