Security Weaknesses in Bluetooth

Markus Jakobsson and Susanne Wetzel

Lucent Technologies - Bell Labs
Information Sciences Research Center
Murray Hill, NJ 07974
USA
{markusj,sgwetzel}@research.bell-labs.com

Abstract. We point to three types of potential vulnerabilities in the Bluetooth standard, version 1.0B. The first vulnerability opens up the system to an attack in which an adversary under certain circumstances is able to determine the key exchanged by two victim devices, making eavesdropping and impersonation possible. This can be done either by exhaustively searching all possible PINs (but without interacting with the victim devices), or by mounting a so-called middle-person attack. We show that one part of the key exchange protocol – an exponential back-off method employed in case of incorrect PIN usage – adds no security, but in fact benefits an attacker. The second vulnerability makes possible an attack – which we call a *location attack* – in which an attacker is able to identify and determine the geographic location of victim devices. This, in turn, can be used for industrial espionage, blackmail, and other undesirable activities. The third vulnerability concerns the cipher. We show two attacks on the cipher, and one attack on *the use of* the cipher. The former two do not pose any practical threat, but the latter is serious. We conclude by exhibiting a range of methods that can be employed to strengthen the protocol and prevent the newly discovered attacks. Our suggested alterations are simple, and are expected to be possible to be implemented without major modifications.

1 Introduction

The ubiquity of cellular phones turn them into a commerce platform of unprecedented importance. While personal computers have allowed e-commerce to flourish within a rather limited socio-economic segment of society, cell phones promise an expansion of electronic commerce to virtually the entire population. At the same time, and given their portable nature, cell phones also promise to extend the possibilities of commerce to what is popularly called mobile commerce, or *m-commerce*. An important step towards the development and penetration of m-commerce is the employment of short-range wireless LANs, such as Bluetooth.

Bluetooth [5,7,8] is a recently proposed standard for local wireless communication of (potentially mobile) devices, such as cellular phones, wireless headsets, printers, cars, and turn-stiles, allowing such devices in the proximity of each

© Springer-Verlag Berlin Heidelberg 2001

other to communicate with each other. The standard promises a variety of improvements over current functionality, such as hands-free communication and effortless synchronization. It therefore allows for new types of designs, such as phones connected to wireless headsets; phones connected to the emergency systems of cars; computers connected to printers without costly and un-aesthetical cords; and phones connected to digital wallets, turn-stiles and merchants.

However, the introduction of new technology and functionality can act as a double-edged sword. While the new technology certainly provides its users with increased possibilities, it can also provide criminals with powerful weapons. Recently, the public has started to pay attention to the need for privacy for applications relating to telephony, with fears of vulnerabilities and abuse mounting. It is likely that public opinion will further strengthen if there is some high-profile case in which somebody's privacy is abused. For some recent concerns, see, e.g., [1,11,13]; for some independent work on the analysis of Bluetooth security, see [4,12]. (The latter of these references present findings of a very similar nature to ours.)

Thus, we argue that careful analysis and prudent design is vital to the success of products. In keeping with this, we exhibit vulnerabilities in the Bluetooth 1.0B specifications, allowing attacks to be mounted on security mode 1 through 3 (where 3 is the most secure mode). We also suggest counter-measures limiting the success of the discovered attacks. These measures are easily implementable – some in software on the application layer, others by relatively simple hardware modifications.

In the first type of attack, we show how an adversary can steal unit keys, link keys and encryption keys from victim devices of his choice. This, in turn, allows the adversary both to impersonate the parties and to eavesdrop on encrypted communication. This can be done either by exhaustively searching through PINs, or by mounting a middle-person attack. The former can be prevented by means of sufficiently long PINs (more than around 64 bits); the latter by means of public key mechanisms on the application layer, or by means of easily implemented security policies.

In the second type of attack, we show how an organization can map the physical whereabouts of users carrying Bluetooth-enabled devices by planting "Bluetooth detecting devices" at locations of interest. Even if the location itself may appear to be innocent, it may be undesirable for users if their whereabouts can be repeatedly correlated with the whereabouts of other users, which would indicate some relation between the users, given sufficient statistic material. In other cases, such as those involving stalkers, users would feel uncomfortable with their location being known, no matter *what* the location is. We note that while existing phones can be located in terms of what cell they are in, the precision is lower than what our attack would provide, and it is only cell towers and service providers that can determine the position. Moreover, it is impractical to attack existing systems by building a rogue network. On the other hand, our attack could allow virtually anybody to install a large number of listening nodes, thus allowing an attacker to determine the location of devices.

