ABSTRACT

The secure “pairing” of wireless devices based on out-of-band communication is an established research direction. Unfortunately, this approach is prone to human errors that lead to man-in-the-middle attacks. To address this and to better motivate the users, this paper proposes the use of computer games for pairing. Games make the pairing process enjoyable and engaging, thus improving its usability and security. The technical contribution of this work is a new pairing system called “Alice Says.” This is a game that achieves pairing based on the memory game Simon. The paper discusses the design and implementation of Alice Says. On a broader note, the paper also points to other problems in wireless and mobile security that are currently lacking optimal solutions, and suggests how games and entertainment can be applied to improve them.

1 Introduction

Short- or medium-range wireless communication – based on technologies such as Bluetooth, WiFi, and RFID (Radio Frequency IDentification) – is becoming increasingly popular and promises to remain so. This popularity unfortunately brings about various security risks. Wireless channels are easy to eavesdrop upon and manipulate. A fundamental security objective is therefore to secure them. In this paper, the term “pairing” refers to bootstrapping of secure communication between two wireless devices in a way that is resistant to eavesdropping and man-in-the-middle attacks. An example of a common use case for this operation is pairing a headset and phone or pairing two phones.

A promising research direction towards solving the pairing dilemma is to leverage an out-of-band (OOB) channel that is governed by human users. Examples of OOB channels include audio, visual, and tactile channels. Unlike classical radio channels, OOB channels are “human-perceptible,” i.e., the underlying transmission and reception that drives them can be perceived by human senses. Due to this property, OOB communication provides authentication and integrity, unlike radio communication. In other words, a user can validate the intended source of an OOB message and an adversary can not manipulate an OOB message in transit.

The usability of an OOB-based pairing method is very important. Since OOB channels typically have low bandwidth, the shorter the data that a pairing method needs to transmit over these channels, the better is its usability. A recent innovation to this end is the development of Short Authenticated String (SAS) based protocols [34, 35, 22, 7, 38] that limit the length of data transmitted over OOB channels to only 15 bits or so. A variety of pairing methods based on visual, audio, tactile, and infrared OOB channels have been proposed based on these protocols. We refer the reader to several excellent surveys and comparative usability analyses of various OOB pairing methods [4, 17, 18, 15]. These pairing methods will be reviewed in the related work section on pairing techniques.

The focus of this paper is on social pairing scenarios, as discussed in [3], in which two different users control their respective devices while pairing them. Examples include pairing laptops, tablets, or cell phones for social or professional reasons. The main advantage of using Bluetooth or WiFi in such scenarios is that no infrastructure is needed and thus ad hoc communication can take place at no cost. For this reason, social scenarios have been emerging rapidly and are already quite popular.

The problem of social pairing is simpler than that of personal pairing [3] where both devices are controlled by a single user. This is because, unlike personal pairing, the devices taking part in social pairing are not usually constrained in terms of input or output interfaces. Cell phones, for example, are equipped with a rich set of interfaces which make the establishment of OOB channels simpler.

1.1 Research Challenges

Unfortunately, even the social pairing problem turns out to be daunting in practice, and has not been addressed in a fully satisfactory manner, despite being subject to several recent years of research. Prior work on pairing raises several fundamental usability and security related concerns and research challenges. The most prominent of these are as follows:

- **Short OOB Strings:** Most existing pairing methods are based on SAS protocols that use very short strings, only 15 bits or so in length. The level of security provided by these methods may therefore not be sufficient for certain applications (i-15-bit SAS limits the success probability of an attack to $2^{-15}$). Increasing the length of SAS strings, on the other hand, may lead to poor usability (and security, as discussed below) because the process will become lengthier. Methods that are automated (e.g., based on cameras) and can transmit longer SAS strings, are also shown to have undesirable usability properties [4].

