Shake Them Up!

A movement-based pairing protocol for CPU-constrained devices

Claude Castelluccia
INRIA and University of California, Irvine
claude.castelluccia@inria.fr

Pars Mutaf
INRIA
pars.mutaf@inria.fr

Abstract

This paper presents a new pairing protocol that allows two CPU-constrained wireless devices Alice and Bob to establish a shared secret at a very low cost. To our knowledge, this is the first software pairing scheme that does not rely on expensive public-key cryptography, out-of-band channels (such as a keyboard or a display) or specific hardware, making it inexpensive and suitable for CPU-constrained devices such as sensors.

In the described protocol, Alice can send the secret bit 1 to Bob by broadcasting an (empty) packet with the source field set to Alice. Similarly, Alice can send the secret bit 0 to Bob by broadcasting an (empty) packet with the source field set to Bob. Only Bob can identify the real source of the packet (since it did not send it, the source is Alice), and can recover the secret bit (1 if the source is set to Alice or 0 otherwise). An eavesdropper cannot retrieve the secret bit since it cannot figure out whether the packet was actually sent by Alice or Bob. By randomly generating n such packets Alice and Bob can agree on an n-bit secret key.

Our scheme requires that the devices being paired, Alice and Bob, are shaken during the key exchange protocol. This is to guarantee that an eavesdropper cannot identify the packets sent by Alice from those sent by Bob using data from the RSSI (Received Signal Strength Indicator) registers available in commercial wireless cards. The proposed protocol works with off-the-shelf 802.11 wireless cards and is secure against eavesdropping attacks that use power analysis. It requires, however, some firmware changes to protect against attacks that attempt to identify the source of packets from their transmission frequency.

1 Introduction

The current trend in consumer electronics is to embed a short-range wireless transmitter and a microprocessor in almost everything. The main motivation is to facilitate communication and cooperation amongst wireless devices in order to reduce their size/cost and increase their functionality. In this context, each device can be seen as a peripheral of the others. For example, a user can use the display and the keyboard of a PDA to access his cellular phone or personal server [21]. Similarly, he can use a cellular phone or PDA to retrieve temperature data sensed by a local sensor [19].

The main security challenge is to securely associate the devices together. For example, when a device receives data from a sensor, it needs to make sure that the data is received from the sensor it has selected and not from an impostor. Furthermore, integrity and privacy are often very important too.

The process of securely associating two wireless devices is often referred to as *pairing*. This process allows two devices, communicating over a short-range radio, to exchange a secret key. This key can then be used to authenticate or encrypt subsequent communication. It is important to notice that the key exchanged in a pairing protocol does not need to be authenticated since the identities (often provided by certificates) do not matter in this context. A user who is pairing two devices together only needs assurance that a key was exchanged between the devices he/she has selected (for example, the two devices he/she is holding in his/her hands).

In summary, a pairing protocol is composed of two separate sub-protocols:

1. *Key exchange* sub-protocol: this protocol is run between the two wireless devices and results in a secret key shared between the two devices.

2. *Pairing validation* sub-protocol: this protocol is executed between the two wireless devices and the user. Its goal is to guarantee (with some large enough probability) to the user that a key was exchanged between the two devices he/she actually wished to pair.

Motivations and design constraints: The motivation of this work is to design a pairing protocol for CPU-constrained devices, such as sensors. Designing pairing protocols for such environment is very challenging because sensors have limited CPU and memory. Also, because of their low costs, most of them cannot rely on tamper resistant components. The consequence of the limited computing and storage capabilities is that modular arithmetic is difficult and therefore, asymmetric cryptography cannot be used. In particular, standard Diffie-Hellman (DH)[6] key exchange protocols are excluded. Even low exponent RSA[18] techniques that allow encryption cost to be minimized are prohibitive when sensors are involved. Our goal is to design a pairing protocol that meets these constraints.

More specifically, we aim at designing a protocol that does **not** use public key cryptography and does not rely on some preconfigured information. Furthermore, the designed protocol must not increase the complexity and the cost of the sensors by requiring additional hardware (a display, an I/O interface or an out-of-band channel, such as an infrared one). Finally, it should not require exotic wireless technologies, but instead work with current wireless networking standards such as 802.11 or 802.15.4 (an emerging Wireless PAN technology, designed for low power sensors). The proposed protocol must be secure against passive and active attacks. In other words, it must not allow active or passive attackers to learn the key exchanged between two paired devices. It must provide protection against Man-in-the-Middle (MitM) attacks that attempt to impersonate one or both of the devices during key agreement. It must also provide some protection against Denial-of-Service (DoS) attacks, i.e. prevent attackers from disrupting the pairing protocol and exhausting the devices' resources, such as their battery.

Contributions: We present a novel secure pairing technique based on a key agreement protocol that does not depend on CPU-intensive operations. Two CPU-constrained wireless devices A and B can establish a shared secret over an anonymous broadcast channel. An anonymous channel is a channel on which an eavesdropper can read the packets that are exchanged but is unable to identify the source. Using such a channel, A can send the secret bit 1 (resp. 0) to B by broadcasting an (empty) packet with the source field set to A (resp. B). Only B can identify the real source of the packet (since it did not send it, the source is A), and can recover the secret bit (1 if the source is set to A or 0 otherwise). An eavesdropper cannot retrieve the secret bit since it cannot figure out whether the packet was actually sent by A or B. By randomly generating n such packets A and B can agree on an n-bit secret key.

The protocol is secure if and only if the packets of A and B cannot be distinguished by an eavesdropper. On a wireless channel, this property is difficult to achieve through protocol design since an eavesdropper can measure the signal strength of each packet and may be able to determine the real source of each packet. Therefore, we propose that during key agreement, the user(s) be very close to each other and shake their devices (i.e. randomly turn and move one around the other) in order to randomize the reception power of their packets by a potential eavesdropper and make power analysis very difficult.

Organization: The paper is structured as follows: Section 2 presents the related work. Section 3 presents the basic ideas of our scheme. Section 4 describes our proposal in detail. Section 5 presents experimental results and analysis. Section 6 presents a discussion. Finally, Section 7 concludes the paper.

2 Related work

2.1 Secure pairing

The problem of secure pairing of wireless devices has been tackled by several researchers. The proposed *key-exchange* solutions can be classified into the four main categories described in this section.

All of the approaches that we review below require some additional mechanism for *pairing validation* (except the last two ones). The only solution proposed in the literature so far is to provide the user with some evidence that both devices computed the same secret key. For example, the devices can both display a hash of the secret key [7]. These solutions are not always practical since they require devices with a display and/or a keyboard.

2.1.1 Public-key cryptography based solutions

These solutions rely on public-key based key exchange protocols such as Diffie-Hellman or RSA [7, 10, 11]. In Diffie-Hellman based schemes, devices exchange their Diffie-Hellman components and derive a key from them. In RSA-based schemes, one of the devices selects a secret key and encrypts it under the other device's public key. The main problem of these solutions is performance. They require that devices perform CPU-intensive operations such as exponentiation, which are prohibitive for CPU-constrained devices.

2.1.2 PIN-based schemes

In Bluetooth, two wireless devices derive a shared key from a public random value, the addresses of each device and a secret PIN number. The PIN number is provided to each device by the user via an out-of-band channel, such

as a keyboard. While this solution is computationally efficient, it requires that *both* devices be equipped with some kind of physical user interface. As a result, this solution cannot be used to pair devices lacking physical interfaces, such as sensors.

