Caveat Emptor:

A Comparative Study of Secure Device Pairing Methods

Arun Kumar and Nitesh Saxena
Computer Science and Engineering Department
Polytechnic Institute of New York Univesity
aashok01@students.poly.edu, nsaxena@poly.edu

Gene Tsudik and Ersin Uzun*
Computer Science Department
University of California, Irvine
{gts,euzun}@ics.uci.edu

***Abstract*—"Secure Device Pairing" is the process of bootstrapping a secure channel between two previously unassociated devices over a (usually wireless) human-imperceptible communication channel. Lack of prior security context and common trust infrastructure open the door for *Man-in-the-Middle* (also known as *Evil Twin*) attacks. Mitigation of these attacks requires user involvement in the device pairing process. Prior research yielded a number of interesting methods utilizing various auxiliary human-perceptible channels, e.g., visual, acoustic or tactile. These methods engage the user in authenticating information exchanged over human-imperceptible channels, thus mitigating MiTM attacks and forming the basis for secure pairing.**

We present the first comprehensive comparative evaluation of notable secure device pairing methods. Our results identify methods best-suited for a given combination of devices and human abilities. This work is both important and timely, since it sheds light on usability in one of the very few settings where a wide range of users (not just specialists) are confronted with security techniques.

I. Introduction

Medium- and short-range wireless communication – based on technologies such as Bluetooth, WiFi, Zigbee and WUSB – is increasingly popular and promises to remain so in the future. At the same time, 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 current everyday usage scenarios where two devices need to "work together", e.g., a Bluetooth headset and a cellphone, a PDA and a wireless printer, or a wireless access point and a laptop.

The surge in popularity of wireless devices brings about various security risks. The wireless communication channel is easy to eavesdrop upon and to manipulate, raising the very real threats, notably, of so-called *Man-in-the-Middle* (MiTM) or *Evil Twin* attacks. Therefore, it is important to secure this channel. However, secure communication must be first 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).

One of the main challenges in secure device pairing is that, due to sheer 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 (such as authenticated key exchange protocols) are unsuitable, since unfamiliar devices have no prior security context and no common point of trust. Moreover, the use of a common wireless channel is insufficient to establish a secure context, since such channels are not perceivable by the human user. The research community has already recognized that some form of human involvement is necessary to address the problem of secure device pairing.[1]

One valuable and established research direction is the use of auxiliary – also referred to as "out-of-band" (OOB) – channels, which are both perceivable and manageable by the human 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 can not 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 is very important. Moreover, 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 (reviewed in Section II) typically require transmitting few bits (e.g., 15) over an OOB channel to achieve reasonable security. However, many devices (e.g., Bluetooth headsets and wireless access points) have limited hardware facilities and/or user interfaces, making it challenging to communicate even a few bits and, in turn, complicating 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

* Corresponding author

[1]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].

978-1-4244-3304-9/09/$25.00 ©2009 IEEE

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 human 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 such a study. First, usability of current device pairing methods remains very unclear. Even methods that have been usability-tested (e.g., [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 an average non-specialist user is often initially clueless about manipulating new devices. A more important issue is that an average 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 focus of our study is on most common pairing scenarios where a single user controls both the devices. The study yields some interesting results which help us identify most appropriate method(s) for a given combination of devices. Although this paper is less technical in nature than traditional security and applied cryptography research, we believe that the topic is very important since it sheds light on usability in one of the few 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.

II. Background

In this section, we describe notable relevant cryptographic protocols and pairing methods. The term *cryptographic protocol* denotes the entire interaction involved, and information exchanged, in the course of pairing. The term *pairing method* refers 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.

A. Cryptographic Protocols

Cryptographic protocols reviewed below are included mainly for the sake of completeness. This section can be skipped without any loss of continuity.

One simple protocol was suggested in [4]: 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 [5] 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.

[6] 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. Subsequent work ([7] and [8]) developed 3-round SAS protocols. Recently, [9], [10] proposed a more efficient SAS protocol which is utilized in a number of pairing methods we tested.

