2021年度　卒業論文

ProVerifを用いたBluetoothの安全性解析:Passkey Entryモードの検証に向けて

関東学院大学

理工学部理工学科

情報ネット・メディアコース

218K6023　河合　悠斗

指導教員　塚田恭章

2022年　3月

概要

　暗号プロトコルの自動検証ツールとしてProVerif[1]がある。本研究では、BluetoothのセキュアシンプルペアリングのYehら[2]のNumeric ComparisonモードについてProVerifを用いた安全性検証の再現実験を行った。検証の結果、先行研究[3]と同様の結果を得られた。さらに改良されたPasskey Entryモードの検証に向けて形式化する上での難しさや予想される攻撃法についての考察を示した。

目次

第1章　はじめに 1

第2章　Bluetooth 3

2-1　Bluetoothの概要 3

2-2　Numeric Comparisonモードの脆弱性 6

2-3　YehらのNumeric Comparisonモード 7

第3章　形式化 9

3-1　開発環境 9

　3-2　ECDL及びECDHの形式化 9

3-3　排他的論理和の形式化 10

3-4　宣言部 10

3-5　クエリー 11

3-5　デバイスA、デバイスBのプロセスとメインプロセス 12

第4章　検証結果 15

第5章　Passkey Entryモードを形式化する際の見通し 16

第6章　まとめ 17

謝辞 18

参考文献 19

第1章　はじめに

近年、インターネットなど技術の発達により、セキュリティの重要性が日に日に増してきている。また、プロトコルの設計などの複雑なシステムの開発において、高度なセキュリティを保証する技術の確立が求められてきている。そこでProVerif[1]などの自動形式検証ツールを用いたプロトコルの安全性の検証が行われている。

ProVerifはBlanchetらが開発した形式モデルで自動形式検証ツールであり、暗号プロトコルで要求される秘匿性や認証性などの安全性を検証可能である。ProVerifの安全性検証は、攻撃者が利用可能な情報を組み合わせることにより、安全性要件に反するパスがないかを網羅的にシミュレートし、安全性が成立しているか検証する。最終的に攻撃者が入手、生成した情報から目的の情報へ到達可能なパスが導出できた場合、その暗号プロトコルは安全性要求を満たさない。逆に到達可能なパスを導出できなかった場合、その暗号プロトコルは安全性要求を満たす。なお、ProVerifのコードは宣言部とプロセス部で構成されている。宣言部は暗号プロトコルに用いる関数や性質、安全性要求などを記述し、プロセス部は暗号プロトコルそのものを記述する。ProVerifはさまざまな暗号プロトコルの検証に用いられ、多くの暗号プロトコルに対して脆弱性を発見することに成功している。よってProVerifのような自動形式検証ツールを利用することは、複雑なプロトコルの安全性を保障できるとともに複雑なプロトコルを利用する際のリスクを減らすことができる。本研究に使用したProVerifのバージョンは2.00である。

一方、Bluetoothは情報端末間または情報端末とキーボードなど周辺機器を接続する際に用いられる無線通信方式である。Bluetoothにおいて通信可能な状態になるよう端末間で相互認証し、関連付けを行う手順をセキュアシンプルペアリング（SSP）と呼ぶ。2007年にSSPを利用したBluetooth2.1+EDR[4]が登場し、ProVerifを用いた安全検証が行われてきた。まずChangら[5]によってSSPのモードの１つであるNumeric Comparisonモードに対する攻撃法が、井上ら[6]によってPasskey Entryモードに対する攻撃法が発見された。このNumeric Comparisonモードの問題に対してはYehら[2]が、Passkey Entryモードの問題に対してはSunら[7]が改良案を提案している。しかし、横山らによるProVerifを用いた安全検証により YehらのNumeric Comparisonモードの攻撃法が確認され、その攻撃法に対する対策が行われた[3]。しかし一方で、改良されたPasskey Entryモードについては安全検証が行われていない。