2.1.3 Physical contact or imprinting

In [19], Stajano and Anderson propose a solution, called *imprinting*, based on physical contact. Two devices get paired by linking them together with an electrical contact that transfers the bits of a shared secret. No cryptography is involved, since the secret is transmitted in plain-text. Furthermore, the key validation phase is not necessary since there is no ambiguity about the devices that are involved in the binding (i.e. MitM attacks are impossible).

While this solution is interesting, it requires each device to have some additional hardware to perform the electrical contact. Similarly, it might be possible to transmit a secret key through an infrared channel to a nearby node. Infrared transmissions require absolute line-of-sight links, making it more difficult for third-party interception. Nevertheless, in both cases, i.e. physical contact or infrared transmission, the complexity and the cost of the devices would increase[1]. This violates one of our design requirements, that the pairing protocol should not require extra equipment. We wish to achieve key agreement through the actual communication channel.

2.1.4 Using a Faraday cage

Alternatively, the two devices can be put in a *Faraday cage* where the secret key is transmitted in plain-text. A Faraday cage is an electrical apparatus designed to prevent the passage of electromagnetic waves, either containing them in or excluding them from its interior space. Consequently, an eavesdropper could not hear the secret key. An idealized Faraday cage is a hollow electrical conductor such as an empty sphere or box. Practical Faraday cages can be made of a conducting mesh instead of a solid conductor. However, this reduces the cage's effectiveness as an RF shield. In our case, the paired devices are small and hence can be enclosed in a conducting box. However, this solution is probably impractical for the general case; a conducting box is not always available. Users cannot possibly foresee when and where they will need secure pairing (and clearly we cannot recommend a user to carry a metal box with her all the time).

Similarly to a Faraday cage, a cable could be used for secret key transmission, instead of wireless link. However, users typically do not have cables available when they need to communicate, and requiring it will be a considerable impediment to secure communication.

One of our design requirements is to develop a protocol that does not require extra hardware/equipment. Solutions based on using a Faraday cage or cable clearly violate this constraint.

2.2 Other shaking-based schemes

As we will describe in detail, "Shake Them Up!" security depends on *shaking* two devices. "Smart-Its Friends"[12], although not related to key agreement nor security, is based on a similar user-device interaction. The authors propose that sensors be equipped with a two-axis accelerometer. When a user takes two devices in one hand and shakes them, the devices generate and broadcast similar movement data. If the difference is below a specified threshold, then the two devices recognize each other as friends and a dedicated connection is established between them. In another related work, "Are you with me?"[15], the authors propose using accelerometers to determine if two devices are carried by the same person.

In "Shake Them Up!", the shaking process has a completely different role (and no accelerometers are used in this paper). Devices are shaken/rotated for randomizing the signal power of the messages received by a potential eavesdropper.

3 Basic ideas

This section describes the main ideas of our scheme. We first describe how two devices, communicating over an anonymous channel, can exchange a secret key without CPU-expensive computation. We then define formally what we mean by "anonymous channels" and describe how they can be implemented in practice.

3.1 Pairing over anonymous channels

This section describes a new technique that allows two parties to securely exchange a secret over an anonymous channel while preventing eavesdroppers from determining its value (or actually any of its bits). By anonymous channel, we mean a broadcast channel that hides the origin of the messages. On an anonymous channel, a passive wiretapper can read the messages that are broadcast, but is unable to identify the source. Anonymous channels require a property that we call *Source Indistinguishability*. This property is defined and discussed in Section 3.2.

Our key exchange protocol was inspired from the protocol proposed by Alpern and Schneider in [3]. In this paper, the author presents a protocol that allows two parties to agree on a secret key on channels for which an eavesdropper cannot tell "who" broadcasts each message. The technique is called "Key exchange using keyless cryptography", or "Keyless key agreement".

For users Alice (A) and Bob (B) to agree on a n-bit key $K_{AB}[n]$, each first chooses its own random 2n-bit string:

$$R_A[1], R_A[2], ..., R_A[2n]$$

$$R_B[1], R_B[2], ..., R_B[2n]$$

User A then broadcasts $2n$ anonymous messages (without sender identifier), one for each bit in R_A. Similarly, user B broadcasts $2n$ anonymous messages (without sender identifier), one for each bit in R_B. The secret key is then defined by the bits $R_A[j]$ sent by A such that $R_A[j] \neq R_B[j]$. Note that there are on the average n such bits. The salient property of the protocol is that the message content is not hidden. All messages are accessible to potential eavesdroppers, which however cannot determine the origin of each message. As a result, they are unable to identify the packets sent by A and therefore identify the correct bits. Since A knows her bits, she can easily identify the bits sent by B. Similarly since B knows his bits, he can easily identify the bits sent by A. Note that the packets transmitted by A and by B must be interleaved. Otherwise it might be easy for an eavesdropper to identify the bits sent by the same source from a timing analysis. $R_A[1]$ and $R_B[1]$ should be sent first, followed by $R_A[2]$ and $R_B[2]$, the transmission order of each pair being randomized, and so on.

The protocol presented by Alpern and Schneider requires the broadcast of $4n$ messages for A and B to agree on an n-bit secret key. We propose an optimization that reduces the number of broadcast messages to n. The overview of our protocol is the following (a more detailed description is presented in Section 4):

1. A selects n/2 random bits $R_A[1], R_A[2], ..., R_A[n/2]$
2. B selects n/2 random bits $R_B[1], R_B[2], ..., R_B[n/2]$
3. A builds n/2 messages $m_A[1], m_A[2], ..., m_A[n/2]$, where the source identifier of $m_A[j]$ is either set to A if $R_A[j] = 1$ or set to B if $R_A[j] = 0$.
4. B builds n/2 messages $m_B[1], m_B[2], ..., m_B[n/2]$, where the source identifier of $m_B[j]$ is either set to B if $R_B[j] = 1$ or set to A if $R_B[j] = 0$.

- A and B send their messages synchronously but in a random order. In other words, the first messages of A and B are sent (in a random order), followed by the second messages (in a random order), and so on. In total, n messages are sent.

- For each message that A (resp. B) receives, it checks whether the source identifier is set correctly (note that only A and B can perform this verification) and sets $K_{AB}[k]$ to 1 if the source is correct or to 0 otherwise.

3.2 Source indistinguishability: definition and requirements

The described key exchange protocol requires the *source indistinguishability* property. In other words, if two parties, A and B, run the previously described key exchange protocol, an eavesdropper should not be able to distinguish the packets sent by A from the packets sent by B. Failing to achieve this property actually leads to an insecure protocol, since the eavesdropper could then recover some (if not all) bits of the exchanged key.

This notion of source indistinguishability is very similar to the notion of ciphertext indistinguishability in encryption schemes [8]. The basic idea behind indistinguishability of an encryption scheme is to consider an adversary (not in possession of the secret key) who chooses two messages, m_1 and m_2, of the same length. Then one of the messages is encrypted and the ciphertext is given to the adversary. The encrypted scheme is considered secure if the adversary cannot tell which of the two messages was encrypted.

We define *source indistinguishability* in a similar way as follows: a communication scheme between two parties A and B is *source indistinguishable* if for a given packet P, emitted by A or B, an eavesdropper cannot tell whether the packet was sent by A or B. More formally, the difference between the probability that the packet was sent by A and the probability that the packet was sent by B should be very small:

$$Pr[source(P) = A] - Pr[source(P) = B] < \epsilon$$

In practice, source indistinguishability requires the communication to be *temporally* and *spatially* indistinguishable. In the following sections, we discuss these requirements in detail [2].

3.2.1 Temporal indistinguishability

Given two parties A and B communicating together, an eavesdropper should not be able, using *timing analysis*, to identify the packets emitted from A from those emitted from B with a probability larger than $1/2$. Furthermore, the eavesdropper should not be able to group packets emitted by the same source.

