Key Replay Attack on Improved Bluetooth Encryption

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

***Abstract*— The Bluetooth encryption algorithm E_0 is considered weak, and there are plans to extend the specification so that it would support several algorithms. However, this does not improve the overall security because an active attacker can set up a previously used encryption key by a replay attack. In this paper, we show how this vulnerability can be exploited to thwart any improvement in the encryption method. We also investigate alternative modifications to the Bluetooth security architecture to overcome this problem.**

***Index Terms*— ACO, Barkan–Biham–Keller attack, Bluetooth, E_0, key replay**

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 protect 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 so called summation combiner. 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. [1, Part H, Sect. 4]

However, some weaknesses have been discovered in the E_0 algorithm. There are incentives to introduce a stronger encryption mechanism to Bluetooth, preferably based on the Advanced Encryption Standard (AES) [2]. Nevertheless, support for E_0 cannot be removed, in order to provide compatibility with legacy devices. This constraint combined with the properties of the current key exchange procedure of Bluetooth yields an active attacker a possibility to discover encryption keys regardless of the algorithm used, assuming that he is able to recover E_0 encryption keys.

The fundamental cause of the problem is that it is possible to replay encryption keys. Barkan *et al.* presented several attack scenarios on the GSM network. The scenario of the attack we are proposing is similar to one of them [3, Sect. 7.1]. Section IV presents the details on how the attack is carried out in Bluetooth systems.

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 [4] when a Bluetooth link is used between the victim devices [5].

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. Attacks on 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} [6]. The amount of work has later been reduced to 2^{61} and the required knowledge of keystream bits to 2^{50} [7]. 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] [1].

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 [8, Sect. 7]. That amount is eminently realistic for an attacker to obtain. Later, Armknecht and Krause showed how to recover the initial state from 2^{23} keystream bits doing a work of 2^{68} [9]. By using a technique called *fast algebraic attack*, which requires some precomputation, the amount of work can be reduced to 2^{55} [10].

The aforementioned attacks concentrate on discovering the initial state of the LFSRs from the keystream bits. However, it has been proven that having an effective algorithm for initial state recovery yields an effective algorithm for recovering the secret key [11, Sect. 4.2].

III. Procedures for Key Exchange and Authentication

This section presents the Bluetooth encryption key exchange and authentication procedures as defined in [1]. The general order in which the related activities take place is:

1) Change of link key

[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.

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 said to be *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 allowed to happen at any other time too. But they are mandatory after link key renewal, when the master authenticates the slave first and then vice versa. 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 specifications suggest 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 [13]. 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, which 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 link key has been changed. 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. 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, namely COF, takes the following value:

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

IV. Active Attack on Strong Encryption Algorithm

Let us now assume that another, alternative encryption algorithm will be inserted to the Bluetooth specifications. We also make the following additional assumptions:

1) A cipher negotiation mechanism is inserted to the Link Manager Protocol (LMP) so that the devices can agree on a common algorithm.
2) The original E_0 algorithm is mandatory to implement for backward compatibility.
3) Given N bits of E_0-encrypted data, the attacker can recover the encryption key.
4) 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.
5) The application used does not restrict the order of execution of authentication procedures.
6) The link key is not changed often (i.e. it remains the same throughout all sessions involved in the attack).
7) The attacker can impersonate the master and send control messages to the slave and by those means obtain N bits of ciphertext.

[2] In [1], 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 [12], this term refers to long-term link keys (combination key equivalents).

We show that if these assumptions hold, then the attacker can decrypt any ciphertext created using the new encryption algorithm, assuming he knows the authentication challenge and the master-supplied random number. At first, in Section IV-A we consider a simple case involving only combination-type link keys. In Section IV-B, we make one additional assumption and show that then even stronger attack is possible. Section IV-C discusses whether the attack can be extended to sessions containing 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

As stated in Section I, our attack is just the Barkan–Biham–Keller attack adapted to Bluetooth. At first, the attacker records a session that is encrypted by using a strong algorithm. The attacker also observes the challenge used in authentication prior to the generation of the encryption key and the random number attached to the encryption start request.

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. However, due to assumption 5, the attacker can request the slave to authenticate itself. Being unaware of the real link key, the attacker of course cannot verify the identity of the slave, but the result is that the challenge supplied by him defines the COF when a new encryption key is derived.

