Evidently secure device associations

Tim Kindberg, Kan Zhang, Seung Hyun Im[1]
Consumer Applications and Systems Laboratory
HP Laboratories Bristol
HPL-2005-40
March 1, 2005*

mobile computing, security, spontaneous associations

A key problem in mobile and ubiquitous computing is that of setting up an association between a pair of devices so that they may communicate securely over a wireless network. It is particularly important to be able to solve this problem for spontaneous associations, which must not depend on pre-existing security values such as certificates, and when the only means of identifying the target device is physical. This paper contributes protocols for validating secure spontaneous associations. The protocols complement existing unauthenticated key-exchange protocols and work over widely used wireless technologies. They improve on previous work by eliminating specialised hardware. We present the protocols and discuss their advantages and limitations.

* Internal Accession Date Only
[1] Dept. of Software and Information Systems, University of North Carolina at Charlotte, 9201 University City Blvd., Charlotte, NC 28223 USA
Approved for External Publication

Evidently secure device associations

Tim Kindberg
Hewlett-Packard Laboratories
Filton Road, Stoke Gifford
Bristol BS34 8QZ, UK
timothy@hpl.hp.com

Kan Zhang, Seung Hyun Im[1]
Hewlett-Packard Laboratories
1501 Page Mill Road
Palo Alto, CA 94304, USA
zhangkan@sbcglobal.net, sim@uncc.edu

Abstract. A key problem in mobile and ubiquitous computing is that of setting up an association between a pair of devices so that they may communicate securely over a wireless network. It is particularly important to be able to solve this problem for spontaneous associations, which must not depend on preexisting security values such as certificates, and when the only means of identifying the target device is physical. This paper contributes protocols for validating secure spontaneous associations. The protocols complement existing unauthenticated key-exchange protocols and work over widely used wireless technologies. They improve on previous work by eliminating specialised hardware. We present the protocols and discuss their advantages and limitations.

1 Introduction

This paper describes techniques for securely associating devices in ubiquitous computing ("ubicomp") environments. In particular, an important characteristic of ubicomp is *spontaneous* device association. Humans carry personal devices with them as they move from one ubicomp environment to another. They may also acquire devices that are meant for use by visitors to the environment – e.g. smart whiteboards and pens. Often, devices become really useful only when they work together: e.g. Johnny borrows a smart pen from David and records digital ink onto the personal storage device on his belt; Mary sends a picture from her camera-phone to Robert's digital picture frame while visiting his house; two teenagers who encounter one another in a shopping mall play a wireless game together with their game-phones or PDAs; two colleagues meet at a conference and transfer a document from one's PDA to the other's laptop.

Spontaneous interactions occur as part of "everyday computing" and are potentially of great value to users. They happen when the time is right given the current activity and circumstances; they require little effort to set up or carry out. However, such interactions are not yet a reality. One problem is security. Only a modest level of security is consistent with everyday computing – high-security applications tend not to be carried out spontaneously. Nonetheless, security is important and achieving it is non-trivial. Referring back to the examples just given, spontaneous interactions occur in untrustworthy surroundings. The physical surroundings themselves may be untrustworthy – the conference where colleagues exchange a document is attended by business rivals. But even if the surroundings are familiar, they are nonetheless typically within wireless reach from unknown parties – whoever lives near Robert when Mary transfers her image; other people with wireless devices in the shopping mall.

In general, a human involved in a spontaneous interaction has established trust (rightly or wrongly) in a particular target device; but they are surrounded by users and devices they do not trust. Our research question is: how can we best enable the human to make secure associations between trusted target devices in spontaneous circumstances?

Cryptographic techniques will not suffice by themselves. Those can achieve only the secure binding of a cryptographic key to certain electronic identifiers (network addresses, device names). But that cannot satisfy the user of what is actually required in spontaneous circumstances: that *this* physical device is securely associated to *that* physical device.

A principle that has now been well established for securing spontaneous associations, first by Stajano and Anderson [14] and subsequently by Kindberg et al [8, 9] and Balfanz et al [1], is that of bootstrapping security properties from *physically constrained channels*. A physically constrained channel is such that only entities in a

[1] Author's present address: Department of Software and Information systems, University of North Carolina at Charlotte, 9201 University City Blvd., Charlotte, NC 28223, USA.