It is clear from the previous definition that a communication system that uses a TDMA (Time Division Multiplexing Access) based MAC (Medium Access Control) protocol cannot provide temporal indistinguishability. In a TDMA-based system, each terminal is given one or several time slots and can transmit only during one of

its slots. As a result, it is very easy for any eavesdropper to identify the packets transmitted by a source or at least identify the packets sent by the same source.

Random access MAC protocols, such as CSMA (Carrier Sense Multiple Access) are more appropriate. CSMA protocols such as Ethernet or wireless Ethernet, multiple nodes are allowed to use the same channel in a random fashion. Before transmitting data a node listens to the channel. If the channel is busy then it waits for a random time and then listens again. If the channel is not busy, then it transmits its packet. In CSMA, collisions may happen when two terminals transmit simultaneously. CSMA/CD (Collision Detection) enables devices to detect a collision. After detecting a collision, a device waits a random time period and then attempts to re-transmit its message. With a CSMA-based system, the order of the packets sent by the different users can easily be randomized. This feature is crucial for the security of our approach. In this case, it will be very difficult for an eavesdropper to use timing information to identify the source.

3.2.2 Spatial indistinguishability

Given two parties A and B communicating together, an eavesdropper should not be able, using *spatial analysis* (or signal strength analysis), to distinguish the packets emitted by A from those emitted by B with a probability larger than $1/2$. In other words, the eavesdropper should not be able to detect the packets' source from their reception power.

This property is very difficult to achieve in practice since waves attenuate according to the *free space propagation* law and the eavesdropper can easily identify the location of the transmitter from the reception power of a received packet, i.e. from *power analysis*. More specifically, according to free space propagation law, the reception power S_r of a packet transmitted with power S_t by a transmitter that is located at a distance d is defined as: $S_r = \frac{S_t\ G_t\ G_r\ K}{d^2}$, where G_t is the antenna gain of the transmitter, G_r is the antenna gain of the receiver and K is a constant that depends on the signal frequency (or wavelength). If the receiver knows S_t and the gain, it can easily estimate d. An eavesdropper listening to a communication between two parties A and B can also use the reception power to identify the source of the packets with a probability larger than $1/2$.

Let's assume that A and B use the same type of antenna (i.e. they have the same gain) and let's define $k = G_t\ G_r\ K$. If A and B transmit their packets with a power uniformly distributed between $[S_t; S_t + \delta]$ then A receives the packets from B with a power uniformly distributed between $[\frac{k\ S_t}{d^2}; \frac{k\ (S_t+\delta)}{d^2}]$. Similarly, B receives the packets from A with a power uniformly distributed between $[\frac{k\ S_t}{d^2}; \frac{k\ (S_t+\delta)}{d^2}]$. If the eavesdropper is listening with a large antenna (i.e. G'_r is large s.t. $k' = G_t\ G'_r\ K > k$) and it is closer to A than to B (i.e. $d_A \leq d_B$), then:

1. the power of A's packets received by the eavesdropper is uniformly distributed between $[\frac{k'\ S_t}{d_A^2}; \frac{k'\ (S_t+\delta)}{d_A^2}]$ and,
2. the power of B's packets received by the eavesdropper is uniformly distributed between $[\frac{k'\ S_t}{d_B^2}; \frac{k'\ (S_t+\delta)}{d_B^2}]$.

Therefore the eavesdropper can identify all the packets received with a power between $[\frac{k'\ S_t}{d_B^2}; \frac{k'\ S_t}{d_A^2}]$ as belonging to B and all the packets received with power between $[\frac{k'\ (S_t+\delta)}{d_B^2}; \frac{k'\ (S_t+\delta)}{d_A^2}]$ as belonging to A. The scheme can only be secure if one of the two following conditions is met:

1. *Condition 1*: $d_A = d_B$. The scheme is secure because the power of the packets sent by A is statistically indistinguishable from the power of the packets sent by B. If $d_A \neq d_B$, the eavesdropper can identify some of the bits of the secret key. The number of bits that can be identified depends of the values of d_A and d_B. If $d_B > (S_t + \frac{\delta}{S_t})^{1/2} d_A$, the eavesdropper can guess the source of all the packets exchanged between A and B and therefore all bits of the secret key. If $d_A < d_B < (S_t + \frac{\delta}{S_t})^{1/2} d_A$, the eavesdropper can guess the source of some percentage of the packets. This percentage depends on the difference between d_A and d_B. Note that if the eavesdropper can monitor at several locations, and receive the same packets with different reception powers, it will be even easier for her to identify the source of the packets.
2. *Condition 2*: A and B move during the pairing phase such that d_A and d_B (and therefore their respective powers) are statistically indistinguishable.

4 Movement-based pairing

This section describes a new protocol that can be used to pair two devices A and B securely and without using expensive public-key cryptographic protocols. Our key agreement protocol is based on the combination of the two following: we first optimize Alpern and Schneider's keyless key agreement protocol and adapt it to an ad hoc configuration. Secondly, we show that the protocol can be secured against power analysis by shaking the two devices around each other. We show that it is secure against DoS (Denial-of-Service) and MitM (Man-in-the-Middle) attacks.

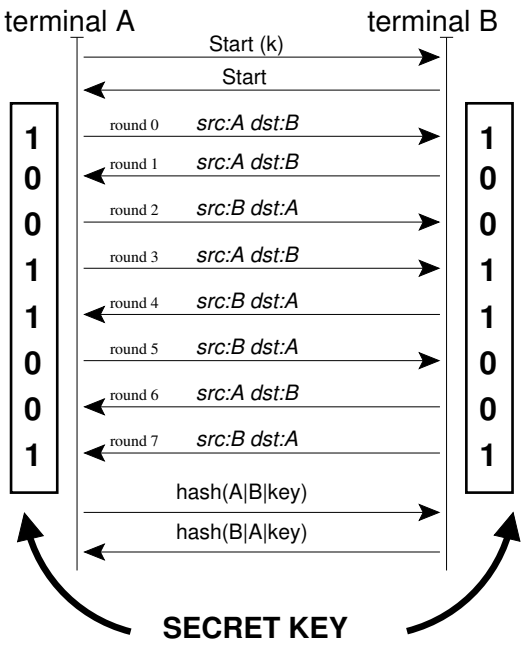


Figure 1: Key agreement protocol for movement-based pairing.

4.1 Key agreement

In our pairing protocol, key agreement is performed using the protocol described in Figure 1. This protocol is an optimization of the approach proposed by Alpern and Schneider [3]. The number of messages per secret bit is reduced (1 instead of 4), making the protocol more energy efficient.

The protocol starts by a START message transmitted by one of the two parties, either A or B. This message contains the value k which is the size of the key, and the address of the packet's source. Upon reception of this message, the other party replies with another START message that contains its address. This exchange allows each device to learn the other party's address and the size of the desired shared key. It should be triggered by the user, for example, pushing a button on both devices.

At each round j, either A or B (with equal probability) broadcasts an empty packet at time t_j, where t_j is randomly selected in the interval $[jT; (j+1)T]$, and T is a constant. Secret bits are represented by the correct or inversed placement of source and destination addresses. If the sender and recipient addresses are correct, the terminals A and B presume a secret bit TRUE (1), otherwise they presume FALSE (0). For example, in Figure 1, the first message is sent by A. The source address is set to A and the destination address is set to B. Since the packet was actually sent by A, the resulting bit (the first bit) of the secret key is then set to 1 by A and B. Note that an eavesdropper cannot identify the real source of the packet and therefore cannot retrieve the value of the exchanged secret bit. Each packet identifies one bit of the secret key. At the end of the k rounds, A and B share a k-bit long secret key. Therefore, if a 60-bit long key is required, the protocol should contain 60 rounds, i.e. 60 packets.

