The AVISPA Library

David von Oheimb

Siemens AG, Munich

UNIGE INRIA-Lorraine ETHZ Siemens AG

Automated Validation of Internet Security Protocols and Applications

Shared cost RTD (FET open) project IST-2001-39252

Context

Design and standardisation of Internet protocols

- Standardisation committees: IETF, W3C, 3GPP, OMA, IEEE
- IETF activity in currently 7 different areas (106 groups)
- Protocols ranging over 5 IP layers
- 20+ different security goals
- Design is costly, time-consuming and error-prone
- Support by formal techniques and tools needed

Overview

The AVISPA Library is a large collection of Internet protocols, specified together with their properties in HLPSL.

It is used to...

- guide, tune and assess the development of the AVISPA tool on a large collection of practically relevant, industrial protocols.
- migrate this technology to developers and standardisation bodies, by providing
  - feedback on current protocol developments
  - examples for specifying and analysing new designs

Protocols and Problems

Deliverable 6.1: *List of Selected Problems*

79 protocols from 33 groups, constituting 384 security problems

Deliverable 6.2: *Specification of the Problems*
in the high-level specification language

describing the AVISPA Library,
a 400+ pages LaTeX (and HTML) document,
and it's growing...

List of Protocols

AAAMobileIP	EAP_AKA	SRP	ASW
CTP-non_predictive-fix	EAP_Archie	RADIUS-RFC2865	FairZG
SIP	EAP_IKEv2	8021x_Radius	SET-purchase
H.530	EAP_SIM	HIP	SET-purchase-HPG
H.530-fix	EAP_TLS	PBK	UMTS_AKA
QoS-NSLP	EAP_TTLS_CHAP	PBK-fix	ISO1
Geopriv	PEAP	PBK-fix-weak-auth	ISO2
Geopriv-two_pseudonyms	S/KEY	Kerberos-basic	ISO3
Geopriv-pervasive	EKE	Kerberos-Ticket-Cache	ISO4
SIMPLE	EKE2	Kerberos-Cross-Realm	2pRSA
LIPKEY-SPKM-known-initiator	SPEKE	Kerberos-Forwardable	LPD-MSR
LIPKEY-SPKM-unknown-initiator	IKEv2-DS	Kerberos-PKINIT	LPD-IMSR
CHAPv2	IKEv2-DSx	Kerberos-preauth	NSPK
APOP	IKEv2-MAC	TESLA	NSPK-fix
CRAM-MD5	IKEv2-MACx	SSH-transport	NSPK-KS
DHCP-delayed-auth	IKEv2-CHILD	TSP	NSPK-KS-fix
TSIG	IKEv2-EAP-Archie	TLS	NSPK-xor

Areas covered (1)

The AVISPA Library largely covers the IETF range of protocols and related security properties.

- Infrastructure (DHCP, DNS, TSP)
- Network Access (PANA)
- Mobility (Mobile IP, UMTS-AKA, seamoby)
- IPv6 (RADIUS, HIP, PBK)
- VoIP, messaging, presence (SIP, H530, IMPP)

Areas covered (2)

- Internet Security (Kerberos, IKE, EKE, TLS, EAP, OTP, ssh, ...)
- Privacy (Geopriv)
- QoS (NSIS)
- Broadcast/Multicast Authentication (TESLA)
- E-Commerce (ASW, FairZG, SET)
- others (ISO-PK, 2pRSA, LPD, ...)

Goals covered

- Authentication (unicast + multicast)
  - Entity authentication (G1)
  - Message origin and integrity (G2/G5)
  - Replay protection (G3)
- Key agreement (reduced to authentication)
  - Key authentication (G7)
  - Key confirmation (G8)
  - Fresh key derivation (G10)
- Confidentiality (G12)

Goals approximated

For these, additional (meta-level) argumentation is required.

- Identity protection ("Anonymity")
  - against eavesdroppers (G13)
  - against peers (G14)
- Non-repudiation
  - Proof of Origin (G18)
  - Proof of Delivery (G19)
- Fair exchange ($\rightsquigarrow$ FairZG)

Views on Security Problems

Example: Needham-Schroeder Public-Key Protocol (NSPK)

Problem Classification: G1 (Entity Auth.), G3 (Replay Prot.), G12 (Secrecy)

