Upgrade of Bluetooth Encryption and Key Replay Attack

Kaarle Ritvanen and Kaisa Nyberg
Nokia Research Center
Helsinki, Finland
{kaarle.ritvanen,kaisa.nyberg}@nokia.com

***Abstract*— After adoption of the Advanced Encryption Standard (AES), security systems of many information and communication systems are being upgraded to support AES based encryption mechanisms. Also within Bluetooth SIG, discussions about replacing the proprietary Bluetooth encryption algorithm E_0 with a new, possibly AES based algorithm have been initiated. The purpose of this paper is to show that this action alone does not improve the overall security because in the Bluetooth system, an active attacker can set up a previously used encryption key by a replay attack. We analyze the problem and show that it is possible to overcome it. Finally, we present alternative modifications to the Bluetooth Baseband and Profiles specifications to support secure use of two different encryption algorithms.**

***Index Terms*— AES, Bluetooth, E_0, encryption, key replay attack**

I. Introduction

Bluetooth is a wireless communications standard intended to be used in personal area networks. Many handheld devices, such as mobile phones, personal digital assistants, and laptop computers, incorporate a Bluetooth radio to enable low-cost wireless data transmission.

Like all modern radio communications systems, Bluetooth uses an encryption algorithm to conceal the transmitted data from eavesdroppers. The encryption algorithm is a stream cipher called E_0, which is based on Linear Feedback Shift Registers (LFSRs) and a summation combiner [1]. The length of the encryption key can be varied between 8 and 128 bits and there is a negotiation procedure by which it can be agreed on. [2, Part H, Chpt. 4]

The E_0 encryption algorithm is a proprietary algorithm designed for the Bluetooth system. There are incentives to introduce a stronger encryption mechanism to Bluetooth, preferably based on the Advanced Encryption Standard (AES) [3]. Nevertheless, support for E_0 cannot be removed, in order to provide compatibility with legacy devices. Previous attacks on the GSM system by Barkan, Biham, and Keller [4] show that two different encryption algorithms within the same system may interfere in an undesirable manner. In one of the scenarios [4, Sect. 7.1], the attacker can force the victim to reuse a previously used encryption key with the weak algorithm, and then recover the key. Then the attacker can decrypt the previously generated ciphertext, even if strong encryption has been used. We show that such an attack may also be possible in Bluetooth if appropriate counter-measures are not taken. Section IV presents the details on how the attack is carried out in the Bluetooth system. The fundamental cause of the problem is that it is possible to replay encryption keys. In Section IV-D, we present our recommendations for the counter-measures that would be sufficient to allow a second encryption algorithm to be securely taken into use in Bluetooth system.

It should be noted that recovering encryption keys is not the only exploit of the possibility for encryption key replay. For instance, Gauthier presented a key replay attack applicable against the EAP-AKA protocol [5] when a Bluetooth link is used between the victim devices [6].

Before presenting the details of our attack, some background information is covered. Section II reviews the state-of-the-art attacks on E_0. The Bluetooth encryption key exchange and authentication procedures are described in Section III.

II. Status of Encryption Algorithm E_0

In 1999, Hermelin and Nyberg showed how it is possible to recover the initial state of the LFSRs from 2^{64} consecutive keystream bits doing a work of 2^{64} [7]. The amount of work has later been reduced to 2^{61} and the required knowledge of keystream bits to 2^{50} [8]. These attacks exploit linear correlations in the summation combiner. Nevertheless, these attacks are of theoretical nature, since the LFSRs are reinitialized after each packet and the length of the keystream never exceeds 2744 bits[1] [2].

At the moment, algebraic attacks seem to be the most effective attacks on E_0. Krause devised an attack requiring a work of 2^{77} but only 128 consecutive bits of known plaintext [9, Sect. 7]. That amount is eminently realistic for an attacker to obtain but the workload is still prohibitive and equivalent to exhaustive key search of a 78-bit key. Later, Armknecht and Krause showed how to recover the initial state from 2^{23} keystream bits doing a work of 2^{68} [10]. By using a technique called *fast algebraic attack*, which requires some precomputation, the amount of work can be reduced to 2^{55} [11], [12].

