*J. Appl. Environ. Biol. Sci.*, 4(7S)199-209, 2014

© 2014, TextRoad Publication

ISSN: 2090-4274
Journal of Applied Environmental and Biological Sciences
*www.textroad.com*

# Parametric Analysis of Formal Systems through Multi Criteria Analytical Hierarchy Process

**Faisal Javed**

Department of Computer Science, Abdul Wali Khan University Mardan

*Received: September 1, 2014*
*Accepted: November 13, 2014*

## ABSTRACT

Ubiquity & mobility in computing systems challenges the requirement engineering process for stating the non functional requirements of the system. One of the solutions is to go for formal system analysis and design. Formal methods help tailor the systems design and augment a solution in to the system at the very beginning that is precise and authentic. Formal analysis tools work towards formally analyzing the systems and thus proving them capable with formal verifications. Selection of formal analysis tool on certain criteria especially when the candidates are in abundance and learning curve of each of them is steep makes the decision hard. In this paper, empirical based multi criteria decision making technique is used for enabling good decisions for the selection of formal tool.

**KEYWORDS:** Formal Analysis, Automated Tools, Multi Criteria decision Making, Analytic Hierarchy Process.

## 1 INTRODUCTION

At present, existing systems are ubiquitous, mobile and performing uniformly 24/7. Providing quality e.g. 'security' to these systems at runtime like firewall has the static nature of locality and in many cases cannot be mobilized. Formal methods comes with a baggage of such software practices which focuses thoroughly on the early stages of SDLC and can tailor systems for attributes expected of them. So, adding quality features to any or all levels of SDLC is handled with a rigor in formal methods. Formal methods comprises of: 1). Mathematical concepts and informal analysis 2). Formalized specification language 3). Automated theorem prover/model checker (formal analysis tools) [1]. According to [2], one can get reference to more than 60 proof tools. For software engineer, who wants to perform formal verification of a secure computing system, instead of handwritten mathematical proof there is an increasing interest to select automated proofing mechanism to verify the stated requirement. Humanly it is not possible to verify the correctness of cryptographic protocol where state space explorations are vast for unbounded number of behaviors. For example Needham-Schroeder Protocol [3] considered secure for seventeen long years until G. Lowe detected flaw using automated tool Casper/FDR [4]. This example shows automated analysis is rudimentary for proving correctness of the system.

Methods and tools constituting formal methods are mooted difficult, pose a high degree of learning complexity and are challenging to start with. The literature regarding these tools normally addresses their strength but not discuss their weaknesses or the applications. Users of such tools find the information insufficient in the selection of tool. As a beginner, one needs proper assistance in selecting a formal tool where the description is not based on abstract of their manual but should be based on practical experiences and framework of criteria for selection [5]. Most of the work found in literature regarding comparison of formal tools is based on experimental data. Tools are compared for their behavior and abilities on the set of protocols. Cheminod et al. [6] compares four state exploration automatic tools (OFMC, SAT, S3A and Casper) for their features and error detection abilities of known flaws over a set of cryptographic protocols. In [7], a comparison of four automated formal verification tools namely the Frege Program Prover (FPP), the KeY system (KeY), Perfect Developer (PD) and Prototype Verification System (PVS) is performed to check their suitability for teaching an advance course in formal software verification. Evaluation of these formal tools is based on specification and

*** Corresponding Author:** Faisal Javed, Department of Computer Science, Abdul Wali Khan University Mardan. Email: faisaljaved@awkum.edu.pk

verification of small programs taught as assignment problems in the course. In [8, 9], a comparison is made between the automatic tools on the model of state space explorations. Some tools explore all the behaviors possible while some only the subset of it called scenarios. In [10], C. Meadows compares NRL and FDR on the example of Needham-Schroeder and concluded that two tools are complementary to each other. FDR being quick in exploring state space but requires user intervention for proving correctness. NRL is slower as compared to FDR but it explores finite number of state spaces automatically without user intervention. K. Healy et al. [11] compares four state space tools for verifying security protocols namely NRL, FDR, Brutus and Athena. Evaluation is based on state space techniques, a useful mean for finding attacks, reproducing known flaws and discovers new flaws. Limited work is found in literature where formal tools are compared on their features and internal structure. In [5], a qualitative comparison based on features of PVS and Isabelle/HOL is presented. A qualitative comparison of two automated formal proof tools Hermes and AVISPA is presented in [12]. All of these comparisons were limited and none explicitly state the best selection where multiple options are considered. A study which is based on quantitative data and comparison that involves multi-criteria decision making technique.