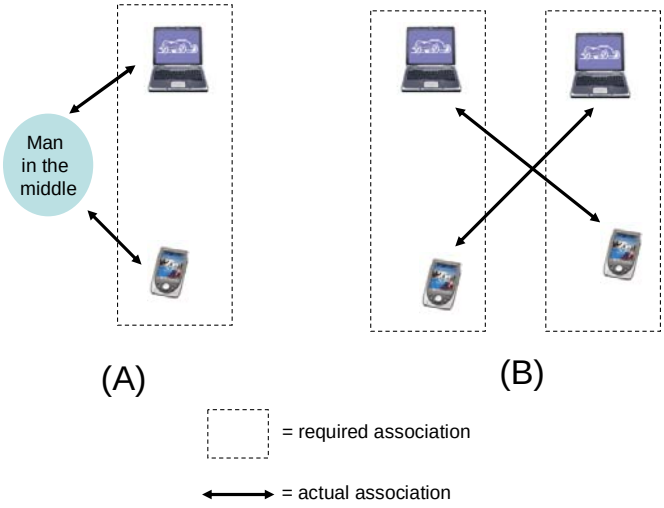

Fig. 1. Maliciously (A) and accidentally (B) incorrect associations

certain physical context may transmit a message over the channel – or vice versa, that only entities in constrained physical circumstances may receive messages over the channel. Examples of such channels are:

direct electrical contact [14], the human body in contact with devices [5, 11], infrared beacons [1, 4], combinations of ultrasound and radio propagation [8], laser beams [9], 60 GHz radio, and audio [1]. The radiative channels are collimated and/or severely attenuated by common building materials.

Physically constrained channels provide a basis for authentication that is *a posteriori* and does not rely on *a priori* knowledge of names or keys. That precisely fits the requirements of spontaneous circumstances. For example, the two teenagers who have just met in the mall in general have no *a priori* knowledge of one another's device names (which anyway may easily be spoofed or confused with other device names), and no cryptographic material in common. But by, for example, shining their laser beams at one another's devices, they may exchange secrets or other authenticating values with a good level of protection against attack. Such a physical mechanism has good evidential value for the human: the human sees the laser's red dot on the other device and is assured of which device they are associating to – as opposed to wondering whether, for example, what they have network-discovered under the name of "Bob's N-gage" is actually the device in the hands of that person over there.

But physically constrained channels require special hardware – even infrared transceivers are increasingly omitted from products – and are often hard to engineer. None of the technologies provides absolute guarantees of physical constraint, because of refraction and reflection and the existence of sensitive receivers that can detect faint signals. Their guarantees are often good enough for everyday computing, but it would be preferable to eliminate such hardware issues.

In this paper, we contribute techniques for securing spontaneous associations that exploit only physical indicators with which personal devices are commonly equipped: LEDs and displays, audio output and vibrators. We describe a system model in which we divide the problem of securing associations so that we need solve only a sub-problem we call *physical validation*, and can rely on well-established techniques for solving the remainder. We present and analyse two protocols for physical validation. We then relate the contribution to existing research and conclude with a summary and discussion of outstanding problems.

2 System Model

In the scenarios we are considering there are two devices, A and B, in line of sight with one another so that any human involved in the association can see both devices and any other user involved. Either A and B are both "personal" devices, each in the possession of a user, Alice and Bob; or one is Alice's personal device and the other is an "infrastructure" device, such as a digital picture frame. The two devices communicate via a wireless network such as 802.11b or Bluetooth.

The goal is to form a secure association between the two devices, and to do so spontaneously. A secure association has taken place when each of the two devices possesses the other's network address and they share a secret key to encrypt their communications. For the association to be spontaneous we require that:

- it is predicated on minimal *a priori* values; for example, it is unrealistic to assume that one device already knows the public key of the other device;
- it is quick and convenient for the user to set up, including capturing *a posteriori* values (i.e. ones captured at the time of making the association) and running through whatever other protocols are required.

Conceptually, we can break down the formation of a secure association into three steps:

1. Exchange network addresses
2. Exchange a secret key without requiring authentication
3. Physically validate the association: verify that the physical entities that exchanged keys are in fact the required devices.

Spoofing attacks are possible in the first two steps. The strongest attack is the man in the middle attack shown in Figure 1(A), in which a malicious entity exchanges keys with each device and thus spoofs each device in relation to the other. Even in the absence of a malicious party, the situation in Figure 1(B) may obtain: two pairs of devices were to be securely associated, but in fact a member of each pair has mistakenly associated with one of the other pair.

