AVISPA in the validation of Ambient Intelligence Scenarios*

Antonio Muñoz Antonio Maña Daniel Serrano
Computer Science Department
University of Málaga, Spain
{amunoz,amg,serrano}@lcc.uma.es

Abstract

Ambient Intelligence (AmI) refers to an environment that is sensitive, responsive, interconnected, contextualized, transparent, intelligent, and acting on behalf of humans. AmI environments impose some constraints in the connectivity framework, power computing as well as energy budget. This makes of AmI a significantly different case within distributed systems. The combination of heterogeneity, dynamism, sheer number of devices, along with the growing demands placed on software security and dependability, make application development vastly more complex. Also, the provision of security and dependability for applications becomes increasingly difficult to achieve with the existing security engineering mechanisms and tools. Furthermore the validation of these mechanisms is even a hard task. In this paper we present an approach to model dynamic changes in ambient intelligence scenarios using the Avispa (Automated Validation of Internet Security Protocols and Applications) model-checking toolsuite. The main goal of our approach consists on providing a starting point in the use of Formal Description Techniques (FDM) for AmI scenarios. The paper studies and assesses the suitability of the Avispa tool for security validation in Ambient Intelligent environments and proposes mechanisms to capture the dynamic context changes in these environments.

1 Introduction

The Information Society Technology Advisory Group vision is that AmI applications will be influenced by the computational, physical and behavioural contexts that surround the user. The concepts of system and application as we know them today will disappear, evolving from static architectures with well-defined pieces of hardware, software, communication links, limits and owners, to architectures that will be sensitive, adaptive, context-aware and responsive to users' needs and habits. AmI ecosystems offer highly distributed dynamic services in environments that will be heterogeneous, large scale and nomadic, where computing nodes will be omnipresent and communications infrastructures will be dynamically assembled. The applications must be able to adapt dynamically to new execution environments. As a consequence pre-defined trust relationships between components, applications and their system environments can no longer be taken for granted. Therefore, the increased complexity and the unbounded nature of AmI applications make it impossible, even for the most experienced and knowledge-able Security, to foresee all possible situations and interactions which may arise in AmI environments and therefore create suitable solutions to address the users' security and dependability requirements. An AmI environments relevant feature is that they will contain a large number of heterogeneous computing and communication infrastructures and devices that will provide new functionalities, enhance user productivity, and ease everyday tasks. However, a remarkable characteristic of these systems is that they all share a double goal: comfort and simplicity of final users. Security, privacy, and trust challenges are amplified with AmI computing model and need to be carefully engineered. However, the security is often a neglected area as AmI developers tend to ignore its importance, fascinated by many other challenges that are present in these environments. Indeed, in most cases it is considered in add-on which can be implemented later, if it all. Commonly we use Formal Description Methods (FDM) when developing and implementing communication protocols. A FDM is a specification method based on a description language that uses strict and unequivocal rules for the expressions of this language (formal syntax), as well as the interpretation of these rules (formal semantic). Specifying communication protocols with security features is even more critical and the protocol validation by means of this kind of techniques becomes essential, in order to provide a higher level of confidence to final users. Taking into account the high relation between AmI systems and human beings and the

*Work partially supported by E.U. through projects SERENITY (IST-027587) and OKKAM (IST- 215032) and DESEOS project funded by the Regional Government of andalusia

typical applications of AmI the use of FDM applied to the analysis of security properties Fh in design and analysis of AmI ecosystems is essential, and it will have a great impact in society in mid term. Among the more important characteristics of AmI to be considered when dealing with the formal specification we highlight; dynamic configuration of AmI is translated into a static security analysis; AmI security poses a high number of sometimes contradictory goals, ubiquity and the connectivity make difficult to ensure the confidentiality in these loosely coupled systems. In the following sections we will present a bird's eye view of our research experience within the AVISPA (Automated Validation of Internet Security Protocols and Applications) tool in Ambient Intelligent scenarios.

2 A Model-Checking tool for AmI:AVISPA

Model checking is emerging as a practical tool for automated debugging of embedded software [5, 9]. In model checking, a high-level description of a system is compared against a logical correctness requirement to discover inconsistencies. Since model checking is based on exhaustive state space exploration, and the size of the state space of a design grows exponentially with the size of the description, scalability remains a challenge.

