# Using audio in secure device pairing

Michael T. Goodrich, Michael Sirivianos, John Solis, Claudio Soriente, Gene Tsudik and Ersin Uzun*

Computer Science Department,
University of California, Irvine, USA
E-mail: goodrich@ics.uci.edu
E-mail: msirivia@ics.uci.edu
E-mail: jsolis@ics.uci.edu
E-mail: csorient@ics.uci.edu
E-mail: gts@ics.uci.edu
E-mail: euzun@ics.uci.edu
*Corresponding author

**Abstract:** Secure pairing of electronic devices is an important issue that must be addressed in many contexts. In the absence of prior security context, the need to involve the user in the pairing process is a prominent challenge. In this paper, we investigate the use of the audio channel for human-assisted device pairing. Fist we assume a common (insecure) wireless channel between devices. We then obviate the assumption of a pre-existing common channel with a single-channel device pairing approach only based on audio. Both approaches are applicable to a wide range of devices and place light burden on the user.

**Keywords:** human-assisted authentication; MITM; man-in-the-middle attacks; secure device pairing; audio; personal device.

**Reference** to this paper should be made as follows: Goodrich, M.T., Sirivianos, M., Solis, J., Soriente, C., Tsudik, G. and Uzun, E. (2009) 'Using audio in secure device pairing', *Int. J. Security and Networks*, Vol. 4, Nos. 1/2, pp.57–68.

**Biographical notes:** Michael T. Goodrich is a Chancellor's Professor in the Department of Computer Science at University of California, Irvine, where he joined the faculty in 2001. Prior to that he was a Professor of Computer Science at Johns Hopkins University. He received his PhD in Computer Science from Purdue University in 1987. His research interests include algorithms for computer security and information assurance.

Michael Sirivianos is a PhD student at University of California, Irvine. His research interests include cooperative content distribution, collaborative unwanted traffic mitigation, and human verifiable secure device pairing. He received a BS in Electrical and Computer Engineering from the National Technical University of Athens in 2002, and an MS in Computer Science from the University of California, San Diego in 2004.

John Solis is currently a PhD student in the Donald Bren School of Information and Computer Science at the University of California, Irvine. His research interests include applied cryptography and network privacy and security. He is currently researching security and privacy mechanisms for Delay Tolerant Networks (DTNs).

Claudio Soriente is a PhD candidate in Computer Science at the University of California, Irvine. He received the MS Degree in Computer Science in 2004 from University of Salerno. His research interests include wireless sensor networks and applied cryptography.

Gene Tsudik is a Professor of Computer Science at the University of California, Irvine. He has been conducting research active in internetworking, network security and applied cryptography since 1987. He obtained a PhD in Computer Science at University of Southern California in 1991. Before coming to UC Irvine in 2000, he was a Project Leader at IBM Research, Zurich Laboratory (1991–1996) and USC Information Science Institute (1996–2000). His research interests include authentication, mobile/wireless network security, anonymity, secure group communication, digital signatures, key management, secure ad hoc network routing, database privacy, secure storage and usable security.

Ersin Uzun is a PhD candidate in Networked Systems at University of California, Irvine. He received his MS Degree in Computer Science in 2006 from University of California, Irvine and BSc Degree in Computer Engineering in 2004 from Bilkent University. His research interests revolve around building secure and usable systems and include usable security, authentication, mobile applications, network security, applied cryptography and digital rights management.

Copyright © 2009 Inderscience Enterprises Ltd.

## 1 Introduction

The proliferation of many types of inexpensive personal devices – such as PDAs, cellphones, smart watches, and MP3 players – has been accompanied by the need to secure their communication with the "outside world". Common applications involve securely connecting one's personal device to an unfamiliar printer, fax machine, wireless projector, network access point or another personal device. One of the main challenges is the problem referred to as: *Secure First Connect*, *Secure Initialisation* or *Secure Device Pairing*. Regardless of the name, this problem entails the establishment of a secure communication channel between previously unassociated devices.

Traditional cryptographic means of establishing secure communication channels (e.g., authenticated key exchange protocols) are generally unsuitable for secure device pairing. This is because mutually unfamiliar devices have no prior context and no common point of trust: no online Trusted Third Party (TTP), no offline Certification Authority (CA), no Public Key Infrastructure (PKI) and, of course, no common secrets.

Lack of any prior security context makes it necessary to involve the user in the device pairing process. This is because of the very real threat of so-called Man-in-the-Middle (MiTM) attacks (Kugler, 2003). A MiTM attack can occur whenever unauthenticated communication is involved. One of the best-known examples is the textbook Diffie-Hellman Key Exchange protocol (Diffie and Hellman, 1976). In it, an attacker can easily masquerade as either party such that, at the end of the protocol, one party (Alice) thinks that she is *talking* to the other party (Bob), whereas, she is actually talking to the adversary.

A number of techniques have been proposed, varying greatly in the assumptions about device features, degree and nature of user involvement as well as environmental factors. (See Section 9 for an overview of related work.)

In this paper, we investigate the use of the audio channel to securely pair devices. We view the audio channel as one of the most ubiquitous communication channels: audio-in or audio-out interfaces are present on the majority of personal devices (e.g., cell-phones, music players, cameras, wireless headsets) and cost very little. We introduce two novel techniques: one based on vocalisable data representation, and the other – on pure audio. Both methods offer interesting and viable alternatives to prior device pairing proposals.

Audio-based two-channel secure pairing uses spoken natural language for human-assisted authentication and requires either a display or a speaker on each device, as well as another common communication channel (e.g., 802.11 or Bluetooth). First, devices exchange their respective public keys over an un-authenticated human-imperceptible wireless channel. Then, the process is authenticated via a syntactically correct English sentence that can either be played through the device speaker or shown on its display. The user matches the sentences produced by the two devices to validate the secure context/channel establishment.

Our second technique avoids the need for a common wireless interface and human-aided data comparison. If both devices have audio-in and audio-out interfaces, solely the audio channel itself can be used to transmit device public keys directly. Since audio is human-perceivable, the user can validate the actual transmitting devices during communication. Using a single (audio) channel spares the user from the need to set up a wireless connection beforehand, which is often complex and time-consuming for non-specialist users (e.g., Bluetooth in-range device discovery or 802.11 ad-hoc network configuration).

In this paper, our main goal is to investigate the design space of audio-based secure device pairing. Our motivation is two-fold:

- While some devices may include a built-in photo (and/or even video) cameras, accelerometers, laser or infra-red transceivers, such interfaces are more expensive and not as common as audio.
- Despite its ubiquity and low cost, the audio channel provides sufficient bandwidth for the purposes of key agreement. Being inherently human-perceivable, the audio channel can be used as the only channel in secure device pairing and thus eliminate the need for an auxiliary wireless channel.

