Available online at www.elixirpublishers.com (Elixir International Journal)

Advanced Engineering Informatics

Elixir Adv. Engg. Info. 36 (2011) 3519-3523

ISSN: 2229-712X

Enhancing Bluetooth security for m-commerce transactions

Kevin Curran[1] and Shane Dempsey[2]
[1]Computing and Intelligent Systems, University of Ulster, UK
[2]Computer Applications Department, Dublin City University, Eire.

ARTICLE INFO
Article history:
Received: 18 May 2011;
Received in revised form:
8 July 2011;
Accepted: 18 July 2011;

Keywords
Bluetooth,
Denial-of-service,
RFCOMM,
TCS BIN.

ABSTRACT
Bluetooth is a short-range wireless cable replacement technology. The Bluetooth Security Architecture is susceptible to a number of attacks such as exhaustive search of short PINs, man-in-the middle attack and denial-of-service attacks. These attacks are made possible by weaknesses exploited in the PIN code, the system of key distribution and the handling of requests from malicious users. This paper also introduces some possible additional security measures which could be implemented to strengthen the overall security architecture of Bluetooth enabled devices.

Introduction

One issue which is slowing the mass adoption of wireless technology is security concerns. Concerns over privacy of wireless communications still ranks high as a deterrent to the adoption of wireless technology. Bluetooth employs data encryption and user authentication to protect wireless communications. Bluetooth devices use a combination of a Personal Identification Number (PIN) and a unique address to identify other Bluetooth devices. Data encryption is also used by the protocol to secure data being transmitted over the air. Bluetooth enabled devices have security features built into them in the link layer of the Bluetooth protocol stack. The Link layer is responsible for setting up the link between two devices and also controlling security services such as authentication and encryption. Bluetooth has a number of different security levels that can be associated with devices and services. When two devices connect for the first time they will setup a relationship. The relationship between the devices will be defined as trusted or untrusted. A trusted device has unrestricted access to all services while an untrusted device has restricted access to services. All Bluetooth devices will offer services. Some services may only require a low level of security i.e. exchanging virtual business cards while others services may require a higher level of security i.e. access to a confidential file. To facilitate this Bluetooth enables devices to operate in one of three different security modes:

- Security Mode 1: No security.
- Security Mode 2: Service-level enforced security.
- Security Mode 3: Link level enforced security.

Security mode 1 is appropriate when the communicating devices have no critical applications and hence no security is required. In security mode 2 it is possible to define different security levels for both devices and services. For devices, two levels of trust exist: trusted and untrusted. For services the requirements for authorisation, authentication and encryption are set independently. In security mode 3 provides link-level security, whereby the Link Manger enforces security at a common level for all applications at the time of connection setup. The security level of a service is defined by three attributes:

- Authorisation Required: Access is only granted automatically to trusted devices or untrusted devices after an authorization procedure. Authorisation always requires authentication to verify that the remote device is the right one.
- Authentication Required: Before connecting to the application, the remote device must be authenticated.
- Encryption Required: The link key must be used to provide encryption, before access to the service is allowed.

This information is stored in the service database of the security manager [5].

Key Management

Bluetooth uses symmetric key cryptography for the purposes of encryption, authentication and authorization. Symmetric key cryptography is an encryption system in which the sender and receiver of a message share a single, common key that is used to encrypt and decrypt a message. Public-key cryptography on the other hand utilizes two keys - a public key to encrypt messages and a private key to decrypt them. Symmetric-key systems are simpler and faster than public-key systems, but their drawback is that both parties must somehow exchange the key in a secure way.

Public-key encryption avoids this problem because the public key can be distributed in a non-secure way, and the private key is never transmitted. Bluetooth specifies five different types of keys :

- Combination key KAB: This key is generated by two units, A and B. It is unique for every pair of units.
- The initialization key, Kinit, is used as link key during the initialization process when no combination or unit keys have been defined and exchanged yet or when a link key has been lost.
- Unit Key KA: The unit key is generated in unit A only. It is used by units with limited memory capacity or units that need to be accessible to a large number of units. The difference between

Tele:
E-mail addresses: kj.curran@ulster.ac.uk,
sdemps.mect1@compapp.dcu.ie

this key and the initialisation is that it is created using a different key generation function.

