Bluetooth のセキュアシンプルペアリングに対する中間者攻撃

野村 大翼† 松尾 和人‡

†‡情報セキュリティ大学院大学情報セキュリティ研究科
221-0835 神奈川県横浜市神奈川区鶴屋町 2-14-1
†mgs102101@iisec.ac.jp

あらまし 本稿では，Bluetooth のセキュアシンプルペアリングの Passkey Entry 方式に対する中間者攻撃を提案する．セキュアシンプルペアリングは端末間認証方式であり，Passkey Entry 方式は同一のパスキーを端末に入力することによって認証を実現している．最近，Lindel や Phan と Mingard によって同方式に対する中間者攻撃が提案された．これらの攻撃では，認証手続きをアボートしてユーザに認証を再試行させる必要があり，その際にユーザが以前と同一のパスキーを入力することを仮定している．提案攻撃手法は，認証再試行の際に以前と異なるパスキーの入力を許す，より現実的な攻撃である．

A Man-in-the-Middle Attack against Secure Simple Pairing in Bluetooth

Daisuke Nomura† Kazuto Matsuo‡

†‡Institute of Information Security
2-14-1 Tsuruya-cho, Kanagawa-ku, Yokohama, Kanagawa 221-0835 JAPAN
†mgs102101@iisec.ac.jp

Abstract This paper proposes a man-in-the-middle attack against the passkey entry mode in the secure simple pairing for Bluetooth. The secure simple pairing is a mutual authentication protocol and the passkey entry mode achieves the authentication by entering the same passkey into the devices to be authenticated. Recently, Lindel, Phan and Mingard proposed man-in-the-middle attacks against this mode. These attacks need to abort the procedure and to make the user to retry it. Moreover, the attacks require that the user enters the same passkey as before in the retries. The proposed attack allows the different passkey in the retry, so that it is more practical than the previous attacks.

1 はじめに

Bluetooth [1][2][3][4]は，情報端末間，又は情報端末と周辺機器を接続する際に用いられる無線通信方式の一つであり，多くの携帯端末やゲーム機などに採用されている．Bluetooth は 2004 年に Bluetooth 2.0 + EDR が登場し，2007 年に Bluetooth 2.1 + EDR が策定された．2009 年には Bluetooth 3.0 + HS と Bluetooth 4.0 が策定されたが，現在は 2.0 + EDR と 2.1 + EDR が普及している．

Bluetooth 端末同士を通信可能な状態にするために，端末間で相互認証し，関連付けを行うことをペアリングと呼ぶ．ペアリングは 2.0 + EDR までは PIN を用いた方式のみであったが，2.1 + EDR においてセキュアシンプルペアリング（以下，SSP と呼ぶ）が追加された．

PIN によるペアリングは Bluetooth 登場直後から安全性が懸念されていた．それは４桁の固定された PIN が多用されていた上，PIN が機器の取扱い説明書に書かれており，その説明書が Web で閲覧できたことにより，攻撃者が PIN を容易に知り得たためである．2001 年に Jakobsson と Wetzel [7]によって，ペアリングの PIN の脆弱性が指摘され，攻撃法が提案された．2005 年には Shaked と Wool [10]によって，Jakobsson と Wetzel の PIN に対する攻撃が実装された．実際に Shaked と Wool は，PIN に対するブルートフォースアタックを行い，Pentium III 450MHz を搭載した PC で 4 桁の PIN を 0.27 秒で導出した．

PIN によるペアリングの脆弱性に対応したものが SSP である．SSP には Numeric Comparison, Just Works, Out of Band, Passkey Entry という４種類の方式がある．Passkey Entry は，数値で構成されるパ

スキーを端末に入力する方式であり，ユーザからは PIN によるペアリングと同じに見えるが，盗聴と中間者攻撃に対して安全であることが仕様に謳われている．

近年，SSP に対する攻撃も提案されている [5][6][8][9]．これらの攻撃には Just Works を利用するものと Passkey Entry に対するものがある．Passkey Entry に対する攻撃は，入力されたパスキーが 1bit ごとに k 回確認されることを利用している．この攻撃はペアリングを攻撃者がアボートしてユーザにペアリングを再試行させる必要があり，その際にユーザが以前と同一のパスキーを入力するという仮定を必要とする．