To investigate the use of audio as a limited authenticated channel (to secure a higher bandwidth insecure channel), we introduce an approach to represent data in meaningful human-audible manner. In it, we use recently proposed Short Authenticated String (SAS) key agreement protocols (Vaudenay, 2005; Pasini and Vaudenay, 2006; Laur et al., 2006; Cagalj et al., 2006). A SAS protocol result (a short string) is mapped to a robust-sounding and syntactically-correct (English-like) sentence. Such a sentence can be either be vocalised or displayed (depending on the device's features) for user comparison. Then, to explore the full potential of using audio, we consider the audio channel as the only means of data transfer between devices. We establish a bi-directional audio-based communication channel and use it for sending the actual public keys. Since audio is human-perceivable and the user can identify its source (Pickles, 1982; Stevens and Davis, 1938; von Békésy and Wever, 1960), this approach eliminates the need for the user-aided second verification phase. Later in the paper, we discuss the use cases, security properties and implementation issues of both approaches as well as their usability later in the paper.

The rest of the paper is organised as follows: the next section presents our classification of the communication channels used in secure device pairing. Section 3 provides a brief overview of SAS protocols. Sections 4 and 5 introduce our approaches with two-channel and single-channel pairing, respectively. We analyse their security in Section 6, and provide a brief implementation description in Section 7. We discuss advantages and limitations of our proposals in Section 8. Finally, we overview related work in Section 9.

## 2 Communication channels

In this section, we define the communication channels involved in our secure pairing solutions.

Securing a communication channel between two devices requires establishing a common secret key. The channel to be secured, hereafter referred to as *main* channel, is typically wireless and not perceivable by human user (e.g., a visible wired channel between devices would be immune to MiTM attacks (Stajano and Anderson, 2000)). Prominent examples include Bluetooth and 802.11. Due to lack of prior history between devices and the threat of MiTM attacks (Kugler, 2003), the main communication channel alone is insufficient for establishing secure communication.

However, another channel – referred to as the *auxiliary* channel – can be used to address this problem. If the auxiliary channel is human-perceivable, the user can verify (authenticate) the actual communicating devices. If both main and auxiliary channels are available, the former can be used to transfer cryptographic protocol messages and the latter – to authenticate message source(s). This approach is practical with many realistic scenarios, since the auxiliary channel is usually more constrained in terms of bandwidth and can not always be used for transferring (potentially long) protocol messages. Alternatively, if the main channel is not available or is difficult for the user to set up, the auxiliary channel can play both roles, i.e., it can be used for authenticated transmission of protocol messages and/or for automated configuration of the main channel.

## 3 Short Authentication String protocols

We now introduce the SAS protocol whereupon one of our device pairing protocols is based.

Early strawman solutions to the device pairing problem required between 80 and 160 bits of data to be transmitted over the auxiliary (human-perceivable) channel. One early protocol (Maher, 1995) involved two devices exchanging their respective public keys over the wireless channel and users authenticating them by comparing truncated $t$-bit hashes of the agreed-upon session key. The pitfall of hash truncation has to do with the number of digits to be examined by the user. Drastically truncating the hash (e.g., to first 16 or 32 bits) enables an attacker to replace the user's public key with second pre-images of the hash.

Assuming typical adversarial capabilities, $t = 50$ (13 hex digits) and $t = 80$ provide sufficient security when ephemeral public keys and one-year-term keys are used, respectively. However, examining 13 hex digits is an error-prone and cumbersome task for a human user.

Recently proposed SAS protocols (Vaudenay, 2005; Pasini and Vaudenay, 2006; Laur et al., 2006; Cagalj et al., 2006) reduce the length of verification to only 16–20 bits. Below, we describe a representative SAS protocol, called DH-SC (Cagalj et al., 2006). Like all other SAS protocols, it uses a commitment scheme which transforms an initially secret input value $m$ into a commitment/opening pair $(c, d)$. In this pair, $c$ reveals no information about $m$ (e.g., $c$ is the public key encryption of $m$) but $c$ and $d$ together (e.g., if $d$ is the decryption key) reveal $m$. In an ideal commitment scheme, it is infeasible to find $d'$ such that $(c, d')$ yield $m' \neq m$. The DH-SC protocol requires communicating parties to compare a string derived from the XOR of two per-session random bit-sequences contributed to by each of the two parties. DH-SC thus effectively reduces the length of the compared values for a given level of security. For example, 20-bit random sequences provide a level of security almost equivalent to that of a 50-bit hash of ephemeral Diffie-Hellman public keys.

DH-SC operates as follows: both Alice and Bob ($A$ and $B$) generate their respective public keys $g^a$ and $g^b$. Then, A and B each generate a $t$-bit random string $N_A$ and $N_B$. They use these strings to generate commitment/opening pairs for the concatenations: $0\|g^a\|N_A$ and $1\|g^b\|N_B$, where 0 and 1 are fixed values used to prevent reflection attacks. In the first message, $A$ sends $B$ its commitment $c_A$ and $B$ responds with its own $c_B$. Then, $A$ sends $B$ her commitment key $d_A$ which B uses to open $c_A$ and obtain a pair: $[g^{a'}, N_A']$. $B$ verifies the commitment pair: $c_A, d_A$ and makes sure that '0' appears at the start of the message. If verification succeeds, $B$ sends $A$ the value $d_B$, which $A$ uses to open $c_B$ and obtain a pair: $[g^{b'}, N_B']$. Next, $A$ checks correctness of this commitment and, if it is valid, both parties proceed with generating respective verification strings for $i_A = N_A \oplus N_B'$ and $i_B = N_B \oplus N_A'$. Finally, the human user(s) simply compare the two strings and formally accept exchanged public keys if there is a match.

## 4 Two-channel secure pairing

We describe our device pairing solution when the main channel, as defined in Section 2, is available and already configured. This channel is used for the exchange of SAS messages.

We assume that the identification of communicating devices is performed visually by the user. The two devices engage in SAS protocol over a wireless channel (e.g., Infrared, 802.11, Bluetooth) and an *audio or visual representation* is then used as a means of authenticating the public keys.

### *4.1 Requirements*

The specifics of target device authentication depend on several factors, such as the public-key-based key agreement algorithm, directionality, the number of human users, and the device equipment. The following basic requirements are common to all use cases:

- There is at least one human user present with a personal device.
- At least one device has an audio-out interface, e.g., a speaker or a headphone plug (for the sake of completeness, the protocol also supports the case of both devices having displays but no audio).
- The two devices must be able to communicate via some multiple-access broadcast medium, e.g., 802.11a/b/g/n, Bluetooth.