- The temporary key Kmaster replaces the original link key temporarily when the master wishes to broadcast or transmit to more than one unit simultaneously using the same encryption key.
- The encryption key Kc : this is a 8-128 bit key, which is derived from the current link key and a random number.

A combination, unit, master or initialisation key can be used as a link key. The link key is a 128-bit random number which two units use to secure all transactions between each other.

Authentication

Bluetooth authentication is the process of verifying that the other device has the same encryption key before enabling encryption on the connection. Authentication in Bluetooth is performed using a challenge-response scheme. Two devices interacting in an authentication procedure are referred to as the Claimant and the Verifier. The Verifier is the Bluetooth device validating the identity of another device. The Claimant is the device attempting to prove its identity. The verifying unit A sends a random number to the claimant B. B computes a signed response (SRES) consisting of the link key that both A and B share, its unique BD_ADDR , and the received random number. A performs the same computation. B sends the SRES to A, which compares the received SRES to the one it computed itself. If they equal, authentication succeeds, otherwise authentication fails.

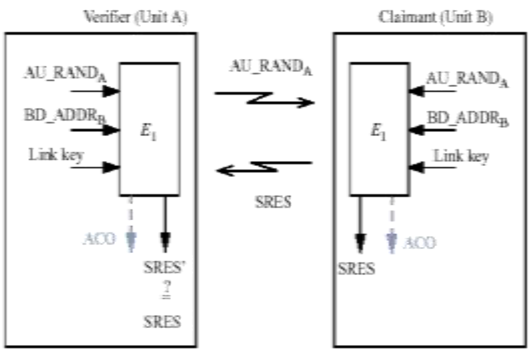

Figure 1: Challenge-response system used by authentication procedure

A successful calculation of the authentication response requires that two devices share a secret key. When two devices do not have a common link key an initialization key (K_{init}) is created based on a PIN and a random number. The procedure for creating the initialization key is called pairing. This procedure involves both devices calculating an initialization key based on a random number and the PINs for the devices. Once both devices have calculated K_{init} an authentication based on this key is done. The verifier checks the authentication response and if correct, the link key is created.

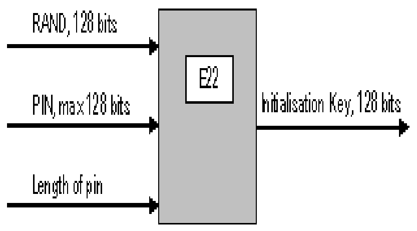

Figure 2: E22 Key generating algorithm for the initialization key

The authentication function is based on the SAFER+ algorithm. SAFER stands for Secure and Fast Encryption Routine, and the algorithms are based on iterated block ciphers, where the same cryptographic function is applied for a specified number of rounds.

Encryption

The encryption key is derived from the current link key. Each time encryption is needed the encryption key will be automatically changed. The Bluetooth encryption procedure is based on a stream cipher. As can be seen from Figure 3 the encryption function operates as follows. The keystream output is XORed with the plaintext bits and sent over the air to the target device. The keystream output is produced using a hash algorithm which is based on four 4 linear feedback shift registers . The inputs to the hash function are the master unit's address, the random number (EN_RAND), a slot number and an encryption key which initialises the LFSRs before the transmission of each packet [7]. The encryption key is created by the Key Generator. The Key Generator takes the current 128 bit link key, the COF (equivalent to ACO variable which is produced by the authentication function), and a random number as input.

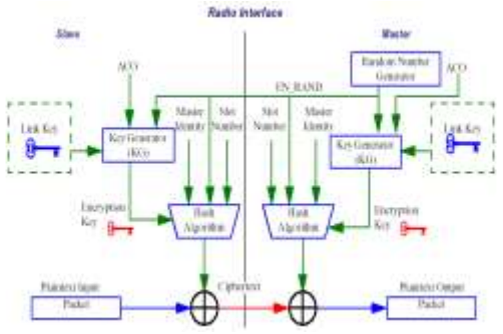

Figure 3: Encryption function

A divide-and-conquer attack on Bluetooth has been attempted [3]. However the attack cannot be successful because the attacker is never provided with a sufficient number of output bits. Bluetooth is designed to output blocks of at most 2746 bits but for the attack to work approximately 2^{64} output bits are required. The attack is based on exploiting weaknesses in the periodicity of the linear feedback registers used by the encryption system.

