Secure Communication Protocol Between a Human and a Bank Server for Preventing Man-in-the-Browser Attacks

Takashi Tsuchiya[1], Masahiro Fujita[1], Kenta Takahashi[2], Takehisa Kato[3], Fumihiko Magata[4], Yoshimi Teshigawara[5], Ryoichi Sasaki[5], and Masakatsu Nishigaki[1(✉)]

[1] Shizuoka University, Hamamatsu, Japan
nisigaki@inf.shizuoka.ac.jp
[2] Hitachi, Ltd., Totsuka, Japan
[3] Toshiba Corporation Industrial ICT Solutions Company, Fuchu, Japan
[4] NTT Secure Platform Laboratories, Musashino, Japan
[5] Tokyo Denki University, Adachi, Japan

Abstract. Man-in-the-Browser (MITB) attacks are caused by malware that infects a web browser; hence, conventional secure communication channels between a machine (bank server) and a machine (web browser) such as SSL cannot prevent the attacks. In this paper, we propose an approach to preventing MITB attacks by constructing secure communication channels between a machine (bank server) and a human (end user). Our approach uses the user as a computational resource and requests the user to process an end side of the channel. Developing a challenge and response protocol that achieves the proposed channel, we conducted a safety evaluation of the protocol. The result shows that the protocol works safely under the assumption that the bank server can send a "challenge that malware in the browser cannot see" to the user. We also show that sending the challenge is feasible by applying CAPTCHA technology.

Keywords: Man-in-the-Browser attacks · Secure communication channel · CAPTCHA

1 Introduction

Recently, illegal money transfers via internet banking have been on the increase [1]. There are various types of illegal money transfers. In particular, Man-in-the-Browser (MITB) attacks have attracted attention. MITB attacks are caused by malware that infects a web browser. The browser is then capable of falsifying a user's web transactions and stealing the user's password. Many internet banking sites have prevented illegal money transfers by constructing a secure communication channel between a machine (bank server) and a machine (web browser) such as SSL [2, 3]. However, this secure communication cannot prevent MITB attacks. This is because the attacks are caused by malware that infects a web browser on the inside of the secure communication channel.

T. Tryfonas (Ed.): HAS 2016, LNCS 9750, pp. 77–88, 2016.
DOI: 10.1007/978-3-319-39381-0_8

To deal with MITB attacks, we propose an approach to preventing them by constructing a secure communication channel[1] between a machine (bank server) and a human (end user). We give an example of protocols to construct the channel, in particular, a challenge and response protocol using a "challenge that malware in the browser cannot see." It should be noted that, when constructing the channel, we cannot use well-known encryption techniques such as SSL. These cryptographic techniques are basically based on high *machine* computing power and therefore can be applied only to secure communication between a machine and *a machine*. In comparison, our protocol enables secure communication between a machine and *a human*. This is the main contribution in this paper. The organization of this paper is as follows. In Sect. 2, we give details on MITB attacks. We introduce our proposal in Sects. 3 and 4, and discuss it in Sect. 5. In Sect. 6, we describe related work. Finally, we present our conclusions in Sect. 7.

2 Internet Banking and MITB Attacks

2.1 Money Transfer Protocol

In this section, we describe a money transfer protocol used for internet banking. Here, for simplicity, we give a simple description.

Most banks use a money transfer protocol like shown in Fig. 1. The entities of the money transfer protocol are as follows.

- Bank server: The bank server is a server of a financial institution that provides the internet banking service. We assume that the bank server is safe, e.g., it is not possible to leak data and modify the processing in the server. The bank server is a machine; it therefore has a high computing power (and memory capacity) and does not have advanced cognitive abilities.
- User: The user is a customer who uses the internet banking service. When remitting his or her money to any account, he or she operates the PC in accordance with a money transfer protocol provided by the financial institution. We assume that the user perfectly operates the PC in accordance with the protocol. The user is a human and therefore has a low computing power (and memory capacity) and advanced cognitive abilities.
- PC: The PC is equipped with a keyboard and a display. It is connected to a bank server via the internet. A web browser is installed on the PC. The user uses the web browser to remit his or her money that is stored in the internet banking service. The PC (in fact, the browser) is a machine; it therefore has a high computing power (and memory capacity) and does not have advanced cognitive abilities.

