Universal Device Pairing using an Auxiliary Device

Nitesh Saxena
Polytechnic University
Brooklyn, NY 11201
nsaxena@poly.edu

Md. Borhan Uddin
Polytechnic University
Brooklyn, NY 11201
borhan@cis.poly.edu

Jonathan Voris
Polytechnic University
Brooklyn, NY 11201
jvoris@cis.poly.edu

ABSTRACT

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

One of the main challenges of secure device pairing is the lack of good quality output interfaces as well as corresponding receivers on devices. In [13], we presented a pairing scheme which is universally applicable to any pair of devices (such as a WiFi AP and a laptop, a Bluetooth keyboard and a desktop, etc.). The scheme is based upon the device user(s) comparing short and simple synchronized audiovisual patterns, such as "beeping" and "blinking". In this paper, we automate the (manual) scheme of [13] by making use of an auxiliary, commonly available device such as a personal camera phone. Based on a preliminary user study we conducted, we show that the automated scheme is generally faster and more user-friendly relative to the manual scheme. More importantly, the proposed scheme turns out to be quite accurate in the detection of any possible attacks.

Keywords

Distributed Protocols, Mobile/Ad-Hoc Systems, Authentication, Security.

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 comes an increase in security risks. Wireless communication channels are easy to eavesdrop upon and manipulate. Therefore, a fundamental security objective is to secure these communication channels. In this paper we will use the term "pairing" to refer to the operation of bootstrapping secure communication between two devices connected via a short-range wireless channel. Examples of pairing from day-to-day life include associating a WiFi laptop with an access point, a Bluetooth keyboard and a desktop, and so on. Pairing would be easy to achieve if there existed a global infrastructure enabling devices to share an on- or off-line trusted third party, certification authority, 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.[1]

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

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

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

Earlier pairing protocols required at least 80 to 160 bits of data to be transmitted over the OOB channel(s). The simplest such protocol [1] involves devices exchanging their public keys over the wireless channel and authenticating them by exchanging (at least 80-bit long) hashes corresponding to the public keys over the OOB channels. The more recent so-called SAS- (Short Authenticated Strings) based protocols, [8] and [11], reduce the length of data

Symposium on Usable Privacy and Security (SOUPS) 2008, July 23–25, 2008, Pittsburgh, PA USA
.

[1] The problem has been at the forefront of various recent standardization activities, see [20].

transmitted over the OOB channels to approximately 15 bits.[2]

Based on the protocols listed above, a number of pairing schemes with various OOB channels have been proposed. These include schemes based on two bidirectional d2d infra-red channels [1], two bidirectional d2d visual channels consisting of barcodes and photo cameras [10], a unidirectional d2d visual channel consisting of blinking LED and video camera plus a unidirectional d2h channel consisting of a blinking LED and a unidirectional h2d channel [15], and two audio/visual d2h channels consisting of MadLib sentences and displayed text [7]. In addition, the SAS protocols trivially yield pairing schemes involving two bidirectional d2h and h2d channels – the user reads 15 bits of data displayed on one device and inputs it on the other, and vice versa. Most recently, [21] performed user studies of pairing schemes based on user comparing the data transmitted over two independent d2h SAS channels.

The aforementioned schemes have varying degrees of usability and are applicable to different device combinations. However, all the above schemes become inapplicable in pairing scenarios where

1. both devices do not have good quality transmitters (such as displays, speakers, etc.), and

2. both devices do not have the necessary receivers (such as cameras, microphones, etc.).

Notice that pairing scenarios involving most common commercial devices, such as laptops and access points, would fall into this category.

A very recent proposal, [17], focuses on pairing two devices with the help of "button presses" by the user. The scheme can be used to pair devices with minimal hardware interfaces (only an LED and a button are required) using one of the SAS protocol. However, as we discuss in the next section and as indicated in the results of [17], the scheme is a bit slow.

In [13] we presented a pairing scheme universally applicable to any pair of devices. The scheme 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 scheme involves users comparing very simple audiovisual patterns, such as "beeping" and "blinking", transmitted as simultaneous streams which form two synchronized d2h channels.

We believe that the universality of a pairing scheme is an advantage in terms of security as well as usability as it can eliminate user confusion as to what process to follow while pairing the devices.

The results of [13] show that users are indeed able to compare OOB output using the Blink-Blink and Beep-Blink combinations. In practice, however, the manual comparison is quite likely to occasionally encounter errors. Though the scheme was designed to minimize the burden placed on users, they must still devote a significant amount of attention to the OOB output during the pairing procedure.

Our Contributions. We ask a question: would the efficiency, robustness and usability of the pairing process improve if users could make use of an auxiliary device to compare the audiovisual OOB patterns instead of manually comparing them as done in [13]? An example of such a commonly available auxiliary device is a personal camera phone. We answer this question affirmatively and present an automated version of the scheme of [13].

In our new scheme the auxiliary device does not need to be trusted with respect to any cryptographic data or shared secrets, nor does it need to communicate (over any in-band wireless channel) with the two devices being paired. Based on a user study we conducted, we show that the automated scheme is generally faster and more user-friendly as compared to the manual scheme. More importantly, the scheme turns out to be quite safe in that it has no false negatives. The proposed scheme is universally applicable to pairing any set of devices (such as access points, keyboards, and other devices that lack any displays) given a suitable auxiliary device.

Our contributions are twofold. First, we present the design and implementation of a scheme to automate the comparison of audiovisual patterns. Second, we evaluate this scheme via a usability study.

Organization. The rest of this paper is organized as follows. In Section 2, we review prior pairing schemes. In Section 3, we describe the security model and summarize relevant protocols. In Section 4, we present our scheme, followed by a description of its design and implementation. Finally, the results of our user tests are presented in Section 5.

2. RELATED WORK

There exists a significant amount of prior work on the general topic of pairing.

In their seminal work, Stajano, et al. [19] proposed to establish a shared secret between two devices using a link created through a physical contact (such as an electric cable). In many settings, however, 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. [1] extended this approach through the use of infrared as a d2d channel – the devices exchange their public keys over the wireless channel followed by exchanging (at least 80-bits long) hashes of their respective public keys over infrared. The main drawback of this scheme is that it is only applicable to devices equipped with infrared transceivers.

Another approach taken by a few research papers is to perform the key exchange over the wireless channel and authenticate it by requiring the users to manually and visually compare the established secret on both devices. Since manually comparing the established secret or its hash is cumbersome for the users, schemes were designed to make this visualization simpler. These include Snowflake mechanism [6] by Levienet et al., Random Arts visual hash [12] by Perrig et al. etc. These schemes, however, require high-resolution displays and are thus only applicable to a limited number of devices, such as laptops.

Based on the pairing protocol of Balfanz et al. [1], McCune et al. proposed the "Seeing-is-Believing" (SiB) scheme [10]. 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. Since the scheme requires both devices to have cameras, it is only suitable for pairing devices such as camera phones.

Goodrich, et al. [7], proposed a pairing scheme based on "MadLib" sentences. This scheme also uses the protocol of Balfanz et al. The main idea is to establish a d2h channel by encoding the data into a MadLib sentence. Device A encodes the hash of its public key into a MadLib sentence and transmits this over a d2h channel (using a speaker or a display); device B encodes the hash of the (received) public key from device A into a MadLib sentence and transmit this over a d2h channel (using a speaker or a display); the user reads and compares the data transmitted over the two d2h channels, and vice versa. The scheme, as proposed in the paper, requires four

[2] For the SAS-based authentication and related prior work, refer to [22].

d2h channels and the user needs to perform two comparisons. This is quite slow and tedious for the user. One can trivially improve the scheme by using a slightly modified protocol, the one where devices exchange the hash (of size at least 160-bits) of the concatenation of both public keys, after exchanging their public keys. The modified scheme would then require only one user comparison. Note that, however, the scheme is not applicable to pairing scenarios where one of the devices does not have a display or a speaker.

