Pairing Devices with Good Quality Output Interfaces

Nitesh Saxena and Jonathan Voris
Polytechnic University
nsaxena@duke.poly.edu, jvoris@cis.poly.edu

Abstract

"Pairing" is referred to as the operation of achieving authenticated key agreement between two human-operated devices over a short-range wireless communication channel (such as Bluetooth or WiFi). The devices are ad hoc in nature, i.e., they can neither be assumed to have a prior context (such as pre-shared secrets) with each other nor do they share a common trusted on- or off-line authority. However, the devices can generally be connected using auxiliary physical channel(s) (such as audio, visual, etc.) that can be authenticated by the device user(s), and thus form the basis for pairing.

In this paper, we present the results of a user study of a technique to pair two devices (such as two cell phones) which have good quality output interfaces (in the form of display/speaker/vibration).

Keywords: *Authentication, Key Agreement, Device Pairing*

1 Introduction

Short-range wireless communication, based on technologies such as Bluetooth and WiFi, is becoming increasingly popular and promises to remain so in the future. With this surge in popularity, come various security risks. A wireless communication channel is easy to eavesdrop upon and to manipulate, and therefore a fundamental security objective is to secure this communication channel. In this paper, we will use the term "pairing" to refer to the operation of bootstrapping secure communication between two devices connected with a short-range wireless channel. Examples of pairing from day-to-day life include pairing of a WiFi laptop and an access point, a Bluetooth keyboard and a desktop, and so on. Pairing would be easy to achieve if there existed a global infrastructure enabling devices to share an on- or off-line trusted third party, a certification authority, a PKI or any pre-configured secrets. However, such a global infrastructure is close to impossible to come by in practice, thereby making pairing an interesting and challenging real-world research problem.[1]

A recent research direction to pairing is to use an auxiliary physically authenticatable channel, called an out-of-band (OOB) channel, which is governed by humans, i.e., by the users operating the devices. Examples of OOB channels include audio, visual channels, etc. Unlike the wireless channel, on the OOB channel, an adversary is assumed to be incapable of modifying messages. It can eavesdrop on, delay, drop and replay them, however. A pairing scheme should therefore be secure against such an adversary.

The usability of a pairing scheme based on OOB channels is clearly of utmost importance. Since OOB channels typically have low bandwidth, the shorter the data that a pairing scheme needs to transmit over these channels, the better the scheme becomes in terms of usability.

Various pairing protocols have been proposed so far. These protocols are generally based on bidirectional automated device-to-device (d2d) OOB channels. Such d2d channels require both devices to have transmitters and corresponding receivers. In settings where d2d channel(s) do not exist (i.e., when at least one device does not have a receiver) and even otherwise, similar protocols can be based upon device-to-human (d2h) and human-to-device (h2d) channel(s) instead. Depending upon the protocol, only two d2h channels may be sufficient, such as in case when the user has to perform a very simple operation (such as "comparison") of the data received over these channels. Clearly, the usability of d2h and h2d channel establishment is even more critical than that of a d2d channel.

Earlier pairing protocols required at least 160 to 80 bits of data to be transmitted over the OOB channels. The simplest protocol [1] involves devices exchanging their public keys over the wireless channel, and authenticating them by exchanging (at least 80-bit long) hashes of the corresponding public keys over the OOB channels. The more recent, so-called SAS- (Short Authenticated Strings) based protocols, [5], [7] and [14], reduce the length of data to be transmitted over the OOB channels to approximately 15 bits.[2]

Based on the above protocols, a number of pairing schemes with various OOB channels have been proposed. These include schemes based on two bidirectional d2d infra-red channels [1]; two bidirectional d2d visual channels consisting of barcodes and photo cameras [6]; a unidirectional d2d visual channel consisting of blinking LED and video camera plus a unidirectional d2h channel consisting of a blinking LED and a unidirectional h2d channel [9]; two audio/visual d2h channels consisting of MadLib sentences and displayed text [4]. In addition, the SAS protocols trivially yield pairing schemes involving two bidirectional d2h and h2d channels – the user reads 15 bits of data displayed on one device and inputs it on the other, and vice versa. [16] proposed pairing schemes that require the user to compare

[1] The problem has been at the forefront of various recent standardization activities, see [15].

[2] The concept of SAS-based authentication was first introduced by Vaudenay in [17].

the data transmitted over two d2h SAS channels.

