An Analysis of Bluetooth Security Vulnerabilities

Creighton T. Hager and Scott F. Midkiff
Bradley Department of Electrical and Computer Engineering
Virginia Polytechnic Institute and State University
Blacksburg, Virginia 24061 USA
{chager, midkiff}@vt.edu

***Abstract*—Bluetooth has been developed to provide mobile *ad hoc* connectivity between a wide range of portable and fixed devices. In this paper, the Bluetooth system is described with an emphasis on its security features and known vulnerabilities. Additional security vulnerabilities were discovered using a scheme called VERDICT. These vulnerabilities are compared to vulnerabilities found in the IEEE 802.11 wireless local area network standard with VERDICT. Vulnerabilities were found as a result of improper validation, exposure, and randomness. These vulnerabilities include device address validation, invalid states, and exposed keys.**

Keywords—Bluetooth; security; wireless personal area networks; vulnerability analysis

I. INTRODUCTION

Bluetooth is an emerging low-cost, simple, and low-power wireless personal area network specification, with standards development underway through the IEEE 802.15 working group. However, amongst other things, security concerns are slowing the mass adoption of Bluetooth and other wireless technologies. Concerns over privacy of wireless communications still ranks high as a deterrent to the adoption of wireless technology [1]. Therefore, by recognizing possible security weaknesses of a wireless system, such as Bluetooth or IEEE 802.11, proper implementation measures can be applied to increase user confidence in such systems.

Lough [2] asserts that all computer attacks are a combination of one or more of the following improper conditions: validation, exposure, randomness, and deallocation. Lough's resulting taxonomy is the Validation Exposure Randomness Deallocation Improper Conditions Taxonomy (VERDICT).

This paper identifies potential vulnerabilities in the authentication, encryption, and wireless communication schemes used in Bluetooth. The VERDICT methodology is used to determine security weaknesses in Bluetooth protocols, key generation, and network initialization. These weaknesses relate to the protocol itself or to potential weaknesses that can occur due to improper, but conformant, implementation.

Section II of this paper gives an overview of Bluetooth security features. Section III explains current known security vulnerabilities of Bluetooth. Section IV briefly outlines VERDICT and describes the potential vulnerabilities in Bluetooth using VERDICT. Section V compares Bluetooth vulnerabilities to those of the IEEE 802.11 wireless LAN standard. Finally, Section VI summarizes and presents conclusions.

This work is supported in part by an Integrative Graduate Education and Research Training (IGERT) grant from the National Science Foundation (award DGE-9987586).

II. BLUETOOTH

The Bluetooth protocol specification [3] is documented in two parts: Volume 1, Core, and Volume 2, Profiles. The Bluetooth Core part specifies components such as the radio, baseband, link manager, service discovery protocol, transport layer, and interoperability with different communication protocols. Here, we assume familiarity with the basic operation of Bluetooth and focus on its security features.

The Bluetooth specification includes security features at the link level. It supports unidirectional or mutual authentication and encryption. These features are based on a secret link key that is shared by a pair of devices. To generate this key a pairing procedure is used when the two devices communicate for the first time.

Bluetooth devices transmit on the license-exempt 2.4-GHz ISM band using FHSS with 1600 hops per second. With respect to security, this reduces "casual eavesdropping" by allowing only devices synchronized with a piconet's master to be able to communicate in the piconet.

Each Bluetooth device has a unique address, allowing users to have some trust in the identity of the device at the other end of the transmission. For Bluetooth devices to communicate, an initialization process uses a Personal Identification Number (PIN). While some devices allow users to manually enter a PIN, the PIN can also be stored in non-volatile memory in the device.

A. Link Level Security Entities

The Baseband specification states that four different entities are used for maintaining security at the link layer: a unique public address, a secret key for authentication, a secret key for encryption, and a random number that is different for each new transaction [3].

Each Bluetooth device possesses a unique identification number, its 48-bit Bluetooth Address (BD_ADDR). This address can be used to identify which device is sending data since no other device will have that address.

A private 128-bit authentication key is used in the authentication processes for each Bluetooth device. The device randomly generates this key.

Encryption in each Bluetooth device requires another private key. This key length is from 8 to 128 bits, depending on the level of security needed for the application.