While it could be argued that this second attack needs a tremendous investment in terms of the infrastructure, we mean that this is not so. In order to derive useful information, it is sufficient for an attacker to place his eavesdropping devices at well chosen locations, such as airport gates (allowing him automatically to determine where people of interest travel). The information obtained could be correlated to user identities by means of side information, such as what can be obtained during a credit card transaction in which the payer carries a Bluetooth device. It may also be obtained by manual effort of the attacker (i.e., by determining the Bluetooth identities of all congressmen by walking around outside congress).

Furthermore, the attacker could leverage his attack off an *already existing* infrastructure, e.g., one that he legally builds for another – and socially more acceptable – purpose. If, for example, a company provides entertainment advice and directions in a city, and employs a vast grid of Bluetooth devices for this purpose, then the same infrastructure could be used for a second purpose without any additional cost.

Finally, our third type of attack is on the cipher and the use of the cipher. First, we show how an attacker can break the security of the cipher requiring 2^{100} bit operations. Then, we show another attack, with time and memory complexity of 2^{66}. While neither of these constitute a practical threat, it exposes a weakness in the cipher, which uses 128-bit keys. Second, we show how the use of the cipher trivially allows an attacker to obtain the XOR of plaintexts communicated between two devices. This is serious since an attacker may know one of the plaintexts already (e.g., by sending it to the phone, and waiting for the phone to transmit it to the headset), and will then be able to determine the other plaintext.

After detailing our attacks, we show how to prevent against them by performing only minor modifications to the Bluetooth specifications.

Outline: We begin by providing an overview of the ideal and actual functionality of Bluetooth (Section 2). This section also includes a brief overview of our attacks. Then, in Section 3, we describe relevant aspects of the standard in detail. In Section 4 we detail our attacks, and in Section 5 we discuss some counter-measures.

2 Overview

The Bluetooth protocol allows portable as well as stationary devices to communicate using short-range wireless methods, forming wireless local area networks of permanent or temporary nature. Let us first consider how these devices *ideally* should operate. First of all, we see that it is important for devices to be able to somehow address each other to ensure that the information goes to the appropriate device. To this end, some identifying information must be associated with each device, and this information must – in an ideal world – be unique[1] to the device in question to avoid "collisions". When one device wants to transmit some information to another device, the intended recipient should receive the message, but ideally, no other device should. (This relates to encryption of information, and is discussed in more detail below.) Furthermore, in an ideal world, no other device should even be able to determine the identity of the sender or the receiver of the information. (This relates to user privacy, and so-called traffic analysis.) More technically, each time two or more Bluetooth-enabled devices are to set up a communication link between each other, they need to generate and exchange one or more keys. These are later used to encrypt the information sent, if desired. In order to allow the participants to control who obtains what information (and the rights associated with the same) it may be that several such keys are exchanged by various groups of devices. It is important that the keys used for purposes of encryption are only known by the parties agreeing to communicate with each other, or attackers would be able to eavesdrop on the communication of honest users.

In order to conform to local jurisdictions, some restrictions are sometimes placed on the type of encryption used. While it is possible that local authorities may require that all communication can be decrypted by some escrow authorities, it is more common that they put bounds on the size of the key used for encryption purposes.

Turning now to the *actual* behavior of Bluetooth, we note that there are two modes of operation for Bluetooth-enabled devices. When a device operates in the first mode, the so-called *discoverable* mode, it responds to queries made by unknown devices, such as potential new piconet (e.g., Bluetooth LAN) group members. On the other hand, while in the second mode, the *non-discoverable* mode, a device only responds to devices with whom it has already set up communication. Furthermore, each device is given a unique identity when manufactured. It is anticipated that the first generation of devices will be able to communicate with other devices that are within an approximate radius of 10 meters (or 30 feet). The range of a second generation of devices is believed to be a tenfold.

When communication is initiated between two devices who have not yet been exposed to each other, they begin by negotiating a key which is later used for purposes of encryption. At the starting point of the key exchange protocol, each device only knows its own keys and other local data. After the termination of the key establishment protocol, the devices have agreed on a link key that they will later use when communicating with each other. Since the devices by definition do not share a cryptographic key until the end of the key exchange protocol, the payload of the packets sent in the course of the communication that takes place

[1] We note that it would, in principle, be possible for one device to use several different identities over time, and for two different devices to use the same identity at different times, while it must not be likely for two different devices to use the same identity at the same time. The uniqueness of identities is therefore *per point in time* and not *per device.* This distinction, however, is not made in the Bluetooth specifications.

during the key exchange protocol is sent in cleartext[2]. When two devices who previously have negotiated a key re-initiate communication after the conclusion of a previous session, they may set up a link key using either an old shared key, or (as when they meet for the first time) negotiate a new one. In the Bluetooth 1.0B specifications, all of the above mentioned keys are symmetric keys.

