Enhancements to Bluetooth Baseband Security

Christian Gehrmann, Kaisa Nyberg
Ericsson Mobile Communcations AB, Ericsson Research, Christian.Gehrmann@ecs.ericsson.se
Nokia Research Center, Kaisa.Nyberg@nokia.com

1. INTRODUCTION

Bluetooth system has been developed by Bluetooth Special Interest Group (Bluetooth SIG) as a cable replacement for short-range connectivity. In Bluetooth, special effort has been taken to develop and standardise adequate security mechanisms and procedures for protecting the wireless radio link. This set of mechanisms is defined in the Bluetooth Baseband specification [5] and is referred to as Bluetooth Baseband security. It is based on strong cryptographic algorithms and well-established security principles. Still, more work is required to integrate Bluetooth Baseband security into various applications that may have very different link layer security requirements. Bluetooth Baseband security is implemented in the Bluetooth module and is common to all Bluetooth units. Also the application specific security functionality may need to be standardised for interoperability.

Recently, Jakobsson and Wetzel identified some potential trap holes in Bluetooth security in [11]. Their main concerns were certain weak options included in the Bluetooth security standard. They also criticised the way Bluetooth units make themselves discoverable by other units just by broadcasting messages that include their unique identities in clear.

The purpose of this paper is to introduce some recent work in the area of Bluetooth Baseband security. Specifically, we address the problems discovered by Jakobsson and Wetzel, and develop possible counter measures. First, a brief introduction to Bluetooth Baseband security is given. Then major security shortcomings are identified. These include usage of the unit key and the short Bluetooth PIN value in the initialisation procedure, on the one hand, and the privacy problem created by location tracking, on the other hand. In section 4, we discuss passkey-based methods of exchanging the Bluetooth link key. Using public key cryptography adequate security can be provided while keeping the passkey short for the user's convenience. Furthermore, in section 5, an application to LAN access is developed enabling access point roaming. Finally, in section 6, we describe a technique that offers reasonable protection against location tracking. Identities of Bluetooth units can be efficiently hidden from unauthorised units using temporary Bluetooth device addresses.

2. THE BLUETOOTH SECURITY CONCEPT

In this section we give an overview of the Bluetooth security concept. We describe the parts relevant for the issues discussed in this paper. Readers who would like to have a detailed description of the Bluetooth Baseband security features are recommended to read the specification [5] or the overview by Persson and Smeets in [15].

2.1 Security modes

Different Bluetooth applications are described in the terms of *profiles.* Some profiles are dedicated to a specific application while others are general profiles utilised by several other profiles. The Generic Access Profile (GAP) [6] defines the generic procedure related to the discovery of Bluetooth devices and the link management aspects of connecting to Bluetooth devices. The GAP also defines the different basic security procedures of a Bluetooth device. A connectable device can operate in three different security *modes:*

Security mode 1: A Bluetooth unit in security mode 1 never initiates any security procedures, i.e., it never demands authentication or encryption of the Bluetooth link.

Security mode 2: When a Bluetooth unit is operating in security mode 2, it shall never initiate any security procedures, i.e., demand authentication or encryption of the Bluetooth link, at link establishment. Instead, security is enforced at channel or connection establishment.

Security mode 3: When a Bluetooth unit is in security mode 3, it shall initiate security procedures before the link set-up is completed. Two different security policies are possible: always demand authentication or always demand both authentication *and* encryption.

Service level access control can be provided using both security mode 3 and security mode 2. Security mode 2 gives better flexibility. When using security mode 2, no security is enforced at channel or connection request. Thus it is possible to allow access to some services without any authentication or encryption and a unit can be totally open to some services, while still restricting access to other services.

2.2 Bluetooth keys and pairing procedure

The security concept includes several kinds of keys. The keys have different purposes and are used either for key exchange, authentication or encryption. The basic idea in the security concept is that trust between *devices* is created at a *pairing* procedure. A pairing is performed between two Bluetooth units. The purpose of a pairing is to create a common shared secret between two units. Below we explain the different key types and the pairing procedure in Bluetooth.

2.2.1 Key types

The common shared secret is called a *link key.* All paired devices, i.e., mutually trusted devices, share a common link key. There are two types of link keys defined: *unit keys* and *combination keys.*

A Bluetooth unit with restricted memory resources might use a unit key. A unit uses the same unit key for *all* its connections. During the pairing procedure the unit key is transferred (encrypted) to the other unit. No pairing is possible between two units that *both* would like to use a unit key. In section 3.1 we discuss unit key drawbacks.

A combination key is a key that is unique to a particular *pair* of devices. The combination key is only used to protect the communication between these two devices. The combination key is calculated during the pairing procedure.

Since a link key (unit or combination key) is used to protect the wireless link between two Bluetooth devices, each unit needs to store the link key it is supposed to use when communicating with a unit that it has been paired with. Hence, each unit needs to keep a link key database. The database contains the device address (48-bit IEEE public address) and the corresponding link key. The link key can be a combination or a unit key.

The link key is used to authenticate other units (see section 2.3). There are four more “keys” in Bluetooth: ciphering key (K_C), temporary key (K_{master}), initialisation key (K_{init}), and a Personal Identification Number (PIN). The ciphering key is the key used to encrypt a Bluetooth link. The ciphering key is derived during authentication (see section 2.3 below). The temporary key is a special key used for broadcast encryption (see [5]). The initialisation key and the PIN are used during the pairing procedure. Below we describe the pairing procedure.

2.2.2 Pairing procedure

The purpose of the paring procedure is to generate a common link key. In Bluetooth this is done in two steps. First, both units calculate an initialisation key, K_{init}. The calculation of K_{init} is based on a (short) common secret, a PIN, known to both devices. Next, the link key (combination or unit key) is calculated. K_{init} is used in the calculation of the link key.

