Reverse Turing Test using Touchscreens and CAPTCHA[*]

Mayumi Takaya[1], Yusuke Tsuruta[2], and Akihiro Yamamura[1†]
[1] *Akita University*
Akita, Japan
{msato, yamamura}@ie.akita-u.ac.jp
[2] *System Research*
Nagoya, Japan
tsuruta2013@gmail.com

Abstract

Smartphones play an important role in the information-communication society and accesses from smartphones to the Internet services increase more and more with the advent of cloud computing. Smartphones have comparatively small displays and it is possible for users to input data through touchscreens. The techniques such as a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) that prevent computer programs from automatically accessing to the Internet services require a traditional console. A challenge image is shown in a display and type a response in a keyboard when facing the existing CAPTCHAs, but this may cause a trouble for smartphone users. Thus, it may stunt the growth or development of smartphones. We propose a reverse Turing test using touchscreens and apply to construct a CAPTCHA suitable for smartphones making use of touchscreens to check whether or not embodied knowledge exists in the response data. We experiment and examine its validity, security and usefulness.

Keywords: Reverse Turing Test, CAPTCHA, Line Trace, Touchscreens

1 Introduction

Following [1] in which a CAPTCHA (Completely Automated Public Turing test to tell Computers and Humans Apart) suitable for smartphones using embodied knowledge of human beings is proposed, we analyze and discuss the CAPTCHA technique utilizing touchscreens to relieve the inconvenience caused by the existing CAPTCHAs when using smartphones. We carry out experiments to examine the usefulness and compared with the existing techniques. We report some experimental results in addition to the ones in [1]. Our motivation comes from rapid increase of smartphone users and urgent threats against the Internet plausibly caused by smartphones. Smartphones play an important role in the information-communication society nowadays and the development of cloud computing promotes the spread of smartphones. The increase of users of smartphones has influenced the use of the Internet as well.

A Turing test is a test that a human determines whether the other party is a computer or a human ([2]), and the reversing Turing test is a test that a computer determines whether the other party is a computer or a

[*]This paper is an extended version of the work originally presented at ICT-EurAsia 2013, Gadjah Mada University, Indonesia, March 2013 [1]. In this extended version, we add a new section on current technologies of touchscreens and a new subsection on experimental results on validity of the proposed technique in which we analyze each experiment in detail, and strengthen the discussion on security analysis. We show several experiments of tracing several characters and symbols, which are not in the original version, using a line tracer. These experiments clarify several weakness of line trace attacks with respect to character shape. Then we discuss the Pandemonium architecture, which is newly used in this extended version to examine character shape and its effect to the proposed technique. In conclusions, we augment future works, in particular, difference between humans and computers is discussed in detail.

[†]Corresponding author: Akita University, 1-1, Tegata Gakuen-machi, Akita 010-8502 JAPAN, Tel: +81-18-889-2799, Web(in Japanese only): `http://crypto.ie.akita-u.ac.jp/yamamura.html`

human. A CAPTCHA is one of the reverse Turing tests that distinguish an access by a computer program such as a crawler from an access by a human using the difference between humans' and computers' shape recognition ability ([3, 4]). There are many CAPTCHA techniques using a console such as a keyboard, a mouse or a display that are standard of the past computer models. There are many differences between smartphones and the past computer models from the point of view of human-computer interface. For example, a smartphone has non-traditional console such as a touchscreen. We consider utilizing such a non-traditional console to construct a new reverse Turing test and apply to a CAPTCHA technique.

For smartphones, data is inputted through the touchscreen by hands, and the display of a smartphone is comparatively small. A CAPTCHA login is requested when a user is accessing to the Internet services through a smartphone exactly same as it is requested to accesses from the desktop PCs. When we use smartphones and face a CAPTCHA, both the challenge image and the virtual keyboard are displayed, and we have to type in the word displayed in the image. However, the virtual keyboard occupies almost half of the display and so the CAPTCHA image must be small. Gossweiler, Kamvar, and Baluja propose CAPTCHA of the image base that displays the image that reverses the top and bottom as a new method that is appropriate for the portable terminal and returns it in the right place. To solve this problem, Gossweiler, Kamvar, and Baluja propose a new image based CAPTCHA in [5], which is based on identifying an image's upright orientation. Yet another technique is proposed in [1] and we analyze the technique in this paper.

One-stroke sketch is taken up as an ingredient in the embodied knowledge in [1]. A touchscreen that is one of the features of smartphones is suitable for faithfully acquiring one-stroke sketch data. One-stroke sketch input data with humans' finger is characterized as a continuous locus resulted by the human hand's physicality and realized as a series of coordinates on the display. The entire character image is not drawn at the same time but it is drawn continuously on the curve along the shape of the character little by little following the tracks of the tip of a finger according to the operation of the arm and the hand. Figure 1 shows the correct order to input α through a touchscreen.

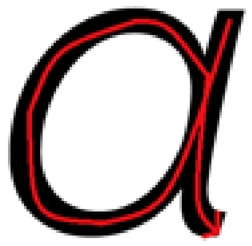

Figure 1: One-stroke sketch of α

We intensify our discussion on experiments and show more experimental results to justify our conclusions in addition to the ones in [1]. The paper is organized as follows. We add a brief explanation of the current touchscreen technologies used in smartphones in Section 2. We explain a new CAPTCHA based on one-stroke characters proposed by [1] in Section 3. We discuss its validity and usefulness in Section 4 and then analyze its security in Section 5. To intensify our discussion, we employ the Pandemonium architecture as a recognition model of humans. Our experimental results indicate that several features of the Pandemonium architecture can be employed to make effective challenge images which can be easily recognized by humans but may not be mounted by line tracer attacks. We discuss several issues and future works to analyze our proposal in the last section.

2 Touchscreens

The touchscreen is a standard input device for a smartphone and data can be inputed by using humans' fingers. Touchscreens have several advantages over the other pointing devices. For example, Shneiderman [6] mentioned the following.