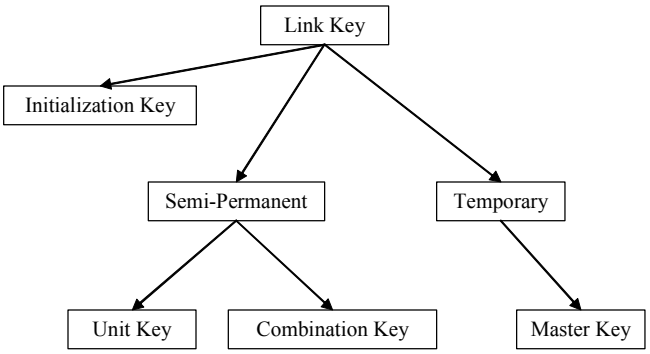

Figure 1. Key configuration.

Every Bluetooth device also has a 128-bit random number that can be regenerated to form a different random number at will by the Bluetooth device. The devices use this random number for various purposes in the encryption and authentication scheme.

B. Key Management

There are several different types of keys defined in Bluetooth. Depending on the type of application, link keys can be combination keys, unit keys, master keys or initialization keys, as illustrated in Fig. 1. In addition to link keys, there is the encryption key. These keys are described below.

1) Link Key: The link key is a 128-bit random number, which is shared between two or more devices and is the base for all security transactions between these devices. The link key itself is used in the authentication routine. Moreover, the link key is used as one of the parameters when the encryption key is derived. The link keys are either semi-permanent or temporary. A semi-permanent link key is stored in non-volatile memory and may be used after the current session is terminated. Consequently, once a semi-permanent link key is defined, it may be used in the authentication of several subsequent connections between the Bluetooth devices sharing it. The lifetime of a temporary link key is limited by the lifetime of the current session; it cannot be reused in a later session.

2) Initialization Key: The initialization key is used as the link key during the initialization process when there are not yet any unit keys or combination keys. It is used only during the installation. The initialization key is needed when two devices with no prior engagements need to communicate. During the initialization process, the PIN code is entered to both devices. The initialization key itself is generated by the E_{22} algorithm, which uses the PIN code, the Bluetooth Device Address of a device and a 128-bit random number generated by the other device as inputs. The resulting 128-bit initialization key is used for key exchange during the generation of a link key. After the key exchange, the initialization key is discarded.

3) Unit Key: The unit key is generated in a single device when it is installed. The unit key is made with the E_{21} key-generating algorithm when the Bluetooth device is in operation for the first time. After it has been created, it is stored in the non-volatile memory of the device and is rarely changed. One device can use another device's unit key as a link key between the two devices. During the initialization process, the application decides which party should provide its unit key as the link key. If one of the devices has limited memory, i.e., it cannot remember any extra keys, its link key is used.

4) Combination Key: The combination key is derived from information from two devices and it is generated for each new pair of Bluetooth devices. The combination key is created during the initialization process. Both devices generate the combination key at the same time. First, both of the units generate a random number. With the E_{21} key-generating algorithm, both devices generate a key, combining the random number and their Bluetooth device addresses. Subsequently, the devices securely exchange their random numbers and calculate the combination key to be used between them.

5) Master Key: The master key is a temporary key that replaces the current link key. It can be used when the master unit wants to transmit information to more than one recipient. The master device generates the master key using the E_{22} key-generating algorithm with two 128-bit random numbers. As all link keys are 128 bits in length, the output of the E_{22} algorithm is, also, 128 bits. The reason for using the key-generating algorithm in the first place is to make sure that the resulting random number is sufficiently random. A third random number is then transmitted to the slave and, using the key generating algorithm and the current link key, both the master and the slave compute an overlay. The new link key (the master key) is then sent to the slave and bitwise exclusive-OR'ed with the overlay. With this, the slave can calculate the master key. This procedure must be performed with each slave with whom the master wants to use the master key.

6) Encryption Key: The encryption key is generated from the current link key, a 96-bit Ciphering Offset Number (COF), and a 128-bit random number. The COF is based on the Authenticated Ciphering Offset (ACO), which is generated during the authentication process. When the Link Manager (LM) activates encryption, the encryption key is generated. It is automatically changed every time the Bluetooth device enters encryption mode.

