ISSN: 2229-6948(ONLINE)
DOI: 10.21917/ijct.2017.0226

# CRYPTOGRAPHIC PROTOCOLS SPECIFICATION AND VERIFICATION TOOLS - A SURVEY

**Amol H. Shinde[1], A.J. Umbarkar[2] and N.R. Pillai[3]**
*[1,2]Department of Information Technology, Walchand College of Engineering, India*
*[3]Scientific Analysis Group, Defence Research & Development Organisation, New Delhi, India*

*Abstract*

***Cryptographic protocols cannot guarantee the secure operations by merely using state-of-the-art cryptographic mechanisms. Validation of such protocols is done by using formal methods. Various specialized tools have been developed for this purpose and are being used to validate real life cryptographic protocols. These tools give feedback to the designers of protocols in terms of loops and attacks in protocols to improve security. In this paper, we discuss the brief history of formal methods and tools that are useful for the formal verification of the cryptographic protocols.***

***Keywords:***

***Formal Verification, Security Protocols, Tools for Formal Analysis***

## 1. INTRODUCTION

Specification of a complete set of steps to be executed by two or more parties with an explicit description of how to proceed in every contingency for carrying out any particular activity is known as protocol. A communication protocol which uses cryptographic operations is known as Cryptographic protocol. Cryptographic protocol contains functions such as distribution of keys to the entities, authentication of principals to each other to make the secure transaction or computation over the network.

Usually the network is assumed to contain adversaries. The adversary has total control of the network by which he can remove any message and insert any new message he is able to construct in the network. The adversary may also have control over some of the principals working in the network. Though the cryptography is assumed to be perfect i.e. it is not possible to undo the encryption or get anything from the plaintext, without knowing the key for decryption. The adversary is able handle the data sent in the protocol to form attacks on the protocol. Some attacks may be dependent on the subtle properties of the cryptographic algorithms or the statistical analysis of message traffic. Formal methods help in checking whether such attacks are possible on the protocol.

Formal methods are somewhat a combination of a mathematical or a logical model of a system and its requirements, together with an effective procedure for determining whether the proof that a system satisfies its requirements is correct [1]. In protocol, several entities want to send some data securely but the adversary can perform a finite set of some operations such as intercepting, concatenating and de-concatenating, encryption and decryption of the data to attack. Such attacks are often non-intuitive so some kind of discrete formal analysis can be used to find them and avoid them. Because of this, it is realized that formal methods are helpful in analysis of cryptographic protocols.

The rest of this paper is organized as follows; section 2 gives brief history of formal methods. Section 3 explains review of existing tools for formal verification of cryptographic protocols. Section 4 gives the work completed using the tools. Section 5 gives the discussion on the protocol specification and verification.

## 2. BRIEF HISTORY OF FORMAL METHODS

Probably formal methods were first mentioned as a possible tool for analysis of cryptographic protocols by Needham and Schroeder in [2], but Dolev and Yao [3] are the first who actually have done work in this area and later Dolev, Even and Karp [4] developed a set of polynomial-time algorithms which are helpful in analyzing the security of restricted class of protocols [1]. Dolev and Yao developed a formal model with an environment in which multiple protocol instances can be running concurrently and in which the cryptographic algorithms obey some set of algebraic properties such as encryption and decryption operations cancel each other.

This model includes an intruder who is able to read, alter and delete traffic and is able to control some members. This model has been used in most of the later work done in the formal analysis of cryptographic protocols. Tools were developed by researchers using the Dolev-Yao model for the analysis of cryptographic protocols. Most of the tools were based on the state space exploration technique. In this technique; a state space is defined and it is then explored by the tool, to check if bad states are reachable. Paths through the space by which an adversary is able to reach a bad state is a successful attack.

Later Burrows, Abadi and Needham published a logic knows as BAN logic [5]. BAN logic has a different approach. It is based on logic of belief. This consists of a set of model operators which describe the relation between principals and data, possible beliefs the principals can hold and inference rules to derive new beliefs from old ones. Lowe demonstrated that by using a general purpose model checker; FDR (Failures Divergences Refinement), the man-in-the-middle attack on the Needham-Schroeder public key protocol can be found out [6].

After this, research got focused on state exploration tools and theorem proving techniques which are based on the Dolev-Yao model. In 1994, Peter Ryan first suggested the idea of using FDR and the CSP (Communicating Sequential Processes) language, and in [7], Roscoe first published work on this approach [1]. In [1], Catherine Meadows has given a history of application of formal methods for analysis of cryptographic protocols and discussed some of the work done in this area as well as described some of the new challenges in the area and the ways in which way they are being met.