B. 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" [11] 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" [4], 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 [12]. 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.[3] The main drawback is that IR has been largely displaced by other wireless technologies (e.g., Bluetooth) and is available on 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" [13], "Random Arts Visual Hash" [14] and "Colorful Flag" [15]. Such methods, however, require both devices to have displays with sufficiently high resolution. Applicability

[3]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.

is therefore limited to high-end devices, such as: laptops, PDAs and certain cell phones. These methods are based on the protocol proposed in [4] which was reviewed earlier. A more practical approach, based on SAS protocols [8], [7], suitable for simpler displays and LEDs has been investigated in [16].

More recent work [17] 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.[4] We use the SiB variant from [9], [10] which only requires one device to have a camera.

A related approach, called "Blinking Lights" has been explored in [9]. 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 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, [18] 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) [19]. 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 [19], L&C is based on the protocol of [4]. In this paper, to reduce the number of words in the MadLib sentences, we use the L&C variant based on SAS protocols [8], [7].

Some follow-on work (HAPADEP [20], [21]) 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 [22] 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 [22] 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 average user 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 [9], 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., [23]) 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 involving technologies that are more exotic, i.e., relatively expensive and uncommon. We briefly summarize a few. [24] suggested using ultrasound as the OOB channel. A related technique uses laser as the OOB and requires each device to have a laser transceiver [25]. A very different OOB channel was considered in "Smart-Its-Friends" [26]: 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" [27]. 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 rare on other devices and physically shaking/twirling is unsuitable for many sensitive as well as stationary and large/bulky devices.

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

- *Sending Device / Receiving Device* – applies to all

[4]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.

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.

III. Study Preliminaries

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

A. 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 usability-tested 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) [12], Compare-and-Confirm, Copy-and-Confirm and Choose-and-Enter [22], Blink-Blink and Beep-Blink [18], as well as four variants of BEDA [3].

There are about twenty methods, counting variations, in *Figure 1*. Common sense suggests that it is very difficult, and perhaps futile, to find a stable set of average users and ask them to test all methods, hoping that results will yield comparative usability metrics. The main obstacle is user fatigue, since twenty methods can not 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, exotic or deprecated based on prior evaluations. We eliminated the following methods from our study (the corresponding cells are white in the first column of *Figure 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 *Figure 1*, have been included in our study. Their inclusion was primarily based upon applicability to a broad and common set of devices.

B. What Devices to Use?

For the entire study, we used two Nokia cellphones models[5]: N73 and E61, as test devices. Both models have been released two years ago (in 2006) and hence do not represent the *cutting edge*. This was done on purpose in order to avoid devices with exotic or expensive features as well as processors faster than those commonly available at present.

Another reason for choosing these devices is the plethora of *common* interfaces available on them. 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 tests, Bluetooth was used as the wireless (human-imperceptible) channel. We consider this choice to be natural since Bluetooth is widely available and inexpensive. It also allows positioning flexibility within reasonable physical space (unlike IR), i.e., $1 - 15$ feet suffices for most settings.

For methods that involve beeping, the phone speaker is trivial to use as 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.[6]

C. 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.

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

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

	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 and a cable		Connect cable to devices	**NONE**	**NONE**	Cable
Talking to Strangers	IR port on both		Find, activate, 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	**NONE**	Align camera on receiving device with displayed barcode on sending device, take picture	Abort or accept on sending device based on receiving device decision	Visual
Blinking Lights	LED + user-input	User-output + Light detector or video camera	**NONE**	Initiate transmittal of OOB data by sending device, align camera or light detector on receiving device.	Abort or accept on sending device based on receiving device decision	Visual
Loud & Clear ▪Display-Speaker ▪Speaker-Speaker	User-input on both + ▪display on one & speaker on other, or ▪speaker on both		**NONE**	Compare: two vocalizations, or display with vocalization	Abort or accept on both devices	▪Audio, or ▪audio + visual
Button-Enabled (BEDA) ▪Vibrate-Button ▪LED-Button ▪Beep-Button	User input + ▪vibration ▪LED ▪beeper	User output + 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 ▪Visual + tactile ▪Audio + tactile
Button-Enabled (BEDA) ▪Button-Button	One button on both + user-output on one		Touch or hold both devices	Simultaneously press and release buttons on both devices 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)	Speaker + user-input	Microphone + user-output	**NONE**	Wait for signal from receiving device.	Abort or accept on sending device	Audio
Audio/Visual Synch. ▪Beep-Beep ▪Blink-Blink ▪Blink-Beep	User-input on both + ▪Beeper on each ▪LED on each ▪Beeper on one & LED on other		**NONE**	Monitor synchronized: ▪beeping, or ▪blinking, or ▪beeping & blinking	Abort on both devices if no synchrony	▪Visual ▪ Audio ▪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