Most recently, in [10], we proposed a new pairing scheme that is universally applicable to pair any two devices, the focus being on devices which do not have good quality output interfaces. The scheme can use any of the existing SAS protocols and does not require devices to have good transmitters or any receivers, e.g., only a pair of LEDs are sufficient. The scheme involves users comparing very simple audiovisual patterns, such as "beeping" (using a basic speaker) and "blinking" (using LEDs), transmitted as simultaneous streams, forming two synchronized d2h channels.

Our Contributions. In this paper, we focus on pairing devices, such as cell phones, which have good quality output interfaces, such as in the form of display, speaker and vibration. Basically, we use the same idea as in [10], however, we make use of better output interfaces which are often available on some devices. We wanted to investigate if the usability and efficiency of the scheme of [10] will improve if devices have better (and in some cases) multiple output interfaces. We extend the scheme of [10] to the following; the Flash-Flash and Vibrate-Vibrate combinations, which send output to the users using a "flashing" screen and vibration, respectively, and the All-All combination, which makes use of all three outputs, flashing screen, vibration and "beeping", simultaneously. We also compare our schemes with a simple scheme (called Num-Num) that requires the user to compare two 4-digit numbers displayed on devices' screens.

Our results indicate that no single scheme is a true winner. However, the All-All combination turns out to be the safest, in that it has very low false negatives, whereas the Vibrate-Vibrate combination turns out to be the most user-friendly.

Organization. The rest of the paper is organized as follows. In Section 2, we review prior pairing schemes. In Section 3, we describe the security model and summarize relevant protocols. In Section 4, we present our scheme, its design, implementation and performance.

2 Related Work

There exists a significant amount of prior work on the general topic of pairing. Due to lack of space, we only summarize it here. For a detailed description, refer to the related work section in [10].

In their seminal work, Stajano, et al. [13] proposed to establish a shared secret between two devices using a link created through a physical contact (such as an electric cable). Balfanz, et al. [1] extended this approach through the use of infrared as a d2d channel – the devices exchange their public keys over the wireless channel followed by exchanging (at least 80-bits long) hashes of their respective public keys over infrared.

Another approach taken by a few research papers is to perform the key exchange over the wireless channel and authenticate it by requiring the users to manually and visually compare the established secret on both devices. Since manually comparing the established secret or its hash is cumbersome for the users, schemes were designed to make this visualization simpler. These include Snowflake mechanism [3] by Levienet et al., Random Arts visual hash [8] by Perrig et al. etc.

Based on the pairing protocol of Balfanz et al. [1], McCune et al. proposed the "Seeing-is-Believing" (SiB) scheme [6]. SiB involves establishing two unidirectional visual d2d channels – one device encodes the data into a two-dimensional barcode and the other device reads it using a photo camera.

Goodrich, et al. [4], proposed a pairing scheme based on "MadLib" sentences. This scheme also uses the protocol of Balfanz et al. The main idea is to establish a d2h channel by encoding the data into MadLib sentences, which the users can easily compare.

As an improvement to SiB [6], Saxena et al. [9] proposed a new scheme based on visual OOB channel. The scheme uses one of the SAS protocols [5], and is aimed at pairing two devices (such as a cell phone and an access point), only one of which has a relevant receiver (such as a camera).

A very recent proposal, [11], focuses on pairing two devices with the help of "button presses" by the user. The scheme described in the paper is based upon a protocol that first performs an unauthenticated Diffie-Hellman key agreement and then authenticate the established key using a short password. Such a short password can be agreed upon between the two devices via three variants using button presses.

Uzun et al. [16] carry out a comparative usability study of simple pairing schemes. They consider pairing scenarios where devices are capable of displaying 4-digits of SAS data. In what they call the "Compare-and-Confirm" approach, the user simply reads and compares the SAS data displayed on both devices. The "Select-and-Confirm" approach, on the other hand, requires the user to select a 4-digit string (out of a number of strings) on one device that matches with the 4-digit string on the other device.

In [12], the authors consider the problem of pairing two devices which might not share any common wireless communication channel at the time of pairing, but only share a common audio channel.