- Touching a visual display of choices requires little thinking and is a form of direct manipulation that is easy to learn.
- Touchscreens have easier hand-eye coordination than mice or keyboard.
- No extra workspace is required as with other pointing devices.

The authors consider that human-computer interaction plays significant role in reverse Turing tests and so it is interesting to study a reverse Turing test using touchscreens. See [7] for more information on human-computer interaction technologies. E.A. Johnson described his work on capacitive touch screens in [8, 9]. Sears et al. [10] summarized single and multi-touch human-computer interaction. Touchscreens are classified into four categories: a resistance film method, an electrostatic capacity method, an optical method, and the supersonic wave method. The resistance film method was adopted in early cellular phone models. The electrostatic capacity method has attracted attentions since it was adopted for iPhone by Apple Computer in 2007. Most of Android OS based smartphones are now equipped with a touchscreen of the electrostatic capacity method. The electrostatic capacity method is divided roughly into the surface electrostatic capacity method and the mirror electrostatic capacity method. In particular, the ITO Grid method in the mirror electrostatic capacity method is adopted in a smart phone. The transparent conductive film of ITO (Indium Tin Oxides) is arranged like the grid to detect x and y coordinates on the glass or the PET (Polyethylene terephthalate) base of a touchscreen of the ITO Grid method, and the position of the finger touch is detected by an electrostatic touch sensor. In this paper, we consider a touchscreen of the electrostatic capacity method. Android OS 3.1 and 9.4 electrostatic capacity method touchscreen type WXGA liquid crystal are used in our experiment. The experiment program was made using the data I/O function Android OS offers.

The operations on a touchscreen include the flick, the drag, the tap, the touch, the pinch out, and the pinch in. The drag is an operation that traces a one-stroke character displayed on a touchscreen and our proposed technique is carried out using the drag. In our experimental environment, coordinates on a touchscreen are acquired once every 0.01-0.02 seconds. Note that the interval for data acquisition depends on system requirements. The multi-touch is to carry out more than one operation at the same time. Android OS fixes pointer ID to distinguish each touch in the multi touch at each input. The data obtained by dragging on the touchscreen is composed of a pointer ID, x-coordinate and y-coordinate according to the time series. The proposed technology should be adjusted according to touchscreen methodology. In fact it is presumed we need adjust data processing in CAPTCHA server according to touchscreen methodology. It is necessary to adjust a CAPTCHA for all touch screen methodology. It is important because the system requirement can be affected by physical aspect of the touchscreen methodology. In this paper we consider only the electrostatic capacity method and leave further study over later.

3 Proposed Technique

3.1 Basic Idea

As a standard authentication protocol, a CAPTCHA server sends a challenge and a user has to respond in a correct way. In the case of the proposed CAPTCHA, the server sends a challenge image that includes

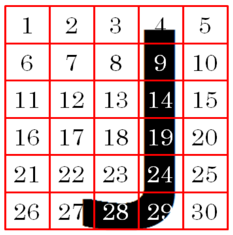

Figure 2: Character "J"

a one-stroke character (or a symbol) and then the user look at the image on a touchscreen and drag the character by a finger tip and sends back the data representing a one-stroke sketch as a response to the challenge. The touchscreen interprets the dragged data as a representation of the one-stroke character, which consists of time series of a pointer ID, *x*-coordinates and *y*-coordinates. The server receives the series of coordinates and check whether or not it is acceptable as a data obtained by a human using implicitly embodied knowledge. If the data received is judged as an output of a computer program then the access is rejected. A user obtains data, a series of coordinates, by dragging the character on the display. The server determines whether or not the data is acceptable by checking the locus is correct and continuous in addition to the correctness of the the starting point and the terminal point.

For instance, suppose that the character "J" is displayed as a challenge image as in Figure. 2. The correct response is obtained by dragging the character along the shape of "J" on the touchscreen. Coordinates of the inputted data, that is, series of coordinates of the locus are checked if the first coordinate is included in the small area 4, and if the following coordinates are in the small area 9 and so on. If the series of ordered coordinates is nearly in the order of the small areas 4, 9, 14, 19, 24, 29, 28, and 27, then it is accepted. If series of coordinates are in the order of the areas 2, 8, 14, 19, and 20, then the data is rejected.

When the server receives a request, the proposed CAPTCHA server send a one-stroke character image as a challenge and it is displayed. The user responds by dragging the shape of the character on the touchscreen. A correct stroke order can be considered as an embodied knowledge of humans. The input data is checked and judged by the CAPTCHA server. Humans may make an unintentional mistake in dragging operation and send back an unacceptable response to the CAPTCHA server. So humans may generate a false negative with non-negligible probability. Therefore, it is necessary to limit the number of trials according to applications. The possibility that a false positive is generated is extremely small as shown in Section 4.

3.2 Authorization

A one-stroke character image is transformed and divided into small rectangles of the same size (Figure 3). Every small rectangle including a part of the displayed character is selected (Figure 4). The selected small rectangles compose of the coordinates of the correct response. We now give an order to these rectangles. The rectangle including the position were humans to start a stroke is given the first order. If there are more than one such rectangles, all of them are given the first order. Next, the rectangles adjacent to the first order rectangles and including the position where finger passes when stroking the character are given second order. We repeat the process until we exhaust all the rectangle including the character part. For example, we illustrate the order of rectangles for α (Figure 5). Note that we give multi-order to the rectangles where a finger passes more than once such as the rectangles 23-27 in Figure 5.

We exemplify the authorization in the case that the one-stroke character α is selected. We follow the

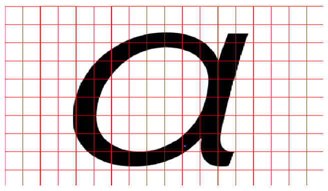
Figure 3: Partition

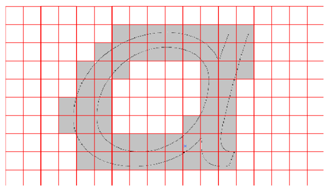
Figure 4: Character area

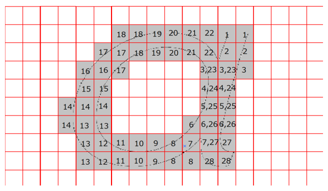
Figure 5: Stroke order

procedure below to decide whether or not a response is acceptable.