Before going into closer detail of the Bluetooth specifications and our attacks on the same, we will present a brief overview of our attacks. The first of these leverages on the fact that keys are essentially sent in the clear, the second uses the fact that all packets contain identifying information, and the third uses existing techniques to attack the cipher.

Eavesdropping and Impersonation. An example of a situation relevant to this attack is when a customer of a cyber café wishes to read email, access her files and possibly print them, using a Bluetooth-enabled laptop or PDA. Her computer would establish a connection to the local computer system and the available printer. An attacker who is able to eavesdrop on our user can therefore listen to the messages exchanged during pairing of the devices. Thus, if no application layer encryption is performed, or the attacker can perform a middle-person attack [6] on this layer, he can consequently obtain a copy of the document she accesses. By impersonating the user, the attacker could possibly alter the emails resp. the data to be printed, which could result in incorrect decisions being made by the user. In another situation, an attacker may try to eavesdrop on the voice data sent between a cell phone and a wireless headset. It is clear that it is not desirable for a system to allow an attacker to eavesdrop and impersonate on the physical layer, independently of whether the application layer introduces further security mechanisms.

Turning to the Bluetooth specifications, we note that these offer two possible ways to establish keys between two devices. A first protocol is used in situations when one of the devices involved in the key exchange has insufficient memory resources to run the second protocol; the second protocol is run if no device involved in the key exchange requests that the first protocol be used.

The objective of the first protocol is to keep down the number of keys stored by the device with limited memory resources. This is achieved by using the unit key[3] of this device as a link key[4] between the two devices. Thus, the other party will learn the unit key of the first party as a result of the key establishment

[2] While it in principle is possible to support public key cryptography on the application layer, and use this for the key establishment protocol on the physical layer, this is not advocated in the specifications. Furthermore, taking this approach still allows middle-person attacks [6] unless certification methods are employed. A related issue is the PIN, which is a form of shared key. If this is communicated out of band, i.e., verbally between users, then an attacker needs to obtain it by exhaustive search, which will succeed as long as short or moderately long PINs are employed.

[3] The unit key is the unique symmetric long-term private key of a device, and is stored in non-volatile memory.

[4] The link key can be described as a temporary symmetric key that is used for one or more sessions.

protocol. While this is the specified functionality, we note that it allows the second device to impersonate the first device at any future point. It also allows him to eavesdrop on all communication between the first device and other devices (including past communication, if recorded).

In the second protocol, the devices select a link key different from their unit keys. The key establishment involves several steps: First, the two devices choose an "initialization key" as a function of the address of one of the device identities, a PIN, and a random number. The length of the PIN code – which directly determines the security – can be chosen between 8 and 128 bits. Typically, it will consist of four decimal digits. The PIN can either be fixed or be arbitrarily selected and entered by the user through a user interface. If no PIN is available, zero is taken as a default value. The PIN and the random numbers are either communicated in the clear; out of band (e.g., entered by the users); or in an encrypted fashion (where the encryption and decryption take place in the application layer). In a second step, the devices each select a random number (different from the one chosen for the computation of the initialization key) and send these to each other, encrypted using the initialization key. In a final step, the devices compute the link key as a function of the two random numbers.

If an attacker can determine the initialization key, then he can also compute the link key. Moreover, because all encryption keys are generated from the link keys, once an attacker knows the link key, he can also decrypt encrypted information between the devices, and impersonate these to each other. If an attacker learns the unit key of a device – we will show how it can be done – then he will be able to impersonate this device in all aspects to any other device, and at any time.

Location and Correlation. For our second type of attack, assume that the attacker has Bluetooth-enabled devices distributed over a city or neighborhood of interest. He may either own these devices (that according to estimates will cost on the order of $10 per each) or he may lease or otherwise gain control over devices owned by others.

In a first attack, an attacker determines how a victim Bluetooth device within some area moves. Given timing information, the attacker can determine the correlation between different devices, i.e., determine who meets whom, and where. A first version of this attack is mounted from the application layer of a Bluetooth compliant device, and therefore uses standard Bluetooth devices without any need for hardware retrofit. The attacker attempts to initiate communication with all devices entering within the reach of the devices he controls. Once a device responds, it will give its identity, which is recorded by the attacker. Thus, the attacker will learn the identities of the victim devices in the vicinity of the devices he controls. The drawback of the attack is that it will only detect victim devices that are in *discoverable mode* – we elaborate on this later. However, this attack could be turned around to let the *victim device* attempt to initiate communication with nearby devices, and these – controlled by the adversary – report the identity of the victim device if the adversary could somehow control the victim device (e.g., by means of a virus or a corrupt website the victim has connected to). Moreover, it would have the advantage – to the attacker – that it would not require the victim to be in discoverable mode, as given control over the victim device would also allow the adversary to switch the victim's mode of operation. Also, it would potentially only require the attacker to control *one* device – the victim device – assuming this could be made to report the Bluetooth identities of responding devices, and that some of these are geographical fix-points with identities and locations known to the attacker. While it can be argued that if the attacker already controls a device, the security is already lost, this is not so, as being able to execute code on a device is not necessarily the same as knowing the device's location.

