A Comparative study of Security Protocols Validation Tools: HERMES vs. AVISPA

Mureed Hussain , Dominique Seret
UFR Math-Info Université René Descartes Paris5
45 rue des Saints Pères 75270 PARIS cedex 06 - France
{mhussain, Seret}@math-info.univ-paris5.fr

***Abstract* — Security protocols require more rigorous and detailed verification than normal communication protocols before their deployment because even a trivial flaw in their architecture may produce drastic results. These verification procedures are based upon the abstract formal methods producing analytical rules to show if a given protocol is secure or not. The technical difficulties to master formal techniques by non-mathematicians have given rise to online validation tools which are easy to use and produce a human readable output. These online tools differ in many parameters. Our work is aimed to present a qualitative comparison of two automatic security protocol analysers Hermes and AVISPA for the some pertinent parameters.**

***Keywords* — Formal validation, security protocols, Hermes, AVISPA, EVA, HLPSL.**

1. Introduction

Securing information requires development of new security protocols. Two fundamental needs for any secure communication are confidentiality and authentication. Furthermore, it is believed that a security protocol must fulfil our needs of integrity and non-repudiation etc. It is a well known fact that many good security protocols had flaws in them which became clear after year of their deployment. Although every such shortcoming is promptly patched, nevertheless, it leaves behind always a feeling of absence of good techniques to verify a protocol's intended properties before hand that is before implementation. This need gave rise to a whole series of formal methods to investigate formally pros and cons of any protocol. Verification of security protocol is just a special case of traditional model-checking techniques where the systems are cryptographic protocols in the hostile network and properties to be verified vary from case to case.

There are four wide classifications [1] which exist in the literature on how these formal methods are conceived and apply. We shall give a very brief introduction of each of these four methods.

The first group of these methods is called Inference-constructed methods which are based on the concept of logic and belief. Through an idealized form and employing logical rules at each step of the protocol, we infer some beliefs (after application of postulates) which show agreement or disagreement with the initial protocol specification. This group consists largely on BAN logic [2] and its variants like GNY logic, Higher Order Logic (HOL) theory, SvO logic (combination of BNa & GNY, VO logics) etc .

Second group [3] can be referred as Proof-construction methods a protocol is defined on the bases of reliable and unreliable principles. Precise roles of each participant and their beliefs are clearly separated. Then, theorems are proved to prove or disprove secrecy of a protocol specification. Generally, at the end, human-readable results are produced.

Third type of methods is attack-constructed formal techniques [6] where probable attacks are constructed from the algebraic properties of the protocols and all possible states of the protocol are verified. Main disadvantages of these techniques is large number of probable states in which a system my go.

In fourth type of formal methods, expert systems are designed which are independent of a particular protocol. The protocol (to be tested) is written is a specific language which is then verified for security properties. Examples of this type of tools include NRL protocol analyzer and Interrogator [9] although they differ in the procedure of how a protocol is proved secure or insecure.

All the methods described above involve very deep understanding of the tools in order to verify security properties of a protocol. In most of the cases, they are like special purpose applications which require significant changes in code each time we wish to analyse a new protocol. Furthermore, each of these tools requires a different language format of specification. This has given rise to the idea of developing a common formal specification language, which might possibly be used in tools which automatically analyse a protocol. Efforts in this domain have given birth to CAPSL (Common Authentication Protocol Specification Language) [11][15]., ISL (Interface Specification Language) [4]which are used in Hermes and AAPA (Automatic Authentication Protocol Analyzer) respectively.

The main problem faced in automatic protocol verification has its origin in the unbounded nature of parameters which are to be verified. Further more, related to this difficulty are the following points:

- The number of sessions of a protocol is not bounded
- The number of participants is not bounded
- The attacker has unlimited memory capacity
- There is no limit on the size of messages known to the attacker
- The attacker is capable of creating new keys and new nonce

These are the general consideration for and formal validation tool and almost all of them take care of these problems.

Rest of the paper is arranged as follows. After introduction and context of the subject, section 2 explains differences in architecture and components of both tools following detailed explanation on the front-end languages in section 3. Section 4 and 5 describe intrusion models and verifiable services, respectively for the both analyzers. Finally in section 6 we briefly underline the different back-end analyzing mathematical tools of both following our observations on the ease of use of each of them and complexity in section 7. We conclude in section 8 with some future directions to explore.