本稿では，Passkey Entry に対する新たな中間者攻撃を提案する．この攻撃はユーザがペアリングを再試行する際にユーザに以前と異なるパスキーの入力を許す，これまでの攻撃と比較してより現実的な攻撃である．

本稿ではまず，第 2 節で SSP を紹介し，第 3 節で SSP の Passkey Entry に対する既知の攻撃を紹介する．そして第 4 節で新たな中間者攻撃を提案し，第 5 節ではその攻撃への対策を示す．最後に第 6 節で本稿をまとめる．

2　セキュアシンプルペアリング

Bluetooth の端末同士を相互認証し，関連付ける手順がペアリングである．ペアリングによって，リンクキーと呼ばれる，認証や暗号鍵生成に用いられる重要情報が生成され，端末間で共有される．

本節では，ペアリングの一方式である SSP の概略を説明する．

SSP は 5 つのフェーズから成る．フェーズ 1 では楕円曲線 Diffie-Hellman (ECDH) によって公開鍵を交換し，フェーズ 2 ではフェーズ 3, 4 で用いられる値を共有し，フェーズ 3 では共有した全ての情報が端末間で共有できていることの確認を行う．フェーズ 4 でリンクキーを生成し，フェーズ 5 で認証･暗号化を行う． 図 1 に，SSP の流れを示す．

以下では，[4]の Vol.2, PartH: Security Specification に従って，端末 A, B のペアリングのフェーズ 1-3 の概略を説明する．各フェーズで用いられるハッシュ関数等の詳細については，[4]を参照されたい．

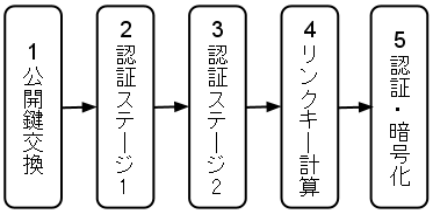

図 1:SSP の流れ

フェーズ 1: 公開鍵交換

まず，双方の端末 A と B は ECDH の公開鍵･秘密鍵のペア (PK_a, SK_a), (PK_b, SK_b) を各々生成し，公開鍵を相手の端末に送信する．端末 A, B は自身の秘密鍵と相手から受信した公開鍵を鍵生成関数 P192 に入力し，共有鍵 $DHKey$ の生成を行う．

$$\text{A: } DHKey_a = \text{P192}(SK_a, PK_b)$$
$$\text{B: } DHKey_b = \text{P192}(SK_b, PK_a)$$

フェーズ 1 を終了すると端末 A, B で $DHKey$ が共有される．即ち，

$$DHKey_a = DHKey_b$$

となる．

DH 鍵交換プロトコルは中間者攻撃に対する耐性がないが，この後のフェーズで中間者攻撃への対応が行われる．

フェーズ 2: 認証ステージ 1

第 2 フェーズには Numeric Comparison, Just Works, Out of Band, Passkey Entry の 4 つの方式が規定されており，端末が持つユーザーインターフェースの I/O によって使用方式が選択される．以下では，その一つの Passkey Entry について詳細に説明する．Passkey Entry には，以下の 2 種類が規定されている．

方式 1: ペアリングする両端末がディスプレイを持たず，キーボードを持つ場合に対応する．ユーザが同一のパスキーを双方の端末に入力する．

方式 2: ペアリングする片方の端末がディスプレイのみを持ち，もう一方の端末がキーボードのみを持つ場合に対応する．一方の端末がランダムに生成･表示したパスキーをもう一方の端末に入力する．