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. This is dangerous because the attacker can claim being able to use only E_0 cipher, enforce the use of the same key as was used in previous session, record some ciphertext, recover the key and finally use that key to decrypt the previously recorded session.

B. Stronger Attack

A stronger attack may be possible if the same ACO is allowed to be used for several 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 [1, 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 key derivation. Attacking on such implementations necessitates only replaying the random number input for E_3, not the authentication challenge. Thus it is not even necessary for the attacker to know the last challenge.

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. [1, 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 stronger attack without ACO replay is always applicable, independently of the implementation, since COF is derived from the master's address.

D. Possible Counter-Measures

Assumption 6 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 always must occur just after that, 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 [1,

[3] Indeed, the encryption key depends on the current link key the attacker does not know. But because of assumption 6, 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.

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. And even the slave supplying the last challenge does not by itself thwart the stronger attack discussed in Section IV-B unless using fresh ACOs is required.

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 the Gauthier attack[4] mentioned in Section I.

Another counter-measure that prevents the same encryption key from being used for two different encryption algorithms is to break assumption 4, that is, not to use the same E_3 algorithm for all encryption algorithms but specify a new one for each. But again, other types of replay attacks would not be neutralized.

V. Conclusion

Under certain assumptions, Bluetooth data encryption can be beaten regardless of the encryption algorithm. The root cause of the problem is that it is possible for masters to replay authentication and key exchange.

All assumptions made in Section IV are realistic, perhaps except for assumptions 3 and 7 to hold simultaneously for any N. The current attacks on E_0 are somewhat theoretical since their complexity or the required amount of ciphertext are pretty high. Nevertheless, the key replay problem should not be neglected. It is useless to replace E_0 if this problem is not tackled. Otherwise an active attacker can reduce the problem of breaking the new cipher to breaking E_0.

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

- The link key is changed frequently.
- Bluetooth application profiles force the slave to provide the last authentication challenge *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 a bit 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. The specifications should encourage avoiding ACO reuse under all circumstances rather than telling the implementors to store it for future use. The last two would only thwart the key recovery attack presented in this paper. The key replay attack by Gauthier would still remain valid. But either of the first two counter-measures would also work against that attack, and therefore are the recommended options.

[4]In [5], no solution to the problem described is suggested. It is just assumed that the attacker can always supply the last authentication challenge, although breaking this assumption in the application profile definition would almost be a complete solution to the problem.

References

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

[2] "Specification for the Advanced Encryption Standard (AES)," NIST FIPS Publication 197, Nov. 2001.

[3] E. Barkan, E. Biham, and N. Keller, "Instant ciphertext-only cryptanalysis of GSM encrypted communications," in *Advances in Cryptology — Proceedings of CRYPTO 2003*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, Aug. 2003, pp. 600–616.

[4] 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`), Apr. 2004.

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

[6] M. Hermelin and K. Nyberg, "Correlation properties of the Bluetooth combiner," in *Proceedings of ICISC '99*, ser. Lecture Notes in Computer Science, vol. 1787. Seoul, South Korea: Springer, Dec. 1999, pp. 17–29.

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

[8] M. Krause, "BDD-based cryptanalysis of key stream generators," in *Advances in Cryptology — Proceedings of EUROCRYPT 2002*, ser. Lecture Notes in Computer Science, L. R. Knudsen, Ed., vol. 2332, Technical University of Eindhoven. Amsterdam, The Netherlands: Springer, 2002, pp. 222–237.

[9] F. Armknecht and M. Krause, "Algebraic attacks on combiners with memory," in *Advances in Cryptology — Proceedings of CRYPTO 2003*, ser. Lecture Notes in Computer Science, D. Boneh, Ed., vol. 2729. Springer, Aug. 2003, pp. 162–176.

[10] F. Armknecht, "Algebraic attacks on stream ciphers," in *Proceedings of 4th European Congress on Computational Methods in Applied Sciences and Engineering*, P. Neittaanmäki, T. Rossi, S. Korotov, E. Oñate, J. Périaux, and D. Knörzer, Eds. University of Jyväskylä, July 2004.

[11] J. Lano and B. Preneel, "Extending the framework of the resynchronization attack," in *Proceedings of 11th Annual Workshop on Selected Areas in Cryptography*, ser. Lecture Notes in Computer Science, University of Waterloo. Ontario, Canada: Springer, Aug. 2004.

[12] "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 P802.11i, Draft 10.0, Apr. 2004.

[13] 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, California, USA, Aug. 1998.