Fig. 1. Feature Summary of Notable Device Pairing Methods

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 standalone 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. [28]. 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. (The term *attack* is limited to MiTM/Evil-Twin attacks in this context).

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 [6]. 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 model 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.[7]

[7]The current CMU implementation of SiB is supported on Nokia models N70 [29] and 6620 [17] as receiving devices. The current Nokia implementation of Blinking-Lights is supported only on Nokia 6630 [30] 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.

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.

IV. Usability Testing Details

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

1) Efficiency: time it takes to complete each method
2) 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 [22], we will refer to the errors in the former category as *safe errors* and the latter as *fatal errors*.
3) Usability: how each method fares in terms of user burden (i.e., ease-of-use perception) and personal preference.

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 influence test results. To remedy the problem, we pursued the study in three batches. Table I 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).

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

TABLE I
Breakdown of Tested Pairing Methods

A. Study Participants

We recruited 20 participants[8] for our three-batch study which lasted over one 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[9] group. As mentioned earlier, we claim that, if highly-motivated and technology-savvy young people do not react well to a given method, the same method will perform a lot worse with average users. On the other hand, a method that fares well with our participants, might not perform equally well with average 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 user demographics and *post-test* – for user 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.

B. Test Cases

Based on the type of each tested method, we created several test-cases simulating normal as well as abnormal 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, 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

[8]It is well-known that a usability study performed by 20 participants captures over 98% of usability related problems [31].

[9]All participants were regular computer users with at least one wireless personal 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 the 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 (respectively) captured with the help of the user, and always accepts it as legitimate. Our purpose was to mainly 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.[10]

C. 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 set-up was more-or-less automated and only a minimal involvement from the test administrator was required.

After giving a brief overview of our study goals (prior to the first batch of study), 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 suffer(ed) from any visual or hearing impairments, or have any 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 III-C, 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 their 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.

D. Test Results

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

[10]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.

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

In 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 Figure 4. Participants were also asked to order methods from most to least preferred, within each category, as reflected in Figure 5 and discussed in Section V-C below.

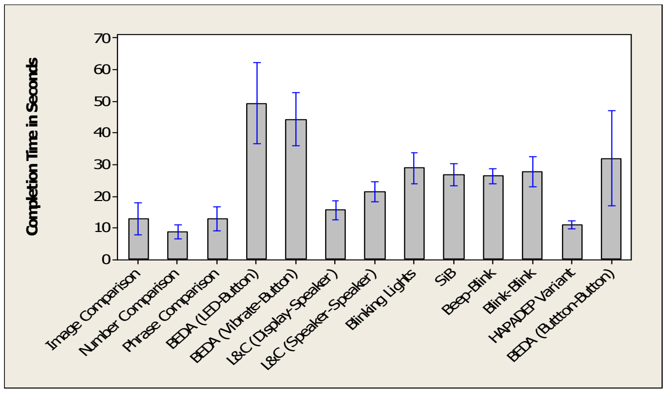

Fig. 2. Time-to-Completion for Successful Pairing

V. Interpreting Results

In this section we attempt to interpret the results of our study. We first consider various mechanical data, i.e., time to completion and error rates. We then analyze the perceived ease-of-use statistics and user preference ratings.

A. Interpreting Time and Error Results

Our results reflected in *Figure 3* and Figure 2 prompt a number of observations.

Successful Execution: One way to interpret the results 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 this performance metrics, tested methods fall into two speed categories: fast and slow. The fastest method is Visual Number Comparison at 8.6 secs 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 secs. Finally, L&C Display-Speaker variant comes in at 15.5 secs. The slow category includes the rest, ranging from L&C Speaker-Speaker variant (21.3 secs) to BEDA LED-Button variant which takes at a whooping 49.5 secs.

