HAPADEP: Human-Assisted Pure Audio Device Pairing [*]

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

ABSTRACT

The number and diversity of electronic gadgets has been steadily increasing and they are becoming indispensable to more and more professionals and non-professionals alike. At the same time, there has been fairly little progress in secure pairing of such devices. The pairing challenge revolves around establishing on-the-fly secure communication without any trusted (on- or off-line) third parties between devices that have no prior association. The main security issue is the danger of so-called Man-in-the-Middle (MiTM) attacks, whereby an adversary impersonates one of the devices by inserting itself into the pairing protocol. One basic approach to countering these MiTM attacks is to involve the user in the pairing process. Therein lies the usability challenge since it is natural to minimize user burden. Previous research yielded some interesting secure pairing techniques, some of which ask too much of the human user, while others assume availability of specialized equipment (e.g., wires, photo or video cameras) on devices. Furthermore, all prior methods assumed the existence of a common digital (human-imperceptible) communication medium, such as Infrared, 802.11 or Bluetooth.

In this paper we introduce a very simple technique called HAPADEP (Human-Assisted Pure Audio Device Pairing). It places very little burden on the human user and requires no common means of electronic communication. Instead, HAPADEP uses the audio channel to exchange both data and verification information among devices. It makes secure pairing possible even if devices are equipped only with a microphone and a speaker. Despite its simplicity, a number of interesting issues arise in the design of HAPADEP. We discuss design and implementation highlights as well as usability features and limitations.

1. INTRODUCTION AND MOTIVATION

The popularity of sophisticated personal devices, such as PDAs, multimedia players, cameras and smartphones, has prompted the need for security mechanisms specifically tailored for such devices. One of the main challenges is the problem referred to as *Secure First Connect*, *Secure Initialization* or *Secure Device Pairing*.[1] 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 on-line Trusted Third Party (TTP), no off-line Certification Authority (CA), no Public Key Infrastructure (PKI) and, of course, no common secrets.

The ideal solution must be seamless and transparent to the human user(s). In theory, the problem is easily surmountable. It is not hard to imagine a standardized device pairing approach, supported by a common PKI, adopted by all device manufacturers. However, this solution is, at best, years away. Any such *standard* would, no doubt, require endless deliberations by industry-wide committees and countless revisions, not to mention the usual manufacturing delays (The cell phone industry is a prominent example).

Taking into account the non-existence of a PKI, user assistance in the device pairing process is simply unavoidable. This is because of the very real threat of so-called Man-in-the-Middle (MiTM) attacks. A MiTM attack can occur whenever unauthenticated communication is involved. One of the best-known examples is the textbook Diffie-Hellman Key Exchange protocol [16]. 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.

Since no pre-shared secrets and no other means of authentication exists between two unfamiliar devices and the initial communication transpires over some human-imperceptible medium (e.g., infrared or radio), MiTM attacks represent a danger that is hard to ignore. The security research community has recognized, and began responding to, this challenge starting in the late 1990-s, i.e., the pioneering work of Stajano, et al. [6]. A number of techniques have been proposed since then, varying greatly in the assumptions about device features, degree and nature of user involvement as well as environmental factors.

[*] (Permission block, and copyright information)

[1] In this paper, we use the term "device pairing" to refer to the problem at hand.

Despite significant recent progress, the design space of the device pairing problem has not been fully explored. In particular, two issues remain un-addressed:

- Denial-of-Service (DoS): unlike MiTM attacks, DoS attacks aim to prevent communication. In the device pairing setting, the goal of a DoS attack is to preclude the two devices from establishing a secure channel. Such an attack is trivial to mount: the adversary only needs to jam the interface of one or both devices. This is easy with Bluetooth or 802.11, but a little harder (since it requires line-of-sight) – yet still doable – with Infrared. The gist of the problem is the inability of the human user not only to determine the source of the attack but even to detect its presence.

- Common Channel: all prior techniques require the existence (and set-up) of a common means of electronic communication between two devices. The devices must have at least one interface in common, be it wired (as in [6]) or wireless (as in [9, 7, 14, 10, 11]). This can be problematic, for two reasons:

  (1) The two devices might not (at least at the time of pairing) have a common interface, e.g., one only has a Bluetooth interface, while the other – 802.11. Why pair such devices? One reason could be because they would later connect to the Internet via respective interfaces and need to communicate securely.

  (2) Wireless interfaces typically take some time and effort to set up and may be a true challenge for an ordinary user. For example, in case of 802.11-equipped devices, each must be put into the ad hoc mode and an ad hoc network must be manually configured. Infrared (IrDA) requires to be activated manually on both devices; it also requires line-of-sight alignment between transceivers. In case of Bluetooth, one device needs to be *discoverable* and then the other must *discover* it. This is tricky if there are many Bluetooth-enabled devices around, and may turn into a real headache if multiple devices share the same (e.g., default) name. Figure 1 shows some screen-shots from connection establishment using 802.11 and Bluetooth.

The work described in this paper attempts to fill the gap left by prior techniques. The proposed protocol – HAPADEP or Human-Assisted Pure Audio Device Pairing – obviates the need for a common human-imperceptible communication channel. It uses the audio channel for communicating both protocol messages and human-perceived authentication/verification information. HAPADEP takes advantage of the fact that most modern devices are equipped with audio in/out interfaces, i.e. a speaker and a microphone, and such interfaces are very inexpensive. Also, as described further in the paper, HAPADEP offers natural means of protection against both DoS and MiTM attacks.

The rest of this paper is organized as follows: we survey related work in section 2 and describe the HAPADEP protocol in section 3, followed by several use case scenarios in section 4. We then describe two HAPADEP variants in section 5 and discuss their respective usability factors in section 6. We then give a brief security analysis in section 7 and address certain limitations in section 8. The paper ends with the discussion in section 9 and summary in section 10.