1 Receiving an access request from a user, the CAPTCHA server sends a challenge image of a one-stroke character to the user.

2 The user drags on the shape of the character displayed on a touchscreen by a finger and obtain data representing the stroke order of the displayed character.

3 The user sends back the data obtained by dragging to the CAPTCHA server as a response.

4 The CAPTCHA server compares the data received with the original character image and determine whether or not the data is accepted.

5 The CAPTCHA server authorizes the user if the data is accepted.

3.3 Comparison with the Existing CAPTCHA

We discuss the usefulness and advantage of the proposed CAPTCHA over the existing techniques provided that a user is trying to access via a smartphone. The screen size of smartphones is about 3.5-5 inch. When a user use a smartphone and face a CAPTCHA challenge, both a CAPTCHA image and a virtual keyboard are displayed on a touchscreen (Figure 6). Then the CAPTCH image may cover almost half of the touchscreen. It is inconvenient for most of the users to respond to the CAPTCHA challenge due to this limited size image and the virtual keyboard. In particular, if a CAPTCHA challenge is case sensitive then the inconvenience gets worse. A user may want to use a stylus, which is invented in [11], instead of fingers if possible. However, a stylus is not usually attached with smartphones and so using a stylus is not an ideal solution. Another possibility is a pointed fingernail (Figure 9) as a stylus. The concept first appeared in the science fiction “Scanners Live in Vain” written by Cordwainer Smith in 1950. The authors consider such a method is inconvenient for most of the smartphone users.

A user has to type more than once to input a character when using a virtual numeric keyboard. For example, when typing “c” in the lowercase letter, one has to press the button for “c” three times (Figure 7). When typing “C” in the uppercase letter, one needs more operations to change the lowercase mode to the uppercase mode. Therefore, the total number of operations becomes enormous if the words are arbitrarily generated using lowercase letters, uppercase letters and figures. Moreover, a wrong character may be inputted by an unintentional typing mistake. For this reason, some existing CAPTCHA use only figures $(0,1,2,\dots,9)$ without using alphabets to improve user’s convenience. Note that if uppercase and lowercase letters are allowed in addition to the figures, $62(=10+52)$ characters can be used. This results in the deterioration of the security; if the challenge is a word consisting of n letters, there are only 10^n cases compared with 62^n cases.

When using the proposed CAPTCHA, the entire display is used for showing the challenge image, and the input is comparatively easy (Figure 8); no additional operations such as changing modes are required.

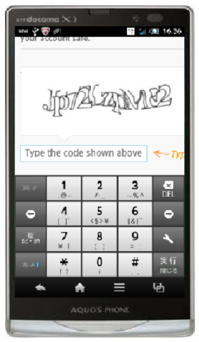

Figure 6: Numeric keyboard

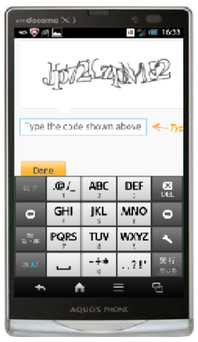

Figure 7: Telephone keypad

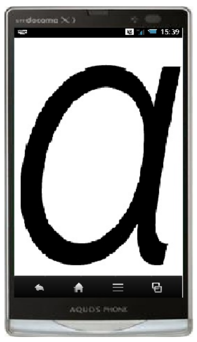

Figure 8: Proposed method

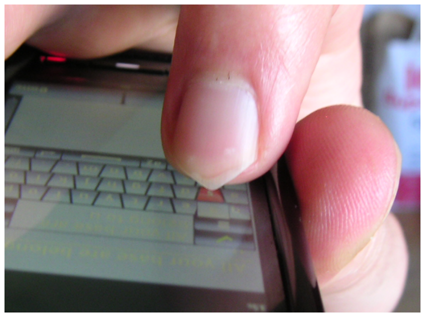

Figure 9: Pointed fingernail

4 Validity of the Proposed CAPTCHA

We examine the validity of the proposed CAPTCHA by experiments; 22 subjects (humans) are asked to respond to several challenge images that represent the symbol "α" and verify that humans can respond in a correct way.

4.1 Experiments

We use a handheld computer (Android 3.1 and processor NVIDIA Tegra 2 mobile processor) equipped with 9.4 type WXGA liquid crystal with the internal organs display touchscreen of the ITO Grid method mirror electrostatic capacity method as the users' machine. The CAPTCHA server is constructed on Windows 7 Professional 64bit, 2048MB memory, and Intel Core i3, and the authentication program is written by using c/c++ compiler MinGW. The size of the challenge images is 1200×700 pixel. The response is accepted if the data inputted by a user passes the character image in a correct order.

The purposes of experiments are summarized as follows (see Table 1). In the experiment 1 and 2, the small zone is set 35×35 pixels and 70×70 pixels, respectively, and we investigate the differences between these two cases. In the experiment 3, the small zone is set 35×35 pixels and we specify the entry speed and the input position and investigate the difference between these cases. In the experiment 4, we investigate the effect caused by the change of characters. In the experiment 5, we investigate the case that the response is accepted only when all set coordinates are passed. In the experiment 6, we investigate the tolerance for human non-intentional errors.

Experiment 1 The instruction "Please trace on the character shape by one stroke" is displayed and Image 1 (Figure 10) is displayed. The small zone on the grid is 70×70 pixel (Figure 12).

Experiment 2 The instruction "Please trace on the character shape by one stroke" is displayed and Image 1 is displayed. The small zone on the grid is 35 × 35 pixel (Figure 13).

Experiment 3 The instruction "Please trace on the character shape by one stroke within 5 seconds" is displayed and Image 1 is displayed. The small zone on the grid is 35 × 35 pixel (Figure 13).