- **Human Errors:** Even while using short OOB strings, several comparison-based pairing methods (i.e., those based on manual comparison of OOB strings) do not offer the theoretical level of security guaranteed by their underlying protocols, as demonstrated in [4]. This is due to the potential these protocols have for human errors. Such errors can be of two forms: fatal and safe [12]. Fatal errors occur when a user accepts a pairing instance, although
the OOB strings on the two devices did not match, leading to a man-in-the-middle attack. Safe errors, on the other hand, occur when a user rejects a pairing instance even when the OOB strings on the two devices match. Such errors undermine the usability of pairing, but can also have an indirect impact on security; a failed pairing necessitates repetition, which may lead to user annoyance and translate into attacks eventually.

- **Rushing User Behavior:** A more serious issue is that the security of pairing often has to rely upon decisions made by users. As a result, a *rushing user* [26] may simply just “accept” the pairing, without having to correctly take part in the decision process. Pairing methods that are based on transfer of OOB strings, and decisions made by devices instead, are naturally resistant to rushing user behavior but are still prone to safe errors [12].

The aforementioned challenges motivate the design of a radically different approach to pairing. The central research question this raises can be summarized as: Can we design pairing methods that can handle longer OOB strings and are as resistant as possible to potential safe errors, fatal errors, and rushing user behavior? This question can, alternatively and fundamentally, be framed as follows: Can users be incentivized in some way so that they correctly take part in the pairing process, thus achieving an improved level of security and user experience?

### 1.2 Motivation: Games for Pairing

To help answer the question posed above, we propose a novel research direction involving the application of computer games for pairing of devices. The incentive that we aim to provide to users, when pairing their devices, is fun and entertainment. Since games are a common form of entertainment, our hypothesis is that they may improve the security as well as usability of pairing, and therefore help address the challenges outlined above. We note that our overarching idea is rooted in human psychology and is based on the principle of extrinsically motivated design [33]. Based on the sheer popularity of games [39], playful approaches to pairing promise to appeal to a large fraction of user population (especially the youth and children [29][41]). Moreover, game-oriented approaches can co-exist with traditional pairing mechanisms.

We try to delve deeper as to why games should be used to address the problem of usable security in general and device pairing in particular. While performing security tasks (such as pairing), users may not be aware of or care about the impact their actions and decisions have on the security or privacy of their devices and data. Due to this lack of engagement and motivation in the security process, users may not do their best at the task or may attempt to skip it entirely.

To address this issue, we propose the reframing of the security task not as a tedious procedure that puts a costly burden on users, but rather as a playful process that is enjoyable and entertaining to complete. It is our aim to transform this security operation from one that users seek to avoid or complete as quickly as possible into one that they relish. As a result, users will be more attentive to and aware of the steps they must follow while executing pairing and will perform better at it. Furthermore, if a game involves competitiveness this will provide another layer of motivation for users to put forth their best possible performance. Another important side effect of utilizing a game is that users might be willing to spend more time during the security process, resulting in increased tolerance for such tasks. In the context of pairing, potentially longer OOB strings can thus be used, providing a higher level of security.

In essence, by contextualizing a security task as playful rather than a chore, the usability burden it imposes may be greatly reduced. We dub this the *Tom Sawyer Effect* after a well known event in Mark Twain’s literary classic, “The Adventures of Tom Sawyer” [24]. In this novel, the boy Tom is punished by being forced to paint a fence on his day off. To escape his plight, the clever Tom treats the task as fun rather than resenting it. Upon observing his delight, his friends insist that they be given an opportunity to paint the fence so that they can enjoy it as well. Much in the same way that Tom convinces his friends to complete what would otherwise be considered an uninteresting job by treating it as a game, we seek to persuade users to be attentive during security operations by making these operations enjoyable. Much like Tom’s friends, users will aim to achieve precisely the same security goals before and after the addition of playfulness, but will be more inclined to participate due to the perception of fun.

### 1.3 Our Contributions

The technical contribution presented in this work is a new pairing system which we call “Alice Says.” Alice Says is based on a popular memory game called Simon Says and it accomplishes the underlying manual task of transfer of OOB strings between the device. We report on the design and implementation of Alice Says.

