Proposal and Implementation of SSH Client System Using Ajax

Yusuke Kosuda and 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. Technology called Ajax gives web applications the functionality and operability of desktop applications. In this study, we propose and implement a Secure Shell (SSH) client system using Ajax, independent of the OS or Java execution environment. In this system, SSH packets are generated on a web browser by using JavaScript and a web server works as a proxy in communication with an SSH server to realize end-to-end SSH communication. We implemented a prototype program and confirmed by experiment that it runs on several web browsers and mobile phones. This system has enabled secure SSH communication from a PC at an Internet cafe or any mobile phone. By measuring the processing performance, we verified satisfactory performance for emergency use, although the speed was unsatisfactory in some cases with mobile phone. The system proposed in this study will be effective in various fields of E-Business.

Keywords: Ajax, SSH, security, mobile phone, mobile PC, PDA, Smartphone.

1 Introduction

In recent years, many web applications based on a technology called Ajax (Asynchronous JavaScript and XML: see Section 2.1 for details) have been developed. Ajax applications, represented by Google Map [1], have greatly contributed to the realization of "Web 2.0." Unlike conventional web applications, Ajax applications can provide functionality and operability equal to those of desktop applications. Since the Ajax applications do not depend on the OS or Java execution environment, it is anticipated that they can be used from mobile phones, PDA, or other mobile terminals. Because of the capability of extremely safe access to a remote computer, SSH (Secure Shell: see Section 2.2 for details) is used daily by server and network administrators as an essential tool for work execution.

The current SSH, however, is not free to use in any environments. Even when an Internet-connected terminal is available, SSH communication is not possible without SSH client software installed in the terminal. For emergency SSH communication, a user needs an Internet-connected terminal where a corresponding SSH client is installed or provided for installation.

C. Godart et al. (Eds.): I3E 2009, IFIP AICT 305, pp. 184–196, 2009.

In this study, therefore, we propose and implement a SSH client system using Ajax. Ajax makes SSH communication available from various kinds of Internet-connected equipment with an installed browser. If widely provided as a service, the proposed system will be very helpful in emergency cases and effective in various fields of E-Business.

The function described previously has already been implemented in programs such as MindTerm using Java [19], [20]. These implementation methods, however, cannot be applied to many existing PCs or mobile terminals (mobile phone, PDA, etc.) without Java execution environment. With the results of this research, we can ensure the safe implementation of the above functions even in environments where it previously was not possible.

2 Elemental Technologies

2.1 Outline of Ajax

Ajax is a technique for web application implementation combined with the existing web development technologies. More specifically, asynchronous HTTP and/or HTTPS (After this, we represent HTTP and/or HTTPS as HTTP(S)) communication is set up by the XMLHttpRequest object of JavaScript and a web page is dynamically rewritten by dynamic HTML, as if a web page read into a web browser rewrites part of itself through communication (Fig. 1). If Ajax is applied to a web search, for example, search results can be displayed on a real-time basis because the search is executed not after the confirmation of input as before, but in the background while the user is inputting the key. By executing communication in the background and using various events and functions of JavaScript, web applications can provide high responsiveness and various functions almost equally as desktop applications.

The XMLHttpRequest object is supported by major web browsers for the PC and its support is successively increased among mobile phones and other mobile terminals. Also, through a working draft at W3C (World Wide Web Consortium) [2], this object is being standardized.

Ajax-applied web applications use such web standard technologies as HTML, JavaScript, and Style Sheets. If only a supporting web browser is available, therefore, the applications can be used independently of the OS or Java environment.

The same origin policy [6] for the security limit is applied to XMLHttpRequest which is one of the Ajax cores. When XMLHttpRequest is directly written on a web page with a script tag, communication can be set up only with the generated-from server (the same protocol port) of the web page. When XMLHttpRequest is written in an external .js file, communication can be set up only with the generated-from server (the same protocol port) of the web page calling the .js file by the script tag.

This limit is a function preferable for safety but disrupts the realization of the target function in this research. To avoid the limit, therefore, the generated-from server becomes a proxy for relaying (mutual conversion is also conducted if the protocols are different).