This paper aims to provide a dossier regarding the selection of formal tool on multi criteria decision making technique. Section 2 briefly explains the five most common formal analysis tools. Section 3 focuses on importance of decision making and Analytical Hierarchy Process (AHP). Section 4 provides the implementation result of applying AHP, contribution of each criteria in decision, sensitivity analysis and trade-off chart. Finally, paper is concluded with section 5.

## 2 Tools for Formal Analysis

A large number of formal analysis tools are available for the formal verification of formal specification. Study of some of the most common formal tools is presented one by one.

### 2.1 CASPA

Stands for “Causality Based Abstraction for Security Protocol Analysis”. Caspa [13] is a push button tool which Supports multiple, simultaneous and yet automated execution of security protocols and effectively can check them for authenticity and secrecy. This doesn’t cause any attenuation or performance compromise to caspa. Caspa follows the cause and effect based calculus built on the notations referred to as Causal graphs. In causal graphs, each node represents an activity where as their effects are represented by the edges. Graphs in Caspa are finite and dependent. Hence these are best suited to demonstrate the protocols executions on an abstract level e.g. the cause effect relation can be written as (authenticity | do(password)). This will be read as authenticity can be checked by forcing passwords. This case provides us with a cause and effect relation, Caspa can audit and show us any irregularity in such relation. This methodology can help the security engineers to test and design a correct protocol structure and amend errors on an early stage. Caspa assures the guaranteed termination of protocols due its primitive nature of finite state causal graphs [14]. The tool can be accessed costless at: [15].

### 2.2 AVISPA

Stands for “Automated Verification of Security Protocols and Applications”. Avispa [16] is an automatic verification and a push button model checker for security sensitive protocols used over large networks, like the Internet. Avispa has contributed to the most exhaustive security protocols set, those have been verified for security concerns. According to [12] “All security protocols functioning on the first 5 OSI layers are verifiable using Avispa and it comprehends more than 85% of Internet Engineering Task Force (IETF) Security Specifications”.

Avispa is composed of components which are related to front end and the back end. It has a layered architecture, at the top of which is a protocol designer which acts as a front end and is used to specify the protocols formally. HLPSL (High Level Protocol Specification language) is used to serve the specification purpose, which in its core is Avispa specific, highly spontaneous, communicative, coherent and role-based language. The manner in which a system behaves is referred to as ’state’ in

HLPSL. Entities taking part in protocol execution, be it the receiver, an initiator or an environment all are referred to as 'role'. All the 'states' have associated variables which keep track of any change in the state. The scripted HLPSL specification of the protocol is provided to the second in hierarchy component, that is IF (intermediate Format). IF acts as an intermediary, between the HLPSL layer and the backend tools, where higher level specifications are re-written to be used by one or all of the backend tools to obtain the desired results. The transformation of specification of the security protocol written in HLPSL to intermediate Format is performed using an intrinsic function knows as HLPSL2IF. Any protocol bearing attributes pertaining to privacy and integrity are all verifiable through Avispa. When a protocol is terminated, it's easy to interpret its results, since if an attack is found, its traces are also generated, the problem apprehended follows the logic which may be affirmative of disconfirming, whether the given resource pool was fully the depleted or the system could not find any justifiable output [16]. The background components of Avispa are comprised of independent functioning units which are seamlessly integrated as following:

- OFMC (On the Fly Model Checker): A Model checker for protocol analysis, falsification and state space exploration.
- CL- AtSe (Constraint logic based attack searcher): Analyzes inference based restraints by employing pre and post conditions. Has strong system of logic and supports elimination of dually used facts.
- SatMC (SAT-based Model-Checker): It takes notations from IF, if susceptible to attack converts it to axiomatic formula and provides this as an input to SAT, which finds anomalies in the proof and counter measures the security breaches.
- TA4SP (Tree Automata based on Automatic Approximations for the Analysis of Security Protocols): works by careful approximate calculation of the degree of mal-ware in protocol. Since it makes use of HLPSL which is formally modular and composite, it can evaluate the protocol as computationally strong or weak in anticipation to intruder's strength.