Figure 1 depicts some use scenarios.

## *4.2 Use cases*

We now consider in more detail the factors that distinguish uses of audio-based secure pairing when both main and auxiliary channels are available.

The first such factor is the *directionality* of authentication, i.e., whether authentication is one-way (unidirectional from target to personal device) or mutual (bidirectional between the two devices). For example, in the former, the personal device may need to authenticate the target device's public key and, in the latter, each device may need to authenticate the other's public key. Since the bi-directional use scenario is a trivial extension of the uni-directional one, we focus on one-way authentication.

Equipment available on each device influences the specifics of authentication. Some devices have both a display and a speaker, while others may have only one of these.

We consider four possible use cases for verifiable authentication of the public key when using the insecure main channel:

*TYPE 1*: Hear and compare two audible sequences, one from each device.

*TYPE 2*: Hear an audible sequence from the target device and compare it to text displayed by the personal device.

*TYPE 3*: Hear an audible sequence from the personal device and compare it to text displayed by target device.

*TYPE 4*: Compare text displayed by the personal device to text displayed by target device.

TYPE 1 might appear as the most difficult for the human user. However, previous work (Soriente et al., 2007) shows that even this more taxing case is acceptable to users.

TYPE 2 and TYPE 3 use cases are very similar, while TYPE 4 does not involve any use of the audio channel.[1] Nevertheless, we include it among our supported cases, since it is an alternative or a fall-back method if TYPES 1, 3 or 3 are not viable. This could happen when both devices only have displays, in noisy environments (e.g., during a concert) or when silence is required (e.g., in a library).

Of course, the equipment factor ultimately determines the burden our solution places on the human user. Table 1 shows the types of user requirements corresponding to possible personal-target device combinations. Looking at rows 3 and 6, the choice between TYPE 3 or 1 and TYPE 2 or 1, respectively, can be dictated by the certain properties of the environment. For example, insufficient light, smoke or fog can make TYPE 1 the only viable choice. A visually-impaired user is also likely to choose TYPE 1 over TYPE 3 or 2, unless one of the two devices has a Braille display. Likewise, the TYPE 4 use case is infeasible for a visually-impaired user, unless both devices have Braille displays. In row 7, the choice between TYPE 3 or 4 is less clear.

**Figure 1** Sample use scenarios (see online version for colours)

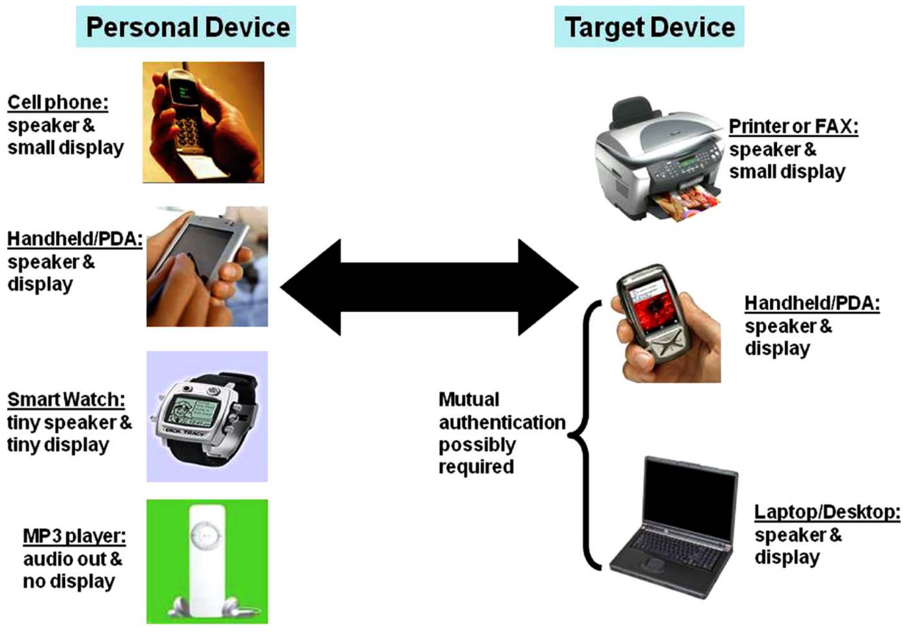

**Table 1** User requirements for various device configurations. The 'Use Type' column indicates allowed use cases, depending on device characteristics. We use '$*$' to denote a 'do not care' condition, We allow the 'Speaker' condition to include any audio-out interface that can vocalise human speech

| | | *Personal device* | | *Target device* | |
|---|---|---|---|---|---|
| *Row* | *Use type* | *Display* | *Speaker* | *Display* | *Speaker* |
| 1 | 1 | No | Yes | No | Yes |
| 2 | 3 | No | Yes | Yes | No |
| 3 | 3,1 | No | Yes | Yes | Yes |
| 4 | 2 | Yes | No | No | Yes |
| 5 | 4 | Yes | No | Yes | No |
| 6 | 2,1 | Yes | Yes | No | Yes |
| 7 | 3,4 | Yes | Yes | Yes | No |
| 8 | 1,2,3,4 | Yes | Yes | Yes | Yes |
| 9 | n.a. | No | No | $*$ | $*$ |
| 10 | n.a. | $*$ | $*$ | No | No |

## *4.3 Vocalisable and readable representations*

In our scheme, exchanged device public keys must be verified by the user. Comparing hexadecimal representations of hashes is a cumbersome task for the average user. In order to make the process faster and less tedious, a hash must be represented in a more convenient form. We represent the output of the SAS protocol as a syntactically correct English-like sentence.

Generated sentences are based on the MadLib puzzles commonly used by children.[2] That is, we generate a syntactically correct (but usually non-sensical) sentence from a string of bits.

The string of bits our scheme uses is the yield of the SAS protocol. This 16–20-bit string is then used to compute phrase-words similar to those used in the S/KEY One-Time Password System (Haller, 1995). Our scheme divides the string into two 8–10-bit parts. Each part is mapped to one word in a two-word MadLib sentence. The text is generated from a template, which consists of a grammatical sentence (or group of sentences) with missing words, each word of a various type, such as: noun, adjective, adverb, verb, boy-name, girl-name, or animal. Each missing word is replaced with a word from a dictionary of appropriate words. The word replacing the MadLib keyword is determined by converting the 8–10-bit part of the short string into an integer and using that as the index into the internal dictionary database. For example, the following is a MadLib sentence produced by our prototype, for encoding a 20-bit string (filled-in words/word-phrases are shown in all caps):

PAUL wants a BYTE.