C. Encryption

The Bluetooth encryption scheme encrypts the payloads of the packets. This is done with a stream cipher E_0, which is re-synchronized for every payload. The E_0 stream cipher consists of the payload key generator, the key stream generator, and the encryption/decryption part.

The payload key generator combines the input bits in an appropriate order and shifts them to the four linear feedback shift registers (LSFR) of the key stream generator. The key stream bits are generated by a method derived from Massey and Rueppel's summation stream cipher generator [4].

Depending on whether a device uses a semi-permanent link key or a master key, there are several encryption modes available. If a unit key or a combination key is used, broadcast traffic is not encrypted because the key is typically

used for more than one session and is potentially less secure. Individually addressed traffic can be either encrypted or not. If a master key is used, there are three possible modes. In encryption mode 1, nothing is encrypted. In encryption mode 2, broadcast traffic is not encrypted, but the individually addressed traffic is encrypted with the master key. And in encryption mode 3, all traffic is encrypted with the master key.

Since the encryption key size can vary from 8 bits to 128 bits, the size of the encryption key used between two devices must be negotiated. In each device, there is a parameter defining the maximum allowed key length. In the key size negotiation, the master sends its suggestion for the encryption key size to the slave. The slave can either accept and acknowledge it, or send another suggestion. This is continued until a consensus is reached or one of the devices aborts the negotiation. The abortion of the negotiation is done by the application in use. Every application has a minimum acceptable key size, and if the requirement is not met by either of the participants, the application aborts the negotiation and encryption cannot be used. This is necessary to avoid the situation where a malicious device forces the quality of the encryption to be low in order to do some harm.

The encryption algorithm uses four LFSRs of lengths 25, 31, 33 and 39, with a total length of 128. The initial 128-bit value of the four LFSRs is derived from the key stream generator itself using the encryption key, a 128-bit random number, the Bluetooth device address of the device, and the 26-bit value of the master clock. The feedback polynomials used by the LFSRs are all primitive, with Hamming weight of 5. The polynomials used are (25, 20, 12, 8, 0), (31, 24, 16, 12, 0), (33, 28, 24, 4, 0), and (39, 36, 28, 4, 0). Information on the fundamentals of LFSRs is found in [4].

D. Authentication

The Bluetooth authentication scheme uses a challenge-response strategy, where a two-move protocol is used to check if the other party knows the secret key. The scheme is illustrated in Fig. 2. The protocol uses symmetric keys, so a successful authentication is based on the fact that both participants share the same key. As a side product, the Authenticated Ciphering Offset (ACO) is computed and stored in both devices and is used later for cipher key generation.

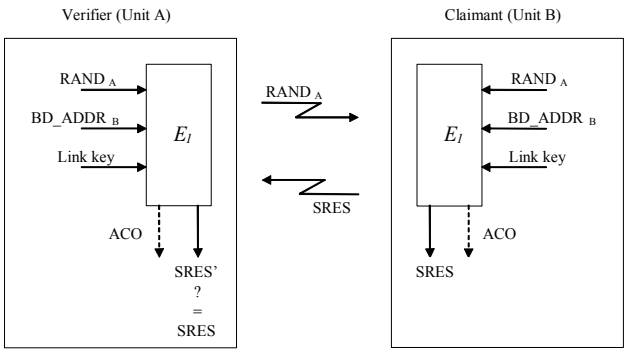

Figure 2. Challenge response mechanism (adapted from [3]).

To authenticate, the verifier first challenges the claimant with a random number. Then, both participants use the E_1 authentication function with the random number, the claimant's Bluetooth Device Address, and the current link key to generate a response. The claimant sends the response to the verifier, who then verifies that the response from the claimant matches the response generated by the verifier.

The application in use indicates who is to be authenticated. So the verifier may not necessarily be the master. Some applications require only one-way authentication, so that only one party is authenticated. This is not always the case, as there could be a mutual authentication, where both parties are authenticated in turn.

If the authentication fails, there is a period of time that must pass until a new attempt at authentication can be made. The period of time doubles for each subsequent failed attempt from the same address, until the maximum waiting time is reached. The waiting time decreases exponentially to a minimum when no failed authentication attempts are made during a time period.

