Lightning talk:

Key exchange protocols over voice channels and verification using Tamarin Prover

Presented by: **Piotr Krasnowski**

PhD supervised by: Prof. Bruno Martin, Dr. Jerome Lebrun and Arnaud Graube

DGA supervision: Thierry Plesse and Sylvain Le Bihan

18 June 2019

Project Outline

Figure: $CBOX^{TM}$ by BlackBoxSecu.

Characteristics:

- end-to-end voice encryption
- audio-to-audio processing
- real-time operation

Key technologies:

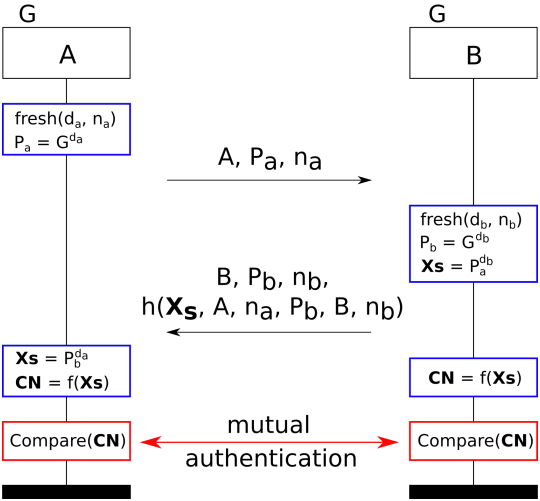
- data over voice channel
- enciphered speech over voice channel
- cryptographic key management

Data over voice channels

- low bitrate ($\sim$ 1.5 kbps)
- channel errors (BER $\sim$ 10 %)
- strong fading (message suppression)

How to exchange session keys?

Example of the protocol



G

A

fresh(d_a, n_a)
$P_a = G^{d_a}$

G

B

A, P_a, n_a

fresh(d_b, n_b)
$P_b = G^{d_b}$
$\mathbf{Xs} = P_a^{d_b}$

$B, P_b, n_b,$
$h(\mathbf{X_s}, A, n_a, P_b, B, n_b)$

$\mathbf{Xs} = P_b^{d_a}$
$\mathbf{CN} = f(\mathbf{Xs})$

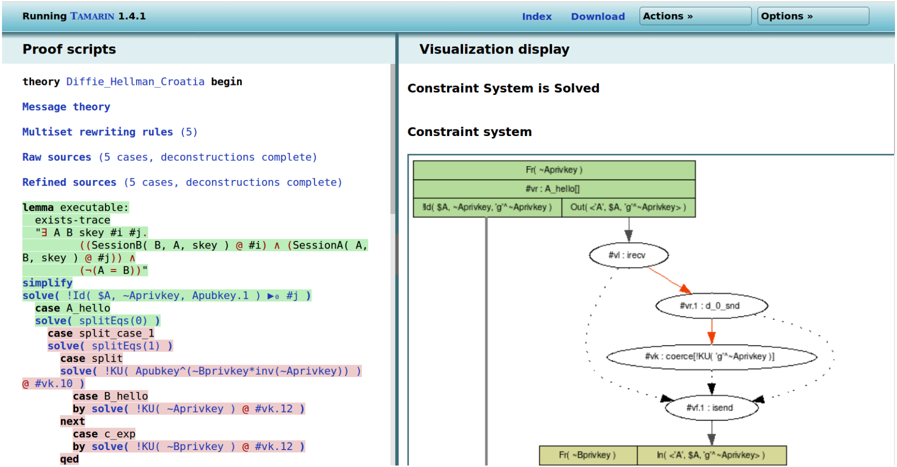
$\mathbf{CN} = f(\mathbf{Xs})$

Compare(**CN**)

mutual
authentication

Compare(**CN**)