Note that the previously described schemes, with trivial modifications, can (and should) all be based upon one of the SAS protocols [8], [11]. Since the SAS protocols require only 15-bits of data to be transmitted over the OOB channel, such a migration will immensely improve the efficiency as well as the usability of these schemes.

Saxena et al. [15] proposed a new scheme based on visual OOB channel. The scheme uses one of the SAS protocols [8], 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.

A very recent proposal, [17], focuses on pairing two devices with the help of "button presses" by the user. The scheme described in the paper is based upon a protocol that first performs an unauthenticated Diffie-Hellman key agreement and then authenticate the established key using a short password. Such a short password can be agreed upon between the two devices via three variants using button presses. The first variant involves the user simultaneously pressing buttons on both devices within certain intervals and each of these intervals are used to derive 3-bits of the password (and thus with 5 button presses, the user is able to inputs the same password on both devices). In the other two variants, one device picks up a short password, encodes each 3-bit block of the password into the delay between consecutive flashing of the device's screen or its vibration. As one device flashes or vibrates, the user presses the button on the other device thereby transmitting the password from one device to another.

One drawback with the above scheme, as described in [17], is that its security is based upon the secrecy of the agreed upon password. At least the button presses and the flashing of the screen can possibly be recorded by a video camera and therefore, the secrecy of the password is not guaranteed. The scheme, however, can easily be based upon a SAS protocol in a straight-forward manner and be used for pairing devices which do not have good transmitters or receivers. Assuming that both devices have an LED and a button each, we can have them transmit their SAS values by blinking of the LED (on one device) and pressing of button (on the other) and vice versa. Unfortunately, this would be quite slow – to transmit a 15-bit SAS value, it will take about a minute in each direction (see the results of the scheme called "D-To-B" in [17]; users can possibly not perform simultaneous "blink-press" faster than 3-4 seconds). One could apply the protocol variant of Saxena et al. [15] to avoid transmission of SAS in the other direction thereby reducing the execution time to close to a minute.

Uzun et al. [21] carry out a comparative usability study of simple pairing schemes. They consider pairing scenarios where devices are capable of displaying 4-digits of SAS data. In what they call the "Compare-and-Confirm" approach, the user simply reads and compares the SAS data displayed on both devices. The "Select-and-Confirm" approach, on the other hand, requires the user to select a 4-digit string (out of a number of strings) on one device that matches with the 4-digit string on the other device. The third approach, called "Copy-and-Confirm", requires the user to read the data from one device and input it onto the other. These schemes are undoubtedly simple, however, the results of [21] seem to indicate that Select-and-Confirm and Copy-and-Confirm are error prone.

In [18], authors 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.

In [13] we presented a pairing scheme universally applicable to any set of devices, irrespective of hardware limitations. The scheme can be based on any of the existing SAS protocols and does not require devices to have good transmitters or any receivers, e.g., only a pair of LEDs is sufficient. The scheme involves users comparing very simple audiovisual patterns, such as "beeping" and "blinking", which are transmitted as simultaneous streams to form two synchronized d2h channels. Recall that we present an automated version of this scheme in this paper.

In an independent result [14], the authors present a scheme similar to the "blinking" scheme we presented in [13]. The scheme of [14] is aimed at the detection of "evil twin" access points in cafés, airport lounges, etc. The two schemes, however, differ significantly in their implementation and therefore in terms of user experience. Firstly, in the scheme of [14], the user controls the time period during which she compares each bit of the SAS data, by pressing and releasing a button on her device. The scheme of [13], on the other hand, is automatic in that this time period is a pre-determined experimental value. Secondly, in [14], the user's device needs to trigger the display of next bit on the other device by sending it a signal over the wireless channel. This requires k such signals for a k-bit long SAS and the user needs to verify whether or not these signals are delayed, dropped or injected. This is unlike the scheme of [13], where only one synchronization signal is sent between the two devices.

The use of an auxiliary device to provide security functionalities has been previously proposed, e.g., in [2, 9, 23, 5]. However, to the best of our knowledge, the use of auxiliary device in device pairing has not been suggested prior to this paper. In addition, unlike prior proposals [2, 9, 23, 5], our scheme does not require the auxiliary device to communicate with other devices, nor does it require it to be trusted with respect to any cryptographic keys.

3. SECURITY MODEL AND APPLICABLE PROTOCOLS

Our pairing protocols are based upon the following communication and adversarial model [22]. 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 (it is important to note that given a binary string 's' over the SAS channel, the adversary can delay/replay the whole of s, but not its individual bits). 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 [3]. 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: [11] and [8]. We depict the protocol of [11] in Figure 1. 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.

Similar to the scheme of [13], which requires the user(s) to manually compare the SAS strings transmitted in the form of "beeping" and/or "blinking", our automated scheme can also be based on any of the existing SAS protocols. This is because in each of these protocols the SAS messages are computed as a common function of the public keys and/or random nonces exchanged during the protocol. Thus the authentication is based upon whether the two SAS messages match or not (see Figure 1, [8]).

4. AUTOMATED COMPARISON OF AUDIO-VISUAL PATTERNS

Manual comparison of SAS data can be achieved using two combinations which we will refer to as Blink-Blink and Beep-Blink [13]. Basically, for the Blink-Blink combination both devices transmit their SAS data via a "blinking" LED. For the Beep-Blink combination one device encodes its SAS data using a "blinking" LED and the other encodes its SAS data using "beeping" through a speaker. In both combinations the SAS data is transmitted in a synchronized fashion for the user to compare. This synchronization can be achieved by one device sending a signal to the other over the wireless channel. Since this synchronization signal can be delayed, each d2h channel also requires an "END" marker which indicates the end of the SAS data. The two END markers also need to be compared by the user.

In this paper we build automated versions of these two schemes which we call Blink-Blink and Audio-Blink. The encoding of SAS data in these setups is performed in a manner very similar to the manual version of the scheme but the decoding and comparison of SAS data is performed by an "Auxiliary Third Device" (ATD), such as a camera phone, which possesses a video camera to capture the blinking LEDs and a microphone to capture the audio output.

1. **"Blink-Blink".** This combination requires both devices to have visual transmitters, the simplest of which are LEDs. In cases where the devices have good displays, one could use the whole screen or a part of it as a transmitter. The two devices encode their respective SAS strings into blinks of a green LED – a '1' bit corresponds to a "ON" period and a '0' bit to an "OFF" period. The two devices also encode the END markers by lighting up a red LED. The auxiliary device ATD captures the blinking LEDs using an on-board video camera, then decodes the SAS strings and compares them as well as the synchronization bits. The result of the SAS and SYNC comparison is indicated to the user, who then accepts or rejects the pairing session on both of the devices being paired.

2. **"Audio-Blink".** This combination requires one device to have an audio transmitter and the other to have a visual transmitter. One device, A, encodes its SAS data using two types of sounds S1, S2('0'/'1') and the END marker using sound S3, and the other device, B, encodes its SAS information through a blinking green LED and represents the END marker by lighting up a red LED. The ATD captures the blinking LEDs using its video camera and the sounds, S1,S2 and S3, using its microphone. It then decodes the two SAS strings and compares them as well as the synchronization bits. The result of the SAS and SYNC comparison is indicated to the user, who then accepts or rejects the pairing session on the two devices being paired.

The security of our scheme is equivalent to the security of the underlying SAS protocol under the assumption that the user correctly transfers the result of comparison from the ATD to the two devices being paired. Note that the ATD does not need to be trusted with respect to any cryptographic secret data, nor does it need to communicate with the two devices being paired. The ATD only needs to possess the correct software (for decoding and comparison of captured data) and hardware (a camera and/or a microphone, in addition to a display and/or a speaker).

5. DESIGN AND IMPLEMENTATION

In this section, we describe the design and implementation of our pairing schemes using the automated Blink-Blink and Audio-Blink combinations.