In addition, we also discuss several other security problems which are lacking optimal solutions and suggest ideas on how entertainment can be used to improve the current state of the art solutions that have been developed to address them.

At a higher level, we believe that our work opens up a new area of research in usable security where security tasks are presented in a playful way by making use of computer games. Designing games that are optimal in terms of speed, error tolerance, and psychological acceptability for a given application is a research challenge.

### 2 Related Work

#### 2.1 Prior Pairing Methods

In this subsection, we present prior pairing methods and their weaknesses. In particular, we discuss whether or not these methods are resistant to rushing user behavior. We divide the methods into two categories [26]: device-controlled (DC) and user-controlled (UC). In a DC method, a device decides the outcome of paring. In a UC method, the user decides the outcome. A DC method is naturally resistant to rushing user behavior but a UC method is not.

Stajano and Anderson [13] proposed establishing a shared secret between two devices using a link created through a physical cable. This is a DC method and is resistant to a rushing user. However, in many settings establishing physical contact might not be possible; the devices might not have common interfaces or it might be too cumbersome to carry the cables. Balfanz, et al. [11] extended this approach through the use of an infrared channel. Here devices exchange their public keys over a wireless channel followed by exchanging hashes of their respective public keys over infrared. This is also a DC method. The main drawback of this technique, is that it is only applicable to devices that are equipped with infrared transceivers. Moreover, the infrared channel is not easily perceptible by human users.

A second approach is to perform the key exchange over a wireless channel and authenticate it by requiring the users to manually compare the established secret on both devices. Since manually comparing the established secret is cumbersome for users, methods were designed to simplify it. These include Goldberg’s Snowflake mechanism [14] and the Random Arts visual hash [5] by Perrig and Song. These methods require high resolution displays and are thus only applicable to a limited number of devices, such as laptops. Furthermore, these are UC methods and are thus vulnerable to a rushing user.

---

1A rushing user is a user who – in a rush to connect her devices – would skip through the pairing process, if possible [26].
Based on the pairing protocol of Balfanz et al. [11], McCune et al. proposed the “Seeing-is-Believing” (SiB) method [16]. SiB involves establishing two unidirectional visual channels; one device encodes data into a two-dimensional barcode and the other device reads it using a camera. SiB is a DC method. Since it requires both devices to have cameras, it is only suitable for pairing devices such as camera phones. Moreover, a recent study [4] shows that users may not be comfortable handling cameras.

Goodrich, et al. [23] proposed “Loud-and-Clear (L&C)”, a pairing method based on “MadLib” sentences. This system encodes OOB data into MadLib sentences that users compare on two devices. This is clearly a UC method, and is not resistant to rushing user behavior.

Saxena et al. proposed a pairing method based on a visual channel [25]. It uses a SAS protocol [34] and is aimed at pairing two devices, A and B, where only B, has a relevant receiver. First, a unidirectional channel is established by device A which transmits SAS data. For example, A could accomplish this using a blinking LED. Device B would then receive this information using a video camera. This is followed by device B comparing the data it received with its own SAS data and displaying the result. Finally, the user reads the result and indicates it to device A. From A to B this is a DC method. In the other direction, however, it is a UC method and thus a rushing user could simply tell A to accept the pairing instance without looking at the pairing outcome on B.

Uzun et al. [12] carried out a comparative usability study of pairing methods. They consider scenarios where devices have at least 4-digit displays. In what they call the “Compare-and-Confirm” approach (a UC method), users read and compare SAS data. The “Select-and-Confirm” approach (a DC method), on the other hand, requires users to select a string on one device that matches with a string on the other device. The third approach, “Copy-and-Confirm”, is a DC method. It requires that users read data from one device and input it on another. Both Select-and-Confirm and Copy-and-Confirm are DC methods. However, since they are based on the protocol of [25], they offer protection against a rushing user only in one direction.