Tamarin Prover



Running TAMARIN 1.4.1

Proof scripts

```
theory Diffie_Hellman_Croatia begin

Message theory

Multiset rewriting rules (5)

Raw sources (5 cases, deconstructions complete)

Refined sources (5 cases, deconstructions complete)

lemma executable:
  exists-trace
  "∃ A B skey #i #j.
         ((SessionB( B, A, skey ) @ #i) ∧ (SessionA( A,
B, skey ) @ #j)) ∧
         (¬(A = B))"
simplify
solve( !Id( $A, ~Aprivkey, Apubkey.1 ) ▶₀ #j )
  case A_hello
  solve( splitEqs(0) )
    case split_case_1
    solve( splitEqs(1) )
      case split
      solve( !KU( Apubkey^(~Bprivkey*inv(~Aprivkey)) )
@ #vk.10 )
        case B_hello
        by solve( !KU( ~Aprivkey ) @ #vk.12 )
      next
        case c_exp
        by solve( !KU( ~Bprivkey ) @ #vk.12 )
      qed
```

Visualization display

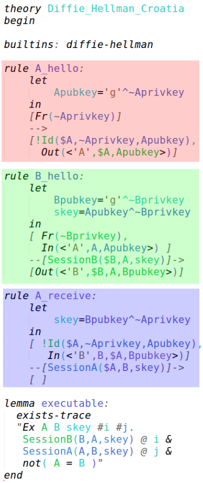
Constraint System is Solved

Constraint system

Fr(~Aprivkey)
#vr : A_hello[]
!Id($A, ~Aprivkey, 'g'^~Aprivkey)
Out(<'A', $A, 'g'^~Aprivkey>)
#vl : irecv
#vr.1 : d_0_snd
#vk : coerce[!KU('g'^~Aprivkey)]
#vf.1 : isend
Fr(~Bprivkey)
In(<'A', $A, 'g'^~Aprivkey>)

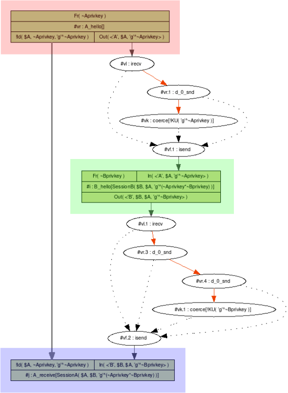
Tamarin Prover

Tamarin code:

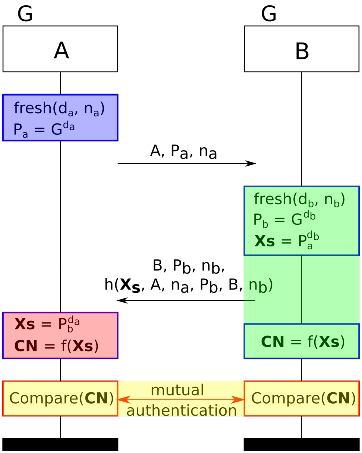


```
theory Diffie_Hellman_Croatia
begin

builtins: diffie-hellman

rule A_hello:
    let
        Apubkey='g'^~Aprivkey
    in
    [Fr(~Aprivkey)]
    -->
    [!Id($A,~Aprivkey,Apubkey),
      Out(<'A',$A,Apubkey>)]

rule B_hello:
    let
        Bpubkey='g'^~Bprivkey
        skey=Apubkey^~Bprivkey
    in
    [ Fr(~Bprivkey),
      In(<'A',A,Apubkey>) ]
    --[SessionB($B,A,skey)]->
    [Out(<'B',$B,A,Bpubkey>)]

rule A_receive:
    let
        skey=Bpubkey^~Aprivkey
    in
    [ !Id($A,~Aprivkey,Apubkey),
       In(<'B',B,$A,Bpubkey>)]
    --[SessionA($A,B,skey)]->
    [ ]

lemma executable:
  exists-trace
  "Ex A B skey #i #j.
   SessionB(B,A,skey) @ i &
   SessionA(A,B,skey) @ j &
   not( A = B )"
end
```