A second version of the location attack succeeds independently of whether victim devices respond to communication requests by strangers, and is simply based on the fact that two devices that have established their relationship and agreed to communicate will address each other when communicating, and this address can be intercepted by the adversary. An example of possible devices is a cellular phone and its wireless headset: When a phone call is received, the phone will transmit a message to the headset, setting up communication between the two. The two devices will then communicate on some pre-selected bands (according to the hopping sequence), and each message they send will have a channel identifier (or Channel Access Code, CAC) attached to it. The CAC is computed from the unique Bluetooth device identifier (the *Bluetooth device address*) of the master device. In our attack, the adversary determines the whereabouts of users by intercepting network traffic in his proximity, extracting the CAC, and using this to identify the master device of the piconet. We note that for this type of location attack to work, the attacker's devices must report information to the application layer not typically reported by Bluetooth devices, and so, the Bluetooth devices performing the attack must either be manufactured to perform the attack, or later modified to do so. This is an important restriction, as it rules out attacks in which proper Bluetooth devices under the control of improper software are used to mount the attack.

Linking Bluetooth identities to human identities. The device identifiers can be linked to the identities of their owners in several ways. One straightforward way presents itself in situations where a consumer identity is known – for example, during a credit card purchase or other identification – and where a Bluetooth device is present and active in the sense needed for the attack to work. However, it is not necessary to perform "certain matches", but it is sufficient that there is a match with some probability, allowing the attacker to infer the identity from several such "likely matches".

Cipher Vulnerabilities. In a third type of attack, we exhibit weaknesses of the cipher and of the use of the cipher. We pose a first attack on the cipher, allowing an attacker to break its security requiring 2^{100} bit operations and a mere 128 bits of known plaintext. Our attack works by guessing the contents of the three smaller LFSRs and the summation register and then determine the contents of the fourth LFSR by means of observing the output string. A second attack uses a known birthday-type attack to break the cipher in time and memory complexity 2^{66}. While these attacks are not of practical relevance, they exhibit vulnerabilities in the cipher that may allow for other and stronger attacks. Finally, we show how the attacker can trivially obtain the XOR of two plaintexts, merely by eavesdropping on the encrypted data. This is possible due to a reuse of the stream cipher output, causing an encryption of a plaintext using the other plaintext.

Remark: We note that some security claims within the Bluetooth community have relied to some extent on the unpredictability of the bandwidth hopping sequence to an outsider [9]. We show that this security assumption is incorrect.

3 Details of the Bluetooth Specification

In the following exposé, we present the details of the Bluetooth specifications that are relevant to our attacks. For simplicity, we refer to the page numbers of the document containing the official 1.0B specifications [7,8] for each piece of supporting information we present.

Device Modes. Devices may be in one out of two modes, the so-called *discoverable* and *non-discoverable* modes (see [8], pp. 29-31). When in the former, the device in question will respond to discovery inquiries ([8], p. 29). Furthermore, a device can either be in *connectable* or *non-connectable* mode (see [8] p. 39). When it is in connectable mode, then it will respond to messages it receives from "already discovered" devices ([7], pp. 99-112).

Addressing. Each device is associated with a unique identifier called the *Bluetooth device address* ([8], p. 25) which is used to establish all communication. If in connectable mode, the so-called device access code (DAC) is used to address the device. Moreover, for each point-to-point or point-to-multipoint communication a particular channel is used. We note that the channel identifier, the so-called channel access code (CAC) as well as the DAC are determined as a deterministic function of the master's unique Bluetooth device address ([7], pp. 143-147) and are always transmitted in the clear ([7], p. 159).

Establishment of Initialization Key. The following protocol is executed before the commencement of the link key generation protocol, and exchanges a temporary initialization key that will be used for encryption and decryption of information in the link key generation protocols. The protocol is as follows:

1. At first, one device chooses a random number and transmits it to the other device. Then, both Bluetooth devices compute an initialization key as a function of a shared PIN, the Bluetooth device address of the device that chose the random number, and the random number itself ([7], p. 153).