Recent papers have focused upon pairing devices which lack good interfaces. Access points and headsets are examples of this kind of device. These constraint oriented pairing solutions include the BEDA method [8], which requires that users transfer SAS strings from one device to another using button presses. In [32, 42], Saxena et al. presented a similar pairing method that is universally applicable. It involves users comparing very simple audiovisual patterns such as “beeping” and “blinking.” The approach of [32] was extended by making use of an auxiliary device such as a smartphone [27]. However, both of these are UC methods and thus offer no protection against a rushing user.

Soriente et al. [9] consider pairing two devices which might not share any common wireless communication channel at the time of pairing but do share a common audio channel. This is a DC method. However, it is limited to devices that possess a speaker at the transmitting end and a microphone at the receiving end. Moreover, this method still requires that users manually compare SAS data.

### 2.2 Computer Games and Security

Our work was inspired by that of Halprin and Naor [31] who proposed the use of games to address random number generation. Computers often use inputs from users as an entropy source. Unfortunately, when asked to cooperate in this endeavor, human users tend to perform poorly by interacting with the machine in a predictable fashion. This is because humans are notoriously bad at behaving randomly. When asked to construct random sequences, people’s outputs are riddled with biases. Interestingly, when placed in competitive situations humans demonstrate a heightened aptitude for behaving randomly [31]. Therefore, gains are noted when users are asked to participate in a game that forces them to behave randomly and then harvests entropy from their actions.

The pairing mechanism that we present in this paper is an example of a Game with a Purpose (GWAP) as conceptualized by von Ahn [20]. This is because it is not simply a game for its own sake, but rather a form of entertainment that simultaneously achieves a well-defined objective. The reCAPTCHA [21] project of von Ahn et al. is also related. It does not involve any entertainment but also fools users into doing work beyond what they may realize. reCAPTCHA not only serves the purpose of a reverse Turing test but also utilizes the responses it receives to aid in the digitization of words. A crucial difference between our proposals and von Ahn’s is that our games are meant to accomplish human work as part of the underlying security mechanism itself rather than solving an offline problem such as labeling images [19]. Also relevant to our proposal is an independent line of research on offline security education and training through playful approaches [36, 28, 2, 37].

### 3 Design of a Pairing Game

#### 3.1 Threat Model

Before discussing the design of our secure device pairing game, it is first necessary to establish the adversarial model it is intended to operate in. We use the model suggested by Vaudenay [38]. Wireless devices may establish two types of communication channels. The first is a traditional wireless connection, which is characterized by a large bandwidth capacity and bidirectionality. The second comprise the set of OOB channels, which feature modest bandwidths but are physically authenticatable. That is, OOB channels are crafted from output which can be perceived by unassisted humans, which allows them to verify transmission sources themselves. This implies that malicious entities are not capable of modifying messages sent via OOB means. OOB channels are not generally secret, however. In other words, adversaries can observe the OOB transmission. In contrast, opponents have a complete control of the conventional wireless channel. Denial-of-service attacks are beyond the scope of this model. Thus we do not consider adversaries who interfere with a user’s ability to complete the pairing game by jamming either the wireless or sensory channels involved.

#### 3.2 Choice of the Game

In order to leverage the Tom Sawyer effect to improve the device pairing experience, a suitable game had to be designed. We took our inspiration from Hasbro’s Simon [6]. While this game was originally a freestanding electronic device, many derivatives have been created that can be played on mobile devices or through a web browser on a traditional computer. This game was selected as a basis for our pairing game for several reasons. Simon is a well established game. Rather than creating a new solution from scratch which users may not have found to be enjoyable, we hoped to leverage the known popularity of Simon. Further, this game is relatively uncomplicated when compared with many contemporary computer games. This was desirable both due to its ease of implementation as well as its suitability for players of different ages and levels of experience. Finally, an important factor in the selection of this game is its close relation to existing device pairing solutions. Previous work has established the use of patterns of synchronized audio and visual output as a viable method of securely associating devices [32, 42, 8]. At its core, playing Simon involves nothing more than the short term memorization of audiovisual patterns and thus minimal changes were required to adapt it for use in pairing. We discuss the details of Simon in the following section.
3.3 Alice Says Game Design

Upon initially starting Alice Says, users are provided a screen with two menu choices: a single player training mode and a two player pairing mode. A single player mode is provided to allow users an opportunity to unwittingly train themselves to improve their device pairing performance. The two player mode is what actually accomplishes device pairing.