A wide variety of model-checking approaches have recently been applied to analyzing security protocols, e.g. [1, 2, 7, 11, 12, 13, 14]. The key challenge they face is that the general security problem is undecidable [8], and even semi-algorithms, focused on falsification, must come to terms with the enormous branching factor in the search space resulting from using the standard Dolev-Yao intruder model, where the intruder can say infinitely many different things at any point.

The main goal of this section is to briefly describe the AVISPA model-checking tool and assess its suitability for security verification in Ambient Intelligent environments. AVISPA has already proven successful in modelling and verifying a large number of widespread security protocols. In our work we have concentrated in probably the most relevant aspect of Ambient Intelligent systems: context awareness. Our study has lead us to the development of a generic mechanism to model AmI protocols in AVISPA, including the most common networks and devices involved in typical AmI scenarios. Furthermore, traditional security protocols could later be embedded in our scheme. AmI ecosystems feature a high level of mobility and diversity. This fact makes it difficult to specify and validate these systems by means of conventional security analysis tools because it is required that specifications include all details about the solution to verify. In order to study the pros and cons of using formal security specification and validation tools in AmI ecosystems we have selected AVISPA [4] because of the independency of the underlying verification engines (back-ends).

2.1 AVISPA description and features

AVISPA is an automatic push-button formal validation tool for internet security protocols. It encompasses all security protocols in the first five OSI layers for more than twenty security services and mechanisms. AVISPA uses a High Level Protocol Specification Language (HLPSL) to feed a protocol in it; HLPSL is an extremely expressive and intuitive language to model a protocol for AVISPA. Its operational semantic is based on the work of Lamport on Temporal logic of Actions. Communication using HLPSL is always synchronous. Once a protocol is fed in AVISPA and modelled in HLPSL, it is translated into Intermediate Format (IF). IF is an intermediate step where re-write rules are applied in order to further process a given protocol by back-end analyzer tools. A protocol, written in IF, is executed over a finite number of iterations, or entirely if no loop is involved. Eventually, either an attack is found, or the protocol is considered safe over the given number of sessions. System behaviour in HLPSL is modelled as a 'state'. Each state has variables which are responsible for the state transitions; that is, when variables change, a state takes a new form. The communicating entities are called "roles" which own variables. These variables can be local or global. Apart from initiator and receiver, environment and session of protocol execution are also roles in HLPSL. Roles can be basic or composed depending on if they are constituent of one agent or more. Each honest participant or principal has one role. It can be parallel, sequential or composite. All communication between roles and the intruder are synchronous. Communication channels are also represented by the variables carrying different properties of a particular environment.

2.2 AVISPA Back-Ends

AVISPA has following four back end tools for mathematical processing. These are On-the-fly Model-Checker (OFMC), The Constraint-Logic-based Attack Searcher (CL-AtSe), SAT-based Model-Checker (SATMC) and Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP). Of the four available AVISPA Back-Ends we selected the SAT-based Model-Checker (SATMC), which builds a propositional formula encoding all the possible attacks (of bounded length) on a protocol and feeds the result to a SAT solver. The advantage of this analysis component is that automatic SAT-based model checking techniques are based on a reduction of protocol (in) security problems to a sequence of propositional satisfability problems that can be used to effectively find

attacks on protocols, which is the objective of a security protocol validation.

The language used in AVISPA is very expressive allowing great flexibility to express fine details. This makes it a bit more complex than Hermes to convert a protocol into HLPSL. Further, defining implementation environment of the protocol and user-defined intrusion model may increase the complexity. Results in AVISPA are detailed and explicitly given with reachable number of states. Therefore regarding result interpretation, AVISPA requires no expertise or skills in mathematics contrary to other tools like HERMES[3] where a great deal of experience is at least necessary to get meaningful conclusions.

