Computer Security Symposium 2013
21 - 23 October 2013

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

横村　雄太†　　岩本　智裕†　　荒井　研一†　　金子　敏信†

†東京理科大学
278-8510 千葉県野田市山崎 2641
{j7312669,j7313615}@ed.tus.ac.jp, k.arai@rs.tus.ac.jp, kaneko@ee.noda.tus.ac.jp

あらまし ProVerifは Blanchet らが開発した形式モデルでの自動検証ツールであり，暗号プロトコルで要求される秘匿や認証などの安全性を検証可能である．一方，Bluetooth において端末間で相互認証を行う手順をペアリングと呼び，Bluetooth2.1 + EDR にてペアリングの方式にセキュアシンプルペアリング (SSP) が追加された．しかしながら，Chang らによって ProVerif を用いた安全性検証で SSP のモードの 1 つである Numeric Comparison モードに対する攻撃法が確認されている．また，Yeh らは Numeric Comparison モードの改良案を提案している．本研究では Yeh らの Numeric Comparison モードについて ProVerif を用いて形式的に記述し，安全性の検証を行う．さらに検証結果についての考察及び攻撃法の対策について議論する．

Formal Verification of Secure Simple Paring in Bluetooth Using ProVerif

Yuta Yokomura†　　Tomohiro Iwamoto†　　Kenichi Arai†　　Toshinobu Kaneko†

†Tokyo University of Science
2641 Yamazaki, Noda, Chiba 278-8510, JAPAN
{j7312669,j7313615}@ed.tus.ac.jp, k.arai@rs.tus.ac.jp, kaneko@ee.noda.tus.ac.jp

Abstract ProVerif is an automatic cryptographic protocol verifier based on the formal model developed by Blanchet et al. This verifier can verify the properties of secrecy and authentication,etc. Meanwhile, the mutual authentication procedure between Bluetooth devices is called pairing. Secure Simple Pairing (SSP) is a new method of the pairing specified in Bluetooth2.1 + EDR. However, Chang and Shmatikov proposed the attack against the Numeric Comparison mode for SSP using ProVerif. Conversely, Yeh et al. proposed the improvement protocol of this mode. In this paper, we describe this improvement protocol using ProVerif and verify the security properties. This paper also discusses the countermeasures against the attack of improvement protocol.

1　序論

暗号プロトコルの自動検証ツールとしてProVerif[1] がある．ProVerifは Blanchet らが開発した形式モデルでの自動検証ツールであり，暗号プロトコルで要求される秘匿や認証などの安全性 (要件) を検証可能である．一方，Bluetooth は情報端末間または情報端末とキーボードなどの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を行う手順をペアリングと呼ぶ．2004 年に登場した Bluetooth2.0+EDR[2] ではペアリングは PIN を用いた方式であった．しかし一般的な PIN によるペアリングの脆弱性は 2001 年に Jakobsson らによって攻撃法が提案されている [3]．そして，この提案を基に 2005 年に Shaked らによって Bluetooth の PIN によるペアリングへの攻撃法が実装された [4]．PIN によるペアリングの脆弱性に対応したのが 2007 年に Bluetooth2.1 + EDR[5] で追加されたセキュアシンプルペアリング (SSP) である．しかしながら，Chang らによって ProVerif を用いた安全性検証で SSP のモードの 1 つである Numeric Comparison モードに対する攻撃法が確認されている [6]．さらに，著者らはその他のモードに対する研究として，ProVerif を用いた安全性検証で Passkey Entry モードに対する攻撃法を確認している [7]．また，Numeric Comparison モードの問題点に対して，Yeh らは Numeric Comparison モードの改良案を提案している [8]．本研究では Yeh らの Numeric Comparison モードについて ProVerif を用いて形式的に記述し，安全性の検証を行う．さらに検証結果についての考察及び攻撃法の対策について議論する．

2 ProVerif

ProVerif は通信路に流れるメッセージを記号に理想化して暗号プロトコルの検証を行う形式手法 (いわゆる，Dolev-Yao モデル [9]) を用いて，暗号プロトコルの安全性を検証している．ProVerif では π 計算を任意の代数で拡張した拡張 π 計算 [10](または Horn 節の集合 [11]) を用いて暗号プロトコルを記述する．さらに，ProVerif は拡張 π 計算を用いて記述された暗号プロトコルを Horn 節の集合に変換する．ある事実 (安全性要件に反する事実など) が Horn 節の集合から導出できないとき，暗号プロトコルの安全性要件が証明される．ProVerif はこの非導出性を証明するために，解決ベースアルゴリズムを用いている．さらなる詳細については [11] を参照されたい．本章では ProVerif における形式化及び，秘匿や認証といった ProVerif で検証可能な安全性について解説する．