### 2.3 Scyther

Scyther [17, 18] is an automatic model checker and a theorem prover which analyzes, verifies and falsifies the security protocols. Scyther is well known for its performance and flexibility. It supports for multiple concurrent protocol executions as well as the multiple sessions of the same protocol. Scyther has two basic components a GUI and a command line tool. The GUI is an optional facility which has dual functionality. First component works as a visual editor and second displays the results. The command-line tool is written in C and vouches for the performance and verification of large protocol sets. The command-line tool can work independent of the GUI. Scyther mentors individually each step of protocol analysis the specification, communication model, the implementation, or the construction of the threat model. The Scyther proclaims its efficiency on the basis that it assures the (i) guaranteed termination of the cryptographic protocols, (ii) bounded sessions- that is newly generated literals are allowed to be evaluated, which results in secrecy assurance. and finally (iii) the generation of the schematics of proof in the form of a tree. Scyther assists to analyze the protocol domain with help of a language known as SPDL. SPDL is based on operational semantics [19]. SPDL represents the protocol behavior as a role. Each role has associated security attributes that it normally stands to. These attributes are denoted as claims. In proof process the correctness of the claim is checked if it's true or not. e.g.; a role can claim to have been uncompromised and authentic. If a role does not have any security claim, Scyhter can update the role with generic claims or can generate a few for evaluation purposes. Such claims are known as automatic claims. Since a protocol may exhibit different behaviors at different instants, Scyther performs the complete characterization of the protocol in question. In Scyther, each behavior instance is represented as role. The complete trace which is finite for each role is generated such that even if an attack exists, it can be pointed out manually even by looking at it. The protocol proof execution, in scyther, returns true or false for the claims or none in case no discrepancy is found. For attacks apprehended, explicit traces are yielded in the form of trace Graphs or are described in XML.

### 2.4 ProVerif

ProVerif is a fully automatic cryptographic protocol verifier that analyzes the security protocols for stated security properties and is developed by Bruno Blanchet. ProVerif takes input in two formats either directly in horn clauses (logical programming rules) or indirectly in extended pi calculus (process) which again translated into horn clause for resolution. Security properties are provided as "queries". Horn clause input is used to check the secrecy of the cryptographic protocol, if fact can be generated from the clause, proof is given. If result is true, no attack is found. If the result is false, attack is found against desired properties and attack trace is generated. If the result is unknown means don't know, cannot be proved, attack trace cannot be reconstructed. ProVerif uses Dolev-Yao as threat model to capture potential threats and attackers. Error messages are translated using emacs. Output of the analyzer is either as Solve (go ahead and analyze) or Spass (to stop before resolution) [20, 21]. ProVerif is used for verification of reachability and secrecy properties (secrecy of the term is tested using secrecy queries and hence proving reachability properties), correspondence assertions (to draw relationships between events in the form "if event then event"), observational equivalence (in distinguishability to proof some complex security properties that cannot be defined as reachability or correspondence assertions), proving authentication and privacy, for reconstruction of attacks and termination result on tagged protocols (protocols with no existence of loops inside) [21]. The distinguishing factors of ProVerif could be that it is fully automatic, models attack that might consists of any number of concurrent protocol execution and verifying protocols for unbounded number of session and unbounded message space. ProVerif has limitations while proving and modeling algebraic properties likes XOR and Diffie-Hellman key exchange. It limits the size of state space for proving certain properties; the technique is called unification [22]. ProVerif is supported with web service tool TulaFale. One of the variant of ProVerif tool is CryptoVerif tool, same goal but computationally stronger than ProVerif and less widely used as compared to ProVerif due to its complexity. ProVerif proves protocol in formal model and can reconstruct attacks while CryptoVerif proves protocol in computational model but cannot reconstruct attacks [21]. ProVerif is used for many day to day applications like in Bluetooth security, electronic voting, secure file sharing, analyzing integrity of email protocols, JFK (Just Fast Keying) protocol, TLS (Transport Layer Security) protocol, authentication protocol in TPM (Trusted Platform Module) and in verifying security of Plutus file system [21,22]. ProVerif is developed using OCaml and requires OCaml (3.0 or higher version) as pre-requisite for installation. It's freely downloadable software and is compatible with Linux, Mac and Windows platform. ProVerif software is distributed under GNU general public license for Linux and BSD license for windows platforms. It's a command line tool and installed using source or binaries [21].

### 2.5 SPIN

SPIN is a very efficient formal model checker used for the design and verification of distributed / concurrent systems. Spin as a formal analysis and verification tool has following objectives [23].

- Design specifications are expressed in a programming like language PROMELA (Process Meta Language).
- Correctness requirements for proving correctness of claims are expressed using notations of LTL (Linear Temporal Logic).
- A methodology for establishing logical relevance in design specifications and correctness requirements.