The protocol is terminated by two messages, that are used to validate the exchanged key. The message sent by A contains the value $a = hash(A|B|key)$, where key is the exchanged key. The message sent by B contains the value $b = hash(B|A|key)$. The order of these two messages is also random. When B receives the value a, it can verify that A has the same key. Similarly, A can verify, upon reception of b, that B computed the correct key.

4.2 Achieving spatial indistinguishability: Shake them up!

As described in Section 4, the previous scheme is secure only if the source of the packets are indistinguishable.

Time indistinguishability is provided by randomizing the order of transmission of packets sent by A and B, as described in the previous section. An eavesdropper can therefore not guess who is going to transmit next. Also, we are using CSMA-based wireless systems, such as 802.11, to guarantee that the access to the channel is also random and does not reveal any information about the source.

As explained previously, *spatial indistinguishability* is more difficult to achieve. We propose to achieve this property with user assistance. The user(s) should shake (i.e. move and turn) the devices during key agreement in order to equalize the average signal strength of the two devices measured by a potential eavesdropper.

The required movements depend on the type of antennas used. For truly omni-directional antennas, antenna orientation will not reveal any signal strength difference between two devices. In these cases, it will be sufficient to take both devices and turn them, one around the other, in order to equalize the effect of distance (between each terminal and the eavesdropper) on the signal strength measurements performed by an eavesdropper. If the antennas are not truly omni-directional, randomizing the distance will not be enough to achieve spatial indistinguishability. Different orientation of the devices may reveal a serious signal level difference. In order to avoid this problem, during key agreement the two devices must be randomly turned to different directions (in our experimentations we used commodity 802.11 cards which are not omni-directional. Section 5 contains more details on this issue).

Clearly, in both cases, having small and lightweight devices (e.g. sensors or small PDAs) will reduce the user burden for key agreement. The user can take one device in each of his hands and randomly move them one around the other according to the horizontal and vertical axes. If the devices are very small, he can take both of them in one hand and shake his arm, like he would do with an orange juice bottle.

The security of the proposed scheme depends on the quality of the movement. Users of our scheme should be aware that it is their responsibility and in their best interest to move the devices properly during the pairing phase. Movement-based operations or "protocols" are quite frequent and accepted in our everyday lives. For example, orange juice or shaving cream bottles are universally shaken prior to use. This is now a well-known and quite a natural protocol. Furthermore it is commonly accepted that it is the responsibility and in the interest of the consumer to perform this shaking operation properly.

4.3 Protection against MitM and DoS attacks

4.3.1 Protection against MitM attacks

Defeating a MitM (Man-in-the-Middle) attack requires assurance for a terminal A that a secret key is really being exchanged with the intended terminal B and not an impostor's device. This is the goal of the *Pairing Validation* protocol, as described in Section 1.

In our case, this problem is reduced to the following issue: "how do the two devices reliably determine each other's address in the presence of many other devices?" As explained in Section 2.2, the smart-its friends scheme solves this problem. However this solution may be considered costly and impractical since it requires extra hardware, namely an accelerometer. Certificates could also be used for authenticating the START messages. However, our goal in this paper is to avoid CPU-expensive operations such as exponentiations or signature verifications.

In our case, $proximity$ (and hence high signal level of START messages) is used for authentication. In "Shake Them Up!", the pairing protocol is triggered by the user by, for example, pushing a button on the devices. At this point, the devices will start to generate START messages at a specified rate. The user will bring the two devices near to each other (possibly resulting in antenna contact). The devices will receive each other's START message with a high signal level, starting the "Shake Them Up!" procedure. Experience with 802.11 cards has shown that a very high signal level can be obtained when two cards touch each other, and a distance about 1 or 2 cm quickly results in a much lower signal level. Using a signal level threshold e.g. 0 or 1 dBm, this device association protocol can be implemented. In our case, the higher the specified rate of START messages, the faster the devices can detect each other with high signal level (when the user finds the correct positioning). Let Q be the period of broadcast START messages. While initiating the pairing process, if the user missed a START message (i.e. it was received at a lower signal level than the specified threshold), the user must wait another Q time units. Consequently, Q must be low, e.g. for example 1 second, to allow the user to easily and quickly start the pairing process. Once the two devices obtain each other's address correctly, MitM attacks will be impossible.

A distant impostor may attempt to foil this technique by increasing its transmission range. However this attack is easily detectable since the victim will receive several START messages at a rate higher than Q.

One of the devices, say device B, may be down and an attacker may profit from this situation to impersonate B. This attack can be prevented if each device has a status LED which indicates whether it is active or down. Even a sensor device (with no display) is likely to have such a LED. Furthermore, if the devices A and B are shaken together (i.e. the user holds the two devices in one hand and shakes his hand), A should receive the messages coming from B with constant signal power. Most likely, the signal power of the different messages sent by the attacker will be different (since the attacker is not shaken together with A and therefore does not follow the same mobility pattern). In this scenario, this attack can easily be detected by A.

4.3.2 Protection against DoS attacks

We can differentiate between two kinds of DoS (Denial-of-Service) attacks on a key agreement protocol. In the first one an attacker may exploit the key agreement protocol to force a victim to perform computationally expensive operations, with the goal of draining its battery or preventing it from performing useful work. Unlike public-key cryptography-based schemes, our protocol is not based on CPU-intensive operations and therefore immune against such DoS attacks. Another DoS attack may consist of sabotaging the key agreement, i.e. making it impossible for the two parties to agree on a same secret key.

In our basic scheme, it is easy for an attacker to insert a bogus packet with source address A and destination address B (or vice versa) and perform what we call a *key poisoning* attack. Such a packet inserted by a third party would generate different secret bits at the terminals A and B. The attacker can insert an arbitrary number of bogus packets, and make it impossible for A and B to agree on a secret key.

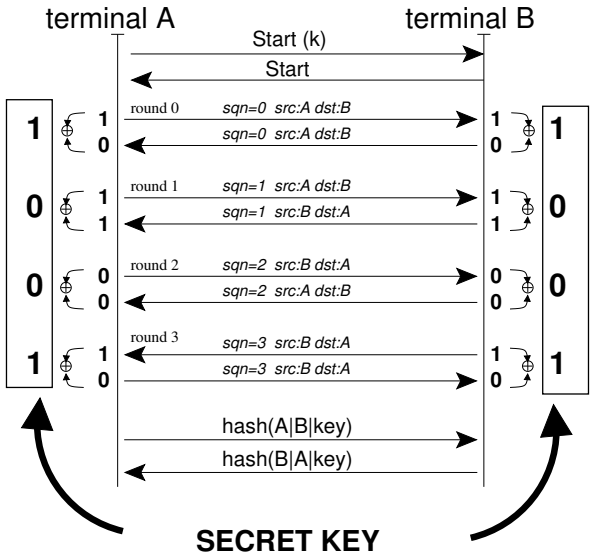


Figure 2: A protocol that resists what we call a *key poisoning DoS attack*. When this protocol is used for key agreement, an active attacker cannot poison (i.e. corrupt) the secret key by inserting bogus messages with the addresses of A and B (see text for details). This protection is obtained at the cost of two messages per secret bit.

