Pervasive and Mobile Computing 5 (2009) 734–749

Contents lists available at ScienceDirect

Pervasive and Mobile Computing

journal homepage: www.elsevier.com/locate/pmc

# A comparative study of secure device pairing methods☆

Arun Kumar[a], Nitesh Saxena[a], Gene Tsudik[b], Ersin Uzun[b,*]

[a] Computer Science and Engineering Department, Polytechnic Institute of New York University, United States

[b] Computer Science Department, University of California, Irvine, United States

ARTICLE INFO

*Article history:*
Received 1 March 2009
Received in revised form 6 June 2009
Accepted 3 July 2009
Available online 18 July 2009

*Keywords:*
Usable security
Device pairing
Human-assisted authentication
Man-in-the-middle attacks
Key agreement

ABSTRACT

"Secure Device Pairing" or "Secure First Connect" is the process of bootstrapping a secure channel between two previously unassociated devices over some (usually wireless) human-imperceptible communication channel. Absence of prior security context and common trust infrastructure open the door for the so-called *Man-in-the-Middle* and *Evil Twin* attacks. Mitigation of these attacks requires some level of user involvement in the device pairing process. Prior research yielded a number of technically sound methods relying on various auxiliary human-perceptible out-of-band channels, e.g., visual, acoustic and tactile. Such methods engage the user in authenticating information exchanged over the human-imperceptible channel, thus defending against MiTM attacks and forming the basis for secure pairing.

This paper reports on a comprehensive and comparative evaluation of notable secure device pairing methods. This evaluation was obtained via a thorough analysis of these methods, in terms of both security and usability. The results help us identify methods best-suited for specific combinations of devices and human abilities. This work is an important step in understanding usability in one of the rare settings where a very wide range of users (not just specialists) are confronted with modern security technology.



## 1. Introduction

Ubiquitous wireless communication, based on technologies, such as Bluetooth, WiFi, Zigbee and WUSB, is very popular and will likely become even more so in the near future. Concurrently, increasing proliferation of personal wireless gadgets (including PDAs, cell phones, headsets, cameras and media players) continuously opens up new services and possibilities for ordinary users.

There are many common everyday scenarios where two devices need to "work together." One particularly common case occurs when both devices are controlled by a single user, e.g., a Bluetooth headset and a cell phone, a PDA and a wireless printer, or a laptop and a wireless access point. Whereas, in the "two-user" case, communication is between two devices, each controlled by its owner, e.g., a pair of cell phones.

The surge in the popularity of personal wireless devices triggers various security risks. The wireless communication channel is relatively easy to eavesdrop upon and to manipulate, which raises the threat of the so-called *Man-in-the-Middle* (MiTM) and *Evil Twin* attacks. Therefore, it is important to secure this channel. However, secure communication must be bootstrapped, i.e., devices must be securely paired or initialized. (We use the term "pairing" to refer to the bootstrapping of secure communication between two devices communicating over a wireless channel.)

☆ A previous version of this paper appeared in Kumar, Saxena, Tsudik, Uzun [A. Kumar, N. Saxena, G. Tsudik, E. Uzun, Caveat emptor: A comparative study of secure device pairing methods, in: IEEE International Conference on Pervasive Computing and Communications, PerCom, 2009].

* Corresponding author.
*E-mail addresses:* aashok01@students.poly.edu (A. Kumar), nsaxena@poly.edu (N. Saxena), gts@ics.uci.edu (G. Tsudik), euzun@ics.uci.edu (E. Uzun).

1574-1192/$ – see front matter © 2009 Elsevier B.V. All rights reserved.
doi:10.1016/j.pmcj.2009.07.008

One of the main challenges in secure device pairing is that, because of the diversity of devices and lack of standards, no global security infrastructure exists today and none is likely for the foreseeable future. Consequently, traditional cryptographic means of secure channel establishment (e.g., SSL/TLS) are insufficient for the problem at hand, since unfamiliar devices have no prior security context and no common point of trust. Moreover, most types of wireless communication (e.g., RF-based) are not perceivable by humans. To this end, the research community has already recognized that some form of user involvement is necessary to address the problem of secure device pairing.[1]

One prominent research direction is the use of auxiliary – also referred to as "out-of-band" (OOB) – channels, which are both perceivable and manageable by the user(s) who own and operate the devices. An OOB channel takes advantage of human sensory capabilities to authenticate human-imperceptible (and hence subject to MiTM attacks) information exchanged over the wireless channel. OOB channels can be realized using senses such as audio, visual and tactile. Unlike the main (usually wireless) channel, the attacker cannot remain undetected if it actively interferes with the OOB channel.[2]

Since some degree of human involvement is unavoidable, usability of pairing methods based on OOB channels becomes very important. Also, because a typical OOB channel is low-bandwidth, there is an incentive to minimize the amount of information to be transmitted, for reasons of both usability and efficiency. Recently proposed pairing methods (overviewed in Section 2) typically require transmitting a few (e.g., 15) bits over an OOB channel to achieve a reasonable level of security. However, many devices (e.g., Bluetooth headsets) have limited hardware facilities and/or user interfaces, which makes it challenging to communicate even a few bits and, in turn, complicates user involvement.

At the same time, many current methods require hardware or interfaces not common across the entire spectrum of devices, including: photo/video cameras, infrared or laser transceivers, accelerometers, speakers, microphones, NFC transceivers, USB ports, keypads or displays. Such features, though present on some devices, are not universal. Whereas, certain other methods require truly minimal interfaces – e.g., LED(s) and button(s) – and are thus applicable to many common devices and pairing scenarios. However, using primitive interfaces tends to impose heavier burden on the user.

**Motivation:** The current situation can be described as *a state of flux*. Although many methods have been proposed, each having certain claimed advantages and shortcomings, no comprehensive and comparative usability study has been conducted. There are several reasons motivating a study. First, usability of current device pairing methods remains very unclear. Even methods that have been tested in terms of usability (e.g., [22,3]) have not been contrasted with other methods, i.e., testing was stand-alone and not comparative. Second, prior methods have been developed by security researchers who, not surprisingly, are experts in security and not usability. What appears as simple or user-friendly to a seasoned professional might not be either to an average user. This is because a non-specialist user is often initially clueless about manipulating new devices. A more important issue is that a user might have insufficient comprehension of security issues and the meaning of participation in secure device pairing. Since this topic has matured enough, it is also the right time for experimental assessment of usability factors.

**Contributions:** We overview prominent device pairing methods, implement them using a common software platform and conduct the *first* comprehensive and comparative field study, focusing on both usability and security. The scope of our study is the most common pairing scenario where a single user controls both the devices.[3] Our study yields some interesting results which help us identify most appropriate method(s) for a given combination of devices. We believe that this topic is very important since it sheds light on usability in one of the rare settings where most users (not just specialists) are confronted with security techniques. Also, since most device pairing methods are developed by highly-skilled specialists who are clearly not representative of the general user population, there is a certain gap between what seems to be, and what really is, usable. We hope that our work will help narrow this gap.

Compared to its prior conference incarnation [4], this paper makes several new contributions. In general, we present thorough statistical analysis of data gathered in the course of our usability study, in addition to the more informal treatment provided in [4]. Specifically, we perform statistical tests to explore and compare tested pairing methods in terms of robustness (i.e., error tolerance) and usability (i.e., speed, ease-of-use and likelihood of successful completion of pairing). In addition to investigating the effect of each usability measure separately for different methods, we study the combined effect of all measures taken together, via principle component and cluster analysis. The new analysis offers insights and statistical evidence about security and usability of individual methods as well as their comparative qualities.

**Organization:** Section 2 summarizes notable cryptographic protocols and secure device pairing methods. Section 3 discusses some pre-study design choices and criteria, followed by usability testing details in Section 4. Next, in Section 5, we present our logged usability study data. In Section 6, we discuss the analysis of the data and attempt to interpret the observations. We conclude with the summary and future work in Section 7.

## 2. Background

We now summarize notable cryptographic protocols and device pairing methods. The term *cryptographic protocol* denotes the entire interaction involved, and information exchanged, in the course of pairing. The term *pairing method* refers

[1] Indeed, this has been the subject of recent standardization activities [1].

[2] It is important to note that this approach only requires the OOB channel to be authenticated but not secret, in contrast to the standard Bluetooth pairing based on "user-selected" secret PINs. Recall that the Bluetooth pairing protocol is insecure against an eavesdropper [2].

[3] A similar study in the context of the two-user scenario is of independent interest; it is being addressed in our on-going work.

to the pairing process as viewed by the user, i.e., the user interactions. As discussed later on, a single cryptographic protocol can be coupled with many pairing methods.