The aforementioned attacks concentrate on discovering the initial state of the LFSRs from the keystream bits. However, it

[1]The Bluetooth specifications state that the maximum size of payload is 2745 bits. This maximum is achieved by type DM5 packets with 228-byte payload, which maps to 2745 bits due to error-correcting channel coding. However, encryption is applied before channel coding and therefore the maximal-length keystream is used with type DH5 packets having 343-byte payload, which equals to 2744 bits.

has been proven that having an effective algorithm for initial state recovery yields an effective algorithm for recovering the secret key [13].

According to Armknecht, recovering E_0 keys using present known plaintext attacks would require about 128 GB of memory and 8 MB of keystream. With present computing machinery, it would take at least 159 years to perform the computations. [14]

Even if not breakable in practice, E_0 is of lower security level than AES based stream ciphers are currently believed to be. Therefore, if a second AES based encryption algorithm is specified for Bluetooth, the Baseband or the application profile specifications must ensure that two different encryption algorithms can coexist in Bluetooth without causing vulnerabilities to each other.

III. Procedures for Key Exchange and Authentication

This section presents the Bluetooth encryption key exchange and authentication procedures as defined in [2]. The general order in which the related activities take place is:

1) Change of link key
2) Mutual authentication
3) Encryption key exchange

In Bluetooth networks, there is one *master device*, with the clock of which the other devices synchronize. These other devices are called *slaves*. The security protocols are always performed only between the master and a slave but never between two slaves. They are not symmetric and depend on these roles, as we will see.

Mutual authentication and key exchange are mandatory after link key renewal but they are allowed to happen at any other time too. Link keys are discussed in Section III-A. Section III-B explains how authentication works in Bluetooth and Section III-C shows how encryption keys are agreed on.

A. Link Keys

Link key is a shared secret between the communicating devices. In principle, there are four types of link keys:

- combination keys
- unit keys
- temporary keys[2]
- initialization keys

Unit keys are used by devices with limited memory resources but their use is deprecated and they are ignored in this discussion. Initialization keys are used when pairing devices. The attack presented in Section IV assumes that the devices have already been paired, so initialization keys neither are interesting in this context.

Temporary keys are used in point-to-multipoint configurations. As the name suggests, such configurations are usually relatively short-lived. Applications may make the slaves use a common encryption key derived from this common temporary link key to allow encryption of broadcast traffic. Note that also unicast traffic is encrypted with the common key if a temporary link key has been set up. After the master has finished broadcasting that needs encryption, the slaves can be told to fall back to the previous link keys.

In most cases, the link key is a combination key. According to the specifications, combination keys are *semi-permanent*, in the sense that they can be changed but typically have long lifetimes. In fact, the specification suggests that combination keys can be stored into non-volatile memory and used to authenticate and generate encryption keys for future sessions. So it is reasonable to assume that link keys do not change very often in point-to-point configurations.

B. Authentication

Bluetooth uses a special algorithm named E_1 to authenticate other devices. It is based on the SAFER+ block cipher [16]. The inputs to E_1 are:

- current link key
- device address of the claimant
- 128-bit challenge

The challenge is generated by the verifier and sent to the claimant. Both parties run E_1 and the claimant sends the response to the verifier that checks whether the results match. E_1 produces also another result, which is called Authenticated Ciphering Offset (ACO). This 96-bit value is used in key exchange and is discussed in Section III-C.

Authentication always takes place for both directions after the link key has been changed. Also the order is fixed: first the master authenticates the slave and then vice versa. This is also true when a temporary multipoint key is taken into use. It is up to the application whether authentication is performed at other times. These additional authentications do not necessarily have to be mutual. In principle, authentication can be performed arbitrarily many times and in arbitrary order unless the application imposes some restrictions on that.

C. Encryption Key Exchange

E_0 encryption keys are generated by an algorithm called E_3, which produces a 128-bit result. If the encryption key is to be shorter than that, the key is shortened by a binary polynomial modulo operation. The inputs to E_3 are:

- current link key
- 128-bit random number
- Ciphering Offset number (COF)

The random number is generated by the master and is supplied to the slave with the control message that requests starting encryption. The last input, COF, takes one of the following values:

- the device address of the master repeated twice, if the link key is a temporary key, or
- ACO produced by the latest authentication, otherwise.