3.3.1 Single Player Mode

The single player mode is essentially identical to the classic version of Simon, only adapted to the context of a mobile device. The various steps involved in this mode are discussed below.

- The user is shown a screen with four adjacent squares which fully occupy the screen, dividing it into quadrants. Each of these squares is a unique and distinctive color. Clockwise starting in the upper left, the colors are green, red, blue, and finally yellow.
- The only other item visible while the game is underway is a counter which tracks the number of rounds that have been played thus far. The counter increments with each round that is successfully passed.
- One of the four quadrant buttons is randomly selected by the device during each round. The selection is indicated to the user in two complementary ways. First, the screen section is lit by increasing its luminance. Secondly, a tone corresponding to that quadrant is played. The notes associated with the four portions of the screen are harmonically compatible irrespective of the order in which they are played. The use of such notes was critical to Simon’s popularity [40, 30].
- Each round encodes two randomly chosen bits in the following manner: “00” corresponds to green, “01” is indicative of red, “10” means blue, and “11” is aligned with yellow.
- If a user presses the correct quadrant, the device lengths the puzzle pattern by displaying the previous pattern element followed by the color encoding another pair of randomly chosen bits. Similar to the previous pattern element, this is again conveyed to the user by brightening the relevant portion of the screen and playing a corresponding melodic tone.
- The above process continues until the user makes an error or a certain pre-determined round threshold value is reached. At this point a “Game Over” message is displayed, informing the user of the number of rounds that he or she was successfully able to play.

3.3.2 Two Player Mode

The two player mode accomplishes the underlying pairing task of transfer of OOB strings between two devices. It differs from the single player, traditional Simon approach in two main ways:

1. The game is split across two devices. One device (A) displays the pattern to the user, but does not handle input. The other device (B) does not display the pattern, but only accepts user input. Using the game, an input OOB string is transferred from A to B. Please recall that each device computes an OOB string as a result of the SAS protocol. In a transfer-based pairing method, these strings then need to be exchanged over the OOB channel and compared by the devices.
2. The game does not conclude when a mistake is made. It continues until a sufficient number of OOB bits have been relayed between the two devices. This makes the game robust to human errors.

As a result of the second difference, a mechanism is required to keep the two devices in sync during the pairing procedure. This is because the device accepting input must be aware of what rounds the user has played to be able to conclude whether or not he or she has committed a mistake. A naive way to handle this would be to simply transmit the current round over the wireless channel. This would ruin the security of the system, however, as the wireless channel is assumed to be totally insecure as per the security assumptions detailed in our threat model. Thus, an adversary could simply transmit an arbitrary round number over this channel, by-passing as much of the process as desired. Instead, we addressed the synchronization issue by integrating “previous” and “next” buttons into the displaying phone’s (A’s) interface. The use of these buttons and other steps involved in the two player mode are provided in Algorithm 1 and intuitively described in the following list.

- In the first round, the user will be provided with a pattern, on device A, of length one. Then the pattern will be extended to length two, and so on. That is, the original pattern consisting of a single color and tone that encodes the first two bits of A’s OOB string will be concatenated with a new value encoding the next two bits. This will form a new pattern that is comprised of two colors and two tones which express four bits. Next, another two bits will be appended to the pattern to form a six bit pattern that is displayed to the user in the form of three colors and three tones. This process continues in an iterative fashion.
- Upon a successful round, the player in control of the display phone A presses next to advance the state of the game. A then displays the new pattern in the next round. If an error is made during the round, on the other hand, A’s user can indicate this to B by pressing the previous button.
- In the event of an error in a round, a new pattern is crafted starting with the pattern portion that was not copied success-