The initialisation key is derived from the Bluetooth address, the PIN, the length of the PIN, and a random number IN_RAND using a noninvertible algorithm based on the SAFER+ block cipher [14]. IN_RAND is generated by one of the devices taking part in the pairing and is sent to the other unit. If the PIN is shorter than 128 bits it is augmented using the Bluetooth address. If one unit has a fixed PIN, the Bluetooth address of the other device is used. If both units use a variable PIN the address of the device that received the IN_RAND is used.

The PIN needs to be known to both units that are to be paired. If the units have a keypad, the PIN can be chosen by the user and entered manually into both devices. Another possibility is that one of the units uses a fixed PIN. Then this PIN is entered into the other device. Both units cannot use a fixed PIN if the pairing should be possible.

K_{init} is used when deriving the link key. The unit key is transmitted from on unit to the other simply by masking it with K_{init}. The combination key is generated as follows. Each entity generates a secret random number LK_RAND of 128 bits, and sends it to the other party encrypted by masking it using the initialisation key. The entities compute their respective key shares from the random number and the device address of each entity, using an algorithm based on SAFER+ block cipher. The length of the key share is 128 bits. The 128-bit combination key is obtained by xoring the two key shares.

2.3 Authentication and encryption

During the authentication process one unit, the verifier, sends a random value to the other unit, the claimant. The claimant has to process the random value together with the secret key, i.e. the link key, to obtain a correct response value. The response value is sent back to the verifier who compares the received value with an expected value pre-calculated by the verifier. The authentication processing uses an authentication function. The Bluetooth authentication function is based on the block cipher SAFER+. The authentication works only one way. If the units want mutual authentication, two consecutive authentication processes must be performed. As a side result, the authentication process generates an extra bit string, the Authentication Ciphering Offset (ACO). The ACO is used for ciphering key generation.

In order to initialise the encryption engine, both units need a common ciphering key. The ciphering key is calculated (in most cases) as a cryptographic hash of the link key, a random value and the ACO. The ciphering engine is a stream cipher that uses four linear feedback shift registers. For details regarding the cipher we refer to [5].

3. SECURITY SHORTCOMINGS

Next we discuss security shortcomings in the Bluetooth security concept. Jakobsson and Wetzel discussed different security weaknesses in Bluetooth in [11]. However, they do not give a correct description of the Bluetooth system and how the weaknesses could be used to attack the system. The three main problems that they list are:

- Unit keys are weak from a security point of view
- Pairing using short PIN values is weak
- The usage of fixed device addresses introduce the risk of location attacks

From a practical point of view, an additional problem is the lack of standardised options of the key exchange procedure. Only PIN based pairing is provided in the standard. Below we briefly discuss the different issues.

3.1 Unit keys

The authentication and encryption mechanisms based on unit keys are the same as those based on combination keys. However, a unit that uses a unit key is only able to use *one* key for all its secure connections. Hence, it has to *share* this key with all other units that it trusts. Consequently all trusted devices are able to eavesdrop on any traffic based on this key. A trusted unit that has been modified or tampered with could also be able to impersonate the unit distributing the unit key. Thus, when using a unit key there is no protection against attacks from trusted devices.

The Bluetooth combination keys would be much more appropriate to use for almost any Bluetooth unit and the Bluetooth SIG does not recommend the use of unit keys.

3.2 Short PIN values

We now describe an attack on the Bluetooth pairing procedure. We here assume that the units calculate a combination key (the similar attack applies to a unit key calculation).

In section 2.2.2 we described the Bluetooth pairing procedure. During the pairing procedure both units calculate an initialisation key. The only secret input to the key calculation is the PIN. In the next step the combination or unit key is calculated. This calculation is protected using the initialisation key. Directly after the exchange of the link key, the authentication procedure is performed. The authentication uses the newly derived link key.

All key derivation algorithms are symmetric algorithms that can be implemented in hardware or in software. The computational complexity of the algorithms is not large. Assume that an intruder records all communication during the key exchange and the first authentication between two units. He can then calculate, for each possible PIN value, the corresponding K_{init}. Furthermore, for each K_{init} value, he can calculate the corresponding link key. Finally, for each link key value he can then check the response value for the observed challenge (or he can issue a challenge himself towards the victim device). If he finds a match, he has obtained the correct link key. Since all calculation steps have low complexity, unless the PIN space is large, the intruder can easily compute the correct link key.

As an alternative, the attacker can obtain the PIN and link key by initiating a key exchange with a victim device and perform the same step as described above.

If the attack described above should succeed, the intruder must be present at the pairing occasion and record all communication. Hence, the Bluetooth SIG does not recommend pairing at public places and strongly encourage the use of long PIN number. However, there is a need for alternative and more secure key exchange options. We discuss such new options in section 4.

3.3 Privacy

In Bluetooth, two or more units sharing the same channel form a piconet. In each piconet one unit is given the special task of being the master, whereas the other unit(s) act as slave(s). All units have a unique device address (BD_addr). The BD_addr has a length of 48 bits and has the following format:

LSB **MSB**

company_assigned	**company_id**	
LAP	**UAP**	**NAP**
<---------------24 bits -------------->	<---8 bits --->	<-------16 bits -------->

The LAP and UAP form the significant part. The Bluetooth access code is the first part of each packet transmitted in Bluetooth. Some of the access codes used in Bluetooth are uniquely determined by the LAP in the Bluetooth device address. There are four different distinct access codes:

- Channel Access Code (CAC)- The CAC is derived from the Master's LAP.
- Device Access Code (DAC) – The DAC is derived from the specific device's (slave) LAP
- Inquiry Access Code (IAC)— Can take two different forms, but is derived from special dedicated LAP values not related to any specific BD_addr.

Bluetooth units are discovered using an *inquiry* procedure. Units that are in *inquire scan* substate, answer to an inquiry with the Frequency Hop Synchronisation (FHS) packet that contains their BD_addr. A connection towards a particular unit is made by a *paging* procedure. A paging unit uses the DAC of the paged unit. A unit that recognises his DAC and that is in the *page scan* substate will reply to a paging. When a connection has been established, the CAC is included in each packet.