A user can easily compare two such displayed or vocalised sentences. Still, we need to ensure that the text sequences are auditorily robust. For example, we need to avoid a situation in which a MiTM attacker manages to interfere with the public key transmissions so that one device outputs the above sentence, while the other device outputs:

PAUL wants a BITE.

To construct auditorily robust text sequences, we need to produce a number of word lists of appropriate parts of speech, with words in each list being as phonetically distant from each other as possible.

Using a metric for phonetic distance similar to that in PGPfone (Juola and Zimmermann, 1996) (but restricted to words of appropriate type) allows us to create auditorily robust word lists for each word category in MadLib sequences, as follows:

- Construct a large set $C$ of candidate words of appropriate type. These should be common English words that can all be used in the same place within a MadLib sequence.
- Select a random subset $W$ of $2^k$ words from $C$, where $k$ is the number of bits needed to represent this word type (e.g., 8 bits for a noun).
- Repeatedly find the closest pair $(p, q)$ of words in $W$ (using the phonetic distance metric) and replace $q$ with a word from $C - W$ for which the distance to any word in $W$ is more than $d(p, q)$, if such a word exists. The resulting set will be a collection of phonetically well-spread words.
- Order $W$ such that each pair of consecutive words are as far from each other as reasonably possible. Doing this optimally is NP-hard (Garey and Johnson, 1990), however, we use a heuristic algorithm based on pair-wise word swapping to come up with a good order.
- Assign integer values to words in $W$ according to Gray Code, so that consecutive integers differ in exactly one bit while their respective code words are distant.

Having described use types and the generation of MadLib text sequences from authentication objects, we now show some sample use cases.

## *4.4 Usage scenarios*

We provide two representative usage examples.

In the first example, we consider a public printer as the target device. Low-end printers are not usually equipped with displays. However, a printer is capable of output using its normal printing functionality. The user can initiate output by pressing a button that prompts the device to print the MadLib sentence yielded by the SAS protocol. The other device (communicating to the printer via a wireless main channel) vocalise or display the same MadLib sentence thus allowing the user to authenticate exchange by comparing the two sentences.

We note that audio also enables visually impaired users to interact with devices. In this context, our solution can be used to aid visually such users not only to establish a secure communication channel between trusted devices, but also to determine whether the device itself is trusted. One sample usage scenario involves the authentication of a bank ATM. A visually impaired user could be given a MadLib sentence when opening a bank account. This MadLib sentence is originally generated from the user's unique account/card number and its expiration date, via a keyed hash with the bank's secret key. When the user inserts the card into an ATM machine, the latter re-generates and vocalises the MadLib sentence corresponding to the account number, thus authenticating itself to the user.

## 5 Single-channel pairing

We now describe device pairing when a wireless channel is not available or its set up poses a burden to the user. In this case, we use audio to exchange all protocol messages between devices. Audio channel's innate properties allow the user to authenticate the actual communicating devices. Consequently, the direct exchange of device public keys over the audio channel suffices for secure device pairing and obviates the need for the second phase.

### *5.1 Discovery and setup issues*

The general pairing protocol introduced earlier in the paper, as well as most prior related work, assume the existence of a common, ready-to-use wireless communication channel. Of course, many modern personal devices tend to have at least one wireless interface. However, there are realistic scenarios where two devices do not – at least at the time of pairing – have a common wireless interface. For example, one might only have Bluetooth and the other – 802.11. Why pair such devices? One reason could be because they would later connect to the internet using their respective interfaces and would still need to communicate securely.

Even if devices are equipped with a common wireless interface, setting up the communication channel might take both time and effort even for an expert user. Non-technical users would consider common channel set-up to be a real challenge. For example, with 802.11-equipped devices, each must be placed into ad hoc mode and a new ad hoc network (complete with an SSID) must be manually configured. Infrared (IrDA) would typically need to activated manually on both devices. It also requires line-of-sight alignment between transceivers. In the Bluetooth case, one device needs to be explicitly made discoverable and the other must discover it. These can be tricky steps for ordinary users, especially, if there are many Bluetooth-enabled devices in proximity: the list of available devices can be long and confusing. In fact, this may turn into a real headache if multiple devices share the same (e.g., 'default') name. This is likely to happen when pairing popular models of personal devices, such as Motorola RZR phones.

### *5.2 Audio as the main channel*

We now introduce an approach that relies only on the audio channel. A speaker and a microphone are the only device interfaces required. We consider both uni-directional and bi-directional key exchange protocol flavours. For the latter, both devices must be equipped with an audio-in and audio-out interfaces. Whereas, for the former, one device must have an audio-out, and the other – an audio-in, interface.

The protocol consists of a single phase. In its unidirectional flavour, the target device sends the object to be authenticated (i.e., its public key) to the personal device: to do so, the target device encodes its public key using an audio codec and plays the resulting audio sequence. The personal device records the audio sequence and decodes it, using the corresponding decoder, to retrieve the target device's public key. In the bidirectional flavour, the transfer is repeated with the devices automatically switching roles.

The user is responsible for triggering the protocol execution, i.e., pushing a button on each device and identifying the playing device(s). At the very end of the protocol, the user is asked whether there was any audible interference (e.g., from some other, perhaps malicious, device). If so, the exchanged public keys are discarded and the protocol is aborted.

Since the encoded version of a public key can be fairly large (several hundred bits), encoding must be done using a *fast* codec, which provides relatively high throughput. Our prototype implementation encodes 240 bits into a 3.4-s audio sequence. Nevertheless, the high transmission rate has the undesirable side-effect of producing rather unpleasant-sounding audio, similar to that of old-fashioned phone modems.

The audio channel is inherently less resistant to noise than wireless radio channel discussed in the previous section. To improve protocol robustness, the hash of the public key is encoded as well and appended to the audio sequence. The personal device uses it as a checksum to detect transmission errors.

Due to the inherent properties of the audio channel, if an attacker tries to play the encoding of its own public key in order to launch an MiTM attack, the user can easily identify the audio source and abort the protocol. In case of audible interference (e.g., a DoS attack) the user can similarly abort the protocol. MiTM attacks that try to transfer public keys to the personal device using non-human-audible frequencies are not taken into account: the application is *tuned* to use and record only the range of frequencies that can be recognised by the human ear. The audio channel also provides some form of protection against DoS attacks, since the user can easily identify the attacker's audio source and disable it. (Of course, we recognise that this rules out hearing-impaired users.)