Experiment 4 The instruction "Please trace on the character shape by one stroke within 5 seconds" is displayed and Image 2 (Figure 11) is displayed. The small zone on the grid is 35 × 35 pixel (Figure 14).

Experiment 5 The instruction "Please trace on the character shape by one stroke within 5 seconds" is displayed and Image 2 is displayed. However, the response is accepted only when every small zone from 1 to 40 is passed in order. The small zone on the grid is 35 × 35 pixel (Figure 14).

Experiment 6 The instruction "Please perform the input which is not related to the displayed character" is displayed and Image 1 is displayed. The small zone on the grid is 35 × 35 pixel (Figure 13).

Table 1: Experiments

	Exp 1	Exp 2	Exp 3	Exp 4	Exp 5	Exp 6
Character α	√	√	√	-	-	√
Character β	-	-	-	√	√	-
35 × 35 pixel	-	√	√	√	√	√
70 × 70 pixel	√	-	-	-	-	-
Specified time (about 5 seconds)	-	-	√	√	√	-

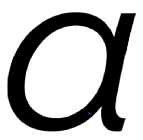
Figure 10: Image 1

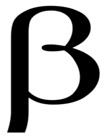
Figure 11: Image 2

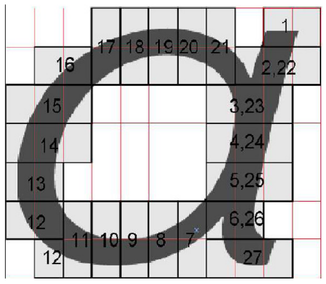
Figure 12: exp 1

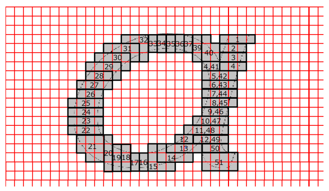
Figure 13: exp 2-3

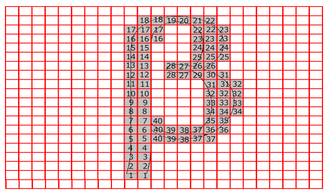
Figure 14: exp 4-5

4.2 Result of Experiments

4.2.1 Experiment 1

We set 70×70 pixel for the small region. The input time is not specified in this experiment and it took about two seconds and the number of coordinates obtained is about 150 on the average. All subjects except for the ones who wrote in an incorrect order are accepted; 95.5% of 21 subjects are accepted. This becomes a result in which having accepted all the input except the tests who did an input different from the assumed writing. Because regions are relatively large and an input deviated from the correct one may be accepted, the acceptance rate is high in this case. We conclude that the acceptance rate gets higher by making regions larger, however, input not related to the displayed character may also be accepted.

4.2.2 Experiment 2

We set 35×35 pixel for the small region in Experiment 2. The input time is not specified as Experiment 1, and it took about two seconds and the number of coordinates obtained is about 150 on the average. Because the size of regions is set small, most of obtained input data lie on the character image. The acceptance rate is 90.9% and so lower than Experiment 1. We saw several input data like 1, 2, 40 instead of 1, 2, 3, and the acceptance rate gets lower compared with Experiment 1. We saw similar input data in Experiment 1, however, the input data is accepted in Experiment 1.

4.2.3 Experiment 3

In Experiment 3, the input time to trace the character is to set five seconds. In addition, we also ask them to pay attention to follow the center of the character image. Because we set the input time, it took about four to six seconds and the number of coordinates obtained is about 300 on the average. The input time is long and so the acceptance rate improve to 95.5%. By the result, we conclude that the CAPTCHA system works even if the size of the small region is not set to 35×35 pixel. Moreover, the number of the input order such as 1, 2, and, 40 decreases in this experiment, and the number of obtained coordinates increases.

4.2.4 Experiment 4

In Experiment 4, the character β is used. Input time is not specified in this experiment and the acceptance rate is 100%. Input time and the number of the obtained coordinates are almost same as Experiment 1. We saw several errors with incorrect order such as 17 and 19, but it is within 10% different from the correct order input.

4.2.5 Experiment 5

The character β is used and input time is not specified like Experiment 4. On the other hand, input data is accepted only when every small zone from 1 to 40 is passed in order in Experiment 5 although it is accepted when 90 percent of small regions are included in Experiment 4. The acceptance rate is 40.9%. It is low as compared with Experiment 4. If only one incorrect data such as 17 and 19 is inputed, then the data is rejected. Thus the acceptance rate turns out to be low compared with Experiment 4.

4.2.6 Experiment 6

The objective of Experiment 6 is to guarantee that any irrelevant input is not accepted by accident. The number of obtained coordinates lie between 30 and 700, and no input is accepted as expected. Therefore, it is thought that the possibility that the false negative happens is extremely low.

4.3 Summary of Experimental Results

The results of the experiments in Section 4.1 are summarized in Table 2. In the experiment 1, 2, and 3, one subject is rejected because the responding data is in the order corresponding to the alphabet "a". Recall that the image indicates the symbol "α". When one writes "α", the order is different from the alphabet "a" although the shape is similar. The difference of the handwritten input of "α" and "a" is due to the culture and a social background in which the subject has grown up, and this is considered an embodied knowledge.

By the results of the experiments 1 and 2, we can conclude that if the zone is bigger, then higher acceptance rate is achieved, on the other hand, if the zone is smaller, then acceptance rate decreased. By the results of the experiments 2 and 4, we can conclude that the shape of the character does not affect the acceptance rate and the acceptance rate is stable for any (simple) character. We are convinced that other characters which are written as one-stroke sketch other than "α" can be used in the proposed CAPTCHA as well. By the result of the experiment 3, we can conclude that if we allow users to write slowly then the acceptance rate will increase but the transmission data gets larger, which is not desired for network congestions. By the results of the experiments 4 and 5, we can conclude that it is necessary to permit width of the order of the inputted coordinates to some degree, that is, we must be tolerant to small errors data, possibly caused by an unintentional errors.