なお，著者らの知る限り，**方式 1** が実装された例はない．

以下では，図 2 に従って Passkey Entry 方式の手順を説明する．

まず**方式 1** の場合，ユーザは端末 A と B に同一の

パスキーを入力する．**方式 2** の場合には，A がパスキーをランダムに生成し，表示する．そして，ユーザは A に表示されたパスキーを B に入力する．A, B に表示・入力された $k$bit のパスキーを各々 $r_a = (r_{a1}, r_{a2},.., r_{ak})$, $r_{ai} \in \{0, 1\}$, $r_b = (r_{b1}, r_{b2},.., r_{bk})$, $r_{bi} \in \{0, 1\}$とする．

Passkey Entry では，パスキーが一致していることを，ビットごとに確認する．即ち $r_{ai} = r_{bi}$を $i = 1, \cdots, k$に対して確認する．以下に， $i$bit 目の確認手順を示す．

Aはナンス N_{ai}を生成し，PK_a，PK_b，r_{ai}と N_{ai}に対しハッシュ関数 f1 を用いてハッシュ値 $C_{ai} = f1(PK_a, PK_b, N_{ai}, r_{ai})$ を計算し，Bに送信する．Bはナンス N_{bi}を生成し，PK_a，PK_b，r_{bi}と N_{bi}に対するハッシュ値 $C_{bi} = f1(PK_b, PK_a, N_{bi}, r_{bi})$ を計算し，Aに送信する．

Aは N_{ai}を Bに送信する．N_{ai} を受信した Bは PK_a，PK_b，r_{bi}, N_{ai} を用いて Aと同じ計算を行い，$C'_{ai} = f1(PK_a, PK_b, N_{ai}, r_{bi})$ を得る．次に Bは $C_{ai} = C'_{ai}$を確認する．$C_{ai} \neq C'_{ai}$ ならば， $r_{ai} \neq r_{bi}$として，ペアリングをアボートし，$C_{ai} = C'_{ai}$であれば N_{bi}を Aに送信する．Aは PK_a，PK_b，r_{ai}, N_{bi}を用いて Bと同じ計算を行い，$C'_{bi} = f1(PK_b, PK_a, N_{bi}, r_{ai})$ を得る．A は $C_{bi} = C'_{bi}$ を確認する．$C_{bi} \neq C'_{bi}$ ならば， $r_{ai} \neq r_{bi}$として，ペアリングをアボートし，$C_{bi} = C'_{bi}$であれば $r_{ai} = r_{bi}$ であるとし，iをインクリメントする．

以上を $i = k$まで繰り返し，r_a, r_bの各ビットが同一であることを確認する．パスキーが 6 文字入力された場合に $k = 20$ であり，20 回の繰り返しが必要となる．確認が k回行われた後に，N_{ak}, N_{bk}を各々N_a, N_bとして，フェーズ 3 に渡す．

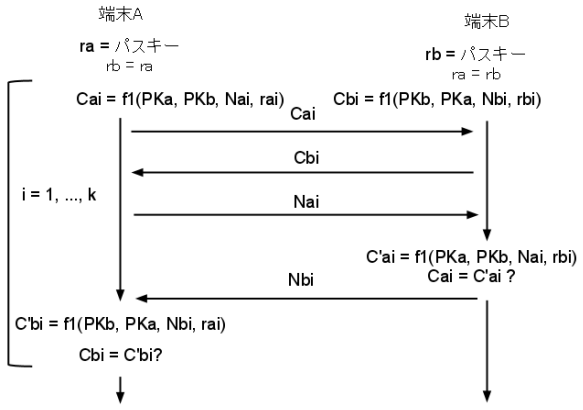

図 2:Passkey Entry の手順

フェーズ 3: 認証ステージ 2

このフェーズは，フェーズ 2 で交換した値が端末間で正しく共有されていることを確認する．フェーズ 1, 2 から引き継がれた $DHKey_a$， $DHKey_b$, N_a, N_b, r_a, r_b, 端末 A, Bの I/O 情報 $IOcapA$, $IOcapB$, Bluetooth MAC アドレス BD_ADDR_a, BD_ADDR_bを用いる．