### 2.1. Cryptographic protocols

One simple protocol for pairing was suggested in [5]: devices A and B exchange their respective public keys $pk_A$, $pk_B$ over the insecure channel and the corresponding hashes $H(pk_A)$ and $H(pk_B)$—over the OOB channel. Although non-interactive, the protocol requires $H()$ to be a (weakly) collision-resistant hash function and thus needs at least 80 bits of OOB data in each direction. MANA protocols [6] reduce the size of OOB messages to $k$ bits while limiting attacker's success probability to $2^{-k}$. However, these protocols require a stronger assumption on the OOB channel: the adversary is assumed to be incapable of delaying or replaying any OOB messages.

[7,8] presented the first protocol based on Short Authenticated Strings (SAS), which limits attack probability to $2^{-k}$ for $k$-bit OOB channels, even when the adversary can delay/replay OOB messages. This protocol utilizes commitment schemes (which can be based upon hash functions such as SHA-1, MD5) and requires 4-round of communication over the wireless channel. Subsequently, [9][4] and [11] independently gave 3-round protocols. Both protocols are modifications of the protocol of [7], and both employ (although differently) a universal hash function in computation of the SAS message. Recently, [12, 13] proposed a more efficient SAS protocol which is utilized in a number of pairing methods we tested.

We note that not all secure device pairing methods can be coupled with SAS- or MANA-like protocols. Specifically, some methods rely on the user to generate a random value and somehow enter it into both devices. The two devices then employ user-supplied value as a one-time common secret to authenticate subsequent key exchange performed over the wireless channel. Methods that fall into this group include BEDA Button–Button [3] and Bluetooth PIN entry [2]. Cryptographic techniques appropriate in this context are so-called Password-Authenticated Key Exchange (PAKE) protocols, e.g., [14].

### 2.2. Device pairing methods

Based on the cryptographic protocols described above, a number of pairing methods have been proposed. They operate over different OOB channels and offer varying degrees of usability.

"Resurrecting Duckling" [15] is the initial attempt to address the device pairing problem in the presence of MiTM attacks. It requires standardized physical interfaces and cables. Though appropriate in the 90-s, this is clearly obsolete today, due to the greatly increased diversity of devices. Requiring a physical equipment (i.e., a cable) also defeats the purpose and convenience of using wireless connections.

Another early method is "Talking-to-Strangers" [5], which relies on infrared (IR) communication as the OOB channel and requires almost no user involvement, except for initial setup. Also, it has been experimented with (unlike many other methods), as reported in [16]. However, this method is deceptively simple since IR is line-of-sight and, setting it up requires the user to find IR ports on both devices – not a trivial task for many users – and align them. Also, despite its line-of-sight property, IR is not completely immune to MiTM attacks.[5] The main drawback of IR is that is has been largely displaced by other wireless technologies (e.g., Bluetooth) and is available on very few modern devices.

We note that "Resurrecting Duckling" and "Talking-to-Strangers" share an important advantage: they require no user involvement beyond initiating the protocol.

Another early approach involves image comparison. It encodes the OOB data into images and asks the user to compare them on two devices. Prominent examples include "Snowflake" [17], "Random Arts Visual Hash" [18] and "Colorful Flag" [19]. Such methods, however, require both devices to have displays with sufficiently high resolution. Applicability is therefore limited to high-end devices, such as: laptops, PDAs and certain cell phones. These methods are based on the protocol proposed in [5] which was reviewed earlier. A more practical approach, based on SAS protocols [11,9], suitable for simpler displays and LEDs has been investigated in [20].

More recent work [21] proposed the "Seeing-is-Believing" (SiB) pairing method. In its simplest instantiation, SiB requires a unidirectional visual OOB channels: one device encodes OOB data into a two-dimensional barcode which it displays on its screen and the other device "reads it" using a photo camera, operated by the user. At a minimum, SiB requires one device to have a camera and the other—a display. Thus, it is not suitable for low-end devices.[6] We use the SiB variant from [12,13] which only requires one device to have a camera.

A related approach, called "Blinking Lights" has been explored in [12]. Like SiB, it uses the visual OOB channel and requires one device to have a continuous visual receiver, e.g., a light detector or a video camera. The other device must have at least one LED. The LED-equipped device transmits OOB data via blinking while the other receives it by recording the transmission

4 An extended version of the same paper appeared in [10].

5 This can be confirmed by anyone who has observed two children (each armed with a TV remote) switching channels on, and competing for control of, the same TV set.

6 Albeit, the display requirement can be relaxed in case of a printer; alternatively, a camera-equipped device can snap a photo of a barcode sticker affixed to the *dumber* device. This prompts different security risks.

and extracting information based on inter-blink gaps. The receiver device indicates success/failure to the user who, in turn, informs the other to accept or abort.

Quite recently, [22] developed a pairing method based on synchronized audio-visual patterns. Proposed methods, "Blink–Blink", "Beep–Beep" and "Beep–Blink", involve users comparing very simple audiovisual patterns, e.g., in the form of "beeping" and "blinking", transmitted as simultaneous streams, forming two synchronized channels. One advantage of these methods is that they require devices to only have two LEDs or a basic speaker.

Another recent method is "Loud-and-Clear" (L&C) [23]. It uses the audio (acoustic) OOB channel along with vocalized MadLib sentences which represent the digest of information exchanged over the main wireless channel. There are two L&C variants: "Display–Speaker" and "Speaker–Speaker". In the latter the user compares two vocalized sentences and in the former—displayed sentence with its vocalized counterpart. Minimal device requirements include a speaker (or audio-out port) on one device and a speaker or a display on the other. The user is required to compare the two respective (vocalized and/or displayed) MadLib sentences and either accept or abort the pairing based on the outcome of the comparison. As described in [23], L&C is based on the protocol of [5]. In this paper, to reduce the number of words in the MadLib sentences, we use the L&C variant based on SAS protocols [11,9].

Some follow-on work (HAPADEP [24,25]) considered pairing devices that – at least at pairing time – have no common wireless channel. HAPADEP uses pure audio to transmit cryptographic protocol messages and requires the user to merely monitor device interaction for any extraneous interference. It requires both devices to have speakers and microphones. To appeal to more basic settings, we employ a HAPADEP variant that uses the wireless channel for cryptographic protocol messages and the audio as the OOB channel. In it, only one device needs a speaker and the other—a microphone. Also, the user is not involved in any comparisons.

An experimental investigation [26] presented the results of a comparative usability study of simple pairing methods for devices with displays capable of showing a few (4–8) decimal digits. In the "Compare-and-Confirm" approach, the user simply compares two 4-, 6- or 8-digit numbers displayed by devices. In the "Select-and-Confirm" approach, one device displays to the user a set of (4-, 6- or 8-digit) numbers, the user selects the one that matches the number displayed by the other device. In the "Copy-and-Confirm" approach, the user copies a number from one device to the other. The last variant is "Choose-and-Enter" which asks the user to pick a "random" 4- to 8-digit number and enter it into both devices. All of these methods are undoubtedly simple, however, as [26] indicates, Select-and-Confirm and Copy-and-Confirm are slow and error-prone. Furthermore, "Choose-and-Enter" is insecure since studies show that the quality of numbers (in terms of randomness) picked by the users is very low.

Yet another approach: "Button-Enabled Device Authentication (BEDA)" [3] suggests pairing devices with the help of user button presses, thus utilizing the tactile OOB channel. This method has several variants: "LED–Button", "Beep–Button", "Vibration–Button" and "Button–Button". In the first two variants, based on the SAS protocol variant [12], the sending device blinks its LED (or vibrates or beeps) and the user presses a button on the receiving device. Each 3-bit block of the SAS string is encoded as the delay between consecutive blinks (or vibrations). As the sending device blinks (or vibrates), the user presses the button on the other device thereby transmitting the SAS from one device to another. In the Button–Button variant, which can work with any PAKE protocol (e.g., [14]) the user simultaneously presses buttons on both devices and random user-controlled inter-button-press delays are used as a means of establishing a common secret.

There are also other methods which require hardware that are relatively expensive and uncommon. We briefly summarize a few. [27] suggested using ultrasound as the OOB channel. A related technique uses laser as the OOB and requires each device to have a laser transceiver [28]. A very different OOB channel was considered in "Smart-Its-Friends" [29]: a common movement pattern is used to communicate a shared secret to both devices as they are shaken together by the user. A similar approach is taken in "Shake-Well-Before-Use" [30]. Both techniques require devices to be equipped with 2-axis accelerometers. Although some recent mobile phones (e.g., iPhone) are equipped with it, accelerometers are not common on other devices and physically shaking/twirling is unsuitable for many sensitive as well as stationary (e.g. access points) and large/bulky devices (e.g., laptops).