Step 1. The user inputs money transfer information X, e.g., account number, amount of money, to the PC.

Step 2. The PC (web browser) sends the information to the bank server.

[1] In this paper, we define secure communication as preventing direct pecuniary damage caused by the falsification of malware.

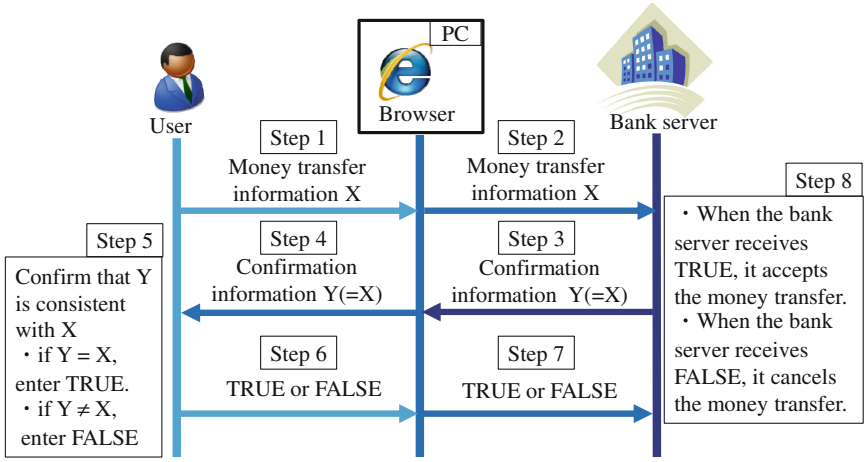

Fig. 1. Money transfer protocol

Step 3. To confirm the information, the bank server sends confirmation information Y to the PC. Typically, Y is consistent with X (Y = X).
Step 4. The PC receives Y and displays it to the user.
Step 5. The user confirms that Y is consistent with X and determines whether to perform the money transfer.
Step 6. When the user agrees with the remittance, the user inputs TRUE (the decision of money transfer) to the PC. When the user wants to cancel the remittance, the user inputs FALSE (the cancel of money transfer).
Step 7. The PC sends the TRUE or FALSE to the bank server.
Step 8. When the bank server receives TRUE, it accepts the money transfer. When the bank server receives FALSE, it cancels the money transfer.

2.2 MITB Attacks

MITB attacks can be classified into two types: information falsification and ID theft [4, 5]. As this paper is the first stage of the research, we focus only on the former type. In this type, malware in a PC (web browser) falsifies the transaction information. The procedure for this type of attack is shown in Fig. 2.

Step 1. The user inputs money transfer information X to the PC.
Step 2. The malware in the PC (web browser) alters X to X′ and sends it to the bank server.
Step 3. The bank server sends confirmation information Y (= X′) to the PC.
Step 4. The PC receives Y (=X′). The malware in the PC alters Y to Y′ (=X) and displays it to the user.