III. Known Vulnerabilities

The Bluetooth Security Architecture, though relatively secure, is not without weaknesses. This section describes previously identified security flaws in Bluetooth's protocol architecture and implementation.

A. Spoofing Through Keys

A simple attack, though not so simple to implement in practice, is a "man-in-the-middle attack" to steal identification and encryption keys before the start of a session [5]. The identification and keys can then be used to impersonate and/or eavesdrop on communications. This problem is, of course, not specific to Bluetooth as most key exchange systems are prone to this type of attack. One way to mitigate such an attack in all systems is to include support for a digital certificate-based authentication system. Another approach, applicable to Bluetooth, is to make it difficult for an attacker to lock onto the frequency used for communication. Making the frequency hopping intervals and patterns reasonably unpredictable might help to prevent an attacker from locking onto the device's signal.

B. Spoofing Through a Bluetooth Address

Each Bluetooth device has a unique address, allowing users to have some trust in the identity of the device at the other end of the transmission. Once this device ID is associated with a user of the device, an intruding device can change its address to match the device address of the user. The intruder can then impersonate the user with the spoofed address.

C. PIN Length

Another vulnerability relates to the PIN itself [3]. Most devices have extremely short (usually 4 decimal digits) PINs. This is itself is a security weakness, though it is a property of the implementation and not the specification. Attackers can exhaustively search through the set of short PINs to determine a key.

IV. SECURITY ANALYSIS OF BLUETOOTH

This section presents an application of VERDICT to Bluetooth. We first provide a brief description of VERDICT and then apply VERDICT to the analysis of Bluetooth. This analysis shows that Bluetooth has vulnerabilities relating to improper validation, exposure, and the potential for improper randomness. We also compare the known Bluetooth vulnerabilities, described in Section III, with the ones found through the application of VERDICT.

A. VERDICT

Lough presents a comprehensive analysis of the types of attacks that are being directed at computer systems and a general taxonomy and methodology called VERDICT that explicitly considers the wireless environment [2]. VERDICT consists of four characteristics: validation, exposure, randomness, and deallocation. Vulnerabilities can be the result of one or more of the four characteristics.

For proper validation of code, all critical conditions and inputs must be checked to guarantee that no buffers overflow to insure reliable operation of programs. Protocol validation includes verification of inputs, messages, and state machines. According to Lough [2], "validation is the most critical aspect of determining if a system or program is secure."

The exposure characteristic reveals the domains or devices that are in contact with each other. Improperly exposed devices may allow intruders to enter a network from a safe distance.

Proper randomness is an important characteristic, especially for computer or protocol security. Random numbers are used in various keys and schemes to establish security. The numbers should be derived from proper random seeds [6]. In addition, seeds should be derived from natural random numbers.

Proper deallocation is the removal of residuals from access, composition, or data. Data residuals caused by improper deallocation result from old content in storage [2]. Composition residuals yield knowledge about how deallocated cells relate to each other. Access residuals are the result from improper deallocation of pointers, yielding dangling references.

With these algorithms and methodologies, one can determine the weaknesses of a system with respect to security. These methodologies allow one to post-analyze attacks and to predict weaknesses and vulnerabilities of future protocols.

B. Application of VERDICT

This section presents the application of VERDICT to analyze Bluetooth security. It is shown below that Bluetooth has vulnerabilities relating to improper validation, exposure, and the potential for improper randomness. As with most systems, most of the vulnerabilities are due to improper validation.

1) Improper Validation: The majority of all security vulnerabilities, in both wired and wireless networks, are caused by improper validation. The vulnerabilities due to improper validation identified for Bluetooth are described below.

a) Device Address Validation: The 48-bit Bluetooth device address needs to be adequately validated. Its in the same format as an IEEE 802.3 address and similar to an Ethernet address. The Bluetooth address can indicate if the address assignment is global or individual. No two addresses should be equivalent. However, if a user is able to change the address, an individual address could be non-unique [7].

