Needham-Schroeder symmetric-key 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

  Needham-Schroeder symmetric-key

- ✧ 機能

  共通鍵暗号を用いた相互認証プロトコル。

- ✧ 関連する文書

  R.Needham and M.Schroeder, "Using encryption for authentication in large networks of computers," Communications of the ACM, 21(12), December 1978.

2. ProVerif の文法による記述

2.1. プロトコル仕様

```
free c: channel.


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


fun nonce_to_bitstring(nonce): bitstring [data,typeConverter].


(* Shared key encryption *)


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

```
fun Dec(bitstring): bitstring.



free hostA, hostB, hostS: host.

table ske_keys(host, host, key).
table honest(host).

(* Queries *)
free secretAKab, secretBKab: bitstring [private].

event beginA(host, host, key).
event endA(host, host, key).
event beginB(host, host, key).
event endB(host, host, key).

(* 1 *)
query x: host, y: host, k: key;
      inj-event(endA(x, y, k)) ==> inj-event(beginB(x, y, k)).

(* 2 *)
query x: host, y: host, k: key;
      inj-event(endB(x, y, k)) ==> inj-event(beginA(x, y, k)).

(* 3 *)
query attacker(secretAKab); attacker(secretBKab).



let processA =
        in(c, (A: host, B: host, S: host));
        if S = hostS then
         if A <> B then
```

```
         get honest(=A) in
        (* Message 1 *)
        new Na: nonce;
        out(c, (A, B, Na));
        (* Message 2 *)
        in(c, m2: bitstring);
         get ske_keys(=A, =S, Kas) in
        let m21 = decrypt(m2, Kas) in
        let (=Na, =B, Kab: key, m214: bitstring) = m21 in
         event beginA(A, B, Kab);
        (* Messsage 3 *)
        out(c, m214);
        (* Message 4 *)
        in(c, m4: bitstring);
        let m41 = decrypt(m4, Kab) in
        let nonce_to_bitstring(Nb) = m41 in
        (* Message 5 *)
        out(c, encrypt(Dec(nonce_to_bitstring(Nb)), Kab));
         (* security *)
        get honest(=B) in
        get honest(=S) in
         event endA(A, B, Kab);
         out(c, encrypt(secretAKab, Kab)).

let processB =
        in(c, (A: host, B: host, S: host));
        if S = hostS then
        get honest(=B) in
        (* Messsage 3 *)
        in(c, m3: bitstring);
        get ske_keys(=B, =S, Kbs) in
         let (Kab: key, =A) = decrypt(m3, Kbs) in
```

```
         event beginB(A, B, Kab);
        (* Message 4 *)
        new Nb: nonce;
        out(c, encrypt(nonce_to_bitstring(Nb), Kab));
         (* Message 5 *)
        in(c, m5: bitstring);
        let m51 = decrypt(m5, Kab) in
         if Dec(nonce_to_bitstring(Nb)) = m51 then
         (* security *)
        get honest(=A) in
        get honest(=S) in
         event endB(A, B, Kab);
         out(c, encrypt(secretBKab, Kab)).

(* Server *)

let processS =
        in(c,(A: host, B: host, S: host));
         get honest(=S) in
        (* Message1 *)
        in(c, m1: bitstring);
        let (=A, =B, Na: nonce) = m1 in
        get ske_keys(=A, =S, Kas: key) in
        get ske_keys(=B, =S, Kbs: key) in
        new Kab: key;
        out(c, encrypt((Na, B, Kab, encrypt((Kab, A), Kbs)), Kas)).

(* Key registration *)

let regist_ske_keys =
        in(c, (X: host, Y: host, Kxy: key));
        if (X, Y) <> (hostA, hostS) && (X, Y) <> (hostB, hostS) then
```

```
        if X <> Y then
         insert ske_keys(X, Y, Kxy).

(* Start process *)

process
         new Kas: key; new Kbs: key;
         insert ske_keys(hostA, hostS, Kas);
         insert ske_keys(hostA, hostS, Kbs);
        insert honest(hostA);
        insert honest(hostB);
        insert honest(hostS);
         (
          (!processA) |
          (!processB) |
          (!processS) |
           (!regist_ske_keys)
        )
```

参加者は自分自身を相手して暗号プロトコルを実行しないと仮定した。

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 x: host, y: host, k: key;
      inj-event(endA(x, y, k)) ==> inj-event(beginB(x, y, k)).
(* 2 *)
query x: host, y: host, k: key;
      inj-event(endB(x, y, k)) ==> inj-event(beginA(x, y, k)).
(* 3 *)
query attacker(secretAKab); attacker(secretBKab).
```

(1) 参加者 A による参加者 B の認証及び交換した鍵の一致。

(2) 参加者 B による参加者 A の認証及び交換した鍵の一致。

(3) 交換した鍵 Kab の秘匿性。

3. ProVerif による評価結果

3.1. 出力

いずれの安全であること（true）を確認できた。

```
RESULT not attacker(secretAKab[]) is true.
RESULT not attacker(secretBKab[]) is true.
RESULT inj-event(endB(x_5194,y_5195,k)) ==>
 inj-event(beginA(x_5194,y_5195,k)) is true.
RESULT inj-event(endA(x_10993,y_10994,k_10995)) ==>
 inj-event(beginB(x_10993,y_10994,k_10995)) is true.
```

3.2. 攻撃の解説

前述のとおり、攻撃は発見されなかった。

4. 形式化

4.1. 方針

特になし。

4.2. 妥当性

特になし。

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 0.2 秒であった。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。