Looking at error rates, 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

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. 3. Summary of Logged Data

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.[11] Image Comparison yields a 15% rate but for "false positive" errors which are considered safe. Somewhat less error-prone are Visual Number Comparison and L&C Speaker-Speaker, each with a 10% rate. The BEDA variants LED-Button and Vibrate-Button are completely error-free, which implies that users can accurately transfer a one-bit result indicating success or failure, from one device to the other. (This was a concern for Blinking-Lights in [9].).

Fortunately, it seems that fastest methods are also relatively error-free. Taking both speed and error-rate into account, the overall best method is Visual Number Comparison, followed by HAPADEP variant and L&C Display-Speaker. Although Visual Phrase and Image Comparison methods are also in the same speed category, there is little motivation for using them. The reason is simple: since both devices must have displays for comparing numbers, phrases or images, it makes more sense to use Number Comparison (over Phrase and Image counterparts) because phrases take more display space than numbers and images require higher-resolution displays. Thus, for pairing scenarios where both devices have displays, Number Comparison is a clear winner. Using similar reasoning, HAPADEP variant appears to be the best choice for devices without displays, but at least a microphone on one and a speaker on the other. Whereas, for scenarios with a display on one device and a speaker (and no display) on the other, L&C Display-Speaker variant seems like the best choice.

Considering Errors: From a more conservative perspective, i.e, considering possible attacks and faults, results can be interpreted differently. In this case, we focus on completion times over all normal and failed cases. Also, even low fatal error rates might not be acceptable. Based on this criterion, if both devices have good-quality displays Image Comparison is best, since it exhibits no fatal errors, few safe errors and its completion time is short. Even when safe errors occur, the only drawback is user having to re-run the pairing method. The next best is L&C Speaker-Display, which has a low fatal error rate, no safe errors and short completion time. Phrase Comparison and Numeric Comparison exhibit similar characteristics with slightly higher fatal error rates.

For pairing scenarios where at least one device does not have a display capable of rendering images, L&C Speaker-Display is the first choice, followed by Phrase and Numeric Comparison, respectively (for the same reasons cited above).

The HAPADEP variant performs well with only few safe errors and short completion time. It thus seems to be an appropriate solution in noise-free scenarios where one device has a microphone and the other – a speaker.

For interface-constrained devices (e.g., pagers, headsets and access points), BEDA Button-Button is the best choice (it is fatal error-free), followed by Vibrate-Button and Led-Button. These are trailed by Blink-Blink and Beep-Blink, both of which have fatal and safe errors; albeit, they are faster than all BEDA variants.

Based on relatively slow completion times for SiB and Blinking Lights, the HAPADEP variant, which is faster, 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.[12] Given a choice between SiB and Blinking Lights, assuming that the sending device has a decent display, the former is preferable due to its speed. However, Blinking Lights is better-suited if the sending device does not have a good quality display.

Overall, if security critical error avoidance has a higher priority than time-to-completion, BEDA variants are the best choice.

[11]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.

[12]Except for some old digital photo cameras that lack video clip recording ability; however, those also lack any means of wireless communication.

B. Interpreting Ease-of-Use

The graph in Figure 4 also leads us to certain more-or-less obvious 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 cellphones 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.

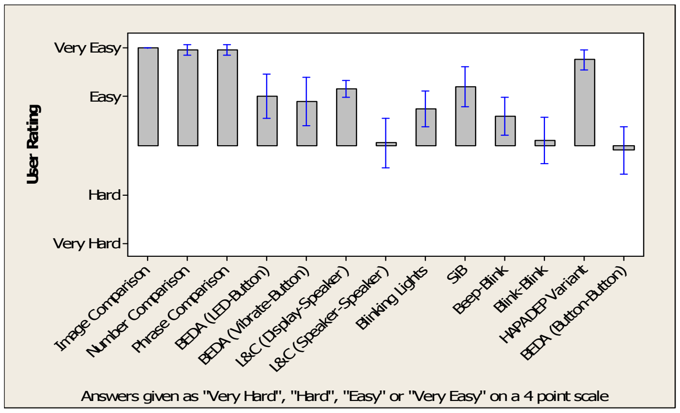

Fig. 4. Participant Ease-of-Use Ratings

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 *Figure 3*: 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 Figure 4) but this may be because our young subjects were loath to admit that something was really difficult.

