

TAMARIN

Adrian Manea

510, SLA

Analizăm formal protocoalele de securitate, pe baza specificării:

Analizăm formal protocoalele de securitate, pe baza specificării:

- acțiunilor protocolului;

Analizăm formal protocoalele de securitate, pe baza specificării:

- acțiunilor protocolului;
- adversarului (tip Dolev–Yao);

Analizăm formal protocoalele de securitate, pe baza specificării:

- acțiunilor protocolului;
- adversarului (tip Dolev–Yao);
- proprietăților dorite (leme).

Declararea funcțiilor definite sau importate:

```
builtins: hashing, asymmetric-encryption  
functions: mac/2, g/0, shk/0 [private]
```

```
rule Register_pk: [ Fr(~ltk) ] --> [ ~Ltk($A, ~ltk), ~Pk($A, pk(~ltk)) ]
```

Sintaxă (stări inițiale)

Declararea funcțiilor definite sau importate:

```
builtins: hashing, asymmetric-encryption  
functions: mac/2, g/0, shk/0 [private]
```

Regula de înregistrare a cheii publice:

```
rule Register_pk: [ Fr(~ltk) ] --> [ ~Ltk($A, ~ltk), ~Pk($A, pk(~ltk)) ]
```

Sintaxă (stări inițiale)

Declararea funcțiilor definite sau importate:

```
builtins: hashing, asymmetric-encryption  
functions: mac/2, g/0, shk/0 [private]
```

Regula de înregistrare a cheii publice:

```
rule Register_pk: [ Fr(~ltk) ] --> [ ~Ltk($A, ~ltk), ~Pk($A, pk(~ltk)) ]
```

Tipuri (sorturi):

Sintaxă (stări inițiale)

Declararea funcțiilor definite sau importate:

```
builtins: hashing, asymmetric-encryption  
functions: mac/2, g/0, shk/0 [private]
```

Regula de înregistrare a cheii publice:

```
rule Register_pk: [ Fr(~ltk) ] --> [ ~Ltk($A, ~ltk), ~Pk($A, pk(~ltk)) ]
```

Tipuri (sorturi):

- $\sim x = \text{variabilă nouă} = x : \text{fresh};$

Sintaxă (stări inițiale)

Declararea funcțiilor definite sau importate:

```
builtins: hashing, asymmetric-encryption  
functions: mac/2, g/0, shk/0 [private]
```

Regula de înregistrare a cheii publice:

```
rule Register_pk: [ Fr(~ltk) ] --> [ ~Ltk($A, ~ltk), ~Pk($A, pk(~ltk)) ]
```

Tipuri (sorturi):

- $\sim x$ = variabilă nouă = $x : \text{fresh}$;
- $\$x$ = variabilă publică = $x : \text{pub}$;

Sintaxă (stări inițiale)

Declararea funcțiilor definite sau importate:

```
builtins: hashing, asymmetric-encryption  
functions: mac/2, g/0, shk/0 [private]
```

Regula de înregistrare a cheii publice:

```
rule Register_pk: [ Fr(~ltk) ] --> [ ~Ltk($A, ~ltk), ~Pk($A, pk(~ltk)) ]
```

Tipuri (sorturi):

- $\sim x$ = variabilă nouă = $x : \text{fresh}$;
- $\$x$ = variabilă publică = $x : \text{pub}$;
- $\#i$ = variabilă temporală = $i : \text{temporal}$;

Sintaxă (stări inițiale)

Declararea funcțiilor definite sau importate:

```
builtins: hashing, asymmetric-encryption  
functions: mac/2, g/0, shk/0 [private]
```

Regula de înregistrare a cheii publice:

```
rule Register_pk: [ Fr(~ltk) ] --> [ ~Ltk($A, ~ltk), ~Pk($A, pk(~ltk)) ]
```

Tipuri (sorturi):

- $\sim x$ = variabilă nouă = $x : \text{fresh}$;
- $\$x$ = variabilă publică = $x : \text{pub}$;
- $\#i$ = variabilă temporală = $i : \text{temporal}$;
- $m = \text{mesaj} = m : \text{msg}$.