そこで、本研究では改良されたPasskey Entryモードの安全性検証に資することを目的に、YehらのNumeric ComparisonモードについてProVerifを用いた安全性検証の再現実験を行った。検証の結果、先行研究[3]と同様の結果を得られた。さらに、改良されたPasskey Entryモードの検証に向けて形式化する上での難しさや予想される攻撃法について考察した。

本研究の構成は以下の通りである。第2章ではBluetoothの説明を行う。第3章ではプロトコルの形式化について説明する。第4章では検証結果について説明する。第5章ではPasskey Entryモードを形式化する際の見通しを述べる。第6章では本研究の総括を行い、今後の課題について述べる。

第2章　Bluetooth

**2-1 Bluetoothの概要**

Bluetoothは情報端末間または情報端末とキーボードなど周辺機器を接続する際に用いられる無線通信方式である。Bluetoothにおいて端末間で相互認証し、関連付けを行う手順をSSPと呼ぶ。SSPにはNumeric Comparison、Just Works、 Out Of Band、 Passkey Entryという４種類のモードがある。Numeric Comparisonモードではペアリングする機器両方の画面に６桁の数字が表示され、人間の目で両方の機器の値が同じかどうか確認し、同じであればyesを送信して認証が完了する。例えばパソコンとプリンターで利用される。それに対してPasskey Entryモードではマスター側で表示された６桁の数字(パスキー)をデバイス側に入力して認証を行う。例えばテザリング、スマートフォンとパソコンで利用される。

　SSPは5つのフェーズからなる。フェーズ1では楕円曲線Diffe Hellman(ECDH)鍵共有により公開鍵を交換し、共有鍵(DHkey)を生成する(図1)。フェーズ2ではフェーズ3、4で用いられる値を共有する(図2、図3)。フェーズ3では共有した全ての値が端末間で正しく共有できていることを検証する(図4)。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5)。

フェーズごとに流れを説明する。

フェーズ１はデバイスA、Bはともに秘密鍵と公開鍵のペア(SKa,PKa)、(SKb,PKb)を生成します。デバイスAはBluetooth Device値A、I/O情報IOcapA、公開鍵PKaを送信する。デバイスBも同様にBluetooth Device値B、I/O情報IOcapB、公開鍵PKbを送信する。最後にデバイスA、Bは自身の秘密鍵と相手の公開鍵を鍵生成関数P192に入力し、DHkey=P192(SKa,PKb)、DHkey=P192(SKb,PKa)の生成を行う。

フェーズ2には前述の４つのモードが規定されており、端末が持つユーザインターフェースI/O情報によってモードが選択される。ここではNumeric ComparisonモードとPasskey Entryモードについてのみ説明する。

Numeric ComparisonモードではデバイスA、BはノンスNa、Nbを生成する。デバイスBはハッシュ値Cb=f1(PKb,PKa,Nb,0)を計算し、ハッシュ値CbをデバイスAに送信する。デバイスA、Bは自身のノンスNa、NbをデバイスB、Aに送信する。デバイスAはハッシュ値Cb’=f1(PKb,PKa,Nb,0)を計算し、Cb=Cb’を検証する。Cb≠Cb’ならばデバイスAはペアリングを中断し、Cb=Cb’ならばデバイスAの相手はデバイスBとなる。デバイスA、Bはハッシュ値Va=g1(PKa,PKb,Na,Nb)、ハッシュ値Vb=g1(PKa,PKb,Na,Nb)を計算し、デバイスの画面に数値を表示する。ここでユーザが目視により確認を行い、異なる場合はペアリングを中断し、同じ場合はデバイスBの相手はデバイスAとなり、フェーズ3に進む。