[2]In [2], keys of this type are called *master keys*, but this term is a bit misleading. In specifications of some other wireless communications systems, such as those of Wireless Local Area Networks [15], long-term link keys (combination key equivalents) are called master keys.

IV. Active Attack on Strong Encryption Algorithm

Let us now assume that a second alternative encryption algorithm is inserted to the Bluetooth system. Then the support for the original E_0 algorithm will be maintained to ensure backward compatibility. Hence, it is necessary to insert a cipher negotiation mechanism to the Link Manager Protocol (LMP) [2, Part C] so that the devices can agree on a common algorithm. Moreover, it is natural to impose change of encryption key after change of encryption algorithm to prevent the same encryption key from being used with two different algorithms.

We also make the following additional assumptions about how the new feature is used in Bluetooth system. The assumptions are realistic and in accordance with the current specification.

1) The same E_3 algorithm is used to generate the encryption keys for all encryption algorithms. This is reasonable, since most modern block ciphers, such as AES, use 128-bit keys.
2) The application does not restrict the order of execution of authentication procedures.
3) The link key is not changed often (i.e. it remains the same throughout all sessions involved in the attack).

Finally, we make the general assumption that passive wiretapping and recording of Bluetooth communication as well as active "Man-in-the-Middle" impersonation is possible in Bluetooth. In particular, we assume that the attacker can impersonate the master to a slave, and send control messages to the slave. Note that we do not assume that the master can adjust its clock as is required by Gauthier's attack [17, Sect. 2].

We show that if these assumptions hold, then it is possible for an active attacker to force a Bluetooth slave device to reuse a previously used encryption key with an encryption algorithm selected by the attacker. In this manner, a situation is created where the attack of Barkan *et al.* works. This violates the requirement that the different encryption algorithms must not pose any threat to each other.

At first, in Section IV-A we consider a simple case involving only combination-type link keys. In Section IV-B, we show that under certain conditions this attack can be even easier to perform. Section IV-C discusses whether the attack can be extended to sessions containing encrypted point-to-multipoint transmissions.

A. Basic Attack

In case of point-to-point configurations, which we are now considering, the value of ACO is directly used as COF, the input to the encryption key generation algorithm E_3. If authentication is performed for both parties, the ACO produced by the latest authentication is used. Hence the factors that determine the encryption key are:

- current link key
- master-supplied random number
- challenge supplied by the verifier of the last authentication
- device address of the claimant of the last authentication

The attack works as follows. At first, the attacker records a session that is encrypted by using a strong algorithm. Prior to that, he sees the master supply the last authentication challenge, observes the random number attached to the encryption start request, and saves those messages.

Later, at a moment best suitable for him, the attacker becomes active and impersonates the master to the slave. The old link key is used, so there is no need for mutual authentication. Now the attacker runs the negotiation procedure to take the weak encryption algorithm into use. As explained in the beginning of Section IV, a procedure to exchange a new encryption key is performed. It may be possible to use an existing ACO value, as discussed in the next subsection. If a new ACO value is needed, the attacker requests the slave to authenticate itself by sending the previously recorded challenge, as allowed by assumption 2. Being unaware of the real link key, the attacker of course cannot verify the response of the slave, but the result is that the challenge he supplies defines the same ACO, as before. Then the attacker initiates encryption by replaying the random number it recorded from the previous session. The resulting encryption key is identical to the one of the session that the attacker recorded.

It is important to note that if the master is the verifier in the last authentication, the encryption key solely depends on values supplied by him.[3] The slave has then no opportunity to affect the key. This enables the attacker to set up the same encryption key by replaying these values, since by assumption 1 the same E_3 algorithm is used with both encryption algorithms. Now the attacker can try to recover the key by using an attack on the weak algorithm, and then decrypt the ciphertext created using the strong algorithm if he succeeds.

B. Using Existing ACO

A variation of the attack may be possible if the same ACO is allowed to be used for several encryption key computations. If the same ACO were used, COF would remain constant for long periods of time, just like the link key. Then we are again in the situation where the master is the only one who affects the encryption key. The specifications do not forbid reusing ACOs. In fact, they encourage using the same ACO for several key computations in certain situations. When discussing mutual authentication after a temporary key has been distributed, they say [2, Part H, Sect. 3.2.8]:

> The ACO values from the authentications shall not replace the current ACO, as this ACO is needed to (re)compute a ciphering key when the master falls back to the previous (non-temporary) link key.