As spin cannot handle unbounded number of verifications therefore syntax/model specified in promela must be bounded and countable behaviors are verified. Spin starts with the specification of a distributed/concurrent software system in a high level language promela using a GUI XSPIN, the front end application. After the design specifications are free from syntax errors the next step is to perform interactive simulation until the program behaves as intended. In the third step, verification of the specification is done using on the fly verifier. If any disprove in the correctness claims are found it is resubmitted to the interactive simulation to remove the causes [23].

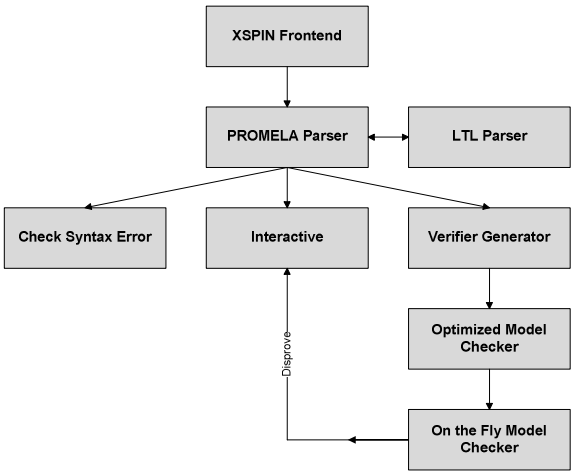


**Figure 1.** Spin structure for specification and verification [23]

The correctness claims are represented as LTL formula. Spin converts LTL to Buchi automata. Computation on a set of global state space and synchronous product of the claim gives another buchi automaton to prove whether the claim is satisfied or not. [23].Spin represents each process as a finite automata, the resulting global system behavior is represented as finite automata. The interleaving Cartesian product of the global behavior of the system is referred to as global state space. This state space represented in a form of graph is called as global reachability graph [23].Spin uses a single procedure to compute a single concurrent component and a buchi automaton of a correctness claim using depth first search. In the best case, if the claim proves correct by on the fly verifier, the state space size is zero. In worst case, it is Cartesian product of claim and system and in normal case, the depth first search algorithm terminates when an acceptance state is found. It is important to know that LTL makes spin an effective verifier, compared to its predecessors, as any LTL formula can be translated to buchi automata [23].

**1 Selection of Formal Analysis Tool and Decision Making**

The selection of formal analysis tool is a decision making process. The tool selection is a very important decision and has incredible impact on the related processes of system development and evolution.

**3.1 Software Engineering Decision Support (SEDS)**

The process of evolution of software development life cycle leads to the development of new tools, methods and techniques. However, we are still dealing with the problem of expected software quality behavior, time to market, budgetary and resource issues. So the challenge is to provide empirical based guidelines for enabling good decisions for the selection of tools, methods and techniques in all aspects of software development. This is the main purpose of Software Engineering Decision Support (SEDS). Decision support is needed for the following reasons [24]:

- Decision problems are not taken seriously, subsequently they are poorly described and understood
- No mechanism is used to make a decision, therefore decision are unplanned and taken at 11th hour.
- Decision taken is not based on any empirical evaluation, model and experience.
- Many stakeholders have conflicting interest and decision taken not considers the interest of all related stakeholders.
- Decision may cause the violation of the critical constraint and is often detected too late.
- Impacts and consequences of the decision are ignored.

- Decision does not involve transparency and are not open to criticism.
- Decision making is not executed and aided using any intelligent tool.

So decision support is needed to cover the entire realm of software development life cycle from 'Software Requirement Engineering' to 'System Deployment'. Decision support is needed to describe, evaluate, sort, rank, select or reject candidate products, COTS, processes, technique or tools [24].

### 3.2 Multi-Criteria Decision Making (MCDM)

Tool selection is a multi-criteria decision making (MCDM) [25] problem. MCDM or MCDA is an acronym used for Multiple Criteria Decision Making / Analysis, a well-known selection problem structuring and solution technique which involves decision and analysis when multiple, complex and conflicting criteria are considered and give solution in the form of best alternative from the set of multiple alternatives. It supports decision maker when an optimal solution to some decision problem does not exists. The basic activities of MCDM include [26]:

- Establishing a set of criteria for selection
- Assigning weights to each criterion representing its importance in the selection under consideration.
- Evaluating the priority of each alternative against each criterion.
- Ranking of the alternatives based on a set of criteria.