## 3. TOOLS FOR FORMAL VERIFICATION

There are number of tools available these days for checking the correctness of protocol. These tools are based on the techniques like state space exploration, theorem proving and type checking. Some these tools are discussed in the rest of this section.

### 3.1 INTERROGATOR

John Millen et al. developed Interrogator [8] in 1983. The earliest version was similar to a finite state model checker. Interrogator is a Prolog program. It is used to search for vulnerabilities in protocol for cryptographic key distribution purpose. It looks for active attacks which defeat the protocol objective. It has to do searching for finite number of rounds to find attack. Interrogator is implemented in LM-Prolog on a Lisp machine [8].

It has a graphical user interface. For Interrogator protocols are specified in text format, edited with Lisp machine facilities, parsed and loaded. Then to set up the attack objective; interactive display is used. If vulnerability is found, it is displayed using a message sequence. This message sequence shows messages before and after modification by an adversary who might be a legitimate user and who is assumed to know the protocol format and all public information.

### 3.2 INATEST

Inatest [9], developed in 1987 by Kemmerer, is a general purpose symbolic execution tool. It analyzes encryption protocols using machine aided formal verification techniques. It can be used to reproduce attacks. It was developed for the purpose of formally specifying the elements of a cryptographic network and the rules and the actions of the cryptographic protocols.

The formal method used here is the Formal Development Methodology (FDM), it is an example of state machine approach to formal specification. The formal specification language used is the variant of Ina Jo [10]. The language is a nonprocedural assertion language. It is an extension of the first-order predicate calculus. Inatest has the ability of specifying algebraic or any other properties that may be relevant to the protocol, but the analysis is done manually without automatic algebraic reductions.

### 3.3 NRL PROTOCOL ANALYZER

The NRL Protocol Analyzer is a tool for the specification and analysis of cryptographic protocols, which has been used on various complex real life protocols [11]. It is a tool written in Prolog. It was developed in 1989 by C. Meadows in Naval Research Laboratory; that is why the name is NRL. It is an automated intruder similar to Interrogator implementing full Dolev-Yao model.

The NRL protocol analyzer uses the approach which models the protocol as an interaction between a set of state machines and attempts to prove a protocol secure. To prove the protocol is secure, insecure states are specified and proofs for unreachability of these states is attempted by using either backward search from the state or proof techniques for reasoning about the state machine models. It uses a symbolic representation of states and supports the use of lemmas to reduce infinite state space to a finite one.

### 3.4 PROVERIF

ProVerif is a tool for automatically analyzing the security of cryptographic protocols [12]. A fully automatic technique is provided by ProVerif to verify correspondences in the security protocols. For ProVerif only specifying the protocol and the correspondences in that protocol is needed. Correspondences are the properties which show that if some event is executed in the protocol then the protocol must have executed some other events before. The events can be described using a logical formula which can contain conjunctions and disjunctions. The process calculus is used for representing the protocol. It is an extension of pi calculus containing arbitrary cryptographic primitives. It is extended with events which are used in the statement of correspondence. These events are the required annotations of the protocol. Then the protocol gets translated into Horn clauses. A resolution based solver is then used for these Horn clauses. The numbers of sessions of the protocol and the size terms the adversary is able manipulate have no bounds. The tool has the limitation that the solving algorithm sometimes in exceptional cases does not terminate. However, unlike state space exploration, a proof with this tool will hold for unbounded number of rounds.

### 3.5 SCYTHER

Scyther [13] is a push button tool with graphical user interface for verification, falsification and analysis of cryptographic protocols. Scyther has features like the possibility of unbounded verification with guaranteed termination, analysis of infinite sets of traces in terms of patterns and multiprotocol analysis support. It is based on pattern refinement algorithm. Bounded and unbounded numbers of runs are verified in Scyther, using a symbolic backwards search based on patterns. The command-line and Python scripting interfaces make it easy to use Scyther for large-scale protocol verification tests. For Scyther, description of the protocol is in the spdl (Security Protocol Description Language) and this specification can be used in following three ways, first; to verify whether the security claims which are described in the description of the protocol hold or not, second; to generate appropriate security claims for the selected protocol automatically and verify them, and third; for analyzing the protocol by doing complete characterization. Every protocol role can be characterized. Scyther analyzes the protocol to provide a finite representation of all traces that contain an execution of the protocol role.

