J Inf Process Syst, Vol.10, No.1, pp.92~102, March 2014
http://dx.doi.org/10.3745/JIPS.2014.10.1.092
pISSN 1976-913X
eISSN 2092-805X

Gesture Input as an Out-of-band Channel

Oyuntungalag Chagnaadorj* and Jiro Tanaka*

Abstract—In recent years, there has been growing interest in secure pairing, which refers to the establishment of a secure communication channel between two mobile devices. There are a number of descriptions of the various types of out-of-band (OOB) channels, through which authentication data can be transferred under a user's control and involvement. However, none have become widely used due to their lack of adaptability to the variety of mobile devices.
In this paper, we introduce a new OOB channel, which uses accelerometer-based gesture input. The gesture-based OOB channel is suitable for all kinds of mobile devices, including input/output constraint devices, as the accelerometer is small and incurs only a small computational overhead. We implemented and evaluated the channel using an Apple iPhone handset. The results demonstrate that the channel is viable with completion times and error rates that are comparable with other OOB channels.

Keywords—Secure Device Pairing, Out-of-band Channel, Authentication, Gesture Input, Accelerometer

1. INTRODUCTION

Over the last few decades, there has been tremendous growth in the area of ubiquitous computing, and numerous portable devices that perform many complex operations and can be used in a wide variety of applications have been developed. Technologies including Wi-Fi, Bluetooth, and ZigBee, have been created to enable wireless communication between these devices. However, compared to wired connections, wireless networks are more vulnerable to security threats, particularly eavesdropping and alteration, also termed as man-in-the-middle (MitM) attack [1]. It is generally assumed that major security issues, including MitM, can be addressed if one's cryptographic public key can be authenticated. In a wired network, authentication is resolved using digital certification and a trusted third party, usually referred to as the certificate authority (CA). However, establishing a trusted third party among mobile devices is not practical, because wireless networks are usually set up on an *ad-hoc* basis involving unfamiliar devices.

User involvement and control in the authentication process bootstraps the problem. In this paradigm, an additional auxiliary channel, called the out-of-band (OOB) channel, exists between two mobile devices, as well as the ordinary wireless channel, as illustrated in

※ A previous version of this paper appeared at the Proceedings of Ubiquitous Information Technology and Applications Conference, CUTE, 2012.
※ We would like to thank the members of the Interactive Programming laboratory of the University of Tsukuba for their invaluable comments and feedback.
Manuscript received June 28, 2013; accepted August 15, 2013.
Corresponding Author: Oyuntungalag Chagnaadorj (jiro@cs.tsukuba.ac.jp)
* Dept. of Computer Science, University of Tsukuba, Tsukuba, Ibaraki, Japan
(chtungalag@iplab.cs.tsukuba.ac.jp, jiro@cs.tsukuba.ac.jp)

Figure 1. In this paper, 'the requester' refers to the device that is about to be authenticated and 'the verifier' refers to the device that does the authenticating. To authenticate the requester, the verifier receives both the cryptographic public key through the wireless channel and the hash of the same key (the authentication data) through the OOB channel. The verifier then generates another hash for the requester's public key and checks it against the received hash data. If cross authentication is required, the same process is repeated in the reverse direction. So far, a number of different OOB channels have been proposed. Many low-bandwidth OOB channels have utilized short authentication string (SAS) protocol [2], which reduces the required authentication data to 15 bits, while affording a reasonable level of security. The SAS protocol can also authenticate the public keys of both devices using a one-directional OOB channel.

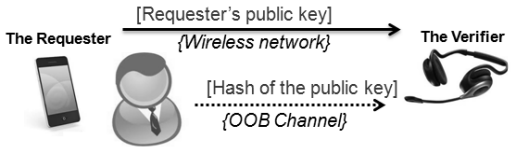


Fig. 1. OOB channel authentication

In this paper, a gesture-based OOB channel is introduced. Using this channel, the requester converts the authentication data into a sequence of gestures and informs the user of them. The user then performs the gestures one by one with a verifier that has an embedded accelerometer. We conducted two experiments. First, the usability of the gesture-based OOB channel was evaluated to compare our channel with previously studied approaches. Our channel had reliable performance with a mean completion time of 16.86 s and a standard deviation of $\sigma = 4.2$ s, as well as a mean error rate of 0.13 per transfer with $\sigma = 0.34$. Four different gesture sets were used in this experiment to determine which library size would be more usable in practice. The second experiment was conducted to determine whether or not user-oriented gesture templates improved the accuracy of gesture recognition. Surprisingly, in terms of the error rate, there was no significant difference between the pre-installed and user-defined templates.