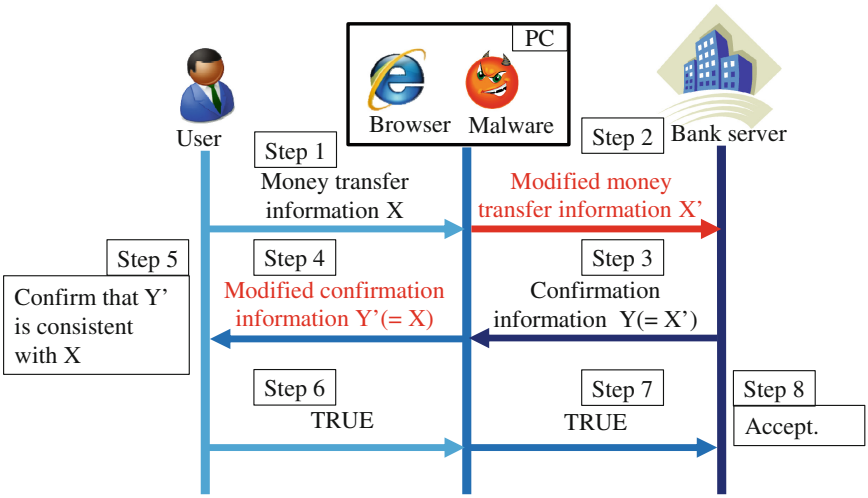

Fig. 2. Information falsification MITB attack

Step 5. The user reads Y′ (=X) and confirms whether it is consistent with X. Since the malware alters Y (=X′) to Y′(=X) in Step 4, the user accepts the money transfer.
Step 6. The user inputs TRUE (the decision of money transfer) to the PC.
Step 7. The PC sends TRUE to the bank server.
Step 8. The bank server receives TRUE and accepts the money transfer (X′).

3 Secure Communication Protocol Between Human and Server

3.1 Concepts

We propose an approach to preventing the information falsification type of MITB attack shown in Fig. 2 by constructing a secure communication channel between a machine and a human. A secure communication channel between a machine (web browser) and a machine (bank server) cannot prevent MITB attacks. This is because the malware in the PC can take over the operation of the web browser. To fundamentally prevent the attacks, it is necessary to use a human as a computational resource and construct a secure communication channel between a machine (bank server) and a human (end user).

It should be noted that, however, humans have only low computing powers. We cannot use a well-known encryption technique such as SSL. Instead, a method that enables secure communication between a machine (who has a high computing power) and a user (who has a low computing power) is needed. We propose a challenge and response protocol using a “challenge that malware in the browser cannot see.”

3.2 Proposed Protocol

Goal. With the information falsification type of attack, malware is able to falsify the information in Step 2 (money transfer information), Step 4 (confirmation information), and Step 7 (TRUE/FALSE) in Fig. 2. In this paper, we construct a secure communication protocol that prevents direct pecuniary damage to a user and bank caused by falsification in Steps 2, 4, and 7.

How to send a challenge that malware cannot see. Our protocol works under the assumption that a bank server is able to send the user a challenge that the malware in a browser cannot see. Let us consider that the server sends a set of data $\alpha_1 \sim \alpha_m$ through a certain type of channel to the user. The data is denoted as $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$. If the following requirements are met in the channel, the malware will not be able to see the challenge that is sent to the user through the channel.

(i). The malware cannot obtain any data $\alpha_i (1 \leq i \leq m)$ from $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$.
(ii). Even if the malware knows a piece of data $\alpha_i (1 \leq i \leq m)$, the malware cannot find which portion of data $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ stands for α_i.
(iii). The user (human) can obtain all data $\alpha_i (1 \leq i \leq m)$ from $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$.

It should be noted here that the malware also can use the channel to send the user a fake set of data $\beta_1, \beta_2, \ldots, \beta_n$. In other words, the malware can generate any fake data $\beta_1, \beta_2, \ldots, \beta_n$ from scratch. This means that if the malware knows the value of data $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ in $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$, it is able to change some values among them (e.g., $\alpha_1 \rightarrow \beta_1$ and send $\{\beta_1, \alpha_2, \ldots, \alpha_m\}$ to the user. However, due to the definitions (i), the malware cannot carry out this falsification. It should be also noted that the malware is able to conduct replay attacks by capturing a genuine$\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ and resending it to the user. However, due to the definitions (ii), the malware cannot alter $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$.

Although there could be various approaches used to develop this sort of channel, in this paper, we will apply CAPTCHAs (Completely Automated Public Turing test to tell Computers and Humans Apart [6]) to implement it, as explained later in Sect. 4. We thus refer to the channel as a "CAPTCHA channel."

Procedure. The proposed protocol works as shown in Fig. 3. Note that {Y, R} means that a pair of data Y and R is conveyed through the CAPTCHA channel, where R is a random number generated by the bank server.

Step 1. The user enters money transfer information X to the PC.
Step 2. The PC (web browser) sends X to the bank server.
Step 3. To confirm X, the bank server sends confirmation information {Y, R} to the PC. Here, Y is equal to X, and R is a random number generated by the bank server.
Step 4. The PC receives {Y, R} and displays it to the user. The user obtains Y and R from {Y, R}.
Step 5. The user confirms that Y is consistent with X and decides whether to remit or cancel.