2.3 First steps facing the validation of AmI protocols

Once we have described AVISPA tools and justified its use we are ready to describe the main contributions of this paper. Concretely, our research aims to study and analyze the use of AVISPA to model and validate protocols in AmI environments. For this purpose, the most characteristics of relevant AmI systems have been identified. Thus, the device and the connection heterogeneity together with the context awareness are the most important ones. Therefore, in order to model AmI systems we afford them from three different and complementary perspectives, as we mentioned above: networks, devices and context changes. Concerning networks, we considered the most widespread ones: GSM, Wifi, Ethernet and Bluetooth. We then identified and modelled the different devices involved in the common AmI scenarios, building a list of those devices together with their most relevant properties, in particular the ones of interest when modelling a full AmI environment. Some of those devices were mobile phones, pdas, laptops, pcs, tpms (Trusted Platform Modules) and smartcards. Most of them are portable: that means that the location of the devices varies and so does the set of reachable networks, possibly leading to a context change when that happens. Plus, being portable also means having batteries that, of course, can run out and hence force some sort of context change as well, so it will be another factor to consider when modelling these electronic items and their behaviours. Finally, we have assumed that each device uses a unique (symmetric) session Key whose length is different from one device to another, in order to reflect a possible difference between types and lengths of keys. Then we biefly list more common devices and properties in AmI scenarios; (i)Mobile phone is a portable that works on GSM networks. (ii)PDA is a portable device and works on GSM and Wifi networks. (iii) Laptop is a portable that works in ethernet, bluetooth and Wifi networks. (iv) The computer is not a portable device and works on ethernet, bluetooth and wifi networks. (v) And finally the TPM (trusted platform modules) and smartcards. Their properties are inherited from the devices they are attached to or embedded in. We modelled them though their own symmetric keys to somehow reflect an improvement in the degree of security they provide.

Context awareness [6] is an essential feature that AmI systems have to tackle in order to act in an adaptive and intelligent way. This context, that might be spatial, temporal, environmental, personal, social, and so on, needs to be modelled, captured, analysed and reasoned about [10]. Reasoning about context needs a model and formalization acts as a knowledge base, and enables inferring more high level knowledge. For example blood pressure and body temperature combined with user current activity and location might reveal user current mood. This mood can then be provided implicitly as an input, so AmI might take some actions as a response. The third pillar in Ambient Intelligent environments is context awareness. As we mentioned above this feature can be produced due to a change in the current network used for communication, the change of service due to low battery power, but also because of the arrival of a "new session" event motivated by a session timeout.

3 Example: THE BASIC MODEL

In this section we provide an example to model the most relevant elements of AmI systems in AVISPA. Therefore, this example is selected to show the system mechanisms developed to help in modelling complex AmI protocols. Agent A has a mobile phone that is connected to a GSM network. She will establish a communication with agent B connected to a remote Ethernet network with a PC. Agent A's side will trigger a context change when, for instance, the mobile phone runs out of battery and she will switch to her PDA to continue the communication with B using one of the two available Wifi networks, Wifi_1 and Wifi_2 (the full list of context changes will be detailed later in the model description). It would be of special interest to be able to model the process that begins in the moment that the come to action.

Five roles have been modelled in the HLSPL language:(i)The two most obvious ones are agents A and B that take part in each back end of a communication process. Their bodies include all the possible (device, network) state machines. (ii)Two network roles, one representing the specific network that agent A is currently connected to and the other one representing the Ethernet network on B's side, which we assume in this example does not change. Neither of them features any special properties (there are no network down situations, package losses, etc.). However our proposed scheme can be extended to include those additional network behaviours and characteristics inside the body of the role definition, which is one of the reasons why

we use roles to model networks (we will show later, the HLPSL supposes by default a single network that is used to send and receive messages without the need to add roles, although it has some flaws). It could actually be possible to include as many network roles as networks agent A can connect to with his mobile phone and PDA (Wifi_1, Wifi_2 and GSM), although, for the sake of simplicity in this example, we have decided to use just one network for both devices. (iii)Finally, a gateway role has been incorporated, which represents an intermediate framework entity or system that allows the transmission of information from one network to the other. It basically takes data from one network and sends them through the other network after making the required encryption adaptations, featuring additional functionality to support the creation and maintenance of sessions. It is relevant to mention that the state machine defined inside this role comprises all the (device, network) scenarios.

We have not modelled the intruder as a role itself, since in that case we would have to model all possible attack traces, task that is performed by AVISPA instead. AVISPA's HLPSL language includes an intruder model whose knowledge can be input by means of the intruder_knowledge instruction, which is powerful enough for our goals. When performing a security validation AVISPA will consider the intruder intelligent enough to use information retrieved from messages exchanged between the actors in a protocol.

One problem with HLPSL modelling is that when a message is sent through one channel all listening communication channels will receive it. That means that if agent A sends a message through the Wifi_1 network using the SND (send) channel, both Wifi_1 and Ethernet networks will receive it in their respective RCV (receive) channels. To model the fact that the Ethernet network cannot directly receive such message, we used couple of fictitious (public/private) keys for each physical network. For instance, for the GSM network, information encrypted with the key of the specific device used is additionally encrypted with the GSM network's fictitious public key.