Some conclusions from our ease-of-use results are as follows:

• Number, Image and Phrase Comparison methods are uniformly preferred, and thus recommended for use, for scenarios with two display-equipped devices. Of the three, Number Comparison represents the lowest common denominator, as it requires the least display quality.

• Audio pairing (HAPADEP) is the preferred choice whenever at least one device lacks a display but one has a speaker, and the other – a microphone, e.g., a Bluetooth headset and a cellphone or a small display-less MP3 player and a laptop.

• Blink-Blink, L&C Speaker-Speaker, and BEDA Button-Button are generally unsuitable for the average user due to low ease-of-use ratings by our technology-aware participants. However, we concede that there are scenarios where one of these methods is the only choice. For example, if one of the devices has no input means aside from a few buttons (e.g., a garage opener or a car key-fob), Button-Button or Blink-Blink might be the only viable pairing options.

• Every other method (SiB, L&C Display-Speaker, Blinking Lights, Beep-Blink, Vibrate-Button and LED-Button) might have a *niche* applications where it is either the best or the only option. (We defer these issues to future work.)

C. Interpreting User Preferences

The 4-part graph in Figure 5 shows user preferences for method groupings.

Group A: includes 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 time-to-completion, error rates, and ease-of-use data, makes it clear that phrase comparison should not be used at all. The choice between image and number comparison should depend on the display quality.

Group B: includes 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.

Group C: includes 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: includes automated methods that impose light user burden: SiB, Blinking Lights and HAPADEP. The 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.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

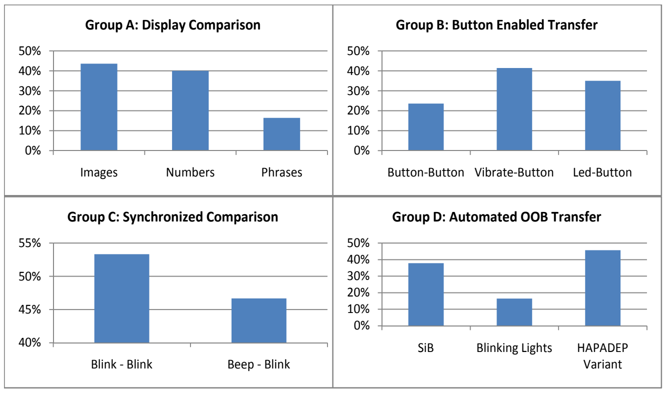

Fig. 5. Participant Preferences

VI. Conclusions and Future Work

We presented the first experimental evaluation of prominent device pairing methods. Results show that some simple methods (e.g., Visual Number and Image Comparison) are quite attractive overall, being both fast and secure as well as acceptable by users. They naturally appeal to settings where devices have appropriate-quality displays. HAPADEP variant seems to be preferable for more constrained devices: it is fast, error-free and requires very little user intervention. LED-Button or Vibrate-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 V-C).
- Gathering user input about perceived security of methods.
- Obtaining comprehensive user evaluation for each method.

Acknowledgements

We thank Nokia for providing the cellphones 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 HAPADEP prototype.