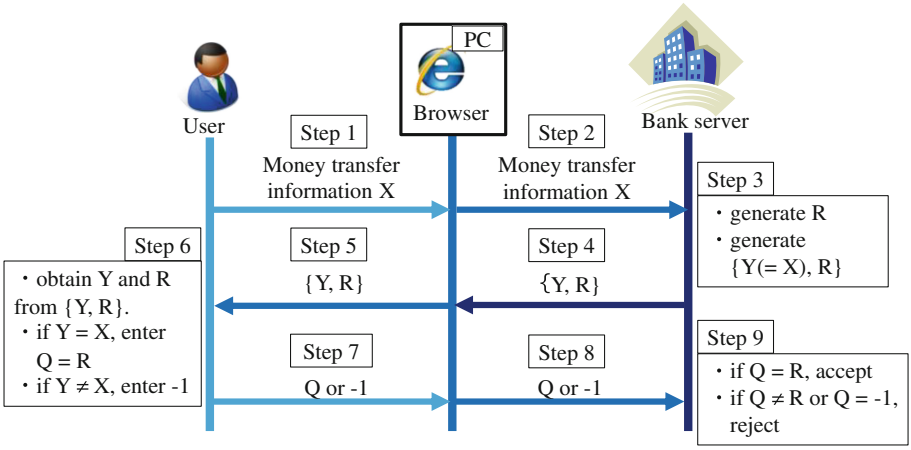

Fig. 3. Proposed protocol

Step 6. When the user wants to remit, the user inputs Q (=R) to the PC. When the user wants to cancel the remittance, the user inputs −1 to the PC.
Step 7. The PC sends Q or −1 to the bank server.
Step 8. When the bank server receives Q that is consistent with R, it accepts the money transfer. When the bank server receives −1 or a value that is not consistent with R, it cancels the money transfer.

3.3 Safety Evaluation

To evaluate the safety of the proposed protocol (Fig. 3), we verified the effectiveness of the protocol under the situation where all combinations of falsifications have occurred.

Table 1 shows the results of the safety evaluation against all combinations of falsification. X, Y, R, and Q falsified in each step are denoted as X′, Y′, R′, and Q′, respectively. "No." is an index to distinguish the falsifications.

As shown in Table 1, falsifications No. 3, 4, 7, 9, 11–13, 15, 18, 21, 23, and 26–28 are 'impossible' owing to the definitions of the CAPTCHA channel in Sect. 3.2. (To be more precise, the malware cannot conduct falsifications highlighted in gray owing to the definition of the CAPTCHA channel in Sect. 3.2.) For example, for No. 3, to alter {Y(=X′), R} to {Y′(=X), R} in Step 5, the malware has to (1) modify {Y(=X′), R} to {Y′(=X), R} or (2) generate {Y′(=X), R} from scratch. Regarding (1), though the malware knows the value of X′, it cannot find the position of X′ from {Y(=X′), R} owing to definition (ii) of the CAPTCHA channel. Thus, the malware cannot modify {Y(=X′), R} to {Y′(=X), R}. Regarding (2), though the malware knows the value of X′, it cannot obtain the R from {Y(=X′), R} owing to definition (i) of the CAPTCHA