Security Manager

The security manager component is the entity that decides what policies are to be enforced when a connection request is made. Based on the service, device type and whether the device is trusted or untrusted the security manager can enforce application level authentication, encryption of the session or any other specific access policies.

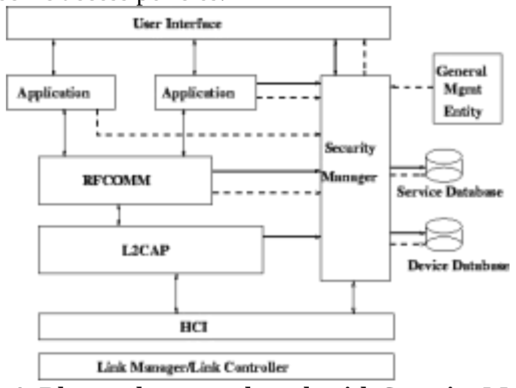

Figure 4: Bluetooth protocol stack with Security Manager

The Security manager needs information regarding devices as well as services before it can make a decision whether or not to allow access and if so, to what services. This information is stored in two databases namely, the Device Database and the Service Database. The Device database stores information about the device type, the trust level (whether trusted or untrusted) and about the link key length. The Service database stores information regarding the authentication, authorization and encryption requirements for the services. It also stores other routing information for the services. The Security manager is a central entity in managing and enforcing security policies. The Security manager connects application level security features and Bluetooth protocol level (Link Layer) security controls. Key Tasks handled by the Security manager are:

- Stores security related information for all services (Service Database).
- Stores security related information for available devices in range (Device Database).
- Processes access requests by protocol implementations or applications (grants access or denies connection).
- Enforces authentication and/or encryption before connection can be established;
- Initiating pairing and querying PIN entry by the user/application.

Depending on the security policy governing access, the security manager might call upon an application protocol to enforce application level security such as a username/password scheme for authentication.

Weaknesses in Bluetooth Security

The Bluetooth Security Architecture is susceptible to a number of attacks such as exhaustive search of short PINs, man-in-the middle attack and denial-of-service attacks. These attacks are made possible by weaknesses exploited in the PIN code, the system of key distribution and the handling of requests from malicious users.

PIN Weaknesses

The strength of the initialisation key is based purely on the used PIN code. If the user has to enter this code manually then the PIN must be short because of usability constraints. PIN numbers are usually only 4 digits in size which means there are only 10000 possible combinations. Combined with the fact that 50% of used PINs are “0000”, the trustworthiness of the initialisation key is very low.

Key Distribution Weaknesses

When a master unit wants to broadcast a message to a number of units in the piconet it will distribute a temporary master key to act as the link key. Since all units will then share a common key, there is nothing to prevent a unit from successfully impersonating another unit. There also exists a problem when two units use a unit keys as a link key. If a device A uses its unit key as the link key for communicating with other devices. Then any device which has the unit key (obtained from A) can use the unit key with a faked device address to calculate the encryption key and eavesdrop on traffic between A and any other device as long as they are using A’s unit key as the link key .

Denial-of-service attacks

The procedure for handling failed authentication attempts allows malicious parties to prevent legitimate users from gaining access to services. If a device repeatedly fails authentication then a record will be created by the verifier that the device is not to be trusted i.e. the device is blacklisted. If the attacker can fake the BD_ADDR identifiers of other units, then they can cause failed authentication attempts. The legitimate user can then no longer even attempt to authenticate themselves because the device offering the service has a record of them in its black list and knows not to trust it because it has made so many failed authentication attempts. This process of filling the verifiers black list with untrusted device addresses, lays the foundation for another type of attack. If the verifier’s available memory is filled up because of all the entries from devices that have repeatedly tried and failed to authenticate themselves then an attacker is free to try different keys without the threat of being blacklisted and denied the right to repeat authentication attempts.

MCommerce Transaction Support Weaknesses

The Bluetooth Protocol Architecture is an amalgamation of Bluetooth specific protocols and others already in use (see Figure 5). One of the reasons for this sort of mixed architecture is that it allows application specific security controls to be implemented that would be transparent to the lower layer security controls.