The protocol depicted in Figure 2 defeats the key poisoning attack. In this protocol, each secret bit is constructed using one packet from A and another from B, i.e. both terminals contribute to the construction of each secret bit. Each secret bit is given a sequence number (which also corresponds to the round number). In order to generate the secret bit i the two terminals generate a packet with correct or flipped address positions, in random order (i.e. the probability that the first packet i will be transmitted by A is 0.5). The outcome of the two packets i are combined by taking their sum (mod 2), or exclusive OR. The result is the secret bit i.

In order to alterate the secret bit i, an attacking node can insert x packets with the same sequence number. In this case both sides will record $x + 2$ bits with the same sequence number, but only two of them will be the same at both sides. Let $\{a_1, ..., a_{x+2}\}$ and $\{b_1, ..., b_{x+2}\}$ the set of bits (with the same sequence number) that the terminals A and B have recorded. Then we have

$$a_1 \oplus \cdots \oplus a_{x+2} = b_1 \oplus \cdots \oplus b_{x+2}$$

if x is even, and

$$b_1 \oplus \cdots \oplus a_{x+2} \neq b_1 \oplus \cdots \oplus b_{x+2}$$

if x is odd.

Key poisoning can be defeated by taking the sum (mod 2) of the bits with the sequence number i, as usual (except that in normal operation $x = 0$). Note that if the attacker inserted an odd number x of packets, the terminal A must invert the resulting secret bit i. This protocol requires twice as many messages as the basic protocol. However, this protocol is only necessary when the user believes that his devices are under DoS attack. Otherwise, the basic scheme should be used.

This protocol fails, however, when A and B do not receive the same messages. This might happen when some of the messages are lost. We therefore suggest that A and B append to each of their messages a hash of all the previous messages they have seen since the beginning of the protocol. As a result, if one or several messages are lost, A and B can detect it immediately (instead of waiting until the end of the protocol and comparing a hash of the derived keys).

5 Experimentations and analysis

In this section, we analyze, by experimentations, the security of the proposed pairing scheme. More specifically, we show that signal power analysis cannot be used by an attacker to retrieve the key exchanged between two devices that use our pairing protocol. We also evaluate the energy cost of our protocol and show that although it requires several messages, it is much more energy-efficient than a Diffie-Hellman based pairing protocol.

5.1 Setup and methodology

We built a testbed with lightweight laptops equipped with a PCMCIA Lucent IEEE 802.11 Wavelan card operating at 2.457GHz (802.11 channel 10) and 2Mbps bit rate and in ad hoc mode. Our cards support RSSI (Received Signal Strength Indicator) and allow us to visualize and evaluate the power of received packets. Our signal level cryptanalyzer is built upon Linux wireless tools[3], and in particular *iwspy* that allows to get per node link quality. The $iwspy$ command takes as argument a MAC address M, and outputs the received signal and noise levels of packets whose source address is M.

Figure 3 depicts a typical measurement that can be carried out by any user using the $iwpsy$ tool and a simple sampling script. Note that $iwspy$ also outputs the noise level which is about -96dBm in our environment.

The received signal strength depends on at least 4 distinct factors:

1. Transmission power: Packets are transmitted at our cards' default value which is 15 dBm[4].
2. Distance between the source and the signal level analyzer.
3. The relative angle between the source and the signal level analyzer: the cards that we use are not

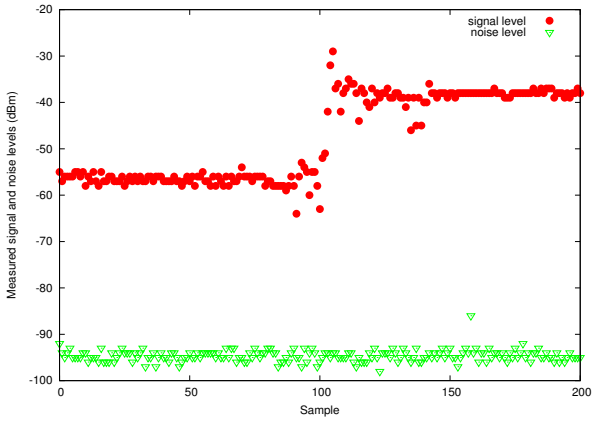


Figure 3: A typical *iwspy* output. In this example, the "iwspied" terminal is stationary while the first ∼100 samples are taken and then it moves to a location that is closer to the signal level analyzer machine. The distance between the two cards is easily determined by signal strength analysis.

omni-directional and the received signal level depends heavily on the relative angle of the two cards.

4. Relative position of the terminals (or, cards): when the two terminals are very close, they may become obstacles for each other. For example, terminal A may be in front of terminal B. In this case, packets from A are received at a higher signal level (even if the above factors had no impact on the signal level difference).

During each experiment, referred as 'scenario', Eve (eavesdropper, or cryptanalyzer) measures the signal strength of the packets sent by Alice (terminal A) and Bob (terminal B) during key agreement. Many different experiments were carried out. We only provide the most representative ones that we consider generic and applicable to almost all situations because they perfectly reflect the points (2) and (3) listed above. Our first scenario, denoted *scenario1*, is illustrated in Figure 4-a. In this scenario Alice and Bob are close to each other (within 0.5 meter) and make two kinds of movements in order to equalize their average signal strength captured by Eve:

- We use commodity wireless Ethernet cards and they are not omnidirectional. Thus, in order to confuse Eve, Alice and Bob must turn their laptop in randomly changing directions (at a reasonable speed). The process requires reasonable effort from the user and takes around 16 seconds for agreeing upon an 80-bit secret key. The details of movement speed and its effect on security will be discussed later.
- Alice and Bob move their devices one around another with a reasonable effort, i.e randomly and at a moderate speed. This helps Alice and Bob to hide their relative distance between their cards and a potential eavesdropper that may be located anywhere.

In scenario1, we consider a pessimistic passive attack where Eve's wireless Ethernet card (the white arrow) is directly oriented to Alice and Bob and situated only 2.2 meters away. This allows Eve to make relatively accurate signal level measurements. In practice, Alice and Bob would probably notice the presence of a third person during key agreement, and look for another place where eavesdroppers cannot approach them. However, in some situations such countermeasures may not be practical. This scenario attempts to capture the cases where the presence of a third person cannot be avoided. Note also that an eavesdropper may have installed hidden signal level cryptanalyzers at strategical points. Thus, the absence of a third person, does not necessarily imply a secure environment. For example, the eavesdropper might be behind a thin wall or partition (e.g. a cubicle wall), and not visible to Alice and Bob. In this scenario Alice and Bob respect the key agreement requirements, hence the key will be secure as we will show below.

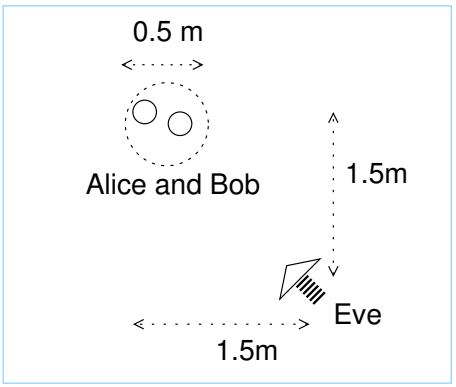
In scenario2 (Figure 4-b), we demonstrate an inappropriate usage of our protocol that we would like to disadvise. In this scenario Alice and Bob are not close to each other. They both move their laptop randomly in every possible direction, but they are always far from each other and their location does not change during key agreement. Eve profits from the distance between Alice and Bob, and directs her card to Alice. Consequently, Alice's packets are received at a higher signal level than that of Bob, rendering the secret key weak.

The scenario3 (Figure 4-c) is even less secure and firmly disadvised. Alice is 4 times closer to Eve than Bob. Eve is located between the two terminals and profits from the situation by directing her wireless Ethernet card to Alice. Although Alice and Bob's cards are perfectly turned in random directions, Eve can easily differentiate between their packets. As a result, the key is extremely weak.