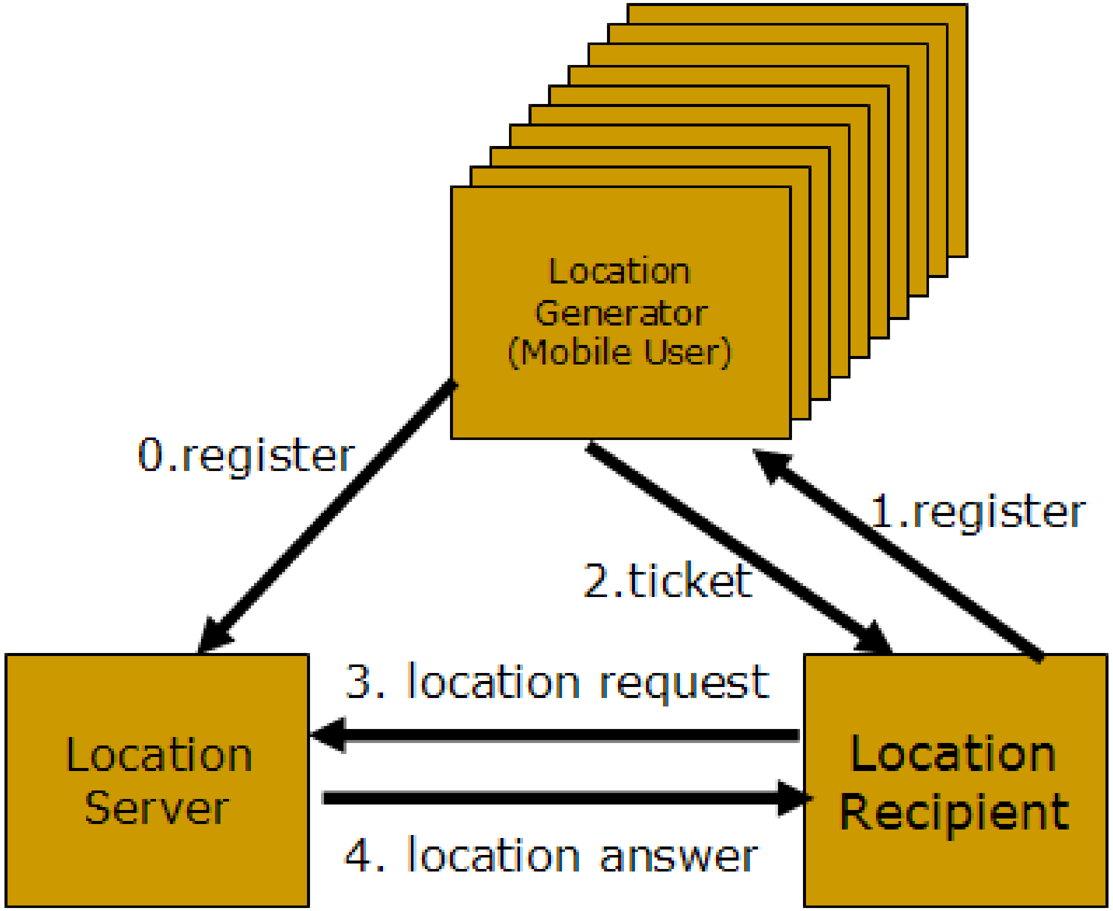
Problems Considered: 4

- secrecy of `na, nb`
- strong authentication on `alice_bob_nb`
- strong authentication on `bob_alice_na`

Designer's view: for a given protocol, there are different goal *types*, and in D6.1 each of them counts *once*, e.g. $|\{G1, G3, G12\}| = 3$

Formal analyst's view: several secrecy and authentication checks (where a single check may cover several goals). Each check counts, e.g. $|\{\text{secrecy}_1, \text{secrecy}_2, \text{authentication}_1, \text{authentication}_2\}| = 4$

Example Specification: Geopriv



Conclusion

The AVISPA Library has provided...

- invaluable feedback on the HLPSL and AVISPA tools developed
- evidence that the AVISPA Tool...
  - can effectively deal with major security problems
  - can be used with just little effort and training
- good examples how to make protocol design efficient and secure

Outlook

The AVISPA Library is the
best publicly available library of security protocols

It will be used as the reference benchmark suite
for automatic security protocol analysers
for several years to come.

Geopriv

Variant with pseudonym for Location Recipient only

Definition Reference

`http://www.faqs.org/rfcs/rfc3693.html` [CMM+04]

Protocol Purpose

Obtain geographical location information restricted by a privacy policy.
Using a pseudonym, the location recipient is anonymous to the location server.