We do not require the secret key exchange to be authenticated, but instead apply the separate step of physical validation. We thus allow key exchange to go ahead – possibly incorrectly – but verify it at the validation step. As we explained in the introduction, this is not the only way of setting up a secure spontaneous association – we could have applied physical authentication in the second step using a physically constrained channel. But the validation approach allows us to take advantage of existing protocols for steps 1 and 2.

For example, the SWAP-CA specification [13] for wireless networking in a home environment introduced what is commonly referred to as the two-button protocol for exchanging network addresses, which Iwasaki et al. [7] investigated more recently. Users simultaneously trigger two devices into an 'association' mode, usually by pressing a button on each device. Each device listens for association messages and correlates them in time to deduce the address of the other device.

Once the devices have exchanged network addresses, they may use, for example, the Diffie-Hellman protocol [3] to exchange fresh secret keys.

To physically validate an association is to securely correlate the key that the devices have exchanged with the physical devices themselves. We therefore require some physical phenomena associated with the devices, over which we have electronic control. Thus, we assume that devices have at least one integral physical indicator. For example, a PDA or laptop has a display, audio output, and often an LED. A mobile phone has a small display, audio output, and a vibrator. We shall show that in some cases, an indicator needs to be perceptible to the human possessing the *other* device, in line of sight up to some reasonable distance; however, there are cases where a device's indicator needs to be perceptible only to the human possessing it.

3 Protocols for Validating Associations

After key-exchange, device A possesses key KA, and device B possesses key KB. To physically validate the association is to verify that KA = KB; there is no man in the middle; and the physical possessors of those keys are indeed the intended devices. To satisfy the human(s) of the key-possessors' physical identities, our protocol provides evidence of the physical association. The human(s) might have connected a secure cable between the two devices. We require similar evidence of a secure association, but over a wireless network rather than an inconvenient cable.

In this section we present two protocols for physically validating the association. The protocols make different assumptions and provide different types of evidence for physical validation. We compare them as we go along.

3.1 Comparing Keys

This protocol works by enabling the human(s) to compare keys directly, as Stajano and Anderson first suggested [14]. It exploits the "unique-key" property of certain key exchange protocols: if a man in the middle manages to exchange keys with each device under that protocol, it is extremely unlikely that those keys are the same. The Diffie-Hellman key exchange protocol has the unique-key property, but any such key exchange protocol will do. By establishing that the two devices possess the same key, we thus verify (1) that those two physical devices are associated, and (2) that there cannot be a man in the middle.

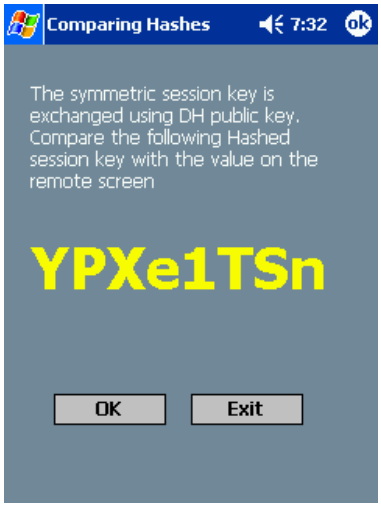

Fig. 2. Base-64 representation of hash of key

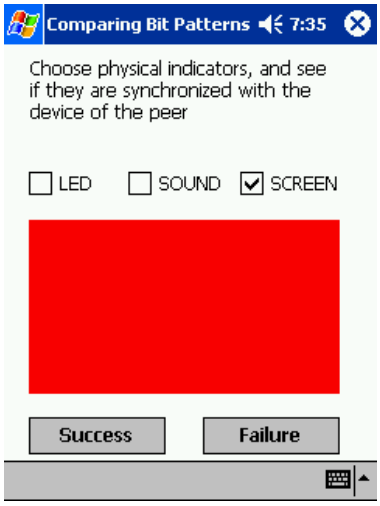

Fig. 3. Comparing keys by rendering their bits. Screen colour is black or red according to whether the bit is 0 or 1

We enable the human or humans involved to compare keys in the following way. First, to keep the keys secret, each device applies a secure hash code H (such as SHA1 or MD5) to form H(KA), H(KB) respectively. Then each renders the first few bits (typically about 48 bits should suffice but the user may opt for more or even less) through its physical indicators. We implemented the following types of indication.