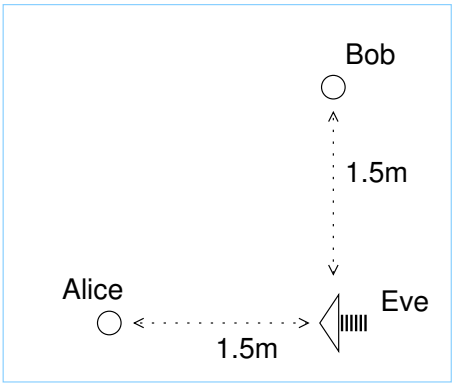
5.2 Signal power analysis

The signal level cryptanalysis results for the above three scenarii are shown in Figure 5.

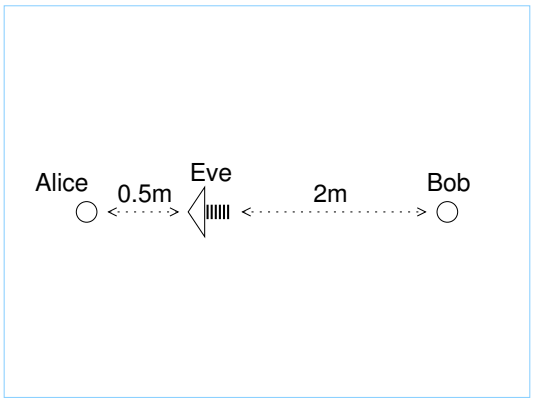
In scenario1, we observe that Alice and Bob's packets are mixed and not easily distinguishable by Eve (the reader may imagine that Eve's vision has only one color regardless of the sender's ID). The only information available to Eve will be the absolute value of signal level difference between two packets captured during



(a) scenario1

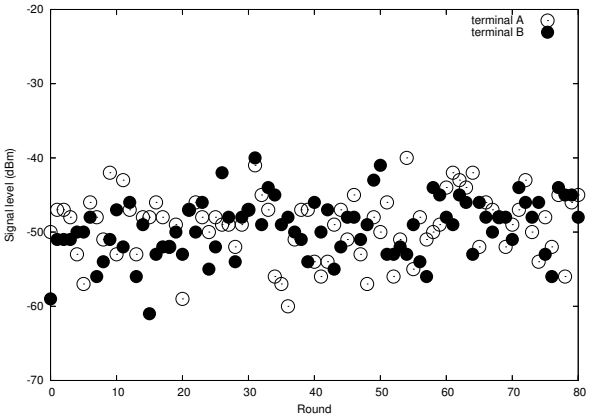


(b) scenario2

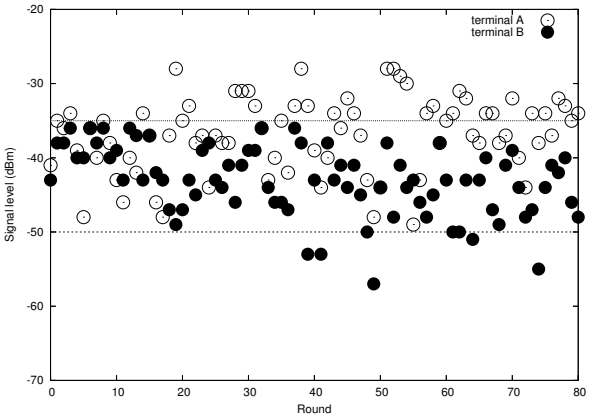


(c) scenario3

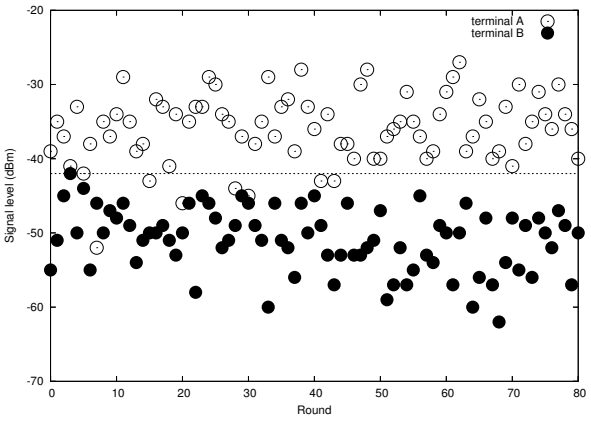
Figure 4: Experimentation scenarii.



(a) scenario1



(b) scenario2



(c) scenario3

Figure 5: Received signal level of terminal A (Alice) and terminal B (Bob) during key agreement. The reader may imagine that Eve's vision has only one color (i.e. all packets are black).

each round (1 packet from Alice, 1 packet from Bob). In Figure 6 we provide a frequency diagram of these signal level differences. It should be noted that, when plotting these histograms we have profited from additional information that is not available to Eve: the sign of the observed differences (i.e. (+) when Alice's packet is received with greater signal level than that of Bob, and (-) otherwise). These histograms were plotted using 1000 samples (i.e. rounds) in order to provide accurate results that correspond to the average case (for a given scenario). For our data collection purposes, Alice and Bob performed the required laptop movements for a much longer time than needed in practice: ~ 3.5 minutes for each experiment (in our experiments Alice and Bob generated 0.2 packets per second, as we will explain later). The histogram that corresponds to scenario1 is centered on $\sim$0 and roughly symmetric. Consequently, we conjecture that Alice and Bob's packets cannot be distinguished using signal strength analysis.

In practice, the results look satisfactory in scenario2. There is no well defined technique (at least to our knowledge at time of writing) that will allow to clearly distinguish Alice's and Bob's packets. Note for example that, above the line -50dBm all packets are Alice's packets. Similarly, below the line -35dBm, we have only Bob's packets. However, unlike the reader, this information is not provided to Eve.

On the other hand, the resulting key is clearly insecure in theory. As shown in Figure 5-b the signal level differences are important: the histogram is centered at 7.34 dBm. Although it is unknown to the attacker, this difference is considerable (making us uncomfortable) and reflects very well the fact that Eve's wireless Ethernet card is directed to Alice. Thus, we base our conclusion on the theoretical security of the protocol and disadvise the type of scenario where Alice and Bob are 'not' close to each other. The security of the secret key in this scenario could be improved by adding more rounds, however this would lead to a very inefficient key agreement. We recommend that Alice and Bob be as close to each other as possible (in addition to turning their devices in random directions).

In the final scenario, scenario3, the situation is clearly worse. The resulting key is not only 'theoretically' breakable as shown by the frequency diagram (centered at 14.92 dBm), but also breakable in practice. Figure 5-c reveals what we call a "break point" which is situated around -41dBm. There is a visible gap at that point where Alice and Bob's packets are clearly separated.

5.3 Energy consumption considerations

In this section, we compare the power consumption of our scheme with the power consumption of a Diffie-Hellman based pairing. For the purpose of this comparison, we assume that the two devices being paired are sensors using TinyOS and that the size of the generated shared key is 72 bits.

