To model mutual authentication between four entities (User, Things, Registration Gateway, and Server) using the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool, you can write a protocol in the HLPSL (High-Level Protocol Specification Language). Here’s an outline of how to structure the authentication process and the corresponding code for these four entities:

### Mutual Authentication Entities:

1. User to Things: The User authenticates to a "Thing" (an IoT device).

2. Things to Registration Gateway: The Thing authenticates to the Registration Gateway (RG).

3. Registration Gateway to Server: The Registration Gateway authenticates to the Server (central authority).

4. User and Server: The User and Server mutually authenticate to each other.

### AVISPA HLPSL Structure:

- Define each entity's role: `User`, `Things`, `Registration Gateway`, and `Server`.

- Specify message exchange and key usage.

- Define session, environment, and goals.

Here’s a sample HLPSL code for mutual authentication among the four entities:

```hlpsl

/\*------------------------------

Mutual Authentication Protocol

Between 4 Entities using AVISPA

------------------------------\*/

/\*------------------------------

ROLE DEFINITIONS

------------------------------\*/

/\* User Role \*/

role User(U, T, RG, S: agent, KUT: symmetric\_key, SKUS: symmetric\_key,

Snd1, Snd2, Snd3, Snd4, Rcv1, Rcv2, Rcv3, Rcv4: channel (dy)) played\_by U

/\* Local variables \*/

var

Nu, Nt, Nr, Ns: nonce,

Mu1, Mu2: message

init state1

transition

1. state1 -> state2

Rcv1(T, Nt)

=|> Nu := new()

2. state2 -> state3

Snd2(T, {Nu, Nt}\_(KUT))

3. state3 -> state4

Rcv2(RG, Nr)

=|> Snd3(RG, {Nu, Nr}\_(KUT))

4. state4 -> state5

Rcv3(S, Ns)

=|> Snd4(S, {Nu, Ns}\_(SKUS))

end role

/\* Things Role \*/

role Things(T, U, RG, S: agent, KUT: symmetric\_key,

Snd1, Snd2, Snd3: channel (dy)) played\_by T

/\* Local variables \*/

var

Nt, Nu: nonce,

Mu1: message

init state1

transition

1. state1 -> state2

Nt := new()

Snd1(U, Nt)

2. state2 -> state3

Rcv2(U, {Nu, Nt}\_(KUT))

=|> Snd3(RG, Nu)

end role

/\* Registration Gateway Role \*/

role RegistrationGateway(RG, T, U, S: agent, KUT: symmetric\_key, KRS: symmetric\_key,

Snd1, Snd2, Snd3: channel (dy)) played\_by RG

/\* Local variables \*/

var

Nr, Nu: nonce,

Mu1: message

init state1

transition

1. state1 -> state2

Rcv3(T, Nu)

=|> Nr := new()

Snd1(U, Nr)

2. state2 -> state3

Rcv4(U, {Nu, Nr}\_(KUT))

=|> Snd2(S, Nu)

end role

/\* Server Role \*/

role Server(S, U, RG: agent, KRS: symmetric\_key, SKUS: symmetric\_key,

Snd1, Snd2: channel (dy)) played\_by S

/\* Local variables \*/

var

Ns, Nu: nonce,

Mu1: message

init state1

transition

1. state1 -> state2

Rcv5(RG, Nu)

=|> Ns := new()

Snd1(U, Ns)

2. state2 -> state3

Rcv6(U, {Nu, Ns}\_(SKUS))

end role

/\*------------------------------

SESSION DEFINITIONS

------------------------------\*/

role session(U, T, RG, S: agent, KUT, KRS, SKUS: symmetric\_key)

composition

User(U, T, RG, S, KUT, SKUS, Snd1, Snd2, Snd3, Snd4, Rcv1, Rcv2, Rcv3, Rcv4)

/\ Things(T, U, RG, S, KUT, Snd1, Snd2, Snd3)

/\ RegistrationGateway(RG, T, U, S, KUT, KRS, Snd1, Snd2, Snd3)

/\ Server(S, U, RG, KRS, SKUS, Snd1, Snd2)

end role

/\*------------------------------

ENVIRONMENT DEFINITIONS

------------------------------\*/

role environment()

const U, T, RG, S: agent

const KUT, KRS, SKUS: symmetric\_key

intruder\_knowledge = {U, T, RG, S, KUT, KRS, SKUS}

composition

session(U, T, RG, S, KUT, KRS, SKUS)

/\* Security Goals \*/

goal

secrecy\_of KUT, KRS, SKUS

authentication\_on Nu, Nt, Nr, Ns

end role

```

### Explanation:

- \*\*Roles:\*\*

- \*\*User\*\*: Initiates the authentication with Things, continues with the Registration Gateway, and finally mutually authenticates with the Server.

- \*\*Things\*\*: Exchanges nonce and mutual authentication messages with the User and forwards the information to the Registration Gateway.

- \*\*Registration Gateway (RG)\*\*: Acts as an intermediary, validating the User via Things and Server.

- \*\*Server\*\*: Finalizes mutual authentication with the User, ensuring secure communication.

- \*\*Keys and Nonces:\*\*

- \*\*KUT\*\*: Symmetric key shared between User and Things.

- \*\*KRS\*\*: Symmetric key shared between Registration Gateway and Server.

- \*\*SKUS\*\*: Symmetric key shared between User and Server.

- \*\*Nu, Nt, Nr, Ns\*\*: Nonces used for mutual authentication.

### Security Goals:

- \*\*Secrecy\*\*: The keys (`KUT`, `KRS`, `SKUS`) are kept secret from the intruder.

- \*\*Authentication\*\*: Ensures authentication on the nonces exchanged between each pair of entities.

### Running in AVISPA:

- This code can be run in the AVISPA tool by saving it as a `.hlpsl` file and using the AVISPA backend tools like OFMC, CL-AtSe, or SATMC for validation.

### Next Steps:

- You can customize this model according to your system's specific requirements, such as adding additional checks or incorporating cryptographic algorithms other than symmetric key encryption.