# EAP-SIM の ProVerif による評価結果

国立研究開発法人 情報通信研究機構

## 1. 基本情報

- ✧ 名前

  Extensible Authentication Protocol Method for Global System for Mobile Communications (GSM) Subscriber Identity Modules (EAP-SIM)

- ✧ 機能

  FDD-TDMA 方式で実現されている第二世代携帯電話規格 (GSM) における SIM を用いた相互認証・鍵交換プロトコル。

- ✧ 関連する標準

  RFC4186 (https://www.ietf.org/rfc/rfc4186.txt)

## 2. ProVerif の文法による記述

### 2.1. プロトコル仕様

```
(*
set ignoreTypes = yes.

A3 と A8 に同じ関数(同じ情報を出力する関数)を用いた場合は
鍵の秘匿性が保証されない.
*)

free c: channel.

type key.
type nonce.
type host.

const Request: bitstring [data].
const Response: bitstring [data].
```

```
const Success: bitstring [data].
const Identity: bitstring [data].
const Start: bitstring [data].
const SIM: bitstring [data].
const Challenge: bitstring [data].

free hostP, hostA: host.

table keys(host, host, key).

(* *)
fun A3(key, nonce): bitstring.
(*
fun A8(key, nonce): bitstring.
*)
reduc forall k: key, n: nonce;
      A8(k, n) = A3(k, n).

fun SHA1(bitstring): bitstring.
fun PRF_aut(bitstring): key.
fun PRF_encr(bitstring): key.



(* MAC *)
fun mac(bitstring, key): bitstring.
reduc forall k: key, x: bitstring;
      checkmac(mac(x, k), k) = x.
reduc forall k: key, x: bitstring;
      getmsg(mac(x, k)) = x.

(* SKE for nonce *)
fun encrypt(nonce, key): bitstring.
```

```
reduc forall x: nonce, k: key;
      decrypt(encrypt(x, k), k) = x.

(**)
(*
fun n2b(nonce): bitstring [data, typeConverter].
fun n2k(nonce): key [data, typeConverter].
*)

(* events *)
event beginPeer(host, key, key).
event endPeer(host, key, key).
event beginAuthenticator(host, key, key).
event endAuthenticator(host, key, key).

event end().
(**)

free secretPeerK_aut, secretPeerK_encr,
     secretAuthenticatorK_aut, secretAuthenticatorK_encr,
     secretTest: nonce [private].

(*
not attacker(new KEKpa).
not attacker(new KCKpa).
*)

(* queries *)
query attacker(secretPeerK_aut).
query attacker(secretPeerK_encr).
query attacker(secretAuthenticatorK_aut).
query attacker(secretAuthenticatorK_encr).
```

```
query p: host, k_aut: key, k_encr: key;
      inj-event(endAuthenticator(p, k_aut, k_encr)) ==>
      inj-event(beginPeer(p, k_aut, k_encr)).

(*
query sid: nonce, p: host, a: host, ena: bitstring, enp: bitstring;
      inj-event(endPeer(sid, p, a, ena, enp)) ==>
      inj-event(beginAuthenticator(sid, p, a, ena, enp)).
*)

query attacker(secretTest).


let procPeer(IMSI: host, Ki: key) =
    in(c, AT_SELECTED_VERSION: bitstring);
    (* Request/Identity *)
    in(c, (=Request, =Identity));
    (* Response/Identity *)
    out(c, (Response, Identity, IMSI));
    (* Request/SIM/Start *)
    in(c, (=Request, =SIM, =Start, AT_VERSION_LIST: bitstring));
    (* Response/SIM/Start *)
    if AT_SELECTED_VERSION = AT_VERSION_LIST then
    new AT_NONCE_MT: nonce;
    out(c, (Response, SIM, Start, AT_NONCE_MT, AT_SELECTED_VERSION));
