Secure Pairing of "Interface-Constrained" Devices Resistant against Rushing User Behavior

Nitesh Saxena and Md. Borhan Uddin

Computer Science and Engineering
Polytechnic Institute of New York University
nsaxena@poly.edu, borhan@cis.poly.edu

Abstract. "Secure Device Pairing" is the process of bootstrapping secure communication between two devices over a short- or medium-range wireless channel (such as Bluetooth, WiFi). The devices in such a scenario can neither be assumed to have a prior context with each other nor do they share a common trusted authority. Fortunately, the devices can generally be connected using auxiliary physical channel(s) (such as audio, visual, tactile) that can be authenticated by the device user(s), thus forming the basis for pairing. However, lack of good quality output interfaces (e.g, a speaker, display) and/or receivers (e.g., microphone, camera) on certain devices makes pairing a very challenging problem in practice.

We consider the problem of "*rushing user*" behavior in device pairing. A rushing user is defined as a user who in a rush to connect her devices, would skip through the pairing process, if possible. Most prior pairing methods, in which the user decides the final outcome of pairing, are vulnerable to rushing user behavior – the user can simply "accept" the pairing, without having to correctly take part in the decision process. In this paper, we concentrate on most common pairing scenarios (such as pairing of a WiFi laptop and an access point), whereby one device (access point) is constrained in terms output interfaces, while the other (laptop) has a decent quality output interface but no receiver. We present the design and usability analysis of two novel pairing methods, which are resistant to a rushing user and require only minimal device interfaces on the constrained device. One of the most appealing applications of our proposal is in defending against common threat of "Evil Twin" attacks in public places (e.g, cyber-cafes, airport lounges).

Keywords: Device Pairing, Authentication, Usability, Security, Evil Twin Attacks, Wireless Communication.

1 Introduction

Short-range wireless communication, based on technologies such as Bluetooth and WiFi, is becoming increasingly popular and promises to remain so in the future. With this surge in popularity, come various security risks. Wireless communication channel is easy to eavesdrop upon and to manipulate, and therefore a fundamental security objective is to secure this communication channel. In this paper, we will use the term "pairing" to refer to the operation of bootstrapping secure communication between two devices connected with a short-range wireless channel. The examples of pairing, from day-to-day life, include pairing of a WiFi laptop and an access point, a Bluetooth keyboard and a desktop. Pairing would be easy to achieve, if there existed a global infrastructure enabling devices to share an on- or off-line trusted third party, a certification authority, a PKI or any pre-configured secrets. However, such a global infrastructure is close to impossible to come by in practice, thereby making pairing an interesting and a challenging real-world research problem. The problem has been at the forefront of various recent standardization activities, see [24].

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

The usability of a pairing method based on OOB channels is clearly of utmost importance. In pairing scenarios (such as pairing of a WiFi laptop and an access point) involving devices that lack good quality output interfaces (e.g, a speaker, display) and/or receivers (e.g., microphone, camera), establishing OOB channels is quite difficult. Minimizing the user burden in pairing such "interface-constrained" devices is thus a very challenging problem. Since the OOB channels typically have low bandwidth, the shorter the data that a pairing method needs to transmit over these channels, the better the method becomes in terms of usability.

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

The earlier pairing protocols [4], [14] require at least 80 to 160 bits of data to be transmitted over the OOB channels. The more recent, so-called SAS- (Short Authenticated Strings) based protocols, [12] and [15], reduce the length of data to be transmitted over the OOB channels to only 15 bits or so, for a reasonable level of security.[1]

Based on the above-mentioned protocols, a number of pairing methods with various OOB channels have been proposed. All prior pairing methods are reviewed in the following section of the paper.

In this paper, we consider the 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

[1] The concept of SAS-based authentication was first introduced by Cagalj et al. [27], followed by Vaudenay [26]. MANA protocols [7] addressed a similar problem.

to be "task-focussed" [6]. For example, in the context of phishing attacks [6], 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 eventually 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.

Our Contributions. In this paper, we concentrate on most common pairing scenarios, wherein one device is constrained in terms output interfaces, while the other has a decent quality output interface but no receiver. A common example of such a scenario is pairing of a WiFi laptop and an access point (the latter is a constrained device with no display or receiver; the former has a full display and keypad, but no receiver). We note that some prior work addresses the problem of pairing interface-constrained devices (e.g., [17][18]). However, this paper is the first, to the best of our knowledge, to address an even more challenging problem of pairing interface-constrained devices in a rushing user resistant manner. We present the design and usability analysis of two novel pairing methods, which are resistant to a rushing user and require only minimal device interfaces on the constrained device. The two proposed pairing methods, called "*color pairing*" and "*alphanumeric pairing*," are based on two different types of output interfaces on the constrained device, i.e., a Multi-Color LED and a Sixteen Segment Display (SSD), respectively. Both of these interfaces are minimal in terms of their cost and size, are commonly available and thus can be easily added onto constrained devices (such as access points, printers).[2] In the color pairing method, the user is required to transfer colors displayed through the multi-color LED of the constrained device to the other device. In the alphanumeric pairing method, the user simply transfers the characters displayed by the SSD of the constrained device onto the other device. Clearly, both our methods are based on different sensory capabilities of human users and have different usability implications

Based on a usability study of the proposed pairing methods, we conclude that the color pairing method based on four distinct colors is quite suitable for most devices and

[2] The use of such interfaces is not just limited to the pairing operation. For example, a multi-color LED can serve the purpose of a general-purpose LED on an access point, and the SSD can serve as a minimal display on a printer.

pairing scenarios, as it turned out to be very efficient, robust to human errors and user-friendly. This method is ideal, as our testing indicates, for defending against a common threat of *evil twin* access points (e.g., in cyber-cafes, airport lounges), in a rushing user resistant manner.[3]

Organization. The rest of the paper is organized as follows. In Section 2, we review the prior pairing methods. In Section 3, we describe the security model and summarize relevant protocols. In Sections 4 and 5, we present the design and implementation of our color pairing and alphanumeric pairing methods. Finally, in Section 6, we discuss our experimental usability study with respect to the proposed pairing methods and the underlying results.

2 Related Work

In this section, we discuss prior pairing methods, their applicability to interface constrained devices and whether or not they are resistant to rushing user behavior. Recall a DC method is resistant to rushing user, whereas a UC method is not.