3 Communication and Security Model, and Applicable Protocols

Our pairing protocols are based upon the following communication and adversarial model [17]. The devices being paired are connected via two types of channels: (1) a short-range, high-bandwidth bidirectional wireless channel, and (2) auxiliary low-bandwidth physical OOB channel(s). Based on device types, the OOB channel(s) can be device-to-device (d2d), device-to-human (d2h) and/or human-to-device (h2d). An adversary attacking the pairing protocol is assumed to have full control on the wireless channel; namely, it can eavesdrop, delay, drop, replay and modify messages. On the OOB channel, the adversary can eavesdrop, delay, drop, replay and re-order messages, however, it can not modify them. In other words, the OOB channel is assumed to be an authenticated channel. The security notion for a pairing protocol in this setting is adopted from the model of authenticated key agreement due to Canneti and Krawczyk [2]. In this model, a multi-party setting is considered wherein a number of parties simultaneously run multiple/parallel instances of pairing protocols. In practice, however, it is reasonable to assume just two-parties running only a few serial/parallel instances of

the pairing protocol. For example, during authentication for an ATM transaction, there are only two parties, namely the ATM machine and a user, restricted to only three authentication attempts. The security model does not consider denial-of-service (DoS) attacks. Note that on wireless channels, explicit attempts to prevent DoS attacks might not be useful because an adversary can simply launch an attack by jamming the wireless signal.

To date, three three-round pairing protocols based on short authenticated strings (SAS) have been proposed [7], [5] and [14]. In a communication setting involving two users restricted to running three instances of the protocol, these SAS protocols need to transmit only k ($= 15$) bits of data over the OOB channels. As long as the cryptographic primitives used in the protocols are secure, an adversary attacking these protocols can not win with a probability significantly higher than 2^{-k} ($= 2^{-15}$). This gives us security equivalent to the security provided by 5-digit PIN-based ATM authentication.

Recall that the pairing scheme that we propose in this paper, similar to the schemes of [10], requires the users to "compare" the data transmitted over two d2h channels. Our scheme can be based on any of the existing SAS protocols. This is because in all protocols, the SAS messages are computed as a common function of the public keys and/or random nonces exchanged during the protocol, and therefore the authentication is based upon whether the two SAS messages match or not (see [7], [5] and [14]).

4 Pairing Using Synchronized Outputs

In this section, we describe the design and implementation of our pairing schemes based on the Flash-Flash, Vibrate-Vibrate and All-All combinations, as well as our study of their experimental usability. We also compare these combinations with the simple scheme Num-Num.

4.1 Design and Implementation

Our objective was to develop pairing schemes that leverage the decent quality output interfaces found on most ubiquitous devices such as mobile phones and controllers. This is unlike the motivation for the schemes in [10] which focused upon devices which lack such output interfaces. The output interfaces that we utilize in our schemes are in the form of "flashing" of screen, "vibration" and "beeping" using a speaker. What we call the Flash-Flash combination is implemented via the flashing of backlit LCD screen and the Vibrate-Vibrate combination using vibration functionality. Basically, we wanted to determine how to use various output interfaces in a way that placed minimal burden on device users by being as short and simple as possible. In particular, since devices of these kind usually feature more than one method of output, we desired to test whether the use of multiple simultaneous output channels made the pairing process any easier for users. To this end, we implemented the All-All combination using the aforementioned flashing display and vibration, as well as a beep in the form of a brief alert noise or "ringtone.". Since we learned from our experience with the schemes in [10] that human users generally do not tend to prefer "asymmetry" (such as when two devices use different output interfaces), we decided not to proceed with combinations involving different output channels, such as Flash-Vibrate, etc. Moreover, it was also shown in [10] that the pairing combination, Beep-Beep, involving two similar audio-based output channels is error-prone, we did not try to tinker with this combination. However, since a pairing scheme (referred to as Num-Num) that requires the user to compare two 15-bit strings encoded as two 4-digit numbers appears to be most basic and simple, we wanted to experiment with it and compare it with our other combinations Flash-Flash, Vibrate-Vibrate and All-All.

A pairing scheme, in its entirety, consists of three phases: (1) the device discovery phase, wherein the devices exchange their identifiers over the wireless channel prior to communicating, (2) the pairing protocol execution phase, wherein the devices execute the desired pairing protocol over the wireless channel, and (3) the authentication phase, where the devices, using the OOB channels, authenticate the messages exchanged during the previous phase. For the sake of our experimentation, we skipped the first two phases and concentrated on the third phase, because our main goal was to test the feasibility of the way we intended to implement the OOB channels, i.e., using the Flash-Flash, Vibrate-Vibrate, All-All and Num-Num combinations. As mentioned previously, our pairing scheme can be built on top of any of the SAS protocols [7], [5] and [14].

