

# Readme

Albanese Daniele, Marco Russodivito e Federico Zappone

2021-05-12 Wed

## Contents

<b>1</b>	<b>Esercizio 1</b>	<b>1</b>
1.1	Denning-sacco . . . . .	1
1.1.1	File Horn . . . . .	1
1.1.2	Output . . . . .	2
1.2	Denning-sacco corretto . . . . .	3
1.2.1	File Horn . . . . .	3
1.2.2	Output . . . . .	4
1.3	Conclusione . . . . .	4

## 1 Esercizio 1

### 1.1 Denning-sacco

#### 1.1.1 File Horn

```
pred c/1 elimVar,decompData.  
nounif c:x.
```

```
fun pk/1.  
fun encrypt/2.
```

```
fun sign/2.
```

```
query c:secret[].
```

```
reduc  
(* Initialization *)
```

```

c:c[];
c:pk(sA[]);
c:pk(sB[]);

(* The attacker *)

c:x & c:encrypt(m,pk(x)) -> c:m;
c:x -> c:pk(x);
c:x & c:y -> c:encrypt(x,y);
c:sign(x,y) -> c:x;
c:x & c:y -> c:sign(x,y);

(* The protocol *)
(* A *)

c:pk(x) -> c:encrypt(sign(k[pk(x)], sA[]), pk(x));

(* B *)

c:encrypt(sign(k, sA[]), pk(sB[])) -> c:encrypt(secret[], pk(k)).

```

### 1.1.2 Output

```

Initial clauses:
Clause 11: c:c[]
Clause 10: c:pk(sA[])
Clause 9: c:pk(sB[])
Clause 8: c:x & c:encrypt(m,pk(x)) -> c:m
Clause 7: c:x -> c:pk(x)
Clause 6: c:x & c:y -> c:encrypt(x,y)
Clause 5: c:sign(x,y) -> c:x
Clause 4: c:x & c:y -> c:sign(x,y)
Clause 3: c:pk(x) -> c:encrypt(sign(k[pk(x)],sA[]),pk(x))
Clause 2: c:encrypt(sign(k_1,sA[]),pk(sB[])) -> c:encrypt(secret[],pk(k_1))
Clause 1: c:new-name[!att = v]
Completing...
goal reachable: c:secret[]

Derivation:
Abbreviations:

```

```

k_1 = k[pk(x)]
clause 8 c:secret[]
  clause 5 c:k_1
    duplicate c:sign(k_1,sA[])
  clause 2 c:encrypt(secret[],pk(k_1))
    clause 6 c:encrypt(sign(k_1,sA[]),pk(sB[]))
      clause 8 c:sign(k_1,sA[])
        duplicate c:x
          clause 3 c:encrypt(sign(k_1,sA[]),pk(x))
            clause 7 c:pk(x)
              any c:x
                clause 9 c:pk(sB[])
RESULT goal reachable: c:secret[]

```

## 1.2 Denning-sacco corretto

### 1.2.1 File Horn

```

pred c/1 elimVar,decompData.
nounif c:x.

fun pk/1.
fun encrypt/2.

fun sign/2.

query c:secret[].

reduc
(* Initialization *)

c:c[];
c:pk(sA[]);
c:pk(sB[]);

(* The attacker *)

c:x & c:encrypt(m,pk(x)) -> c:m;
c:x -> c:pk(x);
c:x & c:y -> c:encrypt(x,y);

```

```

c:sign(x,y) -> c:x;
c:x & c:y -> c:sign(x,y);

(* The protocol *)
(* A *)

c:pk(x) -> c:encrypt(sign((pk(sA[]), pk(x), k[pk(x)]), sA[]), pk(x));

(* B *)

c:encrypt(sign((pk(sA[]), pk(sB[]), k), sA[]), pk(sB[])) -> c:encrypt(secret[], pk(k))

```

### 1.2.2 Output

Initial clauses:

```

Clause 15: c:(v,v_1,v_2) -> c:v_2
Clause 14: c:(v,v_1,v_2) -> c:v_1
Clause 13: c:(v,v_1,v_2) -> c:v
Clause 12: c:v & c:v_1 & c:v_2 -> c:(v,v_1,v_2)
Clause 11: c:c[]
Clause 10: c:pk(sA[])
Clause 9: c:pk(sB[])
Clause 8: c:x & c:encrypt(m,pk(x)) -> c:m
Clause 7: c:x -> c:pk(x)
Clause 6: c:x & c:y -> c:encrypt(x,y)
Clause 5: c:sign(x,y) -> c:x
Clause 4: c:x & c:y -> c:sign(x,y)
Clause 3: c:pk(x) -> c:encrypt(sign((pk(sA[]),pk(x),k[pk(x)]),sA[]),pk(x))
Clause 2: c:encrypt(sign((pk(sA[]),pk(sB[]),k_1),sA[]),pk(sB[])) -> c:encrypt(secret[])
Clause 1: c:new-name[!att = v]
Completing...
RESULT goal unreachable: c:secret[]

```

### 1.3 Conclusione

Dall'output delle due istanze si nota sin da subito che solo nel primo caso *proverif* riesce a raggiungere lo stato *secret[]* e quindi rilevare un errore. Ciò è possibile dato che nella prima istanza si assume che l'attaccante non arrivi mai a conoscenza di una chiave di sessione. Ciò rende infatti il protocollo vulnerabile al *replay attack*, in quanto, se l'attaccante entra in possesso di

una chiave di sessione potrà cifrare un nuovo messaggio che sarà accettato dalla vittima. Nella seconda istanza ciò non accade dato che si aggiunge come ulteriore forma di sicurezza un *nonce* modellato da  $\mathbf{pk}(sA[])$ ,  $\mathbf{pk}(x)$  nella riga:

```
c:pk(x) -> c:encrypt(sign((pk(sA[]), pk(x), k[pk(x)]), sA[]), pk(x));
```