Model Authors

Lan Liu for Siemens CT IC 3, January 2005

Alice&Bob style

```
MU : Mobile User (= Target) (subsumes the Rule Maker)
LR : Location Recipient
LS : Location Server (subsumes the Location Sighter)
1. LR ------ LR.N_LR.{LR}_K_MU_LR -> MU
2. LR <- {N_LR.Psi.K_Psi}_K_MU_LR -- MU
3.                                   MU -- {MU.Psi.K_Psi DT}_K_MU_LS -> LS
% some time later, LR requests the location of MU:
4. LR ----------------- {LS.MU.Psi.K_Psi.K1}_Pk_LS -------------------> LS
5. LR <------------------------- {DT(Loc)}_K1 ------------------------- LS
DT ("data type") describes the accuracy of the location information.
It is a function projecting/filtering Loc to the accuracy allowed by the MU.
```

Model Limitations

For simplicity we model the Location Sighter as part of the Location Server,
which is fine here because the Location Server is allowed to know the identity of the Target.

Problems Considered: 8

- secrecy of filtered_loc, psi, k_psi, k1
- strong authentication on lr_ls_filtered_loc
- strong authentication on lr_mu_n_lr
- weak authentication on ls_mu_psi
- weak authentication on mu_lr_lr

Problem Classification: G1, G2, G3, G12, G20

Attacks Found: None

Further Notes

- The name of LR in the initial contact is modelled as in the clear and encrypted. The encrypted form of the LR information is used by T to authenticate the LR. In reality the initial contact can be part of another protocol, protected via PKI, or unprotected.
- An LR can get a certain `{Psi,K_Psi}` pair from the MU. `K_Psi` is the key related to the pseudonym `Psi` of a LR. `Psi` and `K_Psi` are used for authorisation to get location information from the LS. Although `K_Psi` is the password for `Psi` of `LR`, it could be omitted here because the secrecy of `Psi` suffices.
- `K1` is a temporary key of LR, generated by LR for encryption of the location information sent by LS.
- LS cannot authenticate LR because he knows only the pseudonym of LR, since an important objective of this protocol is the anonymity of LR to LS.
- The secrecy fact for `filtered_loc` is given in the role of the LS (where the secret actually is produced). To make this possible, the LS has LR as its parameter, but only for technical reasons to state the goal. LS does not make use of this “knowledge”, as it should know only LS’s pseudonym.
- In the last step, LS does not know to whom to answer. In reality, an IP address is used, but here, one may regard it is a broadcast.