One of the most promising techniques of MCDM is Analytical Hierarchy Process (AHP).

### 3.3 Analytical Hierarchy Process (AHP)

AHP was developed by Thomas L. Saaty in late 1970s. It's a technique for solving complex decision problems involves mathematics and human judgment. AHP is a formal method used to derive ranking from pair-wise comparisons technique which is based on ratio scale [27]. AHP assist numerous social, economic, business, government decision makers in policy making, marketing strategy, system selection, program selection, resource planning etc. The process of AHP involves following step [28], [29].

| | |
|---|---|
| STEP 1 | Identify Objectives |
| STEP 2 | Identify Criteria, Sub-Criteria And Intensity Level |
| STEP 3 | Identify Alternatives |
| STEP 4 | Draw Hierarchal Tree |
| STEP 5 | Make Pair-wise Comparison Between Criteria, Sub-Criteria Based On Their Intensity Level |
| STEP 6 | Make Pair-wise Comparison Between Alternatives Based On Ratio Scale |
| STEP 7 | Make Your Assessment Consistent |
| STEP 8 | Develop Relative Priority And Ranking |
| STEP 9 | Aggregate The Result |

**Figure 2**. Process of AHP

## 4 IMPLEMENTATION RESULTS

Selection of formal analysis tool (Objective) is based on ISO/IEC 9126 standard for software product quality evaluation. The criteria include Efficiency, Reliability, Usability, Functionality and Portability. Maintainability is not an issue in selection of best fit formal analysis tool that are available as Commercial Off-The-Shelf (COTS) or free-wares as an academic versions. Moreover, maintainability is not a measurable attribute for factors [30]. The criterion of Maintainability therefore is not considered for the efficient selection/decision making process.

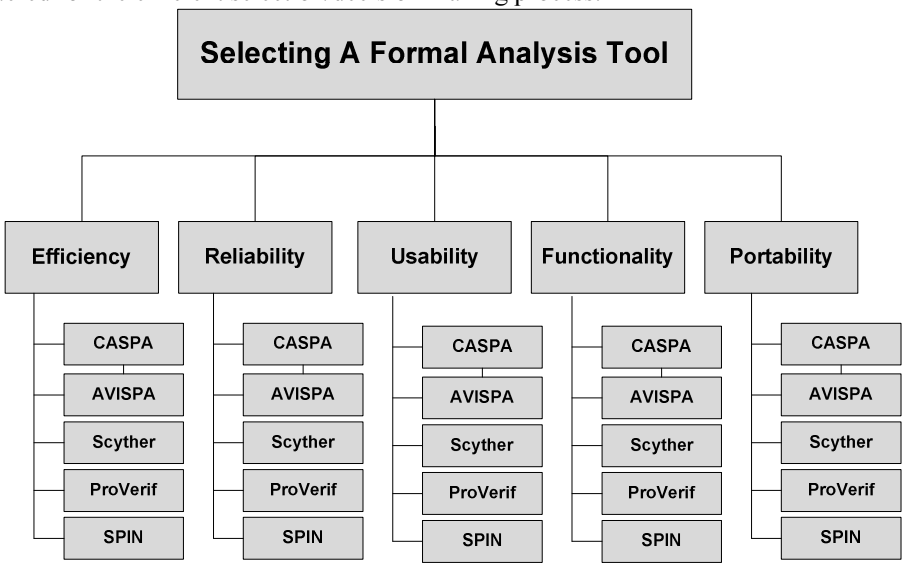


**Figure 3.** Hierarchal Tree / AHP Decision Model

Once the AHP decision model is generated next step is to rate the criteria and alternatives to determine the relative importance of criteria and alternative over one another. Three methods are available to weight the criterion, direct comparison method, pair-wise comparison method and abbreviated pair-wise comparison method. Selecting the weighting method depends on the problem, available information and decision maker experiences. [31]. We use pair-wise comparison method to weight our criteria and alternatives. The result of weighting method in the form of decision scores is shown.