Passkey Entryモードでは事前に10進数6桁の数字(20bit)のパスキーを共有する。デバイスA、BはノンスNai、Nbiを生成する。デバイスA、Bはハッシュ値Cai=f1(PKa,PKb,Nai,rai)、Cbi=f1(PKb,PKa,Nbi,rbi)を生成し、デバイスB、Aに送信する。rai、rbiは事前に共有したパスキー20bitのうちの1bitである。デバイスAはノンスNaiをデバイスBに送信する。デバイスBはハッシュ値Cai’=f1(PKa,PKb,Nai,rai)を計算し、Cai=Cai’を検証する。Cai≠Cai’ならばデバイスBはペアリングを中断し、Cai=Cai’ならばデバイスBはNbiをデバイスAに送信する。デバイスAはハッシュ値Cbi’=f1(PKb,PKa,Nbi,rbi)を計算し、Cbi=Cbi’を検証する。Cbi≠Cbi’ならばデバイスAはペアリングを中断し、Cbi=Cbi’ならばフェーズ2のデバイスA、Bがノンスを生成する手順に戻り、これを20回繰り返す。

フェーズ3ではデバイスA、Bはハッシュ値Ea=f3(DHkey,Na,Nb,0,IOcapA,A,B)、Eb=f3(DHkey,Nb,Na,0,IOcapB,B,A)を生成し、デバイスB、Aに送信する。デバイスAはEb’=f3(DHkey,Nb,Na,0,IOcapB,B,A)を計算し、Eb=Eb’を検証する。Eb≠Eb’ならばデバイスAはペアリングを中断し、Eb=Eb’ならば次に進む。デバイスBはハッシュ値Ea’=f3(DHkey,Na,Nb,0,IOcapA,A,B)を計算し、Ea=Ea’を検証する。Ea≠Ea’ならばデバイスAはペアリングを中断し、Ea=Ea’ならばフェーズ4に進む。

フェーズ4でリンクキーLK=f2(DHkey,“btlk”,A,B)を計算する。ここで、btlkはプロトコル固有の定数である。

フェーズ5では、このLKを用いて共有鍵Kc=E3(LK,EN\_RAND,COF)を得る。ここで、EN\_RAND、COFはプロトコル固有の定数である。

グラフィカル ユーザー インターフェイス, テキスト

中程度の精度で自動的に生成された説明

図1.公開鍵交換(フェーズ1)

ダイアグラム

自動的に生成された説明

図2.Numeric Comparisonモード(フェーズ2)

ダイアグラム

自動的に生成された説明

図3.Passkey Entryモード(フェーズ2)

ダイアグラム, テキスト, 手紙

自動的に生成された説明

図4.フェーズ3

グラフィカル ユーザー インターフェイス, アプリケーション

自動的に生成された説明

図5.フェーズ4,5

**2-2 Numeric Comparisonモードの脆弱性**

Numeric Comparisonモードには文献[5]で指摘されているように脆弱性がある。その指摘されている脆弱性とはフェーズ1の楕円曲線Diffe Hellman鍵共有の公開鍵PKaとPKbに加え、まだいくつかある。

