MimicGesture: Secure Device Pairing with Accelerometer-Based Gesture Input

Oyuntungalag Chagnaadorj and Jiro Tanaka

Graduate School of Systems and Information Engineering
University of Tsukuba
Tennodai 1-1-1, Tsukuba City, Ibaraki Prefecture, 305-8573 Japan
chtungalag@iplab.cs.tsukuba.ac.jp, jiro@cs.tsukuba.ac.jp

Abstract. It is unfeasible to establish a trusted-third party in ad-hoc wireless networks. Transferring authentication data under a human user's control and involvement, called Out-Of-Band (OOB) channel, bootstraps device pairing. A number of OOB channels have been proposed. However, none of them is generally accepted due to their lack of adaptability to various devices.

In this paper, we introduce a new OOB channel, called MimicGesture, which uses accelerometer-based gesture input. We argue that our OOB channel is suitable for all kinds of mobile devices, including I/O constrained devices, as the accelerometer is small in size and requires low computational overhead. We conducted a usability test in order to check the viability of our channel.

Keywords: Device Pairing, Mobile Device, Security, Authentication, Out-of-band Channel, Gesture Input, Accelerometer.

1 Introduction

Personal computers are becoming ubiquitous and, as a result, mobile devices are now an inevitable part of our lives. Various technologies such as Wi-Fi, Bluetooth, ZigBee etc exist to support wireless communication between mobile devices. However, compared to their wired counterparts, wireless networks are more vulnerable to security threads, especially to eavesdropping and alteration, so-called Man-in-the-Middle (MiTM) attack [13].

It is generally assumed that major security issues including MiTM can be addressed if cryptographic functions are secure and one's cryptographic public key can be authenticated. In wired network, authentication is basically solved with components of PKI (Public Key Cryptography); certificate and trusted-third party, usually called Certificate Authority. However, establishing a trusted-third party among mobile devices is not practical, since wireless networks are usually set on a completely ad-hoc basis, involving unfamiliar mobile devices.

One possible solution to authentication of mobile devices is to use a human user's involvement and control in the authentication process. In this paradigm, an auxiliary extra channel, called Out-Of-Band (OOB), exists between two mobile devices, in addition to ordinary wireless channel. Throughout this paper, the device that is about to be authenticated is called ‘the sender’ and the device that is authenticating is called ‘the verifier’. In order to authenticate the sender, the verifier receives both the cryptographic public key through wireless channel and the hash of the same key (authentication data) via OOB channel. Then, the verifier generates hash of the sender’s public key and checks it against the received hash data. Therefore, an OOB channel must be developed so that the human user can supervise the transmission process to ensure integrity of the authentication data. Various OOB channels have been proposed.

In this paper, a new gesture-based OOB channel, MimicGesture, is introduced. In our proposed channel, the sender converts the authentication data into a sequence of gestures and informs the human user about them. Then the user performs gestures one by one with the verifier, which must be equipped with the embedded accelerometer.

We conducted a usability test on MimicGesture to compare it with previously proposed approaches. The result shows that our OOB channel has a reliable performance rate with the mean completion time of 16.86sec (SD=4.2) and the mean error rate of 0.13 (SD=0.34) per transfer. At the same time, we performed a user evaluation of four different gesture sets to see which library size might be more usable in practice.

1.1 Motivation

Several reasons exist, which have driven us to use gesture-based OOB channel. First of all, MimicGesture is appropriate for all devices including I/O constraint devices. Modern accelerometers are tiny in size and lightweight (for example, iPhone 4s accelerometer: 3x3mm and 30mg), so they can be embedded in most mobile devices without affecting the overall functionality. In fact, accelerometers are already inside some commercial products, such as smart phones.

Second, each of the existing OOB channels is for one specific situation only, because there is a wide-variety of mobile devices that have no common features except mobility. Thus, MimicGesture has the potential to be generally accepted since it utilizes the only common aspect in all mobile devices: the ability to move.

Finally, gestures represent one of the promising interactive interfaces in the ubiquitous environment [1]. They are natural and intuitive for humans; thus, gesture-based OOB channel is easy to carry out and requires less effort.

2 Related Work

A significant amount of OOB channels have been introduced in the last decade. McCune et al. proposed a display-camera based OOB channel, Seeing-Is-Believing [7]. The sender encodes the authentication data into a two-dimensional barcode and shows it on its screen. The verifier reads the barcode using a photo camera. A similar but improved approach is taken by Saxena et al [10]. In their work, the sender uses a LED light while the verifier gets it using a light detector.

Goodrich et al. developed a speaker-display based OOB channel [2]. Authenticating data is converted into text; one device speaks it and another device shows the text on its screen. The human user compares the texts to complete authentication. Similar methods are proposed by Prasad et al. [9]. The authenticating data comes up in the form of “beeping” or “blinking” from two devices and the human user ensures their correctness.