2. Functional Architecture and Components

HERMES is a dedicated tool of secrecy verification of the cryptographic protocols using a front end language EVA, especially developed for this purpose. Once a protocol has been clearly defined in standard formal notations, it is then transformed into EVA script explicitly defining the assumptions on all exchanges and the claims which are to be held in order to prove that an intended exchange is secure or not. With the help of user given set of assumption and claims a set of sufficient conditions is calculated to prove secrecy of a protocol. In context of HERMES, secrecy means a specific message should not be made public. A secret will be considered public if its deduction becomes possible from a set of intercepted messages known to intruder or publicly accessible.

HERMES is based on 'protector messages' which guarantee the protection of a secret. These are encrypted messaged with a secret key and is assumed that an intruder can't see them or access them. For a given protocol with secret messages S and hypotheses H on the secret keys, we calculate, on one hand a set of messages is calculated which are responsible of protection of ordinary messages during the execution of the protocol. On the other hand, it adds messages to the secrets which may be helpful to constitute an attack. Those messages are particularly important for this purpose, in which secrets are not protected. For a graceful termination of these calculations, an operator of convergence is used in Hermes. At the end of calculation, Hermes return a set of secrets S' and the protected messages H'. These two sets define conditions in which a protocol can be used without any danger of revealing secrets. That is, protocol secrecy is guarantied if, secrets S' are protected in H' in the messages initially known to an intruder. The proof that a message's' is secret involves three hypotheses; all possible states are reachable, there is no bound number of sessions and there is no limit on the size of the messages during the execution of the protocol.

A typical EVA script has essentially four parts. These are declarations, Initial knowledge, Messages knowledge and Session & Secrets. Declaration section contains variables, algorithms, key types, and principals and initial knowledge section describes the basic knowledge every participant of the protocol possess about every other including certificates or relation of trust if it exists between them. Message knowledge includes all the exchanges which make up of the protocol in EVA syntax. This section equally shows all cryptographic operations which are to be performed on the messages or on some parts of them with the respective keys. Last section comprising sessions and goals lists all the secrets and sessions with respect to principals and variables.

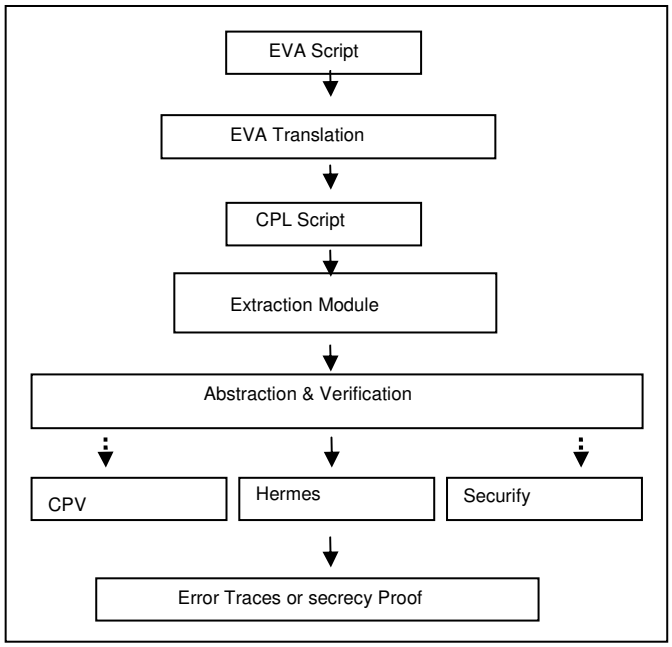

Figure 1. Functional Architecture of HERMES

Automatic Verification of Internet Security Protocols and Applications (AVISPA) is an automatic push-button formal validation tool for internet security protocols. It has been developed in a project sponsored by the European Union. It encompasses all security protocols in the first five OSI layers for more than twenty security services and mechanisms. Further to it, AVISPA covers (that is verifiable by it) more than 85% of IETF security specifications. AVISPA library

available on-line has in it verified with code about hundred problems derived from more than two dozen security protocols.

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 modeled in HLPSL, it is translated into Intermediate Format (IF) [5]. 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.