In this paper, we also identify an important and practical problem of "*rushing user*" behavior in device pairing. We define a rushing user as a user who, in a rush to connect her two devices, tends to skip through the pairing process, if possible. Such a rushing user behavior does exist in practice and in fact, is quite common. It has been shown that computer users tend to be "task-focussed" [31]. For example, in the context of phishing attacks [31], when a user logs on to the website of her bank, her focus is (e.g.,) to pay a bill which is past due; she would tend to ignore any warning indicating a phishing attempt. Similarly, in the context of device pairing, when a user wants to connect her Bluetooth laptop with her cell phone, her primary task is (e.g.,) to transfer her pictures or synchronize her calendar; when she wants to connect her Bluetooth cell phone with a headset, she is eager to speak to someone. The pairing process, from user's perspective, is nothing but a hindrance in her intended task, and therefore she would quickly tend to skip through this process, if possible.

All previously proposed pairing methods can be broadly classified into two categories: (1) device-controlled (DC) method, where the OOB strings are transferred between two devices with the help of the user and the devices decide the outcome of pairing, and (2) user-controlled (UC) method, where the user herself compares the OOB strings output by the devices and decides the pairing outcome. We observe that all UC pairing methods are vulnerable against a rushing user, whereas the DC methods are not. In the UC methods, the user can simply "accept" the pairing, without having to take part in the decision process correctly. On the other hand, in the DC methods, the user is somewhat forced to perform the pairing process correctly, because otherwise she will not be able to connect her two devices.

| | Device/Equipment Requirements | | User Actions | | | |
|---|---|---|---|---|---|---|
| **Pairing Method** | **Sending Device** | **Receiving Device** | **Phase I: Setup** | **Phase II: Exchange** | **Phase III: Outcome** | **OOB Channels** |
| Resurrecting Duckling* | Hardware port (e.g., USB) on both and extra cable | | Connect cable to both devices | **NONE** | **NONE** | Cable |
| Talking to Strangers* | IR port on both | | Activate IR on both & find /align IR ports | **NONE** | **NONE** | IR |
| Visual Comparison: Image, Number or Phrase | Display + user-input on both | | **NONE** | Compare two images, or two numbers, or two phrases | Abort or accept on both devices | Visual |
| Seeing is Believing (SiB)* | Display + user-input | Photo camera + user-output | Activate photo mode on receiving device | Align camera on receiving device with displayed barcode on sending device, take picture | Abort or accept on sending device based receiving device decision | Visual |
| Blinking Lights* | LED + user-input | User-output + Light detector or video camera | Activate light detector or set video mode on receiving device | Initiate transmittal of OOB data by sending device | Abort or accept on sending device based receiving device decision | Visual |
| Loud & Clear<br>▪Display-Speaker<br>▪Speaker-Speaker | User-input on both +<br>▪display on one & speaker on other, or<br>▪speaker on both | | **NONE** | Compare: two vocalizations, or display with vocalization | Abort or accept on both devices | ▪Audio, or<br>▪audio + visual |
| Button-Enabled (BEDA)<br>▪Vibrate-Button*<br>▪LED-Button*<br>▪**Beep-Button*** | User input +<br>▪vibration , or<br>▪LED, or<br>▪beeper | User output +<br>One button + | Touch or hold both devices | For each signal (display, sound or vibration) by sending device, press a button on receiving device | Abort or accept on sending device based receiving device decision | ▪Tactile, or<br>▪Visual + tactile, or<br>▪Audio + tactile |
| Button-Enabled (BEDA)<br>▪Button-Button* | One button on both + user-output on one | | Touch or hold both devices | Simultaneously press buttons on both devices; wait a short time, repeat, until output signal | **NONE (unless synch. error)** | Tactile |
| Copy–and-Confirm* | Display + user-input | Keypad + user-output | **NONE** | Enter value displayed by sending device into receiving device | Abort or accept on sending device based on receiving device decision | Visual |
| Choose-and-Enter* | User input on both devices | | **NONE** | Select "random" value and enter it into each device | **NONE (unless synch. Error)** | Tactile |
| Audio Pairing* (HAPADEP variant) | Speaker + user-input | Microphone + user-output | **NONE** | Wait for signal from receiving device. | Abort or accept on sending device | Audio |
| Audio/Visual Synch.<br>▪**Beep-Beep**<br>▪Blink-Blink<br>▪Blink-Beep | User-input on both +<br>▪Beeper on each , or,<br>▪LED on each, or<br>▪Beeper on one & LED on other | | **NONE** | Monitor synchronized:<br>▪beeping, or<br>▪blinking, or<br>▪Beeping & blinking | Abort on both devices if no synchrony | ▪Visual, or<br>▪ Audio, or<br>▪Audio + visual |
| Smart-its-Friends*, Shake-Well-Before-Use* | 2-axis accelerometers on both + user-output on one | | Hold both devices | Shake/twirl devices together, until output signal | **NONE (unless synch. error)** | Tactile + motion |

***Resistant to a "rushing user" behavior**

**Fig. 1.** Feature summary of notable device pairing methods (methods resistant to "rushing user" behavior are marked with an asterisk (*)).

**Summary of methods:** To summarize our discussion of existing methods, Fig. 1 reflects and compares their prominent features. It uses the following terminology:

- *Sending Device/Receiving Device*—applies to all methods where the OOB channel is used in one direction.
- *Phase I: Setup*—user actions to bootstrap the method.
- *Phase II: Exchange*—user actions as part of the protocol.
- *Phase III: Outcome*—user actions finalizing the method.
- *User-input*—any single-bit input by user, e.g., button.
- *User-output*—any single-bit user-perceivable output, e.g., beeper or LED.
- Methods marked with the asterisk ('*') in the first column offer resistance against "rushing user" behavior.

## 3. Study preliminaries

This section discusses selection criteria for tested methods and devices and the architecture of our software platform.

### 3.1. What methods to test?

As follows from our overview above, there is a large body of prior research on secure device pairing, comprised of a wide range of methods. However, these methods either have not been tested in terms of usability or trial-deployed at all, or have been evaluated in a stand-alone fashion, i.e., have not been compared with other methods. The latter category includes: Talking-to-Strangers (Network-in-a-Box) [16], Compare-and-Confirm, Copy-and-Confirm and Choose-and-Enter [26], Blink–Blink and Beep–Blink [22], as well as four variants of BEDA [3].

There are about twenty methods, counting variations, in Fig. 1. Common sense suggests that it is very difficult, and perhaps futile, to find a stable set of subjects and ask them to test all methods, hoping that results will yield comparative usability metrics. The main obstacle is user fatigue, since twenty methods cannot be tested in one sitting. Consequently, we have to cull the number of methods down to a more manageable number, eliminating those that are obsolete, require hardware which is expensive or uncommon or are deprecated based on prior evaluations. We eliminated the following methods from our study (the corresponding cells are white in the first column of Fig. 1):

- Resurrecting Duckling: obsolete due to cable requirement.
- Talking-to-Strangers: obsolete since IR ports have become uncommon.
- Copy-and-Confirm: performed poorly in prior evaluations due to high user error rate.
- Choose-and-Enter: performed poorly in prior evaluations due to low security.
- Beep–Beep: performed poorly in prior evaluations due to user annoyance and high error rate.
- Beep–Button: since we tested Vibrate–Button and vibration is usually accompanied by buzzing noise, we elected not to test Beep–Button.
- Smart-Its-Friends, Shake-Well-Before-Use as well as Ultrasound- and Laser-based methods: require interfaces uncommon on many different types of devices.

Remaining methods, corresponding to shaded cells in the first column of Fig. 1, have been included in our study. Their inclusion was primarily based upon applicability to a broad and common set of devices.

### 3.2. Test devices

In the entire study we used two Nokia cell phones models[7]: N73 and E61, as test devices. Both models have been released two years earlier (in 2006) and hence did not represent the *cutting edge*. This was done on purpose in order to avoid devices with uncommon or expensive features as well as processors faster than those commonly available at present.

Another reason for choosing these devices is their rich menu of wireless and user interfaces. Recall that our goal is to test many methods utilizing many different OOB channels, including: audio, visual and tactile. For each of these channels, some methods need user-input, user-output or both. The audio channel can require: speaker, beeper or vibration (which produces a buzzing sound as a side-effect). The visual channel can require: LED, screen or camera viewfinder, whereas, the tactile channel can require: vibration, button or keypad. Our test devices have all these features which allows testing all methods consistently. (Otherwise, changing devices across methods would seriously undermine the credibility of results.) Specifically, both N73 and E61 have the following features:

- User-input: keypad (subsumes button), microphone, video camera (subsumes photo).
- User-output: vibration, speaker (subsumes beeper), color screen (subsumes LED).
- Wireless: Bluetooth, IR and, of course, GSM.

In all experiments, Bluetooth was used as the wireless (human-imperceptible) channel. We consider this to be a natural choice, since Bluetooth is widely available and inexpensive. It allows positioning flexibility within reasonable physical space (unlike IR), i.e., 1–15 feet suffices for most settings. It is shorter-range, less susceptible to interference and is simpler to set up than 802.11 WLAN.