Soriente et al. contributed two different OOB channels [11] [12]. In their first channel, as the sender transmits the authentication data in the form of either beep sound or LED light, the human user presses the button synchronously on the verifier’s side. Their second approach is based on a pure speaker-microphone. The speakers on both sides send all necessary information and authenticating data in the form of audio and the microphones on both sides receive what the other side sent.

Comparative studies of OOB channels in terms of both usability and security matters are investigated independently by Kumar et al. [4] and Kainda et al. [5]. Recently, an interesting study has been conducted by Ion et al. [3]. A major finding in this paper is that people tend to select different OOB channels depending on the situations.

3 MimicGesture OOB Channel

We proposed MimicGesture OOB channel, which transfers the authentication data through gestures. The sender informs a human user about the gestures. Then, the user performs gestures one by one with the verifier, which has a built-in accelerometer. If both devices have accelerometer as well as a means to inform gestures, two-directional authentication is possible. In order to simplify the authentication procedure, we suggest an SAS-based one directional, yet mutual authentication protocol [14] [10]. This reduces the length of authentication data down to 15 bits, which is preferable aspect to our channel. The overall authentication process in an SAS-based OOB channel follows six steps (Fig. 1). MimicGesture consists of step 2, 3, and 4.

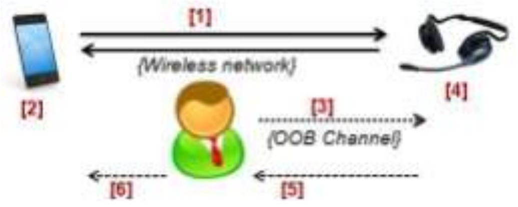


Fig. 1. Authentication Process in SAS-based OOB Channel

1. Public keys and other necessary data for the SAS protocol are exchanged via regular wireless channel.
2. The sender computes SAS data, converts it to gestures, and informs the user about the gestures. Any means can be used as long as it is able to tell the user what gestures he needs to perform. If the sender possesses a graphical display, it can show gestures on its screen. If the sender only has ability to show or tell numbers, it can communicate identification numbers (ID) of gestures. In this case, manual (in paper form or as a web page) is needed for the user to look up the corresponding gesture images.

3. The user performs the gestures with the verifier, which has an accelerometer.
4. The verifier recognizes gestures, converts gestures to SAS authentication data, and compares it with its own created SAS data.
5. The verifier informs the user of the result. The result is either YES or NO; thus an output as simple as a LED light or beep sound is sufficient.
6. The user informs the sender of the result.

3.1 Implementation

A number of accelerometer-based gesture recognition methods, for instance HMM, Neural Network, FDSVM, and DTW have already been proposed. MimicGesture uses Dynamic Time Wrapping (DTW) due to its comparatively high accuracy with fast computational performance and fewer template requirements [8] [15].

DTW is a template based method, so gesture recognition selects the best matched template as a candidate. By definition, device pairing assumes there is no prior contact between the two devices. Therefore, template adaptation is not necessary in MimicGesture and preinstalled templates are used for the gesture recognition.

The DTW algorithm used in this study was implemented by Liu et al. [6]. We also quantize acceleration data of both gestures and templates window size of 50mps with step of 30mps. Unlike uWave, we do not convert acceleration data to discrete numbers. However, two additional changes are applied to optimize the performance. First, we observed that a slower performance affects the error rate more than a higher one. Therefore, two templates for each gesture are used: one is with normal speed, the other is very slow. Second, the exact half of the acceleration data is used for gesture recognition in order to reduce the completion time.

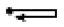
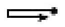
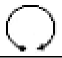
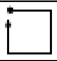
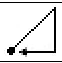
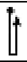
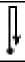
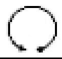
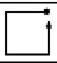
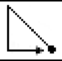
4 Usability Analysis

We tested MimicGesture OOB channel with a desktop PC as a sender and an iPhone 4S (Dual-core 1GHz Cortex-A9 CPU with 515 MB RAM and STMicro 3-axis accelerometer with 100Hz output data) as a verifier. The PC generates a random 15 bit number, converts it to a sequence of gestures, and displays both the number and gesture set. Then, the participant performs the gestures, one by one with the phone. After performing all the gestures, the phone converts the gestures to the original number, and displays the created number on its screen. If the two numbers are the exactly same, it is considered that the number has been successfully transferred.

We placed a button on the interface in order to distinguish gestures from other unwanted movements. Participants push the button both before starting and after ending gestures. Although not implemented in our system, the users could simply either shake, tilt, or tap their mobile device instead of pressing a button.