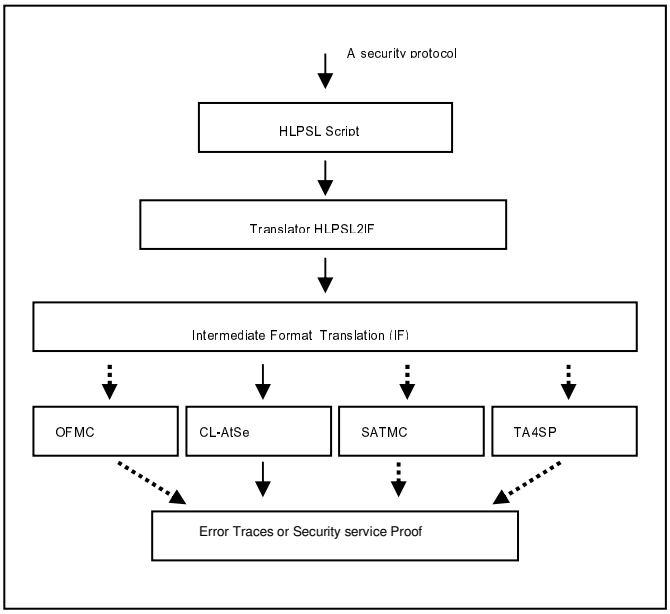

Figure 2. Functional Architecture of AVISPA

System behavior in HLPSL is modeled 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. A typical role in HLPSL can be written as:

```
role Basic_Role (…)
played_by … def=
  owns {θ: Θ}
  local {ε}
  init Init
  accepts Accept
  transition
    event1 ⇒ action1
    event2 ⇒ action2
    …
end role
```

3. Front-end Languages

The language used in HERMES is EVA which is based on CAPSL [11] and its design goals cited by its creators include expressiveness, readability, precision and reusability. We have found it simple and robust to use as compare to the language used in AVISPA. The objectives of readability and reusability are well achieved because EVA syntax is quite straightforward even for a non-programmer and some parts of the same code can be reused for modelling anew protocol. On these two points EVA stands higher than the language of AVISPA However, when it comes to the two remaining goals, that is expressiveness and precision EVA lacks significantly the level and depth provided by HLPSL. In context of expressiveness, for example, in EVA there is no notion of how to differentiate between different environments. Environment can be crucial to the implementation of a security protocol and lack of a procedure to express a given environment in terms of variables can be a real problem. Contrary of EVA, language used in AVISPA treats environment as any other principal and expresses it into a set of variables intuitively. This functionality of HLPSL has given it tremendous flexibility to be used for security protocols created for different environments. Regarding precision also, EVA lacks full support to write down a cryptographic algorithm and its variants. This is not only possible but also more deeply embedded mechanism in AVISPA.

Looking from AVISPA language point of view, here HLPSL is used which is very powerful language with design objectives modularity, expressivity, and embedded role-based orientation as compare to EVA. It allows complicated flow patterns and data structures to be defined and expressed along with the state and environment as roles also. As a general principle, the architecture of AVISPA is so much flexible that any language which is convertible to Intermediate Format (IF) can be used in AVISPA. Apart from the modelling bricks which will be described in the following paragraphs, flow control in the form of if-then-else statements.

The basic specification of a given protocol is done in form of several roles. There are multiple types of roles. They can be

basic describing the action of one principal in a single protocol or sub-protocol. Two or more roles can constitute a composite role. An example of a basic role is given in previous section. Transitions from one state of protocol to another are also important feature of AVISPA. They are like immediate reactions to some actions associated by some special characters. After every transition the state is updated.

The possibility of introducing new function symbols in HLPSL gives it a clear advantage over EVA. Although variables and constants in HLPSL are types to avoid type-flaw attack, however, AVISPA lets use also un-typed model by bypassing the type information if it is required. It is equally possible to introduce new operators in HLPSL apart from the classical ones like exponentiation and XOR. This gives AVISPA a greater flexibility over HERMES for the implementation of new cryptosystems.

However, the price to pay for expressing all fine details is its considerable complexity. Contrary to EVA, communication environment is also a Role is HLPSL and has to be expressed as a principal. All roles, their messages and their respective interactions with the environment make HLPSL fairly complex than EVA. This adds additional work and usability problems for non-expert users.