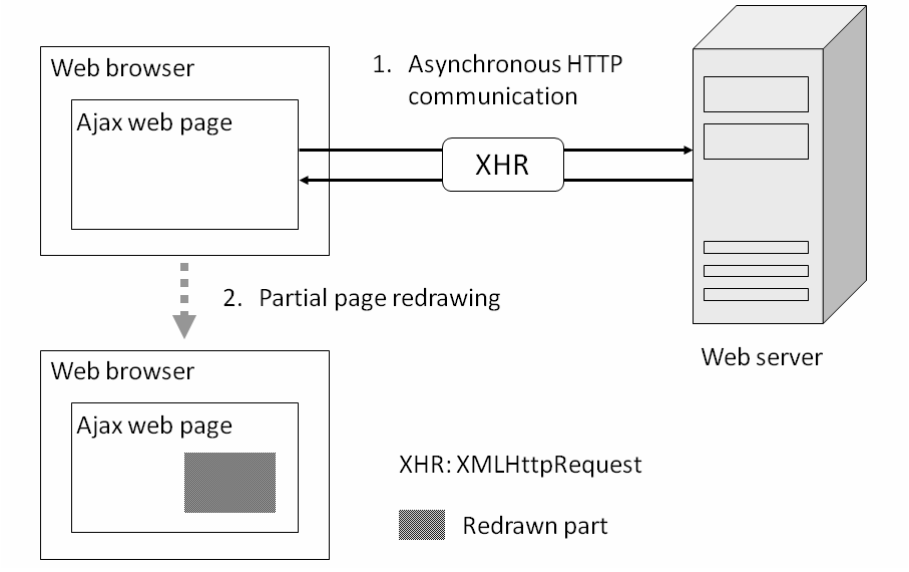

Fig. 1. Image of Ajax web applications

2.2 Outline of SSH

SSH is a communication protocol and program for login to a remote computer and command execution through a network.

Compared with the conventional TELNET and rlogin, SSH has the following safety mechanisms using authentication and encryption technologies:

- Server host authentication for a client to judge whether the connected server is allowed access and also to prevent a man-in-the-middle attack
- User authentication (e.g., password, public key authentication and many other kinds of authentication) for a server to judge whether to accept a connection requested by a user
- Communication channel encryption (e.g., TripleDES, AES, RC4, and other common key encryption) for end-to-end encryption between a client and a server to prevent tapping
- Integrity assurance to prevent tampering and illegal data insertion

For details about SSH, refer to Ref. [4].

3 Existing Systems

3.1 Outline of Existing Systems

There are already Ajax-applied systems using SSH clients [5], [6]. These systems have the configurations shown in Figs. 2 and 3. The existing systems realize SSH

communication by operating an SSH client in a web server through HTTP(S) using XMLHttpRequest from an Ajax web page with an I/O function read into the web browser. Therefore, SSH communication is only partial between a web server and a remote computer.