The following diagram depicts an end to end interaction according to our model. Agent A pretends to send a certain message encrypted with the mobile's key through the GSM network. Before being transmitted the message is encrypted with the GSM network's public key in AVISPA-HLPSL (X_Y means that X is encrypted with key Y). The outcome will be called Msg_send_mobile_gsm (step 1). Then the network decrypts the message and its content is encrypted with the network's private key (Msg_send_mobile_gsm_2, step 2; inv applied to a public key is the corresponding private key). The gateway receives the message and, using GSM's public key reads its content. It then retransmits the message content through the Ethernet network B is connected to, using that Ethernet network's public key (step 3). This message is called Msg_send_ethernet and will be finally read by B after going through the Ethernet network (Msg_send_ethernet_2, step 4).

The response process on B's side would be analogous to the diagram, starting with B sending a Msg_receive_ethernet to the gateway. Gateway would receive and decrypt Msg_receive_ethernet_2 and build Msg_receive_gsm_mobile, which would be forwarded to A's mobile through the GSM network. Finally Msg_receive_gsm_mobile_2 would arrive to the mobile phone.

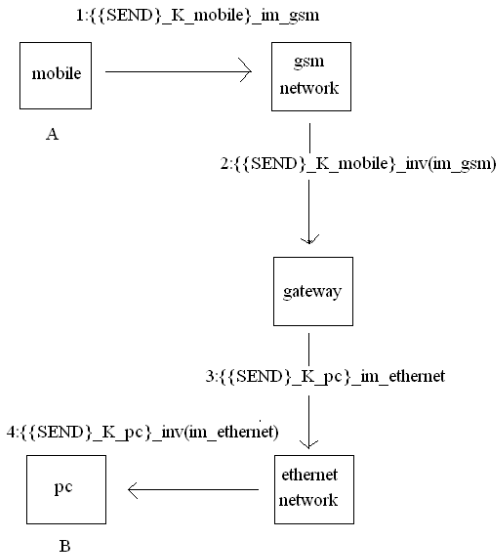

Figure 1. Example shows the messages sequence through different networks.

In our case the intruder knows both the Ethernet and Wifi network's fictitious public key, so it can send/read messages from these networks in the same way as A, B and the gateway. It also knows the agents involved in the communication. Knowledge of the Ethernet network's public key could be removed from that set to model that that network is physically safe (something unlikely to happen in practice).

SECTION 0: Asign state machine, based on initial network value and initial device.

```
INIT_MOBILE_GSM.
(State = initial) /\ (Current_device = mobile) /\
(Current_network = gsm) =|>
(State' := request_session_mobile_gsm)

INIT_PDA_GSM.
(State = initial) /\ (Current_device = pda) /\
(Current_network = gsm) =|>
(State' := request_session_pda_gsm)
```

...

SECTION 1: Every (device, network) pair carries out a simple interaction with the gateway. This interaction starts when A's device sends a session request message (for instance Msg_request_session_mobile_gsm), to which the gateway will reply with a corresponding new session message (Msg_new_session_mobile_gsm_2) in the case that a new session has been created (including a new session key for the device, in this example a mobile phone) or a reuse previous session message (Msg_reuse_session_mobile_gsm_2).

This two-step session request might indeed be more complex in actual implementations, involving a whole protocol with several message exchanges, but since the aim of this document is not to provide specific protocols the (device,network)-gateway interaction has been kept simple; consequently the types of these messages are defined as TBD (to be determined) in the local variables definition. However it is clear that the Msg_new_session_mobile_gsm_2 message should consist of, among other elements, the new session key for the mobile phone conveniently encrypted. The different Update Session tags would deal with retrieving the new session key and making the required updates so that each device can properly resume the communication with the end-point agent. In all cases after this direct interaction with the gateway has finished the regular communication with B can begin.

```
REQUEST_SESSION_MOBILE_GSM.
(State = request_session_mobile_gsm) = |>
SND(Msg_request_session_mobile_gs) /\
(State' := wait_session_mobile_gsm)

REUSE_SESSION_MOBILE_GSM.
(State = wait_session_mobile_gsm)  /\
RCV(Msg_reuse_session_mobile_gsm_2) =|>
(State' := mobile_gsm)
```