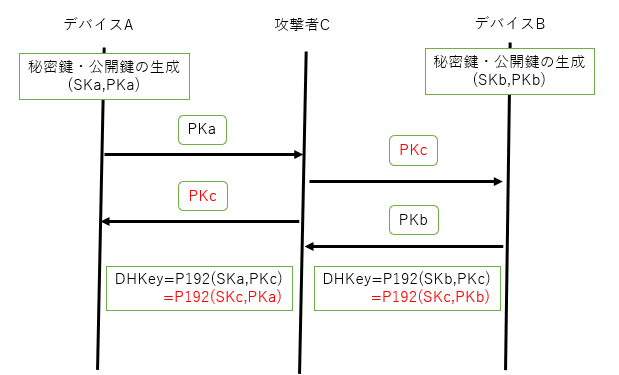
****

図6.フェーズ１の脆弱性

図6にしたがってフェーズ１の脆弱性について説明する。デバイスAが公開鍵PKaをデバイスBに送信する間に攻撃者Cが入り、PKaを自分の公開鍵PKcに書き換え、デバイスBに送信する。デバイスBは公開鍵PKbを攻撃者Cに送り、攻撃者CはPKcをデバイスAに送信する。デバイスAはDHKey=P192(SKa,PKc)を、攻撃者CはDHKey=P192(SKc,PKa)を生成する。デバイスBはDHKey=P192(SKb,PKc)を、攻撃者CはDHKey=P192(SKc,PKb)を生成する。よってデバイスAにとって攻撃者CはデバイスBになりすまし、デバイスBにとって攻撃者CはデバイスAになりすましている。結果、デバイスA、Bは正常にDHkeyを生成できたと思っているが、実際には中間者がいる形になっている。

**2-3 YehらのNumeric Comparisonモード**

本節ではYehらのNumeric Comparisonモードについて説明する。2つのデバイスそれぞれの画面に表示された６桁の数字が同じかどうか確認し、同じならばYes、違えばNoのボタンを押す実験がノキア研究所によって行われ、5人に1人、数字が違っているにも関わらずYesを押したという結果がある[8]。そのため、Yehらは目視確認に伴う人為的ミスへの対策に加え、中間者攻撃に耐性を持つようなNumeric Comparisonモードの改良案を提案した。YehらのNumeric Comparisonモードはフェーズ1の公開鍵交換プロトコルとフェーズ2、フェーズ3の認証プロトコルを組み合わせて改良している。

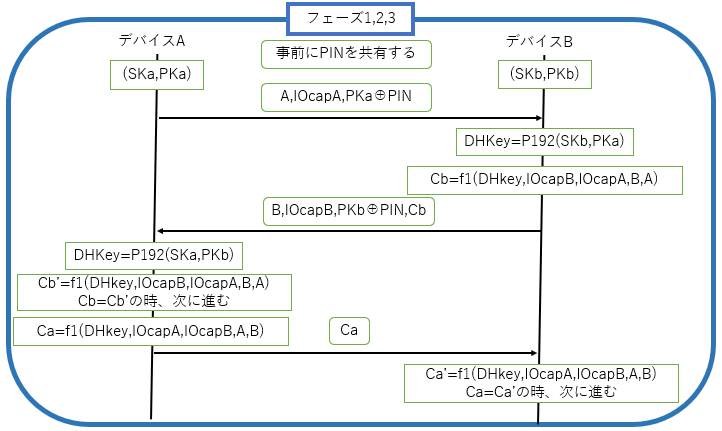


図7.YehらのNumeric Comparisonモード(改良部分)

図7にしたがってYehらのNumeric Comparisonモードを説明する。最初にデバイスA、BともにPINの値を入力し、秘密鍵と公開鍵のペア(SKa,PKa)、(SKb,PKb)を生成する。次に、デバイスAはBluetooth Device値A、デバイスAのI/O情報I/OcapA、PKa⊕PINをデバイスBに送信する。デバイスBは受け取った値PKa⊕PINに対してPINを排他的論理和し、デバイスAの公開鍵PKaを得る。自身の秘密鍵SKbとPKaを鍵生成関数P192に入力し、DHkey=P192(SKb,PKa)の生成を行う。デバイスBはDHkey、デバイスA、BのI/O情報IOcapA、IOcapBとBluetooth Device値A、Bよりハッシュ値Cb=f1(DHkey,IOcapB,IOcapA,B,A)を　計算し、Cb、B、IOcapB、PKb⊕PINをデバイスAに送信する。デバイスAは受け取った値　　　　PKb⊕PINに対して排他的論理和し、デバイスBの公開鍵PKbを得る。自身の秘密鍵SKaとPKbを鍵生成関数P192に入力し、DHkey=P192(Ska,PKb)の生成を行う。デバイスAはDHkey、デバイスA、BのI/O情報IOcapA、IOcapBとBluetooth Device値A、Bよりハッシュ値Cb’=f1(DHkey,IOcapB,IOcapA,B,A)を計算する。デバイスAはCb=Cb’を検証する。Cb≠Cb’ならデバイスAはペアリングを中断し、Cb=Cb’ならばデバイスAはデバイスBにCa=f1(DHkey,IOcapA,IOcapB,A,B)を送信する。デバイスBはハッシュ値Ca’=f1(DHkey,IOcapA,IOcapB,A,B)を計算する。Ca=Ca’を検証する。Ca≠Ca’ならばデバイスBはペアリングを中断し、Ca=Ca’ならばフェーズ4に進む。図5のようにフェーズ4でリンクキーLK=f2(DHkey,“btlk”,A,B)を計算する。ここで、btlkはプロトコル固有の定数である。フェーズ5では、このLKを用いて共有鍵Kc=E3(LK,EN\_RAND,COF)を得る。ここで、EN\_RAND、COFはプロトコル固有の定数である。