Aはハッシュ関数 f3 によってハッシュ値 $E_a = f3(DHKey_a, N_a, N_b, r_b, IOcapA, BD\_ADDR_a, BD\_ADDR_b)$ を計算し，Bに送信する． Bは Aから E_aを受信し，Aと同じ計算を行って $E'_a = f3(DHKey_b, N_a, N_b, r_b, IOcapA, BD\_ADDR_a, BD\_ADDR_b)$ を得る．$E_a \neq E'_a$ ならば共有した値が異なっているとして，ペアリングをアボートする．$E_a = E'_a$ ならば，Bはハッシュ関数 f3 によってハッシュ値 $E_b = f3(DHKey_b, N_b, N_a, r_a, IOcapB, BD\_ADDR_b, BD\_ADDR_a)$ を計算し，Aに送信する．

Aは Bから E_b を受信し， $E'_b = f3(DHKey_a, N_b, N_a, r_a, IOcapB, BD\_ADDR_b, BD\_ADDR_a)$ を計算し，$E_b = E'_b$ を確認する．$E_b \neq E'_b$ ならば共有した値が異なっているとして，ペアリングをアボートする．$E_b = E'_b$ ならば，フェーズ 4 へ進む．

フェーズ 4, 5 については，[4]を参照されたい．

3 既知の攻撃手法

本節では，Passkey Entry に対する従来の攻撃を紹介する．

Passkey Entry に対する攻撃は Lindell [16]や Phan と Mingard[18]により提案された．既知の攻撃は以下の A1, A2 に分類される．

A1: ペアリングを盗聴し，盗聴で得られたデータを用いてパスキーを計算・入手する攻撃

A2: 攻撃者がターゲットの端末のペアリング時に正規の端末を装ってペアリングを試み，パスキーを計算・入手する中間者攻撃

これらの攻撃手法はともに，ペアリングを攻撃者がアボートしてユーザにペアリングを再試行させる必要があり，その際にユーザが以前と同一のパスキーを入力するという仮定を必要とする．さらに，この仮定を満足するためには，(実装例のない)**方式 1** が選択される必要がある．

両攻撃はともに Passkey Entry において，C_{ai}, C_{bi} を計算する際にパスキーr_a, r_bが r_{ai}, r_{bi}として 1bit だけ用いられることを利用した攻撃である．

前者の攻撃は，攻撃者が端末 A, Bのペアリング過程を盗聴し，盗聴で得られたデータからパスキーを計算・入手する攻撃である．盗聴した PK_a, PK_bと N_{ai}, 0

をハッシュ関数 f1 に入力し，C''_{ai} = f1(PK_a, PK_b, N_{ai}, 0) を計算する．$C_{ai} = C''_{ai}$なら $r_{ai} = 0$ であり，$C_{ai} \neq C''_{ai}$なら $r_{ai} = 1$ であることが分かる．この $C_{ai} = C''_{ai}$ の確認を $i = 1, \cdots, k$まで繰り返し行い，最後に得られた r_{ai}を連結すればパスキーr_aが得られる．パスキーを得た攻撃者は，ペアリングを再試行する際にユーザが同一のパスキーを利用した場合には，得られた r_aを用いて端末 *A, B* とのペアリングが可能であるため，中間者攻撃に繋がる．

後者の攻撃は，ターゲットの端末のペアリング時に正規の端末になりすました上で *A-B* の通信を中継・改竄し，それぞれに Passkey Entry でペアリングを試みることによって，パスキーを計算・入手する中間者攻撃である．攻撃者 *M* は *B* になりすまして *A* とのペアリングを実行し，*A* になりすまして *B* とのペアリングを実行する．以下では A2 の攻撃の概略を説明する．

まずフェーズ 1 において，*A, B, M* は ECDH の公開鍵・秘密鍵のペア (PK_a, SK_a), (PK_b, SK_b), (PK_m, SK_m) を各々生成する．次に *A* は PK_aを *M*に送信する．*M* は *A* から PK_aを受信し，PK_mを PK_bとして *A* に送信する．*A* と *M* は以下を計算し，*M-A* で *DHKey*を共有する．