**Table 1.** Result data of pair-wise comparison method

| Lowest Level | CASPA | AVISPA | Scyther | ProVerif | SPIN | Model |
|---|---|---|---|---|---|---|
| **Efficiency** | 0.075 | 0.168 | 0.080 | 0.215 | 0.462 | 0.104 |
| **Reliability** | 0.440 | 0.154 | 0.069 | 0.154 | 0.183 | 0.559 |
| **Usability** | 0.247 | 0.080 | 0.086 | 0.080 | 0.508 | 0.085 |
| **Functionality** | 0.222 | 0.070 | 0.171 | 0.056 | 0.481 | 0.188 |
| **Portability** | 0.220 | 0.082 | 0.173 | 0.082 | 0.442 | 0.065 |
| **Results** | 0.331 | 0.129 | 0.097 | 0.131 | 0.312 | -- |

Result of the comparison is shown as decision scores that reflect best to worst ranking of alternatives.


**Figure 4.** Decision scores for selecting the best alternative

### 4.1 Criteria Contributions and Data Analysis

The chart of criteria contribution shows the contribution of each criterion in each alternative to achieve the goal. The stacked bar histogram of criteria distribution is shown with contribution data table.

**Table 2.** Data sheet for contribution of criteria in decision scores

| | Reliability | Functionality | Usability | Portability | Efficiency |
|---|---|---|---|---|---|
| **CASPA** | 0.246013 | 0.0417141 | 0.0209036 | 0.0142576 | 0.00778827 |
| **SPIN** | 0.102029 | 0.0903141 | 0.0430585 | 0.028628 | 0.0479842 |
| **ProVerif** | 0.0862109 | 0.010431 | 0.00676335 | 0.00531372 | 0.0223053 |
| **AVISPA** | 0.0862109 | 0.0131582 | 0.00676335 | 0.00531372 | 0.0175093 |
| **Scyther** | 0.0385599 | 0.0319897 | 0.00725423 | 0.0111932 | 0.0083335 |

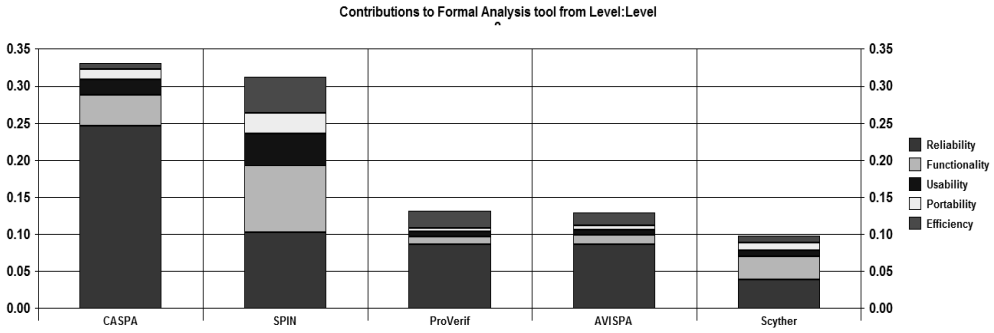


**Figure 5.** Histogram of contribution of criteria in decision scores

### 4.2 Sensitivity Analysis

Sensitivity analysis shows the sensitivity of the decision to change when the relative importance of the criteria changes. Sensitivity analysis displays alternatives' decision scores to weights of the criterion with the criticality of the criterion to the decision when the value of criterion is changed. Sensitivity analysis performed for the selection of formal analysis tools shows that reliability is the most critical criterion compared to other criteria with criticality of 3.4% which means that a change of 3.4% of weight in the value of reliability can change the result or structure of the model. Criticality of efficiency is evaluated as 4.1%, functionality is 5.5%, usability is 6.1% and least critical is portability with criticality equals 7.3%.