In their seminal work, Stajano, et al. [23] proposed establishing a shared secret between two devices using a link created through a physical contact (such as an electric cable). This is a DC method and is resistant to rushing user. However, in many settings, establishing such a physical contact might not be possible, for example, the devices might not have common interfaces to do so or it might be too cumbersome to carry the cables along. Balfanz, et al. [4] extended this approach through the use of infrared as a d2d channel – the devices exchange their public keys over the wireless channel followed by exchanging (at least 80-bit long) hashes of their respective public keys over infrared. This is also a DC method. The main drawback of this method, however, is that it is only applicable to devices equipped with infrared transceivers. Moreover, the infra-red channel is not easily perceptible by human users.

Another approach taken by a few research papers is to perform the key exchange over the wireless channel and authenticate it by requiring the users to manually and visually compare the established secret on both devices. Since manually comparing the established secret or its hash is cumbersome for the users, methods were designed to make this visualization simpler. These include Snowflake mechanism [9] by Levienet et al., Random Arts visual hash [16] by Perrig et al., etc. These methods require high-resolution displays and are thus only applicable to a limited number of devices, such as laptops. Moreover, these are UC methods and thus are vulnerable to a rushing user.

Based on the pairing protocol of Balfanz et al. [4], McCune et al. proposed the “Seeing-is-Believing” (SiB) method [13]. SiB involves establishing two unidirectional visual d2d channels – one device encodes the data into a two-dimensional barcode and the other device reads it using a photo camera. SiB is a DC method. However, since the method requires both devices to have cameras, it is only suitable for pairing devices such as camera phones.

Goodrich, et al. [10], proposed “Loud-and-Clear (L&C)”, a pairing method based on “MadLib” sentences. The main idea of L&C is to encode the OOB data into MadLib

[3] This only involves unidirectional authentication of the access point to the laptop.

sentences and have the user compare these sentences displayed or spoken out on two devices. Clearly, this is a UC method and is thus vulnerable to rushing user behavior. Moreover, the method is not applicable to pairing scenarios where one of the devices does not have a display or a speaker.

Saxena et al. [19] proposed a pairing method based on visual OOB channel. The method uses one of the SAS protocols [12], and is aimed at pairing two devices A and B (such as a cell phone and an access point), only one of which (say, B) has a relevant receiver (such as a camera). First, a unidirectional d2d channel is established by device A transmitting the SAS data, e.g., by using a blinking LED and device B receiving it using a video camera. This is followed by device B comparing the received data with its own copy of the SAS data, and transmitting the resulting bit of comparison over a d2h channel (say, displayed on its screen). Finally, the user reads this bit transmitted and accordingly indicates the result to device A by transmitting a bit over an h2d input channel. In one direction (i.e., from A to B), this is a DC method and is thus resistant to rushing user behavior. In the other direction, however, it is a UC method – a rushing user can simply accept the pairing on A without looking at the pairing outcome on B. It is important to note, however, that in case of any attack, device B will be "locked out" and will not allow any connection to and from device A (and it will detect any connection attempts from an attacking device).[4] This way the user will not be able to establish real communication with device B (e.g., transfer an image file from B to A), and will thus resort to repeating the pairing process. The pairing methods we propose in this paper utilize this unidirectional pairing approach of [19].

Uzun et al. [25] carry out a comparative usability study of simple pairing methods. They consider pairing scenarios where devices are capable of displaying 4-digits of SAS data. In what they call the "Compare-and-Confirm" approach (a UC method), the user simply reads and compares the SAS data displayed on both devices. The "Select-and-Confirm" approach (a DC method), on the other hand, requires the user to select a 4-digit string (out of a number of strings) on one device that matches with the 4-digit string on the other device. The third approach, called "Copy-and-Confirm" (a DC method), requires the user to read the data from one device and input it onto the other. Both Select-and-Confirm and Copy-and-Confirm are DC methods and therefore offer protection against a rushing user. However, these methods are only limited to devices (such as cell phones) which have good quality displays and keypads. Our alphanumeric pairing method is quite similar in flavor to the Copy-and-Confirm method of [25], however, it is applicable to interface-constrained devices. Kuo et al. [11] defined a common baseline for hardware features and a consistent, interoperable user experience across pairing of different devices. This work did not yield any pairing method as such.

Some recent papers have focused upon pairing devices which possess constrained interfaces. These include the BEDA method [21] which requires the users to transfer the SAS strings from one device to the other using "button presses". BEDA is based on the protocol of [19]. The constrained device encodes its SAS string into the time intervals between two consecutive blinkings of the (regular) LED, and as and when this

[4] In case of a pairing failure, device B can keep showing a warning to the user indicating that device A is possibly being connected to an attacker device, and ask the user to "re-pair" the two devices.

device blinks, the user presses a button on the other device. BEDA is a DC method and is therefore resistant to rushing user behavior. BEDA is also universally applicable to most pairing scenarios. However, as indicated in the results of [21], it requires about one minute to complete the pairing process, which might be too slow in practice. Moreover, the user needs to pay close attention to both devices simultaneously to maintain synchronization. This will be particularly hard when one of the devices is distant (e.g., a wall-mounted access point). The methods that we present in this paper are more efficient in comparison to BEDA, as we will see in the later sections of the paper.

In [17], Saxena et al. presented a pairing method universally applicable to any pair of devices. The method can be based on any of the existing SAS protocols and does not require devices to have good transmitters or any receivers, that is, just a pair of LEDs is sufficient. The method involves users comparing very simple audiovisual patterns, such as "beeping" and "blinking", transmitted as simultaneous streams which form two synchronized d2h channels. Most recently, the approach of [17] was extended by making use of an auxiliary device, such as a smartphone [20]. Both these methods, however, are UC methods and thus offer no protection against a rushing user. In an independent result [18], Roth et al. present a method similar to the "blinking" method presented in [17]. The method of [18] is aimed at the detection of evil twin access points. The two methods, however, differ significantly in their implementation and therefore in terms of user experience (see [17] for details regarding the differences). This method is also a UC method and thus offers no protection against a rushing user. The pairing methods we propose in this paper aptly address the problem of evil twin access points, however, unlike [18], our methods also offer resistance against rushing user behavior.

In [22], Soriente et al. consider the problem of pairing two devices which might not share any common wireless communication channel at the time of pairing, but do share only a common audio channel. This is a DC method, however, it is only limited to devices which possess a speaker at the transmitting end and a microphone at the receiving end.

3 Security Model and Applicable Protocols

The pairing protocols, on which our methods are built, are based upon the following communication and adversarial model [26]. The devices being paired are connected via two types of channels: (1) a short-range, high-bandwidth bidirectional wireless channel and (2) one or more auxiliary low-bandwidth physical OOB channel(s). Based on the type of devices being used, the OOB channel(s) can be device-to-device (d2d), device-to-human (d2h), or human-to-device (h2d). An adversary attacking the pairing protocol is assumed to have full control of the wireless channel, namely, he or she can eavesdrop, delay, drop, replay and modify messages. On the OOB channel, the adversary can eavesdrop, delay, drop, replay and re-order messages; however, it can not modify them. In other words, the OOB channel is assumed to be an authenticated channel.