2.1 ProVerif による検証

本節では ProVerif における暗号プロトコルの検証方法について説明する．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探索することによって行う．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を導出できた場合，その暗号プロトコルは攻撃可能，すなわちその暗号プロトコルは安全性要件を満たさない．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件を満たす．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，安全性要件などを記述し，プロセス部は暗号プロトコルそのものを記述する．

2.2 秘匿

秘匿には以下の 2 種類がある [12]．

- 秘匿 (Standard Secrecy)

攻撃者が秘匿対象の情報を入手可能か否かを検証する

- 強秘匿 (Strong Secrecy)

「秘匿対象の情報を暗号化した暗号文」と「その暗号文とは何ら関係ない乱数」との違いを攻撃者が観測可能か否かを検証する．

このように，ProVerif は形式的に記述された暗号プロトコルから，攻撃者が利用可能な規則や関数，プロセスから得られる出力を組み合わせ，秘匿対象の情報に到達可能なパスを導出できた場合，Standard Secrecy が保たれていないとしている (False)．逆に，秘匿対象の情報へ到達可能なパスが存在しない場合，Standard Secrecy は保たれているとしている (True)．

なお，強秘匿に関しては，本論文では扱わないため説明は省略する．強秘匿についての詳細は [13] を参照されたい．

2.3 認証

秘匿以外に検証可能な安全性として認証がある [14]．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なく，本来の

送信者のものであるということを確認できるならば，認証が成立しているといえる．

次に，ProVerif がどのようにして認証の検証を行うのかを説明する．認証が成立しているということは，送信者を特定するための情報を受信者が正しく受け取っているということである．つまり，送信者自身が生成，送信した認証用情報 (パラメータ) と，受信者が受け取った認証用情報が一致するならば，認証が成立しているといえる．ProVerif での認証は，通信プロセス中で認証用情報が生成された時を「事前 event」，確認した時を「事後 event」として定義し，事後 event が実行された際に，それに対応する事前 event が存在しているか否かを検証する．ProVerif では次の 2 種類を検証することができる．

- Non-Injective

事後 event が実行された際に，それに対応する事前 event が存在しているか否かを検証する．Non-injective が成立 (True) すれば，同じ認証用情報を用いて通信を行った送信者が存在するといえる．なお，Non-injective が成立しない (false) の場合は，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) できる事を意味する．これは，すなわちなりすましが可能であることを意味している．

- Injective

事後 event が実行された際に，それに対応する事前 event が存在しているか否かを検証する．さらに攻撃者が認証用情報のコピーを攻撃に利用可能か否かの検証を行う．Injective が成立すれば，同じ認証用情報を用いて通信を行った送信者が存在することが分かる．さらに認証用情報のコピーを用いた攻撃ができないといえる．認証用情報のコピーを用いた攻撃の例として，再生攻撃がある．

3 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の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 端末どうし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 Bluetooth2.0 ＋ EDR まではPIN を用いた方式のみであったが，Bluetooth2.1 ＋ EDR においてセキュアシンプルペアリング（以下では SSP と呼ぶ）が追加された．PIN によるペアリングの脆弱性に対応したのが，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．しかし，2 つのデバイスそれぞれの画面上に表示された 6 桁の数字が同じならば Yes，違っていれば No 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 人が数字が違っているのにも関わらず Yes を押したという結果がある [15]．そのため Tzu-Chang Yeh らは人間の目による確認は人為的ミスが起こる可能性があるとし，Numeric Comparison モードの改良案を提案した [7]．

SSP は 5 つのフェーズからなる．フェーズ 1 では楕円曲線 Diffie 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)．フェーズ 3 では共有したすべ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 3)．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．各フェーズの詳細については [5][8] を参照されたい．

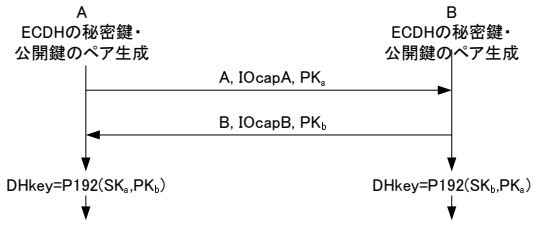

図 1: 公開鍵交換

フェーズ 2 には前述の Numeric Comparison, Just Works, Out of Band,Passkey Entry 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 によって使用モードが選択される．また，Yeh らの提案した Numeric