第3章　形式化

**3-1 開発環境**

本研究では、WSL(Windows Subsystem for Linux)と呼ばれる、Windows10に標準で付属するUNIX環境を土台として、その上にProVerifをインストールし開発環境とした。従来の仮想マシン上のUNIX環境と比較すると、動作がやや軽く、インストールの手順が簡略化され、ある程度PCの知識のある者であれば、ProVerifの環境構築を容易に行える点で優位である。

テキスト

自動的に生成された説明

図4.ProVerifをインストールしたWSL

**3-2 ECDL及びECDHの形式化**

鍵生成関数p192を形式化する際にECDL及びECDHの議論が必要である。そのため、ProVerif上でどのように形式化しているかを解説する。

・楕円曲線上の離散対数問題(ECDL)：既知の (P,aP)に対して、aを求めること

・楕円曲線上の計算DH問題(ECDH)：既知の(P,aP,bP)に対して、abPを求めること

Blanchetらは、Diffie-Hellman鍵共有をどのように表現するかを示し、DL及びCDHを形式化している[1]。同様の考えでECDL及びECDHについて以下のように定義する。なお、funには定義したい関数の関数名、その関数の定義域及び値域を記述し、equationにはfunで定義した関数がどのような関係を有しているかを記述する。

1. (\* P192 \*)
2. type G1.
3. type scalar.
4. const P:G1 [data].
5. fun P192(scalar,G1):G1.
6. equation forall a:scalar, b:scalar;
7. P192(a,P192(b,P)) = P192(b,P192(a,P)).P192(a,P192(b,P))=P192(b,P192(a,P)).

6行目のequationはa(bP)=b(aP)を意味している。すなわち、P、aP及びbPを知っていてもa,bのいずれかを知っていないとabPは計算できないことを意味している。これはECDHを形式化していることに他ならない。

**3-3 排他的論理和の形式化**

　YehらのNumeric Comparisonモードを形式化する際に、排他的論理和（XOR）の議論が必要となる。そのため、ProVerif上でどのように排他的論理和を形式化しているのかを解説する。

1. (\* 排他的論理和 \*)
2. fun xor(G1,G1):G1.
3. equation forall x:G1,y:G1; xor(xor(x,y),y) = x.
4. equation forall x:G1; xor(x, xor(x,x)) = x.
5. equation forall x:G1; xor(xor(x,x),x) = x.
6. equation forall x:G1,y:G1; xor(y,xor(x,x)) = y.