Textual representation. In this technique, the devices display a textual encoding (e.g. base-64) of the key. We explored the technique by adapting an existing implementation of a walk-up kiosk for downloading digital content. A user with a PDA connects to the kiosk via Bluetooth to pay for and download a movie clip or other media – an operation that clearly requires a secure spontaneous association between the personal device and the kiosk. The kiosk has a large display, and when the PDA exchanges a key (supposedly) with it, the kiosk puts up a base-64 encoding of an MD5 hash of the key, and the PDA does likewise (Figure 2). The user compares up to 8 characters of the hashes to verify that the personal device and the kiosk have the same key. If and when the human is satisfied, she presses a key on her personal device to initiate the transaction.

To handle verification requests from multiple concurrent clients to the kiosk, the kiosk has six display sections for hashes, each printed in different colours. Each encoded hash remains on the screen for 20 seconds then disappears. In addition to 64-base alphanumeric symbols for encoding, we considered using various methods, such as using mathematical symbols, special characters, or icons of different faces. First indications are that alphanumeric symbols best fit users' perception capabilities. However, further user studies regarding the encoding method remain to be conducted.

With regard to the performance, we implemented the protocol on one laptop computer with 1.2GHz Pentium III Mobile CPU running Windows XP for the server (public display device) and two HP iPAQ 5450 PDAs with 400MHz PXA CPU and built-in 802.11b card running Windows Pocket PC 2002 for the personal devices. Most of the code was written in C# using Visual Studio .NET 2003 with a few exceptions for device-specific components which were written in C++. On the client side, the initial start up time was approximately 10 seconds due to the generation of the Diffie-Hellman key, but the programming of this function was not optimised. The computation time for encoding and hashing did not significantly affect performance.

Multimedia representation. Textual comparison can be relatively straightforward for a human with a personal device accessing an infrastructure device with a relatively large display. But what of associations between, say, two PDAs or two phones? It is awkward to read text on two such small displays. Therefore we exploit those devices' other physical indicators.

In this variant, each device renders the bit-pattern of its hashed key in a synchronised way. In our implementation, we again experimented with iPAQs, each of which applies an MD5 hash to its key after a Diffie-Hellman key exchange. Users choose up to three different physical indicators to represent '0' and '1' in the bit pattern: an LED, sound (a beep), and the screen (a block drawn in different colours). The pattern repeats when all the bits are used up.

We found that the small size of the iPAQ LED limits the readability of the bit patterns, especially where users are outside the building in daylight. In some cases, users may be disturbed by surrounding noise in crowded areas if only the sound is used to indicate the bit pattern. Our experiment shows that the combination of sound

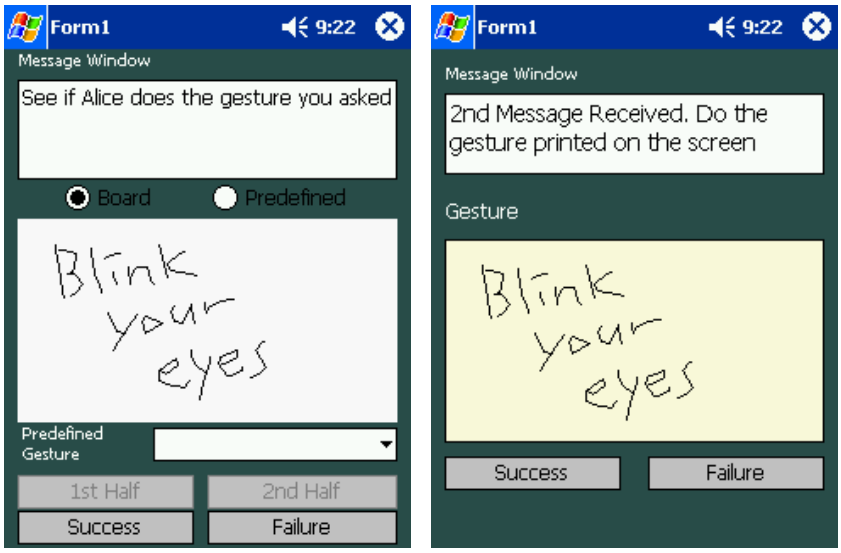

Fig. 4. Verifying secure ad hoc association using the physical interlock protocol

and screen works well both inside and outside the building even in crowed environments. One user listens to the beep while the other user flashes the screen. Figure 3 is a snapshot of the application during execution.