### 3.6 AVISPA

AVISPA [14] (Automated Validation of Internet Security Protocols and Applications) is a push-button tool. A modular and expressive formal language named High Level Protocol Specification Language (HLPSL) is provided by AVISPA for specifying protocols and properties [14]. AVISPA combines different back-ends having different automatic protocol analysis techniques which range from protocol falsification to abstraction based verification methods for infinite number of sessions. The current version of AVISPA integrates four back ends viz. the On-the-fly-model-checker (OFMC), the SAT based Model Checker (SATMC), the Constraint-Logic-based Attack Searcher (CL-AtSe), and the Tree Automata based on Automatic Approximations for the Analysis of Security Protocols (TA4SP) protocol Analyzer [14]. These back-ends analyze protocols assuming there is perfect cryptography and the messages exchanged in the network are controlled by the Dolev-Yao intruder. When protocol terminates, each back end outputs the result of analysis using output format which states whether the input format is solved, system resources are exhausted or the problem is not tackled. AVANTSSAR is further development of the AVISPA. AVANTSSAR is a technology used for the formal specification and Automated VAlidatioN of Trust and Security of Service-oriented ARchitectures.

## 3.7 TAMARIN PROVER

The TAMARIN Prover is a tool for automated symbolic analysis of security protocols with unbounded number of sessions for complex properties [15]. TAMARIN uses backward reasoning like its predecessor Scyther [13]. TAMARIN provides two ways to construct proofs; one is an efficient, fully automated mode and second is an interactive mode. The automated proof of tool on termination gives either the proof of correctness of the protocol or the counterexample. TAMARIN is written in the Haskell programming language. The interactive mode of TAMARIN is implemented as a web server. By using interactive mode the user can explore the proof states, look at attack graphs and manual proof guidance can be combined with automated proof search. TAMARIN has a flexible modeling framework and expressive property language which makes TAMARIN fit for analyzing a wide range of security protocols and expressive property language allows direct specification of temporal properties.

## 3.8 COQ THEOREM PROVER

Coq [16] is a proof assistant used for Calculus of Inductive Constructions which is logical framework. Coq allows the construction of formal proofs and manipulates functional programs with the specification consistently. Coq was designed to develop mathematical proofs and especially to write formal specifications of programs and check their correctness with respect to that formal specification. GALLINA is the specification language provided by the Coq. Coq uses the type-checking algorithm which checks the correctness of the proofs. Coq is having an interactive mode in which the commands are interpreted as typed by the user and then the commands are processed from a file in the compiled mode. More details about Coq are given in Coq’Art [17], a book on practical uses of the Coq system.

## 3.9 ISABELLE

Isabelle is a generic theorem prover used for interactive reasoning of formal theories [18]. Isabelle is a generic system to implement logical formalisms. Isabelle/HOL is a specialization of Isabelle for Higher-Order Logic. Isabelle is originated from the LCF (Logics for Computable Functions) [19]. Robin Milner developed LCF around 1972. Isabelle is evolved in the versions. In the version of 1986, Isabelle was able to accomplish automated proof using tactics and tactical more powerful than LCF's, but this version did not support natural deduction [18]. Isabelle provides useful proof procedures for Constructive Type Theory [20], various first order logics, Zermelo-Fraenkel set theory and higher order logic [18]. Isabelle supports logics from a broad family. They include first-order logics and higher-order logics; which are based on sets, functions and types and domains. The drawback of Isabelle is that its user interface is not so user-friendly. It's hard to use for beginners.

## 3.10 MURφ

Murφ is a general purpose state enumeration tool to analyze cryptographic and security related protocols [21]. For verifying a protocol using Murφ, first the protocol should be modeled in the Murφ language and then this model is augmented with the specification of desired properties. Then system automatically checks if all reachable states of the model satisfy the given specification using explicit state enumeration. The Murφ language is a simple high level language used to describe nondeterministic finite state machines. Most Murφ models are nondeterministic because states allow execution of more than one action. The Murφ language supports scalable models. Murφ supports automatic symmetry reduction of model. In Murφ the desired properties are specified by invariants, which are Boolean conditions. These conditions have to be true in every reachable state. Violation of the invariant in some state will result in printing the error trace which is a sequence of states from the start state to the state in which the violation occurs.