A common drawback of the OOB channels was the inspiration for our work. Even though a number of OOB channels have been proposed, none have been generally accepted, and will not be in the near future because of the lack of their adaptability with the wide range of mobile devices that exist. The gesture-based OOB channel is suitable for all mobile devices, including input/output constraint devices, because modern accelerometers are very small and lightweight For example, the accelerometer inside the iPhone 4S is 3×3 mm and weighs 30 mg. In fact, accelerometers are already embedded in some commercial products, such as smart phones. In addition, gesture recognition requires a relatively small computational overhead compared to vision-based approaches. Furthermore, the portability is the only common feature of mobile devices, and so our OOB channel has the added advantage of utilizing this commonality. Moreover, gestures represent one of the most promising interactive interfaces in the ubiquitous environment [3]; they are natural and intuitive for humans, and so gesture-based operations are easy to implement and require little effort.

The remainder of this paper is organized as follows: section 2 briefly introduces related

work; section 3 explains the main elements of the gesture-based OOB channel; section 4 describes the user experiments and the results; section 5 provides a discussion; and section 6 concludes the paper.

2. RELATED WORK

2.1 Out-of-band Channels

A significant number of OOB channels have been proposed over the last decade. The principle limitation of these approaches, however, is that each channel is limited to one particular situation and is not intended to be used for all devices.

McCune et al. [4] proposed a display camera-based OOB channel, called Seeing-Is-Believing. In this system, the requester encodes the authentication data into a two-dimensional barcode and shows it using the screen on the requester device. The verifier reads the barcode using a photo camera. A similar but improved approach was taken by Saxena et al. [5], whereby the requester uses a light-emitting diode (LED) light and the verifier receives the signal using an optical detector.

Goodrich et al. [6] developed a speaker display-based OOB channel. Authentication data are converted into text; one device speaks it out and the other shows the text on its screen. The user compares the texts to complete the authentication. Similar methods were proposed by Prasad et al. [7]. The authenticating data arrive in the forms of “beeps” or “blinks” (pulses of light) from two devices, and the user ensures their correctness.

Soriente et al. [8, 9] proposed two different OOB channels. In one of them, when the requester transmits the authentication data in the form of either “beeps” or “blinks” from an LED light, the user presses the button synchronously on the verifier side. Their second OOB channel is based on a pure speaker–microphone interaction. The speakers on both sides send all necessary information, including authentication, data in the forms, of audio signals and the microphones on both sides receive the data.

Comparative studies of OOB channels in terms of both usability and security were independently conducted by Kumar et al. [10] and Kainda et al. [11]. Interesting studies have recently been reported by Chong et al. [12] and Ion et al. [13]. The major findings of these papers reveal that there is no dominant device-pairing method in real life. Instead, people prefer different methods, and tend to select different OOB channels depending on the situation.

Another secure-pairing approach is to generate the same shared key in both devices without transferring any data through secure or insecure channels. Up to recently, few reports in this area exist. Mayrhover et al. [14] proposed a system whereby two mobile devices are held and shaken together to create a shared secret key. In this case, both devices must have embedded accelerometers, and the acceleration data are converted into the same key. Lin et al. [15] demonstrated that tapping the same rhythm on both pairing devices can create a shared secret key. Any tappable input device, from a simple button to various sensors, can utilize this method. However, these methods may be vulnerable to a shoulder-surfing attack.