A: $DHKey_a$ = P192(PK_m, SK_a)

M: $DHKey_{ma}$ = P192(PK_a, SK_m)

また，*M* は *B* に PK_mを PK_aとして送信し，*B* は PK_mを受信した後，*M* に PK_bを送信する．*B* と *M* は以下を計算し，*M-B* で *DHKey* を共有する．

B: $DHKey_b$ = P192(PK_m, SK_b)

M: $DHKey_{mb}$ = P192(PK_b, SK_m)

次に，フェーズ 2 において，*A* は C_{ai} = f1(PK_a, PK_m, N_{ai}, r_{ai}) を計算し，*M* に送信する．*M* は C_{ai}を受信し，C_{mai} = f1(PK_m, PK_b, N_{mi}, 0) を計算し，C_{ai}として *B* に送信する．*B* は C_{ai}を受信し，C_{bi} = f1(PK_b, PK_m, N_{bi}, r_{bi}) を計算し，*M* に送信する．*M* は C_{mbi} = f1(PK_m, PK_a, N_{mi}, 0) を計算して C_{bi}として *A* に送信する．*A* は N_{ai}を *M*に送信する．N_{ai}を受信した *M* は N_{mi}を N_{ai}として *B* に送信する．

B は C'_{ai} = f1(PK_m, PK_b, N_{mi}, r_{bi}) を計算し，受信した $C_{ai} = C'_{ai}$を確認する．*B* は，もし $C_{ai} = C'_{ai}$ならば N_{bi}を *M*に送信し，$C_{ai} \neq C'_{ai}$ならばペアリングをアボートする．

B が N_{bi}を送信した場合は $r_{bi} = 0$ であり，*B* がペアリングをアボートした場合は $r_{bi} = 1$ であることが分かる．*M* はこの値を記録する．

N_{bi}を受信した *M* は N_{mi}を N_{bi}として *A* に送信する．N_{bi}を受信した *A* は C'_{bi} = f1(PK_m, PK_a, N_{mi}, r_{ai}) を計算し，$C_{bi} = C'_{bi}$を確認する．$C_{bi} = C'_{bi}$ならば，確認を通過し，$C_{bi} \neq C'_{bi}$ならば *A* はペアリングをアボートする．実際には，$r_{ai} = r_{bi}$であるため，*B* がペアリングをアボートしない限り， *A* もアボートしない．

上記の *M-A* と *M-B* のフェーズ 2 を $i = 1, \cdots, k$まで繰り返す．*A, B* にペアリングをアボートされた場合でも，ユーザがペアリングを再試行する際に同じ攻撃を行い，その際にユーザが以前と同一のパスキーを入力すれば，記録されたパスキーのビット値を利用してさらにパスキーを蓄積していくことができる．$i = 1, \cdots, k$までの繰り返しが終わった後に値を連結すればパスキーr_b (=r_a) が得られる．パスキーを得た攻撃者は，ターゲットの端末が再びペアリングを行う際にユーザが以前と同一のパスキーを入力すれば，*A, B* とのペアリングを成功させることが可能になる．

この攻撃では *k*bit のパスキーの約半分 $k/2$ 回のペアリングの試行回数でパスキーの全 bit が得られることが期待される．

4 提案手法

本節では，Passkey Entry に対する新たな中間者攻撃を提案する．

提案手法は，前節の Passkey Entry に対する攻撃 A2 を改良したものであり，A2 と同様にペアリングを攻撃者がアボートしてユーザにペアリングを再試行させる必要があるが，その際にユーザが以前と同一のパスキーを入力するという，既知の攻撃が必要とした仮定の必要なしに攻撃を行うことが可能である．従って，**方式 1** のみならず，実装例のある**方式 2** に対しても有効な攻撃である．また，A2 はペアリングを数回アボートする必要があるが，提案手法に必要なアボートは 1 回である．以降では**方式 2** を対象として提案手法を述べる．