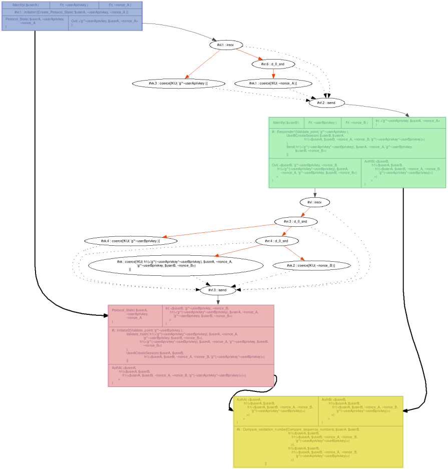
Protocol diagram:


Fr(~Aprivkey)
#vr : A_hello[]
!Id($A, ~Aprivkey, 'g'^~Aprivkey)
Out(<'A', $A, 'g'^~Aprivkey>)
#vl : irecv
#vr.1 : d_0_snd
#vk : coerce[!KU('g'^~Aprivkey)]
#vf.1 : isend
Fr(~Bprivkey)
In(<'A', $A, 'g'^~Aprivkey>)
#i : B_hello[SessionB($B, $A, 'g'^(~Aprivkey*~Bprivkey))]
Out(<'B', $B, $A, 'g'^~Bprivkey>)
#vl.1 : irecv
#vr.3 : d_0_snd
#vr.4 : d_0_snd
#vk.1 : coerce[!KU('g'^~Bprivkey)]
#vf.2 : isend
!Id($A, ~Aprivkey, 'g'^~Aprivkey)
In(<'B', $B, $A, 'g'^~Bprivkey>)
#j : A_receive[SessionA($A, $B, 'g'^(~Aprivkey*~Bprivkey))]

Protocol model in Tamarin



G
A
G
B
fresh(d_a, n_a)
$P_a = G^{d_a}$
A, P_a, n_a
fresh(d_b, n_b)
$P_b = G^{d_b}$
$\mathbf{Xs} = P_a^{d_b}$
B, P_b, n_b,
h($\mathbf{Xs}$, A, n_a, P_b, B, n_b)
$\mathbf{Xs} = P_b^{d_a}$
$\mathbf{CN} = f(\mathbf{Xs})$
$\mathbf{CN} = f(\mathbf{Xs})$
Compare(**CN**)
mutual
authentication
Compare(**CN**)



!Identity($userA)
Fr(~userAprivkey)
Fr(~nonce_A)
#vr.1 : Initiator1[Create_Protocol_State($userA, ~userAprivkey, ~nonce_A)]
Protocol_State($userA, ~userAprivkey, ~nonce_A)
Out(<'g'^~userAprivkey, $userA, ~nonce_A>)
#vl.1 : irecv
#vr.6 : d_0_snd
#vk.3 : coerce[!KU('g'^~userAprivkey)]
#vk.1 : coerce[!KU(~nonce_A)]
#vf.2 : isend
!Identity($userB)
Fr(~userBprivkey)
Fr(~nonce_B)
In(<'g'^~userAprivkey, $userA, ~nonce_A>)
#i : Responder1[Validate_point('g'^~userAprivkey),
UserBCreateSession($userB, $userA,
h1(<$userA, $userB, ~nonce_A, ~nonce_B, 'g'^(~userAprivkey*~userBprivkey)>)
),
Send(h1(<'g'^(~userAprivkey*~userBprivkey), $userA, ~nonce_A, 'g'^~userBprivkey,
$userB, ~nonce_B>)
)]
Out(<$userB, 'g'^~userBprivkey, ~nonce_B,
h1(<'g'^(~userAprivkey*~userBprivkey), $userA,
~nonce_A, 'g'^~userBprivkey, $userB, ~nonce_B>)
>
)
AuthB(<$userB,
h1(<$userA, $userB,
h1(<$userA, $userB, ~nonce_A, ~nonce_B,
'g'^(~userAprivkey*~userBprivkey)>)
>)
>
)
#vl : irecv
#vr.3 : d_0_snd
#vk.4 : coerce[!KU('g'^~userBprivkey)]
#vr.4 : d_0_snd
#vk : coerce[!KU(h1(<'g'^(~userAprivkey*~userBprivkey), $userA, ~nonce_A,
'g'^~userBprivkey, $userB, ~nonce_B>)
)]
#vk.2 : coerce[!KU(~nonce_B)]
#vf.3 : isend
Protocol_State($userA, ~userAprivkey, ~nonce_A)
In(<$userB, 'g'^~userBprivkey, ~nonce_B,
h1(<'g'^(~userAprivkey*~userBprivkey), $userA, ~nonce_A,
'g'^~userBprivkey, $userB, ~nonce_B>)
>
)
#j : Initiator2[Validate_point('g'^~userBprivkey),
Validate_hash(h1(<'g'^(~userAprivkey*~userBprivkey), $userA, ~nonce_A,
'g'^~userBprivkey, $userB, ~nonce_B>),
h1(<'g'^(~userAprivkey*~userBprivkey), $userA, ~nonce_A, 'g'^~userBprivkey, $userB,
~nonce_B>)
),
UserACreateSession($userA, $userB,
h1(<$userA, $userB, ~nonce_A, ~nonce_B, 'g'^(~userAprivkey*~userBprivkey)>)
)]
AuthA(<$userA,
h1(<$userA, $userB,
h1(<$userA, $userB, ~nonce_A, ~nonce_B, 'g'^(~userAprivkey*~userBprivkey)>)>)
>
)
AuthA(<$userA,
h1(<$userA, $userB,
h1(<$userA, $userB, ~nonce_A, ~nonce_B,
'g'^(~userAprivkey*~userBprivkey)>)
>)
>
)
AuthB(<$userB,
h1(<$userA, $userB,
h1(<$userA, $userB, ~nonce_A, ~nonce_B,
'g'^(~userAprivkey*~userBprivkey)>)
>)
>
)
#k : Compare_validation_number[Compare_sequence_numbers($userA, $userB,
h1(<$userA, $userB,
h1(<$userA, $userB, ~nonce_A, ~nonce_B,
'g'^(~userAprivkey*~userBprivkey)>)
>),
h1(<$userA, $userB,
h1(<$userA, $userB, ~nonce_A, ~nonce_B,
'g'^(~userAprivkey*~userBprivkey)>)
>)
)]