Because we compare the hash value of the keys bit by bit, the synchronization of the bit stream is important. When the computing capability of the two devices running this protocol is not identical, the bit patterns become unsynchronised a few seconds after the two devices start the protocol. To resolve this problem, one device becomes the "synchroniser" and the other becomes the "synchronised" based on the magnitude of the IP addresses, and the synchroniser transmits a "sync message" including the bit position in the stream to the synchronised device just before it renders each bit. We did not see any noticeable delays in bit indication with this method.

The multimedia protocols are not yet wholly explored and deserve further experimentation. Other media, such as a phone or PDA's vibrator or (ring)tone synthesisers, could also be used. We chose 0.5 second per bit with reasonable results but it is not clear how long the rendering of each bit should last; there is a trade-off between the user's ability to discern the sameness or difference of the rendered bit patterns, and the length of time taken by the whole procedure. Another option we have not yet tried is for one device to indicate the opposite of the other device; if the bit patterns are the same, this would give the effect of smooth movement of sound or colour backwards and forwards between the two.

3.2 Physical Interlock

In this protocol, once again we validate an association using the unique-key property, by verifying the two devices share the same key and thus proving the absence of a man in the middle. However, in this protocol we show indirectly that the two devices possess the same key. The method requires only a few bits of out-of-band information to be communicated between the parties, which makes it possible to signal using only a few human gestures or physical indicators on the device. Compared to the key-comparison protocol, this protocol has a shorter running time, but it does require a clearly differentiable set of physical indications. We derive our method from the interlock protocol [12].

As before, suppose after running a key exchange protocol, devices A and B end up with secret keys KA and KB, respectively, and we want to verify that KA = KB. Device A first displays a list of *N* pre-defined indication commands, such as "raise your left hand", "touch your nose" or "play the Flight of the Bumblebee ringtone", and asks its user Alice to randomly pick one. (The list of available device indications can be negotiated in a first step; it does not need to be secret.) Device A then encrypts the chosen command using KA and sends the first half of the cipher text to device B. Upon receiving the first half of the cipher text, device B notifies its user Bob and Bob indicates receipt of the first half by a conventional gesture such as raising his right hand. When Alice sees Bob raise his right hand, she instructs device A to send the second half of the cipher text to device B. When device B receives the second half of the cipher text, it puts the two halves together and tries to decrypt the entire cipher text using key KB. If the decryption fails, there may be a man in the middle. Device B indicates that the verification failed and Bob aborts the protocol. Otherwise, device B outputs the decrypted indication command from Alice, and Bob follows the command to give an indication such as playing the chosen ringtone. If Alice

perceives the same indication she picked in the beginning, the verification is successful. Otherwise, the verification failed. Alice and Bob can switch roles and run the protocol again so that Bob can check if the verification is successful.

When the verification is successful, the probability that KA ≠ KB, (i.e., that Alice and Bob are fooled by a man in the middle), is $1/N$, which is the probability that the man in the middle correctly guessed Alice's command. This probability can be made arbitrarily small by running the protocol multiple times. After running m times, the probability becomes N^{-m}. Alice and Bob can run the protocol as many times as they want till they are both convinced that there isn't a man in the middle and KA = KB.

A variation on that method is where each user proposes an indication to the other. Each device A and B encrypts its respective indication, transmits the first half of the cipher text and waits. When it receives the first half of the other's cipher text, it transmits the second half of its cipher text and awaits the second half of the other's cipher text. Each device puts the two halves together and tries to decrypt the entire cipher text using its key. If the decryption fails, the device aborts the protocol and alerts the user. Otherwise, the device displays the other user's selected indication. Each user then makes the displayed indication to the other. When Alice and Bob perceive the selected indication at their counterpart, they know that the verification is successful. Otherwise, the verification failed. Alice and Bob can repeat this protocol as many times as they like to increase the likelihood that no man in the middle is present. This variant of the protocol has the advantage of eliminating the need for Alice to wait until Bob gestures that he has received the first half of the message.

As noted in [12], instead of transmitting the two halves of the cipher text as above, other two-part methods could be used as long as the transmission of the first part effectively commits the sender to the final clear text although the clear text cannot be computed without the use of the second half as well. For example, the first half could be a one-way hash of the cipher text and the second part could be the cipher text itself.

We implemented the first variant of the physical interlock protocol, using iPAQs connected on an 802.11b wireless network. Based on the magnitude of the random IP address assigned, one becomes the sender and the other becomes the receiver. The sender can specify the required indication either by selecting it from a predefined list or by writing it on the screen (see Figure 4). The software encrypts the indication using 56-bit DES without a noticeable delay.