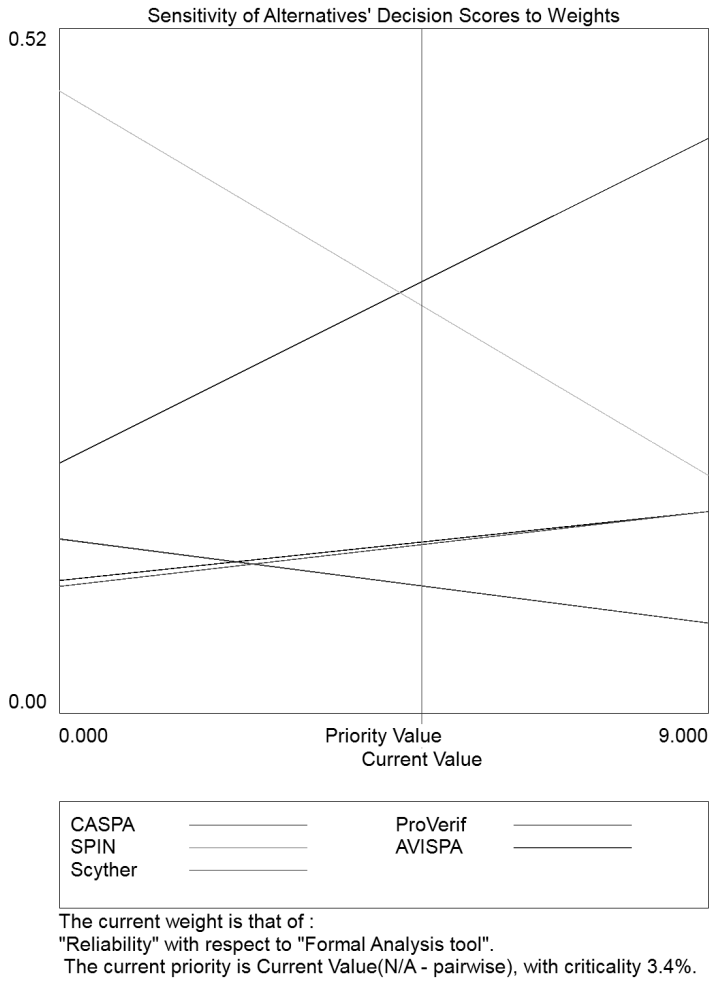


**Figure 6.** Sensitivity analysis of alternatives to Reliability

### 4.3 Trade-offs

Trade off means a cost of losing a quality attribute to gain certain amount of quality attribute. Result shows that reliability is the most significant or critical criteria whereas portability is least significant. Trade off result for one unit change in reliability shows 1.67 unit change in efficiency, 2.00 unit change in usability, 1.33 unit change in functionality and correspondingly, 2.50 unit change in portability. The relative weights show the relative importance of each criterion with respect to reliability.

**Table 3.** Trade off result for one unit change in reliability

| Tradeoff | Scale Units | Worst | Best | Relative Weight | Name |
|---|---|---|---|---|---|
| **1.67** | Default (%) | 0.00 | 100.00 | 60% | Efficiency |
| **1.00** | Default (%) | 0.00 | 100.00 | 100% | Reliability |
| **2.00** | Default (%) | 0.00 | 100.00 | 50% | Usability |
| **1.33** | Default (%) | 0.00 | 100.00 | 75% | Functionality |
| **2.50** | Default (%) | 0.00 | 100.00 | 40% | Portability |

## 5 CONCLUSION

Many formal analysis tools are available for formal verification of secure computing systems. Yet there is no study available on the precise and concrete selection of a tool based on multi criteria of quantitative or qualitative nature. Learning all of these tools or a wild guess in selection is a problem. Other way out is to use some decision support mechanism that is based on expert judgment. A study which is consistent in judgment and is taken from more than one human expert based on judgment. Our solution is based on hierarchal, statistical and mathematical approach i.e. AHP to solve the multi criteria selection problem of purely formal analysis tools. On the contrary, AHP involves human judgement in pair-wise comparison of the order pairs which makes the comparison biased. We tried to remove biasing and make the judgement consistent but still due to leaning nature of human in favour of some alternative the judgements are not completely pure. Our result shows quantitatively that CASPA is the best alternative out of SPIN, ProVerif, AVISPA and Scyther when multiple criteria for selection based on reliability, efficiency, usability, functionality and portability are considered.

Current implementation in AHP could be expanded to other tools and other qualitative/quantitative criteria. Techniques other than AHP e.g. Fuzzy AHP could be used and results are compared. Potential work includes the selection of formal analysis tool on quality metrics as defined by ISO/IEC 9126-4 or other quality models.