Hence, the CAC and DAC can be used to track the location of a specific user. Furthermore, the whole Bluetooth address (LAP, UAP and NAP parts) is sent in the FHS packets used on certain occasions. In section 6 we describe a new solution, which to a large extent prevents location tracking in Bluetooth.

4. SECURITY ENHANCED PIN AND LINK KEY ESTABLISHMENT

After a secure link has been established between two Bluetooth devices, it can be used to secure all subsequent communication between the devices. With higher layer key exchange mechanisms secure link key establishment can be provided and the attack on low-entropy PIN (see section 3.2) can be avoided.

It is proposed in the Baseband specification [5] that cryptographic mechanisms, such as the Diffie-Hellman key exchange, could be used for establishing a sufficiently strong shared secret. However, the use of Diffie-Hellman or any other strong key exchange method has not yet been specified. In this section we describe such methods and consider different aspects of higher layer key exchange. These aspects include flexibility in the choice of key exchange mechanism; we cover both strong manual key exchange (similar to the current pairing) and key exchange based on a key infrastructure.

In order to allow existing key infrastructures and standard key exchange protocols to be used for Bluetooth link key establishment, standard protocol formats needs to be developed. IEEE has developed a standard, 802.1X, for access control in a LAN environment [1]. 802.1X includes methods for authentication and key exchange. The authentication and key exchange in 802.1X is based on the IETF standard Extensible Authentication Protocol (EAP) [7]. 802.1X and EAP is also suitable to use for Bluetooth. This will allow standard key exchange protocols like TLS [8] to be reused for key establishment in Bluetooth.

Even with EAP, many Bluetooth applications will need keys to be established in an *ad hoc* manner using human interaction. The current pairing procedure can be enhanced with external applications for link key establishment and exchange. We discuss two approaches to human assisted authentication of

Diffie-Hellman key exchange. Passkey-based encryption is treated in section 4.1 and passkey-based checking in section 4.2. Passkey-based key establishment can be implemented within the standard protocols using the EAP encapsulations.

4.1 Encrypted Diffie-Hellman key-exchange

Since the publication of the seminal scheme EKE by Bellovin and Merritt [3], [4], there has been an ever-growing interest and theoretical research in the area of passkey authenticated key exchange and authentication protocols. A security proof in the ideal cipher model of the two-flow core protocol of EKE was given in [2]. Also several new passkey authenticated protocols have been proposed. The most serious attack, under which many proposals have collapsed, is the off-line dictionary search to recover the used passkey. Vulnerability under this attack is increased by the requirement that the passkey must be human memorable. For recent theoretical advances in this area, see, e.g.,[12]. For practical development and standardisation effort the IEEE P1363a study group should be mentioned [9], which includes also the Internet draft SRP [17]. However, no generally accepted solution is available yet.

Passkey authenticated key exchange protocols are also potentially useful for link key establishment in various Bluetooth scenarios. A Bluetooth device is typically lightweight mobile equipment with limited computational resources mainly due to limited power supply. Therefore, all superfluous computation should be avoided and the functionality kept at the necessary minimum. Provable security properties even if they cost only “eight times more computation than the standard Diffie-Hellman” [12] cannot be afforded in devices where even the standard Diffie-Hellman is in the frontiers of throughput capability.

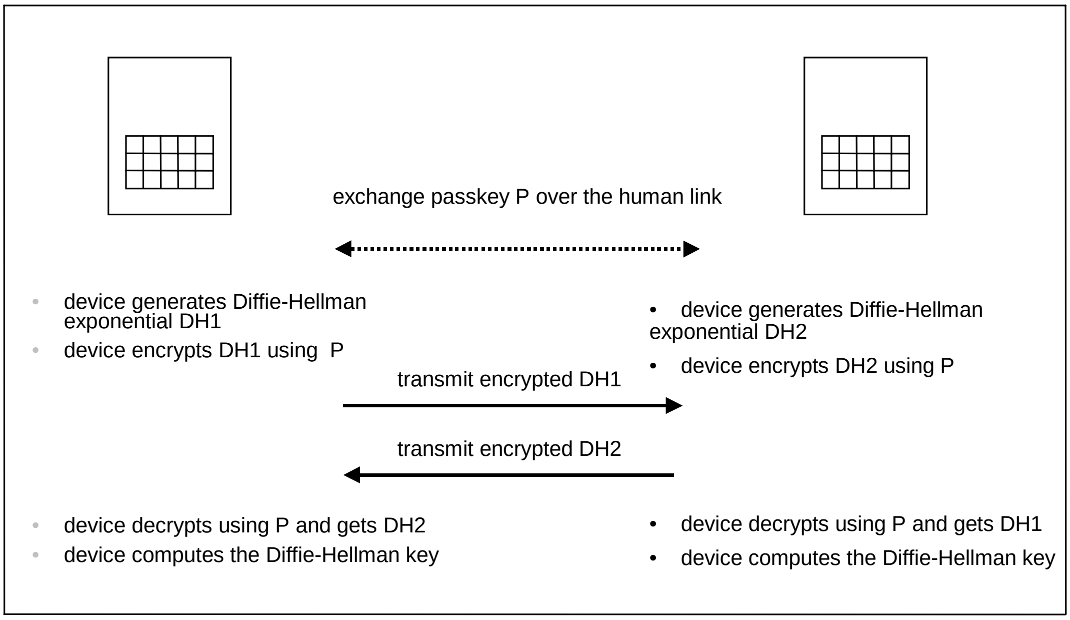

exchange passkey P over the human link
• device generates Diffie-Hellman exponential DH1
• device encrypts DH1 using P
• device generates Diffie-Hellman exponential DH2
• device encrypts DH2 using P
transmit encrypted DH1
transmit encrypted DH2
• device decrypts using P and gets DH2
• device computes the Diffie-Hellman key
• device decrypts using P and gets DH1
• device computes the Diffie-Hellman key

Figure 1, Two-flow core EKE

In Figure 1, the two-flow core EKE, preceded by the passkey generation step, is depicted. It would be suitable for Bluetooth link establishment, and does not add to the computational overhead of the standard Diffie-Hellman key exchange protocol. However, practical implementations are not known to exist.