Let us assume that we want to pair two devices A and B. Assuming that A and B have already performed the device discovery and protocol execution phases over the wireless channel, the task is now reduced to A and B encoding the 15-bits of their respective SAS data, SAS_A and SAS_B, into vibrating, beeping, flashing, or some combination of these outputs and then transmitting it in a synchronized fashion for the user to compare (clearly, the Num-Num combination does not require any synchronziation.) This encoding should enable the user to easily identify both "good" cases, i.e., when $SAS_A = SAS_B$, and "bad" cases, such as when $SAS_A \neq SAS_B$.

In [10] synchronization was achieved by having one device send a synchronization signal S to the other device over the wireless channel. However, this synchronization signal can get or be delayed, resulting in users being fooled into accepting non-matching SAS strings (for example, strings $SAS_A =$ "010010" and $SAS_B =$ "100100", will appear to be equal to the user if the synchronization signal is delayed by a bit). [10] dealt with this issue by using an END marker to indicate the completion of each SAS string. For these pairing schemes we opted to use a different synchronization technique. Since these schemes are targeted for devices that are hand-held and/or easy to manipulate, we assumed that users would be able to press a button to activate the pairing process on two devices simultaneously. This is a reasonable assumption for a wide range of devices such as cell phones. This approach prevents potential synchronization delays by placing the encoded SAS timing in the hands of the user while also avoiding the added complexity introduced by the use of S and an END marker.

Encoding for Flash-Flash. A '1' bit in the SAS string is signaled by lighting up the backlight of a display for a given "signal interval," while a '0' bit is represented by darkening the display for the same interval. Every bit signal is followed by a brief "sleep interval" of display darkness. This is included to facili-

tate "human-comparison" by allowing users to differentiate two separate bit signals. The time required to compare two SAS strings is inversely proportional to the duration of the signal and sleep intervals – the shorter these intervals, the faster the comparison will run, and vice versa. The optimal value for these intervals needs to be determined through experimentation. The best interval duration is a careful balance between giving users enough time to comfortably compare the encoded SAS data and making the comparison process as brief as possible. Figure 1 illustrates this encoding process using a combined interval of 700 msec.

Encoding for Vibrate-Vibrate. We utilize the vibration function commonly used for feedback in cell phones and video game controllers. A '1' bit of a SAS string triggers a vibration for the signal interval and a '0' bit causes the device to remain stationary. Just as was the case with "flashing," both types of bit signals are followed by a brief "sleep interval" of device stillness. As an example, on a '1' bit the device could vibrate for 210 msec and then stay still for 490 msec, while on a '0' bit the device would remain stationary for the entire 700 msec. Refer to Figure 2 for the vibration encoding process using a combined interval of 700 msec. Optimal values for these intervals also must be found through testing.

Encoding for All-All. The All-All scheme is intended to test whether users find multiple simultaneous forms of output helpful or distracting. As such, this method makes use of the aforementioned flashing and vibrating outputs as well as an audio or "beeping" output. That is, a '1' bit of SAS data is represented by a coterminous display brightening, device vibration, and speaker beeping, whereas a '0' bit is presented as a display darkening, device stillness, and speaker silence. This type of encoding is "all-or-nothing" in the sense that the device either outputs all three types of feedback or none at all. As with the single output kinds of encoding, both '1' and '0' bit signals are followed by a short interval of "sleep." Once again, experiments must be conducted to find optimal interval settings for the "all" intervals. Figure 3 shows this encoding with a combined interval of 700 msec

Implementation. We used two Nokia 6030b mobile phones to conduct our experiments. These phones have several good quality output interfaces, specifically a vibration feature, a speaker, and a 16 bit color, 128 by 128 pixel display. The Nokia 6030b runs the Nokia operating system and supports version 2.0 of the Mobile Information Device Profile (MIDP) specification and version 1.1 of the Connected Limited Device Configuration (CLDC) framework, which are both part of the Java Platform, Micro Edition, or J2ME. To utilize these APIs we wrote our test programs in the Java programming language using the Java Wireless Toolkit version 2.5.2 for CLDC. Because we were only working with the authentication phase of the pairing scheme and not the device discovery or pairing protocol execution phase, and did not make use of a wireless channel for synchronization, no actual wireless connection between the two mobile devices was necessary for our tests.