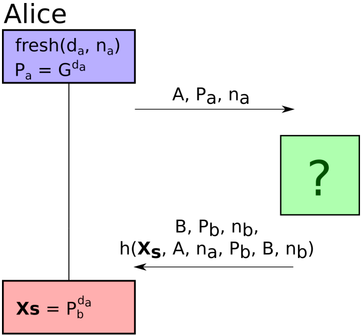
Lemma - example

lemma **secrecy:**

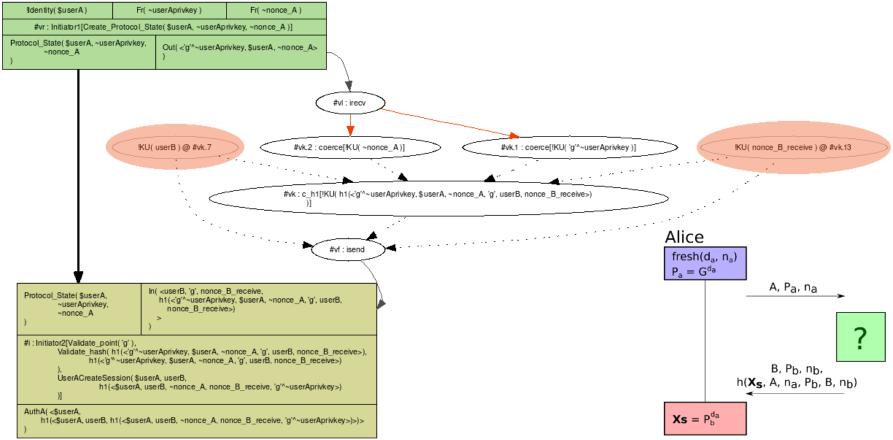
"**All** Alice Bob secret #i .
AliceSession(Alice, Bob, secret) @ i

==> /* implies */

Ex #j. BobSession(Bob, Alice, secret) @ j
& /* and */
/* adversary never knows the secret */
not(Ex #k. KU(secret) @ k)"