3行目のequationは((x⊕y)⊕y)=xを意味している。4、5行目で排他的論理和は可換であることを表している。ただし、ProVerif上では、xor(x,y)=xor(y,x)のような可換則は扱うことができない。

**3-4 宣言部**

以下に残りの宣言部を示す。

1. free c:channel.
2. free PIN:G1[private].
3. type tag.
4. const COF,EN\_RAND,btlk:tag.
5. (\* ハッシュ計算 \*)
6. type nonce.
7. type key.
8. fun hashf1(G1,tag,tag,tag,tag):nonce.
9. fun hashf2(G1,tag,tag,tag):key.
10. fun hashE3(key,tag,tag):key.
11. (\* 共有鍵暗号化 \*)
12. fun enc(bitstring, key): bitstring.
13. reduc forall x:bitstring,y:key;
14. dec(enc(x,y),y) = x.

**3-5 クエリー**

　以下にクエリーを示す。

1. (\* 秘匿クエリー \*)
2. free secretA,secretB:bitstring[private].
3. query attacker(secretA); attacker(secretB).
4. (\* 認証クエリー \*)
5. event endAparam(G1).
6. event beginAparam(G1).
7. event beginBparam(G1).
8. event endBparam(G1).
9. query x:G1; inj-event(endAparam(x)) ==>
10. inj-event(beginAparam(x)).
11. query x:G1; inj-event(endBparam(x)) ==>
12. inj-event(beginBparam(x)).
13. event endAkey(G1,G1).
14. event beginAkey(G1,G1).
15. event endBkey(G1,G1).
16. event beginBkey(G1,G1).
17. query x:G1,y:G1; inj-event(endAkey(x,y)) ==>
18. inj-event(beginAkey(x,y)).
19. query x:G1,y:G1; inj-event(endBkey(x,y)) ==>
20. inj-event(beginBkey(x,y)).
21. (\* Secrecy assumptions 秘密の過程 \*)
22. not attacker(new skA).
23. not attacker(new skB).

2-3行目は共有鍵Kcの秘匿性が成立するか検証している。ここで秘匿性とは攻撃者が秘匿対象の情報を入手できないことを意味する。つまり、攻撃者が利用可能な関数やプロセスから得られる情報を組み合わせ、秘匿対象の情報へ到達可能なパスが存在しない場合、秘匿性が成立し、Trueとなる。逆に秘匿対象の情報へ到達可能なパスが存在する場合、秘匿性が成立せず、Falseとなる。

6-22行目は認証性が成立するか検証している。ここで認証性とはメッセージの受信者はそのメッセージが攻撃者により偽造されたものではなく、本来の送信者のものであるということを意味する。ProVerifでどのようにして検証を行うか説明する。デバイスAまたはBがデバイスBまたはAから何かを受け取った(事後event)ならばその前にデバイスBまたはAがデバイスAまたはBに何かを送っている(事前event)ことが成立していると認証性は満たされている。また、ProVerifではNon-injectiveとinjectiveの2種類の検証をすることができる。

・Non-injective

　事後eventが実行された時に、それに対応する事前eventが存在しているかどうかを検証する。成立する(True)場合、同じ認証情報を用いて通信を行った送信者が存在するといえる。逆に成立しない(False)場合、正規の手順でプロトコルを実行せずにプロトコルを終了(認証成立)できることを意味する。つまり、なりすましが可能であることを意味する。

・Injective

事後eventが実行された時に、それに対応する事前eventが存在しているかどうかを検証する。さらに、攻撃者が認証情報のコピーを攻撃に利用可能かどうかの検証を行う。成立する場合、同じ認証情報を用いて通信を行った送信者が存在するといえる。加え、認証情報のコピーを用いた攻撃ができないといえる。認証情報のコピーを用いた攻撃の例として再生攻撃がある。