提案手法は，A2 と同様に *M* が *B* になりすまして *A* とペアリングを実行し，*A* になりすまして *B* とペアリングを実行する．そして *M-B* のペアリングを成功させるとともに，*M-A* のペアリングをアボートする．その際，ユーザは *B* が *M* とペアリング済みであることを気付かずに *M-A* のペアリングを再試行する可能性が高い．そこで，*M-B* の通信を持続した状態でユーザにペアリングを再試行させ，*M-A* のペアリングを成功させる．そして，*M-A, M-B* のペアリングを完成させる．

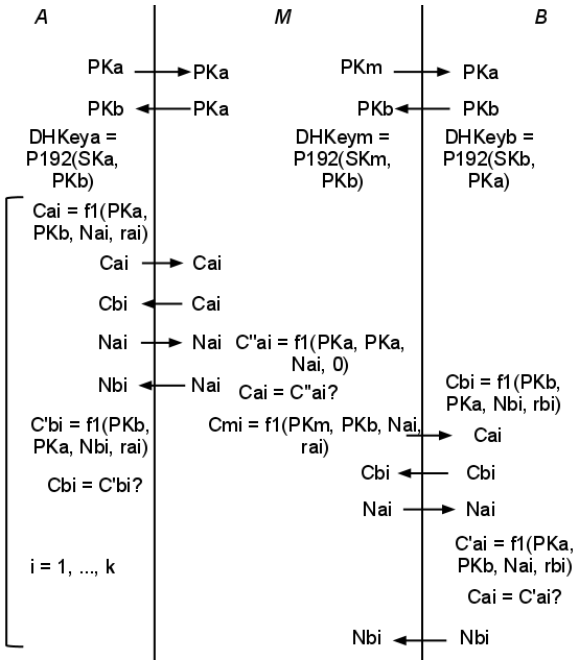

図 3:提案手法 (ステップ 1 - 3)

攻撃の手順

ここでは，6 ステップに分けて提案手法の手順を示す．ステップ 1 は *M-A*，*M-B* のフェーズ 1 に対応した手順である．次のステップ 2, 3 は $i = 1, \cdots, k$ に対して繰り返される．まず，ステップ 2 で *M-A* のフェーズ 2 のパスキー1bitごとの確認から r_{ai} を導出し，ステップ 3 ではステップ 2 で導出した r_{ai} を用いて *M-B* でのフェーズ 2 を行う．そしてステップ 4 においてフェーズ 3 の *M-A* のペアリングをアボートし，ステップ 5 で *M-B* のペアリングを成功させる．最後のステップ 6 では *M-A* のペアリングを再試行の後に成功させ，中間者攻撃を完成させる．図 3 に，提案手法のステップ 1-3 を示す．以下に攻撃手順の詳細を示す．

ステップ 1: 公開鍵交換

攻撃者 *M* は ECDH の公開鍵･秘密鍵のペア (PK_m, SK_m) を生成する．*M* は *A* から受信した PK_a を受信し，PK_a として PK_m を *B* に送信する．*B* は *M* に PK_b を送信し，

$$DHKey_b = \mathrm{P192}(SK_a,\ PK_m)$$

を生成する．*M* は *B* から PK_b を受信し，

$$DHKey_m = \mathrm{P192}(SK_m,\ PK_b)$$

を計算する．ここで，$DHKey_m = DHKey_b$ となる．そして，*A* から受信した PK_a を PK_b として *A* に送り返す．

A が生成する $DHKey_a$ は以下のようになる．

$$DHKey_a = \mathrm{P192}(SK_a,\ PK_a)$$

その後，*A* は r_a をランダムに生成する．そして，ユーザは *A* に表示されるパスキー r_a を *B* に r_b として入力する．

ステップ 2: r_{ai} の導出

A から C_{ai} を受信した後に *M* は C_{ai} を C_{bi} として *A* に送り返す．すると，*A* は N_{ai} を *M* に送信する．*A* から N_{ai} を受信した後に，*M* は PK_a, N_{ai}, 0 をハッシュ関数 f1 に入力して，$C''_{ai} = \mathrm{f1}(PK_a,\ PK_a,\ N_{ai},\ 0)$ を計算する．