4.2 Diffie-Hellman key-exchange with check values

In this section we describe methods where the users of the communicating Bluetooth devices can verify the authenticity of the established link key after it has been created. These methods are based on the presumption that, if an active man-in-the-middle is present in the Diffie-Hellman key exchange, then the established Diffie-Hellman keys will be different in the legitimate devices. Therefore it is sufficient to the users to check that the established keys are equal.

A straightforward method to verify equality of two secret values is to make the devices compute cryptographic hash values and display them to the users to compare. Straightforward application of such checks would require the hash values to be at least 32-40 hexadecimal digits long to prevent a spoofer from mounting a birthday attack.

The birthday attack can be launched by the man-in-the-middle as follows. Let f denote the function for computing the hash value of the Diffie-Hellman key. Assume that a man-in-the-middle C can get the legitimate parties A and B to start the Diffie-Hellman key exchange. So A generates a and sends g^a, and, similarly, B generates b and sends g^b. Both Diffie-Hellman tokens are snatched by the man-in-the-middle, who starts generating two sets of exponents c and d until a match $f(g^{ac})= f(g^{bd})$ is found.
In [16] comparing hash value on the screens of the devices was discussed but considered unpractical for simple devices, mainly because of the required length of the hash values, which simple devices, such as those of thermometers, are not capable of displaying. Another approach to preventing the birthday attack was used in [13]. This approach uses short, say 4-6 hexadecimal digits long check values, but requires a special implementation of the Diffie-Hellman key exchange protocol. The participants split their Diffie-Hellman tokens in two about equally long halves and fully transmit the first halves before transmitting the second halves. Next we present a third approach. Similarly as [13] it allows short check values, but does not require any special implementation of the Diffie-Hellman protocol. The approach is very similar to the standard challenge-response verification of the shared secret, used in peer entity authentication with irreversible algorithm [10]. The main difference is that the response is verified over the human link. Therefore significantly shorter challenge and response can be used.

After having executed the anonymous Diffie-Hellman protocol over the wireless link, the communicating parties authenticate the established key using human interaction. The procedure is as follows:

1. One party generates a challenge, which is a short random string of typical length of 4-6 hexadecimal characters. The challenge is communicated to the other device using human interaction or the wireless link.
2. Both parties use the same method for computing a response from the challenge and from the shared secret key established as a result of the anonymous Diffie-Hellman protocol. The response length of 4-6 hexadecimal digits provides sufficient security. For interoperability, the response computing method needs to be standardised. For example, such a method could be based on a cryptographic hash function such as SHA-1.
3. The check values CV to be compared by the users are a concatenation of the challenge and the response. If the CV values formed by both communicating parties are equal, the Diffie-Hellman key is accepted. Otherwise it is rejected.

The implementation of these steps depends on the user interfaces of the Bluetooth devices. Two different scenarios can be distinguished depending on whether one display is used (A), or two displays are used (B). Two basic examples of protocols using check values are given in Figures 4 and 5.

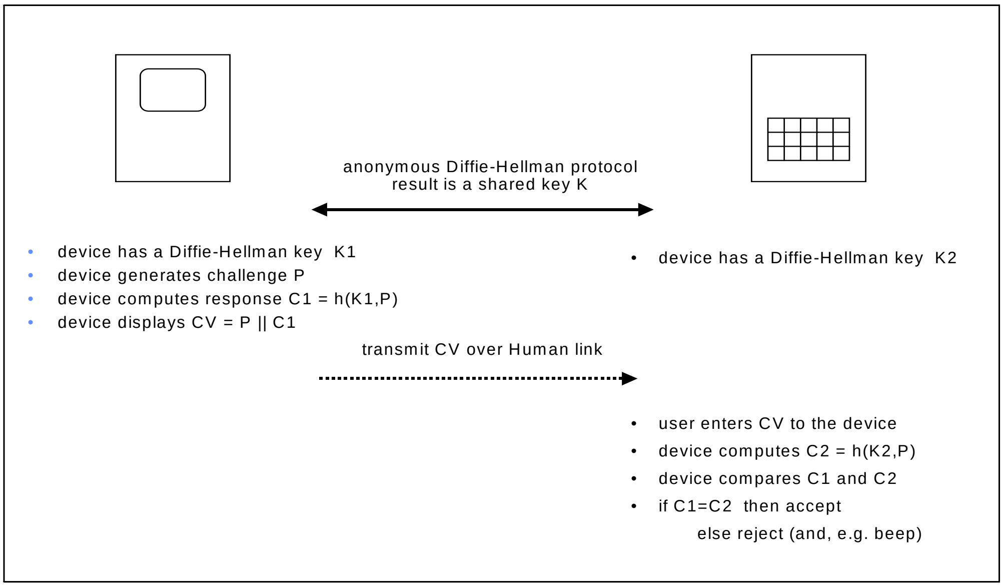

anonymous Diffie-Hellman protocol
result is a shared key K
device has a Diffie-Hellman key K1
device generates challenge P
device computes response C1 = h(K1,P)
device displays CV = P || C1
device has a Diffie-Hellman key K2
transmit CV over Human link
user enters CV to the device
device computes C2 = h(K2,P)
device compares C1 and C2
if C1=C2 then accept
else reject (and, e.g. beep)

Figure 2, Check Value verification with one display – Scenario A

In some application, a most natural implementation may be achieved as a mixture of the above procedures. For example, scenario A can be modified as follows: user enters only the challenge to the second device, and then compares the responses on the displays. Reversely, an alternative implementation of scenario B goes as follows: the equality of the challenges is verified on the displays, and then one device displays the response, which the user subsequently enters to the other device.