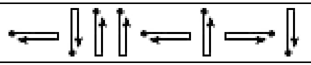
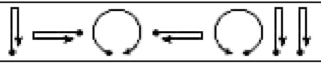
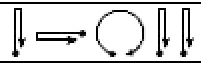
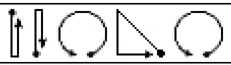
Because the user has to carry out a sequence of gestures (one input= several gestures), we chose gestures that end in the same place they started. Table 1 shows selected gestures and their assigned IDs. The start of gestures is marked with a black dot and the end is marked with an arrow.

Table 1. Gesture Library

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

To obtain the relationship between the size of gesture library and the performance of the channel, four different interfaces are proposed: Digit10, Bit8, Depth6, and Bit4. Each interface utilizes a different encoding method due to their library size (Table 2). In order to convert 15 bit number to gestures, Digit10 changes each digit with corresponding gesture. In Bit8, the number is converted into octal number and then, each octal digit is changed with corresponding gesture. In Depth6, the number is converted into octal number as well. Eight octal digits are divided into two levels. Four gestures represent digits in each level and the remaining two gestures are used to move between levels. Bit4 uses the same encoding as Bit8, except the number is converted into base-four number (Table 2).

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}	

Ten volunteers (three females) participated in our experiment; all of them are university students, both undergraduate and graduate. Participants were asked to perform gesture inputs 12 times (three rounds for each interface).

4.1 Results

Two hypotheses are formulated. First, there is no difference among interfaces in terms of completion time and error rates. Second, there is no difference among rounds in terms of completion time and error rates.

Completion Time.

Table 3 shows the summary of the input completion time of the 4 interfaces. A total of 120 gesture inputs (10participants x 4interfaces x 3times) were entered during the experiment. Input completion time is analyzed with repeated measures ANOVA with interface type as the factor. The result was significant for the within interfaces with F (3, 29) =45.95 (p=0.0000). Therefore, we conducted paired t-test between the completion times of different interfaces. The result suggests that the completion time between interfaces differs significantly (p<0.05) for the most pairs, except Bit8 and Digit10 combination (p=0.904).

Table 3. Completion time (seconds)

Interface	Mean	SD	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

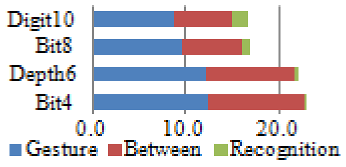


Fig. 2. Time distribution (second)

Figure 2 shows the mean time distribution for performing gestures, gaps between gestures, and gesture recognition of each interfaces. Bit8 and Depth6 interfaces spent more time on the human user's action but less on the gesture recognition compared to Bit8 and Digit10 interfaces.

Error rate.

The error rate of the four interfaces is summarized in Figure 3. The Digit10 has the greatest errors (mean 0.33 errors per input with SD=0.48, 0.06 errors per gesture) whereas Bit4 has the fewest (mean 0.06 errors per input with SD=0.25, 0.008 errors per gesture). The mean error rates of Depth6 and Bit8 were close to each other (0.1 and 0.13 errors per input respectively, 0.014 and 0.019 per gestures respectively).

The result of the repeated measures ANOVA with interface as factor was significant with F (3, 29) =4.43 (p=0.006). We conducted a paired t-test between the error rates of different interfaces. There is no significant difference among Bit4, Depth8, and Bit8 (p>0.5). However, Digit10 differs significantly from each of the other three interfaces with t< -2 (p<0.05). The correlation between the error rate and the size of the gesture library was r=0.897.

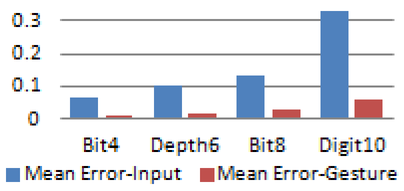


Fig. 3. Mean error rates of interfaces

Table 4. Round completion time (seconds)

Round	Mean	SD	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.

Participants carry out three rounds of tasks. Table 4 summarizes the speed of gesture input for each round. We also conducted repeated measures ANOVA with round as the factor. The result shows that there is a significant difference among rounds with

F (2, 38) =8.48 (p=0.004). However, there are no significant differences in terms of error rates among rounds.

5 Discussion

The results of the experiment show that there is no difference between Digit10 and Bits8 and small difference between Bit4 and Depth6 in terms of completion time. That is, the number of gestures per input (Table 1) has more influence on the completion time than the size of the gesture library, since gesture recognition takes relatively less time (Figure 2) compared to user's action.

The error increased drastically in Depth10 (Figure 3). The size of the gesture library might be one factor (r=0.897). Our log file reveals that approximately 80% of the Depth10 errors are related to triangle gestures. Thus, the selection of proper gestures is also crucial.