    (* Request/SIM/Challenge *)
    in(c, (=Request, =SIM, =Challenge, AT_RAND: bitstring, AT_MAC:
bitstring));
    let (RAND1: nonce, RAND2:nonce) = AT_RAND in
    let SRES1 = A3(Ki, RAND1) in
    let SRES2 = A3(Ki, RAND2) in
```

```
    let Kc1 = A8(Ki, RAND1) in
    let Kc2 = A8(Ki, RAND2) in
    let MK = SHA1((IMSI, Kc1, Kc2, AT_NONCE_MT,
                   AT_VERSION_LIST, AT_SELECTED_VERSION)) in
    let (K_aut: key, K_encr: key) = (PRF_aut(MK), PRF_encr(MK)) in
    if (Request, SIM, Challenge, AT_RAND, AT_NONCE_MT)
       = checkmac(AT_MAC, K_aut) then
    (* Response/SIM/Challenge *)
    event beginPeer(IMSI, K_aut, K_encr);
    out(c, (Response, SIM, Challenge,
            mac((Response, SIM, Challenge, SRES1, SRES2), K_aut)));
    (* EAP-Success *)
    in(c, =Success);
    (* For security analysis *)
    event endPeer(IMSI, K_aut, K_encr);
    out(c, encrypt(secretPeerK_aut, K_aut));
    out(c, encrypt(secretPeerK_encr, K_encr));
    event end().

let procAuthenticator =
    in(c, AT_VERSION_LIST: bitstring);
    (* Request/Identity *)
    out(c, (Request, Identity));
    (* Response/Identity *)
    in(c, (=Response, =Identity, IMSI: host));
    get keys(=IMSI, =hostA, Ki) in
    (* Request/SIM/Start *)
    out(c, (Request, SIM, Start, AT_VERSION_LIST));
    (* Response/SIM/Start *)
    in(c, (=Response, =SIM, =Start, AT_NONCE_MT: nonce,
           AT_SELECTED_VERSION: bitstring));
    if AT_SELECTED_VERSION = AT_VERSION_LIST then
```

```
    (* Request/SIM/Challenge *)
    new RAND1: nonce;
    new RAND2: nonce;
    let AT_RAND = (RAND1, RAND2) in
    let SRES1 = A3(Ki, RAND1) in
    let SRES2 = A3(Ki, RAND2) in
    let Kc1 = A8(Ki, RAND1) in
    let Kc2 = A8(Ki, RAND2) in
    let MK = SHA1((IMSI, Kc1, Kc2, AT_NONCE_MT,
                   AT_VERSION_LIST, AT_SELECTED_VERSION)) in
    let K_aut  = PRF_aut(MK) in
    let K_encr = PRF_encr(MK) in

    out(c, (Request, SIM, Challenge, AT_RAND,
            mac((Request, SIM, Challenge, AT_RAND, AT_NONCE_MT),
K_aut)));
    (* Response/SIM/Challenge *)
    in(c, (=Response, =SIM, =Challenge, AT_MAC2: bitstring));
    if (Response, SIM, Challenge, SRES1, SRES2) = checkmac(AT_MAC2,
K_aut) then
    (* EAP-Success *)
    out(c, Success);
    (* For security analysis *)
    if IMSI = hostP then
    event endAuthenticator(IMSI, K_aut, K_encr);
    out(c, encrypt(secretAuthenticatorK_aut, K_aut));
    out(c, encrypt(secretAuthenticatorK_encr, K_encr));
    event end().

let keyRegistration =
    in(c, (h1: host, h2: host, k: key));
    if (h1, h2) <> (hostP, hostA) then
```

```
    insert keys(h1, h2, k).


process
         new Kpa: key;
         insert keys(hostP, hostA, Kpa);
         (
          (!procPeer(hostP, Kpa))|
          (!procAuthenticator)|
          (!keyRegistration)
         )
```