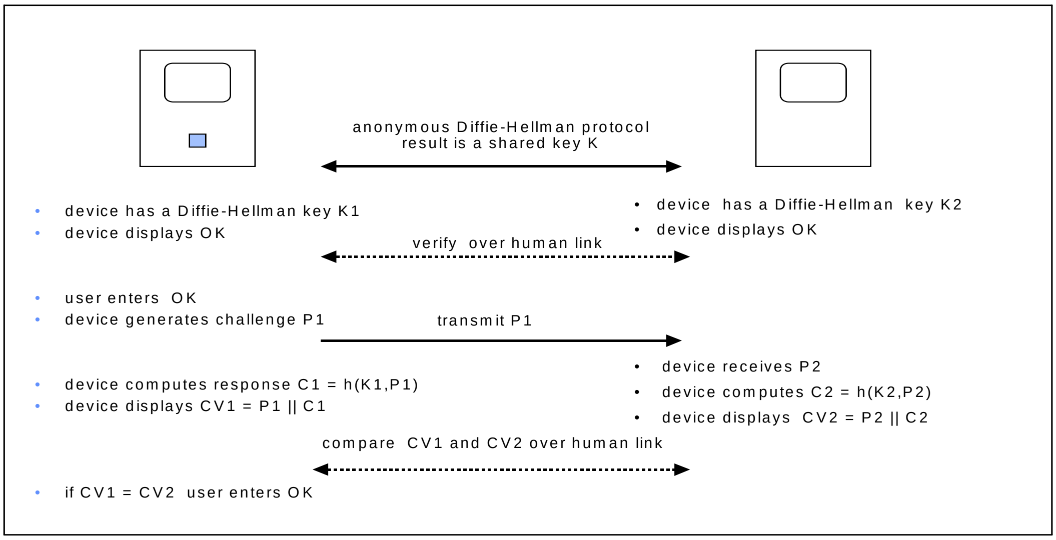

anonymous Diffie-Hellman protocol
result is a shared key K
device has a Diffie-Hellman key K1
device displays OK
device has a Diffie-Hellman key K2
device displays OK
verify over human link
user enters OK
device generates challenge P1
transmit P1
device receives P2
device computes response C1 = h(K1,P1)
device displays CV1 = P1 || C1
device computes C2 = h(K2,P2)
device displays CV2 = P2 || C2
compare CV1 and CV2 over human link
if CV1 = CV2 user enters OK

Figure 3, Check Value verification with two displays – Scenario B

5. ACCESS POINT ROAMING USING THE ENHANCED LINK KEY ESTABLISHMENT

Next, we show how the enhancements in section 4 can be used in an access point roaming scenario. We describe new security architecture for such a scenario. We are considering a situation where a Bluetooth Data Terminal (DT) can move around and access several different LAN Access Points (LAPs) belonging to the same access service provider. In order to be user friendly, manual configuration by the user at each new connection set-up should be avoided.

One general possible security principle for the architecture would be to use totally open (from security point of view) access points that can be accessed by anybody. But, often the service provider would like to restrict the access. Furthermore, the Bluetooth user would like to be sure that he connects to the correct access point and that the traffic sent over the Bluetooth radio interface is not eavesdropped.

The section is structured as follows. First we describe a new key concept for Bluetooth, group keys. The security architecture is based on the usage of group keys. In section 5.1.2 we give an overview of the access point roaming security architecture. Section 5.1.3 and section 5.1.4 describe how secure access point roaming can be achieved by combining the higher layer key exchange mechanisms with the group key concept.

5.1.1 Group keys

We build our architecture on a *group key* concept. By using group keys, with only minor changes, we can use the current Bluetooth security mechanism also for the access point roaming scenario.

We assume that a link key is not necessarily unique for one *link* but is used by one unit for one particular *service*. This type of new link key we called *group key*. We assume that before a unit subscribes to a new service, a *group key* for that particular service is generated. Later, when the user of the unit would like to utilise the service, the keys to use is obtained by getting the service ID using the Bluetooth service discovery and make a lookup in the internal key database. It might be possible for the user to enforce his unit to only use ordinary combination keys for some connections while it still might allow *group key*s for other type of connections. For example the key memory in the host might be like in the example in Table 1 below.

Service	**BD_addr**	**Usage**	**Key**
LAN access A	************	Service dependent	AB124223 23E23A12 1264BEF1 A2845D28B
LAN access B	************	Service dependent	2343AF23 6496ECA A68BEA396 9464B47E
Any	3FA12437BC45	Always	23BD378A 93678928 AB2784BD FE376925
Any	D234BD6A24E9	Always	374585937 2691A373 12FD2839 CF381749

Table 1, Bluetooth key database with group keys

In the table, records for combination keys have the device address (BD_addr) filled with the corresponding Bluetooth unit address. The group keys have the BD_addr filled with the accepted address range (wild card notation). In the example, the two first keys are group keys while the two second are ordinary combination keys.

5.1.2 Architecture

Here we describe an architecture where the Bluetooth Baseband authentication and encryption are used to protect the access link. The architecture can be implemented using EAP encapsulation (see section 4). We assume encrypted Diffie-Hellman key-exchange (see section 4.1). But, any key infrastructure can actually be used.