The results indicate that the completion time is reduced over trial within short period of time (Table 4). Moreover, we have clearly perceived that participants' tense in performing task is relieved over trial.

We believe that Bit8 interface is the best choice for MimicGesture with respect to its higher transmission speed (mean 16.86 sec with SD=4.23) and comparatively lower error rate (mean 13% errors per input). This speed is less than that of some popular OOB channels (Barcode 37sec, Inputting Alphanumeric 40sec, BEDA around 45sec, Beep&Blink combinations around 30sec and Loud&Clear 20sec). The mean error rate is also lower than 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%) [4] [5].

We asked each participant to fill in a small questionnaire regarding their experience. During the experiment, they did not really distinguish the interfaces from each other; thus, the questions were about the overall impressions of the gesture input. 70% of the participants had a positive attitude toward the ease of our method. Also, 70% of the participants were satisfied with the time they spend.

During the experiment, we observed a number of important findings that might help with the improvement of our OOB channel. First of all, the speed of the users varies and it noticeably affects the success of the gesture input. MimicGesture does not adapt its gesture templates, but it could adapt user's speed by giving some feedback. Second, the sequence of gestures is carried out in discrete manner, but participants prefer the continuous gestures. In other words, pressing a button increases the burden on them. Finally, participants expressed that they did not like holding the phone in a fixed position.

6 Conclusion

This paper demonstrates a new OOB channel, called MimicGesture, for secure device pairing. In this channel, the human user performs gestures with a mobile device that has a built-in accelerometer to transmit the authentication data.

A usability study has been conducted in order to check the viability of our channel. The overall evaluation proves that our gesture-based OOB channel is viable and competitive with the existing OOB channels.

We have also performed the user evaluation of four different gesture sets to see which library size might be more usable in practice. We suggest that a library size of 8 gestures is appropriate for MimicGesture OOB channel with respect to its higher transmission speed (mean 16.86 sec with SD=4.23) and comparatively less error rate (mean 13% errors per input).

7 References

1. Canny, J.: The Future of Human-Computer Interaction. In: Queue-HCI. 4, 6 (2006)
2. Goodrich, M. T., Sirivianos, M., Solis, J., Tsudik, G., Uzun, E.: Loud and Clear: Human-Verifiable Authentication Based on Audio. In: Distributed Computing Systems Conference. IEEE Press (2006).
3. Ion, I., Langheinrich, M., Kumaraguru, P., Capkun, S.: Influence of User Perception, Security Needs, and Social Factors on Device Pairing Method Choices. In: Usable Privacy and Security Symposium. ACM Press (2010).
4. Kainda, R., Flechais, I., Roscoe, A.W.: Usability and Security of Out-Of-Band Channels in Secure Device Pairing Protocols. In: Usable Privacy and Security Symposium. ACM Press (2009)
5. Kumar, A., Saxena, N., Tsudik, G., Usun, E.: A Comparative Study of Secure Device Pairing Methods In: Pervasive and Mobile Computing. 5, 6 (2009)
6. Liu, J., Wang, Z., Zhong, L., Wickramasuriya, J., Vasudevan, V.: uWave: Accelerometer-based Gesture Recognition and Its Applications. In: Pervasive Computing and Communications Conference. IEEE Press (2009)
7. McCune, J.M., Perrig, A.: Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication. International Journal of Security and Networks. 4, 1/2 (2009)
8. Niezen, G.: The optimization of gesture recognition techniques for resource-constrained devices. Master of Engineering Thesis, University of Pretoria, South Africa (2008)
9. Prasad, R., Saxena, N.: Efficient Device Pairing Using “Human-Comparable” Synchronized Audiovisual Patterns. In: Applied Cryptography and Network Security Conference. Springer (2008)
10. Saxena, N., Uddin, M. B.: Automated Device Pairing for Asymmetric Pairing Scenarios. In: Information and Communication Security Conference. Springer (2008)
11. Soriente, C., Tsudik, G., Uzun, E.: BEDA: Button-Enabled Device Association. In: Security for Spontaneous Interaction Workshop. (2007)
12. Soriente, C., Tsudik, G., Uzun, E.: HAPADEP: Human-Assisted Pure Audio Device Pairing. In: Information Security Conferenc. Springer (2008)
13. Stajano, F., Anderson, R.: The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In: Security Protocols Workshop. Springer (1999)
14. Vaudenay, S.: Secure Communications over Insecure Channels Based on Short Authenticated Strings. In: International Cryptology Conference. Springer (2005)
15. Wu, J., Pan, G., Zhang, D., Qi, G., Li, S.: Gesture Recognition with a 3-D Accelerometer. In: Ubiquitous Intelligence and Computing Conference. Springer (2009)