Table 2: Experimental Results

	Exp1	Exp2	Exp3	Exp4	Exp5	Exp6
Number of Subjects (People)	22	22	22	22	22	22
Acceptance Number (Time)	21	20	21	22	9	0
Acceptance Rate (%)	95.5	90.9	95.5	100	40.9	0

5 Security Analysis

The main objective of a CAPTCHA is to prevent computer programs from automatically accessing to network services for evil purposes. Therefore, the attacker against a CAPTCHA is a computer program disguising itself as a human being and trying to obtain a legitimate authority to access. Then the security of a CAPTCHA is evaluated by the intractability for computer programs to behave like a human ([3, 12]). We analyze conceivable attacks against the proposal CAPTCHA.

We suppose that the image displayed as a challenge is monochrome, and the character is drawn in the white ground in black; the effect of color should be studied in detail in future. In this case, a computer program can acquire accurately the coordinate data of the area where the character is drawn by examining RGB of the challenge image. Then a computer program has only to enumerate a series of coordinates at which black RGB is appointed and give order to these coordinates according to the correct stroke order of the one-stroke character. For example, if a human write the character "J" then the input data should follow the small areas 4, 9, 14, 19, 24, 29, 28, and 27 like in Figure 2. The number of the coordinates should be nearly same as the standard input by human beings to disguise. In general, a computer program has no information on one-stroke sketch, which is considered as an embodied knowledge of human beings. Each human being has learned such an embodied knowledge since their childhood. A computer program should pick one position from the area of a coordinates with black RGB as the starting point and also as the terminal point and then compose a series of coordinates with black RGB that connects the starting and terminal points in a correct order. It is impossible to execute this task if there is no information on the correct stroke order of the challenge character. If many responses are permitted to the same challenge image, the brute force attack becomes possible in principle. However, the brute force attack can be avoided by permitting only one response to each challenge. Therefore, it is realistic to put the limitation on the challenge frequency.

Now suppose that an attack program can use a database concerning correct stroke order of one-stroke characters. If a program can obtain a series of coordinates with black RGB correctly, then using information on the correct order stroke of one-stroke characters from the database the computer program might be able to deduce a plausible stroke order. We note that it costs a lot to make such a database for attack against the CAPTCHA and so this already has some deterrent effect. In addition, the challenge image is not necessarily based on a one-stroke character or a symbol. An arbitrary curve can be used for a challenge instead of a one-stroke character as far as the curve can be written by one stroke. Making the database on stroke order of arbitrary curves is impossible in principle. It is necessary to investigate humans inclination to a way to follow curves. We may execute transformations on the shapes (and colors) of the character to perplex computer programs. If the transformation processing is a continuous transformation, this occurs no trouble for human beings. The authors set up a hypothesis that humans are better than computer programs with respect to recognition of topologically isomorphic transformation although we do not precisely define a topologically isomorphism here. This difference can be used to distinguish humans and computer programs and this should be studied in the future work.

It seems difficult for computer programs to respond correctly to distorted character image (Figure 17). If the challenge is a color image, the attacker's program has to carry out an edge detection and specify the character. Using the existing CAPTCHA techniques such as adding the distortion to the character, we can make an attacker difficult to detect the character. Moreover, we can employ not only adding the distortion transformation but also camouflaging the background with the dazzle paint. Therefore, the security of the proposed CAPTCHA is at least the existing CAPTCHA techniques because these can be applied to the proposed CAPTCHA. In addition, the method requiring the user to input more than one one-stroke character is effective to improve the security level. The security level can be adjusted according to the system requirement.

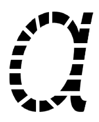
Figure 15: Nicked character

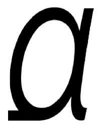
Figure 16: Dead end

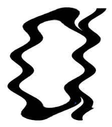
Figure 17: Distorted character

5.1 Line Trace Attack

It seems possible to use the line trace program, which is often used in a robot, to trace the black color area of a challenge image for attacking the CAPTCHA. For this attack, the line trace program has to trace the black color area from the starting point of the character to the terminal point in order to compose the response data. The attack using a line trace program seems a plausible attack as of now. It is necessary for a line trace program to find the starting point to begin tracing, however, it seems intractable to find the starting point because the line trace program checks the local area and determines the next action and does not know it. The starting point is usually given as the input to the line trace program by a human being. A human being looks at the image, comprehends the character and finds the starting point using the embodied knowledge. On the other hand, choosing a starting point is intractable for a line trace program. If a human takes part in the attack, the attacker consists of not only a program but a human, and so this approach is excluded as an attack against a CAPTCHA in principle because an objective of a CAPTCHA is to prevent programs from automatically accessing to a server without human beings assistance. Even if the starting point is obtained in some ways without humans assistance, our approach allows challenges such as a nicked character image (Figure 15), a character with a dead end (Figure 16), or a distorted character image (Figure 17) to perplex the line trace program, which give no trouble to human beings. Therefore, an attack using line trace programs seems intractable.

5.2 Experiments

We examine attacks against the proposed CAPTCHA using a line trace simulator. In the following experiments, a line trace program tries to construct a correct response to challenge images below. Each experiment is executed provided the starting point is given to the program beforehand by a human. Therefore the attack does not complete the objective of attacking a CAPTCHA. However we can roughly evaluate the security of the proposed CAPTCHA. In this experiment, we use a free software Line Trace Ver1.1 (can be downloaded from `http://www.yamamoto-works.jp`).