Therefore, it is highly probable that several implementations do not require a fresh ACO for each encryption key derivation.

[3]Indeed, the encryption key depends on the current link key the attacker does not know. But because of assumption 3, it is constant throughout the attack and in that sense does not affect the encryption key. As regards to the device address, the same holds.

Attacking on such implementations necessitates only replaying the random number input for E_3, not the authentication challenge, thus rendering assumption 2 unnecessary. It is not even necessary for the attacker to know the last challenge, it is required only that the replay takes place when the ACO value is the same as in the recorded session.

C. Point-to-Multipoint Configurations

Let us assume that the application allows the master to make the slaves switch to temporary link and encryption keys, and the attacker has recorded a session that contains such encrypted broadcast episodes. It is clear that the attacker is able to recover such parts of the recorded session that were encrypted using a point-to-point key, since he can replay separately all authentications and key exchanges he has seen. But could the attacker somehow recover broadcast encryption keys too?

Before broadcast encryption can be started, a new temporary link key is created and transmitted to the slaves, in encrypted form of course. But as mutual authentication always occurs after this, there is no way for the attacker to remain undetected, since he does not know the new link key. [2, Part H, Sect. 3.2.8]

However, there can be applications that constantly use the temporary link key. In that case, the temporary key is never relinquished and the attack works well, just like in the point-to-point case. Note that in this case, the attacker need not know the authentication challenge, but can send any plausible value, since COF is derived from the master's address.

D. Possible Counter-Measures

Assumption 3 stated that the link key is not changed often. However, if the specifications dictated that the link key must be changed regularly, that would offer some protection against this replay attack. Replaying the challenge and the random number would no longer yield the same encryption key, had the link key been changed. Moreover, as mutual authentication must always occur after change of link key, changing link keys frequently would certainly offer protection against attacks of this kind. Point-to-multipoint applications constantly switching between combination and temporary group keys naturally use this means of protection.

Another possibility to protect against replay attacks is to make the slave always supply the last challenge. LMP definition rules that the slave supplies the last challenge in mutual authentication after the link key has been changed [2, Part C, Sect. 4.2]. However, this does not by itself prevent the master from initiating new authentication and key exchange procedures immediately after that.

We made the assumption that after each negotiation of encryption algorithm a new encryption key must be exchanged. We assumed that in this process authentication is performed only one way: master authenticates the slave. One might think that requiring mutual authentication would be sufficient to prevent the attacker from replaying an encryption key. However, this is not the case. By impersonating the slave to the real master, the attacker can forward the challenge to the master and get the correct response which it forwards to the slave.

We implicitly assumed that the attacker can freely select the encryption algorithms in protocol negotiation phase. This assumption is based on the fact that currently there is no other integrity protection mechanism than encryption in Bluetooth, and encryption cannot be used before the algorithm has been agreed on. In theory, using message authentication codes based on link keys to protect the negotiation would prevent this attack. However, it would not prevent other types of encryption key replay attacks, such as Gauthier's attack mentioned in Section I.

Another counter-measure that prevents the same encryption key from being used for two different encryption algorithms is to specify a new different E_3 algorithm for each new encryption algorithm. But again, other types of replay attacks would not be neutralized.

V. Conclusion

In this paper we demonstrated that just introducing a second stronger encryption algorithm is not sufficient to upgrade the security level as desired. The root cause of the problem is that it may be possible for a master device to replay authentication and key exchange.

Four alternative approaches to protect against the attack discussed in Section IV were proposed:

- The link key is changed frequently.
- Bluetooth application profiles mandate the slave to provide the last authentication challenge before encryption key derivation *and* forbid using a single ACO to derive several encryption keys.
- Encryption algorithm negotiation is authenticated.
- Different key derivation algorithms are specified for each encryption algorithm.

The first is contradictory to the idea of link keys. According to the specifications, combination keys are semi-permanent, although changing them frequently would really increase security against active attacks. The second approach is neither in line with the specifications, which tell the implementors to store the ACO for future use. The specifications should rather encourage avoiding ACO reuse under all circumstances.