## 3.11 ATHENA

Athena [22] is a fully automatic verification technique for analysis of security protocols. It utilizes both theorem proving and model checking techniques. In Athena, an extension of the Strand Space Model (SSM) is used. Thayer, Herzog and Guttman proposed SSM [23] which was basically used for protocol representation and demonstrated the use of SSM for manually proving certain security properties. Athena uses a fundamentally different representation of protocol executions and takes the advantage of the symbolic state representation. A new logic which is suitable for SSM is developed in which there is a procedure which automatically evaluates well-formed formulae. When the evaluation terminates, either a proof is given or a counter example is generated depending on formula’s validity. The termination of validation procedure is not guaranteed but it can be terminated by limiting the concurrent protocol runs and the message length, in a way similar to the Murφ and Brutus. Athena exploits various state space reduction techniques which are helpful to reduce the state space explosion problem.

## 3.12 BRUTUS

Brutus [24] is a completely automatic special purpose model checker used to verify properties of security protocols. Brutus has an expressive specification language used to state properties of the protocol. Brutus is a lightweight and push button tool which is able to handle multiple sessions. If the algorithm for verification is not designed carefully then multiple sessions may result in the explosion in the explored state space. The algorithm used in the Brutus is a combination of both natural deduction and state space exploration techniques. Brutus supports a broad range of security protocols. If the considered protocol is having a flaw then the tool creates a counterexample or an attack considering the flaw. Brutus uses the reduction techniques which make the protocol verification efficient in both time and space. Due to the powerful deduction technique used in Brutus, one can model multiple instances for many protocols.

### 3.13 CASPER

Casper [25] is a compiler used for analysis of security protocols. The compiler is used to produce the CSP descriptions of the protocol. CSP (Communicating Sequential Processes) [26] is process algebra. Because of this compiler, producing the CSP will take much less time compared to doing the same by hands and making changes in that is also easy. When using Casper, the specification errors are less common and can be easily detected by the compiler. The user specifies the protocols using abstract notation and Casper converts this into CSP code. This CSP code is suitable for checking using the FDR. FDR [27] is a model checker. Casper does not cover all the features of security protocols but it does provide a useful framework for investigating new ideas and considering how to model new features within CSP. Using Casper for generating CSP file is clearly easier than creating that file from the scratch. Casper is able to deal with a very large class of protocols.

Table.1. Brief description of Tools

| Tool | Characteristics |
|---|---|
| Interrogator | • Developed in 1983<br>• It is finite state model checker<br>• Implemented in LM-Prolog<br>• It has GUI |
| Inatest | • Developed in 1987<br>• Formal method used is FDM<br>• Language used for specification: variant of Ina Jo |
| NRL Protocol Analyzer | • Developed in 1989<br>• It is an automated intruder<br>• Written in Prolog |
| ProVerif | • Fully automatic technique is provided<br>• To specify protocols process calculus (extension of pi calculus) is used<br>• Limitation: solving algorithm sometimes does not terminate |
| Scyther | • It has GUI<br>• Verifies bounded and unbounded number of runs<br>• Language used for specification: spdl |
| AVISPA | • It is a push-button tool<br>• Language used for specification: hlpsl<br>• Uses four back-ends |
| TAMARIN Prover | • Used backward reasoning like Scyther<br>• Written in Haskell programming language<br>• Has two modes: fully automated mode and interactive mode |
| Coq theorem prover | • Proof assistant for Calculus of Inductive Constructions<br>• Language used for specification: GALLINA<br>• Uses type-checking algorithm to check correctness of proofs<br>• Has interactive mode |
| Isabelle | • Generic theorem prover<br>• Originated from LCF [19]<br>• Limitation: User interface is hard to use for beginners |
| Murφ | • General purpose state enumeration tool<br>• Language used for specification: Murφ language<br>• Supports automatic symmetry reduction of model |
| Athena | • Efficient and fully automatic verification technique<br>• Utilizes both model checking and theorem proving techniques<br>• Extension of SSM |
| Brutus | • Completely automatic model checker<br>• Lightweight and push-button tool which can handle multiple sessions<br>• Both state space exploration and natural deduction techniques are used |
| Casper | • It is a compiler to produce CSP description of protocols<br>• Does not cover all features of security protocols<br>• Provides useful framework for investigating new ideas |

## 4. APPLICATION OF TOOLS

In this section, we are discussing in brief some of the work done using the tools mentioned above. In [8], the Interrogator tool is used to analyze one of the three protocols, referred as "Needham-Schroeder protocol". A search tool developed by Longley and Rigby [28] having the same approach as that of the Interrogator has been used to find previously unknown flaw in a hierarchical key management scheme [29]. In [30], the TMN protocol [31] is analyzed by using Interrogator, NRL protocol analyzer and Inatest to successfully demonstrate the flaws in the protocol. NRL protocol analyzer is used to analyze the Needham Schroder protocol in [32] and the Internet Key Exchange protocol in [33].