Table 1. Results of safety evaluation

No.	Step(s) in which falsification is conducted	Data in each Step									Possibility
		Step 1	Step 2	Step 3	Step 4	Step 5	Step 6	Step 7	Step 8	Step 9	
1	Step 2	X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y(=X'), R}	Y≠X	-1	-1	Reject	Possible
2	Steps 2,5	X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y(=X'), R'}	Y≠X	-1	-1	Reject	Possible
3		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(= X), R}	Y' = X	Q = R	Q	Accept	Impossible
4		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(≠X), R}	Y' ≠X	-1	-1	Reject	Impossible
5		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(=X), R'}	Y' = X	Q = R'	Q	Reject	Possible
6		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(≠X), R'}	Y' ≠X	-1	-1	Reject	Possible
7	Steps 2,8	X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y(=X'), R}	Y≠X	-1	Q' = R	Accept	Impossible
8		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y(=X'), R}	Y≠X	-1	Q' ≠ R	Reject	Possible
9	Steps 2,5,8	X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y(=X'), R'}	Y≠X	-1	Q' = R	Accept	Impossible
10		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y(=X'), R'}	Y≠X	-1	Q' ≠ R	Reject	Possible
11		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(=X), R}	Y' = X	Q = R	-1	Reject	Impossible
12		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(=X), R}	Y' = X	Q = R	Q' ≠ R	Reject	Impossible
13		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(≠X), R}	Y' ≠X	-1	Q' = R	Accept	Impossible
14		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(≠X), R}	Y' ≠X	-1	Q' ≠ R	Reject	Possible
15		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(=X), R'}	Y' = X	Q = R'	Q' = R	Accept	Impossible
16		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(=X), R'}	Y' = X	Q = R'	Q' ≠ R	Reject	Possible
17		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(=X), R'}	Y' = X	Q = R'	-1	Reject	Possible
18		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(≠X), R'}	Y' ≠X	-1	Q' = R	Accept	Impossible
19		X	X'	{Y(=X'), R}	{Y(=X'), R}	{Y'(≠X), R'}	Y' ≠X	-1	Q' ≠ R	Reject	Possible
20	Step 5	X	X	{Y(=X), R}	{Y(=X), R}	{Y(=X), R'}	Y = X	Q = R'	Q	Reject	Possible
21		X	X	{Y(=X), R}	{Y(=X), R}	{Y'(≠X), R}	Y' ≠X	-1	-1	Reject	Impossible
22		X	X	{Y(=X), R}	{Y(=X), R}	{Y'(≠X), R'}	Y' ≠X	-1	-1	Reject	Possible
23	Steps 5,8	X	X	{Y(=X), R}	{Y(=X), R}	{Y(=X), R'}	Y = X	Q = R'	Q' = R	Accept	Impossible
24		X	X	{Y(=X), R}	{Y(=X), R}	{Y(=X), R'}	Y = X	Q = R'	Q' ≠R	Reject	Possible
25		X	X	{Y(=X), R}	{Y(=X), R}	{Y(=X), R'}	Y = X	Q = R'	-1	Reject	Possible
26		X	X	{Y(=X), R}	{Y(=X), R}	{Y'(≠X), R}	Y' ≠X	-1	Q' = R	Accept	Impossible
27		X	X	{Y(=X), R}	{Y(=X), R}	{Y'(≠X), R}	Y' ≠X	-1	Q' ≠ R	Reject	Impossible
28		X	X	{Y(=X), R}	{Y(=X), R}	{Y'(≠X), R'}	Y' ≠X	-1	Q' = R	Accept	Impossible
29		X	X	{Y(=X), R}	{Y(=X), R}	{Y'(≠X), R'}	Y' ≠X	-1	Q' ≠ R	Reject	Possible
30	Step 8	X	X	{Y(=X), R}	{Y(=X), R}	{Y(=X), R}	Y = X	Q = R	Q' ≠ R	Reject	Possible
31		X	X	{Y(=X), R}	{Y(=X), R}	{Y(=X), R}	Y = X	Q = R	-1	Reject	Possible

channel. Thus, the malware cannot generate {Y′(=X), R} from scratch. Similarly, the malware cannot do the other falsifications. As also shown in Table 1, falsifications No. 1, 2, 4–6, 8, 10–12, 14, 16, 17, 19–22, 24, 25, 27, and 29–31 are 'rejected' owing to the condition $Q \neq R$ or $Q = -1$ in Step 9.

As a result, all the possible illegal money transfers in Table 1 are 'rejected' or 'impossible'. Thus, the proposed protocol is safe against all combinations of falsification patterns. This means that the protocol realizes a secure communication channel between a machine and a human, so it enables MITB attacks shown in Fig. 2 to be prevented.