For methods that involve beeping, the phone speaker is a natural substitute for a beeper. Whenever a button is needed, one of the keypad keys is easily configured for that purpose. An LED is simulated with a small LED image glowing (alternating between light and dark) on the phone screen.[8]

[7] For N73 specs, see: www.nokiausa.com/A4409012, and for E61— europe.nokia.com/A4142101.

[8] Even though both tested phones have LEDs, there are unfortunately no system calls to access them via Java MIDP.

### 3.3. Implementation details

In comparative usability studies, meaningful and fair results can only be achieved if all methods are tested under similar conditions and settings. In our case, the fair comparison basis is formed by: (1) keeping the same test devices, (2) employing consistent GUI design practices (e.g., safe defaults), and (3) unifying targeted (theoretical) security level for all methods. Our goal is to isolate – to the extent possible – user interaction in different methods as the only independent variable throughout all tests. Minimizing any experimenter-introduced, systematic and cognitive bias is also important. We randomize test order, avoid close physical proximity and interaction between the participant and the experimenter, and automate timing and logging to minimize errors and biases.

Some of the tested methods already had prior working prototype implementations. However, these were mostly developed by their authors who aimed to demonstrate implementation feasibility. Consequently, such prototypes are often: incomplete, buggy and/or fragile as well as very dependent on specific hardware/software platforms. It is nearly impossible to provide a uniform testing environment using available prototypes. Modifying them or implementing each from scratch is also not an option, due to the level of effort required. For stand-alone applications, implementing only the user interface is usually enough for the purposes of usability testing. However, distributed applications, such as secure device pairing, need more than just user interface, since a realistic user experience is unattainable without any connection between devices.

To achieve a unified software platform, our implementation used the open-source comparative usability testing framework developed by Kostiainen, et al. [32]. It provides basic communication primitives between devices as well as automated logging and timing functionality. However, we still had to implement separate user interfaces and simulated functionality for all tested methods. We used JAVA-MIDP to implement all methods and created several test-cases for "no-attack" and "under-attack" scenarios. (Here, *attack* is limited to MiTM attacks.)

For all methods, we kept the SAS string (and secret OOB string in Button–Button) length constant at 15 bits. In practice, a 15 bit of SAS provides a reasonable level of security [7]. We also tried to keep all user interfaces similar, while applying same design practices, i.e, safe default selection prompts, clear instructions, simple language, etc. All methods are precisely timed from the start to the end of user interaction.

We believe that, in our implementation, user experience and interaction models are very realistic. For most methods tested, the only difference between our variant and a real method is that we omitted the initial rounds of the underlying cryptographic protocol (e.g., SAS) that use the wireless channel, i.e., do not involve the user. Instead, our implementation supplies devices with synthetic SAS strings to easily simulate normal and MiTM attack scenarios. However, since messages over the wireless channel are completely transparent to the user, our *simulation*, from the user's perspective closely resembles the real-life version.

The only methods that have noticeable difference from real-world implementations are SiB and Blinking–Lights. Due to the difficulty of implementing image and video processing on mobile phones, we choose to simulate their operation.[9] In particular, we saved the captured barcode image (in SiB) and the recorded video of blinking screen (in Blinking–Lights) on the test device and *manually* processed them later. From the user's perspective, the only difference is that s/he never sees the pairing method (SiB or Blinking–Lights) fail even though it could have happened in real life. For instance, in SiB, even if the photo snapped by the user is of insufficient quality for successful real execution (e.g., it fails to capture the entire barcode), the device still shows a "pairing successful" message to the user, in our simulation. However, as mentioned above, we later manually analyze images and videos and determine those that would cause failed executions in a real-world setting. Also the execution times of these two methods were disfavored by few seconds in our implementations since a system security notification was popping up each time the camera was activated by a third party software.

## 4. User study details

Having implemented all selected pairing methods on a common platform, we are ready to start the user study. Our goal is to evaluate and compare the pairing methods with respect to the following factors:

1. Robustness: how often each method leads to false positives (or rejection of a successful pairing instance) and false negatives (or acceptance of a failed pairing instance). Following the terminology introduced in [26], we will refer to the errors in the former category as *safe errors* and the latter as *fatal errors*.
2. Usability: how each method fares in terms of (a) completion time, (b) successful completion (i.e., no safe errors), and (c) users' ease-of-use perception and personal preference.[10]

The main challenge we face is the sheer number of methods to be tested—13. Plus, each method involved a few test-cases of its own. This means that each participating user has to test nearly 50 test-cases. Clearly, it is impossible for a user to test (and for the test administrator to manage) all test-cases in one sitting. Nor is this recommended since fatigue tends to

[9] The current CMU implementation of SiB is supported on Nokia models N70 [33] and 6620 [21] as receiving devices. The current Nokia implementation of Blinking–Lights is supported only on Nokia 6630 [34] as the receiving device. Since we wanted to perform our tests on the same devices throughout, neither implementation could be used. Moreover, porting existing implementations onto our devices was not viable since characteristics of cameras on these phones are quite different and each phone performs its own set of adjustments to images and video, on the software level.

[10] Since the three metrics directly or indirectly affect the usability of the pairing process, we club them all as a means of evaluating for "Usability".

**Table 1**
Breakdown of tested pairing methods.

| Batch I | Batch II | Batch III |
|---|---|---|
| Image comparison | Speaker–Speaker | SiB |
| Number comparison | Speaker–Display | Blinking–Lights |
| Phrase comparison | Button–Button | HAPADEP variant |
| Blink–Blink | LED–Button | Number comparison |
| Beep–Blink | Vibrate–Button | Speaker–Display |
| | | Blink–Blink |
| | | Vibrate–Button |

influence test results. To remedy the problem, we pursued the study in three batches. Table 1 shows the methods tested in each batch. The choice of methods in each batch was based on our estimation (from prior published results) of the time and tedium factor for a given method. Also, some methods in the last batch were repeated, so that all methods could be meaningfully compared (It was not assumed that users remember methods tested in prior batches).

### 4.1. Study participants

We recruited 20 participants[11] for our three-batch study which lasted over a month. They were chosen on a first-come first-serve basis from respondents to recruiting posters and emails. Prior to recruitment, each participant was briefed on the estimated amount of time required to complete the tests and on the importance of completing all three test batches.

Participants were mostly university students, both graduate and undergraduate. This resulted in a fairly young, well-educated and technology-savvy[12] group. As mentioned earlier, we postulate that, if highly-motivated and technology-savvy young people do not react well to a given method, the same method will perform worse when used by the general population. On the other hand, we acknowledge a method that fares well with our participants, might not be equally appreciated by other users. Thus, our study represents only the first step towards identifying methods suitable for the broad cross-section of user population.

We prepared two questionnaires: *background* – to obtain participant demographics and *post-test* – for feedback on methods tested.

None of the study participants reported any physical impairments that could interfere with their ability to complete given tasks. The gender split was: 70% male and 30% female (We attribute the uneven numbers to the nature of the test location.) Gender and other information was collected through *background questionnaires* completed prior to testing.

### 4.2. Test-cases

Based on the type of each tested method, we created several test-cases simulating normal as well as error scenarios. For all methods involving manual comparison of OOB strings, i.e, the ones involving number comparison, phrase comparison, three L&C variants, Blink–Blink and Beep–Blink, we created 5 test-cases each; one where both OOB strings match and four where they do not. The latter test-cases consisted of 3 pairs of OOB strings mismatched in first, last, and middle digit/word/pattern, and one pair of OOB strings mismatched in random multiple (2 or 4) digits/words/patterns. In case of Blink–Blink an additional test-case was generated to test for the lack of synchrony between the two devices. This was unlike Beep–Blink, where the devices were synchronized manually by the users. For the method based on image comparison, we created only two test-cases: one where the two images match and the other where they do not. Mismatch of a few bits in the OOB strings was not tested because in this method the OOB strings are the result of a (visual) collision-resistant hash function and such a mismatch only occurs with a negligible probability.

For BEDA LED–Button and Vibrate–Button variations, we created one matching and one mismatched test-cases. This was done in order to evaluate whether users can correctly transfer (1) the OOB string from the sending device to the receiving device and, (2) the result of pairing (a success or failure) from the receiving device to the sending device. For the BEDA Button–Button variant, we only had one test-case (which we repeated twice) which followed the normal behavior until the pairing was successful.

For automated pairing methods (SiB, Blinking–Lights and the HAPADEP variant), we created one test-case each (and repeated it twice), where the receiving device always receives the image, video or audio captured with the help of the user, and always accepts it as legitimate. The purpose was to evaluate how much burden the automated schemes impose on the user. In other words, we simply wanted to test how easy or hard it would be for a user to take a photo or record a video/audio from a sending device. We did not consider scenarios where an MiTM attacker fools the users into capturing the OOB messages from an attacking device.[13]

11 As is well-known, a usability study performed by 20 participants captures over 98% of usability related problems [35].