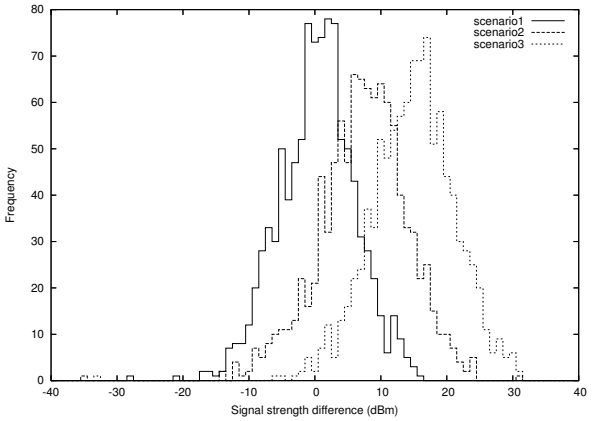


Figure 6: Frequency diagrams for signal level difference. In scenario1, the spatial indistinguishability requirement is satisfied. The histogram is centered on zero and symmetric. Thus, in this paper it is conjectured that an eavesdropper cannot distinguish the source of the packets regarding signal level difference (in scenario1).

With our scheme, each device must receive and send 36 packets. Considering that a TinyOS packet that has a header size of 7 bytes [13], each device must send and receive 2016 bits ($36 \times 8 \times 7$) (as explained in Section 4, in our basic protocol, packets do not have to contain any payload). However transmitting one bit consumes about as much power as executing 800-1000 instructions [9, 13]. Receiving one bit consumes about half as much power as sending one bit. As a result, our protocol consumes as much energy as the execution of about 2.72×10^6 instructions ($2016 \times 900 + 2016 \times 450$).

In comparison, with a Diffie-Hellman based pairing protocol, each device needs to exchange their Diffie-Hellman public component (i.e. g^x, where x is the device's private key). A security equivalent to 72 bits requires to select a modulus of 1024 bits and an exponent of 130 bits [14]. As a result, the device's Diffie-Hellman public component is 1024-bit long. Since the maximum number of payload bits in a TinyOS packet is 232, each device must send (and receive) 5 packets. Therefore, the total number of bits sent and received is 1304 bits: 4 packets containing 232 bits and 1 packet containing 96 bits. This consumes as much energy as the execution of 1.76×10^6 instructions ($1304 \times 900 + 1304 \times 450$). Upon reception of the other party's public component, each device has to exponentiate it with its Diffie-Hellman private key. Exponentiating using the Montgomery algorithm re-

quires $3 \times l \times (l+1) \times (t+1)$ single-precision multiplications, where l is the size of the modulus and t the size of the exponent [16]. With $l = 1024$ and $t = 130$, each device must perform 4.12×10^8 single-precision multiplications. In conclusion, the total power consumed by each device is therefore equivalent to the power consumed by the execution of $1.76 \times 10^6 + 4.12 \times 10^8$ instructions. This cost is about *100 times* larger than the cost of our scheme.

The bandwidth cost of the Diffie-Hellman based solution could be significantly decreased with Elliptic Curve Cryptography. In fact, the security of a 1024-bit Diffie-Hellman key exchange is equivalent to the security of a 135-bit Elliptic Curve Diffie-Hellman (EC-DH) key exchange [14]. Therefore only one packet would be necessary to be exchanged by the two devices. This would reduce the energy cost due to communication by 5. However, as shown in [5], EC-DH key derivation cost is even more expensive than regular DH key derivation. Therefore the total energy cost would still be much higher than the cost of our scheme.

6 Discussion

In this section, we present a discussion of more sophisticated attacks that the "Shake Them Up!" strategy may face.

6.1 RF analysis attacks on reference clocks

How secure is our protocol against a well equipped attacker? An attacker may use more sophisticated equipment, and conduct much more rigorous cryptanalysis. Using an RF test equipment, for example, it is possible to snoop the packets and record their signal shape. By studying the signal shape of each packet, additional information may be discovered and help distinguish one of the device's packets from the other. Although the signal amplitude should not reveal anything more than an RSSI measurement, signal-frequency may reveal a drift between the participants' reference clock implementation.

By current practice, a quartz crystal or crystal clock oscillator stabilizes the carrier and baseband frequencies in an RF transceiver. In order to ensure frequency lock between two devices, and avoid serious phase noise, a tight-stability reference clock is necessary. Nevertheless, a reference clock implementation is never perfect. A random clock drift is generally unavoidable, due to practical difficulties found at the hardware level. Typically, an error of up to $\pm$25ppm (parts per million) is tolerated. This tolerance includes the initial calibration tolerance at 25°C, frequency changes over operating temperature, power and load fluctuations, and aging[17]. For example, at 2.4GHz carrier frequency, a frequency offset of up to $\frac{2.4\times10^9\times2\times25}{10^6}$ =120kHz would be tolerated. [4] reports 250kHz of central frequency accuracy tolerance. The main reason for clock drift is aging. I.e. the clock drift is mostly stable in short-term (except in case of shock, or abrupt temperature changes), but logarithmically increases in time[20, 1]. A clock drift from ±1 to ±5 ppm/year can be incurred depending on the crystal used [2].

Consequently, during "Shake Them Up!", device A may always transmit at a central frequency f_A while the device B transmits at f_B, where $f_A = f_B + \epsilon$. A third party equipped with an RF analyzer can then retrieve the secret key by correlating the packets with the same central frequency. A frequency shift of several kHz is unfortunately too large to be compensated with the *Doppler effect* made by separately shaking the devices. A shaking speed of $\pm$10m/s would only make a Doppler effect between $\pm$50Hz (at 2.4GHz) which probably cannot counterbalance the error ϵ.

A possible defense against this attack consists of adding a deliberate and random frequency offset so that f_A and f_B span over similar frequency ranges. This solution however requires firmware changes, making it a longer-term solution. Let the frequency offset tolerance be $\pm\delta$ and $f_A < f_B$ (i.e. f_A and f_B are within the range $[f - \delta; f + \delta]$) and both devices add a deliberate random frequency shift t to each packet. The devices will have a frequency range between $[f_A - t; f_A + t]$ and $[f_B - t; f_B + t]$ respectively. Assuming that an eavesdropper, Eve, knows f_A and f_B, then she can deduce that the packets transmitted with a frequency larger than f_A+t originate from B, and the packets transmitted with a frequency smaller than $f_B - t$ originate from A.

However, the packets received within range $[f_B - t; f_A + t]$ are emitted by A or B with the same probability. This is illustrated in Figure 7. Let k be the number of packets emitted by each device. It can be shown that, $\frac{k}{2t} \times (f_A - f_B + 2t)$ packets of A (and B) will have a central frequency between $[f_B - t; f_A + t]$. If we wish to use an 80-bit secret key, at least 40 packets of A and 40 packets of B must be in this frequency range. This condition is satisfied if: $\frac{k}{2t} \times (f_A - f_B + 2t) > 40$, i.e.

$$k > \frac{40 \times 2t}{f_A - f_B + 2t}$$

where $t > \delta$ (recall that 2δ is the maximum frequency shift between two devices). In the worse scenario, $f_A = f - \delta$ and $f_B = f + \delta$, i.e. $f_A - f_B = -2 \times \delta$ and $k > 40 \times t/(t - \delta)$. If t is set to $2 \times \delta$, then k can be set to 80. To summarize, by setting t to $2 \times \delta$ and k to 80 (instead of 40 in the basic scheme), the generated secret is secure for any value of f_A and f_B in $[f - \delta; f + \delta]$.

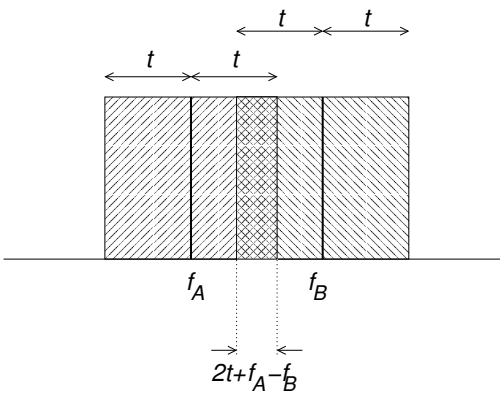


Figure 7: A technique for hiding the clock drift between two devices. In the region where the central radio frequencies of each device overlap, the packets of A and B are indistinguishable.