4 CAPTCHA Channel

4.1 How to Construct a CAPTCHA Channel

The proposed protocol is safe under the assumption that the CAPTCHA channel can be constructed. In this section, we explain a method for applying a CAPTCHA to develop the channel. A CAPTCHA is a Turing test to discriminate humans from machines [6] by using questions that a human can solve easily but a machine cannot.

A CAPTCHA used for the channel needs to meet definitions (i), (ii), and (iii) in Sect. 3.2. Hereafter, a CAPTCHA that meets the definitions and conveys a set of data $\{\alpha_1, \alpha_2, \ldots, \alpha_m\}$ as its answer is defined as $\mathrm{Cd}(\alpha_1, \alpha_2, \ldots, \alpha_1)$. Using $\mathrm{Cd}(\alpha_1, \alpha_2)$, the proposed protocol is described as in Fig. 4. So far, we have been able to send only one digit by using a CAPTCHA as will be explained later. The user needs to repeat our protocol for n times to send n digits of money transfer information.

To meet definition (i), Cd(Y, R) must be a CAPTCHA that machines cannot solve. Many researchers reported that some CAPTCHAs can be solved by machines [7, 8]. Such CAPTCHAs cannot be used as the Cd(Y, R). To meet definition (ii), Cd(Y, R) must be a CAPTCHA where machines cannot find the position of Y and/or R from Cd (Y, R). Figure 5 is an example of a CAPTCHA which does not meet definition (ii) and hence the malware is able to falsify[2]. Such CAPTCHAs cannot be used as the Cd(Y, R), either. To meet definition (iii), Cd(Y, R) must be a CAPTCHA that is human readable. Basically, Cd(Y, R) meets definition (iii) since the CAPTCHA is created with a question that humans can solve easily.

4.2 Example of CAPTCHAs that Meet the Definitions

There could be various CAPTCHAs that realize Cd(Y, R). Figure 6 shows an example. The CAPTCHA is composed of upright objects, upside-down ones, and other objects. This CAPTCHA requests users to count the number of upright objects and the number of upside-down objects. The number of upright objects is Y, and the number of upside-down objects is R.

This CAPTCHA meets definitions (i), (ii), and (iii) as follows. Since malware does not have an ability to recognize whether an object is upright or upside-down [9], it cannot obtain Y and R from this CAPTCHA. Therefore, this CAPTCHA meets definitions (i) and (ii). In addition, understanding upright/upside-down objects and

[2] In the proposed protocol shown in Fig. 3, when the malware obtains X from the user in Step 1 and sends X′ to the bank server in Step 2, the bank server sends Cd(Y(=X′), R) to the PC in Step 3. Suppose that Cd(X′, R) is a CAPTCHA shown in Fig. 5, where X′ is a position of an upright object and R is a position of an upside-down object. This CAPTCHA does not meet the definition (ii) and therefore the malware can find these positions in Cd(X′, R). The malware knows the value of X entered by the user and X′ sent by itself, and it can find the positions corresponding to X′ and X in Cd (X′, R). Then, the malware replaces the position of the X′-th object with the position of the X-th object in Cd(X′, R). Consequently, Cd(X′, R) becomes Cd(X, R). (Even though the malware is not capable of understanding which object is upright and/or upside-down, it can just replace these two objects with each other in Cd(X′, R).) The malware sends it to the user in Step 5, and the transfer of illegal money is successful.