25-26行目はセキュリティにおける前提を示しており、攻撃者が秘密鍵skA、skBを持つことはできないことをProVerifに知らせている。

**3-6 デバイスA、デバイスBのプロセスとメインプロセス**

　以下にデバイスA、デバイスBのプロセスとメインプロセスを示す。

1. (\* プロセス部 \*)
2. let processA(skA:scalar, pkA:G1, A:tag, B:tag,IOcapA:tag, IOcapB:tag)=
3. out(c,(A,IOcapA,xor(pkA,PIN)));
4. in(c,(X:tag, IOcapB':tag, m1:G1, Cb1:nonce));
5. let pkX=xor(m1, PIN) in
6. event beginBparam(pkX);
7. let DHKeyA=P192(skA, pkX) in
8. event beginBkey(pkX,DHKeyA);
9. let Cb1'=hashf1(DHKeyA, IOcapB', IOcapA,X,A) in
10. if Cb1=Cb1' then
11. let Ca1=hashf1(DHKeyA, IOcapA, IOcapB', A,X) in
12. out(c,Ca1);
13. event endAparam(pkA);
14. let LKA=hashf2(DHKeyA, btlk, A, X) in
15. event endAkey(pkA, DHKeyA);
16. let KcA=hashE3(LKA, EN\_RAND, COF) in
17. out(c,enc(secretA, KcA)).
18. let processB(skB:scalar, pkB:G1, A:tag, B:tag,IOcapA:tag, IOcapB:tag)=
19. in(c,(Y:tag, IOcapA':tag, m0:G1));
20. let pkY=xor(m0, PIN) in
21. event beginAparam(pkY);
22. let DHKeyB=P192(skB, pkY) in
23. event beginAkey(pkY, DHKeyB);
24. let Cb1=hashf1(DHKeyB, IOcapB, IOcapA', B,Y) in
25. out(c,(B, IOcapB, xor(pkB,PIN), Cb1));
26. in(c, Ca1:nonce);
27. let Ca1'=hashf1(DHKeyB, IOcapA', IOcapB,Y,B) in
28. if Ca1=Ca1' then
29. event endBparam(pkB);
30. let LKB=hashf2(DHKeyB, btlk, Y, B) in
31. event endBkey(pkB, DHKeyB);
32. let KcB=hashE3(LKB, EN\_RAND, COF) in
33. out(c,enc(secretB, KcB)).
34. (\* main \*)
35. Process
36. new skA:scalar;
37. let pkA=P192(skA,P) in
38. new IOcapA:tag;
39. out(c, IOcapA);
40. new A:tag;
41. out(c,A);
42. new skB:scalar;
43. let pkB=P192(skB,P) in
44. new IOcapB:tag;
45. out(c, IOcapB);
46. new B:tag;
47. out(c,B);
48. ((!processA (skA, pkA,A,B, IOcapA, IOcapB))|
49. (!processB (skB, pkB,A,B, IOcapA, IOcapB)))

デバイスAとデバイスBのプロセスは2-17行目と19-34行目に示している。メインプロセスは37-51行目に示している。

第4章　検証結果

　　YehらのNumeric ComparisonモードをProVerifにより形式化し、先行研究[8]と同様の検証を行った。共通鍵Kcが秘匿性を満たしているか検証した。認証性については(1)参加者が誰であるか(以降、Weak Authentication)、(2)参加者が誰であるかと鍵を誰が生成したか(以降、Strong Authentication)を検証した。

検証結果は以下の表1、2に示す。

表1.秘匿性の検証結果

|  |  |
| --- | --- |
| Property | Result |
| Secret A | True |
| Secret B | True |

表2.認証性の検証結果

|  |  |  |  |
| --- | --- | --- | --- |
| Property | | | Result |
| Strong  Authentication | Injective | A-to-B | False |
| B-to-A | False |
| Non-injective | A-to-B | True |
| B-to-A | False |
| Weak  Authentication | Injective | A-to-B | False |
| B-to-A | False |
| Non-injective | A-to-B | True |
| B-to-A | False |

表1より、共有鍵Kcが秘匿性を満たしていることを確認した。表2より認証性は、AtoB(デバイスAによるデバイスBの認証)の場合、Non-injectiveが成立している。ただし、Injectiveは成立していないため、再生攻撃を受ける。BtoA(デバイスBによるデバイスAの認証)の場合はNon-injectiveもInjectiveも成立していないため、なりすましが可能である。この結果は先行研究[3]で報告されている結果と一致した。