2. RELATED WORK

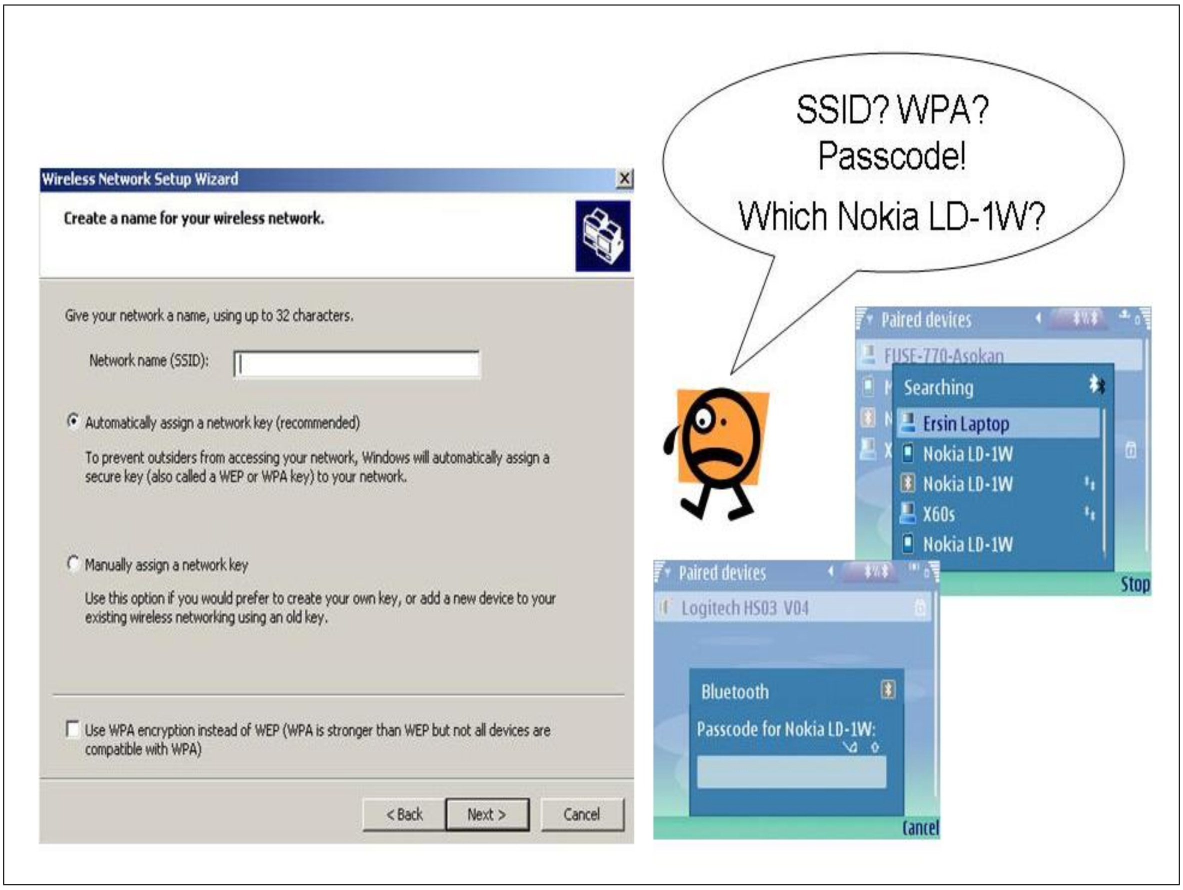
There is a fairly large body of relevant prior work on the general topic of secure device pairing.

The earliest work by Stajano, et al. [6] made a seminal contribution by bringing the problem into the spotlight. The proposed techniques, however, required the use of standardized physical interfaces and cables. The follow-on work by Balfanz, et al. [4] and Feeney, et al. [5] 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: (1) few modern devices are equipped with IrDA interfaces (they are too slow, short-distance and require line-of-sight) and (2) the infrared channel itself is not fully immune to DoS and MiTM attacks.

Another early result by Perrig and Song [3] defines a *visual hash* as the output of a Hash Visualization Algorithm (HVA) – an algorithm capable of mapping random strings to fixed-sized images with properties similar to that of a hash function. This work showed how visual hashes can be used to authenticate the root key of a Certification Authority (CA) with human assistance: 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 a secondary channel (e.g., the Internet) matches the one in the newspaper. The *visual hashes* technique avoids the cumbersome and error-prone process of comparing two hashes byte-by-byte and relies on the user's ability to recognize pictures. Since relatively high-resolution displays are required, this approach is suitable only for certain types of devices, such as laptops or digital cameras.

The *Seeing-is-Believing* technique by McCune, et al. [9] uses the visual channel to perform secure device pairing. One device sends its public key to the other through a human-imperceptible 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 recognize 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. [14] considered a variation of Seeing-is-Believing [9] method by showing how to achieve secure pairing if one device is equipped with a light detector, 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 light detector 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 user to abort. The resulting protocol offers less user burden than SiB and requires less in terms of device features. (However, since few devices have light detectors, the concept was demonstrated using one device equipped with a video camera taking a video clip of the other device which repeatedly turned its display on



Wireless Network Setup Wizard

Create a name for your wireless network.

Give your network a name, using up to 32 characters.

Network name (SSID):

● Automatically assign a network key (recommended)

To prevent outsiders from accessing your network, Windows will automatically assign a secure key (also called a WEP or WPA key) to your network.

○ Manually assign a network key

Use this option if you would prefer to create your own key, or add a new device to your existing wireless networking using an old key.

□ Use WPA encryption instead of WEP (WPA is stronger than WEP but not all devices are compatible with WPA)

Figure 1: Connection setup in 802.11 and Bluetooth

and off.)

Another pairing approach uses a different human-perceptible channel – audio – in the *Loud-and-Clear* system [7]. As usual, the proposed protocols involve two devices exchanging their keys and computing the hash of the exchanged cryptographic material. The hash is later translated in a syntactically correct English-like "Madlib" (non-sensical) sentence that can be played by one of the devices and showed on a text display on the other: the user compares the sequences and verifies the key exchange. The authors consider many other scenarios and variations of the protocol, but the main contribution is the ability to perform secure device pairing between a device equipped with a speaker and another equipped with a simple (line) display. Moreover, the generation of syntactically correct text helps the user in the verification process. Loud-and-Clear also supports the scenario where both devices have speakers but neither has a display.

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 [11] and laser technology (requires each device to be equipped with a laser transreceiver) in their more recent proposal [10]. Holmquist,, et al. *Smart-Its Friends* system [8], proposed the use of a common movement pattern as the security initiator when the two devices are shaken together. This requires both devices to be equipped with two-axis accelerometers; it is also unsuitable for physically large devices.