SECTION 2: Regular communication (send and receive) after the session to use has been resolved. The Last_action variable records, as its name suggests, the last action performed by the device (send or receive). This is useful after a context change has happened: A's device can then normally resume the message interaction from where it was before the change event, being the process transparent to agent B. Once again the protocol boils down to a simple send-receive process, with the highlighted messages being formed in the way described in the diagram prior to these sections (to send, first encrypt with the device's symmetric session key and then encrypt with the network's imaginary public key, to receive proceed decrypting in an analogous way).

```
MOBILE_GSM_1.  (State = mobile_gsm) /\
(Last_action = receive) =|> SND(Msg_send_mobile_gsm)
/\ (Last_action' := send)

MOBILE_GSM_2.  (State = mobile_gsm) /\
(Last_action = send) /\ RCV(Msg_receive_mobile_gsm_2) =|>
(Last_action' := receive)
```

SECTION 3: Context changes are modelled here. There is no state control on the left side of the implication, so a context change only depends on the current (device-network) couple. This means that when analyzing a protocol's security properties if A has engaged in a Section 2-like message exchange AVISPA will take into account both the regular path and the case where a context change has just happened. We are therefore taking advantage of indeterminism: from one point two requirements might be fulfilled and hence both might be validated. Moreover, there might be two context changes that share the same condition but perform different actions: (i) T1.Condition1 $\wedge$ Condition2 $\Rightarrow$ Action1. (ii) And T2.Condition1 $\wedge$ Condition2 $\Rightarrow$ Action2, which might increase the number of paths until three. The context changes detailed are the following (they are just a subset of all the possible context changes in our scenario): (i) The mobile phone runs out of battery and Wifi 1 network is reachable: then the PDA is activated in Wifi_1 network. (ii) The mobile phone runs out of battery and wifi 2 network is reachable: then the PDA is activated in Wifi_2 network. (iii) The PDA runs out of battery and is connected to Wifi_1. We have the possibility to use the mobile phone in GSM. (iv) The PDA runs out of battery and is connected to Wifi_2: The same as the case above. (v) The PDA is connected to Wifi_1 and user moves away so his location changes to a new location where Wifi_1 is not covered but Wifi_2 is reachable. Then the PDA connects to Wifi_2. (vi) The PDA is connected to Wifi_2 and user moves to a new location where Wifi_1 is reachable and Wifi_2 is not covered: Then the PDA connects to Wifi 1. (vii) And finally, the mobile phone is connected to a GSM network and a new session has been created by the gateway for the user due to a spontaneous timeout. Session information stored in the device must be updated (including the new value of K_mobile).

The right side of these implications simply redirects to section 4 where the context change really takes place.

```
CONTEXT_CHANGE_MOBILE_GSM_BATTERY:
(Current_device = mobile) /\ (Current_network = gsm) =|>
(State' : = transition_mobile_gsm_battery_1)

CONTEXT_CHANGE_MOBILE_GSM_BATTERY:
(Current_device = mobile) /\ (Current_network = gsm) =|>
(State' : = transition_mobile_gsm_battery_2)
```

SECTION 4: Essentially, here the current network and device are updated, depending on the nature of the context change, and a session request from Section 1 is started. Before that further unspecified actions could be made, probably dealing with context change recovery-related procedures.

```
TRANSITION_MOBILE_GSM_BATTERY_1:
(State = transition_mobile_gsm_battery_1) =|>
ACTIONS /\ (Current_device' := pda) /\
(Current_network' := wifi_1) /\
(State' := request_session_pda_wifi_1)
```

```
TRANSITION_MOBILE_GSM_BATTERY_2:
(State = transition_mobile_gsm_battery_2) =|>
ACTIONS /\ (Current_device' := pda) /\
(Current_network' := wifi_2) /\
(State' := request_session_pda_wifi_2)

role rolenet(Net:agent,SND,RCV:channel(dy))
played_by Net
...
GSM.RCV({M'}_im_gsm)=|> SND({M'}_inv(im_gsm))
WIFI_1.RCV({M'}_im_wifi_1 =|>
SND({M'}_inv(im_wifi_1))

WIFI_1.RCV({M'}_im_wifi_2 =|>
SND({M'}_inv(im_wifi_2))
end role
```