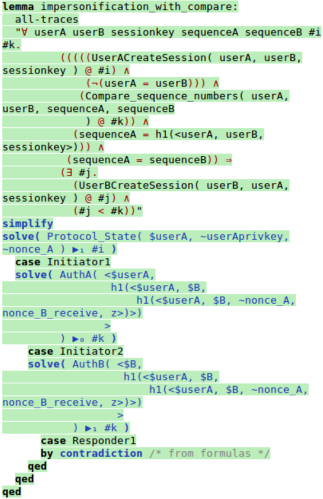
No authentication - Tamarin's attack


!Identity($userA)
Fr(~userAprivkey)
Fr(~nonce_A)
#vr : Initiator1[Create_Protocol_State($userA, ~userAprivkey, ~nonce_A)]
Protocol_State($userA, ~userAprivkey, ~nonce_A)
Out(<'g'^~userAprivkey, $userA, ~nonce_A>)
#vl : irecv
!KU(userB) @ #vk.7
#vk.2 : coerce[!KU(~nonce_A)]
#vk.1 : coerce[!KU('g'^~userAprivkey)]
!KU(nonce_B_receive) @ #vk.13
#vk : c_h1[!KU(h1(<'g'^~userAprivkey, $userA, ~nonce_A, 'g', userB, nonce_B_receive>))]
#vf : isend
Protocol_State($userA, ~userAprivkey, ~nonce_A)
In(<userB, 'g', nonce_B_receive, h1(<'g'^~userAprivkey, $userA, ~nonce_A, 'g', userB, nonce_B_receive>) >)
#i : Initiator2[Validate_point('g'), Validate_hash(h1(<'g'^~userAprivkey, $userA, ~nonce_A, 'g', userB, nonce_B_receive>), h1(<'g'^~userAprivkey, $userA, ~nonce_A, 'g', userB, nonce_B_receive>)), UserACreateSession($userA, userB, h1(<$userA, userB, ~nonce_A, nonce_B_receive, 'g'^~userAprivkey>))]
AuthA(<$userA, h1(<$userA, userB, h1(<$userA, userB, ~nonce_A, nonce_B_receive, 'g'^~userAprivkey>)>)>)
Alice
fresh(d_a, n_a)
$P_a = G^{d_a}$
A, P_a, n_a
?
B, P_b, n_b,
h($\mathbf{X_s}$, A, n_a, P_b, B, n_b)
$\mathbf{Xs} = P_b^{d_a}$

Importance of authentication stage

Verified properties:

- secure shared secret
- resistance against replay attack
- injective agreement (if users authenticated each other)



```
lemma impersonification_with_compare:
  all-traces
  "∀ userA userB sessionkey sequenceA sequenceB #i
#k.
         (((((UserACreateSession( userA, userB,
sessionkey ) @ #i) ∧
             (¬(userA = userB))) ∧
            (Compare_sequence_numbers( userA,
userB, sequenceA, sequenceB
             ) @ #k)) ∧
           (sequenceA = h1(<userA, userB,
sessionkey>))) ∧
          (sequenceA = sequenceB)) ⇒
         (∃ #j.
           (UserBCreateSession( userB, userA,
sessionkey ) @ #j) ∧
           (#j < #k))"
simplify
solve( Protocol_State( $userA, ~userAprivkey,
~nonce_A ) ▶₁ #i )
  case Initiator1
  solve( AuthA( <$userA,
                 h1(<$userA, $B,
                     h1(<$userA, $B, ~nonce_A,
nonce_B_receive, z>)>)
                >
         ) ▶₀ #k )
    case Initiator2
    solve( AuthB( <$B,
                   h1(<$userA, $B,
                       h1(<$userA, $B, ~nonce_A,
nonce_B_receive, z>)>)
                  >
           ) ▶₁ #k )
      case Responder1
      by contradiction /* from formulas */
    qed
  qed
qed
```

Conclusions and related work

Conclusions:

- Tamarin is an extremely useful verification tool
- many examples available online
- requires a careful design - a mistake can be costly!

Related work:

- session key exchange based on long-term key
- challenging for unreliable channels!

Thank you for your attention!