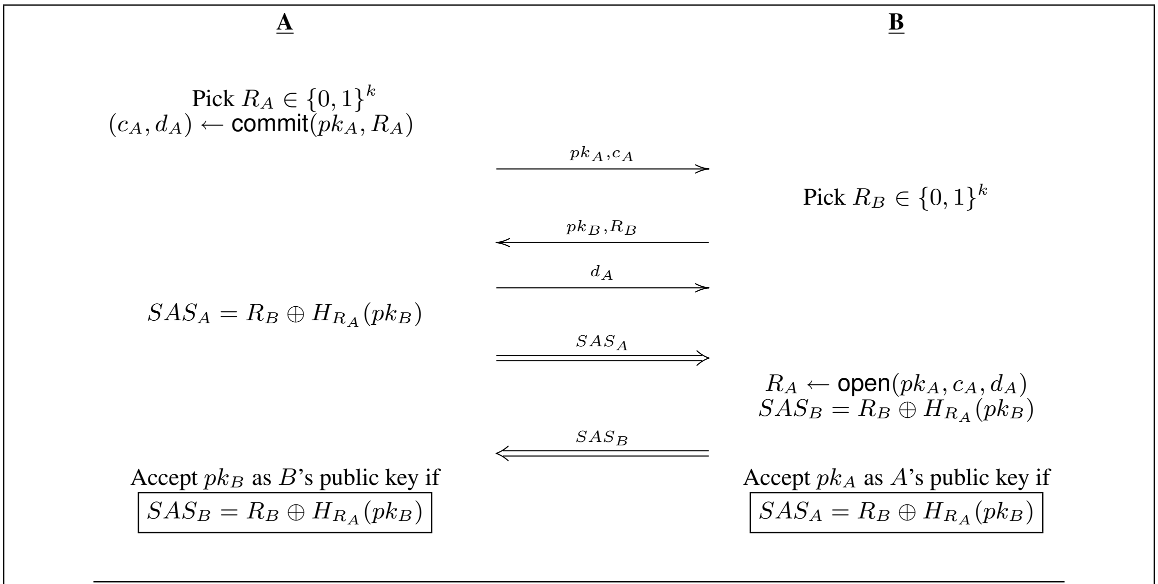
5.1 Hardware Requirements on Devices

For the Blink-Blink setup, the two pairing devices require transmitters in the form of two LEDs each. The ATD requires a good quality video camera as a receiver as well as a screen to present the result of the pairing operation to the user. In the Audio-Blink setup, the necessary transmitters are two LEDs on one pairing device and a speaker on the other. Therefore, in the Audio-Blink setup the ATD requires both a camera and a microphone as receivers. In addition, the ATD must have a screen or speaker to inform its user of the outcome of the pairing operation. Since it already has a video camera and a screen, such a device could also be used as a composite ATD to work with both the Blink-Blink and Audio-Blink setups. As a practical point, an ATD with "capturing" interfaces (such as a camera or microphone) and output interfaces (including a screen or speaker) on opposite sides of the device (as is the case with traditional cameras) are advantageous in terms of pairing usability.

Note that a crucial design element of our scheme is that the ATD does not need to communicate (over any wireless channel) with the devices being paired.

5.2 The Setup

Instead of working with real devices we chose to use the following simulated setup. We implemented both the Blink-Blink and Audio-Blink combinations using a simulator written in Microsoft Visual C#. The simulator has two components. The first is for transmitting the SAS data encoded as "blinking" and/or "speaking audio". The second part simulates the role of the ATD by auto-detecting the transmission, capturing, and comparison of the transmitted data, as well as outputting the pairing result on a screen or


A
B
Pick $R_A \in \{0,1\}^k$
$(c_A, d_A) \leftarrow \mathsf{commit}(pk_A, R_A)$
pk_A, c_A
Pick $R_B \in \{0,1\}^k$
pk_B, R_B
d_A
$SAS_A = R_B \oplus H_{R_A}(pk_B)$
SAS_A
$R_A \leftarrow \mathsf{open}(pk_A, c_A, d_A)$
$SAS_B = R_B \oplus H_{R_A}(pk_B)$
SAS_B
Accept pk_B as B's public key if
$\boxed{SAS_B = R_B \oplus H_{R_A}(pk_B)}$
Accept pk_A as A's public key if
$\boxed{SAS_A = R_B \oplus H_{R_A}(pk_B)}$

$\longleftrightarrow$: the wireless channel
$\Longleftrightarrow$: the OOB channel
pk_A, pk_B: (Diffie-Hellman) public keys of devices A and B
commit() and open(): functions of a commitment scheme based on random oracle model
$H()$: hash function drawn from an almost universal hash function family

Figure 1: The SAS protocol of [11]

through the speaker. To simulate the ATD we used a DELL Vostro 1500 laptop (Intel Core 2 Duo 1.6 GHz; 2 GB RAM) with a DELL Laptop Integrated webcam and microphone (see Figure 2). The webcam has 640×480 resolution and its frame rate is 30 frames per second. The integrated microphone and speaker of the laptop use the Sigmatel audio driver.

Figure 2: Audio-Visual Receiver of ATD: Laptop Camera and Microphone

To simulate the blinking devices performing the SAS transmission we set up 2 LEDs per blinking device on a breadboard (see Figure 4 for the Blink-Blink set-up) and integrated it with a desktop computer (Intel Xenon 1.8 GHz; 1 GB RAM) through a parallel port (DB25 connector). The Microsoft Windows "inpout32.dll" file was used to send data to the LEDs through the parallel port. We also needed a speaker to replicate the device's audio transmitter. The speaker was connected to the desktop machine's standard audio output. SAS data was encoded into audio output using the Microsoft SAPI 5.0 Text-To-Speech (TTS) engine, which represented bits as spoken English words ('one' for 1, 'zero' for 0 and 'stop' for END marker). Refer to Figure 3 for a picture of overall Blink-Blink and Audio-Blink setup.

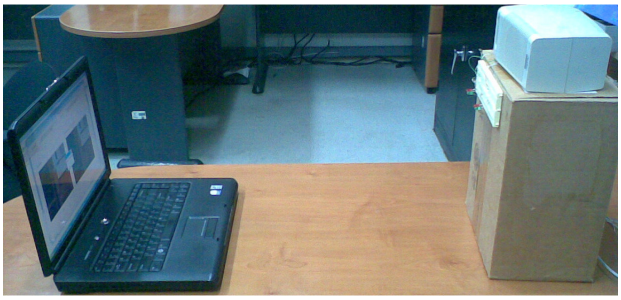


Figure 3: The Overall Setup of Blink-Blink and Audio-Blink

The simulated ATD's receiver was programmed using the Microsoft Windows "avicap32.dll" file to capture streaming video. The laptop webcam was set to preview mode so that video frames were transferred to the video stream buffer as soon as they were captured. For fast frame processing, the ATD simulation application locked the frame buffer and accessed it using direct memory references of pixel locations. The Microsoft Speech API (SAPI) 5.0 Speech Recognition (SR) engine was used to decode the incoming audio. The C#, "Environment.TickCount" variable was used to track the elapsed time on both devices.

5.3 The Role of the User

In our pairing scheme, the role of the user is limited to simply initiating the pairing process by pressing a button on one of the devices being paired, adjusting the camera or microphone of the ATD to focus on the transmitters of the devices being paired, and finally accepting or rejecting the pairing instance based on the output shown on the screen and/or output from the speaker of the ATD. The pairing process can be completed with the following steps for

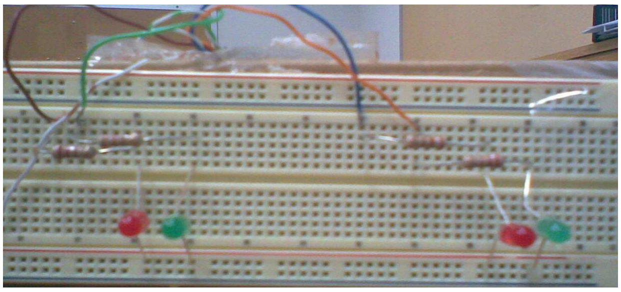


Figure 4: Set-up of Blink-Blink Transmitter: display of two devices on a Breadboard

both the Blink-Blink and Audio-Blink combinations:

1. The user presses a button on one of the pairing devices to start the execution of the SAS protocol.
2. When the devices are done with their SAS computation, they show their ready state (by lighting their (RED) signal LED(s)) to transmit SAS data.
3. The user adjusts the ATD's receiver(s) to focus on the transmitters of the pairing devices and activates the ATD receiver(s) by pressing a ready button on the ATD.
4. The user presses another button on one of the pairing devices, causing them to simultaneously transmit their SAS data.
5. The ATD detects the beginning of the SAS transmission spontaneously and captures the data from the transmitters of both pairing devices. At the end of the protocol session, the ATD decodes and compares the received data. If both the SAS data and the END markers sent by the two pairing devices match, the ATD outputs "Accept", otherwise it outputs "Reject".
6. Based on the Accept/Reject result indicated by the ATD, the user accepts or rejects the pairing instance for the two devices by pressing the appropriate buttons on both the pairing devices.