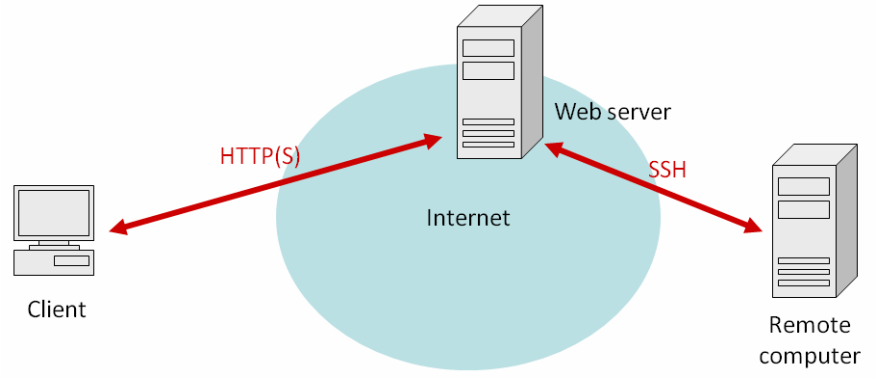

Fig. 2. Network configuration of existing systems

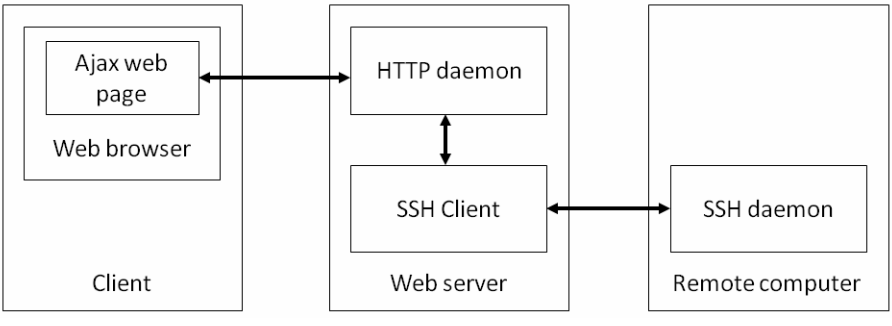

Fig. 3. Software configuration of existing systems

3.2 Problems of Existing Systems

The existing systems are subject to tapping, tampering, or spoofing by a man-in-the-middle attack on a communication channel. This attack may impair the substantially secure function of SSH.

HTTP communication between a web browser and a web server may be attacked by a network attacker because communication data is not encrypted.

Encryption using HTTPS between a web browser and a web server protects communication from an attack on a network. The problem with this method, however, is that processing in the web server is a black box for the user. We cannot deny the possibility of an attack on command input from a web browser or output from a remote computer. In addition, the user cannot verify this possibility.

With a completely reliable web server, the existing systems are suitable for a service using HTTPS in a server at home or at an organization where the user belongs. However, such systems may not be suitable for a service provided to the public.

4 Proposed System

4.1 Concept

The goal of the proposed system is end-to-end SSH communication that ensures the security level of a desktop application using an SSH client for a web application. An Ajax web page of the existing systems is used currently to provide an I/O interface. To the contrary, the Ajax web page has all the functions of an SSH client in the proposed system. Figure 4 shows the function configurations of desktop applications, existing systems, and the proposed system.

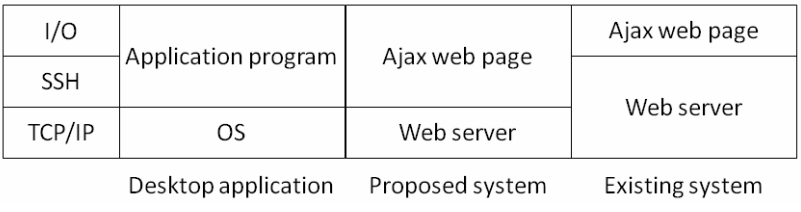

Fig. 4. Functions configuration of implementations

4.2 System Configurations

Figure 5 shows the network configuration of the proposed system. The network configuration is the same as those of the existing systems.

Figure 6 shows the software configuration of the proposed system. The TCP proxy daemon is a program to realize TCP communication by a request using XMLHttpRequest.

Figure 7 shows the network protocols stack of the proposed system.