The single-channel protocol can also be used in scenarios where devices share a common wireless interface. Our approach is still useful since it can dramatically improves usability by eliminating the need to configure such interfaces. Audio, as a broadcast medium, does not need any initial configuration or discovery phase, as do 802.11 or Bluetooth. Whenever devices share another common interface (which they can use for communication after pairing is performed), the single-channel protocol can be easily modified to transfer the necessary information for automatic configuration of such interfaces. In our prototype implementation, we experimented with including the Bluetooth physical addresses into the exchanged messages. This added less than 1 s of extra transmission time. After successful termination, devices are able to automatically initiate secure communication over their Bluetooth interface without any further user involvement.

## 6 Security analysis

We now discuss security properties for the proposed approaches.

### *6.1 Two-channel pairing*

In this case, the communication model includes two channels:

- common bi-directional wireless channel
- a uni-directional auxiliary audio channel (from devices to the user).

In our adversarial model, the attacker has full control over the wireless channel, but, over the auxiliary audio channel, its ability is limited to eavesdropping. In other words, communication over the auxiliary channel is authenticated. Although various Denial of Service (DoS) attacks are possible on the wireless channel, we do not address them, since there are simply no practical solutions, even for the very basic attacks, such as jamming the wireless signal.

As described earlier in Section 3, this approach uses a 3-round SAS protocol. A SAS protocol provides authenticated Diffie-Hellman public key exchange by transmitting only a short $k$-bit (in our case $k = 20$) string over the authenticated channel. Assuming that other primitives, i.e., cryptographic hash functions and public key algorithms are secure, attacker's chances of success are only negligibly higher than $2^{-k}(2^{-20})$.

Since we assume that the attacker's power over the auxiliary channel is limited to eavesdropping, this channel satisfies the authenticated property needed to transfer a short string. If the devices themselves are not compromised, this is a reasonable assumption, since the only option for the attacker is to interfere with either the audio or the visual channel. Such interference can not go unnoticed. On the other hand, if any device is already compromised, the secure pairing becomes pointless, since the adversary can always extract the agreed-upon key from the compromised device once the pairing process is over.

#### *6.1.1 Single-channel pairing*

This approach requires a single channel: bi-directional audio. In our model, the attacker can eavesdrop on this channel but can not delay, replay, drop or modify messages without being detected. This is a realistic assumption, since the audio channel is human-perceivable and the user hears any extraneous party that is loud enough to interfere with legitimate communication between paired devices. Since devices are also assumed to be not compromised, the attacker can not delay or drop messages.

Because the user is aware of any interference on the audio channel and cancels the protocol once an attack is detected, the channel is always authenticated provided the secure pairing protocol completes successfully. Note that, if the protocol is cancelled, the exchanged public keys are discarded and no shared key is ever generated. On the other hand, if the attacker allows the protocol to complete by not interfering with communication then the channel is authenticated and attacker's success probability depends on its ability to solve the Computational Diffie-Hellman (CDH) problem, which is believed to be computationally infeasible.

DoS attacks are still possible, however, our protocol has a clear advantage against such attacks, as compared to previous work. Any DoS attack on the audio channel can be immediately noticed and even traced back to its source (in this case, a loud-enough audio source). In summary, audio-based single-channel pairing is secure against MITM attacks as long as the user can positively identify active attacks on the communication channel. Overall, it is the most resilient pairing solution against DoS attacks.

## 7 Implementation

In this section we describe the prototype implementations of the proposed methods.

### *7.1 Two-channel*

Since this method is intended for a variety of mobile computing platforms, portability is a key requirement. We built the prototype using the highly portable *Ewe* Programming System (Brereton, 2005) which facilitates the development of Java applications.[3] Our implementation runs on any Pocket PC (iPAQ in our experiments) as well as on most Windows PCs.

We use 802.11 as the common wireless channel and run the 3-round SAS-based protocol (described in Section 3) after the user initialises the protocol on both devices. After the 3-round exchange is over, the SAS string is converted to the corresponding MadLib sentence. Depending on the hardware, the user(s) have the option to vocalise or display the sentence on either devices. Recall that there are four types of user requirements or settings. Under the TYPE 1 setting, the user hears the same MadLib vocalised by both devices. Under the TYPE 2 and TYPE 3 settings, the user reads the MadLib from one device and compares it to the same MadLib vocalised by the other devices. Finally, TYPE 4 involves the user reading and comparing two Madlibs, one displayed by each device.

Owing to its modular software design, our solution can utilise a variety of Text-to-Speech (TTS) engines. However, most C/C++ speech engines are platform-dependent, while those written for mobile devices are mostly proprietary. Furthermore, Java-based TTS engines are available for specific JVM-s that are unsuitable for resource-constrained devices, such as smartphones and iPAQ-s. Specifically, Sun offers the JSAPI and FreeTTS Java TTS engine implementations. However, these run only on Java 1.4. We employed an existing TTS application, Digit by Digalo,[4] which is a simple lightweight clipboard reader that uses the *Elan Speech Engine*.[5] Our application copies the text to-be-vocalised onto the system clipboard and Digit speaks it out automatically or when the user presses a button on the application window. Digit is initialised and terminated from within the Ewe program.

Ewe does not provide a complete API for low-level cryptographic primitives. Thus, we added a lightweight cryptographic API to Ewe's Java libraries for our needs. For this purpose, we ported the *Bouncy Castle* crypto package (bou, 2006) for JDK 1.3. For hashing we used Ewe's built-in SHA-1.

The FreeTTS and Bouncy Castle crypto package are written solely in Java and do not link to native platform-specific libraries, thus providing platform independence. So far, we tested our system on Pocket PC and Windows PC; it can also be used with other Ewe-supported platforms by simply changing the TTS engine.

### *7.2 Single-channel implementation*

In this implementation, choosing the right codec is a crucial step. The two main requirements are: reliability (low error rate) and high throughput. In this respect, we can use any fast and reliable codec that can cope with reasonable amount of background noise.

Our codec is based on the results of the Digital Voices project at PARC (Lopes, 2001). 240 bits are encoded in a 3.4-second MIDI audio sequence where the first 160 bits represent the actual public key (in the EC-DSA cryptosystem) and the last 80 bits correspond to a folded hash of the public key, for error checking. The Bouncy Castle (bou, 2006) crypto package is used for hash computation. The length of the audio sequence represents a reasonable trade-off between speed and robustness for this codec: raising the tempo (throughput) of the encoder would result in a shorter sequence, but the recording would be less robust against background noise. As mentioned earlier, the resulting sound is unpleasant for the human ear, however, its duration is blessedly short. Alternative implementations produce more pleasant sound at a cost of longer duration: the same number of bits can be encoded in a 15-second audio sequence reproducing a piano score that is more pleasant for the human ear.