5.3.1 *Blink-Blink: Encoding using "Blinking" LEDs*

For the Blink-Blink combination, each pairing device is equipped with two LEDs which act as transmitters – one data LED and one synchronization (SYNC) LED. The data LED is used for SAS data transmission, while the SYNC LED is used to send the END marker. This marker is used at the beginning and end of SAS transmission to protect against delay attacks. The data and SYNC LEDs are of distinct colors. A green LED is used to send data and a red LED is used to transmit SYNC information. Thus a glowing green LED indicates a data bit of '1' and an unlit green LED represents a data bit of '0'. A glowing red LED, shown at the end of the SAS transmission, indicates presence of a SYNC bit. The red LED is also lit up prior to sending SAS data to inform users that the device has performed the necessary calculations and is ready to display its data. After completing the SAS data calculation, the devices being paired issue a ready signal by turning on their red LEDs. Next, the user turns on the ATD's camera, focuses it on the lit LEDs of the pairing devices, and presses a button on the ATD's camera to make it "ready" and enable video streaming. The user then presses a button on one of the pairing devices. This causes both devices performing the pairing to start transmitting their SAS data simultaneously through their green LEDs. Each frame of captured blinking LED video requires an experimentally determined value of 250 milliseconds to be successfully captured by the ATD's video camera. The data transmission is preceded by two additional video frames. We call the first of these "All-OFF". In this frame the data and SYNC LEDs of both pairing devices are in the OFF state (that is, not lit up). The second preliminary is dubbed "All-ON". In this frame all of both devices' LEDs are in the ON state (that is, lit up). Both these frames are needed for the receiver to learn where the devices' LEDs are located in the environment. Therefore, in order to transmit 15 bits of SAS data and one SYNC bit a total of 18 frames are required. This results in a transmission time of $250 \times 18 = 4500\ ms = 4.5\ sec$.

5.3.2 *Auto Detection of "Blinking" Data Transmission*

Prior to the first frame of data transmission (All-OFF) being detected, the video frames are "fast processed". This is a single pass process works using pre-adjusted threshold values. In other words, no threshold value adjustment takes place during fast processing. As such, it takes only 0-16 milliseconds of processing time per frame, which is less than the frame streaming rate (i.e., $\geq$ 33.33 milliseconds per frame). During this phase the threshold values are pre-adjusted in such a manner that the capturing of video frames (saving frames to memory from the video stream buffer) is deterministically triggered after the detection of data transmission. This occurs correctly when the pairing devices are within a certain distance of the ATD's inputs (about 2-3 feet).

The ATD's camera detects the beginning of data transmission by continuously monitoring the incoming video stream frames from the beginning of its ready state, which is indicated by the glowing of the red LED on each of the two pairing devices. Starting from the ready state all video frames are monitored to detect the All-OFF state of the LEDs of the devices being paired. If an LED switches from its OFF state to its ON state, its luminance increases while its saturation decreases, and vice versa. This property was used to detect the transition of the pairing device's LEDs from the ready state to the All-OFF state. The luminance (lum) and saturation (sat) of each pixel of each video frame is measured using the following algorithm, as depicted in [4].

```
Get the RBG values to the range 0-255
Find min and max values of R, B, G
L = (maxcolor + mincolor)/2
if(maxcolor==mincolor)
  S=0;
elseif(L < 128)
  S=((maxcolor-mincolor)*256)/(maxcolor+mincolor)
elseif(L >= 128)
  S=((maxcolor-mincolor)*256)/(256*2-maxcolor-mincolor)
```

When the user presses the "start session" button on the ATD, the camera takes the first frame (which captures the ready state with the red LEDs of the pairing devices turned on) and measures the saturation and luminance of each pixel of the frame. These measurements are stored in two byte arrays, one for luminance and the other for saturation. For each subsequent incoming frame, the lum and sat of each pixel are measured and compared against the ready state's lum and sat values for the same pixel. If a pixel matches the criteria for transition from the ON state to the OFF state (an increase in sat over the threshold value of SAT_TH and a decrease of lum by at least LUM_TH) the corresponding pixel is compared to groups of transitioning pixels within a certain proximity to each other. If there is a match with any existing groups, the pixel count of that group is incremented; otherwise a new group is created. So, after a single pass over all the pixels in a frame, the total number of groups created and their corresponding member counts is calculated.

Occasionally there are transient changes of luminance or saturation in small areas of video frames without any viable cause (i.e., no LED is actually present) which may cause a group to be created. However, a glowing LED will have a group with a larger member count of pixels than such anomalies. The groups with a number of members smaller than a threshold value MEM_COUNT_TH must thus be filtered out. Therefore, we end up with the total count of groups with more members than MEM_COUNT_TH. If the number of observed groups that meet this threshold condition is equal to the number of LEDs which changed their state from ON to OFF (i.e., that transitioned from the ready state to the All-OFF state) we know that we have received the All-OFF frame. This processing must be performed for every frame until the All-OFF frame is detected.

When the All-OFF frame is detected, the timestamps of future frames are pre-calculated by adding the transmission frame interval value to the base timestamp of the All-OFF frame. This value is calculated as follows: $(250\ milliseconds) \times current\_frame\_position$. The rest of the frames are captured at their corresponding timestamps as tracked by the C# "Environment.TickCount" variable.

5.3.3 Decoding of Blinking Data from Captured Video Frames

After the capturing of all the required frames is complete, the captured frames are "fine processed" to extract their SAS data. The number of LEDs present is determined via analysis of the All-OFF and All-ON frames based on their maximum color difference for all the pixels, $max(dR, dG, dB)$. This process is performed by building bit strings based on the threshold value for $max(dR, dG, dB)$ for each row of pixels. Consecutive or adjacent '1' values in the bit strings, over a certain threshold, indicate the presence of an LED. This technique is described in greater detail in [16].

If the calculated LED count does not match the actual amount of LEDs, the threshold for $max(dR, dG, dB)$ is adjusted. The LED counting process then starts over with this new threshold. This process continues a given number of times, until the LED count matches the number of LEDs which are actually present. If the exact number of LEDs are detected, the dimension and location of these LEDs are extracted and stored in memory. The RGB color values of the LED OFF and ON states are also extracted from the All-OFF and All-ON frames and stored in memory for use in determining the state of LEDs in the forthcoming data frames. In these data frames, only the pixels corresponding to LED locations and dimensions are explored to detect whether an LED is in its OFF or ON state. These pixels are matched against the initial RGB values of the LEDs ON and OFF states. In this way, SAS data is retrieved exploring the data frames. Similarly, data frames are checked for red LEDs to be in OFF state and final SYNC frame is checked- if all the green LEDs are in OFF state and red LEDs are in ON state or not. If SYNC and data LEDs are not in desired states in all frames, it is regarded as SYNC failure(natural or malicious). It requires less than one second for the simulator to decode the blinking data from all the captured frames.