Algorithm 1 Alice Says Pseudocode

<table>
<thead>
<tr>
<th>Algorithm 1 Alice Says Pseudocode</th>
</tr>
</thead>
<tbody>
<tr>
<td>input: OOB strings a and b, on devices A and B, resp.</td>
</tr>
<tr>
<td>display string d = [a[0],a[1]]</td>
</tr>
<tr>
<td>k = 1</td>
</tr>
<tr>
<td>while k &lt; threshold (threshold is the number of bits that need to</td>
</tr>
<tr>
<td>be matched, which is equivalent to the lengths of a and b) do</td>
</tr>
<tr>
<td>displayPattern(d)</td>
</tr>
<tr>
<td>I = getInputPattern()</td>
</tr>
<tr>
<td>correctMatch = true</td>
</tr>
<tr>
<td>for n &lt; length(I) {n is used as an index into the input string</td>
</tr>
<tr>
<td>and device B’s OOB string b} do</td>
</tr>
<tr>
<td>if I[n] != b[n+offset] {offset is the index into b, of bits that</td>
</tr>
<tr>
<td>have been successfully matched so far} then</td>
</tr>
<tr>
<td>correctMatch = false</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end for</td>
</tr>
<tr>
<td>if correctMatch then</td>
</tr>
<tr>
<td>d = [d[0],d[1],...,d[m-1],a[k+1],a[k+2]] {m is the length of</td>
</tr>
<tr>
<td>d}</td>
</tr>
<tr>
<td>k = k + 2</td>
</tr>
<tr>
<td>displayCorrect() {user of A presses the next button}</td>
</tr>
<tr>
<td>else</td>
</tr>
<tr>
<td>d = [a[k+1],a[k+2]]</td>
</tr>
<tr>
<td>displayError() {user of A presses the previous button}</td>
</tr>
<tr>
<td>end if</td>
</tr>
<tr>
<td>end while</td>
</tr>
<tr>
<td>displaySuccess()</td>
</tr>
</tbody>
</table>
fully. This is done because, as an invariant, the devices know that all of the rounds up until this point have been successful.

- After an error is made, the input device $E$ will compare its running tally of successful matching bits, of its own OOB string, to how many rounds need to be performed. If more rounds are necessary, the game continues.
- If mutual authentication is desired, the roles of the phones can be swapped following a successful game in a single direction. After this, the game play would proceed in precisely the same way with the roles of the two phones and users reversed.

### 3.3.3 Example Usage Scenario

The following is an example of an anticipated game play pattern. Let us assume a legitimate pairing session in which a user is consistently able to follow 5 rounds. There are 30 bits in the OOB string that need to be compared.

- The user will first be provided with a pattern of length one in round one. The user will successfully match this pattern. Then the pattern will be extended to length two in round two, and so on.
- Let us say that on the sixth round, the user makes a mistake. At this point, the user has successfully transferred the first 10 bits of the OOB string. In the next round, the game will begin a new pattern starting with the 11th and 12th bits of the OOB string (with one color being displayed). The process is repeated as before.
- Let us say that the user makes another mistake at round 11. Now, the game will begin with a new pattern starting with the 21st and 22nd OOB bits. The process is repeated as before.
- After successfully completing the next 5 rounds (i.e., a total of 15 rounds), all 30 bits will have been conveyed, concluding the game.

### 3.4 Security Guarantees

An important aspect of Alice Says (pairing mode) design is how it handles attacks. If a session has been attacked, the OOB strings calculated on the two devices will be different. Even if a user “correctly” transfers the displayed pattern, Alice Says will register an error. This will either occur as the first bit of a pattern or a subsequent bit of the pattern. If the attack occurs as a subsequential bit, the bits prior to the error will be registered as a match for that session and the pattern will begin anew with the attacked bit, making it the first bit of the next pattern. Thus, one way or another, the attacked bit will end up as the first bit of the next pattern, and users will be unable to proceed by identifying the single color pattern that has been displayed to them.