そして，*M* は，$C_{ai} = C''_{ai}$ を確認する．$C_{ai} = C''_{ai}$ ならば $r_{ai} = 0$ であり，$C_{ai} \neq C''_{ai}$ ならば $r_{ai} = 1$ である．この r_{ai} の値をステップ 3 に引き継ぐ．

次に *M* は N_{ai} を N_{bi} として *A* に送り返す．*A* は $C'_{bi} = \mathrm{f1}(PK_b, PK_a, N_{bi}, r_{ai})$ を計算し，$C_{bi} = C'_{bi}$ を確認する．*A* が受信した値はそれぞれ $C_{bi} \leftarrow C_{ai}$, $PK_b \leftarrow PK_a$, $N_{bi} \leftarrow N_{ai}$ であり，$C'_{bi} = \mathrm{f1}(PK_a, PK_a, N_{ai}, r_{ai})$ であるので，A には $C_{bi} = \mathrm{f1}(PK_a, PK_a, N_{ai}, r_{ai}) = C'_{bi}$ と見えるため， *A* の確認を通過する．

ステップ 3: r_{ai} を *M-B* のフェーズ 2 で利用

M は，PK_m, PK_b, *A* から受信した N_{ai} とステップ 2 で導出したパスキー r_{ai} を使って $C_{mi} = \mathrm{f1}(PK_m,\ PK_b,\ N_{ai},\ r_{ai})$ を計算して，C_{ai} として *B* に送信する．*B* は $C_{bi} = \mathrm{f1}(PK_b, PK_m, N_{bi}, r_{bi})$ を計算し，*M* に送信する．

B から C_{bi} を受信した後に，*M* は *A* から受信した N_{ai} を *B* に送信する．

B は $C'_{ai} = \mathrm{f1}(PK_m,\ PK_b,\ N_{ai},\ r_{bi})$ を計算し，*B* は $C_{ai} = C'_{ai}$ を確認する．

M はステップ 2 で得た正しい r_{ai} を用いて $C_{mi} = \mathrm{f1}(PK_m,\ PK_b,\ N_{ai},\ r_{ai})$ $(\rightarrow C_{ai})$ を計算しているため，この確認は通過する．

最後に *B* は *M* に N_{bi} を送信する．

M は $i = 1, \cdots, k$ まで，ステップ 2, 3 をアボートせずに繰り返すことで，*M-A, M-B* のフェーズ 2 を通過する．

ステップ 4: *A* とのペアリングアボート

M-A のペアリングは，フェーズ 3 で失敗する．何故ならば，*M* は *A* の PK_a に対応する秘密鍵 SK_a を知らないため，*A* との $DHKey$ の共有ができず，フェーズ 3 で *M* が *B* を装って *A* に送信すべき値 E_b を計算することができないからである．そこで，*M* は適当な値を E_b として *A* に送信し，*A* にペアリングをアボートさせる．

ステップ 5: Bとのペアリング成功

M は E_m = f3($DHKey_m$, N_a, N_b, r_b, $IOcapM$, BD_ADDR_m, BD_ADDR_b) を計算し，E_aとして Bに送信する．

Bは E_aを Mから受信し，E'_a = f3($DHKey_b$, N_a, N_b, r_b, $IOcapM$, BD_ADDR_m, BD_ADDR_b) を計算する．すると，$E_a = E'_a$となり，確認を通過する．そして Bは E_b = f3($DHKey_b$, N_b, N_a, r_a $IOcapB$, BD_ADDR_b, BD_ADDR_m) を計算し，Mに送信する．Mは E_bを受信する．

フェーズ 3 以降の確認も通過し，M-Bのペアリングは成功する．ペアリング成功後も Mは Bとの通信を維持する．

ステップ 6: Aとのペアリングのリトライ

Aにペアリング失敗との表示が出るため，ユーザはペアリングを再試行する．ユーザは，Aが新たに生成・表示したパスキーを Bに入力する．Bと Mは通信中であるため，Mは正しいパスキーを Bから入手できる．このパスキーを用いて正当な手順を踏むことで，Mは Aとのペアリングにも成功し，中間者攻撃が成立する．