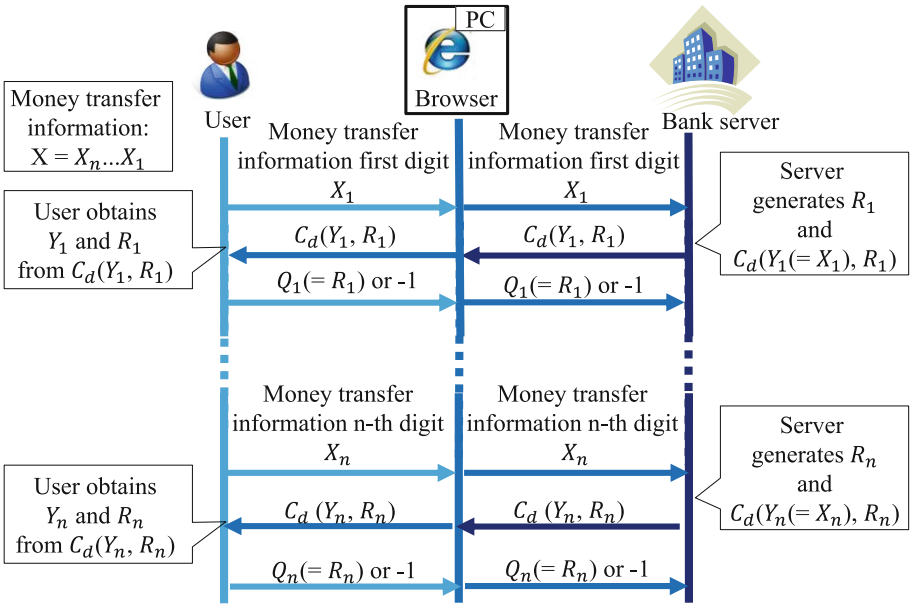

Fig. 4. Money transfer protocol using Cd(Y, R)

Fig. 5. Example of CAPTCHA that does not meet the definition (ii). Each position from the left object to the right one corresponds to values 0∼9. Among them, the position of upright object corresponds to Y, and the position of upside-down object corresponds to R (in this example, Y = 1 and R = 6)

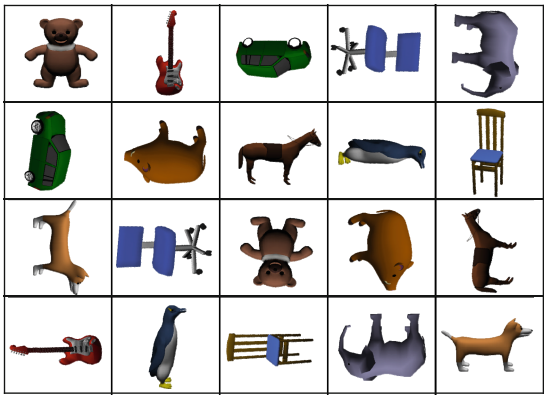

Fig. 6. A CAPTCHA that meets the definition of a CAPTCHA channel. The number of upright objects is Y. The number of upside-down objects is R (in this case, Y = 6 and R = 4)

counting them is an easy task for humans [9], so this CAPTCHA meets definition (iii). One weakness of this CAPTCHA is that Y and R should be a small number, e.g., a one-digit number; otherwise, it would take a lot of time for users to count them. That is the reason why the proposed protocol described in Fig. 4 sends the money transfer information one digit number by one digit number (i.e., the range of Y is $0 \sim 9$). The range of R is also $0 \sim 9$ in our proposal (Fig. 6). The design of a more effective CAPTCHA that meets all the definitions is one of the biggest future studies.

5 Consideration

5.1 Random Falsification Attack

In the proposed protocol, when the bank server receives a value that is consistent with R, it accepts the money transfer. If a value randomly generated by the malware in Step 8 happened to be consistent with R, the bank server accepts the money transfer. We refer to this attack as a "random falsification attack". The success probability of a random falsification attack is $1/|R|$, where $|R|$ is the order (i.e., the number of elements) of R. The range of R is $0 \sim 9$ in the CAPTCHA used in our protocol (Fig. 6). Thus, the success probability of this attack is 1/10.

As shown in Sect. 4, our protocol sends money transfer information one digit by one digit. To remit money to any account, the malware must succeed in falsifying all n digits. Thus, the success probability of an illegal money transfer is $(1/10)^n$. In typical Japanese banks, the length of an account number is seven digits, indicating that the probability $(1/10)^7$. Given the purpose of preventing illegal money transfers, the proposed method is considered to have a sufficiently high attack tolerance.

5.2 Usability

The user needs to repeat our protocol for n times to send n digits of money transfer information. The user has to solve the CAPTCHA n times. Compared with the conventional transfer protocol (Fig. 1), the proposed protocol places more of a burden on the user. We will experiment with evaluating usability in the future.