All aforementioned techniques adhere to one common design element: they use two channels to perform the pairing process. The primary human-imperceptible channel is used to exchange the cryptographic material, and, then, a secondary human-perceptible channel is used to verify the integrity of the process. The main drawback of this approach is the requirement that devices must have a common communication channel, such as 802.11a/b/g/n, Bluetooth, IrDA, and WiBro. In some scenarios, a common communication channel might not be available at the time of pairing and, even if available, establishing communication over it might be time-consuming and cumbersome.

3. HAPADEP PROTOCOL: GENERAL OPERATION

The HAPADEP protocol relies only on the audio channel. A speaker and a microphone are the only required device features. (In fact, in its simplest, unilateral flavor, HAPADEP eliminates the need for a microphone in one of the devices.) We consider both unilateral and bilateral key exchange protocol flavors. When talking about the former, we use the term *personal* to denote the device receiving the public key and *target* to denote the device sending (delivering) its public key. In the bilateral protocol, we use a more generic term *peer*.

A unilateral protocol is said to be *verifiable* if the user, at the end of the protocol execution, is sure that one device has correctly received the public key sent by the other device. (The same property trivially extends to bilateral key exchange). The notion of verifiable key exchange is somewhat different from the notion of authenticated key exchange (which is well-known in the cryptographic literature): the latter guarantees the *owner* of the received cryptographic material, while the former guarantees its *sender*. However, since pairing scenarios involve direct exchange of information – rather than delivery via intermediaries – verified key exchange is sufficient to ensure security.

The protocol consists of two phases, as shown in Figure 2. During the first *transfer* phase the target device sends its public key (and any other cryptographic material) to the personal device; during this phase, the user is only responsible for triggering the execution of the protocol, i.e., pushing a button on each device. To transfer the key over the audio channel, the target encodes its key using the codec and plays the resulting audio sequence. The personal device records the audio sequence and decodes it, using the corresponding decoder, to retrieve the key. In a bilateral key exchange protocol, the transfer phase is repeated with the devices that automatically switch roles. Since encoded public keys are fairly large (several hundred bits), encoding is done using a *fast* codec that provides faster data transmission rate. Although the faster transmission rate has a side-effect of producing rather unpleasant-sounding audio, the transfer only takes about 3 seconds (6 seconds for the bilateral case). Moreover, the human user is not expected to pay close attention or be actively involved at this stage.

During the second *verification* phase the user is directly involved in verifying that the key exchange is secure and successful. A relatively *slow* codec is used to encode the digest of the cryptographic material exchanged in the *transfer* phase. The bit rate of this codec can be much lower than the *faster* codec used in the first phase[2], however, the output needs to be pleasant to the human ear and easily recognizable. In the verification phase, each device plays the audio sequence created using the *slow* codec and waits for the user to decide whether the two sequences match. The user is expected to listen to both sequences carefully and indicate (e.g., by pressing a key) a match or lack thereof.

It might seem like the verification phase is unnecessary. Indeed, we could imagine a simpler protocol which would avoid the verification phase relying instead on the human user to detect exactly *which* device is playing during the transfer phase: if the target is actually playing the audio sequence, any audible MiTM will be noticed. However, two issues would arise: First, the output of the fast codec is not pleasant to the human ear which might discourage users from paying attention and identifying the audio source. Second, it is important to verify the successful termination of the protocol and give a satisfying proof to the user. Note that the user can be sure that the target device is playing but s/he cannot tell whether or what the personal device is recording.

Another naïve approach might be to use the slow (more pleasant-sounding) codec for the cryptographic exchange and play the recorded audio on the receiver. Then, the user would be expected to determine whether the sequentially played sequences by the devices are the same. However, the amount of data encoded as audio would be much larger which would result in long audio sequences, making it tiresome for the user.

Since both devices are capable of playing, recording and comparing audio sequences, it might also seem possible to have the devices themselves (without any user involvement) check whether the two audio sequences played during the verification phase are identical. This approach would remove all burden from the user. However, it would be easy for a malicious device to participate to both transfer and verification phases, pretending to be the target device. In other words, the presence of the user telling which device (among several) is playing is an essential feature termed by Balfanz, et al. as *Demonstrative Identification* [4].

If verification is fully automated, the user is expected to be present and terminate the protocol whenever s/he hears any audio interference (she has to do that even if the devices are indicating successful completion). This assumption would actually harm the usability and the security of protocol since the user reaction would be unpredictable if there are other sounds in the environment. For example, a user pairing two devices in an open cafe' might not be able to detect whether the source of interference is due to the engine noise of the passing cars, the coffee grinder behind the counter, or an actual attacker (many devices can produce sounds somewhat similar to the *fast* codec output). In this case, the pairing might not complete if the user chooses to carefully follow instructions. Or, if the user does not follow instructions, s/he might end up pairing with the attacker's device.

Since the protocol must be fast, secure and usable, we claim that the need for two phases and two different codecs is well-justified.

4. COMMON USE SCENARIOS

We consider several common HAPADEP use scenarios.

- **Type 1:** The simplest is the unidirectional (one-way) protocol whereby the target device (often multi-user in nature, e.g., a printer) delivers its public key to a personal device (e.g., a cellphone).

- **Type 2:** The bi-directional version involves both devices exchanging respective public keys, yet not computing any shared secret or confirming ownership (of public keys).

- **Type 3:** The next scenario also entails bidirectional public key exchange but with devices confirming ownership of public keys, by signing with the corresponding private key, e.g., via a variation of the well-known Station-to-Station protocol (STS) [15]. The establishment of a shared secret is optional in this scenario.