12 All participants were regular computer users with at least one wireless personal device.

13 Such attack scenarios appear more plausible in case of automated audio rather than in visual channels, e.g., in a noisy environment. However, simulating such attack scenarios and testing them in a meaningful and fair manner for both channels seems hard.

### 4.3. Testing process

Our study was conducted in a variety of campus venues including, but not limited to: student laboratories, cafés, student dorms/apartments, classrooms, office spaces and outdoor terraces. This was possible since the test devices were mobile, test setup was more-or-less automated and only a minimal involvement from the test administrator was required.

After giving them a brief overview of our study goals (prior to the first batch), we asked the participants to fill out the background questionnaire in order to collect demographic information. In this questionnaire, we also asked the participants whether they had any visual or hearing impairments, or any other condition that may interfere with their sensing of vibration, holding objects steady or their reflexes. Next, the participants were given a brief introduction to the cell phone devices used in the tests.

Each participating user was then given the two devices and asked to follow on-screen instructions shown during each task to complete it. As already mentioned in Section 3.3, to reduce the learning effect on test results, the tasks were always presented to the user in random order. User interactions throughout the tests and timings were logged automatically by the testing framework. After completing the tasks in each batch of the tests, each user filled out a post-test questionnaire form, where they provided feedback on various methods tested in that particular batch. The users were also given a few minutes of free discussion time, where they explained to the test administrator about their experience with the various methods they tested.

## 5. Logged observations

We collected data in two ways: (1) by timing and logging user interactions, and (2) via questionnaires and free discussions.

For each method, completion times, errors and actions were automatically logged by the software. All logged data is summarized (rather densely) in Fig. 2. For easier understanding, the timing information for each method is graphed (separately) in Fig. 3.

Through the post-test questionnaire, we solicited user opinions about all tested methods. Participants rated each method for its ease-of-use: very easy, easy, hard or very hard. The ease-of-use ratings are graphed in Fig. 4. Participants were also asked to order methods from most to least preferred, within each group of methods (i.e., Display Comparison, Button-Enabled Transfer, Synchronized Comparison, Automated OOB Transfer), as shown in Fig. 5.

## 6. Analysis of logged observations and interpretations

We conducted analysis (of statistical nature, wherever applicable) in order to better understand the logged observations presented in the previous section and identify the significance of similarities and differences among tested methods. In this section, we discuss the results based on the analysis of robustness and usability measures.

### 6.1. Analysis of robustness measures

Observing the average error rates as depicted in Fig. 2, we find that most methods fare relatively well, reporting either no errors or low rates of around 5%, which is considered safe (note that such errors are unlikely to occur in practice, under normal circumstances). However, as error rates climb into 10% and above range, there might be reasons for concern. The highest error rates are reported by Blink–Blink (30%) and Beep–Blink (20%), although only for the OOB strings mismatches in leading bits.[14] Image Comparison yields a 15% rate but for "false positive" errors which are considered safe. Somewhat less error-prone are Number Comparison and L&C Speaker–Speaker, each with a 10% fatal error rate. L&C Display–Speaker showed a smaller, 5% fatal error rate, only in test-cases involving the first word mismatch. BEDA variants LED–Button and Vibrate–Button are completely error-free, which implies that users correctly synchronize the blinking and vibration on one device with the button pressing on the other device. This also means that users accurately transferred a one-bit result indicating success or failure, from one device to the other. (This was a concern for Blinking–Lights in [12].).

Since the fatal and safe error rates represent categorical data, we used pairwise chi-square tests to identify significant differences between methods. In our user study, we have tested the security of applicable methods under varying levels of differences between the comparison values (SAS strings) via a number of test-cases, wherever applicable. When all test-cases were accounted in, the differences among fatal errors corresponding to various methods were not found to be significant. However, we observed some significant differences when the beginning of the SAS strings (e.g., the first word, first digit, or first blinking or beeping signal) were different. In this specific attack case, as the average error rates show, Blink–Blink yields significantly more fatal errors than Phrase Comparison and Display–Speaker. Moreover, Beep–Blink also encounters significantly more fatal errors than Phrase Comparison when the mismatch is at the very beginning of the SAS-strings.

[14] Since first bit mismatch occurs with a probability of $2^{k-1}$ for $k$-bit long strings, one can instead use $(k+1)$-bit strings to achieve the same level of security.

| Method name | Specific test case | Avg. completion time (seconds) | Avg. fatal error rate | Avg. safe error rate |
|---|---|---|---|---|
| Image Comparison | Matching Images | **12.7 (sd*=10.7)** | N/A | **15%** |
| | Mismatched Images | 6.7 (sd=3.8) | 0% | N/A |
| Number Comparison | Matching Numbers | **8.6 (sd=4.9)** | N/A | 0% |
| | 2-Digit Mismatch | 7.3 (sd=4.5) | 0% | N/A |
| | First Digit Mismatch | 4.5 (sd=1.6) | **10%** | N/A |
| | Last Digit Mismatch | 5.3 (sd=2.6) | 0% | N/A |
| | Middle Digit Mismatch | 7.1 (sd=5.2) | **5%** | N/A |
| Phrase Comparison | Matching Phrases | **12.7 (sd=8.0)** | N/A | **10%** |
| | 2-Word Mismatch | 6.7 (sd=3.4) | **5%** | N/A |
| | First Word Mismatch | 6.3 (sd=3.4) | 0% | N/A |
| | Last Word Mismatch | 9.4 (sd=8.3) | 0% | N/A |
| | Middle Word Mismatch | 7.2 (sd=4.4) | **5%** | N/A |
| BEDA (Led-Button) | Accept Signal | **49.5 (sd=27.5)** | N/A | 0% |
| | Reject Signal | N/A | 0% | N/A |
| BEDA (Vibrate-Button) | Accept Signal | **44.3 (sd=18.0)** | N/A | 0% |
| | Reject Signal | N/A | 0% | N/A |
| Loud & Clear (Display-Speaker) | Matching Phrases | **15.5 (sd=6.3)** | N/A | 0% |
| | 2-Word Mismatch | 13.6 (sd=7.0) | 0% | N/A |
| | First Word Mismatch | 11.7 (sd=3.7) | **5%** | N/A |
| | Last Word Mismatch | 12.3 (sd=5.5) | 0% | N/A |
| | Middle Word Mismatch | 11.6 (sd=3.4) | 0% | N/A |
| Loud & Clear (Speaker-Speaker) | Matching Phrases | **21.3 (sd=6.8)** | N/A | 0% |
| | 2-Word Mismatch | 18.5 (sd=4.8) | **5%** | N/A |
| | First Word Mismatch | 18.6 (sd=4.2) | **10%** | N/A |
| | Last Word Mismatch | 20.0 (sd=9.3) | **5%** | N/A |
| | Middle Word Mismatch | 23.8 (sd=17.7) | 0% | N/A |
| Blinking Lights | Accepting Receving Device | **28.8 (sd=10.4)** | N/A | 0% |
| Seeing Is Believing | Accepting Receving Device | **26.9 (sd=7.5)** | N/A | **5%** |
| Audio/Visual (Beep-Blink) | Matching Patterns | **26.3 (sd=5.3)** | N/A | **5%** |
| | First Bit Mismatch | 28.9 (sd=14.0) | **20%** | N/A |
| | Last Bit Mismatch | 30.5 (sd=27.7) | 0% | N/A |
| | Middle Bit Mismatch | 27.6 (sd=13.5) | **5%** | N/A |
| | 4-Bit Mismatch | 31.7 (sd=17.4) | 0% | N/A |
| Audio/Visual (Blink-Blink) | Matching Patterns | **27.8 (sd=10.3)** | N/A | 0% |
| | 4-Bit Mismatch | 24.9 (sd=5.1) | **5%** | N/A |
| | First Bit Mismatch | 25.0 (sd=5.1) | **30%** | N/A |
| | Last Bit Mismatch | 23.2 (sd=3.3) | 0% | N/A |
| | Synchrony Bit Mismatch | 25.5 (sd=8.0) | **5%** | N/A |
| | Middle Bit Mismatch | 24.9 (sd=5.3) | **5%** | N/A |
| HAPADEP Variant | Accepting Receving Device | **10.8 (sd=2.6)** | N/A | **5%** |
| BEDA (Button-Button) | Normal Protocol Behaviour Until Pairing Is Successful | **31.9 (sd=32.1)** | N/A | N/A |

