暗号プロトコル評価結果

独立行政法人　情報通信研究機構

1.**プロトコル名**：Basic Access Control (ISO/IEC 11770-2-6)

2.**関連する標準**

Machine Readable Travel Documents Technical Report

http://www.csca-si.gov.si/TR-PKI_mrtds_ICC_read-only_access_v1_1.pdf

3.**使用したツール**：Proverif

4.**評価の概要**：Proverif による評価では、攻撃の可能性は発見されなかった。

5.ProVerif による評価

5.1.シーケンス記述

```
free c: channel.


type key.
type host.
type nonce.
type tag.


const GET_CHALLENGE: tag.
const MUTUAL_AUTHENTICATE: tag.
const HDR1: tag.
const HDR2: tag.


fun encrypt(bitstring,key): bitstring.
reduc forall x: bitstring, y: key; decrypt(encrypt(x,y),y) = x.


fun mac(bitstring, key): bitstring.


fun ske_key_gen(bitstring): key.
fun mac_key_gen(bitstring): key.
```

```
not attacker(new MRZ_info).

event icc_recv_msg(channel, bitstring).
event icc_send_msg(channel, bitstring).
event ifd_recv_msg(channel, bitstring).
event ifd_send_msg(channel, bitstring).

let processIFD(ocr: channel) =
    in(ocr, MRZ_info: bitstring);
    let K_ENC = ske_key_gen(MRZ_info) in
    let K_MAC = mac_key_gen(MRZ_info) in
    (* Message1 *)
    out(c, GET_CHALLENGE);
    (* Message2 *)
    in(c, R_ICC: nonce);
    (* Message3 *)
    new R_IFD: nonce;
    new K_IFD: key;
    let m3enc = encrypt((R_IFD, R_ICC, K_IFD), K_ENC) in
    out(c, (MUTUAL_AUTHENTICATE, m3enc, mac(m3enc, K_MAC)));
    (* Message4 *)
    in(c, (m4enc:bitstring, m4mac:bitstring));
    if m4mac = mac(m4enc, K_MAC) then
    let (=R_ICC, =R_IFD, K_ICC: key) = decrypt(m4enc, K_ENC) in
    (* Events *)
    let KS_ENC = ske_key_gen((K_ICC, K_IFD)) in
    let KS_MAC = mac_key_gen((K_ICC, K_IFD)) in
    in(c, (=HDR1, menc: bitstring, mmac: bitstring));
    if mmac = mac((HDR1, menc), KS_MAC) then
    let secretICC = decrypt(menc, KS_ENC) in
    event ifd_recv_msg(ocr, secretICC);
    new secretIFD: bitstring;
```

```
    let menc' = encrypt(secretIFD, KS_ENC) in
    event ifd_send_msg(ocr, secretIFD);
    out(c, (HDR2, menc', mac((HDR2, menc'), KS_MAC))).


let processICC(ocr: channel, MRZ_info: bitstring) =
    out(ocr, MRZ_info);
    let K_ENC = ske_key_gen(MRZ_info) in
    let K_MAC = mac_key_gen(MRZ_info) in
    (* Message1 *)
    in(c, =GET_CHALLENGE);
    (* Message2 *)
    new R_ICC: nonce;
    out(c, R_ICC);
    (* Message3 *)
    in(c, (=MUTUAL_AUTHENTICATE, m3enc:bitstring, m3mac:bitstring));
    if m3mac = mac(m3enc, K_MAC) then
    let (R_IFD: nonce, =R_ICC, K_IFD: nonce) = decrypt(m3enc, K_ENC) in
    (* Message4 *)
    new K_ICC: key;
    let m4enc = encrypt((R_ICC, R_IFD, K_ICC), K_ENC) in
    out(c, (m4enc, mac(m4enc, K_MAC)));
    (* Events *)
    let KS_ENC = ske_key_gen((K_ICC, K_IFD)) in
    let KS_MAC = mac_key_gen((K_ICC, K_IFD)) in
    new secretICC: bitstring;
    let menc = encrypt(secretICC, KS_ENC) in
    event icc_send_msg(ocr, secretICC);
    out(c, (HDR1, menc, mac((HDR1, menc), KS_MAC)));
    in(c, (=HDR2, menc':bitstring, mmac':bitstring));
    if mmac' = mac((HDR2, menc'), KS_MAC) then
    let secretIFD = decrypt(menc', KS_ENC) in
```

```
    event icc_recv_msg(ocr, secretIFD).


process
         new ocr: channel; new MRZ_info: bitstring;
         (!(processICC(ocr, MRZ_info) | processIFD(ocr)))
```