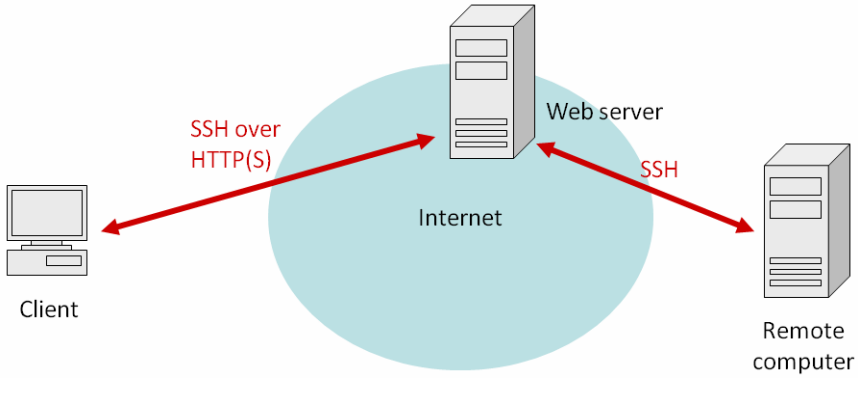

Fig. 5. Network configuration of proposed system

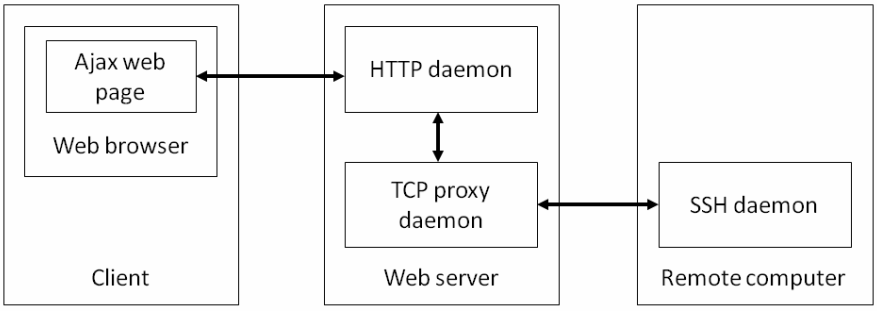

Fig. 6. Software configuration of proposed system

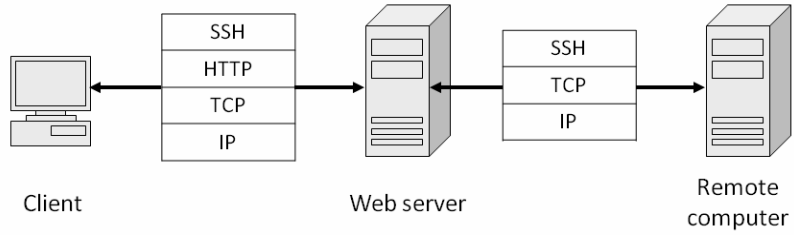

Fig. 7. Protocols stack of proposed system

4.3 Operation Outline

Figure 8 shows the communication sequence of the proposed system.

The proposed system operates in order from (1) to (6), listed as follows.

(1) Web page acquisition
By using the web browser, the user accesses the HTTP daemon of the web server and acquires the Ajax web page. Thus, the software configuration becomes that shown in Fig. 6.

(2) TCP connection request
The user enters the remote computer name, port number, user name, and other information into the Ajax web page for login. With the remote computer name and port number as arguments, the Ajax web page sends a TCP connection request to the TCP proxy daemon through the HTTP daemon by using XMLHttpRequest.

(3) Establishment of TCP connection
By receiving the request of (2), the TCP proxy daemon establishes a TCP connection with the SSH daemon. This establishes a virtual TCP connection from the Ajax web page to the SSH daemon.

(4) Establishment of SSH connection
Through the web server, the Ajax web page establishes an SSH connection with the SSH daemon on the virtual TCP connection established at (3) (SSH over HTTP).