### 2.2. 攻撃者モデル

上述の記述に含まれる。

### 2.3. セキュリティ要件

上述の記述の (* queries *) に該当する。

```
(* (1) *)
query p: host, k_aut: key, k_encr: key;
      inj-event(endAuthenticator(p, k_aut, k_encr)) ==>
      inj-event(beginPeer(p, k_aut, k_encr)).
(* (2) *)
query attacker(secretPeerK_aut).
query attacker(secretPeerK_encr).
query attacker(secretAuthenticatorK_aut).
query attacker(secretAuthenticatorK_encr).
```

(1) 認証する側ロール A(authenticator)が認証される側ロール P(peer)を正しく認証すること。すなわち、認証する側ロール A(authenticator)が認証される側ロール P(peer)を相手として、鍵 k_aut 及び鍵 k_encr を用いて実行を完了するとき、ロール A を相手として鍵 k_aut 及び鍵 k_encr を用いて認証される側のロール P が存在する。

(2) 認証の際に交換した鍵 k_aut 及び鍵 k_encr の秘匿性。

## 3. ProVerif による評価結果

### 3.1. 出力

安全性のうち(1)は成り立つ (true) という結果、(2)は成り立たない (false) という結

果が出力された。

```
RESULT       inj-event(endAuthenticator(p_30,a_31,r_32,k_33))       ==>
inj-event(beginPeer(p_30,a_31,r_32,k_33)) is false.
RESULT  (even  event(endAuthenticator(p_638,a_639,r_640,k_641))  ==>
event(beginPeer(p_638,a_639,r_640,k_641)) is false.)
RESULT not attacker(secretAuthenticator[]) is false.
RESULT not attacker(secretPeer[]) is false.
```

すなわち、(2)の結果より、認証の際に交換された鍵の秘匿性が保証されないことが指摘された。これは内部で使われる関数 A3 と A8 が同じ、あるいは強い相関性がある場合には、この可能性を考慮する必要があることを意味する。一般的には、異なる関数であることが期待されるが、携帯電話会社などに実装がゆだねられており、IETF 文書では規定されていない。そのため、上記の条件下においては攻撃の可能性が存在する。

### 3.2. 攻撃の解説

受動的攻撃のため、攻撃シーケンスは正常実行の場合と同じである。

関数 A3 及び関数 A8 の実装はオペレーター(携帯電話会社等)に委ねられており、そのセキュリティ要件は明確でない。

EAP-Response/SIM/Challenge において、ロール Peer は関数 A3(Kpa, RAND1)及び関数 A3(Kpa, RAND2)を計算し、これらの値を含むメッセージの MAC を計算して AT_MAC とする。MAC 関数の出力から鍵以外の入力を知ることができると仮定すると、攻撃者は関数 A3(Kpa, RAND1)及び関数 A4(Kpa, RAND2)を知ることができる。関数 A3 及び関数 A8 実装が、関数 A3(Kpa, RAND1)及び関数 A4(Kpa, RAND2)から関数 A8(Kpa, RAND1)及び関数 A8(Kpa, RAND2)の値を計算できるようなものの場合、攻撃者は関数 A8(Kpa, RAND1)及び関数 A8(Kpa, RAND2)を知ることができる。これを用いてセッション鍵 K_aut 及び鍵 K_encr を知ることができる。このため、セッション鍵の秘匿性が成り立たない。

## 4. 形式化

### 4.1. 方針

暗号プロトコルではハッシュ関数あるいは MAC として 2 引数の関数 A3 及び関数 A8 が用いられる。これらの関数は、オペレーター(携帯電話会社)が実装することになっており、必要な安全性要件は明確には定義されていない。このため、これらの関数は理想的な一方向性関数であるとした。その一方、これらの関数は 1 回の暗号プロトコル実行では全て同

じ鍵を第一引数として用いられるため、実装方法によってはたとえば関数 A3(k, x)から関数 A8(k, x)を知ることができる可能性がある。このため、できるだけ広範な攻撃を想定するため、関数 A3 及び関数 A8 は同じ関数であるとした。

### 4.2. 妥当性

関数 A3 及び関数 A8 は鍵つきのハッシュ関数あるいは MAC であるため、理想的な一方向性関数とみなしても問題ないと考えられる。また、関数 A3 と関数 A8 を同じ関数とするのは、より安全側に倒した定式化であるため、健全な定式化になっているが、発見した攻撃が実際には不可能である可能性もある。

### 4.3. 検証ツールとの相性

プロトコル仕様、攻撃者モデル、セキュリティ要件を ProVerif で記述するにあたって、特に制限はなかった。

### 4.4. 検証ツール適用時の性能

検証時間は 0.1 秒未満であった。実行環境は以下のとおり。

- ✧ Intel Core i7 L620 2.00HGz
- ✧ Windows7 上の VirtualBox 仮想マシン上の Ubuntu Linux 12.04.1 LTS
- ✧ メモリ 512MB
- ✧ ProVerif 1.86pl3

## 5. 備考

本文書は、**総務省「暗号・認証技術等を用いた安全な通信環境推進事業に関する実証実験の請負　成果報告書**」からの引用である。