6 Related Work

6.1 Anti-malware

MITB attacks are caused by malware that infects a web browser. To prevent these attacks, the user should remove all malware in their web browser. Dedicated software, for example, PhishWall Premium [10], is able to detect and remove malware on the user's computer. However, given the current situation where subspecific malware is being created every day, the effect of the software is limited. In fact, it has been reported that it is difficult to detect Zeus, a typical malware to perform MITB attacks,

by using security software due to a large number of subspecies [11]. Our proposed method is not to detect the malware, and thus prevents MITB attacks even if the web browser is infected by malware.

6.2 Transaction Signing

Transaction signing is a measure using secure hardware that is independent of the PC (hereinafter referred to as "token") to prevent MITB attacks [12, 13]. The procedure of transaction signing is as follows. The user generates a verification code based on the money transfer information by using a token. The user sends the money transfer information and the verification code together to the bank server. The bank server receives them and verifies the integrity of the money transfer information and the verification code.

Two methods of transaction signing have been reported. One method uses a token distributed by the bank [12]. With this method, it is necessary for users to always carry the token. Users may lose or not carry the token outdoors. In addition, the bank has to incur huge costs to distribute the tokens to all users. The other method is using a smart phone as a token [13]. With this method, the above problem cannot occur. However, given a situation where it has been reported that there is a large number of malicious apps, it is difficult to ensure that the smart phones are secure hardware anymore. Even if we prevent any infection at present, malware is expected to evolve in the future. This problem is similar to that of the anti-malware shown in Sect. 5.1.

As shown in Fig. 3, the proposed method can be implemented without any additional device to the conventional transfer protocol (Fig. 1). Therefore, a problem as described above cannot occur.

7 Conclusion

In this paper, we proposed an approach to preventing MITB attacks by constructing a secure communication channel between a machine (bank server) and a human (end user). Developing a challenge and response protocol that achieves the proposed channel, we conducted a safety evaluation of the protocol. The results showed that the protocol works safely under the assumption that a bank server can send a "challenge that malware in the browser cannot see" to the user. Sending the challenge is feasible by applying CAPTCHA technology. We will consider a CAPTCHA that is more suitable for the proposed protocol and perform usability experiments.

References

1. Online banking users suffer ¥1.4 billion in damage. http://www.japantimes.co.jp/news/2014/01/30/national/online-banking-users-suffer-1-4-billion-in-damage/#.VriNThiLS00
2. Bank of Taiwan. http://www.bot.com.tw/English/BankServices/ElectronicBankingServices/BOTSSLInternetBanking/Pages/default.aspx

3. Bank of America. https://www.bankofamerica.com/onlinebanking/online-banking-security-faqs.go
4. Man-in-the-Browser (MitB). https://www.trusteer.com/en/glossary/man-in-the-browser-mitb
5. Man In The Browser attacks scare banking world. http://securityaffairs.co/wordpress/17538/cyber-crime/man-browser-attacks-scare-banking.html
6. The Official CAPTCHA Site. http://www.captcha.net
7. Yan, J., Ahmad, A.S.E.: Breaking visual CAPTCHAs with naïve pattern recognition algorithms. In: 2007 Computer Security Applications Conference, pp. 279–291 (2007)
8. Golle, P.: Machine learning attacks against the ASIRRA CAPTCHA. In: 2008 ACM CSS, pp. 535–542 (2008)
9. Ross, S.A., Alex Halderman, J., Finkelstein, A.: Sketcha: A captcha based on line drawings of 3D models. In: Proceedings of the 19th International Conference on World Wide Web, pp. 821–830 (2010)
10. PhishWall. http://www.securebrain.co.jp/eng/web/phishwall.php
11. Symantec White Paper - Banking Trojans. https://www4.symantec.com/mktginfo/whitepaper/user_authentication/21195180_WP_GA_BankingTrojansImpactandDefendAgainstTrojanFraud_062611.pdf
12. SafeNet eToken 3500. http://www.pronew.com.tw/download/doc/eToken3500_PB_(EN)_web.pdf
13. Saisudheer, A.: M. TECH: smart phone as software token for generating digital signature code for signing in online banking transaction. IJCES **3**(12), 1–4 (2013)