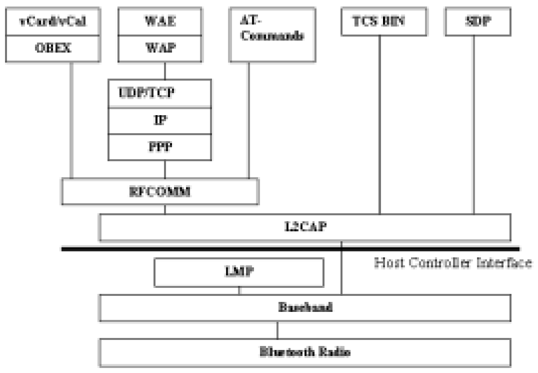

Figure 5: Bluetooth specific and application oriented protocols

RFCOMM, TCS BIN, AT Commands and the other adopted protocols such as OBEX, TCP/UDP/IP, PPP and WAE/WAP form the application oriented protocols that run over the Bluetooth specific protocols. Each of these protocols can enforce their own security policies. For example RFCOMM will enforce the Bluetooth security policy for cordless telephony and OBEX will enforce the Bluetooth security policy for file transfer and synchronisation. The Bluetooth protocol contains controls for authenticating another Bluetooth enabled device but not the end user. If this feature is needed it has to be accomplished with application level security. The most obvious and widely used authentication scheme is the username/password system or more commonly with wireless devices the entering of a PIN into the device. For usability reasons this key is short but application level functionality could extend this PIN internally to be up to 128 bits in size (discussed later).

Strengthening Bluetooth Security

The Bluetooth Security Architecture has been described earlier as being susceptible to a number of attacks such as exhaustive search of short PINs, man-in-the middle attack, denial-of-service attacks and mCommerce transaction support. We now describe solutions to overcome the weaknesses exploited in the PIN code, the system of key distribution, the handling of requests from malicious users and authentication issues.

PIN Weakness Problem

The PIN code of the device can be fixed, so that it needs to be entered only to the device wishing to connect. Another possibility is that the PIN code must be entered to both the devices during connection. To strengthen the security of the initialisation key a stronger and hence longer PIN needs to be used. An alternative to having the user manually enter a PIN is to use application level key agreement software with longer 128 bit codes. The PIN code would not need to be entered manually to each device but would be produced by an application program and exchanged with, for example, Diffie-Hellman key agreement. The Diffie-Hellman method for key agreement allows two hosts to create and share a secret key. The PIN stored by the Bluetooth device must then adhere to the rules for creating Diffie-Hellman parameters. The PIN must be a combination of two numbers :

1. a prime number, p (greater than 2).
2. a base, 'g', an integer that is smaller than p.

If the verifier and the claimant follow the procedure for exchanging keys using the Diffie-Hellman method they can both generate a common key value which could be used as the value for the PIN (see Appendix 1).

Key Distribution Problem

Both of the above attacks are based on shared secret keys. The security mechanism is based on the fact that users within a piconet can be trusted not to be malicious users which are a naïve assumption to make. To get around having to distribute shared secret keys, public key cryptography can be incorporated. One reason why Bluetooth choose not to use public key cryptography is because it uses more computational power than symmetric key cryptography, but elliptic curve cryptosystems may present a solution to this problem in the near future. Another reason for implementing symmetric key cryptography is that it does not require a trusted third party to distribute keys. To keep transmissions from breaking up in the heavily used ISM band used by Bluetooth, Bluetooth employs a fast frequency hopping scheme when transmitting data. The transmission frequency is changed 1600 times per second. Only synchronised devices can communicate so a way to guard against the man-in-the-middle form of attack would be to make it difficult for an attacker to lock onto the frequency used for communication. This could be done by making the frequency hopping intervals and patterns reasonable unpredictable. Another security measure which would be relatively easy to implement would be a capability to reduce the range of a Bluetooth device. If you only wanted the Bluetooth devices on your person to communicate i.e. your Bluetooth enabled PDA, laptop and phone then you would only require a range of approximately 1 meter. Any malicious third party would then find it difficult to get within range to do any damage.

Denial of Service Attack Problem

To solve this problem a security policy could be implemented at the application level to query the device database and to delete the oldest entry if there is no room to store additional addresses of devices which have failed authentication. Other policies could be implemented at an application level to provide statistical analysis to guard against the above forms of attack i.e. by measuring the frequency of incoming requests, if the device recognises that it is abnormally high then it could implement a blocking procedure to reject authentication requests for a period of time. Yet another attack against Bluetooth is the battery draining denial of service attack. Every Bluetooth device contains a random number generator which it used to provide input to the security functions. This generator is usually implemented with software. Bluetooth uses random numbers for contacting other devices and for authentication and encryption. Many systems tend to use badly designed random number generators, or use them in ways that make attacks easier. Random number generation must adhere to the requirements set by the Bluetooth specification i.e. that random numbers be non-repeating and that they are randomly generated (i.e. impossible to predict).