Various human factors come into play with this protocol. First, users may not consider it to be discreet enough in certain circumstances. Second, the protocol leaves it to the user to select the required indication. That puts the user (who knows the context) in control; but on the other hand users may prove poor at choosing indications randomly. The protocol would be more secure if the device chose the required indications, and users simply exported a list of indications they were prepared to make.

3.3 The Harmony Protocol

This "harmony" protocol requires users to compare multimedia streams at the two devices, like the multimedia representation protocol for comparing keys. However, keys are essentially random bit strings, and so their multimedia representation tends to be meaningless to humans – a fact which does not help comparison. By contrast, this protocol enables us to play out two "harmonised" streams: for example, a bass part at one device, and a piano part at another; or a tune at one device and a pattern of changing colour synchronised with it at another. A special case is where the two streams are identical.

As with the previous protocols, we assume the unique-key property. But the harmony protocol has an additional constraint. The protocol is designed for wireless technologies where it is possible for any devices with the same set of pre-configured parameters to tune to a common channel and receive all packets sent on this channel without on-the-spot negotiation. Examples of these are IEEE 802.11b, IEEE 802.11a and IEEE 802.11g. Conversely, wireless technologies using frequency hopping, such as Bluetooth, HomeRF and IEEE 802.11-Frequency-Hopping, do not meet this criterion.

We shall assume that the two devices communicate via 802.11b, in *ad hoc* mode, using a predefined channel and SSID. One user, say Bob, agrees to be the "receiver"; and Alice to be the "sender". Bob begins by listening for packets with the network interface in "monitor" mode. In that mode, his device can receive all packets on the network, regardless of the 802.11 SSID or BSSID. Even if an attacker succeeds in placing Alice and Bob in separate cells ("independent basic service sets"), Bob can still receive Alice's packets if she is in radio range, regardless of their address. Unfortunately, however, Bob cannot transmit packets when the network interface is in monitor mode.

Device A renders a "source" multimedia stream such as a tune. While the stream plays out, it consults a corresponding "harmonised" stream. When an event fires in the harmonised stream, Device A does not render it but it sends a message to device B to render it.

Thus device A emits a sequence of "indicator packets" I_i at times T_i, to what A believes to be the network address of device B, at a port number designated for this protocol. Device A broadcasts the packets (because B cannot transmit in monitor mode, and so cannot send 802.11 ACK packets).

Each indicator packet I_i (i = 0, 1, ... N) is constructed as follows:

$I_i = \{N_i, command\}$KA

– where N_i is a nonce and *command* is a command such as "switch light on", "switch light off" or "play trumpet note F#"; and $\{M\}$K denotes encryption of the message M by key K.

There are important constraints on the command sequence. First, each successive command must produce a state s' distinct from the previous state s. Second, each state s must have a minimal perceptible duration $t_{min}(s, s')$ and intensity (e.g. volume, brightness) before the subsequent state. While the perceptibility of a given state is in general a function of the ensuing state, it may be approximated as a function only of the medium in which the states are rendered; for example, there is a minimal perceptible note duration and volume in music, and a minimal time for a display to produce a perceptible change.

On receipt of a packet at the port designated for this protocol, device B decrypts the message with the expected key that it believes it shares with A. If what it recovers contains a recognizable command, B executes it; but only if it obeys the constraints outlined in the previous paragraph: (a) the command must produce a state s' different from the previous state s, and (b) the state s lasted for at least a minimum time $t_{min}(s, s')$. If a packet arrives early (the previous state has not lasted long enough), B puts the new command on a 'hold-back' queue. When the current state has endured for the minimum time allowed, B next executes the latest command on the hold-back queue that will produce a different state, if such exists, maintaining any commands that arrived subsequently on the hold-back queue. It discards any messages on the hold-back queue not covered by the foregoing.

The user(s) compares the streams of indicators on the devices. Device B's stream should be the stream expected by Alice, and it should be recognisably in harmony with her device's stream.

If device B fails to render a stream of indicators or if the streams of indicators are not individually recognizable or harmonious, then the user can conclude that they have failed to associate securely and may try another round of the association protocol (with a fresh key). Since packet delivery is unreliable, device B's stream might be missing some parts. Verification may thus fail, but that would be a conservative result (Alice and Bob may have falsely suspected a man in the middle when there is none). Similarly, a concurrent run of the protocol might play the same stream at exactly the same time, leading the protocol to abort; but that is unlikely.