5.3.4 Audio-Blink*: Encoding using Speech*

The Microsoft Speech API (SAPI) 5.0 Text-To-Speech (TTS) engine was used to encode SAS data into speech.[3] We used the English pronunciation of "one" to represent a data bit '1', the English pronunciation of "zero" for a data bit '0', and the English pronunciation of "stop" to act as an END marker i.e., a SYNC bit. During the transmission of the All-OFF and All-ON frames of the data transmission process, an audio "zero" and an audio "one" are transmitted respectively. Through preliminary experimentation we determined the time required to accurately transmit and detect each bit was 400 milliseconds with a SAPI "speech rate" setting of '+8'. It was difficult for the SAPI speech recognizer to detect the synthesized speech at a speech rate higher than this setting. This time interval, which is slower than the 250 millisecond interval that was used for Blink-Blink was necessitated by the relatively slower rate at which the SAPI TTS and SR engines could operate on the SAS data. Thus the Audio-Blink setup requires $400 \times 18 = 7200\ ms = 7.2\ sec$ of transmission time.

5.3.5 Decoding of Speech

We used the Microsoft SAPI Speech Recognizer (SR) to detect the SAS data transmitted as English speech. We created a grammar containing three rules to recognize the spoken words "one", "zero" and "stop". After detecting the captured speech, it is then converted to a bit string. The speech decoding is done by an event generator which detects each separate word and "live decodes" the English words into a bit string. The timing of each bit is compared to its calculated expected arrival time. If a transmitted bit is delayed beyond a certain threshold value as determined by the SR engine's delay tolerance, the corresponding bit will cause a SYNC mismatch to be reported as the result of the pairing process. Thus, a delayed SYNC bit is detected by comparing an observed bit's arrival time to its anticipated arrival times, while a missing SYNC bit is caught by monitoring for absence of the "STOP" signal.

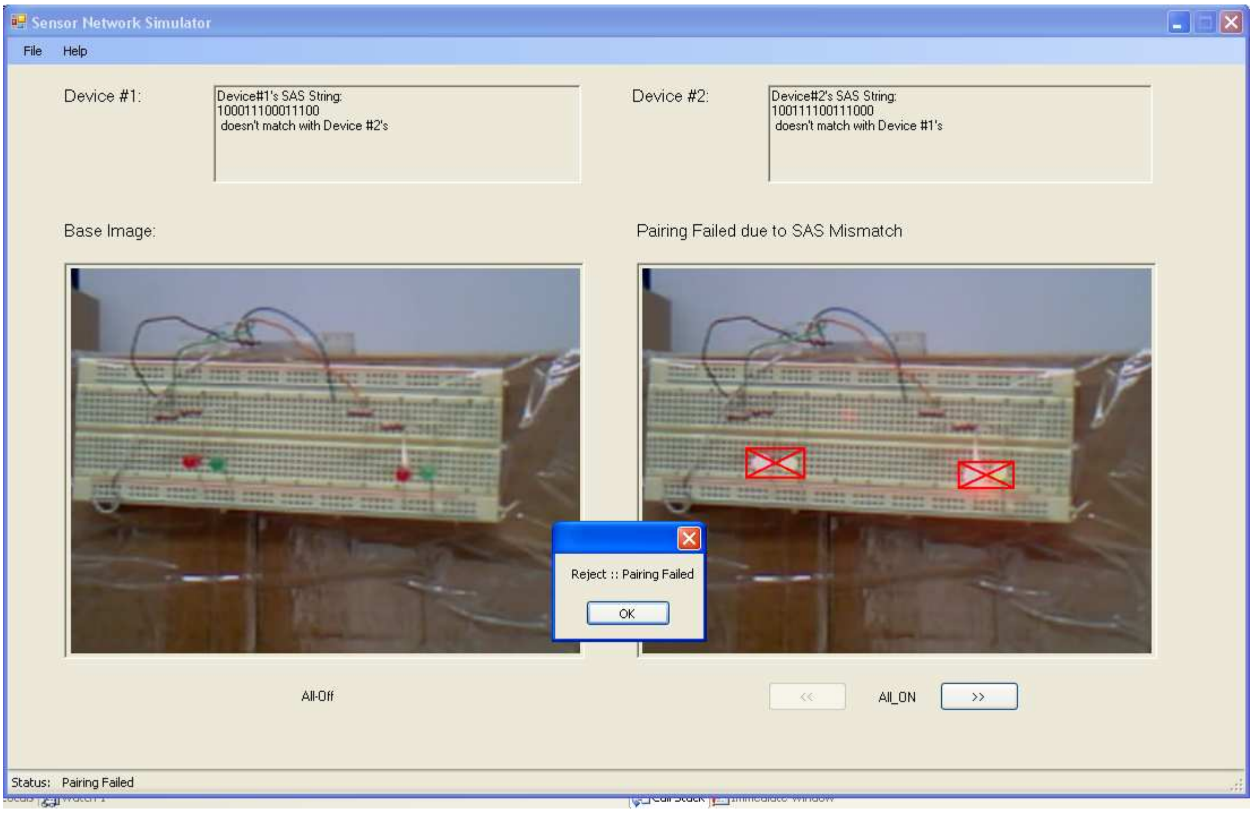
5.3.6 Indicating the Result of Comparison

The result of the comparison of the SAS data and SYNC bits, and thus whether the pairing succeeded or failed, is displayed on the screen of the ATD for both the Blink-Blink and Audio-Blink setups. For the Audio-Blink setup the result is also "spoken" through audio by the TTS engine on the ATD. The pairing procedure is successful if all of the OOB signals from the two devices being paired match. The OOB SAS data signals and SYNC signals must all match in order for the pairing instance to be considered a success. A successful pairing instance is indicated to the device user by drawing a green rectangle across the LEDs of the two devices whose OOB data matched. A failed pairing instance is presented to the user by drawing a red rectangle across the LEDs of the two devices whose OOB signals did not agree. The result of the pairing process is also provided through a message box and a status bar on the GUI of the ATD so the device user can easily detect the result. See figures 5 and 6 for pictures of this output from our implementation.

5.3.7 User Testing Interface

We implemented a usability testing interface on our device simulators to measure the usability of the Blink-Blink and Audio-Blink combinations using an ATD. Each step of user interaction during the automated pairing process was implemented on our two simulation computers to imitate the actions that would be necessary to

[3] We tried to train the engine to learn beeping sounds of different frequencies and later detect them based on the acoustic patterns of the beep. However, the SAPI is unable to learn and detect acoustic patterns. It is only able to detect English and a few other common languages.



Device #1:

Device#1's SAS String:
100011100011100
 doesn't match with Device #2's

Device #2:

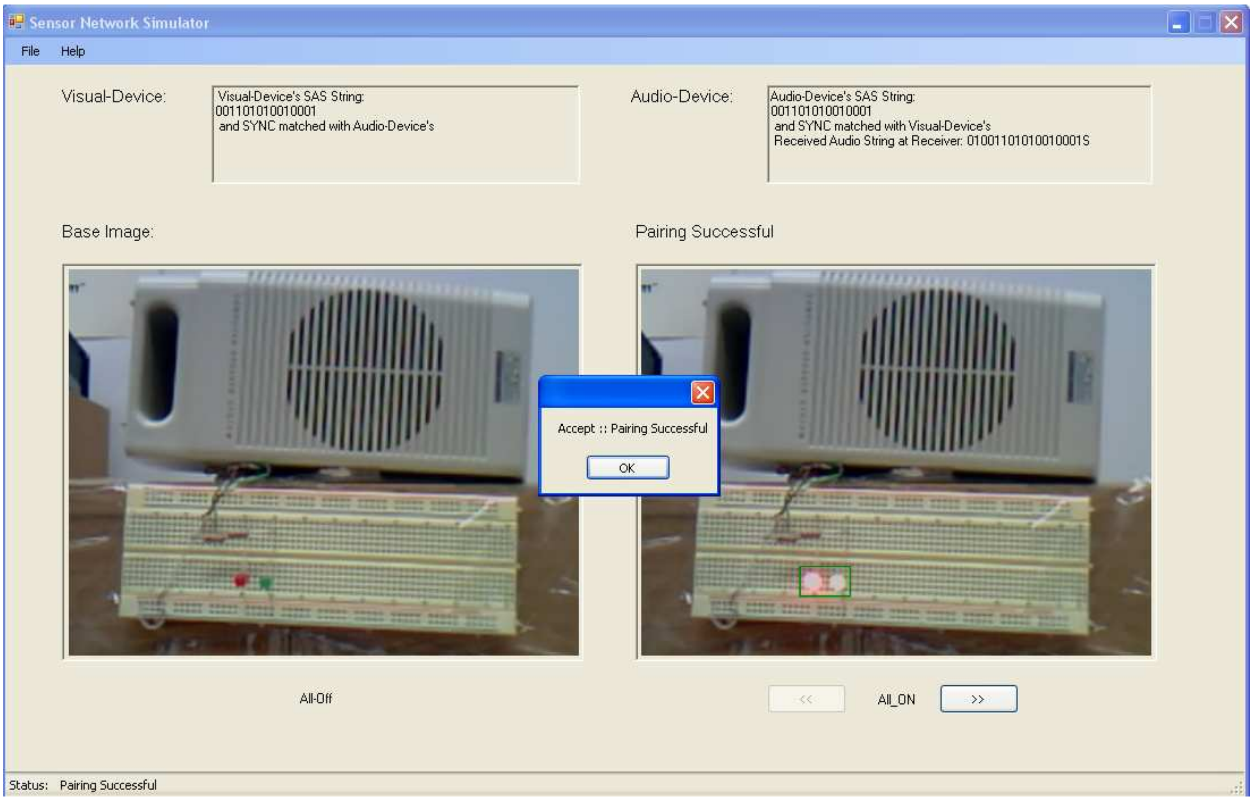
Device#2's SAS String:
100111100111000
 doesn't match with Device #1's