To run the protocol in Type 1 scenario, only the personal device is required to have a speaker and a microphone. The user triggers the execution of the protocol by putting the personal device in recording mode and the target device in playing mode. The latter plays the audio sequence that represents the encoded version of its public key, while the former records it. After recording, the receiver decodes the audio sequence and obtains the key. To verify, the user prompts (e.g., by pressing a button on each) the devices to play out the audio sequence. (The target device does

[2] Since verification data is small, the low bit rate is not a concern here

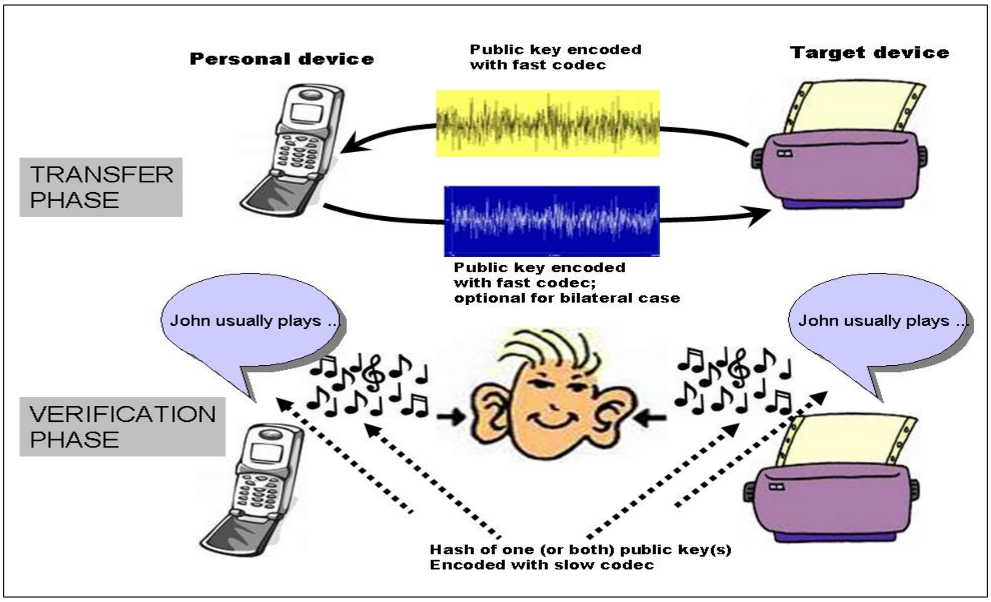

Personal device
Public key encoded with fast codec
Target device
TRANSFER PHASE
Public key encoded with fast codec; optional for bilateral case
John usually plays ...
John usually plays ...
VERIFICATION PHASE
Hash of one (or both) public key(s) Encoded with slow codec

Figure 2: HAPADEP Operation

not record anything and thus does not need an audio-in interface). As mentioned above, we use a *slow* codec to produce an audio sequence that is easily recognizable by human ear. Barring any ambient interference, the user can tell whether both devices are playing the same sequence.

It is easy to see that Type 2 and 3 scenarios are a poor match for a multi-user target device, such as a public printer, fax or wireless access point. This is because such devices provide a common service (printing, faxing, net access) and may not care about authenticating the user's personal device, at least not at the point of initial contact. Whereas, scenarios 2 and 3 are very appropriate for pairing single-user (or personal) devices like Laptops, PDAs, mobile phones, Bluetooth headsets, etc. Unlike Type 1, Types 2 and 3 scenarios require both devices to have audio-in and audio-out interfaces.

Two key differences between Type 2 and 3 scenarios are: (1) the delay, and (2) the cost incurred by cryptographic operations. In Type 2, the protocol involves two fast-codec noisy messages (to exchange the two public keys) between devices and a final slow-codec human-verifiable audio sequence which corresponds to the hash of both keys. In Type 3, on the other hand, the two devices must exchange at least three fast-codec (noisy) messages before the final human-verifiable sequence. Also, each device incurs the cost of (at least) one signature generation and one signature verification.

5. IMPLEMENTATION

In the implementation of HAPADEP, choosing the right codecs is crucial. In the case of *fast* codec, the two main requirements are reliability (low error rate) and high bit-rate. In this respect, any reliable and fast codec that can cope with reasonable amount of background noise can be used in the implementation. The *slow* codec, on the other hand, has to be chosen very carefully since it directly affects the usability and the security of the protocol. In HAPADEP, user's ability to tell whether the verification sequences are the same one or they are different is crucial for the security of the protocol. The bit rate of the *slow* codec is not a concern, but the output has to be easily recognizable by the user.

In our implementation, the *fast* codec is based on the results of the *Digital Voices* project at PARC [13]. 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 is a folded hash of the public key for error checking. The *Bouncy Castle* [1] crypto package is used for hash computation. The length of the audio sequence is a reasonable trade-off between speed and robustness for this codec: raising the tempo (bit rate) of the encoder would result in a shorter sequence, but the recording would be less robust against background noise.

For the human-verifiable audio generation (*slow* codec), we implemented (and experimented with) two different approaches:

- Using pleasant-sounding short melodies
- Using grammatically correct MadLib sentences

The implementations details for each approach are as follows:

Generating a Melody: The codec uses the hash of the exchanged cryptographic protocol data to produce a MIDI score played by a piano. Using this codec, playing time for the resulting MIDI sequence (generated from a 80 bit input) takes about 5.6-seconds. For each byte of the input, the first four bits are used to select a specific *symbol* to add to the score from among the seven major chords, the seven minor chords, pause and sustain; the second four bits together with the previous ones and the present octave, are used to select the octave at which the note is played. The result is an easily recognizable audio sequence, very similar to sounds produced by a child playing a toy piano. Each device can replay the sequence multiple times so that the user (if desired) can make sure that the two devices play the same (or different) sequences.

Table 1: Participant Profile

Gender	Male	75%
	Female	25%
Age	18-24	30%
	25-29	40%
	30-34	20%
	35-40	5%
	40+	5%
Education	High School	5%
	Bachelor	20%
	Masters	60%
	PhD	15%
Playing any instrument	YES (Amateur 100%)	35%
	NO	65%
Professional music activity	YES	5%
	NO	95%

Generating a Sentence: Another way to convert cryptographic data to human-verifiable audio is by producing grammatically correct sentences. The logic is similar to the MadLib game which was also used in the *Loud-and-Clear* [7] device pairing technique. We employed the same MadLib generation method (in fact, we used the same code) to create grammatically correct but non-sensical English sentences consisting of 8 S/KEY-generated words. Each word is chosen from a dictionary of size 2^{10}; the input length is 80 bits (same as with the melody generation).

6. USABILITY ANALYSIS

Armed with two versions of HAPADEP software, we were interested to investigate their respective usability factors. To this end, we performed usability experiments discussed in this section.