The security notion applied to a pairing protocol in this setting is adopted from the model of authenticated key agreement by Canneti and Krawczyk [5]. In this model, a multi-party setting is considered wherein a number of parties simultaneously run multiple/parallel instances of pairing protocols. In practice, however, it is reasonable to assume that there are only two parties running just a few serial or parallel instances of the pairing protocol. For example, during the authentication of an ATM transaction there are only two parties, namely the ATM machine and a user. Further, the user is restricted to only three authentication attempts. The security model does not consider denial-of-service (DoS) attacks. Note that with a wireless channel explicit attempts to prevent protocol-level DoS attacks are not useful because an adversary can simply launch an attack by jamming the wireless signal.

To date, two three-round pairing protocols based on short authenticated strings (SAS) have been proposed: [15] and [12]. In a communication setting involving two users restricted to running three instances of the protocol these SAS protocols need to transmit only k $(= 15)$ bits of data over the OOB channel. As long as the cryptographic primitives used in the protocol are secure, an adversary attacking one of these protocols can not win with a probability significantly higher than 2^{-k} $(= 2^{-15})$. This gives us security equivalent to that provided by 5-digit PIN-based ATM authentication. The pairing methods proposed in this paper are based upon the SAS protocols mentioned above, with a variation presented in [19], as discussed in Section 2.

4 "Color Pairing" Using a Multi-Color LED

In this section, we discuss the design and implementation of a novel color-based pairing method, called "color pairing", which is resistant against rushing user. In our method, one device (denoted as A) is equipped with a "Multi-Color" LED [2] and the other device (denoted as B) has a display and a keypad. Device A encodes its SAS data into colors and displays each color one-by-one through the LED, the user reads each color and accordingly selects the corresponding color (from all possible displayed colors) on device B, maintaining synchronization (i.e., transition between two consecutive colors) of transmission and reception. In this manner, the SAS data is transferred from device A to device B, while maintaining synchronization. Once device B decodes the received SAS data, it compares it with its own local copy; and accordingly accepts or rejects the pairing instance. Notice that our method is a DC method and is thus resistant to rushing user behavior.

Clearly our method is based on the number of "human-distinguishable" colors an off-the-shelf multi-color LED is capable of displaying – the more the number of such colors, the more the number of SAS bits can be transmitted everytime. In the following subsections, we describe the selection of such human-distinguishable colors, the encoding of SAS data into these colors and rushing user resistant transmission and processing of SAS data.

4.1 Selection of "Human-Distinguishable" Colors

In the 24-bit RGB color model, there are 2^{24} (i.e., about 16-million) distinct colors. Out of these colors, our goal was to find out the colors which can be generated using a multi-color LED and which are easily and unambiguously distinguishable by human users without any prior training. One recent study [8] shows that there are only 11 non-conflicting colors identified for categorical images. In this set of 11 colors of [8], there seems to be some pairs of colors which do not appear that distinct, e.g., light green and dark green, and blue and dark blue. If a user is shown blue (without showing dark blue beforehand) and asked to identify whether it is dark blue or blue, it is highly likely that the user will be confused, because the user would not be sure whether a more or less bright version of the color exists or not. We generated the same set of 11 colors and some more colors on monitor screen using the simulated annealing method as described in [8]. However, with the help of some preliminary testing on some human users, we concluded that the number of distinct colors is definitely not more than 11; in fact, it might be less than 11. This was also because all the 11 colors of [8] were not easily distinguishable by human users on multi-color LED as light and dark shades of the same color (e.g., light green and dark green) turned out to be confusing/conflicting with respect to surrounding light.

Based on our initial experimentation and testing, as described above, we finally decided to stick to a maximum of 8 human-distinguishable colors. With 8 colors, we can encode 3-bits of SAS data at a time. These eight colors are comprised of 3 primary colors (Red, Green, Blue), 3 secondary colors (Yellow, Magenta, Cyan) and two tertiary colors (Orange and Violet). With the help of some initial testing, we assured ourselves that these 8 colors are unambiguous as displayed both on a monitor screen and on a multi-color LED. We used 24-bit RGB model for the colors; thus above eight colors have the following RGB values: Red (255,0,0), Green (0,255,0), Blue (0,0,255), Yellow (255,255,0), Magenta (255,0,255), Cyan (0,255,255), Orange (255,127,0) and Violet (127,0,255). We used primary, secondary and tertiary colors for color pairing because these colors are easy to generate on a multi-color LED. The primary colors are directly generated by LED cathodes and mixing the primary colors in different ratios generated secondary and tertiary colors.

4.2 Generating “Human-Distinguishable” Colors on a Multi-Color LED

Multi-Color LED [2] is a specialized form of LED which has one common Anode and 3 Cathodes for three primary colors (Red, Green and Blue). Each of the primary colors can be produced by directly turning ON each cathode (keeping others OFF). For producing secondary colors, we need to turn ON two of the cathodes concurrently. For example, Yellow is a combination of Green and Red. So, turning ON the green and red cathodes generated Yellow. Similarly, Magenta is produced by turning ON the red and blue cathodes concurrently and cyan is produced by turning on the green and blue cathodes concurrently. Tertiary colors Orange (Red + Yellow = $2\times$ Red + Green) and Violet (Blue + Magenta = $2\times$ Blue + Red) are produced by turning ON two cathodes simultaneously; but, currents are varied between cathodes to produce the tertiary colors. For example, to produce Orange which is a combination of Red and Green with ratio 2:1, current flow in Red cathode is kept twice than that of Green cathode. Similarly, Violet is produced by passing twice the amount of current in Blue cathode than that of Red cathode.

For varying the current on different cathodes, we designed a circuit (as shown in Figure 1(a)), using NPN transistors, different resistors and controlled it through a computer by sending the data through the parallel port. Each cathode of the multi-color LED is controlled by 3 pins of parallel port connected with different values of resistors. By sending 9-bit binary data to parallel port, current flows in cathodes are controlled.