2. In order to confirm the success of the transaction (i.e., to confirm that both devices hold the same key), a mutual verification[5] is performed. This is based on a challenge response scheme in which a first unit chooses a random number and computes a function of the other device's Bluetooth address, the random number and the newly generated key ([7], p. 154). The chosen random number is transmitted to the other device, who computes the function on its Bluetooth address, the random number received, and the keys, and responds to the first device with the result of the computation. The first device verifies that the received value is the same value as it computed. Then, the roles are switched. The verification is deemed successful if the corresponding results in each round match.

Link Key Generation I. When one of the devices involved in the link key generation protocol has a shortage of memory, it requests that this first link key generation protocol is employed (see [7], p. 197 for the format of the request). The protocol ([7], pp. 153-155) is as follows:

1. The devices establish an initialization key using the above protocol.
2. The Bluetooth device with restricted memory capabilities encrypts its unit key using the initialization key. The resulting ciphertext is transmitted to the other device ([7], p. 155).
3. The receiving unit decrypts the received message using the initialization key, and uses the resulting key as a link key ([7], p. 155). The sender of the message uses his unit key as a link key – note that the two devices consequently use the same link key, as the plaintext the receiver obtains after decrypting the received ciphertext is the unit key of the sender.

Link Key Generation II. This second link key generation protocol is run when both devices have sufficient memory resources (see [7], p. 197 for the format of the request to use this protocol). The protocol (described on pp. 155-156 of [7]) is as follows:

1. The devices establish an initialization key using the previously detailed protocol.
2. Both devices, call these A and B, choose random numbers, $rand_A$ and $rand_B$ respectively. The device A (B) then computes the number LK_K_A (LK_K_B) as a function of $rand_A$ ($rand_B$) and its unique device address. (We refer to [7], p. 155 for the exact format of the computation, which, however, is not of importance to understand our attack.)
3. A and B encrypt their random numbers $rand_A$ and $rand_B$ using the initialization key. The resulting ciphertexts are exchanged.
4. Both units decrypt the received ciphertexts using the symmetric initialization key. Since both units know each others' unique device identifiers they can compute the other party's number LK_K_B (LK_K_A).
5. Both units compute the link key as $LK\_K_A \oplus LK\_K_B$.

[5] This step is called *authentication* in the Bluetooth 1.0B specifications.

6. A mutual verification is performed to confirm the success of the link key generation as in step 2 of the initialization key establishment protocol.

Cipher Use. Let A and B be two devices that have set up a link key, from which an encryption key is computed. The encryption key is (along with other data) used to seed the stream cipher (as described in [7], p. 163, fig. 14.6, and onwards). The output of the stream cipher is used to encrypt the plaintexts. Turning to figure 14.5 on page 162 of [7], we see that the stream K_c is XORed with plaintext $data_{A-B}$ in device A, to form a ciphertext which we will call $cipher_{A-B}$. This ciphertext is sent from A to B. Device B then decrypts $cipher_{A-B}$ by XORing the same stream K_c to it, obtaining $data_{A-B}$. Note that this output is fed to the second XOR gate in device B, and XORed with $data_{B-A}$. The result, let us call it $cipher_{B-A}$ is sent to device A, where it is further processed to obtain $data_{B-A}$.

4 Attacks

Eavesdropping and Impersonation. The basis of both key generation protocols is the protocol for establishment of the initialization key. This key is computed as a function of a PIN, a random number and the Bluetooth device address of the so-called claimant ([7], p. 153). If no PIN is available (in which case zero is taken as the default) or if it is transmitted in clear between the units, then the PIN is known to the attacker. If the PIN is communicated out of band (e.g., entered on each device by the user) then the attacker can still learn it by exhaustive search over all possible PINs, if weak or not sufficiently long PINs are used. This can be done as follows:

Offline PIN crunching. Let us first consider the setting where the attacker eavesdrops on two devices and wishes to determine what key they establish. We then consider a version in which the attacker starts the key exchange process with one victim device, determines what PIN this device used, and establishes a key with the victim device based on this “stolen” PIN.