## 8 Discussion and limitations

Because audio is probably the most common interface across all personal devices, we believe that proposed approaches can accommodate pairing in for a wide range of scenarios and devices. Most devices equipped with some form of a wireless interface also have at least an audio out interface. Nevertheless, embedding audio-in/out interfaces on devices that do not already possess them would require very little in terms of extra costs. Moreover, audio interfaces tend to take small amounts of packaging and surface area.

Despite many advantages, proposed approaches clearly have some notable limitations. Both of them are a poor choice for users who are hearing-impaired. Also, visually-impaired users would not be able to read text displayed by one or both devices. Furthermore, they are unsuitable for noisy environments, such as factories, convention floors or stadiums. They are also not viable in places where playing loud audio is forbidden, such as movie theaters, libraries, and certain areas in hospitals. If two channels are used, both devices must be within hearing range of the user. If only the audio channel is used to transfer data, sufficient proximity is required so that the personal device can *hear* its counterpart.

Our experiments were conducted with devices separated by 1–2 feet (30–60 cm) with the user standing between them. In noise-free settings, larger distances are possible. However, we do not expect to exceed 5–6 feet (1.5–1.8 m) with commodity devices, such as PDAs or cellphones. We also note that many prior methods (Stajano and Anderson, 2000; McCune et al., 2005; Feeney et al., 2002; Holmquist et al., 2001; Balfanz et al., 2002; Kindberg and Zhang, 2003b) have the same proximity limitations. In contrast, we expect that the Blinking-Lights method (Saxena et al., 2006) and the laser-based (Kindberg and Zhang, 2003a) techniques are somewhat better in this regard, allowing distances upwards of 20 feet (6 m).

In addition to protection against impersonation and MiTM attacks, proposed approaches offer users the ability to detect some on-going DoS attacks in real time. Other techniques can do little against DoS attacks that aim to jam device interfaces used for the human-imperceivable communication. (Digital jamming is not readily detectable by humans.) Our single-channel approach allows a user to detect the presence of extraneous noise and perhaps even to identify its source.

Although we did not yet conduct any formal usability studies for the methods described here, some prior work analysed the usability of user comparison of MadLib sentences (Soriente et al., 2007). Based on previous results, we can confidently say that this more efficient implementation of the two-channel authentication method is equally usable and places relatively low burden on the user.

On the other hand, we are unaware of any previous work that analyses the human ability to detect audio sources in the context of secure device pairing. However, there is extensive literature in other fields (such as Pickles, 1982; Stevens and Davis, 1938; von Békésy and Wever, 1960), which clearly confirms that a healthy human is quite capable to identify the source of a sound (i.e., perform sound localisation). Since sound localisation is exactly what we are asking from the user in the single-channel pairing, we argue that an average human being with no hearing impairment can successfully determine whether the sound originates from the intended device(s).

We have performed a small-scale (five-person) test trial where an extraneous remote-controlled device was hidden in varying locations near the genuine devices during the pairing protocol execution: behind the user, under the user's chair or under the desk in front of the user. We triggered the remote-controlled device to transmit data, while the genuine devices were exchanging their public keys. All five participants easily detected the third (extraneous) device. Although previous work and our initial studies suggest that the security based on human sound localisation ability is robust, a formal study with a much bigger participant group is needed to determine the error rate and exact security of this specific application.

## 9 Related work

As mentioned earlier, PKI-oriented solutions involve rigid hierarchies and require the existence of trusted offline Certification Authorities (CAs) (Kohnfelder, 1978). Unstructured certification techniques such as PGP (Zimmermann, 1995) assume a certain small degree of separation among certified entities (i.e., a web of trust). An alternative is an online TTP, such as Kerberos (Miller et al., 1987; Steiner et al., 1998). However, such solutions require constant presence and availability of an online TTP which is not realistic in our envisaged scenarios. For this reason, the remainder of this section focuses on closely related prior research.

Stajano and Anderson (2000) made a seminal contribution by bringing the problem into the spotlight. The proposed techniques, however, required the use of standardised physical interfaces and cables. The follow-on work by Balfanz et al. (2002) and Feeney et al. (2002) made progress by suggesting the use of infrared communication as the human-verifiable side-channel. Though timely in its day, this approach is no longer viable since:

- few modern devices are equipped with IrDA interfaces
- the infrared channel itself is not fully immune to DoS and MiTM attacks.

Capkun et al. (2003) proposed a further extension to allow two previously unassociated devices to establish a key utilising one-hop transitive trust.

Another approach involves graphical visualisation of the hash of the exchanged cryptographic material. The user needs to compare the visual output on both devices. In order to make the comparison easier, researchers devised visual metaphors to represent the hash. Levien a proposed a 'snowflake' mechanism (Levien, 1996). Relyinh on the user's ability to recognise pictures (Perrig and Song, 1999) proposed *visual hashes* – the output of an algorithm capable of mapping random strings to fixed-sized images with properties similar to that of a hash function. Their work demonstrated how visual hashes can be used to authenticate the root key of a CA with human assistance: the CA publishes the visual hash of its key in a newspaper and each user can easily verify whether the output of their local HVA with the CA key as input, received through another channel (e.g., the internet) matches the one in the newspaper. Since relatively high-resolution displays are required, this approach is suitable only for certain types of devices, such as laptops, digital cameras, PDAs or high-end phones. Last, Dohrmann and Ellison devised a colourful 'flag' representation (Ellison and Dohrmann, 2003).

The *Seeing-is-Believing* technique by McCune et al. (2005) uses the visual channel to perform secure device pairing. One device sends its public key to the other through a human-imperceivable channel (such as 802.11) and, at the same time, displays a visual encoding of the public key in the form of a bar code. (If there is no display, the use of barcode stickers is suggested). The receiver device, with the help of the user, takes a picture of the bar code and compares it with the one computed locally, using the received public key as input. The protocol does not rely on human visual ability, i.e., the user is not required to recognise pictures, but it requires at least one device to have a photo camera, and the quality of the picture, either printed or displayed, to be quite good. If both devices must send their cryptographic material, then each requires a photo camera and the above sequence must be repeated twice, thus increasing user burden.

Recently, Saxena et al. (2006) considered a variation of Seeing-is-Believing (McCune et al., 2005) method by showing how to achieve secure pairing if one device is equipped with a camera, while the other has at least a single LED. As before, the two devices exchange public keys via some wireless channel, such as 802.11. Then, the device equipped with an LED uses its blinking capability to transmit the hash. The device with a camera records the blinking pattern, extracts the hash and compares with the hash computed as a result of the protocol. If they match, it asks the user to accept on the other device; otherwise it asks the user to abort. This protocol requires less in terms of device features, but not all devices have a light detector or a camera. Moreover, the comparative usability study in Uzun et al. (2007) indicates that users are generally not adept in following the prescribed order of interaction if it involves more than one device.