2.2. Accelerometer-based Gesture Input

Our method is closely related to gesture input involving mobile devices with accelerometers. Because many mobile devices are too small to be equipped with conventional input peripherals such as a keyboard, gestures hold promise as an alternative input method for such devices. Jones et al. [16] developed a tilt gesture-based text entry system for mobile devices, and achieved a mean error rate of 0.09 errors per character. Choi et al. [17] were able to recognize nine digits and five symbols that were "written in the air" as an input to a phone handset. Similar work was also carried out by Agrawal et al. [18]. When drawn in the air with the smart phone, an image was generated from the acceleration data.

Gesture inputs have also been used for security purposes. Patel et al. [19] developed a system in which a shake gesture enables to access public terminals. Chong et al. [20] proposed a system that uses accelerometer-based gestures to input a password to access the mobile device.

3. GESTURE-BASED OUT-OF-BAND CHANNEL

We propose an out-of-band channel that transfers authentication data through gestures. In this channel, the requester informs the user of the gestures. The user then performs the gestures one-by-one with the verifier using a built-in accelerometer. If both devices have an accelerometer, as well as a means to inform the gestures, a two-directional authentication is possible. However, to simplify the overall authentication process, we suggest a SAS-based one-directional, mutual authentication protocol [2] [7]. The SAS protocol reduces the length of authentication data to 15 bits, making it appropriate for a number of OOB channels.

Authentication in an SAS-based OOB channel follows six steps, as shown in Figure 2. Steps 1, 5, and 6 are the same in all OOB channels, whereas steps 2–4 vary depending on the channel. Each step is described in detail below.

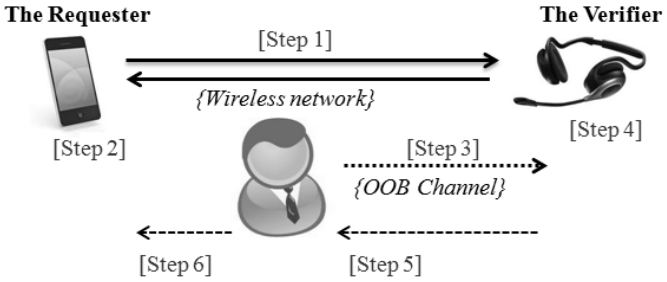


Fig. 2. Authentication process in SAS-based OOB channel

Step 1. Public keys and other necessary data for the SAS protocol are exchanged via a regular wireless channel.

Step 2. The requester computes the SAS data, converts them into gestures, and informs the user of the gestures. Any means can be used so long as it tells the user which

gestures he or she must perform. For example, if the requester device has a graphical display, it can show the gestures on the screen. If the requester can report only numbers, an identification number (ID) can be used for the gestures. In this case, a hardcopy or web-based manual is needed for the user to look up the corresponding gesture images.

Step 3. The user performs the gestures with the verifier that has a built-in accelerometer.

Step 4. The verifier recognizes the gestures, converts them into SAS authentication data, and compares the data to the original SAS data.

Step 5. The verifier informs the user of the results of this comparison. The result is either yes or no, where any simple output, such as the blink of an LED light or a beep, is sufficient.

Step 6. The user informs the sender of the result.

3.1 Implementation

A number of accelerometer-based gesture recognition methods, including the hidden Markov method, neural networks, FDSVM, and dynamic time wrapping (DTW), have been proposed. We used the DTW method because of its comparatively high accuracy, fast computational performance, and the need for relatively few templates [21, 22].

Because DTW is a template-based method, gesture recognition selects the best-matched template as a candidate. The DTW algorithm used in this study was first implemented by Liu et al. [23]. We quantize acceleration data of both the gestures and templates with a window size of 50 mps and a step of 30 mps. Unlike uWave in [23], we do not convert acceleration data into discrete numbers. However, in the present study, only half of the acceleration data were used for gesture recognition to reduce the completion time.

4. USER STUDY

4.1 Experimental Configuration

We implemented the gesture-based OOB channel using a desktop PC as the requester and an iPhone 4S (with a dual-core 1 GHz Cortex-A9 CPU with 515 MB of RAM and an STMicro 3-axis accelerometer with a 100 Hz output data rate) as the verifier. The PC generated a random 15-bit number, converted it into a sequence of gestures, and displayed both the number and the gesture set. Following this, the user participant carried out the gestures, one by one, using the phone. Once the gestures were performed, the phone converted them into a number, and displayed the number on the screen of the handset. If the original number on the PC screen and the number on the phone matched, the number had been successfully transferred.