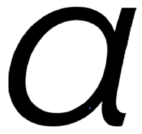
Figure 18: Alpha

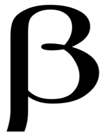
Figure 19: Beta

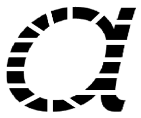
Figure 20: Nicked alpha

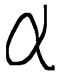
Figure 21: Handwritten alpha

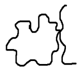
Figure 22: Distorted alpha

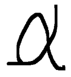
Figure 23: Alpha with a dead end

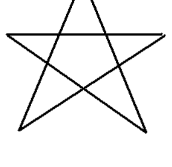
Figure 24: Star

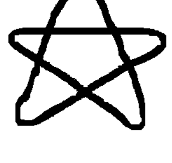
Figure 25: Handwritten star

The line trace program succeeded in making an acceptable response only to the challenge images in Figure 21, Figure 25 and Figure 27. It failed in the other images; see Table 3 for the summary of the result. Note that the character in Figure 26 has two parts and so the line tracer cannot trace the second

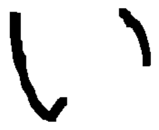

Figure 26: Japanese hiragana

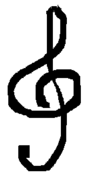

Figure 27: Handwritten G clef

part of the character. For the same reason, the line tracer cannot trace Figure 20. In Figure 23, the line tracer enter the dead end and failed to come out there. This is also similar to the failure in Figure 19; see Figure 30. We conclude that countermeasures leading the line trace program to a dead end (Figure 23) or putting nicks in the character shape (Figure 20) are considerably effective whereas these do not cause any troubles to human beings.

The tracer succeeds in Figure 21 but fails in Figure 18 although both represent the character α. Figure 18 is generated by transforming from a standard font whereas Figure 21 is handwritten. The tracer succeeds only in Figure 21 because the two lines clearly cross at the junction; see Figure 28. On the other hand, the junction is unclear in Figure 18; see Figure 29. Similarly, the tracer succeeds in Figure 27 although the image looks complicated. These experiments indicate that line trace program's inclination; it tends to go straight when it faces a junction. There are three junctions in Figure 27 and going straight is always correct stroke order.

The line trace program fails to trace when the angle is sharp in the character shape. For example, the line tracer fails to trace Figure 24. Angles are important characteristics in recognizing characters for humans. Selfridge [13, 14] introduced the Pandemonium architecture to explain image constancy phenomena. In this architecture, an image is first perceived in its part before the whole. The architecture is composed of several groups of demons working independently to process the visual stimulus. It consists of image, feature, cognitive and decision demons. Each group of demons is assigned to a specific stage in recognition. According to the architecture, the alphabetical character is recorded in the retina as an image demon and is compared with feature demons, which are representing specific feature such as vertical lines, horizontal lines, oblique lines, angles and so on. Next the cognitive demon closest to the combination of feature demons is selected and it calls decision demons. Then it recognizes a specific character. The feature demons are classified into the vertical line, the horizontal line, the oblique line, the right angle, the acute angle, the continuous curve and the discontinuous curve. In images in the experiments in Section 5.2, all aspects except the discontinuous curve can be found. We conjecture that a line tracer is not good at tracing acute angles and return at some point. The reader is referred to [15] for more information on the Pandemonium architecture. The experiment (Figure 24) indicates that Pandemonium architecture affects not only humans' recognition but line tracers for a possibly different reason.

Table 3: Experimental Result

	Fig.18	Fig.19	Fig.20	Fig.21	Fig.22
Trace Success	-	-	-	√	-
Trace Failure	√	√	√	-	√
	Fig.23	Fig.24	Fig.25	Fig.26	Fig.27
Trace Success	-	-	√	-	-
Trace Failure	√	√	-	√	√

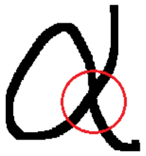
Figure 28: Junction

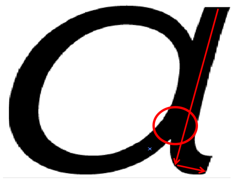
Figure 29: Failure in α

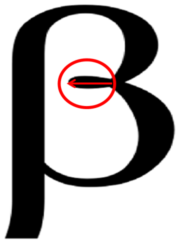
Figure 30: Failure in β

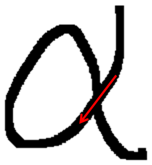
Figure 31: Success in handwritten α

We show some of experimental results corresponding to Figures 24, 25, 26, and 27. The figures indicate the point where the line tracer starts and stops.

The result on Figure 25 is interesting. The line tracer succeeded in tracing even though the image includes the symbol representing a star like Figure 24. The image in Figure 24 contains an acute angle and so the line tracer stops. On the other hand, the image in Figure 25 contains no acute angle and so the line tracer succeeded in tracing, however, this does not mean the line trace attack works. In fact, it failed to make an correct response to the proposed CAPTCHA because it does never stop tracing until we forced to stop. To make a correct response, it is required to give a correct stroke order, but the line tracer did not. This experiment indicates another countermeasure against line trace attacks using symbols where the correct stroke order starts and stops at the exactly same position. Note that the symbol star enjoys such a property. A line tracer does not know where it should stop in tracing such symbols. It may be possible to tune up line trace programs to stop at the point where it is expected, however, we have to evaluate its costs to realize such a mechanism.

As we can expect the line tracer stops when it has traced only half of the whole character in Figure 26. This indicates that using characters consisting of more than one storokes is another effective countermeasure against line trace attacks.

The line tracer stops at the acute angle of the character in Figure 27.

In addition, we show some other experimental results. The figure 36 and 37 include a Japanese hiragana and the character “r”, respectively. The line tracer traces at some point then stops at the point from which the correct stroke returns. This kind of motion seems difficult for the line tracer by these results even though humans knows by the embodied knowledge but line tracer cannot deal with such stroke. and we make a hypothesis that a line tracer is not good at returning at a point of a vertex of an acute angle.