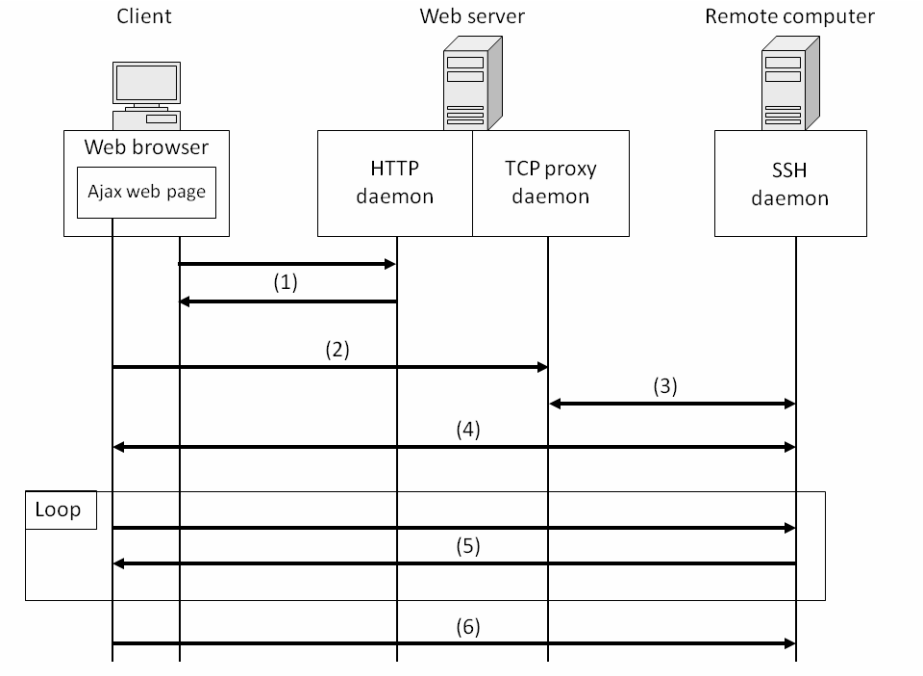

Fig. 8. Communication sequence of proposed system

(5) SSH receive processing
The Ajax web page receives SSH packets repeatedly from the SSH daemon through the web server. For this processing, Comet [7] is used to keep a request from the web browser at the web server for a specified time and return a response when the receive data is generated. When SSH packets are received, they are decrypted and their check bytes are calculated by JavaScript for screen refreshing and other processing.

(6) SSH send processing
Commands entered to the Ajax web page by the user are acquired by JavaScript events. The commands are encrypted and their check bytes are calculated by JavaScript. Then SSH packets are generated and sent to the SSH daemon through the web server.

5 Implementation

A prototype system was implemented (Ajax web page and TCP proxy daemon in Fig. 6).

The functions listed in Table 1 were mounted. The SSH protocols are SSH1 [8] and SSH2 [9], [10], [11], [12], [13], [14], [15], [16]. Many vulnerabilities of SSH1 have already been noted and so the use of SSH2 is recommended. Because of easy development, however, we decided to use SSH1 for implementation.

Table 1. Functions of implemented system

SSH protocol	SSH1
User authentication	Password authentication
Common key encryption	Triple DES
Communication	Each time a key is pressed

5.1 Operation Check

Figures 9 and 10 show screen shots of the implemented prototype. Table 2 gives the IP address configurations of the machines used for the performance check.

Figure 11 shows the communication of (2) and later processes, described in Section 4.3 "Operation Outline," captured on the web server. The communication type is HTTP between the client and the web server and SSH (SSH1) between the web server and the remote computer. Figures 12 and 13 show the packets of No. 176 and 177 in Fig. 11 in detail.