Comparison モードは，フェーズ 1 とフェーズ 2，フェーズ 3 を組み合わせ，改良したものとなっている．

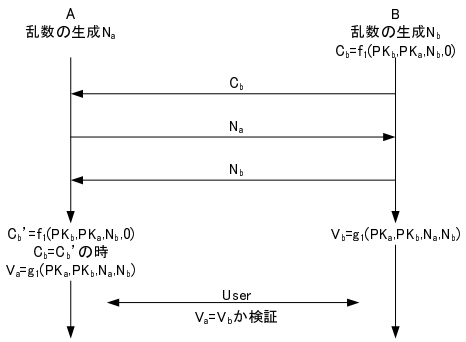

図 2: Numeric Comparison モード (フェーズ 2)

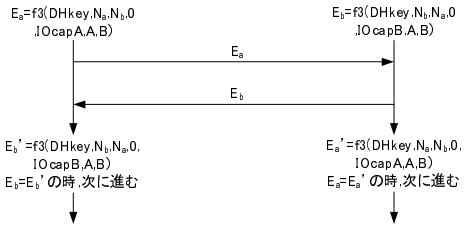

図 3: フェーズ 3

3.0.1 Yeh らの Numeric Comparison モードの流れ

本節では，Yeh らの Numeric Comparison モードについて説明する．Yeh らの NumericComparison モードは中間者攻撃に対し，耐性を持つように改良された．このモードはフェーズ 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改良したものである．図 1 に従って，Yeh らの NumericComparison モードの手順を説明する．端末 A,B ともに PIN の値を入力する．次に，端末 A は BD 値 (Bluetooth Device 値)A，端末 A の I/O 情報 $IOcapA, PKa \oplus PIN$ を端末 B に送信する．端末 B は受け取った値 $PKa \oplus PIN$ に対して PIN を排他的論理和し，端末 A の公開鍵 PK_a を得る．自身の秘密鍵 SK_b と PK_a に対して，鍵生成関数 $P192$ に入力し，$DHkey$ の生成を行う．

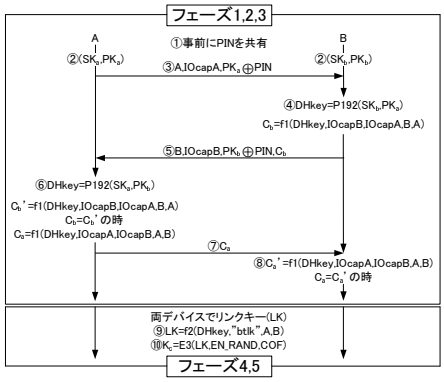

図 4: Yeh らの Numeric Comparison モード

$$B : DHkey = P192(SK_b, PK_a)$$

鍵生成関数 $P192$ の仕様については [5] を参照されたい．端末 B は $DHkey$，端末 A,B の I/O 情報 $IOcapA, IOcapB$ と BD 値 A, B に対しハッシュ値

$$C_b = f1(DHkey, IOcapB, IOcapA, B, A)$$

を計算し、$Cb, B, IOcapB, PK_b \oplus PIN$ を端末 A に送信する．ハッシュ関数の仕様については [5] を参照されたい．端末 A は受け取った値 $PK_b \oplus PIN$ に対して PIN を排他的論理和し，端末 B の公開鍵 PK_b を得る．自身の秘密鍵 SK_a と PK_b に対して，鍵生成関数 $P192$ に入力し，$DHkey$ の生成を行う．

$$A : DHkey = P192(SK_a, PK_b)$$

端末 A は $DHkey$，端末 A,B の I/O 情報 $IOcapA, IOcapB$ と BD 値 A,B に対しハッシュ値

$$C'_b = f1(DHkey, IOcapB, IOcapA, B, A)$$

を計算する．端末 A は $C_b = C'_b$ を検証する．$C_b \neq C'_b$ なら端末 A はペアリングをアボートし，$C_b = C'_b$ ならば端末 A は端末 B に C_a を送信する．