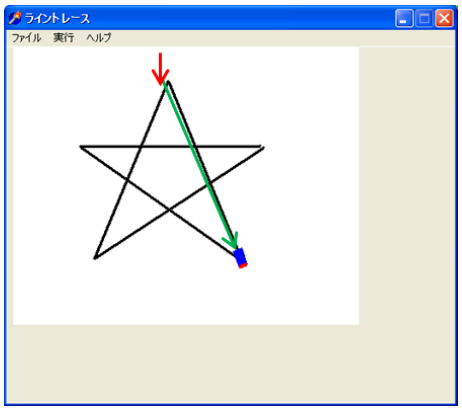

Figure 32: Star

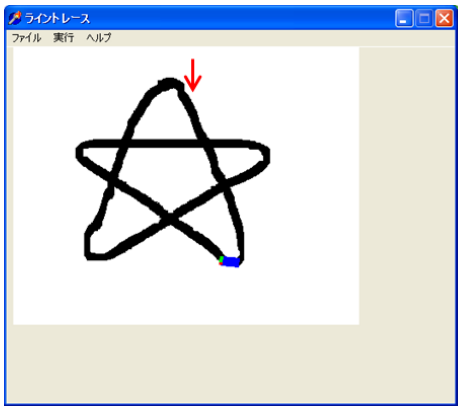

Figure 33: Hnadwritten Star

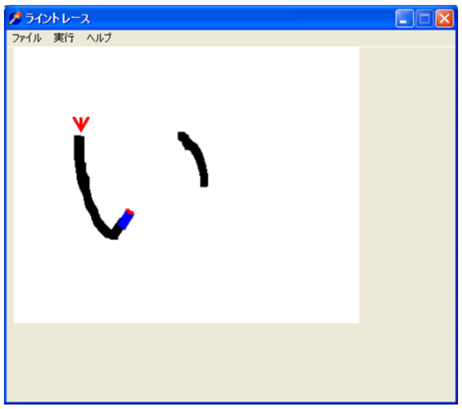

Figure 34: Japanese Hiragana

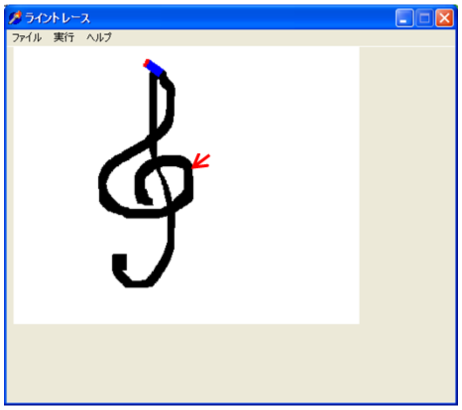

Figure 35: Handwritten G clef

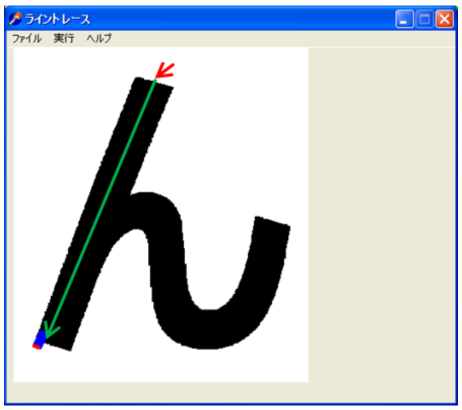

Figure 36: Japanese Hiragana

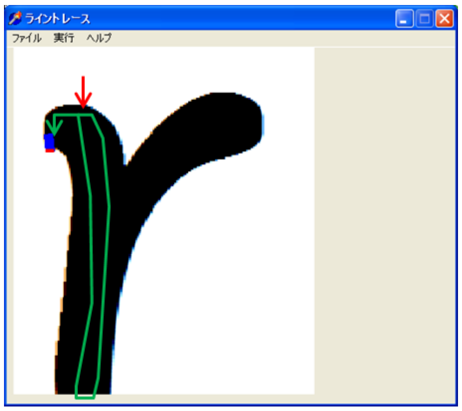

Figure 37: r

6 Conclusions

Following [1], we discussed the CAPTCHA technique utilizing touchscreens to relieve the inconvenience caused by the existing CAPTCHAs when using smartphones in this paper. We implemented the proposed technique and carry out experiments to examine the usefulness and compared with the existing techniques. In particular, we carry out more experiments in addition to the ones in [1]. We found that the proposed technique would be useful, however, there are several things which have not been examined in the paper. We list some of them as our future works.

First, automatic generation of challenge images, which is mandatory to make a realistic applications, has not been made yet, while we made challenge images and the authorization program by hand for our experiment. It is expected to make an automatic challenge image generator by randomly transforming one-stroke images. It is important to apply continuous transformations to a one-stroke character in order to challenge images still have correct order stroke features.

Second, there are several touchscreen methodologies as we briefly explained in Section 2. We used the electrostatic capacity method in our experiment and did not carry out experiments on the other platforms. It is expected that sensitivity of the touchscreen methodology affects the error rate of data inputted. It should be considered the other platforms to realize the proposed CAPTCHA and we leave it as one of future works.

Third, our experiment of a line tracer attack is based on a free software without any tuning-up. Thus, the line tracer tries to find black area within 3 bits ahead. The line trace program can be tuned up especially for tracing one-stroke characters. It is easy to make the sensing are wider. However, this may make the line tracer moves incorrectly when tracing one-stroke characters. If the sensing area of the line tracer is wider, then the defense method can be considered as follows. As an original of a challenge image, we choose some complicated image such as Figure 27, in which a line tracer with wider sensing area will find more than one ways to go next. We conjecture the probability that it makes a wrong decision is much bigger than the case of a normal line tracer without any tuning-up. It is a future work to examine a line tracer with wider sensing area and complicated one-stroke images.