References

[1] J. Suomalainen, J. Valkonen, and N. Asokan, "Security Associations in Personal Networks: A Comparative Analysis," in *Security and Privacy in Ad-hoc and Sensor Networks Workshop (ESAS)*, F. Stajano, C. Meadows, S. Capkun, and T. Moore, Eds., no. 4572, 2007, pp. 43–57.
[2] M. Jakobsson and S. Wetzel, "Security weaknesses in bluetooth," in *The Cryptographer's Track at RSA (CT-RSA)*, 2001.
[3] C. Soriente, G. Tsudik, and E. Uzun, "BEDA: Button-Enabled Device Association," in *International Workshop on Security for Spontaneous Interaction (IWSSI), UbiComp Workshop Proceedings*, 2007.
[4] D. Balfanz *et al.*, "Talking to strangers: Authentication in ad-hoc wireless networks," in *Network and Distributed System Security Symposium (NDSS)*, 2002.
[5] C. Gehrmann, C. J. Mitchell, and K. Nyberg, "Manual authentication for wireless devices," *RSA CryptoBytes*, vol. 7, no. 1, 2004.
[6] S. Vaudenay, "Secure communications over insecure channels based on short authenticated strings," in *Advances in Cryptology-CRYPTO*, 2005.
[7] S. Laur and 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.
[8] S. Pasini and S. Vaudenay, "SAS-Based Authenticated Key Agreement," in *Public key cryptography (PKC)*, 2006.
[9] N. Saxena *et al.*, "Extended abstract: Secure device pairing based on a visual channel," in *IEEE Symposium on Security and Privacy*, 2006.
[10] N. Saxena and M. B. Uddin, "Automated device pairing for asymmetric pairing scenarios," in *Information and Communications Security (ICICS)*, 2008, pp. 311–327.
[11] F. Stajano and R. J. Anderson, "The resurrecting duckling: Security issues for ad-hoc wireless networks," in *Security Protocols Workshop*, 1999.
[12] 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.
[13] I. Goldberg, "Visual Key Fingerprint Code," 1996.
[14] A. Perrig and D. Song, "Hash visualization: a new technique to improve real-world security," in *International Workshop on Cryptographic Techniques and E-Commerce*, 1999.
[15] C. M. Ellison and S. Dohrmann, "Public-key support for group collaboration," *ACM Transactions on Information and System Security (TISSEC)*, vol. 6, no. 4, pp. 547–565, 2003.
[16] V. Roth *et al.*, "Simple and effective defense against evil twin access points," in *ACM conference on Wireless network security (WISEC)*, 2008.
[17] 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.
[18] R. Prasad and N. Saxena, "Efficient device pairing using "human-comparable" synchronized audiovisual patterns," in *Conference on Applied Cryptography and Network Security (ACNS)*, June 2008.
[19] M. Goodrich *et al.*, "Loud and Clear: Human-Verifiable Authentication Based on Audio," in *IEEE International Conference on Distributed Computing Systems (ICDCS)*, 2006.
[20] C. Soriente, G. Tsudik, and E. Uzun, "HAPADEP: human-assisted pure audio device pairing," in *Information Security*, 2008, pp. 385–400.
[21] M. Goodrich *et al.*, "Audio-based secure device pairing," in *International Journal of Security and Networks (IJSN)*, vol. 4, 2009.
[22] E. Uzun, K. Karvonen, and N. Asokan, "Usability analysis of secure pairing methods," in *International Workshop on Usable Security (USEC)*, 2007.
[23] V. Boyko, P. MacKenzie, and S. Patel, "Provably Secure Password-Authenticated Key Exchange Using Diffie-Hellman," in *Advances in Cryptology-Eurocrypt*. Springer, 2000, pp. 156–171.
[24] T. Kindberg and K. Zhang, "Validating and securing spontaneous associations between wireless devices," in *Information Security Conference (ISC)*, 2003, pp. 44–53.
[25] R. Mayrhofer and 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.
[26] L. E. Holmquist *et al.*, "Smart-its friends: A technique for users to easily establish connections between smart artefacts," in *Ubiquitous Computing (UbiComp)*. London, UK: Springer-Verlag, 2001, pp. 116–122.
[27] R. Mayrhofer and H. Gellersen, "Shake Well Before Use: Authentication Based on Accelerometer Data," in *Pervasive Computing (PERVASIVE)*. Springer, 2007.
[28] K. Kostiainen and E. Uzun, "Framework for comparative usability testing of distributed applications," in *Security User Studies: Methodologies and Best Practices Workshop*, 2007.
[29] J. M. McCune, Personal Communication, Mar 2008.
[30] K. Kostiainen, Personal Communication, Mar 2008.
[31] L. Faulkner, "Beyond the five-user assumption: Benefits of increased sample sizes in usability testing," *Behavior Research Methods, Instruments, & Computers*, vol. 35, no. 3, pp. 379–383, 2003.