1. *Case I: Eavesdropping.* The attacker exhaustively guesses all PINs up to a certain length. The adversary verifies the correctness of each guess plainly by performing the verification step of the initialization key protocol (i.e., the second step) based on his guess, and the random strings communicated in the clear (see [7], p. 195). If the result is correct then his guess is correct with an overwhelming probability. We note that the adversary is passive in that he only receives, and does not transmit.
2. *Case II: Stealing by participation.* The attacker first performs one PIN guess, and performs step 1 of the protocol for establishment of the initialization key. He then performs step 2 with the victim device. Let our attacker be the party that initiates the first round of the challenge - response protocol. (These are performed sequentially.) With an overwhelming probability, the

response verification will output 'correct' if and only if the victim device does not cheat, and the attacker has guessed the correct PIN. (Since the intention of the challenge - response protocol is to output 'correct' if and only if a given initialization key is consistent with the PIN and the random strings sent.) After obtaining the challenge - response transcript from the victim, the attacker computes the corresponding initialization key for each PIN he wishes to verify (according to the function used in step 1 of the protocol for establishment of the initialization key) and then (locally and without interaction) runs the verification algorithm on the computed initialization key and the obtained challenge - response transcript. If the verification algorithm outputs 'incorrect', then the attacker performs the verification computation on the keys corresponding to the next PIN he wishes to verify. This is repeated until the verification algorithm outputs 'correct', at which time the attacker has found the PIN used by the victim device. He then continues the key establishment protocol as before using the found key.

We note that the attack is performed off-line once the attacker obtains a challenge - response pair. Therefore, the back-off method employed to avoid PIN guessing does not add any security. In fact, the exponential back-off *benefits the attacker* as it gives him extra time to exhaustively search PINs.

Thus, the attacker can learn the symmetric initialization key for several common scenarii. Since the security of the subsequent steps of the key establishment rely on the secrecy of the initialization key ([7], p. 153), the attacker can decrypt the communication in this phase if he knows the initialization key. If the attacker obtains the initialization key, he will therefore also obtain the link key. Furthermore, since the encryption keys are computed from the link keys ([7], p. 156), he will be able to obtain these as well.

While the above attack extracts link and encryption keys, it is also possible for an attacker to obtain the unit key of a device (after which he can impersonate the device, and obtain the resulting link keys.) Namely, if a device has limited memory resources, it will request the use of the first key establishment protocol, in which its unit key is used as the link key ([7], p. 154). Consequently, an attacker will be able to obtain unit keys plainly by initiating communication with such a device and record what key this device proposes. It is also possible for an attacker to obtain this key merely by eavesdropping. By first obtaining the initialization key as above, merely by eavesdropping, he can then obtain the unit key as well.

We will now consider a third attack, in which an attacker might have already obtained the link key used by two devices, and where these two devices have completed the communication. Our attacker now contacts each one of them (posing to be the other) and sets up two new link keys[6]. This is therefore a middle-person attack [6]. The two devices will still believe that they talk to each other, and that the other one initiated the communication. The attacker will

[6] If the attacker has not obtained the previously used link key, he can pretend its loss and thus enforce the negotiation of an initial link key.

either make both of them slaves of their end of the communication, or both masters. (This is done in a protocol negotiating who is slave vs. master, and is executed right before the key establishment, see [7], p. 95, 123 and 1042.) The victim devices will therefore follow different hop sequences, since a device will follow the hop sequence based on the identity of the device he believes is the piconet master. Therefore, they will not see the messages they transmit for each other (since they are listening and transmitting in an unsynchronized manner) but only the messages the attacker chooses to send them. Consequently, the attacker is able to impersonate the two devices to each other.

Location Attacks. If a device is in discoverable mode ([7], p. 29-31) then it will respond to inquiries unless other baseband activity prohibits it ([7], p. 29). (To find each other, two or more devices scan the frequencies in some pseudo-random orders, and at different relative speeds, causing the slaves to eventually detect the master's signal and to respond with their respective identities. They then establish a frequency hopping sequence, which is a pseudo-random sequence whose seed is the master's clock and identity. (See [7], p. 43 and p. 127 for more details.)

When responding to an inquiry, a slave transmits its identity on the baseband ([7], p. 56 and p. 110). Therefore, an attacker can determine the location and movements of victim devices by maintaining geographically distributed devices that continuously inquire all devices entering within their reach, and recording the identities given in the responses. Since devices will use the same identities all the time ([7], p. 143), this allows the attacker to determine their movements. Given timing information, the attacker can quite simply establish what devices travel together for longer periods of time, or repeatedly meet.

Similarly, the attacker might (by means of corrupt software or websites) be able to induce the victim device to scan for devices to connect to, causing the victim device to reveal its identity to these devices. If we assume that the adversary has control over the victim device, it does not matter what mode the latter is in, given that this is switchable from the application layer.

Also regardless of whether a device is in discoverable mode or not, an attacker who is eavesdropping on the baseband can determine the CAC associated with each message he intercepts. Since the CAC is deterministically computed from the master unit's unique Bluetooth device address[7] he can then index victims by their CACs. Alternatively, he can determine the relationship between device identifiers and CACs using a database of pre-computed relations.