セキュリティの記述のため、認証プロトコルの実行後に鍵 KS_MAC 及び KS_ENC を生成し、新たなメッセージを送信する動作を追加している。このメッセージは KS_ENC で暗号化し、暗号文から KS_MAC によって計算した MAC をつけて送信している。

5.2.攻撃者モデル

シーケンスの表記に含まれる。

5.3.セキュリティプロパティの記述

```
(* (1) *)
query attacker(new secretICC).
query attacker(new secretIFD).


(* (2) *)
query x: channel, y: bitstring;
      inj-event(icc_recv_msg(x,y)) ==> inj-event(ifd_send_msg(x,y)).
(* (3) *)
query x: channel, y: bitstring;
      inj-event(ifd_recv_msg(x,y)) ==> inj-event(icc_send_msg(x,y)).
```

(1) KS_ENC の秘匿性。より正確には、KS_ENC でメッセージを暗号化したときのメッセージの秘匿性。
(2) ICC(電子パスポート)による IFD(リーダ・デバイス)の認証。
(3) IFD による ICC の認証。

5.4.検証結果

○評価ツールの出力

```
RESULT inj-event(ifd_recv_msg(x_19,y_20)) ==>
inj-event(icc_send_msg(x_19,y_20)) is true.
RESULT inj-event(icc_recv_msg(x_1090,y_1091)) ==>
inj-event(ifd_send_msg(x_1090,y_1091)) is true.
RESULT not attacker(secretIFD_17[mmac = v_2046,menc = v_2047,m4mac =
v_2048,m4enc = v_2049,R_ICC = v_2050,MRZ_info = v_2051,!1 = v_2052]) is
true.
RESULT not attacker(secretICC[m3mac = v_2899,m3enc = v_2900,!1 =
v_2901]) is true.
```

全ての安全性が成り立つことが検証された。

〇攻撃に関する解説
攻撃は発見されなかった。

5.5.モデル化

〇モデル化プロセス
仕様書の Normative Append 5 節に即してモデル化を行った。
・パスポートに記載されている MRZ_info には十分ランダムな値が含まれると仮定した。
・シードから秘密鍵暗号及び MAC の鍵を生成するアルゴリズムは、理想的な一方向性関数として定式化した。
・メッセージのビット列としての表現を抽象化した。具体的には、パディングの省略やビット列の連結のペア関数による定式化を行った。
・秘密鍵暗号及び MAC はそれぞれ理想的な安全性をもつものとして定式化した。

5.6.モデル化の妥当性

MRZ_info のランダムネスについては、これを仮定しないと明らかに安全性が成り立たないため、これを仮定した。ただし、この仮定が成り立たない場合があることが報告されている。(http://dl.acm.org/citation.cfm?id=1784707.1784749)
その他の仮定は、形式検証を行う場合に必要な最小限の仮定であり、十分安全な暗号及び MAC を用いる場合は妥当であると考えられる。

5.7.評価ツールとの相性

〇暗号プロトコルの記述可能性

特に制限はなかった。

〇セキュリティプロパティの記述可能性

特に制限はなかった。

5.8.評価ツールの性能

評価には約 0.13 秒を要した。

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

5.9.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