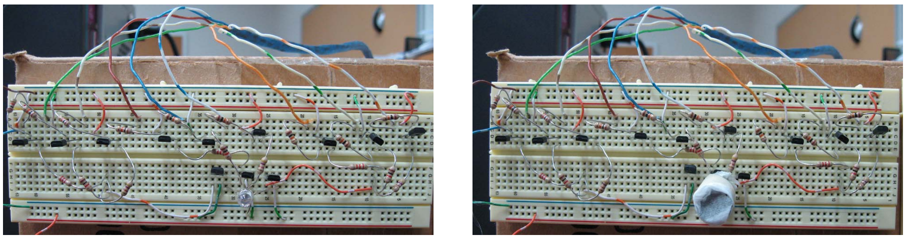

(a) Without color mixer coop (b) With color mixer coop

Fig. 1. Multi-Color LED Circuit on Breadboard

The multi-color LED was enclosed with a black plastic hollow coop and covered with thin layer of tissue paper from outside (as shown in Figure 1(b)). This was done in order to generate the secondary and tertiary colors by properly mixing the primary colors.

4.3 Implementation: Transmission and Decoding

We developed an application in Visual C# to control the LED controller circuit on breadboard connected with a computer through the parallel port (as shown in Figure 1). The SAS data is mapped onto color values and color values are used to activate the corresponding cathodes of the LED in order to generate the human-distinguishable colors. When the user starts the pairing process, the application starts showing each color on the LED first and then asks the user to select the corresponding color on the screen from a list of all possible human-distinguishable colors. After selection of each color by the user, the application turns OFF the LED and asks the user to verify whether or not the LED turned OFF and accordingly press “Yes” or “No” button (respectively) to proceed and display the next color. This is done in order to keep the transmission resistant against insertion, deletion, delay and/or replay of synchronization signals between the two devices. A synchronization signal is sent, over the wireless channel, from device B with color input screen to device A displaying colors, as soon as the user selects the color on B. This instructs device A to now show the next color. Turning OFF of the LED indicates to the user that the synchronization signal has been correctly received by device A; if the LED stays ON, it indicates a synchronization error/attack.

The above process continues until the whole SAS data is transmitted encoded through human-distinguishable colors unless there is synchronization error (i.e., the LED is not turned OFF and the user presses on “NO” button or vice versa). After transmission (through human distinguishable colors on LED) and reception (from users’ color selection on screen) of the SAS data , the application shows the result (failure/success) of pairing. If there are no synchronization errors and if the SAS strings match, the pairing is deemed successful; otherwise, the pairing fails.

For 15-bit SAS data and N human-distinguishable colors, we require $\frac{15}{log_2 N}$ “passes” for transmission of SAS data. In each pass, one of N colors is shown by the multi-color LED and user selects the corresponding color from N colors on screen.

5 “Alphanumeric Pairing” Using a Sixteen-Segment Display

Sixteen Segment Display (SSD) [3] is an inexpensive, commercially available minimal alphanumeric display. It is capable of showing all the (capital) English alphabets and all digits (0-9). Due to its low cost, small size, availability and good layout, SSD is quite suitable to be incorporated for pairing operation.

In the “alphanumeric pairing” method, the SAS data is encoded into alphanumeric characters and displayed on the SSD of device A one-by-one. The user simply reads each displayed character and types it onto the keypad of device B. Similar to the pairing method based on the multi-color LED (as described in previous section), after each character is typed in by the user, the display is turned OFF and the user is asked to verify whether it is turned OFF or not, before displaying the next character. In this manner, the SAS data is transferred from device A to device B, while maintaining synchronization. Once device B decodes the received SAS data, it compares it with its own local copy; and accordingly accepts or rejects the pairing instance.

5.1 Encoding of SAS Data into Alphanumeric Characters

SSD can show 10 digits and 26 capital letters of English alphabet; i.e., a total of 36 alphanumeric characters. We wanted to keep the set of characters as unambiguous as possible so that the users with no prior knowledge of the character layout on SSD can easily identify the characters. To this end, we decided to discard 4 characters, ‘0’, ‘O’, ‘5’, and ‘S’, which appear quite ambiguous (without prior knowledge of the layout, users might mistake a ‘0’ for a ‘O’ or a ‘5’ for an ‘S’, and vice versa). This left us with a character space of size 32.

For 15-bit SAS transmission using 32 alphanumeric characters, a total of $\frac{15}{log_2(32)} = 3$ passes, i.e., 3 characters, need to be transferred between the two devices. After each

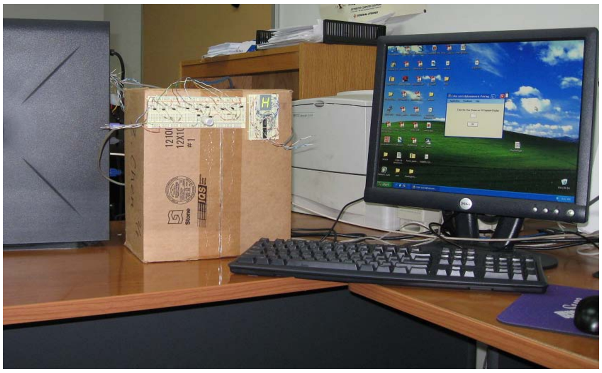


Fig. 2. Overall Experimental Setup of the Color and Alphanumeric Pairing (the cardboard box with the multi-color LED and the SSD, simulates, e.g., an access point in a cyber-cafe; the desktop simulates the laptop.)

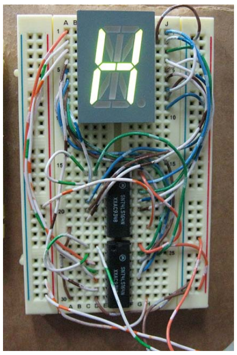


Fig. 3. Sixteen Segment Display (displaying character ‘H’)

pass, the SSD display is turned OFF and the user is asked to verify whether it is indeed OFF and accordingly press “Yes” or “No” button to proceed.

To implement our new pairing method using SSD, we extended our C# application we developed for color-based pairing (as described in previous section). The SSD was connected with the parallel port of the computer via two “serial-in-parallel-out” shift registers. Serial data and clocks were sent from the parallel port to the shift registers and shift registers supplied the data in parallel to the SSD. Figure 3 shows the snapshot of the setup of the SSD we used.

6 Experiments and Results

6.1 Experimental Setup

To test our color and alphanumeric pairing methods, we used the following set-up. The application which controls the multi-color LED and alphanumeric display is running on a DELL Desktop computer (1.8 GHz CPU, 1 GB RAM, WinXP Pro SP2) connected with multi-color LED on breadboard and sixteen segment display (as shown in Figure 2) through parallel port (DB25 Connector). This computer works as both transmitter and receiver of SAS data in both color-based and alphanumeric pairing. In color pairing, the breadboard with multi-color LED connected with parallel port simulates the transmitting device and application running on computer having interface for selecting colors simulates the other device. Similarly, the breadboard with sixteen segment display connected with parallel port of computer simulates one device and application running on the desktop computer having character input interface simulates another device in alphanumeric pairing.