Pairing Failed due to SAS Mismatch

Reject :: Pairing Failed

Status: Pairing Failed

Figure 5: Result of the Blink-Blink Setting: "Failed Pairing"



Visual-Device:

Visual-Device's SAS String:
001101010010001
 and SYNC matched with Audio-Device's

Audio-Device:

Audio-Device's SAS String:
001101010010001
 and SYNC matched with Visual-Device's
 Received Audio String at Receiver: 01001101010010001S

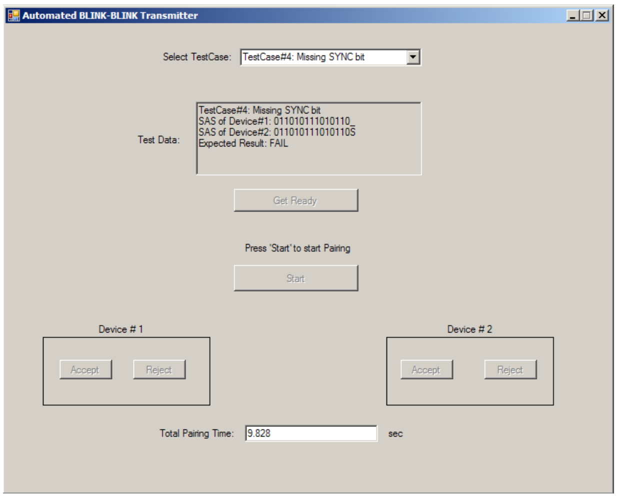
Pairing Successful

Accept :: Pairing Successful

Status: Pairing Successful

Figure 6: Result of the Audio-Blink Setting: "Successful Pairing"

perform the pairing process on actual devices. Our test implementation also included the measurement of user timing, that is, the time it took users to execute the pairing protocol via the simulators. The result of each testing instance, including this execution time, was stored in log files for analysis of the usability of the automated pairing schemes. See figure 7 for an example of the testing GUI.


Automated BLINK-BLINK Transmitter
Get Ready
Press 'Start' to start Pairing
Start
Device # 1
Accept
Reject
Device # 2
Accept
Reject

Select TestCase: TestCase#4: Missing SYNC bit

Test Data:
TestCase#4: Missing SYNC bit
SAS of Device#1: 011010111010110_
SAS of Device#2: 011010111010110S
Expected Result: FAIL

Total Pairing Time: 9.828 sec

Figure 7: Automated Transmitter and User Timing Interface

5.3.8 Comparison with the Manual Schemes

Manual versions of the Blink-Blink and Beep-Blink combinations were also implemented (with setup and timing intervals similar to those described in [13]). The user testing functionality, as well as the timing of user pairing, was also implemented. The recorded user testing data was stored in log files for future analysis of user preferences and comparison of the automated and manual device pairing schemes.

5.4 Usability Testing

Despite the automated nature of this scheme, assessing its usability was still an important aspect of the system's development. We performed user testing to answer a number of critical questions regarding the automated pairing process. First and foremost, we wished to verify that the automated process could be used to reliably detect matches and mismatches in the SAS data. That is, we wanted to make sure that the automated setup minimized, if not eliminated, all *safe errors* (i.e. false positives, or identifying a match as a mismatch) and *fatal errors* (i.e. false negatives, or identifying a mismatch as a match) in the pairing procedure [21]. Secondly, we wanted to gain a qualitative sense of whether users thought the addition of an ATD improved the ease and comfort of the pairing operation. Next, user testing was performed to measure how long it took users unfamiliar with the system to complete a pairing operation. Finally, tests were conducted to check that the pairing scheme was robust when used on devices with different speaker volumes and at various distances from the camera and microphone inputs of the ATD.

Both the automated and manual pairing setups were tested by 20 subjects. All of our testers were college students interested in working with new technology but not familiar with the underlying theory of the pairing procedure. Each tester was given a short summary of the secure device pairing scheme and its potential applications. Additionally, the testers were provided with both verbal and written instructions explaining how to operate the test devices in the automated and manual setups. The subjects did not receive specialized training of any kind prior to performing the tests, however.

5.5 Testing Setup and Test Cases

The manual and automated experiments for both the Blink-Blink and Audio-Blink output combinations were conducted in a graduate research lab of our university. Test volunteers were asked to perform four kinds of tests. These were the automated setup with the Blink-Blink combination, the automated arrangement with the Audio-Blink combination, the manual system with the Blink-Blink combination, and the manual layout with the Beep-Blink combination. For both combinations under the automated setup, 3 test cases were designed to test for matches and mismatches in SAS strings and 2 test cases were performed to test for synchronization delays, that is, END marker or SYNC bit mismatches.

In addition, 2 test cases were performed for the automated Blink-Blink combination with the ATD at different distances from the simulated pairing devices to test the robustness of the ATD's camera input. Normal test cases were carried out with the pairing devices at a distance of approximately 16 inches from the ATD's camera, while the distance test cases were done with the devices set about 10 inches away and 30 inches away. For the Audio-Blink combination, in place of the Blink-Blink distance tests, two volume test cases were designed to gauge the robustness of the ATD's microphone input. The standard test cases were created with the pairing device's volume set to medium, while one volume test was done with the volume on the lowest setting and another on the highest setting. In order to provide our test subjects with a basis for comparison, we also performed manual tests of the Blink-Blink and Beep-Blink combinations identical to those done in [13]. However, to obtain a fair and meaningful comparison between the manual and automated schemes, we did not train the subjects with a matching learning instance for the Beep-Blink combination, as was done in [13]. For these tests we used the exact same test cases as the automated tests except for the distance and volume ones. To summarize, 7 test cases were administered for both output combinations using the ATD and 5 test cases were given for both combination types without the ATD, for a total of 24 test cases per subject.

The SAS data utilized in these test cases were randomly generated, but fixed from test volunteer to test volunteer to prevent some users from receiving strings that were easier to identify than others (as an example, most users would not have a problem differentiating between the encoding of "0000" and "1111", while the more subtle difference between the strings "1010" and "0101" might be harder to notice). During the manual tests, these strings were presented to the test subjects in a random order. This was done to minimize the effects of learning and fatigue on the test results. In other words, we wanted to prevent users from anticipating future test cases based on previous ones or losing motivation to pay proper attention during the pairing procedure.