The last two approaches would only thwart the attacks on multiple encryption algorithms presented in this paper. The key replay attack by Gauthier would still remain valid. But either one of the first two counter-measures would also work against that attack, and therefore are the recommended options.

References

[1] R. A. Rueppel. Correlation immunity and the summation generator. In H. C. Williams, editor, *Advances in Cryptology — CRYPTO '85*, volume 218 of *Lecture Notes in Computer Science*, pages 260–272, Santa Barbara, CA, USA, August 1985. Springer.

[2] Core System Package [Controller volume]. Volume 2 of Specification of the Bluetooth System, Version 1.2. Promoter Members of Bluetooth SIG, November 2003.

[3] Specification for the Advanced Encryption Standard (AES). NIST FIPS Publication 197, November 2001.

[4] E. Barkan, E. Biham, and N. Keller. Instant ciphertext-only cryptanalysis of GSM encrypted communications. In Boneh [18], pages 600–616.

[5] J. Arkko and H. Haverinen. Extensible Authentication Protocol Method for UMTS Authentication and Key Agreement (EAP-AKA). Internet Draft (`draft-arkko-pppext-eap-aka-12.txt`), April 2004.

[6] E. Gauthier. A man-in-the-middle attack using Bluetooth in a WLAN interworking environment. 3GPP TSG SA WG3 Meeting #32, S3-040163, February 2004.

[7] M. Hermelin and K. Nyberg. Correlation properties of the Bluetooth combiner. In J-S. Song, editor, *Proceedings of ICISC '99*, volume 1787 of *Lecture Notes in Computer Science*, pages 17–29, Seoul, South Korea, December 1999. Korea University, Springer.

[8] P. Ekdahl and T. Johansson. Some results on correlations in the Bluetooth stream cipher. In *Proceedings of the 10th Joint Conference on Communications and Coding*, Obertauern, Austria, 2000.

[9] M. Krause. BDD-based cryptanalysis of key stream generators. In L. R. Knudsen, editor, *Advances in Cryptology — EUROCRYPT 2002*, volume 2332 of *Lecture Notes in Computer Science*, pages 222–237, Amsterdam, The Netherlands, 2002. Springer.

[10] F. Armknecht and M. Krause. Algebraic attacks on combiners with memory. In Boneh [18], pages 162–176.

[11] N. T. Courtois. Fast algebraic attacks on stream ciphers with linear feedback. In Boneh [18], pages 177–194.

[12] F. Armknecht. Improving fast algebraic attacks. In B. K. Roy and W. Meier, editors, *Proceedings of Fast Software Encryption 2004*, volume 3017 of *Lecture Notes in Computer Science*, pages 65–82, Delhi, India, February 2004. Springer.

[13] F. Armknecht, J. Lano, and B. Preneel. Extending the framework of the resynchronization attack. In *Proceedings of Selected Areas in Cryptography 2004*, Lecture Notes in Computer Science, Waterloo, Ontario, Canada, August 2004. University of Waterloo, Springer.

[14] F. Armknecht. Algebraic attacks on stream ciphers. Presentation in minisymposium "Secure Crypto for Industry" at ECCOMAS 2004, July 2004.

[15] IEEE Standard for Information Technology; Telecommunications and information exchange between systems; Local and metropolitan area networks; Specific requirements; Part 11: Wireless LAN Medium Access Control (MAC) and Physical Layer (PHY) Specifications; Amendment 6: Medium Access Control (MAC) Security Enhancements. IEEE Standard 802.11i-2004, July 2004.

[16] J. L. Massey, G. H. Khachatrian, and M. K. Kuregian. Nomination of SAFER+ as Candidate Algorithm for the Advanced Encryption Standard (AES). 1st AES Conference, Ventura, CA, USA, August 1998.

[17] Notes on Gauthier's replay attack on the UE functionality split scenario. 3GPP TSG SA WG3 Meeting #32, S3-040091, February 2004.

[18] D. Boneh, editor. *Advances in Cryptology — CRYPTO 2003, 23rd Annual International Cryptology Conference, Proceedings*, volume 2729 of *Lecture Notes in Computer Science*, Santa Barbara, CA, USA, August 2003. Springer.