For color pairing circuit, we used one multi-color LED (08L5015RGBC) [2], 12 NPN Transistors (ZTX450), 3 categories (0.5k, 1k, 2k Ohms) resistors - 3 of each value and nine 10k Ohm resistors. For alphanumeric pairing circuit, we used one sixteen segment display (AND-8010GCLB) [3], two serial-in-parallel-out shift registers (SN74LS164). Both the color and alphanumeric pairing circuits are powered from DC power source of the computer.

As mentioned in prior sections, an application running on desktop computer is developed in Microsoft Visual C# for controlling both the circuits of color and alphanumeric pairing. The application also supports all necessary functionality for an automated user testing. The application accepts the inputs from users in both color and alphanumeric pairing, shows the result of pairing and finally records the users feedback as part of the usability testing. The application also keeps track of users inputs and timings and logs the result of pairing and all necessary information regarding users background and their feedback. A couple of screen-shots of the execution of our application for both color and alphanumeric pairing are shown in Figures 4 and 5.

6.2 Usability Testing

In order to test how both of our pairing methods fare with users, and especially to figure out if the users are easily and correctly able to transfer the colors and alphanumeric characters as displayed by the multi-color LED and the SSD, respectively, we performed a thorough and systematic usability study. We focused on a common security application of detecting Evil Twin access points (as in [18], but in a rushing user resistant manner). Note that this only requires unidirectional authentication. Thus, we did not incorporate in our tests the final step of the protocol of [19] whereby the user accepts or rejects the pairing on device A based on the pairing outcome shown by device B. See Figure 2 depicting our experimental set-up.

6.3 Testing Framework

For creating a user-friendly but realistic testing framework, we extended the circuit controller application (running on the desktop computer) by implementing the usability testing and user feedback collection functionality on it.

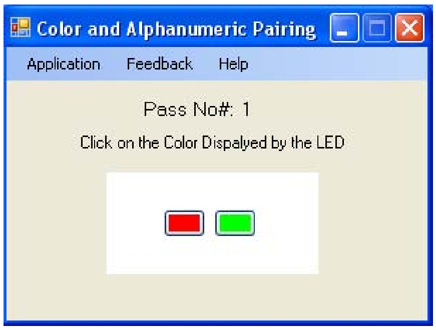
For color pairing, the users are instructed to transfer the colors as displayed by the multi-color LED to the desktop screen by clicking on the corresponding "color buttons". For testing our color pairing method with respect to pairing time and user errors, we conducted our usability testing in 3 settings - using 2, 4 and 8 colors. For 15-bit SAS string, "2-Color" pairing method requires $\frac{15}{log_2 2} = 15$ passes. Similarly, "4-Color" pairing requires $\lceil \frac{15}{log_2 4} \rceil = 8$ passes and "8-Color" pairing requires 5 passes. For all the settings, each pass is comprised of four steps: (1) Displaying of a SAS-encoded color on the LED, (2) Selection of the color by the user on desktop screen, (3) Turning off of the LED (i.e., with no color being displayed on the LED) and (4) Verification by the user whether the LED is in OFF state.

For the 2-Color pairing method, we selected first two colors (Red and Green) out of the primary colors (Red, Green and Blue). For the 4-Color pairing method, we selected primary colors (Red, Green, Blue) and first color (Yellow) out of the of secondary colors (Yellow, Magenta and Cyan). For the 8-Color pairing method, we selected primary colors (Red, Green, Blue), secondary colors (Yellow, Magenta, Cyan) and two colors (Orange and Violet) out of tertiary colors (Azure, Violet, Rose, Orange, Chartreuse and Aquamarine). A few screen-shots of user interfaces for color pairing methods are shown in Figure 4.

For testing the alphanumeric pairing method with respect to pairing time and user errors, the application is configured to take alphanumeric inputs from the users. Users were instructed to input the character, displayed on sixteen segment display, onto the desktop keyboard. For 32 alphanumeric characters and 15-bit SAS data, it requires $\frac{15}{log_2 32} = 3$ passes for the pairing process. Each pass is comprised of the following steps: (1) Displaying of a SAS-encoded character on the sixteen-segment display, (2) Inputting the corresponding character on desktop keyboard, (3) Turning off of the display (i.e., with no character being displayed on the SSD) and (4) Verification by the user whether the SSD is in OFF state. The application converted all characters to uppercase on the input text-box and displayed the result of pairing after the completion of 3 passes. A couple of screen-shots of user interfaces for the alphanumeric pairing are shown in Figure 5.

6.4 Test Cases

For each of the three methods of color pairing (i.e., using 2, 4 and 8 colors), two test cases were created; thus, a total of 6 test cases were executed by each user for color

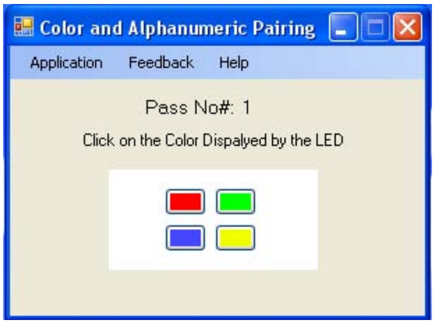


Pass No#: 1

Click on the Color Dispalyed by the LED

🟥 🟩

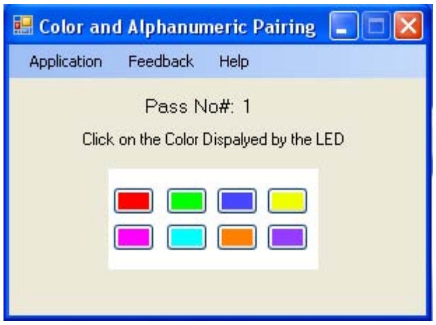
(a) Using Two Colors



Pass No#: 1

Click on the Color Dispalyed by the LED

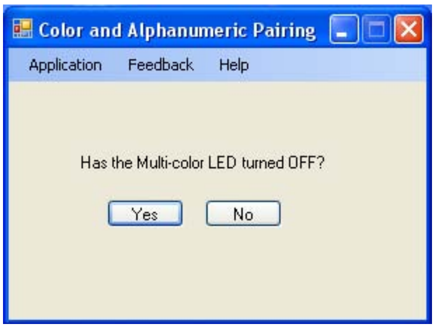
(b) Using Four Colors



Pass No#: 1

Click on the Color Dispalyed by the LED

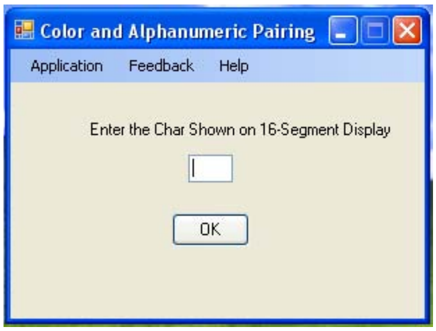
(c) Using Eight Colors



Has the Multi-color LED turned OFF?