If the stream comparison succeeds, then nonetheless there might a man in the middle (or a set of colluding men in the middle) who shares a key KA with A, and a key KB with B, and which relays a corresponding packet to B as soon as it has decrypted an indicator packet from A. If the man in the middle processes packets quickly enough, users will not notice the difference.

To detect the man in the middle, device B looks for evidence of packet relaying. It records data about all the indicator packets it cannot decrypt – that is, packets destined for the port designated by this protocol. The packet data it records is the time of arrival and the source MAC address. For each indicator packet J_j (j = 0, 1, ...) arriving at time t_j whose command it executed, B examines the source addresses of all recorded indicator packets in a "just-before" time interval $[t_j - \delta, t_j]$, where δ is an experimentally determined parameter of this protocol.

Device B signals to the user a "suspected failure to associate" if there is a significant number Sig(N) of indicator packets in the just-before intervals for the J_j (j = 0, 1, ...), whose source MAC address is the same. For that would suggest that a man in the middle is present: A issued the just-before packets, and the man in the middle decrypted them, encrypted them for B and sent them on to B. The value of Sig(N) is again a heuristically chosen parameter.

An attacker might try to send more commands than A sent, in the hope of reducing the number of just-before packets below the threshold Sig(N). However, the constraints on command execution ensure that all such additional packets will produce a perceptible effect. "Extra" packets would thus need to be harmonious packets.

Care is required in the choice of harmonised streams. First, there must be a sufficient rate of events in each stream for the term "harmony" to be meaningful. Second, it should not be possible for an attacker to fool Alice by having device B play the wrong stream, but nonetheless one that is in harmony with Alice's stream. To help Alice spot that, she can set her own device to play or omit the harmonised stream together with the source stream. Moreover, the minimum duration values $t_{min}(s, s')$ should be quite large to prevent the attacker from inserting "filler" commands that are hard to spot. We suggest that the harmony should thus be something like a "walking bassline" in music, or the equivalent on a display.

The harmony protocol seems promising in principle but in practice there are significant implementation issues. First, its success depends crucially on the choice of parameters Sig(N), δ and $t_{min}(s, s')$, which in turn depend on network characteristics and the parameters that determine how well humans can discriminate between indication streams. Second, it may be that only certain types of "harmonisation" lead to successful comparisons; for example, "harmonised" streams of the same medium may need to be identical for users to compare them

satisfactorily. Third, an attacker with sophisticated hardware could in principle read Alice's packets but also jam them, so that Alice's packets become invisible to Bob. It remains for us to investigate these issues.

4 Related Work

Bluetooth provides a model for secure association that is spontaneous in that the association can be set up without prior agreement. But the method is awkward: it requires response to a challenge that must be entered at both devices and communicated from one human to another.

Stajano and Anderson [14] originally suggested a less laborious solution to the secure spontaneous association problem in the form of the "resurrecting duckling" protocol. That protocol exchanges a key over an authenticating 'physical' channel; they suggested direct electrical contact between the devices. Others have gone one step further in using human touch as a way of associating devices [5, 11], although this does not seem to have been used to secure the associations.

Various other authors have investigated the idea of authenticating key exchange more generally using physical properties. Anon et al employed a variety of longer-range "physically constrained" channels [10] to achieve the transfer of data in such a way that the user can tell with which physical device the data originated, without having to physically touch that device. Those channels include infrared beacons, lasers [9], and combinations of radio frequency and ultrasound signals [8], all of which are attenuated by walls and can be directed with some degree of accuracy. Balfanz et al [1] pursued a similar idea, using what they termed "location-limited" channels, including infrared and audio signals. They observed that it can be preferable to send authenticating material – which does not have to be secret – on the constrained/limited channel, and use that to verify messages sent over the wireless network.

The requirement to achieve secure associations spontaneously eliminates some more conventional solutions. Virtual Private Network connections (often used over 802.11) require pre-registration and login. A secure service discovery system [2] relies on certificates and, for verification of those certificates, either a trusted third party or a solution for secure spontaneous key distribution such as we have suggested. Anyway, as we pointed out above, cryptographic authentication may satisfy a user that she is communicating securely with a trusted logical entity, but not that she is communicating with a particular device. The distinction matters: a different device controlled by the same authority may be exposed in some way, even if the device itself is trusted.