The gateway role can be divided into two sections: a section for the sessions, symmetric to role A's section 1, dealing with session requests by agent A and generating spontaneous new sessions (this one would be a context change triggered by the gateway), and the regular message section, where normal messages from A are forwarded to B and vice versa. See how to model the creating a new key (a fresh value) we use the X' := new() assignment.

```
role
rolegateway(Gateway,A,B:agent,SND,RCV:channel(dy))
 played_by Gateway
...
REUSE_SESSION_MOBILE_GSM:
Previous_session = true /\
RCV(Msg_request_session_mobile_gsm_2) =|>
Current_device' := mobile /\
Current_network': = gsm /\
SND(Msg_reuse_session_mobile_gsm)
NEW_SESSION_MOBILE_GSM:
RCV(Msg_request_session_mobile_gsm_2) =|>
K_mobile' := new() /\
Current_device' := mobile /\
Current_network' : = gsm /\
ADDITONAL ACTIONS /\
Previous_session' := true /\
SND(Msg_new_session_mobile_gsm)
SPONTANEOUS_SESSION_MOBILE_GSM:
Current_device = mobile /\
Current_network = gsm =|>
K_mobile' := new() /\
ADDITONAL ACTIONS /\
Previous_session' := true /\
SND(Msg_new_session_mobile_gsm)
...
SEND_MOBILE_GSM.
RCV(Msg_send_mobile_gsm_2) =|>
SND(Msg_send_ethernet)

RECEIVE_MOBILE_GSM.
Current_device = mobile /\
Current_network = gsm /\
RCV(Msg_receive_ethernet_2) =|>
SND(Msg_receive_mobile_gsm)
...
end role
```

On B's side we only show the regular message reception and answer section. Regarding context changes, we only contemplate for B the new session context change that has its origins in a gateway's timeout.

```
role
rolenet2(Net2:agent,SND,RCV:channel(dy))
played_by Net2
...
ETHERNET.RCV({M'}_im_ethernet) =|>
SND({M'}_inv(im_ethernet))
end role

role
roleB(A,B,Gateway:agent,SND,RCV:channel(dy))
played_by B
...
RECEIVE_ETHERNET.

RCV(Msg_send_ethernet_2) =|>
SND(Msg_receive_ethernet)
...
end role

role
session(A,B,Net ,Net2,Gateway,Start_device: agent,
Start_network: protocol_id)
...
composition
roleA(A,B,Gateway,Start_network,
Start_device,SND1,RCV1)/\
roleB(A,B,Gateway,SND2,RCV2)
/\rolenet(Net,SND3,RCV3)/\
rolenet2(Net2,SND4,RCV4)/\
rolegateway(Gateway,A,B,SND5,RCV5)
end role

role environment()
...
intruder_knowledge=
{a,b,gateway,net,net2,im_wifi_1,im_wifi_2,im_ethernet}
....
session(a,b,net,net2,gateway,mobile,gsm)
end role

environment()
```

The HLSPL code shown above simply models the scenario represented in the diagram at the beginning of this document. For this model to be suitable for AVISPA validation, security specific predicates and blocks must be included in the code. Briefly, every time a message is sent then at the end of that sentence a witness instruction should be written to impose that authentication (strong or weak one) is required; similarly a request/wrequest instruction should be written on the receiver's side. The secret instruction would indicate that some value must be a secret among a set of specified agents. We could apply this one to our model, as we are concerned about the secrecy of the different session keys (K_mobile, K_pda, etc.). We would have the following line in Section 2 of roleA:

```
MOBILE\_GSM\_1.
(State = mobile\_gsm) /\
(Last\_action = receive) =|>
SND(Msg\_send\_mobile\_gsm) /\
(Last\_action' := send) /\
secret(K\_mobile,secret\_1,{A,Gateway})
\vspace{0.5cm}
```

The last action predicate means that the session key K_mobile used by the mobile phone in encrypting the message Msg_send_mobile_gsm should only be known by the mobile phone itself and the gateway (the latter is indicated in the form of a set as third parameter); secret_1 would be

the identifier of this secrecy condition and is needed because at the end of the whole model description a goals section will state which of the goals must be validated, referencing the identifiers of all secret/authentication predicates specified in the code. In our case this section would simply be: secrecy_of secret_1.

A final specification should include a secret(A,B,C) predicate after every message sent by agent A. Although it is not implemented yet, it is expected that future versions of AVISPA will support validation of more complex goals, taking into account time aspects (always in the future, some time in the future, etc). In fact, actually a syntax exists for time-related goals that is successfully parsed and translated by AVISPA, although it is not functional in reality and cannot be used. Properties derived of indeterminist network or context change behaviours could be validated if such goals were supported. Unfortunately other properties such as the probability of an event happening do not seem that could be possible to model in the near future.