*Estimated Standard Deviation from the sample

**Fig. 2.** Summary of logged data.

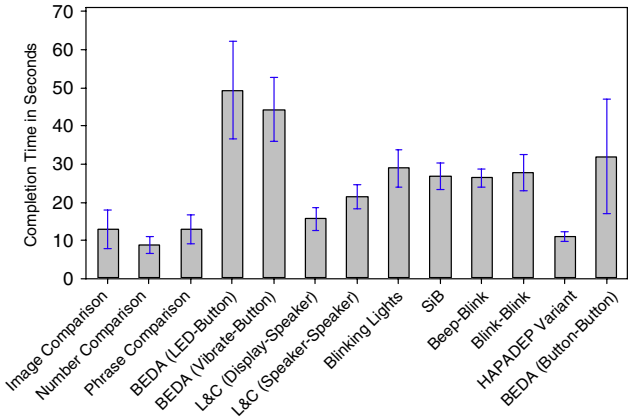


**Fig. 3.** Time-to-completion for successful pairing.

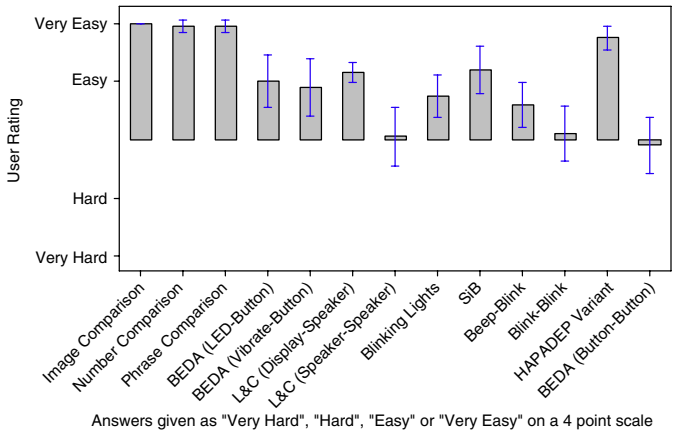


**Fig. 4.** Participant ease-of-use ratings.

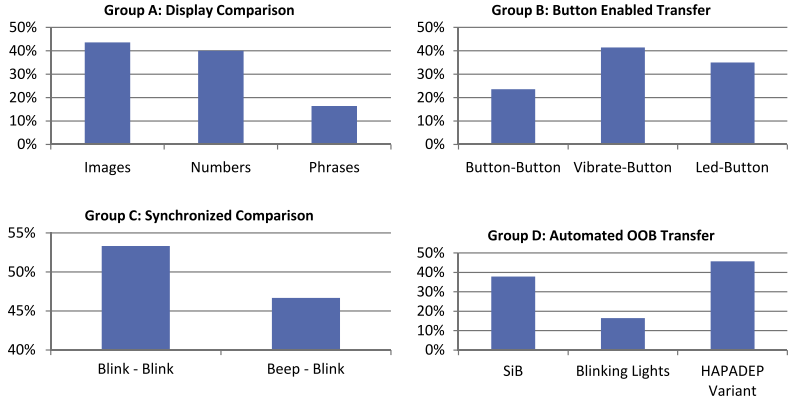


**Fig. 5.** Participant preferences.

In the pairwise chi-square tests for safe errors, we did not find any significant differences between methods. However, we encountered some weak evidence ($p = 0.07$) showing that Image Comparison leads to more safe errors than Number Comparison, Display–Speaker, Speaker–Speaker, Blinking–Lights, Blink–Blink and all BEDA variants.

### 6.2. Analysis of usability measures

#### 6.2.1. Completion timings

One natural way to compare tested methods is by looking at completion time under normal circumstances, i.e., when no errors occur (note that in most practical settings, attacks or faults will not occur). Based on the average results for successful pairing, as observed from Fig. 3, the fastest method is Visual Number Comparison at 8.6 s for a successful outcome—the most common scenario for methods that report low error rates. It is closely followed by the HAPADEP variant at 10.8 secs. Phrase and Image Comparison methods are next, each requiring 12.7 s. Finally, L&C Display–Speaker variant comes in at 15.5 s. The methods which were on the slower side include the rest, ranging from L&C Speaker–Speaker variant (21.3 s) to BEDA LED–Button variant, which takes 49.5 s on average.

We also conducted analysis of variance (ANOVA) and pairwise (paired one-tailed) $t$-tests between the completion times (under normal circumstances) of different methods. ANOVA revealed that the method has a significant effect on completion time and at the %5 significance level, we observed the following pairwise differences between methods:

- Number Comparison and HAPADEP Variant *are significantly faster than* LED–Button, Vibrate–Button, Blinking–Lights, SiB, Beep–Blink, Blink–Blink and Button–Button.
- Image Comparison and Phrase Comparison *are significantly faster than* LED–Button, Vibrate–Button, Blinking–Lights and Button–Button.
- Speaker–Display *is significantly faster than* LED–Button, Vibrate–Button and Button–Button.
- Beep–Blink, Blink–Blink, SiB, Blinking–Lights and Speaker–Speaker *are significantly faster than* LED–Button and Vibrate–Button.

- Button–Button *is significantly faster than* LED–Button

In our tests, the BEDA variants, SiB, Blinking–Lights and HAPADEP tests were each run twice. However, we did not find any significant difference in the completion timings between the two runs of these methods. In other words, we did not find any statistically significant effect of learning/training on these methods. Across different methods, wherever applicable, we also ran tests to evaluate the effect of the type of attack case (mismatch in first, last, middle bit or word) on completion time. However, we did not find any significant differences.

#### 6.2.2. Ease-of-use ratings

The graph in Fig. 4 also leads us to certain intuitive observations. Not surprisingly, image, number and phrase comparisons were ranked as easiest methods. All three are intuitive and perhaps even familiar to some users who have performed Bluetooth device pairings in the past. Note that, especially, number comparison is very close to the common Bluetooth method used to pair cell phones and laptops. Also, users might be familiar with images and pass-phrases used in two-factor web authentication. Audio pairing (HAPADEP) was probably ranked as nearly very easy (close to other comparison methods) not due to user familiarity but rather because of low level of user involvement.

Both SiB and L&C Speaker–Display were ranked as moderately easy, appreciably lower than the aforementioned easiest four. The former is surprising since SiB requires very little from the user in terms of memory or correlation—merely aligning the camera and snapping a photo. Perhaps the novelty of using a camera for security purposes had a somewhat startling effect. It could also be due to us having tested a simulation of SiB. In an actual SiB implementation, when the camera-equipped device detects a barcode (displayed by the sending device), it shows (on the viewfinder screen) a "yellow rectangle" drawn across the barcode. This serves as the cue for the user to capture the picture. We expect that a real SiB implementation would be better in terms of user experience than our mocked-up version.

Beep–Blink, L&C Speaker–Speaker and BEDA Button–Button methods are among the hardest to use. None of this is surprising in light of result of Fig. 2: Beep–Blink had a 20% error rate, L&C Speaker–Speaker – 5%–10%, whereas, BEDA Button–Button had among the slowest completion times in addition to being a rather unfamiliar technique – pressing buttons in synchrony is not difficult or challenging but it is certainly new to users. None of the methods were rated as very hard (notice the blank strip above and below the "Hard" mark in Fig. 4) but this may be because our young subjects were loath to admit that something was really difficult.

Analysis of the variance revealed that the method has a significant effect on ease-of-use rating. In pairwise (paired one-tailed) $t$-tests (at %95 confidence), the following significant differences are observed:

- Image Comparison *is significantly easier to use than* all LED–Button, Vibrate–Button, Display–Speaker, Speaker–Speaker, Blinking–Lights, Beep–Blink, Blink–Blink and Button–Button methods.
- Number Comparison and Phrase Comparison *are significantly easier to use than* LED–Button, Vibrate–Button, Speaker–Speaker, Blinking–Lights, Beep–Blink, Blink–Blink and Button–Button.
- HAPADEP Variant *is significantly easier to use than* Vibrate–Button, Speaker–Speaker, Beep–Blink, Blink–Blink and Button–Button.
- LED–Button, Display–Speaker and SiB *are significantly easier to use than* Speaker–Speaker, Blink–Blink and Button–Button.
- Vibrate–Button *is significantly easier to use than* both Speaker–Speaker and Button–Button.
- Blinking–Lights *is significantly easier to use compared to* Button–Button.

We also ran some statistical tests to figure out how correlated different users' ratings are with respect to methods that are inherently similar. For example, we wanted to see whether people who liked handling photo cameras, also liked working with video cameras. Indeed, for SiB and Blinking–Lights, we found a correlation between user ratings (cf $= 0.45$, $p = 0.05$). We also found correlation between BEDA variants: LED–Button and Vibrate–Button (cf $= 0.91$, $p \ll 0.01$), Button–Button and Vibrate–Button (cf $= 0.57$, $p = 0.01$), Button–Button and LED–Button (cf $= 0.53$, $p = 0.02$).