We recruited 20 subjects for the experiments. Subjects were chosen on a first-come first-serve basis from the respondents to recruiting posters. Subjects were mainly university students which resulted in a fairly young, well-educated and technology-savvy participant group. The demographics and related background information of the participants are summarized in Table 1.

Test Procedure: Our usability tests were conducted in a variety of campus venues (depending mainly on the subjects' preferences), including, but not limited to: cafés, student dorms and apartments, classrooms, office spaces and outdoor terraces. After giving a brief overview of our study goals, participants were asked to fill out the background questionnaire (see Figure A-1 in the Appendix) to collect demographic information and learn about their music- and computer-related skills and background. Next, users were given a brief introduction to the mobile phones used in the tests to demonstrate some basic operations needed during the test.

Each user was then given the two devices and asked to follow on-screen instructions to complete the given tasks. A user was asked to pair the devices four times in total; twice using the melody variant and twice using the MadLib (sentence) variant. Each variant was tested once with no attack assumption (where verification sequences matched) and once under attack simulation (verification sequences were forced to be different) in order to analyze users' ability to distinguish matching and different sequences. To reduce the learning effect on test results, the four tasks were presented to the user in random order. The transition between tasks were automated (four executions are started automatically one after the other) and the user actions were logged automatically by the testing framework [12]. After completing the tasks, each participant filled out a post-test questionnaire (see Figure B-1 in the Appendix) form and was given 5 minutes of free discussion time followed by a short interview.

Table 2: Summary of the logged data

Method	Completion Time (sec.)	Fatal Error Rate	Safe Error Rate
Melody (No Attack)	62.15	N/A	10%
Melody (Under Attack)	74.5	15%	N/A
Sentences (No Attack)	56.95	N/A	0%
Sentences (Under Attack)	80.5	5%*	N/A

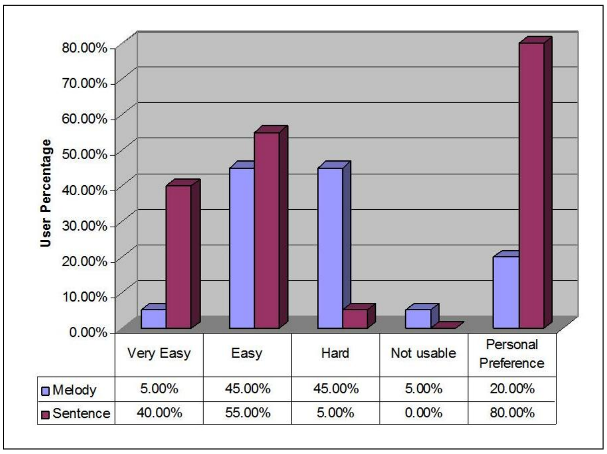
Results: We collected data in two ways: (1) by timing and logging user interaction, and (2) via questionnaires and structured interviewing.

Completion time was automatically logged by the software. The sentence-based variant was faster then the melody-based variant when the two values matched. Whereas, the melody-based variant was ahead when the two values differed. Albeit, average completion time hovered around 68 seconds for both methods, as shown in table 2. Although playing a melody takes less time than vocalizing a sentence, the users replayed melodies more to be able to decide whether they were same. When the sentences matched, participants re-played them 1.3 times on average, and 1.75 times when they didn't match. The average play count for melodies was 1.5 for matching and 1.8 for non-matching sequences.

In HAPADEP, if the user indicates (forces) a match in case of two different audio sequences, the protocol clearly cannot be assumed secure. Such an carelessness can allow the attacker to succeed in an impersonation or MiTM attack. Due to its grave effect on security, we call such errors as *fatal errors*. On the other hand, indicating no match for matching values does not introduce any security vulnerabilities but simply voids the current pairing session. We call such errors as *safe errors* due to their *benign* nature. As shown in Table 2, using melodies for verification caused higher *fatal* and *safe* error rates. Many participants stated that they would recognize different melodies better if they had a chance to execute the protocol few times and get their hearing "adjusted" before the tests. Subjects who tested the melody variant with non-matching sequences (before matching ones) complained about their initial tendency to tolerate the difference between melodies, since they did not know how much difference they should have expected. Those who tried matching melodies first also complained about the same issue but claimed that they tended to be alerted by slight differences in sequences, due to different quality speakers in the devices. We believe that this is a fundamental problem with the current melody variant since the security software cannot tolerate any insecure trial-and-error learning period.

On the other hand, comparing sentences resulted in acceptable error rates. There was only one subject who accidentally pressed the *same* button for different sentences causing a *fatal error* to be logged. However, the subject realized his mistake immediately and asked if there was a way to cancel. So, even in this case, the cause of the error was not the user's inability to recognize non-matching sentences but our poor GUI design which facilitated this kind of interaction. Security risks of this type of errors can be classified as being lower than those due to unnoticed errors, since the user is aware of the mistake and can thus take an immediate recovery action. However, security software should be free of such errors, since it is hard to accurately foresee the damage an attacker can cause even in few seconds.

In the post-test questionnaire, we solicited user opinions and preference about the tested methods. Participants found comparing sentences easier and more usable in general and preferred to play the sentences one after the other between devices. Comparing melodies got lower usability rankings from the majority of the participants, and we observed that it was usable only if both devices started to play the melody at the same time (more-or-less in stereo). Participants were more sensitive to background noise or distractive elements when they were comparing melodies and expressed this concern to us. As their personal choice, 80% preferred comparing sentences over melodies. Data gathered from post-test questionnaires is summarized in Figure 3.



	Very Easy	Easy	Hard	Not usable	Personal Preference
□ Melody	5.00%	45.00%	45.00%	5.00%	20.00%
■ Sentence	40.00%	55.00%	5.00%	0.00%	80.00%

Figure 3: Participant Opinion

After they filled the post-test questionnaire, we interviewed the participants about their experience with current pairing technologies and HAPADEP. We found out that 70% of the participants tried to setup a secure wireless 802.11x home network and 45% of subjects tried bluetooth pairing before. When we told them HAPADEP (in melody or sentence flavor depending on their choice) can be used as a replacement for those procedures, all people that had previous Wi-Fi pairing experience told they would prefer to use HAPADEP instead. 56% of the people with previous bluetooth pairing experience also said they would prefer HAPEDEP and 22% said they may prefer HAPADEP in certain secenarios but not always. Only four participants had tried infra-red communication; two of them said they could not get it to work and would prefer HAPADEP instead.