We note that several devices will map to the same CAC, since the CAC is computed only from 24 out of the relevant 32 bit Bluetooth device address of the master. However, this is not a big practical limitation to the attacker, since collisions between two randomly selected devices only occur with probability one over sixteen millions, making them very unlikely. Also, the attacker may have sales or other information that can narrow down the remaining possibilities. It

[7] Bit 39 to 62 of the CAC equal bit 1 to 24 of the Bluetooth device address ([7], p. 143-145).

is also likely that the attacker would be willing to tolerate some probability of misclassification as long as he is right most of the time.

Hopping Along. In order for an adversary to be able to follow a conversation within a piconet, he needs either to listen in to all the bands or follow the master and slaves on the frequencies on which they communicate.

In the U.S. and most other countries 79 bands have been assigned for use by Bluetooth devices, in Spain and France only 23 ([7], p. 43). Therefore, a simple device consisting of 79 (23) "listeners" in parallel can easily be built, and scan all bands.

In order to follow the communication using a *single* Bluetooth device, the attacker needs to establish what seed is used for the pseudo-random hopping sequence. For devices in the inquiry substate (page substate), the seed is deterministically derived from the inquiring device's own clock and the general inquiry access code[8] (an estimate of the paged device's clock and its DAC) whereas in the connection substate, the seed is determined by the clock and Bluetooth device address of the master ([7], pp. 127-138). For inquiry, only 32 dedicated hop frequencies are used. By responding to an inquiry, a device reveals its clock as well as its Bluetooth device address. Thus, the attacker can determine the seed for the paging hopping sequence by scanning through the inquiry frequencies and eavesdropping on the response messages. Subsequently, he can derive the seed for the hopping sequence of the piconet as the master will reveal his identity and clock during paging.

A Combined Attack. If an attacker first obtains the unit or link keys of a device, and later can pinpoint its position, it can also eavesdrop on its communication in a very effective manner. (In jurisdictions where only weak encryption is permitted, or no encryption at all, then the attack could be performed without knowledge of the keys.)

More specifically, the attacker would determine the device identifier and clock of his targeted victim, which we assume is a master device. From this, he can obtain the hopping sequence. By intercepting traffic on the corresponding bands, the attacker can obtain large portions of the communication, if not all. If the victim device moves out of reach of one attacker device, then nearby attacker devices would search for its appearance.

Cipher Attacks. Let us start by our attack on the cipher. An attacker can guess the content of the registers of the three smaller LFSRs and the summation register with a probability of 2^{-93}, given the sizes of these registers. He then computes the contents of the 39-bit register by "reverse engineering" this from the outputs of the other LFSRs and the summation register. Finally, the attacker determines whether his guess is correct by comparing a string of the actual output to the generated output. (In total, this needs approximately 128 bits of ciphertext and known plaintext.) The reverse engineering and the verification

[8] The general inquiry access code (GIAC) is common for all devices.

takes approximately 2^7 bit operations, making the total complexity of the attack 2^{100}, which is less than the complexity of 2^{128} encryptions for a brute force attack. We note that the above attack only obtains the key used for one frame. However, since the key used for a frame is computed in the same way as the sequence itself, we could obtain the master key by applying the attack twice.

Another known attack against this kind of ciphers has previously been described by Golic [3]. In a precomputation phase, an attacker randomly selects N internal states of the cipher, and computes the corresponding output key stream. These N key streams are sorted and stored in a database. Then M bits of the actual keystream are observed. If $M * N > 2^{132}$ then one expects to see a collision between the actual keystream and a keystream in the database. By choosing $M = N = 2^{66}$, this shows that the cipher can be broken with time and memory complexity 2^{66}.

Turning to our attacks on the *use of* the cipher, it is clear from our previous description that $cipher_{B-A} = data_{A-B}\ XOR\ data_{B-A}$ (with some potential shifting of one of them due to clocking.) Therefore, an attacker eavesdropping on the encrypted data sent between the devices will learn this value without any further action. If he knows one of the plaintexts, or parts of this, he will be able to derive the other, or parts of this.

5 Counter-Measures to Our Attacks

It is important to note that the disclosed vulnerabilities can be avoided by relatively simple modifications, some of which we will review here (but without making any claims of these being the most suitable methods of avoiding the attacks).

PIN length. In order to avoid a situation in which an attacker is able to obtain the secret keys of victim devices, it is important to use sufficiently long and sufficiently random PINs. If users chose PINs uniformly at random, then 64 bit PINs appear to be secure. (We note that an attacker will not expend more effort to derive the keys than to target some other point of the system, such as the encryption scheme [4] or the cell phone-to-base station link.)