A button was pressed to distinguish the gestures from other unwanted movements; that is, the participants pressed the button before starting and after ending the gestures. An alternative is that the users simply shake, tilt, or tap their mobile device instead of pressing the button. Because the user has to carry out a sequence of multiple gestures for one input, we chose gestures that ended in the same place where they started. Table 1 shows selected gestures and their corresponding IDs.

Table 1. Gesture library. The start of each gesture is marked with a black dot and the end is marked with an arrow

ID	Gesture	ID	Gesture	ID	Gesture	ID	Gesture	ID	Gesture
0		2		4		6		8	
1		3		5		7		9	

4.2 Gesture Encoding

To obtain a relationship between the size of the gesture library and the performance of the channel, four different interfaces are proposed: Digit10, Bit8, Depth6, and Bit4. The gesture library of the Bit4 interface consists of the first four gestures in Table 1, while Depth6, Bit8, and Digit10 consist of six, eight, and ten gestures, respectively.

Table 2. Interface summary

Interface	Gesture Library	Gestures per Input	Gesture Convert (Example: 13595_{10})	
Bit4	0, 1, 2, 3	8	03110123_4	
Depth6	Bit4+ 4, 5	6-10	32433_8	
Bit8	Depth6+ 6, 7	5	32433_8	
Digit10	Bit8+ 8, 9	5	13595_{10}	

To convert 15-bit SAS authentication data into a sequence of gestures, each interface utilized a different encoding method based on the library size, as shown in Table 2. Digit10 changes each digit, or ID in our case, with a corresponding gesture directly. In Bit8, the number is converted into an octal number and each octal digit is changed with a corresponding gesture. In Depth6, the number is also converted into an octal number. Eight octal digits are divided into two levels as shown in Figure 3, where four gestures represent digits in each level and the remaining two gestures are used to move between levels. Bit4 applies the same encoding as Bit8, except the number is converted into a quaternary number.

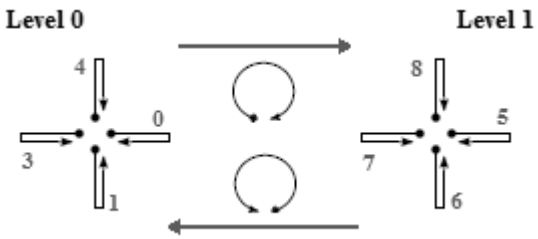


Fig. 3. Depth6 conversion

4.3 Experiment One

In this experiment, we investigated the usability of the gesture-based OOB channel, specifically the completion time, error rate, and learnability. Two hypotheses were formulated. First, there is no difference between the interfaces in terms of completion time and error rates. Second, there is no difference between the rounds in terms of the completion time and error rates.

Ten volunteers (three females and seven males) participated in our experiment; all of them were either undergraduate or graduate students, and all were right-handed. Before beginning the experiment, a brief introduction was given on how to hold the phone and what kind of gestures were to be used. The participants were allowed to practice one time. Every participant performed the gesture inputs 12 times with three rounds for each interface. In each round, the order of the interfaces was randomly generated and queued to reduce bias for certain interfaces. The experiment lasted for approximately 20 minutes. The total input time was measured from the beginning of the first gesture to the completion of the final gesture.

With the 10 participants, a total of 120 gesture inputs were entered during the experiment (four interfaces and three rounds). To analyze the collected data, repeated measures analysis of variance (ANOVA) was used. If the results were statistically significant, we further analyzed the data using a paired t-test between every two interfaces to locate the differences.

Completion Time: Table 3 summarizes the input completion time of the four interfaces. Repeated measures ANOVA with interface type as the factor was significant with $F(3, 29) = 45.95$ and $p = 0.0000$. Therefore, we conducted paired t-tests and the results suggested that the completion time between interfaces differed significantly ($t > 2$, $p < 0.05$) for most pairs, except Bit8 and Digit10 combinations ($t = 0.012$, $p = 0.904$). The correlation between the completion time and the size of the gesture library was $r = -0.812$.