No bits were prepended to the SAS data for "padding" unlike the tests in [13]. Such padding was designed to provide human users with a chance to focus their attention on the OOB channel transmitting the SAS information, which is superfluous when the SAS data is transmitted to a device instead of a human user. The padding bits were also omitted from the manual test cases strings to keep as close of an equivalence as possible between the manual and automated test setups. Thus each test case string consisted of just the 15 bits of SAS data and nothing else.

As discussed in the Section 5, the two devices performing the pairing operation were simulated using LEDs connected to the parallel port and a speaker of a desktop computer. A laptop with a built in camera and microphone acted as the ATD. When the Blink-Blink combination was in use, four LEDs (two green for transmitting data and two red for the END marker) were used to simulate the two devices being paired, while the Audio-Blink combination required two LEDs (one green for data, the other red for the END marker) to represent one device and a speaker to act as the other. The manual test cases were administered similarly to those of [13]. As described in Section 5.3, users initiated a pairing session by clicking a "Get Ready/Start" button on the desktop machine. This caused the devices to light up their red END marker LEDs, indicating that they were ready to begin transmitting their SAS signal. Next, users clicked the "Start" button on the desktop to initiate the blinking of the attached LEDs and/or sound output, as per the Blink-Blink or Audio-Blink channels being tested.

The automated test cases proceeded similarly, with the addition of the laptop posing as the ATD. Users also initiated the automated tests by clicking the desktop's "Get Ready/Start" button. Users next focused the laptop's camera on the newly lit LEDs, being careful to check that all LEDs were visible on the camera's display. Next, a "Start Session" button was pressed on the laptop to start the capturing of video and/or sound output from the desktop simulating the two pairing devices. After this, the "Start" button on the desktop was clicked to begin the transmission of the SAS data. After the laptop recorded the SAS data, it reported an "accept" or a "reject" status message. The final duty of the tester was to click two buttons on the desktop machine, one for each simulated device, corresponding to this status message.

5.6 Test Timing and Results.

For manual comparison, the output interval that users felt most comfortable and committed the least errors with was established in [13]. These values were 500 millisecond for the Beep-Blink combination and 800 millisecond for the Blink-Blink combination. This left us to determine the optimal interval for automated comparison using the camera and microphone inputs of the ATD. The signal duration for the automated configuration was not dependant on the perception of human users, but instead on the capture rate of the ATD's inputs. In other words, we had to determine how quickly the SAS data output by the pairing devices could be captured by the ATD. This ideal output speed was determined by starting with the intervals used in manual comparisons and decreasing it by 50 millisecond until the ATD's camera started missing LED blinks or the microphone was unable to pick up all of the transmitted sound. As stated in Sections 5.3.1 and 5.3.4, we discovered the best intervals for automated SAS comparison to be 250 millisecond for the Blink-Blink combination and 400 millisecond for the Audio-Blink combination.

The results of our tests with the automated and manual test setups are presented in Tables 1 and 2 respectively. At no point during our tests did a user make an error in transferring the result indicated by ATD onto the devices being paired. That is, users were able to reliably transfer the one bit result ("accept" or "reject") from the ATD to the devices being paired. Therefore, we only consider errors that occur at the ATD for the automated setup. When manually comparing the audiovisual patterns users committed a few safe errors and also committed several of the more dangerous fatal errors. Out of the 100 manual Blink-Blink tests performed, 2 safe errors and 2 fatal errors were committed causing both error rates to be 2.00%. While only one safe error occurred (yielding a safe error rate of 1.00%) during the manual comparison of the Beep-Blink combination, its fatal error rate was higher. For this combination fatal errors occurred 20 times out of 100 tests, or at a rate of 20.00%.

When aided by the ATD, on the other hand, users committed no fatal errors whatsoever. The only errors to occur when using the automated setup were safe ones. When using the Blink-Blink output, only two safe errors occurred over the course of 140 tests, yielding an error rate of 1.43%. Safe errors were a more common occurrence with the Audio-Blink output. Using this combination, 10 safe errors occurred out of 140 tests, producing an error rate of 7.14%. Since more safe errors occurred with the automated Audio-Blink combination than with the Blink-Blink type of output, the probable cause of these errors is the microphone picking up background noise which the ATD confused for SAS data. This claim is further substantiated by the fact that 4 of the 10 safe Audio-Blink errors took place with the pairing device's speaker set to a lower volume. The automated pairing setups were robust to volume increases as well as changes in distance, however. As long as the LEDs of the pairing devices were discernible on the viewfinder of the ATD, it was able to accurately record the SAS output.

The automated Blink-Blink combination took the least time for users to execute, with an average pairing time of 13.079 seconds. Using an ATD added some time to the operation of the pairing protocol, but this was counteracted by the very fast 250 millisecond period at which the device's camera could monitor the SAS data output. When users manually detected the SAS data, the pairing process averaged 20.983 seconds, which is 7.904 seconds slower than the automated scheme. This is because human users require an 800 millisecond interval to comfortably and accurately follow the Blink-Blink SAS output. For the Audio-Blink combination, on the other hand, the automated setup took 15.261 seconds, which was 1.678 seconds longer than the 13.583 second average runtime of the manual Beep-Blink pairing process. This is due to the fact that the ATD required a 400 millisecond interval to accurately detect SAS data in audio form, which is not much of a speedup over the 500 millisecond which human users found comfortable. The 100 millisecond interval speedup was not enough to compensate for the additional steps needed to manipulate the ATD during the automated pairing process.

After the tests were completed users were asked to answer a very brief questionnaire in order to qualitatively gauge which pairing setup they preferred. Users were posed a single question, "If you had to use a pairing scheme on a daily basis, which would you prefer - the manual setup or the automated setup?" 80% of the test subjects (i.e., 16 out of 20 of them) responded that they would prefer to use the automated scheme over the manual scheme. This suggests that users were willing to deal with a small amount of additional pairing steps in order to not have to monitor the SAS output as closely.

6. DISCUSSION

We can draw the following conclusions from the results of our tests using an ATD to aid in the process of pairing two devices using either the Blink-Blink combination or the Audio-Blink combination. Our results indicate that using an ATD makes the pairing process safer and less burdensome for users. However, this comes at the cost of additional safe errors (albeit low in number) due to occasional signal processing errors by the ATD.

Security. One of the restrictions of a pairing system that utilizes human comparison of OOB output is that the length of data that can be transmitted over these channels is limited by the bit string length that users can reliably and comfortably compare. The SAS proto-

Combination	Average User Timing (seconds)	% Safe Error Rate	% Fatal Error Rate
Blink-Blink	13.079 (*sd*[a]*=3.524*)	1.43	0.00
Audio-Blink	15.261 (*sd=3.387*)	7.14	0.00

[a] Estimated Standard Deviation from the sample

Table 1: Results of User Tests based on Automated Comparison

Combination	Average User Timing (seconds)	% Safe Error Rate	% Fatal Error Rate
Blink-Blink	20.983 (*sd=3.107*)	2.00	2.00
Beep-Blink	13.583 (*sd=2.659*)	1.00	20.00

Table 2: Results of User Tests based on Manual Comparison

cols make it possible to transmit a sufficient number of bits of data to provide security comparable to ATM PIN authentication. Obtaining a level of security beyond this, however, would require the transmission of longer bit strings that would tax users' ability to tell the difference between pattern matches and mismatches. The core advantage that the automated pairing scheme has over the manual setup is that it does not rely on human comparison of SAS data while retaining the universality of the manual scheme. Therefore the automated pairing system makes it possible to obtain stronger security by transmitting longer strings of SAS data that would be beyond the capability of the manual setup.

On the other hand, there are some security considerations that must be taken into account for the automated setup that do not apply to the manual system. One of these is that when the Audio-Blink combination is employed users must make sure that their ATD is picking up audio from the pairing device producing the audio output and not somewhere else in the environment. If this is not done, an adversary could potentially launch an attack by injecting audio from its own device into the pairing process. Thus, while using Audio-Blink combination in a noisy environment, with a variety of devices around, the users need to pay close attention in determining the source of the audio recorded by their ATD.