Protecting unit keys. In order to avoid that devices learn the unit key of devices (in the first key establishment protocol), the device with the low memory capabilities may use some large-enough set of keys, one for each device it communicates with, or may generate such keys by using its unit key as the input to a pseudo-random generator. (If the seed is also based on the Bluetooth device address of the other party, it can easily be recomputed every time it is needed, limiting the amount of necessary storage.)

Application layer security. One may use application layer key exchange and encryption methods to secure the communication, on top of the existing Bluetooth security measures. We note that if standard certificate-based methods are employed, it is possible to defend against middle-person attacks.

Policies protecting against middle-person attacks. Recall that our middle-person attack relies on convincing both devices to become masters, or both become slaves, in order to avoid jamming of the communication channel by the attacker. Therefore, certain aspects of the middle-person attack may be avoided by means of policies governing what device may take the role of master vs. slave, and under what circumstances.

Physical protection. Our attacks on the key exchange rely on the attacker being able to detect the signals transmitted by the victim devices. The use of a Faraday's cage (with the form factor of a metal coated plastic bag) may be useful to obtain security against this attack.

Pseudonyms against CAC location attacks. If two devices use different and random pseudonyms for each session, in lieu of the deterministically generated CACs, then it will not be possible for an attacker to perform the CAC location attack. For even finer granularity, one may change the CACs pseudorandomly from packet to packet, much like the hopping sequence is derived. The devices may determine what pseudonym or pseudonym seed to use at the time of their first key exchange, or at any subsequent initiation of communication. While this modification cannot be software based (as it has to be performed on the Bluetooth chip itself) it is hoped and anticipated not to require any major modifications of the design.

Cipher. The attacks against the cipher can be avoided by replacing the cipher, e.g., with AES [2], and not to use plaintexts to encrypt plaintexts.

Conclusion

We have exhibited three types of vulnerabilities in the current version of the Bluetooth specifications. While the designers of the standard have been aware of the existence of eavesdropping and impersonation attacks *per se*, the specifications do not seem to anticipate or be concerned with location attacks, nor the presented attacks against the cipher. We hope that our findings will raise the awareness of threats to Bluetooth and that future versions of the standard are modified to defend against our attacks. (We note with sadness that such modifications have not been made to the upcoming version 1.1 of the specifications.)

Acknowledgments

We wish to thank Daniel Bleichenbacher for important observations regarding the cipher strength, and Jan De Blauwe, Fabian Monrose and Bülent Yener for helpful discussions and feedback. We also thank Alexandra Boldyreva, Ari Juels and members of the Bluetooth Security SIG for helpful suggestions regarding the presentation of the results.

References

1. A. Colden: "Expansion of Wireless Technology Could Bring as Many Problems as Benefits", The Denver Post, August 14, 2000, `www.newsalert.com/bin/story?StoryId=CozDUWaicrfaTv0LsruXfu1m`
2. J. Daemen and V. Rijmen, `http://csrc.nist.gov/encryption/aes/`
3. J. Dj. Golić: "Cryptanalysis of Alleged A5 Stream Cipher", Proceedings of Eurocrypt '97, Springer LNCS 1233, 1997, pp. 239–255.
4. M. Hermelin and K. Nyberg, "Correlation Properties of the Bluetooth Combiner", Proceedings of ICISC '99, Springer LNCS 1787, 1999, pp. 17-29.
5. The Official Bluetooth SIG Website, `www.bluetooth.com`
6. "RSA Laboratories' Frequently Asked Questions About Today's Cryptography, Version 4.1", `www.rsasecurity.com/rsalabs/faq/`
7. "Specification of the Bluetooth System", Specification Volume 1, v.1.0B, December 1, 1999. See [10].
8. "Specification of the Bluetooth System", Specification Volume 2, v.1.0B, December 1, 1999. See [10].
9. "Bluetooth FAQ - Security", `www.bluetooth.com/bluetoothguide/faq/5.asp`, November 15, 2000.
10. `www.bluetooth.com/developer/specification/specification.asp`
11. M. Stoll, "Natel-Benützer im Visier der Staatsschützer", SonntagsZeitung Zürich, December 28, 1997. `www.sonntagszeitung.ch/1997/sz52/93419.HTM`
12. J.T. Vainio, "Bluetooth Security," Proceedings of Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, Seminar on Internetworking: Ad Hoc Networking, Spring 2000, `www.niksula.cs.hut.fi/~jiitv/bluesec.html`
13. L. Weinstein: "Cell Phones Become Instant Bugs!", The Risks Digest, Volume 20, Issue 53, August 10, 1999, `catless.ncl.ac.uk/Risks/20.53.html#subj1.1`