Manual peer device authentication has also been studied in Maher (1995), Gehrmann et al. (2004), Gehrmann and Nyberg (2004) and Larsson (2001). The common element of these solutions is that the user is required to compare short numerical check values, which are generated by hashing or taking the MAC of the authentication object. Their limitation is that sufficient security dictates that the check values need to be relatively lengthy (substantially more than the 4-digit hex vector (Maher, 1995) suggests) rendering their comparison a cumbersome and error-prone task for humans. Gehrmann et al. (2004) improves on Maher (1995) by shortening the required length of the check value, yet the user is burdened with the task of typing a short 2–4 hex digit key using the non-user-friendly input interface of a personal device. Larsson (2001) also proposes a solution to reduce the length requirement for the check values. However, it requires a temporary shared secret between the two peers, thus it is not applicable for devices with no prior association. More recently, Cagalj et al. (2006) tackled the problem of user-friendly mutual authentication with three commitment-based (Vaudenay, 2005) techniques. The first requires the user to compare approximately half the number of bits of the previous solutions. The other two techniques rely on the specifics of the radio channel; they use distance bounding and integrity codes.

There have been other proposals suggesting the use of technologies that are more expensive and less common. Kinberg et al. suggested an approach requiring RF and ultrasound receiver/transmitters on both devices in their earlier work (Kindberg and Zhang, 2003b) and laser technology (requires each device to be equipped with a laser transceiver) in their more recent proposal (Kindberg and Zhang, 2003a). Holmquist et al. *Smart-Its-Friends* system (Holmquist et al., 2001), proposed the use of a common movement pattern as the security initiator when the two devices are shaken together. In addition, Mayrhofer et al. proposed 'Shake well before use' (Mayrhofer and Gellersen, 2007). Both approaches require the devices to be equipped with two-axis accelerometers. Since a user is required to shake both devices together, they are unsuitable for physically large/bulky devices. A more recent proposal (Mayrhofer et al., 2007) also uses ultrasound to verify the location of the transmitting devices and as the auxiliary channel for message transfer as part of a key verification protocol.

Several standardisation bodies also recognised the importance of the problem and have begun working on specifying more usable and more secure procedures for device association. Wi-Fi Alliance is working on specifications for Wi-Fi Protected Setup (Alliance, 2007). Microsoft has released specifications for Windows Connect Now-NET (Microsoft, 2006), which is closely related to Wi-Fi Protected Setup. Bluetooth Special Interest Group has released specifications on Simple Pairing (Group, 2006). The Universal Serial Bus (USB) forum has recently released the specifications for Wireless USB Association Models (Specification, 2006) which specifies the procedures for pairing two Wireless USB devices. Unlike research proposals, standards specifications have to consider devices with a wide range of hardware capabilities. Consequently, specifications do not dictate a single pairing method. All of them support the use of at least one type of auxiliary channel. For example, Bluetooth Simple Pairing supports the use of Near Field Communication (NFC) and Wireless USB Association Models support the use of USB cables.

In summary, the aforementioned techniques and specifications require particular hardware and/or interfaces that may not be available on many devices. There are common pairing scenarios, such as a wireless printer and a laptop, an access point and a PDA, or a wireless headset and a desktop, which are not supported by any of the previously mentioned protocols. Even in some pairing scenarios where the previous schemes seem to apply, one would still need a combination of several such schemes to accommodate a considerable fraction of possible pairing scenarios. Moreover, the usability of such a combination would be very questionable, especially since no comprehensive usability study has been performed for many of these complex schemes. Moreover, even the very basic pairing methods have not fared very well when employed by ordinary (technically non-savvy) users (Uzun et al., 2007).

## Acknowledgements

This work was supported in part by the National Science Foundation under grants CCR-0312760, OCI-0724806, and IIS-0713046. Preliminary results of this research appeared in Goodrich et al. (2006).

Products mentioned in this article are solely owned by their trademark owners and we claim no rights to those trademarks.

## References

Bouncy Castle Crypto APIs (2006) http://www.bouncycastle.org/

Alliance, W. (2007) 'Wi-fi protected setup specification', *WiFi Alliance Document*, January.

Balfanz, D., Smetters, D., Stewart, P. and Wong, H.C. (2002) 'Talking to strangers: authentication in ad-hoc wireless networks', *Symposium on Network and Distributed Systems Security (NDSS '02)*, San Diego, California, USA.

Brereton, M. (2005) *Ewe Java vm for Pocketpc*, Available at http://www.ewesoft.com/

Cagalj, M., Capkun, S. and Hubaux, J. (2006) 'Key agreement in peer-to-peer wireless networks', *Proceedings of the IEEE (Special Issue on Cryptography and Security)*, pp.467–478.

Capkun, S., Hubaux, J. and Buttyan, L. (2003) 'Mobility helps security in ad hoc networks', *Proceedings of the ACM Symposium on Mobile Ad Hoc Networking and Computing(MobiHoc 2003)*, pp.46–56.

Diffie, W. and Hellman, M.E. (1976) 'New directions in cryptography', *IEEE Transactions on Information Theory*, IT Vol. 22, No. 6, pp.644–654.

Ellison, C. and Dohrmann, S. (2003) *Public-key Support for Group Collaboration*, New York, NY, USA, ACM Press, Vol. 6, pp.547–565.

Feeney, L.M., Ahlgren, B., Westerlund, A. and Dunkels, A. (2002) 'Spontaneous networking for secure collaborative applications in an infrastructureless environment', *Demonstration session: Int'l Conf. on Pervasive Computing (Pervasive 2002)*, December.

Garey, M.R. and Johnson, D.S. (1990) *Computers and Intractability; A Guide to the Theory of NP-Completeness*, W.H. Freeman & Co., New York, NY, USA.

Gehrmann, C., Mitchell, C. and Nyberg, K. (2004) 'Manual authentication for wireless devices', *RSA Cryptobytes*, Vol. 7, No. 1, p.2937.

Gehrmann, C. and Nyberg, K. (2004) 'Security in personal area networks', in Mitchell, C.J. (Eds.): *Security for Mobility*, IEE, London, pp.191–230.

Goodrich, M.T., Sirivianos, M., Solis, J., Tsudik, G. and Uzun, E. (2006) 'Loud and clear: human-verifiable authentication based on audio', *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, Lisboa, Portugal, pp.10–17.