M-Commerce Transaction Support Problem

For MCommerce, transaction support would need to be provided for a public key infrastructure. In a public key infrastructure users are provided with unique public/private key pairs. A limitation in using PKI in a wireless environment is the need for a trusted third party who is responsible for issuing digital certificates.

Digital certificates contain the public/private key information required for authentication and encryption. Public key encryption can solve the authentication problems of the symmetric key infrastructure and can also enforce non-repudiation of transactions which is a vital feature for enabling MCommerce transactions. This infrastructure would be difficult to incorporate into an ad-hoc network but in the scenario where a dynamic node is communicating with a static node then a PKI system may work. The static node could be a fixed terminal which provides a service to dynamic nodes, for example a ticket vending machine in an airport terminal, but it would be Bluetooth enabled so any other Bluetooth device could communicate with it.

The service provider node (server) could then issue short term certificates to nodes requesting services (clients). The certificates would be sent daily from a certification authority to the server node. When a client node needs to interact with the server to purchase a ticket it would obtain a certificate from the server node and validate it.

The certificate would need to be validated against a certificate for a trusted authority stored on the client node. The reason for the short term certificate is to allow the certification authority to revoke services to the server node if it needs to, by ceasing to issue further short-lived certificates. Hence the clients will not consider the server authenticated. In this way, the client-server channel is secured. It is an extra layer of security to help prevent rogue devices from issuing certificates [6]. Alternatively one can use WAP over Bluetooth and utilise the PKI architecture implemented by WAP.

Conclusion

The Bluetooth protocol contains many security procedures that support the general nature of the usage models for Bluetooth enabled devices. The security provided by Bluetooth is probably adequate for most usage models i.e. connecting devices in a small ad hoc network, but for transferring sensitive information such as credit card details, Bluetooth would not be secure enough. For services that require increased security, added security functionality should be added to the higher layers of the Bluetooth protocol stack.

References

[1] f. Stajano and R. Anderson “The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks”. http://www.cl.cam.ac.uk/~fms27/duckling/,1999

[2] Catherina Candolin “Security Issues for Wearable Computing and Bluetooth technology” http://www.tml.hut.fi/~candolin/Publications/BT/btwearable.pdf

[3] M. Hermelin. “Cryptographic properties of the Bluetooth combination generator.” Master’s thesis, Helsinki University of Technology, 2000

[4] Marjaana Traskback "Security of Bluetooth: An overview of Bluetooth Security"
[5] Thomas Muller Bluetooth WHITE PAPER: Bluetooth Security Architecture, Version 1.0 15 July 1999. http://cnscenter.future.co.kr/resource/hot-topic/wpan/1c11600.pdf
[6] WTLS mini certificates and Public Key Infrastructure (PKI) http://www.rave-tech.com/WAP/certtext.html
[7] Wireless Security Perspectives Vol.3, No.1. January, 2001
[8] Specification of the Bluetooth System V1.0 B
[9]Juha T. Vainio. "Bluetooth Security". http://www.niksula.cs.hut.fi/~jiitv/bluesec.html

Appendix 1

The "Diffie-Hellman Method For Key Agreement" allows two hosts to create and share a secret key.

1) First the hosts must get the "Diffie-Hellman parameters". A prime number, 'p' (larger than and "base", 'g', an integer that is smaller than 'p'. They can either be hard coded or fetched from a server.

2) The hosts each secretly generate a private number called 'x', which is less than "p - 1".

3) The hosts next generate the public keys, 'y'. They are created with the function:

y = g^x % p

4) The two host now exchange the public keys ('y') and the exchanged numbers are converted into a secret key, 'z'.

z = y^x % p

'z' can now be used as the key for whatever encryption method used to transfer information between the two hosts. Mathematically, the two hosts should have generated the same value for 'z'.

z = (g^x % p)^x' % p = (g^x' % p)^x % p

All of these numbers are positve integers

x^y means: x is raised to the y power

x%y means: x is divided by y and the remainder is returned