4 Conclussions

We showed new approach for modelling AmI system with AVISPA. This approach is based on the point of view of devices, networks and especially, context changes. These context changes are the real innovation in these ecosystems. Then we conclude this section with some considerations in the use of AVISPA tool, as well as a discussion about improve this tool.

AVISPA provides functionality to validate protocols in circumstances where context changes can happen. Then a plan of an attack when a context change is produced can be provided, such as in latter example shown.

We identify certain desirable features among these available in AVISPA. We must highlight the role parameterization, as a useful mechanism. This facilitates the modelling process of AmI protocols in AVISPA.

One aspect to improve is the fact that channels are defined to work in broadcasting mode. This obliges us to use symmetric fictitious keys to indicate the destination, which complicates the final code and does not provide an elegant and natural way to represent this situation. Concerning the functionality provided, we believe that a mechanism to provide protocol bi-simulation is missing. Protocols in real world may interfere when they are executed simultaneously. Also some mechanism to model temporal generic predicates, such as "..if A sends to B then do.." is lacking. However, it is foreseen that the next version of this tool provides this functionality. Avispa protocol simulation tests if any intruder is sniffing in a concrete channel, then a plan where this situation is possible is found. However, the analyst cannot obtain statistics or probabilities values.

Despite of these limitations, Avispa has proven to be very useful tool. One of the more relevant advantages to work with Avispa is that we do not need to model intruders, as an additional role. The tool provides mechanisms to generate plans automatically. Then all state spaces are generated from a knowledge that we provided.

References

[1] R. Amadio and D. Lugiez. On the reachability problem in cryptographic protocols. *In Proceedings of Concur'00.*, LNCS 1877, Springer-Verlag:pp. 380–394., 2000.

[2] M. Boreale. Symbolic trace analysis of cryptographic protocols. *In Proceedings of ICALP'01.*, LNCS 2076, Springer-Verlag.:pp. 667–681, 2001.

[3] L. Bozga and Y. Lakhnech and M. Perin. Hermes,a tool verifying secrecy properties of unbounded security protocols. *In 15th international conference on Computer-Aided Verification (CAV'03).*, Lecture Notes in Computer Science, Springer Verlag., 2003.

[4] Y. Chevalier and L. Compagna and J. Cuellar and H. Drielsma and J. Mantovani, S. Mödersheim, and L. Vigneron. A high level protocol specification language for industrial security-sensitive protocols. In *Proceedings of Workshop on Specification and Automated Processing of Security Requirements (SAPS)*, Linz, Austria, September 2004. (13 pages).

[5] E. Clarke and O. Grumberg and D. Peled. Model checking. *The MIT Press.*, 1999.

[6] A. K. Dey and G. D. Abowd. Towards a better understanding of context and context-awareness. *In the Proceedings of the CHI 2000 Workshop on The What, Who, Where, When, and How of Context-Awareness.*, 2000.

[7] B. Donovan and P. Norris and G. Lowe. Analyzing a library of security protocols using casper and fdr. *In Proceedings of FMSP'99 (Formal Methods and Security Protocols).*

[8] N. Durgin and P. Lincoln and J. Mitchell and A. Scedrov. Undecidability of bounded security protocols. *In Proceedings of FMSP'99 (Formal Methods and Security Protocols).*, 1999.

[9] G. Hotzmann. The model checker spin. *IEEE Trans. Software Engineering.*, 23(5).:279–295, 1997.

[10] J. Krogstie. Research areas and challenges for mobile information systems. *International Journal of Mobile Communication.* ., 2(3), 2004.

[11] J. K. Millen and V. Shmatikov. Constraint solving for bounded-process cryptographic protocol analysis. *In Proceedings of CCS'01.*, ACM Press.:pp. 166–175, 2001.

[12] J. C. Mitchell and M. Mitchell and U. Stern. Automated analysis of cryptographic protocols using murphi. *In Proceedings IEEE Symposium on Security and Privacy.*, 1997.

[13] P. Ryan and S. Schneider and M. Goldsmith and G. Lowe and B. Roscoe. Modelling and analysis of security protocols. *Addison Wesley.*, 2000.

[14] D. Song and S. Berezin and A. Perrig. Athena: a novel approach to efficient automatic security protocol analysis. *Journal of Computer Security.*, 9:47–74, 2001.