Table 3. Completion time(s)

Interface	Mean	σ	Min	Max
Bit4	22.90	5.69	11.80	36.20
Depth6	25.02	7.11	11.60	38.80
Bit8	16.86	4.23	8.30	25.30
Digit10	16.76	4.79	8.10	28.20

Fig. 4. Time distribution

Figure 4 shows the distribution of the mean times for performing the gestures, gaps between the gestures, and the gesture recognition time of each interface. The Bit4 and Depth6 interfaces required more time for user input but less time for gesture recognition compared to the Bit8 and Digit10 interfaces.

Error rate: The error rate of the four interfaces is summarized in Figure 5. The Digit10 interface exhibited the greatest error rate, with a mean error per input of 0.33 and $\sigma = 0.48$, with 0.06 errors per gesture, whereas Bit4 had the fewest, with a mean error per gesture of 0.06 and $\sigma = 0.25$, with 0.008 errors per gesture. The mean error rates of Depth6 and Bit8

were similar, at 0.1 and 0.13 errors per input and 0.014 and 0.019 per gesture, respectively. Repeated measures ANOVA tests with interface as a factor were significant with $F(3, 29) = 4.43$ and $p = 0.006$. We conducted paired t-tests between the error rates of the different interfaces. There was no significant difference between Bit4, Depth8, and Bit8 ($p>0.5$). However, Digit10 differed significantly from the other interfaces, with $t < 2$ and $p < 0.05$. The correlation between the error rate and the size of the gesture library was quite high with $r = 0.897$.

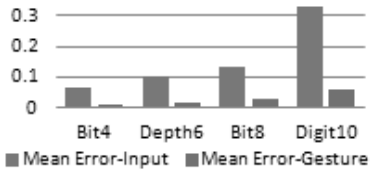


Fig. 5. Mean error rates

Table 4. Round completion time(s)

Round	Mean	σ	Min	Max
First	21.7	11.7	10.3	39.8
Second	20.9	6.7	8.1	38.4
Third	18.8	7.2	8.3	32.5

Learnability: The participants performed three rounds of tasks. Table 4 summarizes the speed of gesture input for each round. Repeated measures ANOVA tests with round as the factor showed that there was a significant difference between the rounds with $F(2, 38) = 8.48$ and $p = 0.004$. However, there were no significant differences in terms of error rates between the rounds.

4.4 Experiment Two

In the second experiment, we investigated whether or not the error rate decreased if users inserted their own gesture templates into the mobile device. It is quite possible that our gesture-based OOB can be extended so that users can carry out the required gestures with the verifier device before they start the authentication. In the previous experiment, the Bit8 interface showed the best results, so we selected Bit8 as the base interface for the second experiment. Only two interfaces were implemented: one used preinstalled templates while the other used user-defined templates when identifying the gestures.

A different set of 10 volunteers participated in the second experiment (two males and eight females, all of whom were right-handed). Prior to commencing the experiment, each participant inserted the first eight gestures shown in Table 1 into the phone by performing them to create their own templates. Once all eight templates were completed, the participant carried out the gesture input six times (three rounds for each interface).

We assessed whether there would be a difference between the preinstalled templates and the user-defined templates in terms of the error rate. From the 10 participants, we collected a total of 60 gesture inputs (three rounds and two interfaces). Because we only had two groups of data to compare, we used a paired t-test instead of repeated measures ANOVA. However, no significant difference was found between the two interfaces ($t(29) = 1.79$, $p = 0.083$).

5. DISCUSSION

It appears that the Bit8 interface is the best choice for our gesture-based OOB channel, considering the higher negative correlation between completion time and the size of the gesture library and the similar error rate between Bit4, Depth6, and Bit8 interfaces. Bit8's transmission speed of 17 s is faster than some of the popular OOB channels, such as Barcode (37 s), Alphanumeric (40 s), BEDA (45 s), Beep&Blink combinations (30 s), and Loud&Clear (20 s). The Bit8 interface has a mean error rate of 13%, which is lower than that of Beep&Blink combinations (about 20–30%), Barcode (53%), Alphanumeric Copy&Enter (23%), many versions of Compare&Confirm (16–36%), Alphanumeric Compare&Select (30%), and Numeric Copy&Enter (13%) [10, 11].