From our usability analysis, we conclude that the HAPADEP melody variant is not mature enough to provide both usability and security. In the rest of the paper, we assume a MadLib (sentence-based) variant.

7. SECURITY ANALYSIS

In HAPADEP, two devices establish either a unidirectional or bidirectional secure channel by exchanging their public keys over the audio channel. Assuming that the devices are not compromised, the cryptographic primitives and the public key schemes are assumed secure; an attacker can only perform Denial-of-Service (DoS) or Man-in-the-Middle (MiTM) or impersonation attacks.

To perform a DoS attack, the attacker can play loud audio and prevent the personal device from recording what the target device is playing. While the HAPADEP codec is quite robust against background noises, the attacker volume level might be so high that the sound played by the target device cannot be *heard* by the personal device. After recording, the integrity check at the personal device would fail and the transfer phase has to be repeated numerous times. However, such DoS attacks can be recognized (actually, heard) by the user. Other DoS attacks might involve the adversary using very low or very high (not audible to the human ear) frequencies. However, decoders can be easily tuned to filter out such frequencies. Given a robust and appropriately tuned decoder, the communication channel can be forced to always be human-perceptible, not allowing any such DoS attack to be unnoticed.

In an impersonation attack, the attacker's goal is to convey its public key to the personal device by impersonating the target. In the verification phase, however, both devices vocalize sentences that represent their respective views of the exchanged cryptographic material. Note that the target device would compute the sentence based on what it has sent and the personal device computes it based on what it has received. Assuming that the underlying hash function is second pre-image resistant (i.e., the probability of finding another input string that hashes into the same value is negligible), any impersonation attack would result in different sentences computed on, and vocalized by, the devices. In our usability tests, we observed that the users are quite capable of recognizing matches and mismatches in device-vocalized MadLib sentences, even in reasonably noisy and crowded environments. We also note that an active impersonation attack would involve a third (adversarial) device attempting to super-impose its sound over one or both legitimate devices. This can be easily heard by the user; thus we observe that HAPADEP provides a certain degree of real-time DoS and MiTM attack detection.

8. HAPADEP LIMITATIONS

The proposed HAPADEP technique clearly has some notable limitations which we now summarize.

- First and foremost, similar to Loud-and-Clear, HAPADEP is a poor choice for users who are hearing-impaired. Also, again like Loud-and-Clear, HAPADEP is unsuitable for noisy environments, such as factories, convention floors or stadiums.

- On a related note, HAPADEP needs sufficient proximity between devices. Our experiments were conducted with devices separated by 1-2 feet (30-60 cm). In noise-free settings, larger distances are possible; however, we do not expect to exceed 5-6 feet with commodity devices, such as PDAs or cellphones. We also note that many prior methods [6, 7, 9, 5, 8, 4, 11] have the same proximity limitation. In contrast, we expect that the Blinking-Lights [14] and the laser-based [10] techniques are somewhat better in this regard, allowing distances upwards of 20 feet.

- HAPADEP requires both devices to have audio in and out interfaces. Nonetheless, it is possible to trivially modify the protocol so that only one device would need audio-in (microphone) but both would need audio-out (speaker). We do not consider this to be a limitation *per se*, since speakers and microphones are routinely present on most modern devices, with very few exceptions (e.g., wireless access points).

9. DISCUSSION

While the only requirement for both devices are a speaker and a microphone, the user must be able to perform two (usually) simple tasks:

- Recognize which device is playing the MadLib sentence, among possibly several nearby devices
- Recognize whether two devices are indeed vocalizing the same sentence. (User can replay the sentences as much as she wants and can choose to play them simultaneously or sequentially).

Our usability tests indicated that most non-hearing-impaired adults are capable of performing both activities and so HAPADEP is an easy to use alternative to prior techniques.

In addition to security against impersonation and MiTM attacks, and unlike other proposals, HAPADEP offers users the ability to detect some on-going DoS attacks. Other techniques can do little against DoS attacks that aim to jam device interfaces used for the human-imperceptible communication. In HAPADEP, a user can detect the presence of extraneous noise and might even be able to identify its source.

The implementation of HAPADEP is straightforward and the source code is available online [2]. We implemented, and experimented with, two variants, based on melodies and sentences (MadLibs). Although the former performed somewhat poorer than the latter in our experiments, our original motivation for using melodies was two-fold:

- Melodies can be generated on-the-fly without storing any lookup dictionaries.
- Many devices, including those on the low end of the spectrum, are capable of playing chords, but not text-to-speech (TTS).

Storing lookup dictionaries for sentence generation would take up additional storage space of about 50 KB of ROM. TTS engines usually need better computation capabilities and more memory as well. (There are several embedded TTS engines with small footprints that work on almost any cell phone or PDA, but they still do not run on more constrained devices, e.g., bluetooth headsets). However, our usability study clearly indicates that the melody variant is not the best approach. However, we are currently evaluating possible improvements, such as using mixed instruments, different algorithms, forcing simultaneous play of the verification melodies, etc.

10. SUMMARY

This paper introduced HAPADEP – a new approach to secure device pairing. HAPADEP can be viewed as an extension of the previously proposed Loud-and-Clear technique [7] where **all** communication is conducted over the user-perceptible audio channel. HAPADEP is easy to implement and deploy, as our experience indicates. It also offers some built-in protection against DoS and MiTM attacks. The former, in particular, distinguishes it from prior solutions. In addition, HAPADEP doesn't require any common digital interface or initial communication setup thus representing the most usable and lowest-cost device pairing solution to-date.

Acknowledgements

We thank C. Lopes for the help with the audio codecs and N. Saxena for his comments on the early stages of this work.

11. REFERENCES

[1] Bouncy Castle Crypto APIs. http://www.bouncycastle.org/.

[2] HAPADEP website. http://sconce.ics.uci.edu/hapadep/.

[3] Adrian Perrig and Dawn Song. Hash visualization: A new technique to improve real-world security. In *Proceedings of the 1999 International Workshop on Cryptographic Techniques and E-Commerce (CrypTEC '99)*, pages 131–138, July 1999.