第5章　Passkey Entryモードを形式化する際の見通し

Passkey Entryモードを形式化する上で考えられる難しさを説明する。フェーズ1、3、4、5は同じであるから変化するところはフェーズ2だけである。Numeric Comparisonモードと異なる点は、入力した10進数６桁の値(20bit)のノンスを1bitずつ計算に用いて、図3の流れを20回繰り返す点であり、これがProVerifで形式化する上で、難しいと考えられる。また、先行研究[6]よりPasskey Entryモードにはなりすましの攻撃法が確認されている。さらに2-2に示したようにフェーズ１に脆弱性があり、その脆弱性がPasskey Entryモードでも影響している可能性があると考えられる。よって改良されたPasskey Entryモードにフェーズ１の脆弱性の対策、フェーズ2のなりすましへの対策を講じていると考えられるが、YehらのNumeric Comparisonモードのように一部だけ修正してもまだなりすましの可能性が排除できない。それに加え、再生攻撃も可能であるのではないかと予想される。

第6章　まとめ

BluetoothのSSPのYehらのNumeric Comparisonモードの形式化を行い、ProVerifを用いて安全検証を行った。その結果、先行研究[3]と同様の結果を得られた。また、改良されたPasskey Entryモードに向けて形式化する上での難しさや予想される攻撃法について考察した。

　今後は改良されたPasskey Entryモードについて調査し、第五章のPasskey Entryモードを形式化する上で考えられる難しさの対策を講じた上でProVerifを用いて形式化し、検証することを目標としたい。

謝辞

　本研究を進めるにあたり、多大なるご指導，ご助力を頂いた指導教員の塚田恭章教授には深く感謝いたします。また、ProVerifのインストール環境の構築を手伝ってくださった先輩、相談に応じて頂いたネットワークセキュリティ研究室の皆様にも深く感謝いたします。ここで深謝の意を表し、謝辞とさせて頂きます。

参考文献

[1] Bruno Blanchet：“ProVerif”,  
<https://prosecco.gforge.inria.fr/personal/bblanche/proverif/>

[2] T.C.Yeh, J.R.Peng, S.S.Wang, and J.P.Hsu：

“Securing Bluetooth Communications”, International Journal of Network Security,

Vol.14, No.4, pp.229-235, July 2012

[3] 横山 雄太, 岩本 智裕, 荒井 研一, 金子 敏信：

“ProVerifを用いたBluetoothのセキュアシンプルペアリングの形式的検証”, Computer Security Symposium 20,2013

[4] Bluetooth SIG,Bletooth2.1+EDR Core

Specification,BluetoothSIG,2007

[5] R.Chang and V.shimatikov: Formal Analysis of Authentication in Bluetooth Device Paring, Proc. Joint Workshop on Foundations of Computer Security and Automated Reasoning for Security Protocol Analysis (FCS-ARSPA 2007),pp.45-61(2007)

[6] 井上 博之, 荒井 研一, 金子 敏信： “ProVerifによるBluetoothのセキュアシンプルペアリングに対する形式的な安全検証”,日本応用数理学会2013年春の研究部会連合発表会「数理的技法による情報セキュリティ」(FAIS)セッション, 2013

[7] Da-Zhi Sun, Yi Mu, Willy Susilo：“Man-in-the-

middle attacks on Secure Simple Pairing in

Bluetooth standard V5.0 and its countermeasure”, Personal and Ubiquitous Computing 22, 55-67 (2018)

[8] E.Uzum, K.Karvonen, and N.Asokan：“Usability Analysis of Secure Pairing Methods”, Nokia Research Center Technical Reports 2007

<http://research.nokia.com/tr/NRC-TR-2007-002.pdf>