4.2 Usability Testing

Our schemes were tested with a total of 40 subjects. All our subjects were enthusiastic college students who were familiar with mobile phones but not particularly proficient with the technology. Each tester was given a short summary of our secure device pairing schemes and their potential applications. We explained the experimental setup of the two devices and what they were expected to do while working with the Flash-Flash Vibrate-Vibrate, All-All and Num-Num combinations, but did not give training of any kind to the subjects prior to performing the tests.

The first goal of our user tests was to determine which of the output combinations enabled them to most easily identify SAS signal matches and mismatches. Put differently, we wanted to determine which encoding caused users to commit the least amount of *safe errors* (a false positive, that is, identifying a good case as a bad one or a match as a mismatch) and *fatal errors* (a false negative, that is, identifying a bad case as a good one or a mismatch as a match) [16]. The second goal of our tests was to establish an optimal timing interval for the type of output that users were most comfortable with.

The Set-up and Test Cases. The experiments for the Flash-Flash, Vibrate-Vibrate, All-All and Num-Num combinations were conducted in a graduate research lab room of our university. These test cases were designed to test for matches and mismatches in SAS strings, and consisted of a fixed set of randomly generated strings that were presented to the test subjects in a random order to prevent users from guessing the outcome of test cases based on previous ones. Similar to the findings of [10] as well as some preliminary testing, we observed that users had an easier time noticing mismatches early in a SAS string if the string was prepended with 3 '1' bits as padding to provide users with a learning distance. We used 3 padding bits, which combined with 15 bits of SAS data to produce 18 bit long test case strings. Prior to each test case, the administrator of the test would configure the test parameters on both phones, such as which SAS string and what timing interval to use. Volunteer users then started the test process by pressing a button on each phone simultaneously. After a very brief (100 msec) delay both devices started signaling their particular SAS string via vibrating, flashing, or vibrating, flashing and beeping in accordance with the Flash-Flash, Vibrate-Vibrate and All-All combination in use for that particular test case. For the Num-Num combination, users were not required to press the buttons simultaneously and were asked to compare the 4-digit numbers corresponding to two SAS values displayed on devices' screens.

The Vibrate-Vibrate tests. Since the Vibrate-Vibrate combination was a fresh scheme we didn't have any prior experience with, we decided to test it first. Having established Vibrate-Vibrate as the pairing output of choice, we set about finding the best interval for this pairing scheme with respect to user comfortability and the overall runtime of the pairing scheme. We experimented with a number of such intervals ranging between 300-800 msecs. Each overall interval consisted of approximately 30% signal interval and 70% sleep interval. For example, an overall interval of 300 msecs corresponds to 80 msec of signal interval and 220 msec of sleep interval, 400 msec

Interval	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$
Bits	1	0	1	1	0
Bit Signal	flash	darkness	flash	flash	darkness

Figure 1. Encoding for Flash-Flash using a sleep interval of 700 msec

Interval	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$
Bits	1	0	1	1	0
Bit Signal	vibration	stillness	vibration	vibration	stillness

Figure 2. Encoding for Vibrate-Vibrate using a sleep interval of 700 msec

msecs corresponds to 120 msec of signal interval and 280 msec of sleep interval, 500 msecs corresponds to 150 msec of signal interval and 350 msec of sleep interval, and so on. Each of the 20 volunteers for these tests performed 20 Vibrate-Vibrate pairing test cases for a total of 400 test runs (4 users also participated in 35 preliminary test cases for a grand total of 435 tests). As pointed before, 3 '1' bits were provided as padding on a 15 bit SAS string to produce total test string lengths of 18 bits. Our findings for the Vibrate-Vibrate tests with different intervals are depicted in Table 1.

The results indicate that users commit errors when Vibrate-Vibrate is run with an interval of 300 msec – 650 msec. The error rates range from 2% for a 500 msec interval to around 19% for a 550 msec interval. It is worth noting, however, that these error are comparable to the combinations presented in [10]. The most promising results, however, occur in the 700 msec – 800 msec intervals. For these values, there were no errors observed at all. At this interval level users can easily and comfortably detect all errors in the SAS strings. The entire comparison process takes between 13-15 seconds to complete. This shows that as the sleep interval duration increases, users feel increasingly comfortable performing comparisons using the Vibrate-Vibrate combination. Moreover, almost all users indicated that with time interval around 700 msec, they were able to identify both matching and non-matching test instances with great ease and that they liked the scheme. Having recorded these promising results and users' positive feedback, we moved on to our second bout of tests using which we wanted to compare Vibrate-Vibrate with other combinations.