Fig. 9. Screenshot of login dialog

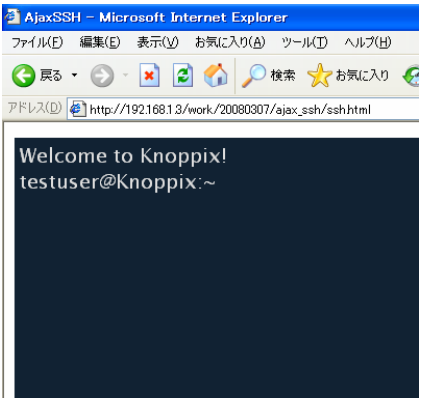

Fig. 10. Screenshot of terminal window

Table 2. IP address configurations

Client	192.168.1.5
Web server	192.168.1.3
Remote computer	192.168.1.2

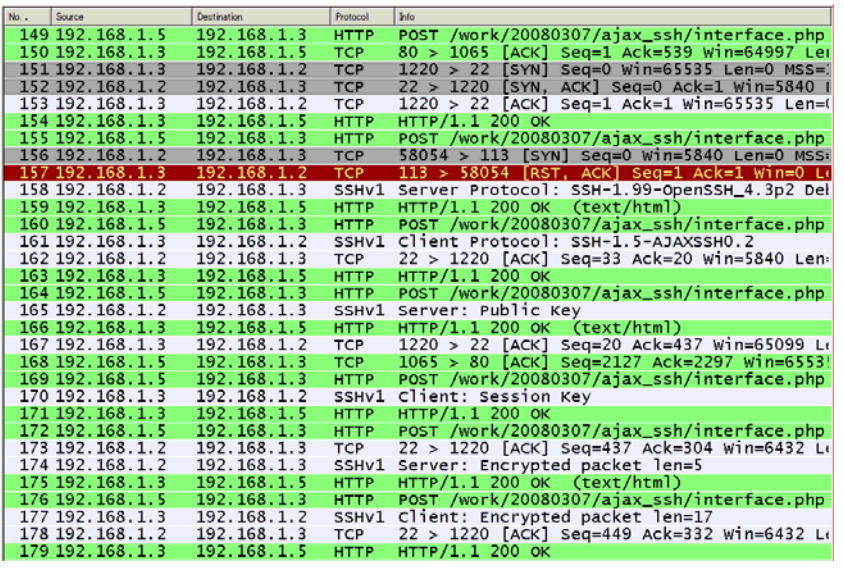

Fig. 11. Packets captured on the web server

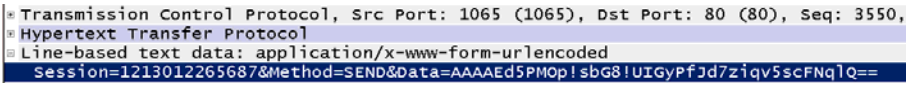

Fig. 12. Detail of the HTTP packet (No.176)

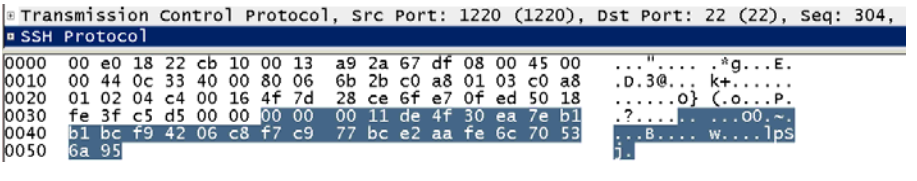

Fig. 13. Detail of the SSH encrypted packet (No.177)

In the line starting from "Session=" of the HTTP packet shown in Fig. 12, the data of "AAAAEd5PMOp" (omitted hereinafter) after "Data=" is encoded into Base64. This corresponds to "00 00 00 11 de 4f " (omitted hereinafter) of the SSH encrypted packet (i.e., TCP payload) shown in Fig. 13. This indicates end-to-end SSH communication from the client to the remote computer.

The implemented system was verified to operate with the following main web browsers (all running on Windows XP Home Edition SP2):

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

5.2 Support of Cell Phone

In addition, we separately implemented an Ajax web page for mobile terminals and verified its operation by using the PC site viewer (Opera Mini 3.1) provided with the mobile phone KDDI AU W53S. The web page for the PC is the same as that for PC except for communication at every line feed.

6 Evaluations of Proposed System

6.1 Safety Evaluation

6.1.1 Safety against Illegal Access to Ajax Web Page with SSH Client Function

The proposed system is based on the web server's assumption that the Ajax web page is read normally into a web browser to provide the SSH client function. This assumption consists of two conditions:

(A) The page in the web server is normal.
(B) The page is not tampered on a communication channel when being downloaded from the web server to the web browser.

The condition of (B) can be realized by using HTTPS.

The condition of (A) is not easy to satisfy completely. The components (HTML, JavaScript, and Style Sheets) of the page allow the browsing of their source codes on a client. Therefore, a knowledgeable person can analyze them (about 2.5 K steps under the current implementation). A whitelist-type program may also enable automatic judgment. Under these circumstances, a server provider is considered to have psychological resistance against illegal behaviors. This prevention will be easy if Internet Mark [17], proposed separately by the authors, becomes popular. Internet Mark is a mechanism in which a trusted third party guarantees the validity of contents.

6.1.2 Safety against Man-in-the-Middle Attack in the Existing Systems

In the proposed system, an Ajax web page has the SSH client function and directly handles SSH packets, as mentioned in Section 4.1, to allow end-to-end encryption between a client and a remote computer. With the SSH login to a remote computer and later command exchange, this encryption prevents a man-in-the-middle attack that used to be a problem in the existing systems.

6.1.3 Safety of General SSH Client

In SSH, a client generally verifies the public key of a connect-to server to prevent a man-in-the-middle attack. For this verification, the SSH desktop application compares the public key with those on the public key list registered in the known_hosts file. This function is not implemented in the proposed system now but will be realized if a password-based encryption function (see Ref. [18]) is added to an Ajax web page that provides the SSH client function and if the encrypted known hosts file is stored in the web server.

Judging from the above, the proposed system is not inferior to the SSH desktop application in safety against a man-in-the-middle attack.

The proposed system cannot solve a Distributed Denial of Service (DDoS) attack at any layer lower than SSH where the existing SSH cannot secure safety.