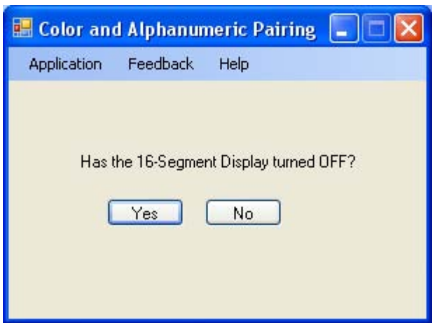
(d) Turn OFF Verification Screen

Fig. 4. Usability Testing of Color Pairing Using Multi-Color LED



Enter the Char Shown on 16-Segment Display

(a) Character Input Interface



Has the 16-Segment Display turned OFF?

(b) Turn OFF Display Checking Screen

Fig. 5. Usability Testing of Alphanumeric Pairing

pairing. In each color pairing method, all the colors of that particular method appeared at least once (in other words, no user missed out a single color) and the colors appeared in random order on the multi-color LED.

For alphanumeric pairing, 15-bit SAS data requires each test case to show $\frac{15}{log_2 32} = 3$ alphanumeric characters. Thus, it required $\lceil\frac{32}{3}\rceil = 11$ test cases to show all the 32 alphanumeric characters at least once to each user. Alphanumeric characters were shown randomly on the sixteen segment display and each user executed a total of 11 test cases of alphanumeric pairing.

6.5 Test Participants

We recruited 20 subjects for the usability testing of both color and alphanumeric pairing. Subjects were chosen on a first-come first-serve basis from respondents to recruiting posters and email advertisements. At the end of the tests, the participants were asked to fill out an on screen questionnaire through which we obtained user demographics and their feedback on the methods tested.

Recruited subjects were mostly university students, both graduate and undergraduate, with CS and non-CS backgrounds. This resulted in a fairly young (ages between 18-35 [*mean*=24.15, *se*=0.7549]), well-educated participant group. All participants were regular computer and cell phone users. 18 out of 20 participants reported they have previously used a wireless accessory such as access point/modem/router. 12 out of 20 participants reported they have previously used a bluetooth device such as bluetooth-headset, mouse or keyboard. Eight participants were familiar with the vulnerability of an un-encrypted wireless channel and five participants chose the statement “Un-encrypted wireless channel isn’t vulnerable to attack” and 7 participants were not sure about the statement. None of the study participants reported any physical impairments that could have interfered with their ability to complete given tasks and none of them had any visual disability, color vision problem or color blindness. The gender split was: 4 females and 16 males.

6.6 Testing Process

Our study was conducted in a graduate student laboratory of our university. Each participant was given a brief overview of our study goals and our experimental set-up. Each participating user was then asked to follow on-screen instructions on the desktop computer for both color and alphanumeric pairing. No training of any sort was given. Basically, the participants played the role of the user in the color and alphanumeric pairing process i.e., they transferred colors shown by the multi-color LED and alphanumeric characters shown by the Sixteen Segment Display to the application running on Desktop Computer. Each user completed 6 color pairing tests (two test cases for each category using 2, 4 and 8 colors) and 11 alphanumeric pairing test cases. Pairing outputs, user interactions throughout the tests and timings were logged automatically by the testing framework.

After completing the deputed test cases for both the color and alphanumeric pairing in the above manner, the participants were asked to give some qualitative feedback on the tested methods. For color pairing, participants were asked to score on a 1-10 scale (1-Low, 10-High) how distinct the colors were as shown by the monitor screen and by the multi-color LED; how easy they found to read and transfer the colors from multi-color LED to monitor screen. Users were also asked to choose- which color pairing method they preferred the most: 2-Color, 4-Color or 8-Color pairing.

For alphanumeric pairing, participants were asked to score on a 1-10 scale (1-Low, 10-High) how distinct were the alphanumeric characters as shown by the sixteen segment display, how easy they found to read and transfer the characters from sixteen segment display to the computer.

Participants' demographic information such as age, gender, educational qualification, visual and color vision disability, computer, wireless and bluetooth device usage experience and knowledge on security of wireless channel were all collected through this questionnaire. All user data and feedback were logged by the testing framework for later analysis.

6.7 Test Results

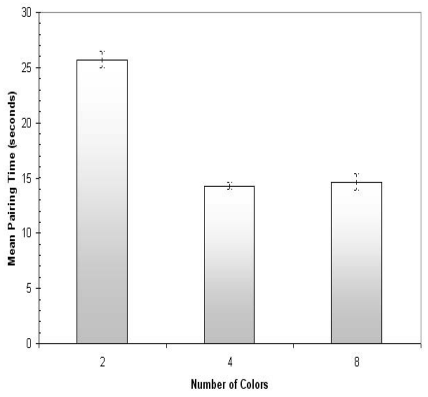
Each of our 20 subjects executed 6 color pairing and 11 alphanumeric test cases, leading to a total of 120 color pairing (i.e., 40 test cases for each of 2-Color, 4-Color, and 8-Color pairing methods) and 220 alphanumeric pairing test cases.

Errors. Most of the test cases completed successfully giving expected results. In some cases, however, we observed a few errors, which we categorize and describe below.