Group, B.S.I. (2006) *Simple Pairing Whitepaper*, http://www.bluetooth.com/Bluetooth/Apply/Technology/Research/Simple_Pairing.htm.

Haller, N. (1995) Available at rfc1760: The s/key one-time password system.

Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M. and Gellersen, H-W. (2001) 'Smart-its friends: a technique for users to easily establish connections between smart artifacts', *UbiComp '01: Proceedings of the 3rd International Conference on Ubiquitous Computing*, Springer-Verlag, London, UK, pp.116–122.

Juola, P. and Zimmermann, P. (1996) 'Whole-word phonetic distances and the pgpfone alphabet', *Proceedings of the Fourth International Conference on Spoken Language Processing (ICSLP)*, Vol. 1, http://www.asel.udel.edu/icslp/cdrom/vol1/005/a005.pdf.

Kindberg, T. and Zhang, K. (2003a) 'Secure spontaneous device association', in Dey, A.K., Schmidt, A. and McCarthy, J.F. (Eds.): *Ubicomp*, Vol. 2864 of Lecture Notes in Computer Science, Springer, pp.124–131.

Kindberg, T. and Zhang, K. (2003b) 'Validating and securing spontaneous associations between wireless devices', in Boyd, C. and Mao, W. (Eds.): *ISC*, Vol. 2851 of Lecture Notes in Computer Science, Springer, pp.44–53.

Kohnfelder, L.M. (1978) *Towards a Practical Public-key Cryptosystem*, BSc Thesis, MIT Department of Electrical Engineering, Cambridge, MA, USA.

Kugler, D. (2003) 'Man in the middle attacks on bluetooth', *Financial Cryptography*, Lecture Notes in Computer Science, Guadeloupe, French West Indies, pp.149–161.

Larsson, J-O. (2001) *Higher Layer Key Exchange Techniques for Bluetooth Security*, Open Group Conference, Amsterdam.

Laur, S., Asokan, N. and Nyberg, K. (2006) 'Efficient mutual data authentication using manually authenticated strings', in Pointcheval, D., Mu, Y. and Chen, K. (Eds.): *CANS*, Vol. 4301 of *Lecture Notes in Computer Science*, Springer, pp.90–107.

Levien, R. (1996) *Pgp Snowflake*, Source code available at: http://packages.debian.org/testing/graphics/snowflake.html

Lopes, C. (2001) *The Digital Voices Project Home Page*, http://www.isr.uci.edu/lopes/dv/dv.html

Maher, D.P. (1995) *Secure Communication Method and Apparatus*, US Patent Number 5,450,493.

Mayrhofer, R. and Gellersen, H. (2007) 'Shake well before use: authentication based on accelerometer data', *Proc. Pervasive 2007: 5th International Conference on Pervasive Computing*, Toronto, Canada, pp.144–161

Mayrhofer, R., Gellersen, H. and Hazas, M. (2007) 'Security by spatial reference: using relative positioning to authenticate devices for spontaneous interaction', *Ubicomp*, Innsbruck, Austria, pp.199–216.

McCune, J.M., Perrig, A. and Reiter, M.K. (2005) 'Seeing-is-believing: using camera phones for human-verifiable authentication', *Proc. S&P2005: IEEE Symposium on Security and Privacy*, Oakland, CA, USA, pp.110–124.

Microsoft (2006) *Windows Connect Now-ufd and Windows Vista Specification*, version 1.0, http://www.microsoft.com/whdc/Rally/WCN-UFD Vistaspec.mspx, Microsoft

Miller, S., Neuman, C., Schiller, J. and Saltzer, J. (1987) 'Kerberos authentication and authorization system', *Project Athena Technical Plan*, section E.2.1.

Pasini, S. and Vaudenay, S. (2006) 'SAS-based authenticated key agreement', in Yung, M., Dodis, Y., Kiayias, A. and Malkin, T. (Eds.): *Public Key Cryptography*, Vol. 3958, Lecture Notes in Computer Science, Springer, pp.395–409.

Perrig, A. and Song, D. (1999) 'Hash visualization: a new technique to improve real-world security', *Proceedings of the 1999 International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99)*, pp.131–138.

Pickles, J. (1982) *An Introduction to the Physiology of Hearing*, Academic Press, San Diego.

Saxena, N., Ekberg, J-E., Kostiainen, K. and Asokan, N. (2006) 'Secure device pairing based on a visual channel', *2006 IEEE Symposium on Security and Privacy*, Oakland, California, USA, pp.306–313.

Soriente, C., Tsudik, G. and Uzun, E. (2007) 'Hapadep: human asisted pure audio device pairing', *Cryptology ePrint Archive*, Report 2007/093.

Specification, W.U. (2006) *Association Models Supplement*, revision 1.0, In USB Implementers Forum, http://www.usb.org/developers/wusb/

Stajano, F. and Anderson, R. (2000) 'The resurrecting duckling: security issues for ad-hoc wireless networks', *Security Protocols: 7th International Workshop*, Springer, pp.204–214.

Steiner, J.G., Neuman, C. and Schiller, J.I. (1998) *Kerberos: An Authentication Service for Open Network Systems*, Manuscript.

Stevens, S. and Davis, H. (1938) *Hearing, Its Psychology and Physiology*, Wiley, New York.

Uzun, E., Karvonen, K. and Asokan, N. (2007) *Usability Analysis of Secure Pairing Methods*, Technical Report NRC-TR-2007-002, Nokia Research Center.

Vaudenay, S. (2005) 'Secure communications over insecure channels based on short authenticated strings', *Advances in Cryptology – CRYPTO '05: The 25th Annual International Cryptology Conference*, Vol. 3621, Lecture Notes in Computer Science, Santa Barbara, California, USA, Springer-Verlag, August, pp.309–326.

von Békésy, G. and Wever, E. (1960) *Experiments in Hearing*, McGraw-Hill, New York.

Zimmermann, P. (1995) *The Official PGP User's Guide*, MIT Press, Cambridge, MA, USA.

## Notes

[1]However, note that TYPES 2–4 use cases all require non-visually-impaired users.

[2]In a MadLib puzzle, a funny story is created by having blanks in a text filled in with syntactically-appropriate words chosen by the player or an opponent. Mad Libs is a registered trademark of Penguin Group (USA) Inc.

[3]Ewe is currently available for the following platforms: Pocket PC (Windows CE), MS SmartPhone, Casio BE-300, HandHeldPC Pro, Sharp Zaurus, Linux PC, Windows PC and any Java 1.2 VM.

[4]See: www.digalo.com

[5]See: www.elantts.com