The Bluetooth Baseband authentication is used to control that only legitimate users are able to connect to the LAN. We distinguish between two different situations (from the DT point of view):

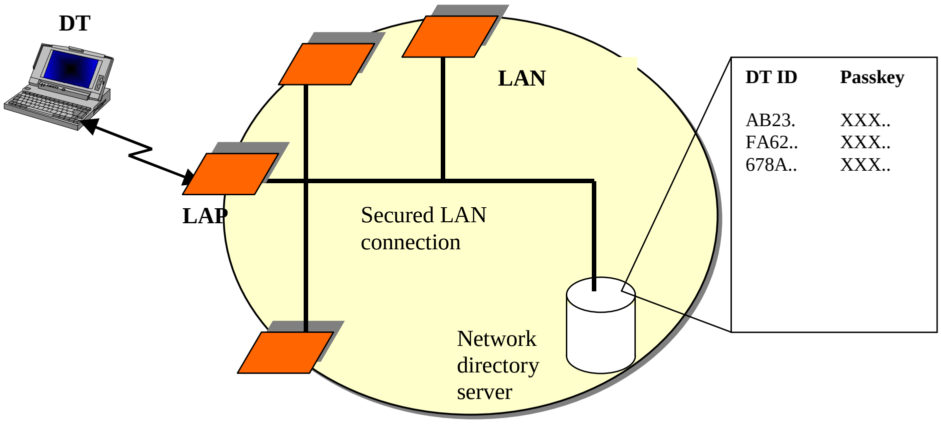
1. **Establish initial trust relationship:** Initially a DT tries to connect to a network to which it has not been connected previously. Hence, link keys must be exchanged. Subsequent connections are handled automatically or almost automatically without any interaction with higher layer security mechanisms.
2. **Subsequent access to LAPs:** Here we utilise the group unit concept to allow fast convenient access to different LAPs.

5.1.3 Initial trust based on passkeys

Next we describe how the necessary group key is obtained, i.e., how to create the necessary initial trust relation.

Assume a user would like to register his DT for getting LAN access through LAPs installed by a certain LAN access service provider or organisation. This can be done, for example, using one of the following two options:

- The user registers the DT at the LAN access provider through some regular (non-Bluetooth) procedure (phone, office, Web etc.)
- The user is getting LAN through his own organisation and the DT need some preconfiguration in order to be allowed to access the network through LAPs.



DT ID	Passkey
AB23.	XXX..
FA62..	XXX..
678A..	XXX..

Figure 4, LAN access network architecture with subscription server containing user PINs

We assume that when a DT user subscribes to a LAN access service it gets a unique ID that identifies the service provider. Together with the ID the user also receives a secret passkey. In order to provide high security the passkey should be sufficient large. The passkey is generated by the LAN access service provider using a secure random generator and is generated independently for each DT subscriber in the LAN. The DT user (or someone on behalf of the DT user) needs to enter the passkey manually into the device, in its well-protected LAN access service database. The database entry consists of two values:

- LAN access service ID
- Passkey for the particular LAN access service

Also at the registration the user is given to the LAN access provider a unique DT ID. This ID can be LAN access specific or it can be the Bluetooth device address.
As part of the subscription, the LAN access provider stores the passkey and corresponding DT ID in a central secure database. All LAPs in the access network need to have secure access to this database as described in Figure 4. The access to the database can be secured by any standard method.
Now, the initial connection is performed according to Figure 5.

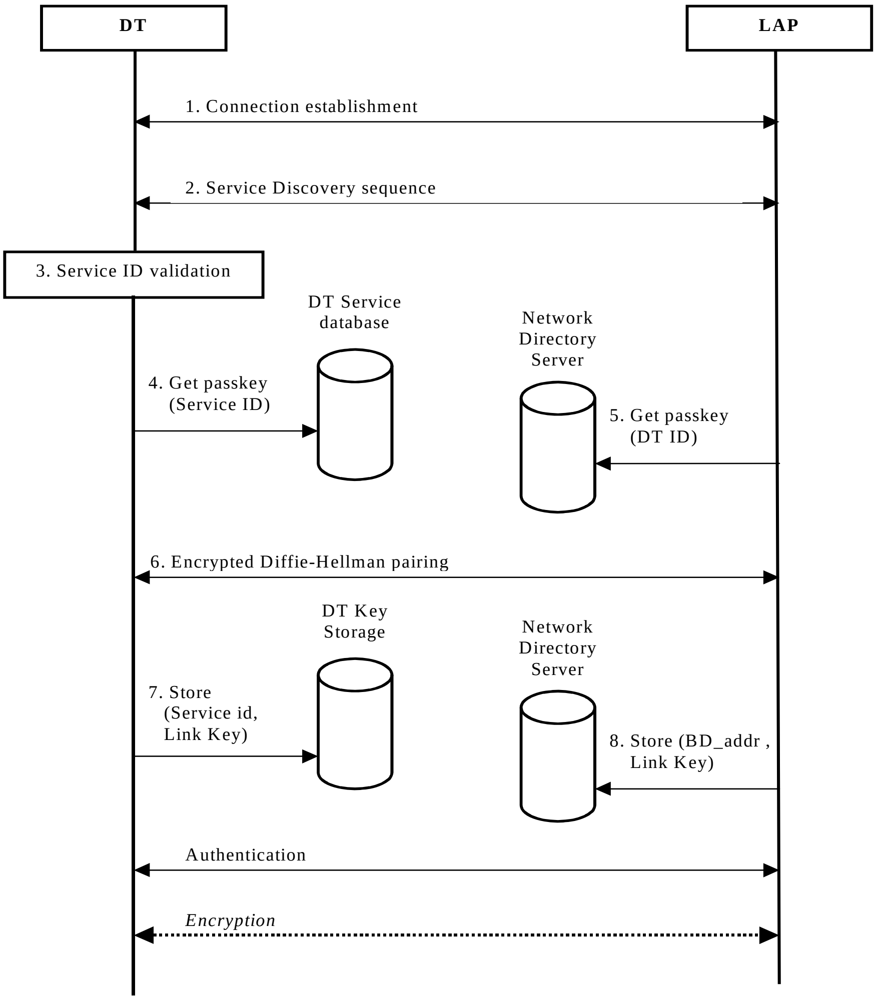

DT
LAP
1. Connection establishment
2. Service Discovery sequence
3. Service ID validation
DT Service database
Network Directory Server
4. Get passkey (Service ID)
5. Get passkey (DT ID)
6. Encrypted Diffie-Hellman pairing
DT Key Storage
Network Directory Server
7. Store (Service id, Link Key)
8. Store (BD_addr , Link Key)
Authentication
Encryption

Figure 5, Initial pairing procedure

Below we give a detailed description of each step in the pairing procedure:

1. The DT connects to the LAP or the LAP connects to the DT using the Bluetooth paging procedure.
2. The DT acts as a Service Discovery Protocol (SDP) client and search for LAN access service record on the LAP. The DT receives the service ID of the LAP. The LAP may perform a similar

service discovery sequence on the DT to obtain the DT ID. However, if the DT ID is the device address of the DT this is not necessary.