$$C_a = f1(DHkey, IOcapA, IOcapB, A, B)$$

端末 B は $DHkey, IOcapA, IOcapB$ と A, B に対しハッシュ値

$$C'_a = f1(DHkey, IOcapA, IOcapB, A, B)$$

を計算する．端末 B は $C_a = C'_a$ を検証する．$C_a \neq C'_a$ なら端末 A はペアリングをアボートし，$C_a = C'_a$ ならフェーズ 4 に進む．フェーズ 4 でリンクキーの計算はハッシュ関数 $f2$ を用いて，

$$LK = f2(DHkey, "btlk", A, B)$$

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，ハッシュ値 $E3$ を用いてハッシュ値

$$K_c = E3(LK, EN\_RAND, COF)$$

を得る．ここで，EN_RAND, COF はプロトコル固有の定数である．

4 SSP の形式的記述について

本節では，ProVerif 上で Yeh らの Numeric Comparison モードをどのように形式化するかを解説する．

4.1 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 の議論が必要となる．そのため，ProVerif 上でどのように ECDL 及び ECDH を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形式化する．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. fun P192(scalar, G1):G1
2. equation forall a: scalar, b: scalar
   P192(a,P192(b,P))=P192(b,P192(a,P))

2 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．さらに，1 行目の fun の記述そのものが $P192(a, P) = aP$ を意味していることがわかる．すなわち，P 及び aP を知っていても a を知らないと aP は計算できないことを意味している．

4.2 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. fun xor(bitstring,bitstring) : bitstring.
2. equation forall x : bitstring, y : bitstring;
   xor(xor(x,y),y)=x.
3. equation forall x : bitstring;
   xor(x,x)=zero.
4. equation forall x : bitstring;
   xor(zero,x)=x.
5. equation forall x : bitstring;
   xor(x,zero)=x

2 行目の $equation$ は $((x \oplus y) \oplus y) = x$ を意味している．3 行目の $zero$ は $bitstring$ 長の 0 を意味し，x 同士を排他的論理和したものは 0 になる事を表している．4,5 行目では x と 0 の排他的論理和は可換である事を表している．ただし，ProVerif 上では，$xor(x, y) = xor(y, x)$ のような可換則は扱う事が出来ない．

5 検証結果

Yeh らの Numeric Comparison モードを ProVerif により形式化し，検証を行った結果，秘匿，認証について以下の結果となった．秘匿については共通鍵 K_c が秘匿性を満たしているかを検証した．認証については (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を検証した (以降，Strong Authentication)．

5.1 Yeh らの Numeric Comparison モードの検証結果

表 1 より，Yeh らの Numeric Comparison モードが秘匿性を満たすことを確認した．これは，共

通鍵 K_c が秘匿性を満たしていることを意味する．表 2 より認証は，$AtoB$(端末 A による端末 B の認証) の場合，$Non-injective$ が成立している．ただし，$Injective$ は成立しておらず，再生攻撃を受ける．$BtoA$(端末 B による端末 A の認証) の場合は $Non-injective$ も $Injective$ も成立していない．すなわち，なりすましが可能である．

Property	Result
Secret A	True
Secret B	True

表 1：秘匿の検証結果

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 2：認証の検証結果

6 ProVerifにより検証された攻撃方法

本章では，Yeh らの Numeric Comparison モードに対して攻撃者がどのように「再生攻撃」,「なりすまし」を行っているのかを記述する．

6.1 再生攻撃

$AtoB$ 認証について調べた結果，複数の事後 event に対して，同一の事前 event が実行されることが確認された．すなわち，再生攻撃を受ける．端末 A は $(A, IOcapA, PK_a \oplus PIN)$ の情報を端末 B に送る．この情報は毎回同じ情報を送っている．よって，攻撃者はこの情報得て，それを多量に B に送りこみ，リソースに負荷を与え，動作を停止させるといった DoS 攻撃が可能である．

6.2 なりすまし

$BtoA$ 認証について調べた場合，事後 event が確認された時に，それに対応する事前 event が存在しないパスが確認された．よって，攻撃者は端末 A へのなりすましが可能である．以下になりすましの手順を記述する．端末 A は $(A, IOcapA, PK_a \oplus PIN)$ の情報を端末 B に送る．攻撃者 M はこの情報を盗み，$(B, IOcapB, m_M)$ に変え，端末 B に送る．ここで，m_M は乱数である．また，端末 B はハッシュ値 $C_b = f1(P192(SK_b, xor(m_M, PIN)), IOcapB, IOcapB, B, B)$ を作り，$C_b, B, IOcapB, PK_b \oplus PIN$ を端末 A に送る．M はこのやりとりを遮断し，C_b を端末 A に送信する．端末 A は受信した C_b と自身で生成した C'_b が一致しないことから，通信をアボートする．一方，M は端末 B から得た C_b をそのまま B に送信する．端末 B は端末 A から送られてきた C_a だと思い，C'_a を計算する．