Propagarea cheii în rețea:

```
rule Get_pk: [ !Pk(A, pubkey) ] --> [ Out(pubkey) ]  
/* !Pk = acțiune persistentă, se poate repeta oricând e nevoie */
```

```
rule Reveal_ltk: [ !Ltk(A, ltk) ] --[ LtkReveal(A) ]-> [ Out(ltk) ]
```

Propagarea cheii în rețea:

```
rule Get_pk: [ !Pk(A, pubkey) ] --> [ Out(pubkey) ]  
/* !Pk = acțiune persistentă, se poate repeta oricând e nevoie */
```

Compromiterea cheii (dezvăluirea de către adversar) este reprezentată ca un fapt în acțiune = urmă \neq stare:

```
rule Reveal_ltk: [ !Ltk(A, ltk) ] --[ LtkReveal(A) ]-> [ Out(ltk) ]
```

Sintaxă (funcționare)

Creează fir pentru client, alege server non-determinist:

```
rule Client_1: [ Fr(~k), ~Pk($S, pkS) ] --> [ Client_1($S, ~k), Out(aenc(~k, pkS))
```

```
rule Client_2: [ Client_1(S, k), In(h(k)) ] --[ SessKeyC(S,k) ]-> []
```

```
rule Serv_1: [ ~Ltk($S, ~ltkS), In(request) ]  
  --[ AnswerRequest($S, adec(request, ~ltkS)) ]->  
  [ Out(h(adec(request, ~ltkS))) ]
```

Sintaxă (funcționare)

Creează fir pentru client, alege server non-determinist:

```
rule Client_1: [ Fr(~k), ~Pk($S, pkS) ] --> [ Client_1($S, ~k), Out(aenc(~k, pkS)) ]
```

Folosind cheia, se menționează pe urmă că s-a stabilit o legătură cu serverul S, pe baza cheii k:

```
rule Client_2: [ Client_1(S, k), In(h(k)) ] --[ SessKeyC(S,k) ]-> []
```

```
rule Serv_1: [ ~Ltk($S, ~ltkS), In(request) ]  
  --[ AnswerRequest($S, adec(request, ~ltkS)) ]->  
  [ Out(h(adec(request, ~ltkS))) ]
```


Sintaxă (funcționare)

Creează fir pentru client, alege server non-determinist:

```
rule Client_1: [ Fr(~k), ~Pk($S, pkS) ] --> [ Client_1($S, ~k), Out(aenc(~k, pkS)) ]
```

Folosind cheia, se menționează pe urmă că s-a stabilit o legătură cu serverul S, pe baza cheii k:

```
rule Client_2: [ Client_1(S, k), In(h(k)) ] --[ SessKeyC(S,k) ]-> []
```

Răspunsul serverului (menționat tot pe urmă):

```
rule Serv_1: [ ~Ltk($S, ~ltkS), In(request) ]  
  --[ AnswerRequest($S, adec(request, ~ltkS)) ]->  
  [ Out(h(adec(request, ~ltkS))) ]
```

lemma Client_session_key_secrecy:

" not(/* nu este cazul ca */

Ex S k #i #j.

/* clientul a stabilit o cheie de sesiune k cu serverul S */

SessKeyC(S, k) @ #i

/* iar adversarul să cunoască cheie */

& K(k) @ #j

/* fără a fi aplicat o dezvăluire pe S */

& not(Ex #r. LtkReveal(S) @ r)

)

"

Proprietăți (leme)

lemma Client_auth:

```
" /* Pentru toate cheile de sesiune 'k'
    stabilite de clienți cu serverul 'S' */
  ( All S k #i. SessKeyC(S, k) @ #i
    ==>
    /* există un server care a răspuns la cerere */
    ( (Ex #a. AnswerRequest(S, k) @ a)
    /* sau adversarul a aflat cheia înainte de a fi stabilită */
      | (Ex #r. LtkReveal(S) @ r & r < i)
    )
  )
"
```

/* Execuție nevidă */

lemma Client_session_key_honest_setup:

```
exists-trace
  " Ex S k #i.
    SessKeyC(S, k) @ #i
    & not(Ex #r. LtkReveal(S) @ r)
  "
```

Proprietate suplimentară: autorizare injectivă

lemma Client_auth_injective:

```
" /* Pentru toate cheile de sesiune 'k' stabilite cu serverul 'S' */
  ( All S k #i. SessKeyC(S, k) @ #i
    ==>
      /* există un server care a răspuns */
      ( (Ex #a. AnswerRequest(S, k) @ a
        /* și niciun alt client n-a avut exact aceeași cerere */
        & (All #j. SessKeyC(S, k) @ #j ==> #i = #j)
        ) /* sau adversarul a aflat cheia înainte de a fi stabilită */
        | (Ex #r. LtkReveal(S) @ r & r < i)
      )
    )
"
```

Bibliografie



Basin, D. A., Cremers, C., Dreier, J., and Sasse, R. (2017).
Symbolically analyzing security protocols using tamarin.
SIGLOG News, 4(4):19–30.



Cremers, C. and Mauw, S. (2012).
Operational Semantics and Verification of Security Protocols.
Springer.



Tamarin, T. (2019).
Tamarin-prover manual.
tamarin-prover.github.io.