6.2 Performance Evaluation

From the user's point of view, the time required for login and the I/O processing performance of the implemented system were measured.

Tables 3 and 4 list the client specifications, and Table 5 lists the average of 10 measurements for each. (A) to (C) describe the measuring methods and outline the client processing at each measuring point.

(A) Login measurement

Time was measured from when the user pressed the login button on the login screen shown in Fig. 8 until the screen shown in Fig. 9 was display for command input.

The login processing includes RSA encryption operation for server host authentication and MD5 hash operation for session ID generation.

(B) Input measurement

Time was measured from when a function (argument: key code) was called 100 times synchronously by a for loop at a key input event until echo-back (the server outputs and sends back input characters) after the exit from the for loop.

The input processing includes Triple DES encryption operation and CRC32 check byte operation.

(C) Output measurement

Time was measured from when a program was called from the remote computer to read random character strings for 100 lines (80 characters/line) and a command was entered from the client until the display was completed.

The output processing includes a Triple DES decryption operation and CRC32 check byte operation.

Table 3. Specification of client PC

OS	Windows XP Home Edition SP2
CPU	AMD Mobile Sempron 3100+ 1.80 GHz
Memory	512 MB
Web browser	Internet Explorer 6.0

Table 4. Specification of client mobile phone

OS	REX OS + KCP
CPU	ARM9E
Memory	—
Web browser	Opera mini 3.1 (8.60)

Table 5. Result of measurement

	Processing Time (second)	
	PC	Mobile Phone
Login	0.82	46.38
Input (100 characters)	4.05	286.04
Output (100 lines)	0.93	73.73

According to calculations from the results, 24.72 characters can be entered from a PC per second and it takes approximately 0.01 second to display one line. These values are considered practical for ordinary usage.

For a mobile phone, however, every processing ended in a severe result for ordinary use. For login, 20 seconds was necessary for the RSA encryption operation executed twice (1024 and 768 bits). The large communication overhead, which can be ignored in a PC, also lowers the performance of a mobile phone. In an emergency, however, the use of SSH may be unavoidable. For example, only one administrator while traveling without a PC may access a server using SSH. In such a case, the proposed system is worth using because it is available for various terminals even though the performance may be low.

We did not measure the time required to load an Ajax web page to a web browser. Since the code set of an Ajax web page is extremely small at approximately 100 KB, the time can be considered negligible.

7 Conclusion

In this study, we proposed a SSH client system using Ajax, independent of the operating environment. Unlike existing systems, the proposed system could greatly increase the substantial safety of SSH by end-to-end SSH communication such as that on a desktop application.

By implementation, we found the proposed system to run at a speed satisfactory for regular use by a PC. The proposed system is not fast enough for regular use by a mobile phone, but can be used in an emergency if there is no alternative.

In the future, we will enhance the proposed system for practical use by supporting SSH Protocol 2 and multi-byte characters, and we will also study its usage to take full advantage of the features.

References

1. Google Map, http://maps.google.com/maps
2. W3C Working Draft, The XMLHttpRequest Object, http://www.w3.org/TR/XMLHttpRequest/
3. Mozilla Japan, http://www.mozilla-japan.org/projects/security/components/ame-origin.html
4. Barrett, D.J., Silverman, R.E., Byrne, R.G.: SSH, the Secure Shell The Definitive Guide, 2nd edn. O'Reilly, Sebastopol (2005)
5. http://blog.bz2.jp/archives/2005/09/ajax_ssh.html
6. Ajaxterm, http://antony.lesuisse.org/qweb/trac/wiki/AjaxTerm
7. 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. Yoshiura, H., Shigematsu, T., Susaki, S., Saitoh, T., Toyoshima, H., Kurita, C., Tezuka, S., Sasaki, R.: Authenticating Web-Based Virtual Shops Using Signature-Embedded Marks – A Practical Analysis. In: The 2000 Cambridge International Workshop on Security Protocols (2000) (in Cambridge)
18. Burnett, S., Paine, S.: RSA Security's Official Guide to Cryptography. The McGraw- Hills Company, New York (2001)
19. MindTerm, http://www.appgate.com/products/80_MindTerm/
20. JTA - Telnet/SSH for the JAVA(tm) platform, http://www.javassh.org/