[4] Dirk Balfanz, D.K. Smetters, Paul Stewart, and H. Chi Wong. Talking to strangers: Authentication in ad-hoc wireless networks. In *Symposium on Network and Distributed Systems Security (NDSS '02)*, February 2002.

[5] L. M. Feeney, B. Ahlgren, and A. Westerlund. Demonstration abstract: Spontaneous networking for secure collaborative applications in an infrastructureless environment.: International conference on pervasive computing (pervasive 2002). 2002.

[6] Frank Stajano and Ross Anderson. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Security Protocols, 7th International Workshop*, 1999.

[7] M. T. Goodrich, M. Sirivianos, J. Solis, G. Tsudik, and E. Uzun. Loud and clear: Human-verifiable authentication based on audio. In *ICDCS '06: Proceedings of the 26th IEEE International Conference on Distributed Computing Systems*, 2006.

[8] L. E. Holmquist, F. Mattern, B. Schiele, P. Alahuhta, M. Beigl, and H.-W. Gellersen. Smart-its friends: A technique for users to easily establish connections between smart artefacts. In *UbiComp '01: Proceedings of the 3rd international conference on Ubiquitous Computing*, pages 116–122, London, UK, 2001. Springer-Verlag.

[9] Jonathan M. McCune, Adrian Perrig, Michael K. Reiter. Seeing-Is-Believing: Using Camera Phones for Human-Verifiable Authentication. In *2005 IEEE Symposium on Security and Privacy*, pages pp. 110–124, 2005.

[10] T. Kindberg and K. Zhang. Secure spontaneous device association. In A. K. Dey, A. Schmidt, and J. F.

McCarthy, editors, *Ubicomp*, volume 2864 of *Lecture Notes in Computer Science*, pages 124–131. Springer, 2003.

[11] T. Kindberg and K. Zhang. Validating and securing spontaneous associations between wireless devices. In C. Boyd and W. Mao, editors, *ISC*, volume 2851 of *Lecture Notes in Computer Science*, pages 44–53. Springer, 2003.

[12] K. Kostiainen and E. Uzun. Framework for comparative usability testing of distributed applications. http://sconce.ics.uci.edu/CUF/.

[13] C. Lopes. The *digital voices* project home page.

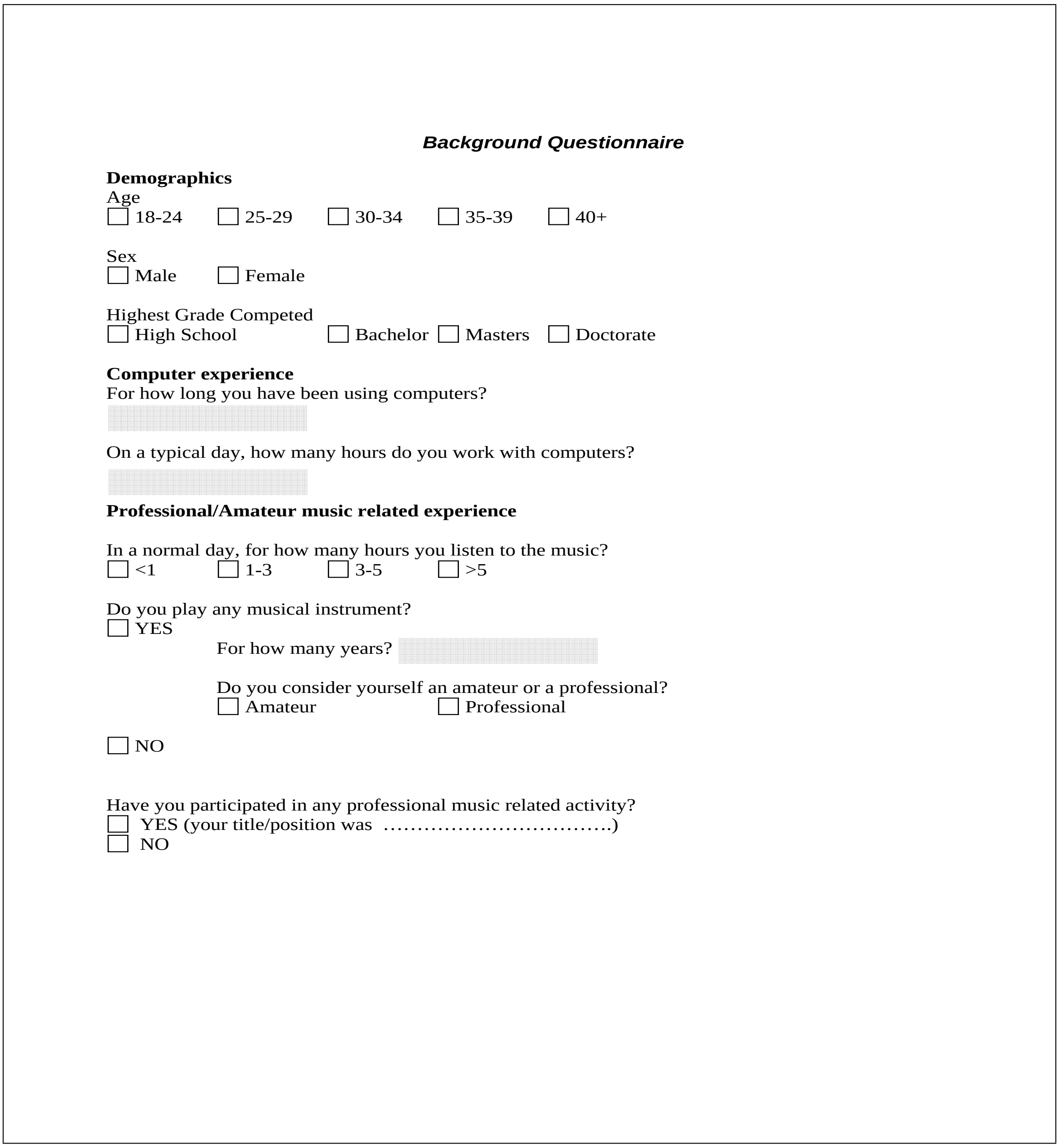
[14] Nitesh Saxena, Jan-Erik Ekberg, Kari Kostiainen, and N. Asokan. Secure Device Pairing based on a Visual Channel. In *2006 IEEE Symposium on Security and Privacy*, 2006.