3. The DT checks that it knows the service ID received over the SDP protocol. Otherwise, the DT interrupts the connection procedure.
4. The DT asks the internal service database for the passkey corresponding to the service ID.
5. The LAP makes a secure network connection towards the central network directory server (see Figure 4) to obtain the passkey corresponding to the received DT ID.
6. The DT and LAP performs a higher layer passkey based pairing procedure according to any of the methods described in section 4.1. As a result of the pairing the DT and LAP share a common link key (group key).
7. The DT stores the derived key in its internal key memory (see Table 1).
8. The LAP stores the group key for the DT in the network directory server. The key is might be identified by the BD_addr of the DT.

5.1.4 Subsequent access to LAPs

We assume the usage of security mode 2. This means that no security procedures are initiated before a channel establishment request has been received or a channel establishment procedure has been initiated. We assume the group key concept that we described in 5.1.1. The group key concept can only be used together with security mode 2. If the DT connects to the LAN for the fist time, authentication and encryption is performed according to the description in section 5.1.3. For all other cases, the procedure is as described in Figure 6.

Below we give a detailed description of each step in the secure connection establishment:

1. The DT connects to the LAP or the LAP connects to the DT using the Bluetooth paging procedure.
2. The DT acts as a SDP client and search for LAN access service record on the LAP. The DT receives the service ID of the LAP.
3. The DT checks that it knows the service ID received over the SDP protocol. Otherwise, the DT interrupts the connection procedure.
4. If this is *not* the first time the DT connects to this particular LAN, the DT read the *group key* corresponding to the received service ID from the DT key storage.
5. The LAP makes a secure network connection towards the network directory server to obtain the link key corresponding to the BD_addr (if the BD_addr is used as DT ID) of the connected DT.
6. The link key is used to perform mutual authentication and encryption of the access link.

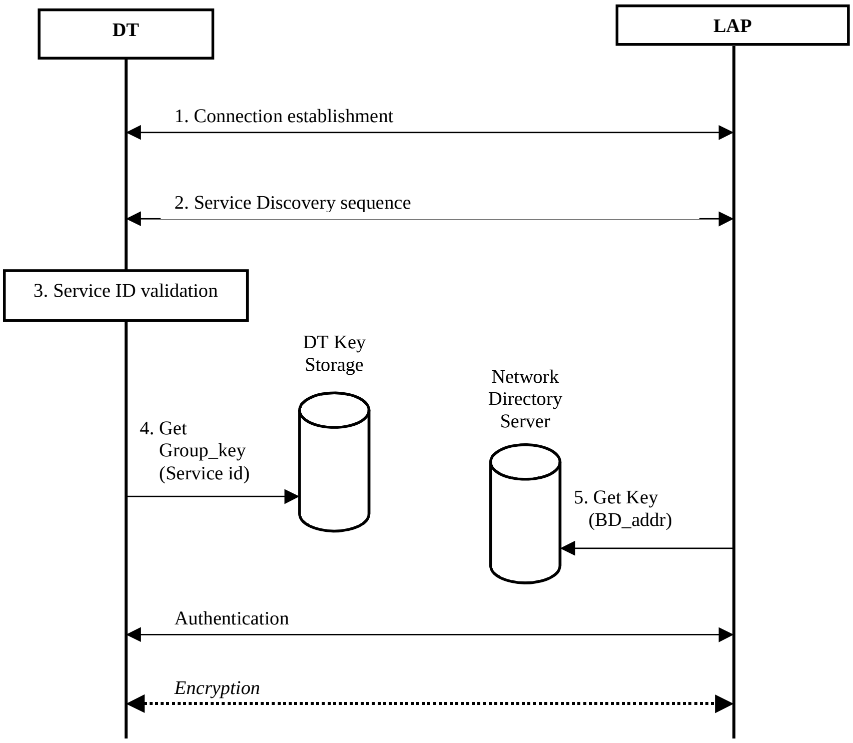


Figure 6, Connection procedure with Baseband authentication and encryption

6. ANONYMITY MODE

Finally, we describe the usage of an anonymity mode in Bluetooth. The purpose with the anonymity mode is to reduce the privacy problem that we described in section 3.3. We use an approach where short-lived addresses are chosen at random, but where all units have the current long-lived addresses.

A Bluetooth module can operate in anonymity or non-anonymity mode. Each anonymity-enabled device uses two addresses:

1. A 48 bits BD_addr (the same as in the current specification).
2. An 48 bits "active" address BD_addr_active.

The BD_addr_active is obtained as follows:

- At each update occasion the new BD_addr_active, is obtained by choosing 48 bits independently and by random. These bits determine all fields in BD_addr_active.

- The BD_addr_active should be updated regularly according to some fixed time intervals, and also at power up. However, at a time when the device is a master device in any piconet, the BD_addr_active must not be updated.

The CAC is *always* calculated from the BD_addr_active of the master and will hence change over time.

6.1 Inquiry procedure

The inquiry procedure is exactly the same as in the current specification. However, the address returned by the node in inquiry scan mode is BD_addr_active and *not* the BD_addr.

6.2 Paging and page scan procedures

We propose two *page scan* substates. In the first substate, unit listens to DACs based on the active as well as the long-lived address. One substate shall be the active substate. When a unit is in the second substate, it only listens to DACs based on the active address. The user can choose to only enter the first substate when the user explicitly chooses to enter that substate. We distinguish between two different paging situations:

1. Paging a previously paired unit.
2. All other cases

6.2.1 Paging a previously paired unit

In this case we assume that the paging unit knows the BD_addr of the unit that it would like to connect to. Furthermore, we assume that the units are bonded and have agreed on an alias address (see section 6.3). A unit will only respond to this paging if it is in the first substate.