## REFERENCES


1. Kelly, John C. "Formal Methods Specification and Analysis Guidebook for the Verification of Software and Computer Systems Volume II: A Practitioner's Companion." (1997).
2. Kohlhase, Michael, and Carolyn Talcott. "Database of existing mechanized reasoning systems." (1999): 6.
3. Needham, Roger M., and Michael D. Schroeder. "Using encryption for authentication in large networks of computers." Communications of the ACM21.12 (1978): 993-999.
4. Lowe, Gavin. "Breaking and fixing the Needham-Schroeder public-key protocol using FDR." Tools and Algorithms for the Construction and Analysis of Systems. Springer Berlin Heidelberg, 1996. 147-166.
5. Griffioen, David, and Marieke Huisman. A comparison of PVS and Isabelle/HOL. Springer Berlin Heidelberg, 1998.
6. Cheminod, Manuel, et al. "Experimental comparison of automatic tools for the formal analysis of cryptographic protocols." Dependability of Computer Systems, 2007. DepCoS-RELCOMEX'07. 2nd International Conference on. IEEE, 2007.
7. Feinerer, Ingo, and Gernot Salzer. "A comparison of tools for teaching formal software verification." Formal Aspects of Computing 21.3 (2009): 293-301.
8. Cremers, Cas JF, Pascal Lafourcade, and Philippe Nadeau. "Comparing state spaces in automatic security protocol analysis." Formal to practical security. Springer Berlin Heidelberg, 2009. 70-94.
9. Cremers, Cas, and Pascal Lafourcade. Comparing state spaces in automatic security protocol verification. ETH, Department of Computer Science, 2007.
10. Meadows, Catherine A. "Analyzing the Needham-Schroeder public key protocol: A comparison of two approaches." Computer Security—ESORICS 96. Springer Berlin Heidelberg, 1996.
11. Healy, Kieran, Tom Coffey, and Reiner Dojen. "A comparative analysis of state-space tools for security protocol verification." WSEAS Transactions on Information Science and Applications 1.5 (2004): 1256-1261.

12. Hussain, Mureed, and Dominique Seret. "A comparative study of security protocols validation tools: HERMES vs. AVISPA." Advanced Communication Technology, 2006. ICACT 2006. The 8th International Conference. Vol. 1. IEEE, 2006.
13. Backes, Michael, et al. "The CASPA tool: Causality-based abstraction for security protocol analysis." Computer Aided Verification. Springer Berlin Heidelberg, 2008.
14. Backes, Michael, et al. "The CASPA Tool: Causality-based Abstraction for Security Protocol Analysis (Tool Paper)."
15. Available: http://www.infsec.cs.unisaarland.de/caspa/download.htm
16. Armando, Alessandro, et al. "The AVISPA tool for the automated validation of internet security protocols and applications." Computer Aided Verification. Springer Berlin Heidelberg, 2005.
17. Cremers, Casimier Joseph Franciscus. "Scyther: Semantics and verification of security protocols." Dissertation Abstracts International 68.02 (2006).
18. Cremers, Cas JF. "The Scyther Tool: Verification, falsification, and analysis of security protocols." Computer Aided Verification. Springer Berlin Heidelberg, 2008.
19. Cremers, Cas, and Sjouke Mauw. "Operational semantics of security protocols."Scenarios: Models, Transformations and Tools. Springer Berlin Heidelberg, 2005. 66-89.
20. Blanchet, Bruno. "ProVerif automatic cryptographic protocol verifier user manual." CNRS, Departement dInformatique, Ecole Normale Superieure, Paris(2005).
21. Blanchet, Bruno, and Ben Smyth. "ProVerif: Automatic Cryptographic Protocol Verifier User Manual & Tutorial (2011)."
22. Peters, M., and P. Rogaar. "A review of ProVerif as an automatic security protocol verifier." (2011).
23. Holzmann, Gerard J. "The model checker SPIN." IEEE Transactions on software engineering 23.5 (1997): 279-295.
24. Ruhe, Günther. "Software engineering decision support: methodology and applications." Innovations in decision support systems 3 (2003): 143-174.
25. Parker, Kevin R. "Selecting software tools for IS/IT curricula." Education and Information Technologies 15.4 (2010): 255-275.
26. Triantaphyllou, Evangelos, et al. "Multi-criteria decision making: an operations research approach." Encyclopedia of electrical and electronics engineering 15 (1998): 175-186.
27. Zhai, Yun. Non-numerical ranking based on pairwise comparisons. Diss. McMaster University, 2010.
28. Özdağoğlu, Aşkın, and Güzin Özdağoğlu. "Comparison of AHP and fuzzy AHP for the multi-criteria decision making processes with linguistic evaluations."İstanbul Ticaret Üniversitesi Fen Bilimleri Dergisi 6.11 (2007): 65-85.
29. Saaty, Thomas L. "Multicriteria Decision Making: The Analytic Hierarchy Process: Planning, Priority Setting." Resource Allocation 2 (1990).
30. Mohamed, Abdallah Sami Abbas Shehata. Decision support for selecting COTS software products based on comprehensive mismatch handling. Diss. University of Calgary, 2007.
31. Eldrandaly, Khalid. "GIS software selection: a multicriteria decision making approach." Applied GIS 3.5 (2007): 1-17.