There have been some interesting proposals for spontaneously associating devices without security. In Section 2 we mentioned the SWAP-CA specification [13] and related work by Iwasaki et al. [7] in which users press buttons at the same time to swap network addresses. The Smart-Its project [6] introduced a method to establish an association between two handheld devices by holding them together and shaking them.

Finally, there is considerable literature on the psycho-physical characteristics that determine how well humans can tell whether two streams of indicators are synchronised. A seminal piece of research was Wertheimer's experiments on synchronised sparks that led to the field of Gestalt psychology [15]. Wertheimer measured the relationships between the timing of sparks and the observer's perception of synchronization or movement. However, we have not yet discovered work on users' abilities to make multimedia correlations.

5 Discussion

In summary, this paper has tackled the problem of secure spontaneous device association through the separable problem of validating an association, and has contributed protocols for achieving validation. The protocols complement existing unauthenticated key-exchange protocols and work over widely used wireless technologies. They improve on previous work by eliminating specialised hardware.

Our protocols are designed for everyday use. We provide a degree of security that is consonant with "spur of the moment" interactions such as we have described. For example, someone at a conference could reasonably transfer a sensitive document to a colleague's PDA using our protocols; a teenager playing a wireless game could reasonably use them to associate to a peer in a shopping mall, ensuring they know who they are playing with. However, we have not addressed denial of service attacks.

We implemented and tested the protocols, showing how we dealt with some of the issues that arose. However, we have yet to perform user studies. All of our protocols seem to require significant user attention. Our next research question is: does this degree of human involvement lead to desirable (protocol-compliant) behaviours, and retain the spontaneous quality we require?

References

1. Balfanz, D., Smetters, D.K., Stewart, P., and Wong, H.C. Talking to strangers: authentication in ad-hoc wireless networks. Network and Distributed System Security Symposium; San Diego; CA; USA, February 2002.
2. Czerwinski, S., Zhao, B., Hodes, T., Joseph, A., and Katz, R. An Architecture for a Secure Service Discovery Service. Proc. fifth annual ACM/IEEE international conference on Mobile computing and networking, 24-35, 1999.
3. Diffie, W. and Hellman, M. "New Directions in Cryptography", IEEE Transactions on Information Theory, v. IT-22, n. 6, Nov 1976.
4. Feeney, L., Ahlgren, B., and Westerlund, A. Spontaneous Networking: An Application-oriented Approach to Ad Hoc Networking. IEEE Communications Magazine vol. 39, no. 6, (2001).
5. Fukomoto, M., Shinagawa, M., and Sugimura, T. A Broad-band Intrabody Communication System with Electro-Optic Probe. In proceedings 1st International Conference on Appliance Design, Bristol, UK, May 2003.
6. Holmquist, L.E., Mattern, F., Schiele, B., Alahuhta, P., Beigl, M., and Gellersen, H.W. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. Proc. Ubicomp 2001, Springer-Verlag LNCS 2201, pp. 273-291, 2001.
7. Iwasaki, Y., Kawaguchi, N., and Inagaki, Y. Touch-and-Connect: A Connection Request Framework for Ad-hoc Networks and the Pervasive Computing Environment. In Proc. IEEE PerCom (2003), 20-29.
8. Kindberg, T., and Zhang, K. Validating and Securing Spontaneous Associations between Wireless Devices. Proc. 6th Information Security Conference (ISC'03), October 2003.
9. Kindberg, T., and Zhang, K. Secure Spontaneous Device Association. Proc. UbiComp 2003, Seattle, USA, October 2003.
10. Kindberg, T., Zhang, K., and Shankar, N. Context authentication using constrained channels. Proc. 4th IEEE Workshop on Mobile Computing Systems & Applications (WMCSA), June 2002.
11. Partridge, K., Newman, S., and Borriello, G. Facile: A Framework for Attention-Correlated Local Communication. WMCSA, October 2003.
12. Rivest,R., and Shamir, A. How to Expose an Eavesdropper, Communications of the ACM, Vol. 27, No. 4, April 1984.
13. Shared Wireless Access Protocol (Cordless Access) Specification (SWAP-CA), Revision 1.0, The HomeRF Technical Committee, 17 December 1998.
14. Stajano, F., and Anderson, R. The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks. In B. Christianson, B. Crispo and M. Roe (Eds.) Security Protocols. 7th International Workshop Proceedings, Lecture Notes in Computer Science, Springer-Verlag, 1999.
15. Wertheimer, M. Experimentelle Studien über das Sehen von Bewegung, Zeitschrift für Psychologie 61, p.161, 1912.