While a Bluetooth address is similar in format to an Ethernet address, it is also similar with respect to security. Since, there is no validation of addresses, these addresses can be spoofed. This is similar to IP address spoofing.

b) Invalid States (Link Control): There are two major states used in the Bluetooth link controller: STANDBY and CONNECTION. In addition, there are seven substates, page, page scan, inquiry, inquiry scan, master response, slave response, and inquiry response. The substates are interim states that are used to add new slaves to a piconet. The major states and substates and their relations are shown in Fig. 3. To move from one state to the other, a device uses either commands from the Bluetooth link manager or internal signals in the link controller, such as the trigger signal from the correlator and the timeout signals [3].

One bit can represent the two major states and three bits can represent the seven substates. Designers must ensure that the invalid eighth substate, associated with the unused combination of three bits, is never reached. If it is reached by some unforeseen circumstance, there must be a way in the state machine to transition to an appropriate valid state. Without this transition, the device could become unstable or behave incorrectly in some other way.

c) Invalid States (Encryption Modes): If a slave device has received a master key, as described in Section II.B, there are three possible combinations for encryption [3], as listed in Table I. In this case, all units in the piconet use a common link key, specifically the master key. A specific Link Manager command is required to activate encryption for both broadcast and individually addressed unicast traffic.

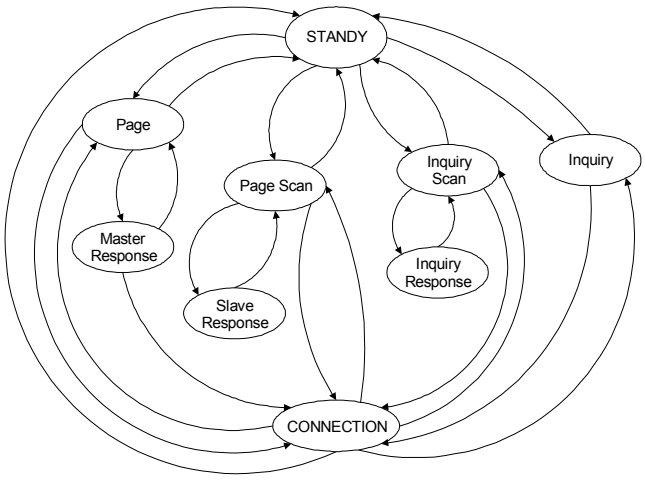

Figure 3. State diagram of link controller (adapted from [3]).

TABLE I. POSSIBLE ENCRYPTION MODES FOR A SLAVE IN POSSESSION OF A MASTER KEY (ADAPTED FROM [3])

State Number	Broadcast Traffic	Unicast Traffic
1	No encryption	No encryption
2	No encryption	Encryption (master key)
3	Encryption (master key)	Encryption (master key)
4 (invalid state)	Encryption (master key)	No encryption

A design can use two bits to represent the four states or combinations of traffic encryption, allowing a maximum of four states. The invalid fourth state must not be reached because broadcast traffic would then be encrypted, but point-to-point traffic would not have encryption. This would allow a fraudulent master to request data from the slave that would be unencrypted.

d) Encryption Keys: The Bluetooth specification [3] states that the master cannot use different encryption keys for broadcast messages and individually addressed unicast traffic. The master may tell several slave devices to use a common link key and, hence, indirectly to also use a common encryption key and broadcast encrypted information. This reveals a minor weakness in the protocol, allowing an intruder to decipher and use only one link key and, therefore, one encryption key for intercepting information for all devices in a piconet.

Another minor weakness is presented in the use of encryption keys. This weakness is exposed when each Bluetooth device implementing the baseband specification needs a parameter defining the maximum allowed key length, i.e., the maximum number of octets in the key. For each application, a number is defined indicating the smallest acceptable key size for that particular application. Before generating the encryption key, the involved devices must negotiate the key size [3]. Users should be aware of the minimum key size for their applications. Otherwise, a deceptive device could enforce a weak protection on a link by claiming a small maximum key size.

e) Link Keys: There is also a problem in the unit key scheme. Authentication and encryption are based on the assumption that the link key is the shared secret between participants. All other information used in the procedures is public. As an example, consider the process of spoofing a device with a link key as illustrated in Fig. 4. Suppose that devices A and B use A's unit key as their link key (Step 1 in Fig. 4). At the same time or later, device C may communicate with device A and use device A's unit key as the link key (Step 2). This means that device B, having obtained device A's unit key earlier, can use the unit key with a faked device address to calculate the encryption key and, therefore, listen to the traffic between devices A and C (Step 3). Device B can also authenticate itself to device A as device C and to device C as device A.