5　攻撃への対策

ここでは，4 節で提案した中間者攻撃への対策を議論する．対策は 2 点考えられる．

まず第一に提案攻撃手法も既知の攻撃同様に，パスキーを1bitごとに確認することを攻撃に利用している．そこで，パスキーを 1bit ごとに確認するのではなく，全 bit を一括して確認することで提案攻撃手法を無効化できる．Phan と Mingard [9]が彼らの攻撃への対策として同様の指摘をしている．しかし，Passkey Entry は 1bit ごとに k回に分けてパスキーを確認することで，可変長パスキーに対応しているが，全 bit の確認を一括して行うためには，ハッシュ関数 f1 の拡張等の方式の大幅な修正が必要となる．

第二に，$PK_a \neq PK_b$の確認をフェーズ 1 で行うことが挙げられる．提案した攻撃の中で，Mと Aの一度目のペアリングでは Mが PK_aを PK_bとして Aに送り返している．Aが $PK_a \neq PK_b$を確認し，$PK_a = PK_b$である場合にペアリングをアボートすれば，Mは Aに表示されたパスキーを導出できず，提案攻撃手法は成立しない．ECDH の公開鍵が同一になる確率は極めて低く，この確認は実用的な影響を与えない．

6　まとめ

本稿では Bluetooth のセキュアシンプルペアリングの Passkey Entry 方式に対する新たな中間者攻撃を提案した．

提案手法では，ペアリングを攻撃者がアボートしてユーザにペアリングを再試行させる必要があるが，その際にユーザが以前と同一のパスキーを入力するという既知の攻撃が必要とする仮定を必要とせず，以前と異なるパスキーの入力を許すので，より現実的な攻撃であると言える．また，既知の攻撃では数回ペアリングをアボートする必要があったが，提案手法では 1 回であるため，ユーザに攻撃が気付かれにくい．さらに，既知の攻撃手法は実装が知られていない方式に対するものであったが，提案手法は実在する方式に対しても適用可能である．

参考文献

[1]Bluetooth SIG, Bluetooth 4.0 Core Specification, Bluetooth SIG, 2009.

[2]Bluetooth SIG, Bluetooth 3.0 + HS Core Specification, Bluetooth SIG, 2009.

[3]Bluetooth SIG, Bluetooth 2.1 + EDR Core Specification, Bluetooth SIG, 2007.

[4]Bluetooth SIG, Bluetooth 2.0 + EDR Core Specification, Bluetooth SIG, 2004.

[5]K. M.J. Haataja，K. Hypponen, Man-In-The-Middle Attacks on Bluetooth: A Comparative Analysis，a Novel Attack，and Countermeasure, ISCCSP 2008, Malta，1 2-14 March 2008 pp. 1096 – 1102, 2008.

[6]K. Haataja, P. Toivanen, *Two Practical Man-In-The-Middle Attacks on Bluetooth Secure Simple Pairing and Countermeasure*, IEEE TRANSACTIONS ON WIRELESS COMMUNICATIONS，VOL. 9，NO.1，J ANUARY 2010 pp. 384 - 392, IEEE Communications Society IEEE Signal Processing Society, 2010.

[7]M. Jakobsson, S. Wetzel, *Security weakness in Bluetooth*, Topics in Cryptology CT-RSA 2001, LNCS 2020 pp.176 – 191, Springer, 2001.

[8]A. Y. Lindell, Attacks on the Pairing Protocol of Bluetooth v2.1,Blackhat, 2008.

[9]R. C.-W. Phan, P. Mingard, *Analyzing the Secure Simple Pairing in Bluetooth v4.0*, Wireless Personal Communication, Springer Science+Business Media, 2010.

[10]Y. Shaked, A. Wool, *Cracking the Bluetooth PIN*, International Conference On Mobile Systems, Applications And Services pp.39 - 50, 2005.