4. Intrusion Models

Hermes uses Dolev-Yao intruder model [6] which supposes that an intruder has all means to interfere the network and can capture traffic as much as required for analyses. Also, the intruder has unlimited time to attack the network and its capacities in terms of available power and memory are also unlimited to break the system. Moreover, it can intercept or emit messages, split or constitute messages and can re-generate messages but it can't prevent the legitimate principals to receive the messages.. Further to this it is assumed that the cryptographical functions used in the protocols are perfect, that is, cryptanalysis attacks aren't possible on them. This is the intrusion model which has been implemented in most of the formal methods. Hermes and AVISPA both employ this model. Hermes implements the complete standard model in it with no options to modify or redefine it.

Although AVISPA uses the same model and approach for defining intruder model, however it gives an edge over Hermes by allowing integration of any new models in HLPSL and hence going into deeper details of the communication channel and environment. Thus in AVISPA, the possibility to define and integrate new intruder models according to the specific environment and need comes naturally while in Hermes there isn't any such possibility. Typically intruder model is defined as a function of the quality of the channel on which the communication is taking place. More over, there is better and greater flexibility of reconfiguration the model parameters in AVISPA than Hermes.

5. Verifiable Security Services

Main objective of any security protocol is provision of security services. A flaw in the protocol introduces vulnerabilities due to which attack on the protocol or on its particular step of execution becomes possible. Therefore it is very important that before using automatic analyses, one makes sure which of the security services can be verified. With this point of view, Hermes provides only secrecy proof against an eavesdropping attack. That is, attacks are verified only against security services: confidentiality and privacy. To some extent communication authenticity is also possible. If it becomes possible to 'see' a message which has been encrypted with the pre-defined secret keys or shared keys, then Hermes shows the messages, breakable keys, and the patterns in form of 'bad-patterns'. On the other hand, if all the communication and keys are securing then the result is termed as good-patterns. Also, it is also possible to verify message authenticity with the same way but not the participant's authenticity.

AVISPA has greater capability than Hermes in terms of verifiable security services**.** The list of security services and mechanisms which are possible to verify in AVISPA include confidentiality, integrity, authentication of communication and data origin, anonymity, third-party authorization, non-repudiation and key-management properties and of a given protocol or communication exchange. It is worth noting that this list is not exhaustive.

Contrary to EVA where secrets and sessions are defined for respective principals without specification of parallel or sequential, in AVISPA goals are precisely defined with respect to a required security service and roles and special syntax is used for parallel or sequential sessions. After the compilation of the code (in HLPSL) AVISPA produces results in the form of reachable number of states if specified goals are achieved. In case of an unsafe protocol, that is if a specified objective could not be achieved with the given set of messages, AVISPA will not be able to run the program for finite number of states thus convergence wouldn't be possible.

6. Back-end Analytical Tools

There are three back-end tools in HERMES to verify the security properties. These are Cryptographic Protocol Verification (CPV) [12], HERMES [4] and SECURIFY [13]. SECURIFY uses pre-defined theorems to compute the secret or the rules which will lead to three possible results; yes, fail and no-response. Yes, fail and 'no-response' signify 'successful compilation-safe protocol', 'unsafe-flawed-protocol', and 'un-terminated algorithm', respectively. It uses the model of Millen-rueß which performs backward research and results in one of the three possible outcomes stated above. Securify takes .cpl which is itself outcome of .eva script file. This EVA file contains the protocol and the security properties to be verified.

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).

The back-end analytical tool OFMC is used for a protocol where algebraic properties of the cryptographic functions are important. After the intermediate processing (IF), several transition states are explored on-demand.

CL-AtSe is constraint based system. Its strategy is to simplify the original protocols and eliminate redundancy states using heuristics. It is built in a modular way so that new specifications for cryptographic functions can easily be integrated. SATMC takes a transitional state from IF and looks for any security property violation. It generates a formula depicting the violation and translates it into an attack.

TA4SP shows vulnerability of a protocol or predicts its soundness by a careful estimation of the intruder's capabilities. The use of CL-ATSE is preferable in an environment where there is need to take into account the algebraic properties of the algorithms.