The devices make a Baseband connection using the normal paging mechanism and the BD_addr of the slave. The access codes and frequency hopping sequence are based on the BD_addr_active of the master.

Once the connection is made at Baseband level, but before any authentication takes place, the master sends to the slave a special packet, which contains the BD_addr_alias that was agreed upon the last time that the two devices met. The slave can look up this value to obtain the true BD_addr of the master. The subsequent authentication and encryption operations then take place using the BD_addr of the master, not the BD_addr_alias or the BD_addr_active.

6.2.2 All other cases

The paging procedure will be exactly as in the current specification. The only difference is that the access codes (CAC and DAC), as well as the frequency hopping schemes, are based on the BD_addr_active addresses.

6.3 Obtaining the BD_addr and BD_addr_alias of the other unit

It remains to define how a unit gets hold of the long-lived BD_addr of the other unit. After making an inquiry and a page according to the schemes above, the master does not know the BD_addr of the slave and vice versa. However, after the connection has been set up, the units perform a security pairing procedure.

When the pairing is done and an encrypted connection is established, the two units exchange (if desired) their respective BD_addr and agree on a common 48 bits randomly chosen alias BD_addr_alias. The BD_addr_alias will have a dual role in that it will, to the master, serve as an alias for the slave and vice versa. It can be chosen by either unit and sent to the other. Each unit needs to keep a database of all the BD_addr_alias of the units that it is bonded to. Each time the BD_addr_alias is used (i.e. sent to the slave at the connection creation) and after the units have set up a secure connection, the BD_addr_alias is updated and sent to the paged unit.

7. CONCLUSIONS

We have discussed recently discovered shortcomings in the Bluetooth systems. We have shown that almost all of these shortcomings can be avoided with enhancements to the Bluetooth Baseband security. With new pairing procedures based on public key methods the weak pairing procedure can be avoided while the user convenience is maintained. Furthermore, we have shown that by extending the current link key concept and by utilising the enhanced pairing procedure, secure and convenient access point roaming can be achieved using Bluetooth Baseband security. We have also introduced a new anonymity mode that can be used in Bluetooth for improved privacy to avoid location tracking. Parts of the results presented in this paper are currently under discussion in the Bluetooth SIG for possible inclusion in future releases of the specification.

ACKNOWLEDGEMENTS

We thank Dan Willey for providing us with information on EAP encapsulation techniques. Exchange of ideas with all other members of the Bluetooth SIG Security Expert Group has also been useful when preparing this paper.

In relation to mechanisms presented in Section 4.2 discussions with Valtteri Niemi are gratefully acknowledged.

REFERENCES

[1] 802.1X LAN MAN Standards Committee of the IEEE Computer Society, “Standard for Port based Network Access Control”, *IEEE Draft P802.1X/D11*, March 27, 2001.

[2] M. Bellare, D. Pointcheval, and P. Rogaway, “Authenticated key Exchange Secure against Dictionary Attacks”, *Advances in Cryptology -- Eurocrypt 2000*, LNCS 1807, pp. 139-155, Springer-Verlag 2000.

[3] S. Bellovin and M. Merritt, “Encrypted Key Exchange: Password-Based Protocols Secure against Dictionary Attacks”, *Proceedings of the Symposium on Security and Privacy*, pp. 72-84, IEEE, 1992.

[4] S. Bellovin and M. Merritt, “Cryptographic protocol for secure communications”, US Patent #5,241,599, August 31, 1993.

[5] Bluetooth SIG, *Specification of the Bluetooth system, Core, Part B*, “Baseband specification”, Version 1.1, 22 February 2001, at http://www.bluetooth.com/.

[6] Bluetooth SIG, *Specification of the Bluetooth system, Profiles, Part K.1*, “ Generic Access Profile”, Version 1.1, 22 February 2001, at http://www.bluetooth.com/.

[7] L. Blunk and J. Vollbrecht, “PPP Extensible Authentication Protocol (EAP)”, *RFC 2284*, 1998.

[8] T. Dierks and C. Allen, “The TLS Protocol Version 1.0”, *IETF RFC 2246*, January 1999.

[9] IEEE P1363a: “Password-based authentication and key agreement protocols”, http://grouper.ieee.org/groups/1363/StudyGroup/Passwd.html

[10] IS 9798-4, Information technology - Security techniques - Entity authentication - part 4: “Mechanisms using a cryptographic check function”

[11] M. Jakobsson and S. Wetzel, “Security Weakness in Bluetooth”, *RSA Security Conference 2001*, April, 2001

[12] J. Katz, R. Ostrovsky, and M. Yung, “Efficient Password-Authenticated Key Exchange Using Human-Memorable Passwords”, *Advances in Cryptology -- Eurocrypt 2001*, LNCS 2045, pp. 475-494, Springer-Verlag 2001.

[13] D. Maher, “Secure communication method and apparatus”, US Patent #5,450,493, September 12, 1995.

[14] J. Massey, G. Khachatrian, and M. Kuregian, “Nomination of SAFER+ as Candidate Algorithm for the Advanced Encryption Standard (AES) ”, NIST AES Proposal 1998, see also http://csrc.nist.gov/encryption/aes/round1/round1.htm#algorithms

[15] J. Persson and B. Smeets, “Bluetooth Security --An Overview”, *Information Security Technical Report*, Vol. 5, No. 3, pp. 32-43, 2000.

[16] F. Stajano, R. Anderson, “The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks”, *1999 AT&T Software Symposium*, available at B. Christianson, B. Crispo, and M. Roe (Eds*.). Security Protocols, 7th International Workshop Proceedings*, LNCS, vol. 1796, Springer-Verlag 1999.

[17] T.Wu, “The SRP Authentication and Key Exchange System”, *IETF RFC 2945*, September 2000.