[15] P. C. v. O. W. Diffie and M. J. Wiener. Authentication and authenticated key exchanges. *Designs, Codes and Cryptography*, 2, 1992.

[16] W. Diffie and M. E. Hellman. New directions in cryptography. In *IEEE Transactions on Information Theory*, pages IT–22(6):644–654, November 1976.

APPENDIX

A. BACKGROUND QUESTIONNAIRE



Background Questionnaire

Demographics

Age

□ 18-24 □ 25-29 □ 30-34 □ 35-39 □ 40+

Sex

□ Male □ Female

Highest Grade Competed

□ High School □ Bachelor □ Masters □ Doctorate

Computer experience

For how long you have been using computers?

On a typical day, how many hours do you work with computers?

Professional/Amateur music related experience

In a normal day, for how many hours you listen to the music?

□ <1 □ 1-3 □ 3-5 □ >5

Do you play any musical instrument?

□ YES

For how many years?

Do you consider yourself an amateur or a professional?

□ Amateur □ Professional

□ NO

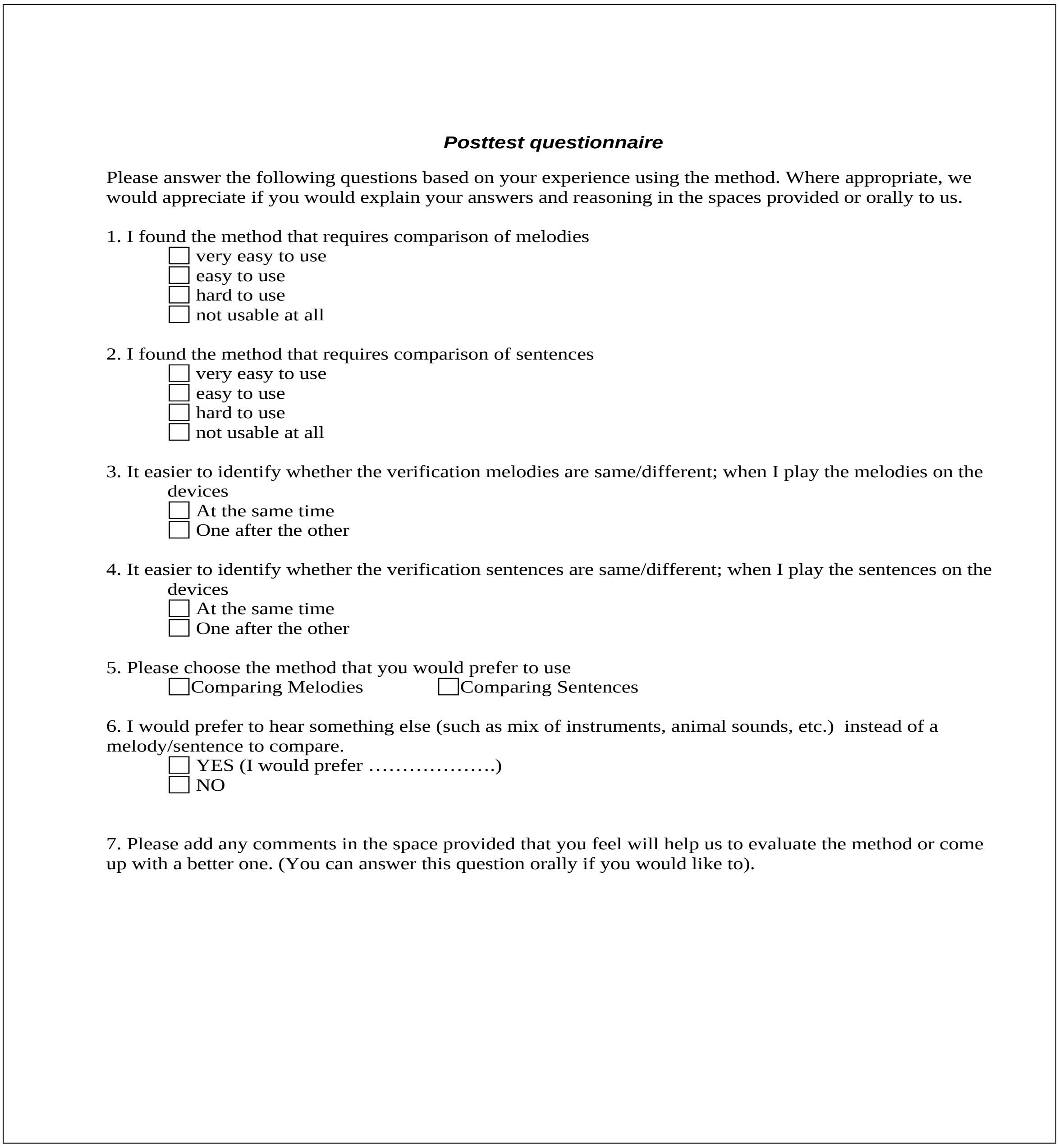
Have you participated in any professional music related activity?

□ YES (your title/position was ……………………………..)

□ NO

Figure A-1: The background questionnaire

B. POST-TEST QUESTIONNAIRE



Posttest questionnaire

Please answer the following questions based on your experience using the method. Where appropriate, we would appreciate if you would explain your answers and reasoning in the spaces provided or orally to us.

1. I found the method that requires comparison of melodies

□ very easy to use
□ easy to use
□ hard to use
□ not usable at all

2. I found the method that requires comparison of sentences

□ very easy to use
□ easy to use
□ hard to use
□ not usable at all

3. It easier to identify whether the verification melodies are same/different; when I play the melodies on the devices

□ At the same time
□ One after the other

4. It easier to identify whether the verification sentences are same/different; when I play the sentences on the devices

□ At the same time
□ One after the other

5. Please choose the method that you would prefer to use

□Comparing Melodies □Comparing Sentences

6. I would prefer to hear something else (such as mix of instruments, animal sounds, etc.) instead of a melody/sentence to compare.

□ YES (I would prefer ……………….)
□ NO

7. Please add any comments in the space provided that you feel will help us to evaluate the method or come up with a better one. (You can answer this question orally if you would like to).

Figure B-1: The post-test questionnaire