7. Usability and Complexity

HERMES is generally easier than AVISPA to use but significantly harder when it comes to interpret the results in case a flaw is detected by the back-end analytical tool. Any eventual flaw is translated into dozens of mathematical rules which we have found difficult to interpret or understand. However, the error messages in case of a syntax error are explicitly shown before compilation. In case a given protocol is secure, result shows merely "good patterns" without giving any detail of on what bases this result is obtained and what are the steps followed. Thus, Hermes is easier for expressing a protocol into EVA but harder to get the meaningful results.

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. Debugging is also relatively difficult than EVA. On the other hand, results in AVISPA are more detailed and explicitly given with reachable number of states. Therefore regarding result interpretation, AVISPA requires no expertise or skills in mathematics contrary to HERMES where a great deal of experience is at least necessary to get meaningful conclusions.

8. Conclusion and Future Work

In this paper we have presented a qualitative overview of the two automatic formal validation tools for security protocols. We have compared them with respect to their complexity, front-end languages, verifiable security services, intrusion models and back-end analytical tools. Whole process of formal validation can be divided into tree steps namely abstraction, processing and result interpretation. Hermes is generally good for first step but relatively insufficient for second and third steps. Therefore, for simple protocols requiring only secrecy proof we recommend Hermes while for big and complex protocols where security is critically important with varying environments we suggest AVISPA (because it will let you define your own intrusion model according to your needs).

The main interest of our research group lies in the development of light weight security protocols for the networks comprising of devices with limited capacities of CPU, battery and memory. We are exploring different options of formal validation for these protocols. We intend to study thoroughly and deeply AVISPA and bring appropriate modification to suite our special needs.

REFERENCES

[1] Stefanos Gritzalis, Diomidis Spinellis, and Panagiotis Georgiadis. Security protocols over open networks and distributed systems: Formal methods for their analysis,design, and verification. Computer Communications, 22(8):695-707,May 1999

[2] Burrows M., Abadi M., Needham R., A Logic of Authentication, *ACM Transactionson Computer Systems*, 8(1), (1990) 18-36.

[3] Bolignano D., An approach to the formal verification of cryptographic protocols,*Proceedings of the Third ACM Conference on Computer and Communications Security*, (1996) 106-118, ACM Press.

[4] Liana Bozga, Yassine Lakhnech and Michael Périn. 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, July 2003.

[5] L. Bozga, Y. Lakhnech, and M Périn. L'outil de vérification HERMES. Technical Report EVA-6-v1, Verimag, Grenoble, France, May 2002. Available from http:://www-eva.imag.fr

[6] D. Dolev and A. Yao. "*On the Security of Public-Key Protocols"*. IEEE Transactions on Information Theory, 2(29), 1983.

[7] Y. Chevalier et al," *A High-Level Protocol Specification Language for Industrial Security-Sensitive Protocols*",: www.avispa-project.org

[8] L. Lamport.," *The temporal logic of actions"*. ACM Transactions on Programming Languages and Systems, 16(3):872–923, May 1994.

[9] Meadows C., The NRL Protocol Analyzer: An overview, *Journal of Logic Programming*, Vol. 26, No. 2, (1996) 113-131.Kemmerer R., Analyzing encryption protocols using formal verification techniques, *IEEE Journal on Selected Areas in Communications*, 7(4), (1989) 448-457.

[10] http://www.avispa-project.org

[11] Grit Denker, Jonathan Millen, and Harald Ruess. The capsl integrated protocol environment.Technical report, SRI International, 2000

[12] J. Goubault-Larrecq. A method for automatic cryptographic protocol verification. In International Workshop on Formal Methods for Parallel Programming, Theory and Applications. volume 1800 of LNCS, 2000.

[13] Technical report on EVA No 13, A guide for SECURIFY, Véronique Cortier, December 2003. http://www-eva.imag.fr/bib/EVA-TR13.pdf

[14] M. Burrows, M. Abadi, and R. Needham. A Logic o Authentication. *ACM Transactions on Computer Systems*, 8(1):18–36, 1990.

[15] P. Calhoun, J. Loughney, E. Guttman, G. Zorn, and J. Arkko. RFC 3588: Diameter Base Protocol, Sept. 2003. Status: Proposed Standard.

[16] J. Clark and J. Jacob. A Survey of Authentication Protocol Literature: Version 1.0. www.cs. york.ac.uk/˜jac/papers/drareview.ps.gz, 1997.