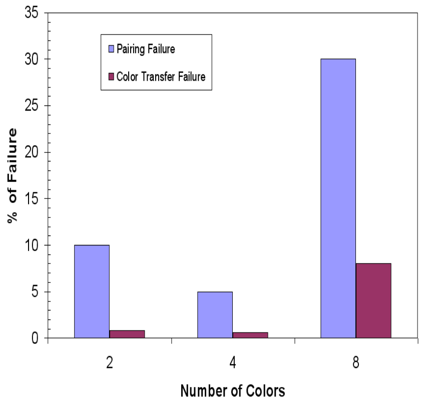
– *Color Pairing Errors:*
  In the 2-Color pairing method, 4 users clicked on wrong colors for a total of 5 times in 4 test cases. Thus, 4 test cases failed out of 40 test cases. So, *pairing failure* rate = $\frac{4}{40} \times 100\% = 10\%$. Users failed to transfer 5 colors out of $40 \times 15 = 600$ colors. So, *color transfer failure* rate = $\frac{5}{600} \times 100\% = 0.83\%$.
  In the 4-Color pairing method, 2 users clicked on wrong colors for a total of 2 times in 2 test cases. This led to a *pairing failure* rate = $\frac{2}{40} \times 100\% = 5\%$ and a *color transfer failure* rat e= $\frac{2}{320} \times 100\% = 0.625\%$.
  In the 8-Color pairing method, 12 users clicked on wrong color for a total of 16 times in 12 test cases, leading to a *pairing failure* rate = $\frac{12}{40} \times 100\% = 30\%$. and a *color transfer failure* rate = $\frac{16}{200} \times 100\% = 8\%$.
  The graph presented in Figure 6(b) depicts the pairing failure rates and color transfer failure rates for the three color pairing methods.
– *Alphanumeric Pairing Errors:*
  In the alphanumeric pairing, 9 users made a total of 12 errors in 12 test cases (i.e., single character errors in each failed test case) out of 220 test cases as listed in Figure 8.
  Therefore, the *pairing failure* rate = $\frac{12}{220} \times 100\% = 5.45\%$. 20 users transferred a total of $20 \times 11 \times 3 = 660$ characters and out of them, 12 characters were transferred as incorrectly So, *character transfer failure* rate = $\frac{12}{660} \times 100\% = 1.818\%$.

Test Timing. The mean pairing time of color pairing using 2, 4 and 8 colors are shown in the graph of Figure 6(a). Clearly each of the color pairing methods requires less than 30 seconds of pairing time and the 4-Color pairing method is the fastest of them all [*mean*=14.289 seconds, *se*=0.3138], whereas the 2-Color is the slowest [*mean*=25.7318 seconds, *se*=0.7545].

The average time taken by each user (over the 11 test cases) to perform the alphanumeric pairing is depicted in Figure 7. Clearly, it shows that most of the users completed the alphanumeric pairing within 15 seconds [*mean*=9.393 seconds, *se*=0.2670].



(a) Pairing Time with Standard Error



(b) Pairing and Color Transfer Failure Rates

Fig. 6. Results of Color Pairing - using 2, 4 and 8 Colors

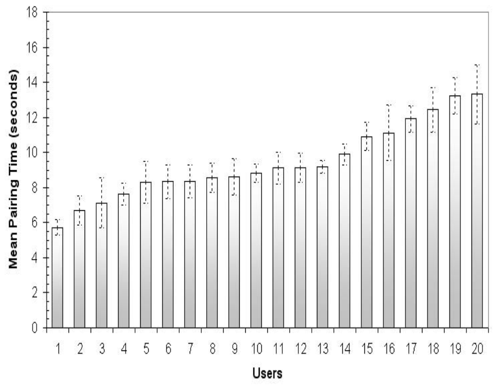


Fig. 7. Result of Alphanumeric Pairing: User Timing with Standard Error (users are sorted by average time)

Displayed on SSD	Transferred by User	# of Occurrences (out of 660)
G	6	4
Q	0	4
1	I	3
B	D	1

Fig. 8. User Errors in Reading and Transferring Alphanumeric Characters

These timings are commensurate with the timings of the pairing methods presented in [17] [18]. Recall that the latter methods are not resistant to a rushing user behavior.

User Feedback. The user feedback was collected on the distinctiveness of all 8 colors as shown by the multi-color LED and on monitor screen and on the easiness to transfer these colors. It was assumed that distinctiveness and unambiguity of these 8 colors would automatically imply the distinctiveness and unambiguity of the colors used in the 2-color and 4-color pairing methods. As our results show, most users found the methods robust and quite easy to work with. The qualitative results we obtained through the user feedback questionnaire on color pairing are shown in Table 1.

The users were also asked to compare the three methods in terms of the distinctiveness of underlying colors, ease of transfer and selection of colors and overall work-load in the whole process. In this respect, 2 out of 20, i.e., 10% users preferred the 8-color

Table 1. User Feedback Score [1 (Low) - 10 (High)] with Standard Error (*se*) on Color Pairing

Basis	**Score**
Distinctness of the colors on Monitor Screen	9.45 (*se*=0.1352)
Distinctness of the colors on Multi-Color LED	7.4 (*se*=0.3934)
Easiness of color transfer from Multi-Color LED to Monitor Screen	8.6 (*se*=0.3276)

Table 2. User Feedback Score [1 (Low) - 10 (High)] with Standard Error (*se*) on Alphanumeric Pairing

Basis	**Score**
Distinctness of the characters on 16-Segment Display	8.85 (*se*=0.2542)
Easiness of character transfer from 16-Segment Display to Computer	9.05 (*se*=0.3118)

pairing method, 10 out of 20, i.e., 50% users preferred the 4-color pairing method and 8 out of 20; i.e., 40% users preferred the 2-color pairing method.

The results of the user feedback questionnaire on alphanumeric pairing are depicted in Table 2. Similar to the color pairing methods, most users found the method robust and quite easy to work with. We did not find any notable correlation of the subjects' age, gender and technical expertise with the results obtained for both color and alphanumeric pairing.

7 Conclusions and Future Work

In this paper, we addressed a challenging problem of pairing interface-constrained devices in a rushing user resistant manner. We can draw the following conclusions from the results of usability testing of our color and alphanumeric pairing methods.

Among the color pairing methods, the 4-color method turns out to be a clear winner in terms of pairing speed, errors and usability. This is mainly due to the reason that the four colors used in the 4-color pairing method are quite distinct and unambiguously identifiable by human users, as opposed to the eight colors used in the 8-color pairing method. In comparison to the 2-color method, on the other hand, the 4-color method requires fewer passes, thus speeding up the pairing process and reducing user burden.

As our results show, the alphanumeric pairing method also turned out to be quite robust to errors, fast and user-friendly. The errors resulting through this method could potentially be further reduced by using less confusing non-alphabetic characters.

Both 4-color pairing and alphanumeric pairing methods can also be easily adopted on devices which have good quality output interfaces (such as a full display on a cell phone). Notice that on such devices, there will be no need for synchronization between the two devices, as all of SAS data (e.g., all colors and all digits) can be displayed on one single screen. In fact, the "Copy-and-Confirm" method of [25] is nothing but our alphanumeric pairing method for devices with good quality displays.

Between the 4-color pairing and alphanumeric pairing methods, we find the latter to be slightly faster, but, the former to be slightly more robust to errors. In terms of usability also, the two methods turned out to be comparable. However, since a multi-color LED is smaller in size and slightly cheaper than the sixteen segment display, we believe that 4-color pairing would be a more practical choice. Moreover, LED colors can also be comprehended from a distance (e.g., in case of a wall-mounted access point). To support the 4-color pairing method, device manufacturers would only need to use a multi-color LED in place of regular LED(s) which are quite common on most devices.