2) Improper Exposure: There are two instances of improper exposure in the Bluetooth protocol. The first deals with the non-secret link key. The second concerns the switching of master and slave device roles. These are discussed below.

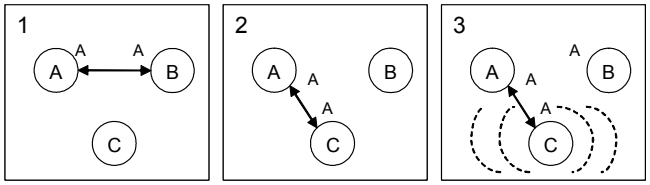

Figure 4. Spoofing through link key (adapted from [8]).

a) Non-Secret Link Key: As discussed in Section IV.B, the link key can be used to decrypt traffic between a device that has previously interacted with a fraudulent device and another Bluetooth device. As long as the fraudulent device is within range of the other Bluetooth devices, it can then listen to the traffic.

b) Master-Slave Switching: The second exposure vulnerability involves the switching of a master and a slave device. There are three occasions when a master-slave (MS) switch is desirable [3]. First, an MS switch is necessary when a unit paging the master of an existing piconet wants to join this piconet, since, by definition, the paging unit initially is master of a "small" piconet only involving the pager (master) and the paged (slave) unit. Second, an MS switch is needed when a slave in an existing piconet wants to set up a new piconet, involving itself as the master and the current piconet master as the slave. The latter case implies a double role of the original piconet master; it becomes a slave in the new piconet while still maintaining the original piconet as master. Third, a more complicated example is when a slave wants to fully take over an existing piconet, i.e., the switch also involves transfer of other slaves of the existing piconet to the new piconet. Clearly, this can be achieved by letting the new master setup a completely new piconet through the conventional paging scheme. However, that would require individual paging of the old slaves, and, thus, take an unnecessarily long time. Instead, letting the new master utilize the timing knowledge of the old master is more efficient. As a consequence of the MS switch, the slaves in the piconet have to be transferred to the new piconet, changing their timing and their hopping scheme.

Prior to the MS switch, encryption, if present, must be stopped in the old piconet. This presents a serious vulnerability to any devices still sending information to the master of the existing piconet. An intruder device could pose as a device requesting a take over of the existing piconet to create and manage its own piconet. While the master is transferring timing information to the device, the intruder could listen in on the now unencrypted messages of the old piconet.

One step in preventing such an attack is to disable the Allow_Role_Switch parameter in the HCI_Create_Connection link control command [3]. However, this is only a temporary solution as more support needs to be provided at the higher layers of the Bluetooth specification in terms of security and information transfer during an MS switch.

3) Improper Randomness: Within Bluetooth, the requirements placed on the random numbers used are that they be nonrepeating and randomly generated [3]. Hardware implementers must ensure that random seeds are generated from a good pseudo-random algorithm or, if possible, are a

natural random number. Unfortunately, the generated random number is usually transferred over the air interface and is not necessarily validated (see Section IV.B).

The PIN is used, both directly and indirectly, in the derivation of the Bluetooth security keys (see Section II.B). The PIN can be a fixed number embedded within the Bluetooth device. Alternatively, the PIN can be selected arbitrarily by the user, and then entered in both communicating devices that have to match at the two devices. Entering a PIN in both units is more secure than using a fixed PIN in one of the units and, thus, should be used whenever possible. If no PIN is available, a default value of zero is to be used [3].

For many applications the PIN code will be a relatively short string of numbers. Typically, it may consist of only four decimal digits. If the PIN is small or, even worse, has the value zero, then an exhaustive search can derive the initialization key (see Section II.B).