Users therefore need a mechanism for restarting a pairing session. To achieve this, after a certain threshold of single color pattern mismatches have occurred, an error message – indicating the possibility of an attack – will be displayed. At this point, users can discard the session and start over with a new one. Note that single color pattern mismatches are unlikely in unattacked sessions as most users can match at least one color. Thus, the only way for a critical error to occur in this system is for users to incorrectly match a single color. Given an OOB string of length $n$, there is only a $1/4 + n * 2^{n-1}$ chance of this occurring even if the user is not paying any attention. This is because there is a $1/4$ chance of a user randomly striking a particular color and a $n * 2^{n-1}$ chance that two OOB strings, in the presence of an attack, mismatch in just one bit. Note that while it is theoretically possible for a player to complete pairing with Alice Says by making random color quadrant choices, this would take prohibitively long to achieve and can be ruled out in practice.

### 4 Implementation of Alice Says

To develop Alice Says, we preserved the popular aspects of Simon while updating it to a two player mobile device setting. Its user interface is dominated by four large color buttons as was the case with its ancestor. Also intended to mimic the original was the association of a unique tone with each of these keys. A critical aspect of the original game’s appeal was the fact that these sounds were designed to be harmonic irrespective of the order in which they were played. This is important as the game play involves striking the inputs in a random order. We thus tried to mimic the original game’s sounds by assigning an A note to the first input, an A note one octave higher to the second, a D note that is a perfect fourth above the initial A note to the third button, and finally a G note that was a perfect fourth higher than the D to the last key [40, 30].

We utilized two Nokia N97 mobile phones to realize our Alice Says prototype. These devices support the Java Platform, Micro Edition (Java ME) environment, which we designed our code in. We crafted a user interface that was as intuitive and user friendly as possible. We utilized the lower level APIs provided by the Nokia N97 SDK. The device accepting user input kept track of whether or not an error occurred and automatically adjusted the pattern length accordingly. The N97s were well suited for exploring the usability of device pairing. They featured a resistive touch screen and stereo speakers, for example.

### 5 Other Applications of Games for Security

The Tom Sawyer effect can be applied to various security issues, especially in the context of wireless and mobile computing, to enhance usability. Halprin and Naor have already applied this principle to the dilemma of random number generation to great effect [31]. Another area where it may be fruitful to apply this concept is that of authentication. For example, the usability of current mobile phone password managers, such as KeePassMobile [10], can be improved. These applications suffer from poor usability by requiring that users manually transfer passwords from a phone screen to the authentication terminal. Games similar to Alice Says can be adopted to address this.

Another application in which games may be of use is authentication to a remote server as a replacement for CAPTCHA mechanisms. In order to prove to remote servers that a human user is really behind a given request, users will be challenged by playing a game that is relatively easy for humans to complete but difficult for computers. Note that currently deployed textual CAPTCHAs are not suitable for mobile devices, due to their small form factor. Finally, games may be designed to supplement the security and privacy of “something you have” authentication techniques by having users play a short movement game in order to unlock their access tokens, such as RFID tags. This idea is similar in spirit to the recently proposed Secret Handshakes scheme [1] but is aimed at providing an enhanced level of usability.

### 6 Conclusions

In this paper, we contributed “Alice Says,” a novel system for pairing devices via a game. More broadly, we considered the problem of designing pairing methods that in some way incentivize users to put forth more effort and correctly take part in the pairing process, thus providing improved security as well as enhancing the overall user experience. We dubbed this the Tom Sawyer Effect. To this end, we proposed a general research direction of applying computer games to solve tricky issues in usable security. The incentive that we provide to users while they pair their devices is fun and entertainment. Since games are a popular form of entertainment, our hypothesis was that they may improve the security as well as usability of pairing and help solve the challenges outlined above. As
part of our future work, we will conduct a formal usability study of Alice Says, and contrast its usability and security with that of a traditional pairing mechanism (such as based on numeric transfer).

At a higher level, we believe that our work opens up a new area of research in usable security where security tasks are presented in a playful way by making use of computer games. Designing games that are optimal in terms of speed, error tolerance, and psychological acceptability for a given application remains an open research challenge. We hope that our work will motivate other researchers and practitioners to come up with novel games for addressing lingering problems in usable security.

7 References