Fatal Pairing Errors. Our results show that applying an ATD makes the pairing procedure safer because it completely eliminates all fatal errors. These errors are dangerous because, when committed, a user has accepted a pairing instance where something has gone wrong due to either an accidental error or a malicious occurrence.

When performing pairing using manual comparison, fatal errors can also be reduced via user training, as was shown with the Beep-Blink combination in [13]. As an example of such training, [13] demonstrated that the high fatal error rate observed in the Beep-Blink combination can be significantly reduced if the user is given one matching learning instance. However, training-based solutions can not be relied upon to completely eliminate all fatal errors. In practice, it is quite likely that a user will end up committing these errors no matter what, e.g., due to a slight distraction while performing the manual process. Thus, the strongest advantage of the automated pairing system is that it accomplishes the complete elimination of fatal errors where other solutions can not.

Safe Pairing Errors. Safe errors, on the other hand, are relatively benign because users can simply run another pairing session after one has occurred. Moreover, we believe that with higher quality ATD receivers and improvements to the audio and visual detection algorithms, the amount of safe errors can be further reduced, if not eliminated altogether. Our results show that the ATD is already robust in terms of the positioning of the pairing devices relative to the ATD. With a more sophisticated audio detection algorithm the number of safe Audio-Blink errors could be reduced as well. Either way, the safe errors that occur during the automated pairing process are far less problematic than the fatal errors that occur without the help of an ATD.

More safe errors occurred when the automated Audio-Blink combination was used than when the manual Beep-Blink combination was employed. There are two reasons for this. One is that the SAPI used by the ATD to perform the automated audio decoding is more susceptible to ambient noise than a human who manually "decodes" the audio signal. Another is that the SAPI's decoding rate is not constant. This occasionally caused the audio SAS data to be received slightly late, which was interpreted as a delay attack by the ATD.

Usability. Furthermore, use of an ATD reduces the burden placed on users during the pairing process. A large majority of users preferred the automated pairing setup because they did not have to focus nearly as intently on the devices performing the pairing when using the automated scheme as they did when using the manual setup. Since one of the essential qualities of a good pairing procedure is usability, this is another key advantage of the automated testing system. Another noteworthy insight is that users can easily transfer a one bit result("accept" or "reject") from one devices to another. This was consistently done by our test subjects when they transferred the pairing outcome from the ATD to the pairing devices.

We also want to note that the usability of our automated schemes is expected to improve with a real implementation involving a smartphone. Recall that our current proof-of-concept simulated set-up is a bit crude in terms of ATD functionalities.

Pairing Speed. Whether the ATD is a help or a hindrance in terms of pairing speed depends on the difference between the rate at which the ATD can process the SAS data and the speed at which human users can comfortably monitor the SAS output. For the Blink-Blink combination, the automated configuration took less time to use than the manual one. This is because the ATD could monitor visual input at a much higher rate (250 ms) than a human user (800 ms). When using the Audio-Blink output, however, the ATD slowed the process down slightly because it could not keep track of audio data much faster (400 ms) than a human user (500 ms) and required extra steps to execute. Clearly, the speed benefit imparted by the use of a ATD is dependant on the particulars of the OOB channel used to transmit the SAS data.

7. CONCLUSION AND FUTURE WORK

To conclude, there are several advantages to using an ATD to pair two devices. The input and output interfaces that are required to automate the pairing process are increasingly common on personal devices such as palmtops, PDAs and "smart" phones. As these devices become more ubiquitous, it will therefore be more likely that users will have a suitable device present to take advantage of during the pairing process. Using an ATD makes it possible to provide stronger security than would be possible through manual comparison by using longer SAS. Furthermore, an ATD can be used more efficiently to pair devices having simultaneous multiple-bit output interfaces, such as a single device using more than one data LEDs or speaker producing multiple frequency tones or audio outputs(i.e., multiple bits of SAS data transmission at a time). This increase in parallel OOB output would be overwhelming for a human to compare but could easily and efficiently be handled by an ATD. Finally, using an ATD to automate the pairing procedure in no way hinders the universality of the pairing scheme. The pairing devices can still use input and output interfaces that are of low quality, and an ATD can handle two different forms of SAS output (such as audio and visual output in the case of the Audio-Blink combination) at the same time without any special alterations.

In a practical environment it would probably make more sense not to restrict users to the manual or automated setup alone. In many situations users may not have access to an ATD that features receivers of the type and quality necessary to automate the pairing process. In such circumstances a user has no choice but to manually pair the devices. When a suitable ATD is available, however, our tests demonstrate it to be beneficial to the pairing process by reducing dangerous errors and increasing usability. Recall that both the manual and automated schemes (given a suitable ATD) are universally applicable to any pairing scenario, as these schemes only require the pairing devices to have low-cost and commonly available hardware interfaces.

In our future work, we plan to test both the manual and automated schemes more rigorously and with a wider range of subjects. In addition, we will explore other simpler and more usable ways to leverage an auxiliary device for performing the pairing operation. As a related work, we also plan to perform a comparative usability study of most existing pairing techniques.

Acknowledgements

The authors would like to thank the anonymous reviewers for their useful comments.

8. REFERENCES

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

[2] M. Burnside, D. Clarke, B. Gassend, T. Kotwal, S. Devadas, and R. Rivest. The untrusted computer problem and camera-based authentication. In *Pervasive Computing (Pervasive)*, 2002.

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

[4] J. D. Foley and V. D. Andries. *Fundamentals of Interactive Computer Graphics. 2nd Edition*. Addison-Wesley, Reading, Massachusetts U.S.A., 1990.

[5] E. Gieseke and J. McLaughlin. Secure web authentication with mobile phones using keyed hash authentication. CSCI E 170 Final Project, Harvard University Extension, 2005.

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

[7] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and Clear: Human-Verifiable Authentication Based on Audio. In *International Conference on Distributed Computing Systems (ICDCS)*, 2006.

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

[9] A. Madhavapeddy, D. Scott, R. Sharp, and E. Upton. Using camera-phones to enhance human-computer interaction. In *Ubiquitous Computing (Adjunct Proceedings: Demos)*, 2004.

[10] 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.

[11] S. Pasini and S. Vaudenay. SAS-Based Authenticated Key Agreement. In *Theory and Practice of Public-Key Cryptography (PKC)*, 2006.

[12] A. Perrig and D. Song. Hash visualization: a new technique to improve real-world security. In *Cryptographic Techniques and E-Commerce (CrypTEC)*, 1999.

[13] R. Prasad and N. Saxena. Efficient device pairing using human-comparable synchronized audiovisual patterns. In *Applied Cryptography and Network Security (ACNS), to appear*, 2008.

[14] V. Roth, W. Polak, E. Rieffel, and T. Turner. Simple and effective defenses against evil twin access points. In *ACM Conference on Wireless Network Security (WiSec), short paper*, 2008.

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

[16] N. Saxena and M. B. Uddin. Device pairing using unidirectional physical channels. In *Mobile and Wireless Networks Security (MWNS)*, 2008.

[17] C. Soriente, G. Tsudik, and E. Uzun. BEDA: Button-Enabled Device Association. In *International Workshop on Security for Spontaneous Interaction (IWSSI)*, 2007.

[18] C. Soriente, G. Tsudik, and E. Uzun. Hapadep: Human asisted pure audio device pairing. Cryptology ePrint Archive, Report 2007/093, 2007.

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

[20] J. Suomalainen, J. Valkonen, and N. Asokan. Security associations in personal networks: A comparative analysis. In *European Workshop on Security and Privacy in Ad hoc and Sensor Networks (ESAS)*, 2007.

[21] E. Uzun, K. Karvonen, and N. Asokan. Usability analysis of secure pairing methods. In *Usable Security (USEC)*, 2007.

[22] S. Vaudenay. Secure communications over insecure channels based on short authenticated strings. In *International Cryptology Conference (CRYPTO)*, 2005.

[23] M. Wu, S. Garfinkel, and R. Miller. Secure web authentication with mobile phones. `http://dimacs.rutgers.edu/Workshops/Tools/abstract-wu-garfinkel-miller.pdf`.