#### 6.2.3. Successful task completion

As explained in 6.1 for safe errors, there was no observed significant differences among the tested methods with respect to successful task completion. However, there is some statistical evidence to support lower successful completion rate for image comparison method.

#### 6.2.4. All usability measures taken together

A usable pairing method should perform well with respect to all (not just one of the) usability measures we discussed so far. To this end, we performed linear cross-correlations among the tested methods across the three usability measures, i.e., task completion time, ease-of-use and successful task completion. Table 2 shows the correlation coefficients and their respective statistical significance.

The coefficients from $-0.3$ to $-0.1$ and 0.1 to 0.3 are generally regarded as small [36] and in line with the findings of [37], we cannot regard any of our usability measures as sufficiently correlated with others that they could be justifiably omitted. On the other hand, since the measures *are* lowly correlated, it does make sense to also look at them as a whole and therefore perform a principal component analysis for all usability measures.

**Table 2**
Cross-correlation of usability measures.

| | Task completion time | Successful task completion |
|---|---|---|
| Successful task completion | 0.067<br>($p = 0.28$) | – |
| Ease-of-use rating | −0.219<br>($p \ll 0.01$) | −0.11<br>($p = 0.08$) |

**Table 3**
Principle components of usability measures.

| | PC1 | PC2 | PC3 |
|---|---|---|---|
| Eigenvalue | 1.2767 | 0.9473 | 0.7760 |
| Proportion of variance | 0.426 | 0.316 | 0.259 |
| Cumulative proportion | 0.426 | 0.741 | 1.000 |

**Table 4**
Factor loadings of PC1 and PC2.

| | PC1 | PC2 |
|---|---|---|
| Task completion time | 0.625 | −0.408 |
| Successful task completion | 0.416 | 0.895 |
| Ease-of-use rating | −0.661 | 0.178 |

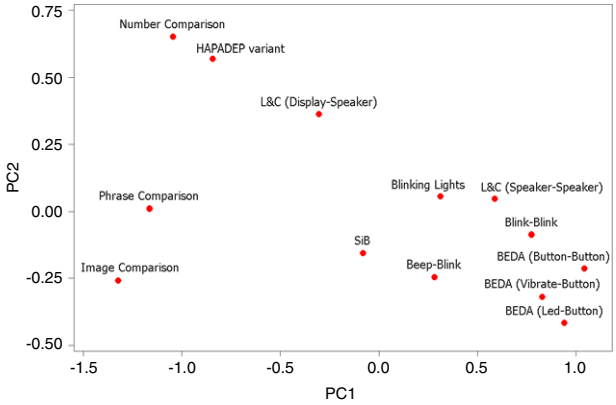


**Fig. 6.** Score plot of methods by PC1 and PC2.

**Principal component and cluster analysis:** Table 3 lists the three principal components, denoted PC1, PC2 and PC3, that explain 100% of the variance in the logged data. As the first two components together explain nearly 75% of the variance, PC3 is disregarded in the following evaluations. Table 4 shows the factor loadings of PC1 and PC2. As shown, PC1 factors in completion time and usability rating more in comparison to PC2, while PC2 has a higher loading of successful task completion (i.e., low safe errors).

Fig. 6 depicts how each method scores with respect to PC1 and PC2. Due to the respective loads of task performance time and usability ratings in PC1 and PC2, it is reasonable to regard the methods located towards the upper left of the graph as "good" and methods towards the right bottom as "bad" in terms of usability.[15] Fig. 7 shows how methods form three clusters, based on their usability measures. The two figures indicate that our methods can be partitioned into two sets, with good and poor usability overall. Methods with good usability (green and blue colors in Fig. 7) include Number Comparison, HAPADEP, L&C Display–Speaker, Image Comparison and Phrase Comparison. The rest (highlighted red in 7) exhibit poor overall usability. Among "good" methods, Number Comparison, HAPADEP and L&C Display–Speaker (highlighted green) are better overall compared to Image Comparison and Phrase Comparison (blue). This is mainly because Image and Phrase Comparison are prone to safe errors, which lowers the rate of successful task completion. Looking further down into "green" methods, Number Comparison and HAPADEP are better than L&C Display–Speaker, which can be attributed to the latter's relatively slow speed and lower ease-of-use ratings. Among "red" methods, BEDA button-button variant performs the worst (note PC2 is not relevant while comparing BEDA methods); BEDA LED–Button and Vibrate–Button variants are marginally better, followed by Blink–Blink, then Blinking–Lights and L&C Speaker–Speaker, and finally Beep–Blink and SiB.

[15] Note that this categorization is only based on the three usability measures and does not take into account the robustness measures, such as, fatal errors.

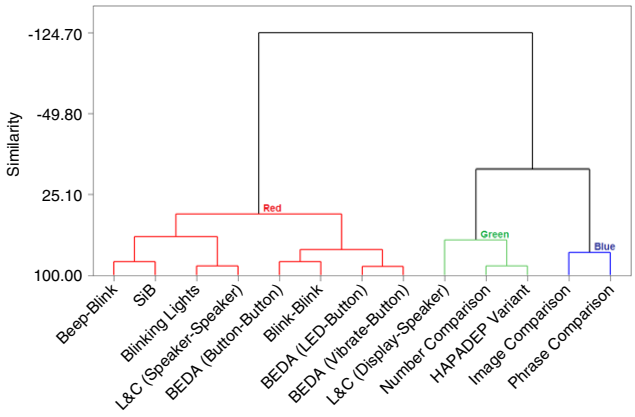


**Fig. 7.** Result of cluster analysis based on principal components.

*6.2.5. Users' personal preferences on method groups*

We also sought users' individual preferences. The 4-part graph in Fig. 5 shows user preferences for method groups.

**Group A:** all comparison-based methods needing displays on both devices. Although number comparison was the fastest, image comparison is actually slightly preferable, although both hover around 40%. Phrase comparison is trailing by a wide margin at 20% and is clearly not well-liked. Combining these results with our analysis in previous subsection, makes it clear that phrase comparison should not be used at all. Image comparison can be ruled out in favor of numeric comparison, as the former exhibits high safe error and poor task completion rates.

**Group B:** BEDA variants suitable for scenarios where at least one device is interface-challenged. It is not surprising that Vibrate–Button is the preferred choice since it involves less user burden: vibration is hard to ignore or miss, and it is easier to coordinate vibration with button presses than to press two buttons simultaneously or to react to a blinking LED. User preferences are also consistent with our analysis in previous subsection, as shown in Fig. 6 (note PC2 is not relevant for Vibrate–Button and LED–Button, as they did not encounter any safe errors in our tests).

**Group C:** synchronized Beep–Blink and Blink–Blink methods. This is similar to Group A since manual comparison is required in both. However, Group C is geared for display-less devices with LEDs and/or beepers. Since these methods are not fast, have relatively high error rates, and are considered hard by users, we do not recommend them at this time.

**Group D:** automated methods that impose light user burden: SiB, Blinking–Lights and HAPADEP. Similar to our principle component and cluster analysis, this chart shows that Blinking–Lights is least preferred, probably due to its longer time-to-completion (more than twice that of HAPADEP) and more tedious operation (taking a video clip is more involved than snapping one photo in SiB.) HAPADEP exhibits slightly higher (5%–6%) user preference over SiB, most likely because of faster run-time. not group SiB with other In summary, we do not recommend Blinking–Lights. Whereas, HAPADEP variant is well-suited for most scenarios, except noisy environments and whenever one device has neither a speaker nor a microphone.

### *6.3. Summary of results*

Based on our analysis of robustness and usability measures, and user preferences, we can make the following conclusions.

- Number comparison is an overall winner with respect to our usability measures. Compared to image and phrase comparison, it is much less likely to encounter safe errors. Note that safe errors imply that the user needs to repeat the pairing process, which can lead to annoyance and, in turn, increase the likelihood of fatal errors. Number comparison can be used on any devices capable of displaying few numeric digits. The fatal error rate in number comparison was low (10% in first digit mismatch and 5% in middle digit mismatch) which would provide a reasonable level of security. In scenarios where security has a high priority and where one of the devices has a speaker, L&C Display–Speaker variant might be a better choice, since it exhibited only a 5% fatal error rate in test-cases involving first word mismatches. Recall that L&C Display–Speaker was slightly less user-friendly (mostly because of its relatively slower speed) with respect to number comparison.
- Audio pairing (HAPADEP) is the preferred method whenever one device has a speaker but lacks a display, and the other device has a microphone, e.g., a Bluetooth headset and a cell phone or a small display-less MP3 player and a laptop. Whenever applicable, HAPADEP might be a preferred choice over Number Comparison, since it is resistant to rushing user behavior.
- For interface-constrained devices (e.g., pagers, headsets and access points), BEDA Vibrate–Button is the best choice, followed by LED–Button and Button–Button. These are trailed by Blink–Blink and Beep–Blink, both of which have (high) fatal and safe errors; albeit, they are faster than all BEDA variants. (Recall that BEDA variants are all resistant to rushing

user behavior.) L&C Speaker–Speaker showed slightly better usability compared to BEDA methods but yielded some fatal errors (10%). It is thus only advised for scenarios where two devices only have speakers and input buttons. However, if security trumps usability, BEDA-Beep button might be a better choice.
- Due to relatively poor usability of SiB and Blinking–Lights, HAPADEP variant would be preferred as long as the environment is not too noisy. If one device has a camera – as required in SiB and Blinking–Lights – it is also very likely to have a microphone.[16] Given a choice between SiB and Blinking–Lights, assuming that the sending device has a decent display, the former is preferable due to its better usability. However, Blinking–Lights might be better if the sending device does not have a display of sufficient quality.