4) Improper Deallocation (Residuals): Improper deallocation might allow an intruder to use residuals from access, composition, or data to restore private information. This can include retrieval of device addresses or packet data. Residuals due to improper deallocation in Bluetooth are described below.

a) Deallocation Following a Master-Slave Switch: The MS switch vulnerability, as described earlier, falls into VERDICT's category of improper deallocation as well as the improper exposure category. The master device should make sure that all communication is halted in its piconet before a switch is made, otherwise unencrypted information could be appropriated.

b) Authentication After Encryption: Another security problem related to deallocation involves authentication after encryption. The HCI_Authentication_Requested command is used to try to authenticate a remote device associated with a specified Connection Handle [3]. The master or slave device must not issue the Authentication_Requested command with a Connection_Handle corresponding to an encrypted link. On an authentication failure, the Host Controller or Link Manager should not automatically detach the link. The Host is responsible for issuing a Disconnect command to terminate the link if the action is appropriate. If this is not done, then an intruder need not authenticate after spoofing an encrypted link, thereby making intrusion easier.

C. Comparison to Known Vulnerabilities

As described in Section III, there are three vulnerabilities in the Bluetooth protocol that have been previously cited in the open literature: (i) spoofing through the Bluetooth device address [7], (ii) spoofing through the security keys [8], and (iii) usage of the PIN [3]. These vulnerabilities constitute a subset of the vulnerabilities found using VERDICT. Improper device address validation correlates to spoofing with the Bluetooth device address. Spoofing through the security keys can be associated with improper link key validation. The weakness of the PIN compares to improper randomness.

V. Bluetooth Versus IEEE 802.11 Security

Bluetooth and IEEE 802.11 are two well-known types of wireless networks. This section briefly reviews the vulnerabilities in IEEE 802.11 identified by Lough using VERDICT [2] and then discusses the similarities and differences between Bluetooth and IEEE 802.11 with respect to security vulnerabilities discovered using VERDICT.

A. IEEE 802.11 Vulnerabilities

This section describes vulnerabilities found using VERDICT and reported in [2].

1) Improper Validation: There are seven cases of improper validation identified by Lough [2].

- The MAC address of IEEE 802.11 is not properly validated, thus, address spoofing is possible.
- There are three states in the general IEEE 802.11 state machine that determine the relationships that a station has with others. The fourth state is an invalid state and can cause associations without authentication as well as instability.
- If a deauthentication message is coupled with a spoofed message, stations can have their connections maliciously cut.
- Many Request to Send (RTS) frames can be sent in a flood, thus tying up the medium and causing a denial of service.
- If fragmentation packets are spoofed and labeled incorrectly, a receiving machine's IP stack may crash trying to reconstruct the packet.
- If the sequence number of packets can be predicted, frames could be spoofed and the intruder could overwrite information at a receiving station.
- A sixteen-bit field in the management frame determines what type of cryptographic authentication is to be used. If the sixteen bit number is '0,' it is an "open system" with no validation.

2) Improper Exposure: Three potential vulnerabilities result from improper exposure.

- There is no validation to ensure that a mobile station is not associated with (i.e., exposed to) more than one access point.
- When a station seeks to establish an Independent Basic Service Set (IBSS), beacon and probe frames are sent out. There is no validation to ensure that the sender is a legitimate member of the group.
- An intruder listening for IEEE 802.11 networks can get some information about the network and may be able to join it from a safe distance.

3) Improper Randomness: There are two possibilities of improper randomness in IEEE 802.11.

- If the random numbers generated are deterministic, then they are derived from a seed. If the seed is found, a station could potentially be prevented from transmitting. An intruder can discover the pseudo-random time period for the binary exponential backoff algorithm and selectively jam the device.
- In IEEE 802.11a, which uses Orthogonal Frequency Division Modulation (OFDM), the data is scrambled to

eliminate potentially long strings of 1s and 0s. The initial state of the scrambler can be determined if it is not properly randomized.

4) Improper Deallocation (Residuals): VERDICT did not find any protocol vulnerabilities due to improper deallocation.

B. Comparison of Vulnerabilities

This section presents similarities and differences in vulnerabilities for IEEE 802.11 and Bluetooth.

1) Improper Validation: Most of the vulnerabilities of 802.11 and Bluetooth occur with improper validation. Both protocols have the potential for improper address validation, possible invalid states, and problems with authentication. The potential flood problem in 802.11 is not found in Bluetooth, as its high frequency hopping rate mitigates this type of attack. A fragmentation attack is also unlikely as Bluetooth employs the use of a segmentation and reassembly protocol.