Interval (msec)	Safe Error Rate (%)	Fatal Error Rate (%)
300	16.667	5.556
400	6.122	4.082
500	10.000	2.000
550	6.250	18.750
600	4.167	8.333
650	4.348	8.696
700	0.000	0.000
750	0.000	0.000
800	0.000	0.000

Table 1. Responses of 20 users when tested for the Vibrate-Vibrate combination

Comparison Tests and their Interpretation. Based on some of our initial tests, we found out that the optimal values of time interval for the Flash-Flash and All-All combinations were 1000 msecs and 700 msecs, respectively. We then performed comparison tests to compare the Vibrate-Vibrate combination (with 700 msecs interval), Flash-Flash combination (with 1000 msecs interval), All-All combination (with 700 msec interval) and the Num-Num combination. Among 20 test volunteers, 60 test cases were performed for each of the Flash-Flash, Vibrate-Vibrate, All-All and Num-Num combinations. The results of these comparison tests are depicted in Tables 2 and 3. The timings shown are averaged over all test runs and include user reaction timings. Unlike our previous tests, we did get some errors this time. Out of all schemes we tested, users committed the least amount of safe errors, in fact none, when using the Num-Num combination. The Num-Num combination also turned out be the fastest. With the All-All combination, the fatal errors were the lowest, around 1.67%. The Vibrate-Vibrate and Flash-Flash combinations both showed some amount of safe as well as fatal errors, with number of errors with the Vibrate-Vibrate combination being on the lower side.

Users were also asked to give a qualitative ranking of which output combination they found to be the easiest to use. About 54% of the test subjects found Vibrate-Vibrate to be the best, while about 36% picked Num-Num and only about 9% selected All-All. None of the subjects rated Flash-Flash the best.

The above test results and user feedback indicate that **none** of the schemes can be termed as a sole winner. All-All and Num-Num turned out to be more or less complementary to each other – All-All has the lowest fatal error rate while Num-Num has the lowest safe error rate. This means that feeding in three simultaneous outputs to the users as in All-All does help users to catch a non-matching instance more accurately, however, it also distracted them in detecting a matching instance. Analogously, Num-Num turned out to be quite easy for the users to detect a matching instance, however, it also made them too lax at times and made them miss some non-matching instances. Flash-Flash and Vibrate-Vibrate on the other hand, showed intermediary error rates, with users missing a few matching as well as a few non-matching instances. While there were some exceptions, users generally stated that they found the visual-only output of Flash-Flash difficult to focus on and the multiple outputs of All-All to be distracting. On the other hand, users found that the tactile feedback of Vibrate-Vibrate demanded less attention and was easier to keep track of.

5 Conclusion

Based on the results obtained, we make the following conclusions regarding the applicability of Vibrate-Vibrate, Flash-Flash, All-All and Num-Num combinations to pairing two de-

Interval	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$	$\leftarrow 700ms \rightarrow$
Bits	1	0	1	1	0
Bit Signal	flash& vibration& beep	darkness& stillness& silence	flash& vibration& beep	flash& vibration& beep	darkness& stillness& silence

Figure 3. Encoding for All-All using a sleep interval of 700 msec

Combination	Average Timing (sec)	Safe Error Rate (%)	Fatal Error Rate (%)
Vibrate-Vibrate	14.3	3.333	5.000
Flash-Flash	18.0	8.333	10.000
All-All	15.2	10.000	1.667
Num-Num	2.1	0.000	9.091

Table 2. Responses of 20 users when tested for the Vibrate-Vibrate, Flash-Flash, All-All and Num-Num combinations

Combination	Ranked #1 (%)	Ranked #2 (%)	Ranked #3 (%)	Ranked #4 (%)
Vibrate-Vibrate	54.55	9.09	36.36	0.00
Flash-Flash	0.00	45.45	18.18	36.36
All-All	9.09	27.27	18.18	45.45
Num-Num	36.36	18.18	27.27	18.18