$$B : C'_a = f1(P192(SK_b, xor(m_M, PIN)), IOcapB, IOcapB, B, B)$$

この時，$C_b = C'_a$ になることからなりすましが可能である．

7 攻撃法への対策

本章では，前章で検証された攻撃方法への対策について記述する．

7.1 再生攻撃対策

$DHkey$ を生成するために，端末 A から端末 B に送る情報 $(A, IOcapA, PK_a \oplus PIN)$ を X とし，端末 B から端末 A に送る情報 $(B, IOcapB, PK_b \oplus PIN, C_b)$ を Y とする．PK_a と PK_b が同じで $DHkey$ が毎回変わらないことが原因で再生攻撃が出来るため，$DHkey$ の作り方を毎回変えることで対策をする．PK_a, PK_b の値は変えないという前提で，解説する．

以下に対策の流れを示す．

端末 A は乱数 N_a を生成し，公開鍵 PK_a と連結し，$(A, IOcapA, (PK_a, N_a) \oplus PIN)$(以降 X') を端末 B に送信する．ただし，ProVerif では (a, b) で a と b の連結を意味する．

端末 B は乱数 N_b を生成し，ハッシュ関数 $(hash)$ と受信した X' から得た PK_a と N_a を用いて $DHkey(= hash(N_a, N_b, P192(SK_b, PK_a)))$ を生成する．そして生成した $DHkey$ を用いて，ハ

ッシュ値$C_b(= f1(DHkey, IOcapB, IOcapA, B, A))$を生成する．その後，端末Bは$N_b$と$PK_b$を連結し，$(B, IOcapB, (PK_b, N_b) \oplus PIN, C_b)$(以降Y')を端末Aに送信する．

Y'を受信した端末AはY'から得たPK_aとN_bを用いて，$DHkey(= hash(N_a, N_b, P192(SK_a, PK_b)))$を生成する．

上記対策の流れの説明をする．乱数と公開鍵を連結したビット列とPINを排他的論理和したものを送ることで，乱数を端末Bに送る事が出来る．そして端末A,Bが生成した乱数は同じビット数なので，受信した(PK_a, N_a)からPK_aとN_aを取り出す事が可能である．$DHkey$をハッシュ関数と乱数を用いて生成することで，毎回$DHkey$を変える事が出来る．

7.2 なりすまし対策

端末Bに対して$B, IOcapB$が送ることが可能で，この情報を使って$DHkey$が生成されることが，なりすましの原因なので，端末Bに入力される情報が端末B自身のBD値，I/O情報かどうかをチェックする．以下に対策の流れを示す.

端末Bに対して$B, IOcapB$が入力された場合，ペアリングをアボートし，$B, IOcapB$以外ならばペアリングを続ける．

7.3 対策後の検証結果

対策後のYehらのNumeric Comparisonモードを ProVerifにより形式化 (付録) し，検証を行った結果，5.1節の表1,2の結果がすべて$True$となった．これは，すなわち再生攻撃もなりすましも不可能になった事を意味する．

8 まとめ

本稿では，Numeric Comparisonモードの改良案として提案されている，Yehらの Numeric Comparisonモードに対して ProVerifによる安全性の検証を行った．ProVerifによる検証を行うにあたり，ECDL，ECDH，排他的論理和の形式化を行った．検証の結果，再生攻撃となりすましが可能である事を確認したため，Yehらの Numeric Comparisonモードに対して，再生攻撃となりすまし対策を行った．その結果，再生攻撃となりすましが不可能になる事を確認した．

参考文献

[1] B.Blanchet, B.Smyth and V.Cheval(2013) ProVerif 1.87beta6: Automatic Cryptographic Protocol Verifier, User Manual and Tutorial.

[2] Bluetooth SIG,Bluetooth2.0+EDR Core Specification,BluetoothSIG,2004

[3] M.jakobsson,and S.Wetzel: Security weakness in Bluetooth, Topics in Cryptology - CT-RSA2001,LNCS2020,pp176-191, Springer(2001).

[4] Y.Shaked, and A.Wool:Cracking the Bluetooth PIN, Proc. 3rd USENIX/ACM Conf. Mobile Systems, Applications, and Services (MobiSys2005),pp.30-50(2005).