2) Improper Exposure: Bluetooth and 802.11 both share a similar vulnerability in improper exposure due to the extended range of their wireless transmissions. Although Bluetooth typically transmits with a range of up to 10 m, a Class 1 Bluetooth device has a maximum power output of 100 mW and, thus, can transmit information up to 100 m. This range is comparable to that of 802.11 and an intruder could receive information at a safe distance. Bluetooth might even be considered worse than 802.11 in this aspect, as its range could be extended with relayed information through piconets and scatternets.

However, Bluetooth is less vulnerable than 802.11 in one instance of improper exposure. Bluetooth can implement mutual authentication, while 802.11 only uses unidirectional authentication; i.e., mobile devices authenticate themselves to the access point, but not vice versa.

3) Improper Randomness: Bluetooth has potential vulnerabilities that can be caused by improper randomness. With proper hardware implementation and maximizing the length and randomness of the PIN, these weaknesses can be resolved. However, this is not the case with IEEE 802.11a, which uses OFDM to alleviate inter-symbol interference. As a side affect, long strings of 0's or 1's might be created, so a scrambler is needed to randomize the data. If the scrambling polynomial is discovered from one of the systems (since they all require it to unscramble the data) then an intruder could gain access to the original data.

4) Improper Deallocation (Residuals): IEEE 802.11 does not seem to have any problems with residuals. However, there is a possibility of this vulnerability occurring in the Bluetooth protocol. As stated in Section 5.1.4, a master-slave switch might have a problem with unencrypted communication during the switch. This is a possibility if the slaves in the piconet are still attempting information transfer to the master (a residual address).

VI. CONCLUSIONS

This paper gives an overview of the Bluetooth specification, with an emphasis on security features, and applies VERDICT [2] to analyze potential security vulnerabilities in the specification and in the implementation of Bluetooth. Several vulnerabilities are found in the categories of improper validation, exposure, and randomness. These vulnerabilities include device address validation, invalid states, and exposed keys. If these errors are present in actual implementations of the Bluetooth protocol, intrusions are possible in Bluetooth scatternets and piconets. It is also possible that additional vulnerabilities exist in Bluetooth that VERDICT did not find. Future work will utilize an experimental test environment to confirm these security vulnerabilities and to investigate others.

Although Bluetooth provides some improvement over IEEE 802.11 with respect to security, it also inherited some of the weaknesses that are still present in most wireless systems today. To solve these problems, proper hardware implementation with strict configuration management needs to be enforced and some aspects of the Bluetooth protocol need to be redesigned. Network and application level solutions may also be applied.

REFERENCES

[1] Bluetooth Special Interest Group, "The Bluetooth Specification, v.1.1," Feb. 22, 2001. Available at http://www.bluetooth.com/dev/specifications.asp.

[2] J. Rice, "Collaborative production strategies for technological innovation and leadership in network industries: The case of Bluetooth," Queensland University of Technology, Australia, 2000.

[3] D. L. Lough, "A Taxonomy of Computer Attacks with Applications to Wireless Networks," Ph.D. Dissertation, Virginia Polytechnic Institute and State University, Blacksburg, VA, April 2001. Available at http://scholar.lib.vt.edu/theses/available/etd-04252001-234145/.

[4] B. Schneier, Applied Cryptography, 2nd ed., John Wiley & Sons, Inc., New York, NY, 1996.

[5] M. Jakobsson and S. Wetzel, "Security weaknesses in Bluetooth," *Cryptographers' Track at RSA (CT-RSA) Conf.*, San Francisco, CA, 2001.

[6] D. K. Gifford, "Natural random numbers," Massachusetts Institute of Technology, Technical Report MIT/LCS/TM-371, Massachusetts Institute of Technology, Aug. 1988.

[7] "Palowireless Bluetooth Research Center," http://www.palowireless.com/bluetooth/, 2001.

[8] J. T. Vainio, "Bluetooth security," *Proc. Helsinki University of Technology, Telecommunications Software and Multimedia Laboratory, Seminar on Internetworking: Ad Hoc Networking*, Spring 2000. Available at http://www.niksula.cs.hut.fi/~jiitv/bluesec.html.