Lastly, we should study recognition ability of humans and computer programs with respect to one-stroke characters and their correct stroke order. We mention the Pandemonium architecture in Section 5.2. We should study the relation between line tracer's recognition ability and the Pandemonium architecture. In particular we should study discontinuous characters. Note that the Pandemonium architecture deals with not only one-stroke letter but general images. In constructing a CAPTCHA, it is important to make challenge images different from the original letter but somehow humans can find the original. In this paper we suppose that human beings are shown only image which can be traced by one-stroke, that is, and we did not discuss humans' ability to compliment letters with several nicks. In our case, we did not try to consider challenge images of one-stroke character with several nicks, that is, discontinuous curves. We considered only a nicked image 20 and the Japanese hiragana 26 as discontinuous images and so more detailed study is necessary. It is discussed line tracer's ability to trace discontinuous images and its relation to the Pandemonium character in [16]. Furthermore we have not discussed influence of colors on recognition by line tracers. For example, it may be effective to use colored challenge images. We have not examine behavior of line tracers for colored images yet, however, it is predicted hard to program a line tracer to trace a colored images because it follows the RGB specified in advance. If the RGB gradually changes then a line tracer does not know its change. Unless a reader is color blind, he or she can recognize the character α in Figure 38 and a distorted α in Figure 39. It may be harder to recognize a distorted α in Figure 40. We believes that using colors is an effective countermeasure against line trace attacks and effects on humans recognition ability as a future works.

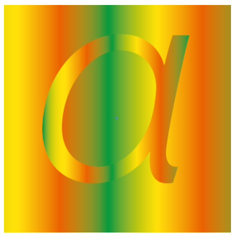
Figure 38: Color 1

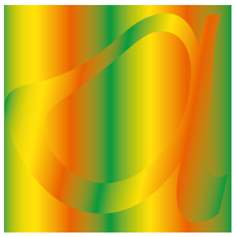
Figure 39: Color 2

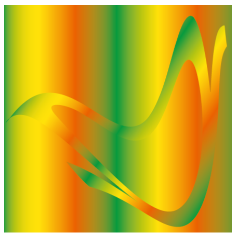
Figure 40: Color 3

References

[1] Y. Tsuruta, M. Takaya, and A. Yamamura, "CAPTCHA suitable for smartphones," in *Proc. of the 2013 International Conference on Information and Communication Technology (ICT-EurAsia'13), Yogyakarta, Indonesia, LNCS*, vol. 7804. Springer-Verlag, March 2013, pp. 131–140.

[2] A. M. Turing, "Computing Machinery and Intelligence," *Mind*, vol. 59, no. 236, pp. 433–460, 1950.

[3] L. V. Ahn, M. Blum, N. J. Hopper, and J. Langford, "CAPTCHA: using hard AI problems for security," in *Proc. of the 22nd International Conference on Theory and Applications of Cryptographic Techniques (EUROCRYPT'03), Warsaw, Poland, LNCS*, vol. 2656. Springer-Verlag, May 2003, pp. 294–311.

[4] L. V. Ahn, M. Blum, and J. Langford, "Telling humans and computers apart automatically," *Communications of the ACM*, vol. 47, no. 2, pp. 56–60, February 2004.

[5] R. Gossweiler, M. Kamvar, and S. Baluja, "What's up CAPTCHA?: a CAPTCHA based on image orientation," in *Proc. of the 18th International Conference on World Wide Web (WWW'09), Madrid, Spain*. ACM, April 2009, pp. 841–850.

[6] B. Shneiderman, "Touch Screens Now Offer Compelling Uses," *IEEE Software*, vol. 8, no. 2, pp. 93–94, March 1991.

[7] ——, *Designing the user interface (2nd ed.): strategies for effective human-computer interaction*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1992.

[8] E. A. Johnson, "Touch Display - A novel input/output device for computers," *Electronics Letters*, vol. 1, no. 8, pp. 219–220, 1965.

[9] ——, "Touch Displays: A Programmed Man-Machine Interface," *Ergonomics*, vol. 10, no. 2, pp. 271–277, 1967.

[10] A. Sears, C. Plaisant, and B. Shneiderman, "A new era for high precision touchscreens," in *Advances in human-computer interaction*, H. R. Hartson and D. Hix, Eds. Ablex Publishing Corp., 1992, vol. 3, pp. 1–33.

[11] T. L. Dimond, "Devices for reading handwritten characters," in *Proc. of Eastern Joint Computer Conference: Computers with deadlines to meet (IRE-ACM-AIEE '57), Washington, D.C., USA*. ACM, December 1957, pp. 232–237.

[12] G. Mori and J. Malik, "Recognizing objects in adversarial clutter: breaking a visual captcha," in *Proc. of the 2003 IEEE Conference on Computer Vision and Pattern Recognition (CVPR'03), Madison, Wisconsin*. IEEE, June 2003, pp. 134–141.

[13] O. G. Selfridge, "Pandemonium: a paradigm for learning in Mechanisation of Thought Processes," in *Proc. of a Symposium Held at the National Physical Laboratory, London*, November 1958, pp. 513–526.

[14] O. G. Selfridge and U. Neisser, "Pattern recognition by machine," in *Computers & thought*, E. A. Feigenbaum and J. Feldman, Eds. MIT Press, 1995, pp. 237–250.

[15] R. L. Klatzky, *Human Memory: Structures and Processes*, ser. A Series of books in psychology. W. H. Freeman, 1975.

[16] M. Takaya, H. Kato, T. Komatsubara, Y. Watanabe, and A. Yamamura, "Recognition of one-stroke symbols by humans and computers," *Procedia - Social and Behavioral Science (to appear)*, 2013.

Mayumi Takaya is an assistant professor of Dept. of Computer Science and Engineering, Akita University. She has been working for 15 years since 1977 on various database systems at Hitachi Software Eng. Ltd. She has been working for more 20 years since 1992 on database applications, numerical analysis for material properties and mobile computer applications at Akita University. Her research interests are cognitive science and digital museum.

Yusuke Tsuruta received his B.E. and M.E. degree from Akita University in 2011 and 2013. He is working at ITOCHU Techno-Solutions Company. His research interest is virtualization technologies that logically integrate system infrastructure.

Akihiro Yamamura received his master degree in Mathematics from Shimane University in 1988 and his PhD in Mathematics (combinatorial semigroup theory) from University of Nebraska-Lincoln in 1996. His research interests include public key cryptography, cryptographic protocols, authentication, algorithms of combinatorial structures, combinatorial semigroup and group theory. He is currently professor of department of computer science and engineering at Akita University.