## 7. Conclusions and future work

We presented the first experimental evaluation of prominent device pairing methods. We pursued a thorough analysis of these methods in terms of both security and usability. Our analysis provides clear insights and mathematical evidence as to how secure and usable tested methods are, and how they compare with one another.

Results show that simple Number Comparison is quite attractive overall, being both fast and secure as well as readily acceptable by users. It naturally suits settings where devices have appropriate-quality displays. HAPADEP variant seems to be preferable in scenarios where one device has a speaker and the other—a microphone: it is fast, error-free and requires very little user intervention. Vibrate–Button or LED–Button are best-suited for devices lacking screens, speakers and microphones.

We believe that this is an important and timely first step in exploring real-world usability of secure device pairing methods. Items for future work include:

- Experiments with more diverse participant pools.
- Experiments involving different (more diverse) devices.
- Simulation and testing of attack scenarios for automated methods (Group D of Section 6.2.5).
- Gathering and analysis of user-input regarding perceived security of various methods.
- Obtaining comprehensive user evaluation for each method.

## Acknowledgements

This work was supported by NSF Awards CT-ISG 0831397 and 0831526. We thank Nokia for providing the two cell phones used in the study. We are also very grateful to Kari Kostiainen (NRC) and Jonathan McCune (CMU) for helping us with the Blinking–Lights and Seeing-is-Believing methods, respectively. Finally, we thank N. Asokan (NRC) for his valuable feedback and Claudio Soriente (UCI) for his work on the HAPADEP prototype.

## References

[1] J. Suomalainen, J. Valkonen, N. Asokan, Security associations in personal networks: A comparative analysis, in: F. Stajano, C. Meadows, S. Capkun, T. Moore (Eds.), in: Security and Privacy in Ad-hoc and Sensor Networks Workshop, ESAS, vol. 4572, 2007, pp. 43–57.
[2] M. Jakobsson, S. Wetzel, Security weaknesses in bluetooth, in: The Cryptographer's Track at RSA, CT-RSA, 2001.
[3] C. Soriente, G. Tsudik, E. Uzun, BEDA: Button-Enabled Device Association, in: International Workshop on Security for Spontaneous Interaction IWSSI, UbiComp Workshop Proceedings, 2007.
[4] A. Kumar, N. Saxena, G. Tsudik, E. Uzun, Caveat emptor: A comparative study of secure device pairing methods, in: IEEE International Conference on Pervasive Computing and Communications, PerCom, 2009.
[5] D. Balfanz, et al., Talking to strangers: Authentication in ad-hoc wireless networks, in: Network and Distributed System Security Symposium, NDSS, 2002.
[6] C. Gehrmann, C.J. Mitchell, K. Nyberg, Manual authentication for wireless devices, RSA CryptoBytes 7 (1) (2004).
[7] S. Vaudenay, Secure communications over insecure channels based on short authenticated strings, in: Advances in Cryptology-CRYPTO, 2005.
[8] M. Čagalj, S. Čapkun, J.-P. Hubaux, Key agreement in peer-to-peer wireless networks, Proceedings of the IEEE 94 (2) (2006) 467–478.
[9] S. Laur, K. Nyberg, Efficient mutual data authentication using manually authenticated strings, in: International Conference on Cryptology and Network Security, CANS, vol. 4301, Springer, 2006, pp. 90–107.
[10] S. Laur, K. Nyberg, Efficient mutual data authentication using manually authenticated strings, in: CANS, 2006, pp.90–107.
[11] S. Pasini, S. Vaudenay, SAS-based authenticated key agreement, in: Public key cryptography, PKC, 2006.
[12] N. Saxena, et al. Extended abstract: Secure device pairing based on a visual channel, in: IEEE Symposium on Security and Privacy, 2006.
[13] N. Saxena, M.B. Uddin, Automated device pairing for asymmetric pairing scenarios, in: Information and Communications Security, ICICS, 2008, pp. 311–327.
[14] V. Boyko, P. MacKenzie, S. Patel, Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman, in: Advances in Cryptology-Eurocrypt, Springer, 2000, pp. 156–171.
[15] F. Stajano, R.J. Anderson, The resurrecting duckling: Security issues for ad-hoc wireless networks, in: Security Protocols Workshop, 1999.
[16] D. Balfanz, et al. Network-in-a-box: How to set up a secure wireless network in under a minute, in: USENIX Security, 2004, pp. 207–222.
[17] I. Goldberg, Visual Key Fingerprint Code, http://www.cs.berkeley.edu/iang/visprint.c, 1996.
[18] A. Perrig, D. Song, Hash visualization: A new technique to improve real-world security, in: International Workshop on Cryptographic Techniques and E-Commerce, 1999.
[19] C.M. Ellison, S. Dohrmann, Public-key support for group collaboration, ACM Transactions on Information and System Security (TISSEC) 6 (4) (2003) 547–565.

16 Except for some old digital photo cameras that lack video recording ability; however, those also tend to lack any means of wireless communication.

[20] V. Roth, et al. Simple and effective defense against evil twin access points, in: ACM Conference on Wireless Network Security, WISEC, 2008, pp. 220–235.
[21] J.M. McCune, A. Perrig, M.K. Reiter, Seeing-is-believing: Using camera phones for human-verifiable authentication, in: IEEE Symposium on Security and Privacy, 2005.
[22] R. Prasad, N. Saxena, Efficient device pairing using "human-comparable" synchronized audiovisual patterns, in: Conference on Applied Cryptography and Network Security, ACNS, 2008.
[23] M. Goodrich, et al. Loud and clear: Human-verifiable authentication based on audio, in: IEEE International Conference on Distributed Computing Systems, ICDCS, 2006.
[24] C. Soriente, G. Tsudik, E. Uzun, HAPADEP: Human-assisted pure audio device pairing, in: Information Security, 2008, pp. 385–400.
[25] M. Goodrich, et al., Audio-based secure device pairing, International Journal of Security and Networks 4 (2009).
[26] E. Uzun, K. Karvonen, N. Asokan, Usability analysis of secure pairing methods, in: International Workshop on Usable Security, USEC, 2007.
[27] T. Kindberg, K. Zhang, Validating and securing spontaneous associations between wireless devices, in: Information Security Conference, ISC, 2003, pp. 44–53.
[28] R. Mayrhofer, M. Welch, A human-verifiable authentication protocol using visible laser light, in: International Conference on Availability, Reliability and Security (ARES), IEEE Computer Society Washington, DC, USA, 2007, pp. 1143–1148.
[29] L.E. Holmquist, et al., Smart-its friends: A technique for users to easily establish connections between smart artefacts, in: Ubiquitous Computing, UbiComp, Springer-Verlag, London, UK, 2001, pp. 116–122.
[30] R. Mayrhofer, H. Gellersen, Shake well before use: Authentication based on accelerometer data, in: Pervasive Computing, PERVASIVE, Springer, 2007.
[31] R. Dhamija, J.D. Tygar, M.A. Hearst, Why phishing works, in: International Conference for Human–Computer Interaction, CHI, 2006.
[32] K. Kostiainen, E. Uzun, Framework for comparative usability testing of distributed applications, in: Security User Studies: Methodologies and Best Practices Workshop, 2007.
[33] J.M. McCune, Personal Communication, Mar. 2008.
[34] K. Kostiainen, Personal Communication, Mar. 2008.
[35] L. Faulkner, Beyond the five-user assumption: Benefits of increased sample sizes in usability testing, Behavior Research Methods, Instruments, & Computers 35 (3) (2003) 379–383.
[36] J. Cohen, P. Cohen, S.G. West, L.S. Aiken, Applied multiple regression/correlation analysis for the behavioral sciences, Lawrence Erlbaum Associates, Hillsdale, NJ, 1983.
[37] E. Frkjr, M. Hertzum, K. Hornbk, Measuring usability: Are effectiveness, efficiency, and satisfaction really correlated? CHI '00: Proceedings of the SIGCHI Conference on Human Factors in Computing Systems, 2000 pp. 345–352, http://doi.acm.org/10.1145/332040.332455.