Table 3. Responses of 20 users when tested for the Vibrate-Vibrate, Flash-Flash, All-All and Num-Num combinations

vices which have decent quality output interfaces. Clearly, no single scheme is a true winner. In practice, we will most likely encounter both safe as well as fatal errors while using any of these combinations. Since most users did not like Flash-Flash that much and since Flash-Flash suffered from both safe as well as fatal errors, we believe that it is not a good choice. This leaves us with Vibrate-Vibrate, All-All and Num-Numcombinations. We noticed that All-All and Num-Num are complementary to each other in terms of errors – All-All has very few (in fact, the fewest of all) fatal errors while Num-Num has very few safe errors. This means that All-All is probably the *safest* of all combinations, as it is easy to detect any non-matching instances (potentially arising due to attacks) using All-All. On the contrary, Num-Num makes it easier for the users to detect matching instances, however, it tends to make users too lax and accept a few non-matching instances as well. Based on our testing using the comparison of 4-digits in Num-Num, we believe that it might be risky to use it as it is. Perhaps comparison of longer (say, 6-digit long) numbers would help in reducing fatal errors with respect to Num-Num[16] as these would require the users to be more attentive.

The combination that most users preferred was Vibrate-Vibrate, although it showed slightly higher safe error rates than Num-Num and sightly higher fatal error rates than All-All. Undoubtedly, our results indicate that Vibrate-Vibrate is *most usable*. We feel that since users like this scheme so much, it might help in improving its security and yield much lower error rates in practice. Of course, Vibrate-Vibrate is only applicable to pairing two devices with vibration capabilities (phones, game controllers, etc.). On a positive side, however, Vibrate-Vibrate is a good (and by far the only "noise-less") choice for bootstrapping communication between devices of two visually impaired people. Recall that the Blink-Blink combination of [10] was shown to be highly error prone because it is hard for users to determine the actual source(s) of "beeping".

In our future work, we plan to test the schemes presented in this paper more rigorously and also compare them with the scheme of [4].

References

[1] D. Balfanz, D. Smetters, P. Stewart, and H. C. Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *NDSS*, 2002.

[2] R. Canetti and H. Krawczyk. Analysis of key-exchange protocols and their use for building secure channels. In *EUROCRYPT*, 2001.

[3] I. Goldberg. Visual Key Fingerprint Code, 1996. Available at http://www.cs.berkeley.edu/iang/visprint.c.

[4] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and Clear: Human-Verifiable Authentication Based on Audio. In *ICDCS*, 2006.

[5] S. Laur, N. Asokan, and K. Nyberg. Efficient mutual data authentication based on short authenticated strings. IACR Cryptology ePrint Archive: Report 2005/424, 2005.

[6] J. M. McCune, A. Perrig, and M. K. Reiter. Seeing-is-believing: Using camera phones for human-verifiable authentication. In *IEEE Symposium on Security and Privacy*, 2005.

[7] S. Pasini and S. Vaudenay. SAS-Based Authenticated Key Agreement. In *PKC*, 2006.

[8] A. Perrig and D. Song. Hash visualization: a new technique to improve real-world security. In *CrypTEC*, 1999.

[9] N. Saxena, J.-E. Ekberg, K. Kostiainen, and N. Asokan. Secure device pairing based on a visual channel. In *IEEE Symposium on Security and Privacy (ISP'06), short paper*, 2006.

[10] N. Saxena and R. Prasad. Efficient device pairing using human-comparable audiovisual patterns. In *Submission*, `http://cis.poly.edu/~nsaxena/docs/sr07.pdf`, 2007.

[11] C. Soriente, G. Tsudik, and E. Uzun. BEDA: Button-Enabled Device Association. In *IWSSI*, 2007.

[12] C. Soriente, G. Tsudik, and E. Uzun. HAPADEP: Human Asisted Pure Audio Device Pairing. In *eprint*, 2007.

[13] F. Stajano and R. J. Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols Workshop*, 1999.

[14] Stanislaw Jarecki and Nitesh Saxena. Encryption-based authenticated key agreement using short authenticated strings. In *Submission*, 2007.

[15] J. Suomalainen, J. Valkonen, and N. Asokan. Security associations in personal networks: A comparative analysis. In *ESAS*, 2007.

[16] E. Uzun, K. Karvonen, and N. Asokan. Usability analysis of secure pairing methods. In *USEC*, 2007.

[17] S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *CRYPTO*, 2005.