[5] Bluetooth SIG,Bluetooth2.1+EDR Core Specification,BluetoothSIG,2007

[6] 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-ARSPA2007),pp.45-61(2007).

[7] 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性検証,"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.

[8] 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.

[9] D.Dolev, and A.C.Yao."On the security of public key protocols." IEEE Transactions on Information Theory. 29, 2, 1983, p198-207.

[10] M.Abadi and C.Fournet,"Mobile Values, New names, and Secure Communication,"Proceedings of the 28th ACM SIGPLAN-SIGACT Symposium on Principles of Programming Languages(POPL'01), pp.104-115, 2001.

[11] B.Blanchet,"From Secrecy to Authenticity in Security Protocols,"9th International Static Analysis Symposium(SAS'02), volume 2477 of Lecture Note on Computer Science, pp.342-359, 2002.

[12] B.Blanchet ”An Efficient Cryptographic Protocol Verifer Based on Prolog Rules, ” In14th IEEE Computer Security Foundations Workshop,pp.82-96,2001.

[13] B.Blanchet, ”Automatic Proof of Strong Securicy for Security Protocols, ”In IEEE Symposium on Security and Privacy,pp.86-100,2004.

[14] B.Blanchet, ”From Secrecy to Authenticity in Security Protocol, ”9th International Static Analysis Symposium,volume 2477 of Lecture Notes on Computer Science,pp342-359,2002.

[15] E.Uzun, 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

付録

対策後の Yeh らの Numeric Comparison モードにおける，ProVerif のコードをプロセス部のみ以下に記述する．

```
let processA(skA:scalar, pkA:G1, A:tag, B:tag,
             IOcapA:tag, IOcapB:tag)=
    new Na:nonce;
    out(c,(A,IOcapA,xor((pkA,Na),PIN)));
    in(c,(X:tag, IOcapB':tag, m1:bitstring,
          Cb1:nonce));
    if X<>A then
    if IOcapB'<>IOcapA then
    let (pkX:G1, Nb':nonce)=xor(m1, PIN) in
    event beginBparam(pkX);
    let DHKeyA=hash(Na, Nb', P192(skA,pkX)) in
    event beginBkey(pkX,DHKeyA);
    let Cb1'=hashf1(DHKeyA, IOcapB', IOcapA,X,
                    A) in
    if Cb1=Cb1' then
    let Ca1=hashf1(DHKeyA, IOcapA, IOcapB', A,
                   X) in
    out(c,Ca1);
    event endAparam(pkA);
    let LKA=hashf2(DHKeyA, btlk, A, X) in
    event endAkey(pkA, DHKeyA);
    let KcA=hashE3(LKA, EN_RAND, COF) in
    out(c,enc(secretA, KcA)).


let processB(skB:scalar, pkB:G1, A:tag, B:tag,
             IOcapA:tag, IOcapB:tag,)=
    in(c,(Y:tag, IOcapA':tag, m0:bitstring));
    if Y<>B then
    if IOcapA'<>IOcapB then
    let (pkY:G1, Na':nonce)=xor(m0, PIN) in
    event beginAparam(pkY);
    new Nb:nonce;
    let DHKeyB=hash(Na', Nb, P192(skB,pkY)) in
    event beginAkey(pkY, DHKeyB);
    let Cb1=hashf1(DHKeyB, IOcapB, IOcapA', B,
                   Y) in
    out(c,(B, IOcapB, xor((pkB,Nb),PIN),Cb1));
    in(c,Ca1:nonce);
    let Ca1'=hashf1(DHKeyB, IOcapA', IOcapB,Y,
                    B) in
    let Ca1=Ca1' then
    event endBparam(pkB);
    let LKB=hashf2(DHKeyB, btlk, Y, B) in
    event endBkey(pkB, DHKeyB);
    let KcB=hashE3(LKB, EN_RAND, COF) in
    out(c,enc(secretB, KcB)).

  process
    new skA:scalar;
    let pkA=P192(skA,P) in
    new IOcapA:tag;
    out(c,IOcapA);
    new A:tag;
    out(c,A);
    new skB:scalar;
    let pkB=P192(skB,P) in
    new IOcapB:tag;
    out(c,IOcapB);
    new B:tag;
    out(c,B);
  ((!processA(skA, pkA,A,B, IOcapA, IOcapB,))|
   (!processB(skB, pkB,A,B, IOcapA, IOcapB)))
```