In our experiments, there was no difference between Digit10 and Bit8, and only a small difference between Bit4 and Depth6 in terms of completion times. In other words, the number of gestures per input had a greater influence on the completion time than did the size of the gesture library, because gesture recognition took less time than the user's actions.

The error increased dramatically when using Depth10 (as shown in Fig. 5). The size of the gesture library may be one factor ($r = 0.897$). Our log file revealed that approximately 80% of the Depth10 errors were related to triangle gestures. It follows that the selection of proper gestures is very important.

As shown in Table 4, the completion time decreased over trials in a short period of time while the accuracy remained the same. Moreover, participants became more confident in the execution of these tasks over the course of the experiment.

One unexpected finding is that the user-defined templates did not improve the accuracy of the channel. However, these results should be interpreted with caution as the sample size was relatively small.

Each participant was asked to complete a brief questionnaire. Following the experiment, they were not able to clearly distinguish one interface from another. Therefore, the questions were on the overall impressions of the gesture input. Almost two-thirds of the participants had a positive attitude with respect to the ease of the method, and 70% were satisfied with the time required.

During the experiments, we made a number of important observations that may help to improve our OOB channel. First, the speed of the users varied, which noticeably affected the success of gesture input. Therefore, the gesture-based OOB channel should be able to adapt to user speed by providing feedback. Secondly, the sequence of gestures was carried out in a discrete manner; however, participants preferred continuous gestures. In other words, pressing a button before and after every gesture placed an additional burden on the user. Finally, the participants indicated that they did not like holding the phone in a fixed position.

6. CONCLUSION

We have demonstrated an out-of-band channel for secure device pairing whereby the user performs gestures using a mobile device with a built-in accelerometer to transmit authentication data. User tests were conducted to assess the viability of the channel. Overall, the results demonstrate that the channel is practical, while the performance metrics were similar to those of other OOB channels.

We suggest that a library of eight gestures is appropriate for a gesture-based OOB channel, based on the higher transmission speed of 17 s and the comparatively low error rate of 13% demonstrated in our tests.

REFERENCES

[1] F. Stajano and R. Anderson, "*The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks*", *Proceedings of the Security Protocols Workshop*, UK, April, 1999.

[2] S. Vaudenay, "*Secure Communications over Insecure Channels Based on Short Authenticated Strings*", *Proceedings of the International Cryptology Conference*, USA, April, 2005.

[3] J. Canny, "The Future of Human-Computer Interaction", *Queue-HCI*, vol. 4, no. 6, 2006.

[4] J.M. McCune and A. Perrig,"Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication", *International Journal of Security and Networks*, vol. 4, no. 1/2, 2009.

[5] N. Saxena and M.B. Uddin, "*Automated Device Pairing for Asymmetric Pairing Scenarios*", *Proceedings of the Information and Communication Security Conference*, UK, October, 2008.

[6] M.T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik and E. Uzun, "*Loud and Clear: Human-Verifiable Authentication Based on Audio*", *Proceedings of the Distributed Computing Systems Conference*, Portugal, July, 2006.

[7] R. Prasad and N. Saxena, "*Efficient Device Pairing Using Human-Comparable Synchronized Audiovisual Patterns*", *Proceedings of the Applied Cryptography and Network Security Conference*, USA, June, 2008.

[8] C. Soriente, G. Tsudik and E. Uzun, "*BEDA: Button-Enabled Device Association*", *Proceedings of the Security for Spontaneous Interaction Workshop*, 2007.

[9] C. Soriente, G. Tsudik and E. Uzun,"*HAPADEP: Human-Assisted Pure Audio Device Pairing*", *Proceedings of the Information Security Conference*, 2008.