In conclusion, our results show that the 4-color pairing method is quite appropriate for most pairing scenarios: it is fast, robust, user-friendly, resistant against rushing user behavior, and can be incorporated on most devices with only a little modification. Based on our testing, 4-color pairing method turns our to be ideal for defending against a common threat of evil twin access points, in a rushing user resistant manner. Alphanumeric pairing is suitable for devices that can afford to have a sixteen segment display.

In our current design of the 4-color pairing method, we did not consider "color-blindness", a common visual disability. It is found that most color-blind people are "red-green" color blind (i.e., they can not distinguish between red and green colors) [1]. To address color-blindness, one could select distinct colors other than red and green in our 4-Color pairing method. In our future work, we will design and evaluate such a variation of the 4-color pairing method.

Acknowledgments. The authors would like to thank Vikram Padman and Kurt Rosenfeld for their help and discussion on the setup involving Multi-Color LED.

References

1. Color blindness. On-line article Published by University of Illinois Eye and Ear Infirmary, `http://www.uic.edu/com/eye/LearningAboutVision/EyeFacts/ColorBlindness.shtml`
2. Datasheet and Specification for Multi-Color LED. Electronix Express/RSR Electronics, `http://www.elexp.com/a_data/0815015rgbc.pdf`
3. Datasheet and Specification of Sixteen Segment Display, `http://www.purdyelectronics.com/pdf/AND8010-B.pdf`
4. Balfanz, D., Smetters, D., Stewart, P., Wong, H.C.: Talking to strangers: Authentication in ad-hoc wireless networks. In: Network & Distributed System Security (NDSS) (2002)
5. Canetti, R., Krawczyk, H.: Analysis of key-exchange protocols and their use for building secure channels. In: Pfitzmann, B. (ed.) EUROCRYPT 2001. LNCS, vol. 2045, p. 453. Springer, Heidelberg (2001)
6. Dhamija, R., Tygar, J.D., Hearst, M.A.: Why phishing works. In: International Conference for Human-Computer Interaction (CHI) (2006)
7. Gehrmann, C., Mitchell, C.J., Nyberg, K.: Manual authentication for wireless devices. RSA CryptoBytes 7(1), 29–37 (Spring 2004)
8. Glasbey, C., van der Heijden, G., Toh, V., Gray, A.: Colour displays for categorical images. Color Research and Application 32, 304–309 (2007)
9. Goldberg, I.: Visual Key Fingerprint Code (1996), `http://www.cs.berkeley.edu/iang/visprint.c`
10. Goodrich, M.T., Sirivianos, M., Solis, J., Tsudik, G., Uzun, E.: Loud and Clear: Human-Verifiable Authentication Based on Audio. In: International Conference on Distributed Computing Systems (ICDCS) (2006)

11. Kuo, C., Walker, J., Perrig, A.: Low-cost manufacturing, usability, and security: An analysis of bluetooth simple pairing and wi-fi protected setup. In: Dietrich, S., Dhamija, R. (eds.) USEC 2007. LNCS, vol. 4886, pp. 325–340. Springer, Heidelberg (2007)
12. Laur, S., Asokan, N., Nyberg, K.: Efficient mutual data authentication using manually authenticated strings. In: Pointcheval, D., Mu, Y., Chen, K. (eds.) CANS 2006. LNCS, vol. 4301, pp. 90–107. Springer, Heidelberg (2006)
13. McCune, J.M., Perrig, A., Reiter, M.K.: Seeing-is-believing: Using camera phones for human-verifiable authentication. In: IEEE Symposium on Security and Privacy (2005)
14. Pasini, S., Vaudenay, S.: An optimal non-interactive message authentication protocol. In: The Cryptographers' Track at the RSA Conference (CT-RSA) (2006)
15. Pasini, S., Vaudenay, S.: SAS-Based Authenticated Key Agreement. In: Yung, M., Dodis, Y., Kiayias, A., Malkin, T.G. (eds.) PKC 2006. LNCS, vol. 3958, pp. 395–409. Springer, Heidelberg (2006)
16. Perrig, A., Song, D.: Hash visualization: a new technique to improve real-world security. In: International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC) (1999)
17. Prasad, R., Saxena, N.: Efficient Device Pairing using Human-Comparable Synchronized Audio Visual Patterns. In: Bellovin, S.M., Gennaro, R., Keromytis, A.D., Yung, M. (eds.) ACNS 2008. LNCS, vol. 5037, pp. 328–345. Springer, Heidelberg (2008)
18. Roth, V., Polak, W., Rieffel, E., Turner, T.: Simple and effective defenses against evil twin access points. In: ACM Conference on Wireless Network Security (WiSec) (2008)
19. Saxena, N., Ekberg, J.-E., Kostiainen, K., Asokan, N.: Secure device pairing based on a visual channel. In: IEEE Symposium on Security & Privacy, short paper (2006)
20. Saxena, N., Uddin, M.B., Voris, J.: Universal Device Pairing using an Auxiliary Device. In: Symposium On Usable Privacy and Security (SOUPS) (2008)
21. Soriente, C., Tsudik, G., Uzun, E.: BEDA: Button-Enabled Device Association. In: International Workshop on Security for Spontaneous Interaction (IWSSI) (2007)
22. Soriente, C., Tsudik, G., Uzun, E.: HAPADEP: Human Asisted Pure Audio Device Pairing. In: International Information Security Conference (ISC), Taipei, Taiwan (September 2008)
23. Stajano, F., Anderson, R.J.: The resurrecting duckling: Security issues for ad-hoc wireless networks. In: Security Protocols Workshop (1999)
24. Suomalainen, J., Valkonen, J., Asokan, N.: Security associations in personal networks: A comparative analysis. In: Stajano, F., Meadows, C., Capkun, S., Moore, T. (eds.) ESAS 2007. LNCS, vol. 4572, pp. 43–57. Springer, Heidelberg (2007)
25. Uzun, E., Karvonen, K., Asokan, N.: Usability analysis of secure pairing methods. In: Dietrich, S., Dhamija, R. (eds.) USEC 2007. LNCS, vol. 4886, pp. 307–324. Springer, Heidelberg (2007)
26. Vaudenay, S.: Secure communications over insecure channels based on short authenticated strings. In: Shoup, V. (ed.) CRYPTO 2005. LNCS, vol. 3621, pp. 309–326. Springer, Heidelberg (2005)
27. Čagalj, M., Čapkun, S., Hubaux, J.-P.: Key agreement in peer-to-peer wireless networks. Proceedings of the IEEE 94(2), 467–478 (2006)