HLPSL Specification

```
role locationRecipient(
        MU, LR, LS         : agent,
        K_MU_LR            : symmetric_key,
        Pk_LS              : public_key,
        Snd, Rcv           : channel(dy)) played_by LR def=

local
        State              : nat,
        N_LR, Psi          : text,
        K_Psi              : symmetric_key,
                           % password for pseudonym Psi of a certain LR,
                           % generated by MU and stored by LS
        K1                 : public_key, % could also be: symmetric_key
        Filtered_Loc       : message

init State := 0
```

```
transition

        0. State   = 0 /\ Rcv(start)
       =|> State':= 2 /\ N_LR' := new()
                      /\ Snd(LR.N_LR'.{LR}_K_MU_LR)
                      /\ witness(LR, MU, mu_lr_lr, LR)

        2. State   = 2 /\ Rcv({N_LR.Psi'.K_Psi'}_K_MU_LR)
       =|> State':= 4 /\ K1' := new()
                      /\ secret(K1', k1, {LR, LS})
                      /\ Snd({LS.MU.Psi'.K_Psi'.K1'}_Pk_LS)

        4. State   = 4 /\ Rcv({Filtered_Loc'}_K1)
       =|> State':= 6 /\ request(LR, LS, lr_ls_filtered_loc, Filtered_Loc')
                      /\ request(LR, MU, lr_mu_n_lr, N_LR)

end role
```

```
role mobileUser(
        MU, LR, LS : agent,
        K_MU_LR    : symmetric_key,
        K_MU_LS    : symmetric_key,
        Snd_LR, Snd_LS,
        Rcv        : channel(dy)) played_by MU def=

local
        State    : nat,
        N_LR     : text,
        Psi      : text,
        K_Psi    : symmetric_key,
        DT       : function

const  psi, k_psi : protocol_id

init State := 1
```

```
transition

        1. State   = 1 /\ Rcv(LR.N_LR'. {LR}_K_MU_LR)
       =|> State':= 3 /\   Psi' := new()
                      /\ K_Psi' := new()
                      /\ secret(  Psi,   psi, {MU, LR, LS})
                      /\ secret(K_Psi,k_psi, {MU, LR, LS})
                      /\ Snd_LR({N_LR'.Psi'.K_Psi'}_K_MU_LR)
                      /\ witness(MU, LR, lr_mu_n_lr, N_LR')
                      /\ wrequest(MU, LR, mu_lr_lr, LR)
                      /\ DT' := new()  % chooses some accuracy
                      /\ Snd_LS({MU. Psi'. K_Psi'. DT'}_K_MU_LS)
                      /\ witness(MU, LS, ls_mu_psi, Psi')
end role
```

```
role locationServer(
        MU, LR,  % but LS does not use identity of LR, which addresses G14
        LS       : agent,
        Psi_Table: (agent.text.symmetric_key.function) set,
        Pk_LS    : public_key,
        K_MU_LS  : symmetric_key,
        Snd, Rcv : channel(dy)) played_by LS def=

local   State             : nat,
        K1                : public_key,
        Na                : text,
        K_Psi             : symmetric_key,
        Psi               : text,
        DT                : function,
        Loc               : text

init    State := 7
```

```
transition

       7. State  = 7  /\ Rcv({MU. Psi'. K_Psi'. DT'}_K_MU_LS)
                               % actually, LS should learn MU here
       =|>State':= 9  /\ Psi_Table':= cons(MU.Psi'.K_Psi'. DT', Psi_Table)
                      /\ wrequest(LS, MU, ls_mu_psi, Psi')
                                      % need MU here for technical reasons

       9. State  = 9  /\ Rcv({LS. MU'. Psi'. K_Psi'. K1'}_Pk_LS)
                      /\       in(MU'. Psi'. K_Psi'. DT, Psi_Table)
 % LS checks the information MU, Psi and K_Psi, and looks up DT in the table.
       =|>State':= 11 /\ Loc':= new()
                      /\ secret(DT(Loc'),filtered_loc, {LR, LS, MU})
                      % in any case, MU is allowed to know its own location!
                      /\ Snd({DT(Loc')}_K1')
                      /\ witness(LS, LR, lr_ls_filtered_loc, DT(Loc'))

end role
```

```
role session(MU, LR, LS : agent,
             Psi_Table  : (agent.text.symmetric_key.function) set,
             K_MU_LR    : symmetric_key,
             Pk_LS      : public_key,
             K_MU_LS    : symmetric_key
            ) def=

local SLR, SMULR, SMULS, SLS, RMU, RLR, RLS : channel(dy)

composition

           locationRecipient(MU, LR, LS, K_MU_LR, Pk_LS, SLR, RLR)
        /\ mobileUser       (MU, LR, LS, K_MU_LR, K_MU_LS, SMULR, SMULS, RMU)
        /\ locationServer   (MU, LR, LS, Psi_Table,Pk_LS, K_MU_LS, SLS, RLS)

end role
```

```
role environment() def=

local
        Psi_Table: (agent.text.symmetric_key.function) set
        % shared between all instances of LS

const   ls_mu_psi, lr_mu_n_lr, k1, filtered_loc,
        ls_lr_k_psi, lr_ls_filtered_loc, mu_lr_lr: protocol_id,
        mu, lr, ls                    : agent,
        k_MU_LR, k_MU_i, k_i_LR : symmetric_key,
        pk_LS                         : public_key,
        k_mu_ls, k_i_ls               : symmetric_key

init    Psi_Table := {}

intruder_knowledge = {mu, lr, ls, pk_LS, k_MU_i, k_i_LR, k_i_ls}
```

```
composition

           session(mu, lr, ls, Psi_Table, k_MU_LR, pk_LS, k_mu_ls)
        /\ session(mu, lr, ls, Psi_Table, k_MU_LR, pk_LS, k_mu_ls)
        %  repeated session to check for replay attacks

        /\ session(i , lr, ls, Psi_Table, k_i_LR,  pk_LS, k_i_ls)
        %  the intruder can play the role of the mobile user MU

%       /\ session(mu, i , ls, Psi_Table, k_MU_i,  pk_LS, k_mu_ls)
%       It does not make much sense to let the intruder play the role of LR
%       since then the intruder is allowed to know the (secret) location of MU.

end role
```

```
goal

        secrecy_of filtered_loc, psi, k_psi, k1 % addresses G12

      % authentication and integrity of location object:
        authentication_on lr_ls_filtered_loc     % addresses G2 and G3

      % additional authentication goals, not in RFC3639:

        authentication_on lr_mu_n_lr             % addresses G1 and G3,
           % and G20: MU authorizes LR to receive the location via LS

        weak_authentication_on ls_mu_psi         % addresses G1
        weak_authentication_on mu_lr_lr          % addresses G1

end goal
```

```
environment()
```