6.2 Camera-assisted packet captures

Another type of attack that "Shake Them Up!" may be exposed to is a video camera assisted attack. Using a signal level analyzer that is synchronized with a video camera, an attacker may correlate different device locations and their respective signal power during the shaking process. Users may employ different strategies against this threat such as hiding the sensor devices with their hand (provided that the devices are small).

Hiding the devices has also the additional benefits that it protects our scheme against eavesdroppers with unidirectional antennas. Since the location of the sensors are hidden, an eavesdropper cannot aim an antenna at one of the sensors for identifying its packets.

7 Conclusion

In this paper we presented a novel secure pairing scheme for CPU-constrained devices without a need for special hardware or interfaces. Using an existing communication channel such as 802.11 or 802.15.4, two devices that are close to each other can agree on a secret key using an algorithm that does not depend on CPU-intensive operations. On the other hand, user assistance is required for *shaking* the devices during key agreement in order to preserve key secrecy.

One alternative could consist of randomly varying the signal level (in software) during key agreement. However, this solution is not secure because an eavesdropper may aim an unidirectional antenna at one device, identify its packets and therefore retrieve the secret key. Furthermore we have discovered by experimentations that if the two devices are not shaken, one of them can mask the signal of the other one and attenuate its transmission power significantly. Consequently, the packets from each device are received at a different signal level, and the secret key can easily be retrieved. Shaking solves all these problems. It seems to be the only solution that can address all kinds of RSSI-based signal level analysis threats that our key agreement protocol may face. The proposed protocol works with off-the-shelf 802.11 wireless cards and is secure against eavesdropping attacks that use power analysis. It requires, however, some firmware changes to protect against attacks that attempt to identify the source of packets from their transmission frequency.

One limitation of our scheme is that it is specific to random media access technologies. For example, it is not suitable for TDMA-based protocols and, therefore, cannot be used with Bluetooth devices. Our scheme requires CSMA-based systems, such as 802.11 or 802.15.4 (an emerging Wireless PAN technology, designed for low power sensors). Another noticeable limitation is that it requires that the transmission power of both devices be similar. This was the case with the 802.11 devices that we used for our experimentations. However, for some wireless technologies, a power control protocol might be required to adjust the transmission power accordingly.

Objects with microprocessors and wireless transceivers surround us. Today's users are more and more technology and security-aware. Almost all users today learned that a system access password should contain non-alphanumeric characters. We have learned (or are forced to learn) how to handle computer viruses. Technology and information security have become part of our everyday lives. Thus, we believe that future users can also learn that *two small devices must be shaken well before secure use*. This is actually a very common protocol that we already execute everyday. For example, orange juice or shaving cream bottles are universally shaken/moved before usage. This is now a well-known and quite a natural "protocol". Furthermore it is commonly accepted that it is the responsibility and in the interest of the consumers to perform this shaking operation properly. Similarly, in our case by shaking the devices well, the user can make sure that the two devices are paired securely.

Acknowledgement

We would like thank Roy Want, Gene Tsudik and the anonymous reviewers for their excellent remarks that helped us improve this paper.

References

[1] Fundamentals of Quartz Oscillators. HP Application Note 200-2.

[2] http://www.telluriantech.com. Specialty Crystals, Quartz Crystals.

[3] ALPERN, B., AND SCHNEIDER, F. Key exchange using "Keyless Cryptography". *Information processing letters 16*, 2 (February 1983), 79–82.

[4] CHAYAT, N. 802.11a PHY Overview. Slides available at: http://www.nwest.nist.gov/mtg3/papers/chayat.pdf.

[5] DAI, W. Speed benchmarks for various ciphers and hash functions. URL:http://www.eskimo.com/∼weidai/.

[6] DIFFIE, W., AND HELLMAN, M. New directions in cryptography. *IEEE Transactions on Information Theory IT-22*, 6 (1976), 644–654.

[7] GEHRMANN, C., AND NYBERG, K. Enhancements to bluetooth baseband security. In *Nordsec'01* (Kopenhagen, Denmark, November 2001).

[8] GOLDWASSER, S., AND BELLARE, M. Lectures notes in cryptography. URL:http://www.cs.ucsd.edu/users /mihir/papers/gb.html.

[9] HILL, J., SZEWCZYK, R., WOO, A., HOLLAR, S., CULLER, D. E., AND PISTER, K. S. J. System architecture directions for networked sensors. In *Architectural Support for Programming Languages and Operating Systems* (2000), pp. 93–104.

[10] HOEPMAN, J.-H. Ephemeral pairing in anonymous networks. Available at: http://www.cs.kun.nl/∼jhh/publications/anon-pairing.pdf.

[11] HOEPMAN, J.-H. The ephemeral pairing problem. In *8th Int. Conf. Financial Cryptography* (Key West, Florida, February 9-12 2004), pp. 212–226.

[12] HOLMQUIST ET AL, L. A. Smart-Its Friends: A Technique for Users to Easily Establish Connections between Smart Artefacts. In *Ubicomp 2001* (Atlanta, Georgia, September 30, October 2 2001).

[13] KARLOF, C., SASTRY, N., AND WAGNER, D. Tinysec: A link layer security architecture for wireless sensor networks. In *Second ACM Conference on Embedded Networked Sensor Systems (SenSys 2004)* (November 2004).

[14] LENSTRA, A. K., AND VERHEUL, E. R. Selecting cryptographic key sizes. *Journal of Cryptology: the journal of the International Association for Cryptologic Research 14*, 4 (2001), 255–293.

[15] LESTER, J., HANNAFORD, B., AND G., B. "Are You with Me? - Using Accelerometers to Determine If Two Devices Are Carried by the Same Person". In *Pervasive 2004* (Vienna, Austria, April 21-23 2004).

[16] MENEZES, A. J., VAN OORSCHOT, P. C., AND VANSTONE, S. A. *Handbook of applied cryptography*. CRC Press series on discrete mathematics and its applications. 1997. ISBN 0-8493-8523-7.

[17] OGILVIE, B. Clock Solutions for WiFi (IEEE 802.11). Saronix(tm) application note, 2003.

[18] RIVEST, R., SHAMIR, A., AND ADLEMAN, L. A method for obtaining digital signatures and public-key cryptosystems. *Coomunications of the ACM 21* (1978), 120–126.

[19] STAJANO, F., AND ANDERSON, R. The resurrecting duckling: Security issues for ad-hoc wireless networks. In *Proceedings of the 7th International Workshop on Security Protocols* (1999), pp. 172–194.

[20] VIG, J., AND BALLATO, A. Frequency Control Devices. Reprinted from Ultrasonic Instruments and Devices, Academic Press, 1999.

[21] WANT, R., AND PERING, T. New Horizons for Mobile Computing. In *First IEEE International Conference on Pervasive Computing and Communication (PerCom'03)* (Dallas, Texas), pp. 3–8.

Notes

[1] Since infrared channels require line-of-sight links, they cannot be efficiently used for the actual communication between the sensors.

[2] We assume that packets do not carry information that can help identify the source address. Thus we concentrate our efforts on temporal and spatial indistinguishability problems.

[3] Available at: http://www.hpl.hp.com/personal/Jean_Tourrilhes

[4] Note that the spatial indistinguishability property requires that the two devices set the same transmission power. We observed that almost all vendors set it to 15 dBm. Otherwise, the devices should modify their transmission power to a specified value and keep it constant during key agreement.