[10] A. Kumar, N. Saxena, G. Tsudik, and E. Usun, "A Comparative Study of Secure Device Pairing Methods", *Pervasive and Mobile Computing*, vol. 5, no. 6, 2009.

[11] R. Kainda, I. Flechais and A.W. Roscoe, "*Usability and Security of Out-Of-Band Channels in Secure Device Pairing Protocols*", *Proceedings of the Usable Privacy and Security Symposium*, USA, July, 2009.

[12] M.K. Chong and H. Gellersen, "*How Users Associate Wireless Devices*", *Proceedings of the Computer Human Interactions Conference*, Canada, May, 2011.

[13] I. Ion, M. Langheinrich, P. Kumaraguru and S. Capkun, "*Influence of User Perception, Security Needs, and Social Factors on Device Pairing Method Choices*", *Proceedings of the Usable Privacy and Security Symposium*, USA, July, 2010.

[14] R. Mayrhofer and H. Gellersen, "*Shake Well Before Use: Authentication Based on Accelerometer Data*", *Proceedings of the Pervasive Computing Conference*, Canada, May, 2007.

[15] F.X. Lin, D. Ashbrook and S. White, "*RhythmLink: Securely Pairing I/O-Constrained Devices by Tapping*", *Proceedings of the User Interface Software and Technology Symposium*, USA, October, 2011.

[16] E. Jones, J. Alexander, A. Andreou, P. Irani and S. Subramanian, "*GesText: Accelerometer-Based Gestural Text-Entry Systems*", *Proceedings of the Human Factors in Computing Systems Conference*, USA, April, 2010.

[17] E. Choi, W. Bang, S. Cho, T. Yang, D. Kim and S.Kim, "*Beatbox: Music Phone: Gesture-based Interactive Mobile Phone Using Tri-axis Accelerometer*", *Proceedings of the Industrial Technology Conference*, Hong-Kong, December, 2006.

[18] S. Agrawal, I. Constandache, S. Gaonkar and R.R. Choudhury, "*PhonePoint Pen: Using Mobile Phones to Write in Air*", *Proceedings of the Networking, Systems, and Applications for Mobile Handhelds Workshop*, Spain, August, 2009.

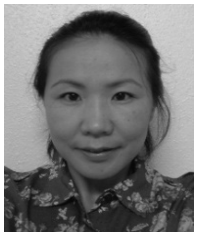
[19] S.N. Patel, J.S. Pierce and G.D. Abowd, "*A Gesture-based Authentication Scheme for Untrusted Public Terminals*", *Proceedings of the User Interface Software and Technology Symposium*, USA, October, 2004.

[20] M.K. Chong and G.Marsden, "*Exploring the Use of Discrete Gestures for Authentication*", *Proceedings of the Human Computer Interaction Conference*, Sweden, August, 2009.

[21] G. Niezen, The optimization of gesture recognition techniques for resource-constrained devices [dissertation]. University of Pretoria, South Africa, 2008.

[22] J. Wu, G. Pan, D. Zhang, G. Qi and S. Li, "*Gesture Recognition with a 3-D Accelerometer*", *Proceedings of the Ubiquitous Intelligence and Computing Conference*, Australia, July, 2009.

[23] J. Liu, Z. Wang, L. Zhong,J. Wickramasuriya and V. Vasudevan, "*uWave: Accelerometer-based Gesture Recognition and Its Applications*", *Proceedings of the Pervasive Computing and Communications Conference*, USA, March, 2009.



Oyuntungalag Chagnaadorj

She is a PhD candidate in computer science at University of Tsukuba, Japan. Her research interests include secure pairing of mobile devices and ubiquitous computing. She received a BS in computer science at National University of Mongolia, Mongolia in 1998 and a MS in software engineering at Ritsumeikan University, Japan in 2004.

Jiro Tanaka

He is a Professor of Department of Computer Science, University of Tsukuba. His research interests include ubiquitous computing, interactive programming, and computer-human interaction. He received a BSc and a MSc from University of Tokyo in 1975 and 1977. He received a PhD in computer science from University of Utah in 1984. He is a member of ACM, IEEE and IPSJ.