ProVerif has been used to analyze various protocols such as the Trusted Platform Module (TPM) in [34], certified email protocol [35] in [36], Just Fast Keying (JFK) [37] protocol in [38]. A secure file system Plutus [39] has been studied with ProVerif in [40]. In [41], comparison of ProVerif and AVISPA is done by applying on various protocols. N. Dalal et al. [42] have done a comparative analysis of the tools Scyther and ProVerif by using them on Kao Chow Authentication Protocol, 3-D Secure Protocol, Andrew Secure RPC Protocol, Needham-Schroeder Public Key Protocol, Diffie-Hellman Key Exchange Protocol and Challenge Handshake Authentication Protocol.

AVISPA tool has been used for specifying and analyzing a lot of protocols. All these protocols can be seen at [43].

TAMARIN has been used to analyze many authenticated key exchange protocols with respect to their intended adversary models [44]. In [45], Simon Meier et al. presented the Tamarin prover for the symbolic analysis of security protocols. For automated verification of Group Key Agreement protocols

TAMARIN prover is used in [46]. The ARPKI protocol is presented and verified by using TAMARIN prover in [47].

OPT's (Origin and Path Trace) [48] security properties have been formally verified using the Coq interactive theorem prover in [49]. Isabelle is used to give a mechanized proof of the Basic Perturbation Lemma in [50] and to verify the correctness of Warren Abstract Machine (WAM) in [51].

Murφ has been used to verify the Fair Exchange Protocol in [52], to analyze two related contract signing protocols: the optimistic contract signing protocol and the abuse-free contract signing protocol in [53], to analyze the 4-Way Handshake key management protocol in 802.11i in [54] and to analyze Needham-Schroeder protocol, variants of Kerberos and TMN protocol in [55].

About 30 protocols from [56] have been checked by using Athena [22]. In [24], as a case study Brutus has been used to analyze Needham-Schroeder public key protocol, 1KP and Wide Mouthed Frog protocol. The Wide Mouthed Frog protocol has also been analyzed by using Casper in [25]. In [57], Casper and FDR were used to analyze the protocols from the Clark and Jacob's library of security protocols [56] and in [58] FDR, a model checker for CSP, is used to detect errors in the TMN protocol.

L. Xinfeng et al. [59] proposed time dependent cryptographic protocol logic (TCPL) for analysis of cryptographic protocols. Logic formulas are to be specified in XML, and then, TCPL method to will automatically verify cryptographic protocol. In, [60] formal specification Reliable Broadcasting Protocol is attempted without analysis.

CryptoVerif [61], Cryptographic Protocol Shapes Analyzer (CPSA) [62], Maude-NRL Protocol Analyzer (Maude-NPA) [63] are the some tool for analysis. In [64], a state of art report of cryptographic protocols is used. In the report specific protocols like TLS, SSH, IPSec, Kerberos and Application Specific Protocols like WEP/WPA, MTS/LTE, Bluetooth, ZigBee, EMV's standardization Efforts and limitations are given.

## 5. DISCUSSION

This paper presents the motivation and brief history of the specification and verification of cryptographic protocols. Paper reviewed the tools for the specification and verification of cryptographic protocols and will help the researchers review the protocol analysis tools.

From the review it can stated that cryptographic protocols should be checked by latest formal method to find the loops and attacks. Latest state-of-the-art tools with artificial intelligence will be useful to find loops and attacks in cryptographic protocol.

## REFERENCES

[1] C. Meadows, "Formal Methods for Cryptographic Protocol Analysis: Emerging Issues and Trends", *IEEE Journal on Selected Areas in Communications*, Vol. 21, No. 1, pp. 44-54, 2003.

[2] R.M. Needham and M.D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers", *ACM Communications*, Vol. 21, No. 12, pp. 993-999, 1978.

[3] D. Dolev and A.C. Yao, "On the Security of Public Key Protocols", *IEEE Transactions on Information Theory*, Vol. 29, No. 2, pp. 198-207, 1983.

[4] D. Dolev, S. Even and R.M. Karp, "On the Security of Ping-Pong Protocols", *Information and Control*, Vol. 55, No. 1-3, pp. 57-68, 1982.

[5] M. Burrows, M. Abadi and R.M. Needham, "A Logic of Authentication", *Proceedings of the Royal Society A Mathematical, Physical and Engineering Science*, pp. 18-36, 1990.

[6] G. Lowe, "Breaking and Fixing the Needham-Schroeder Public-Key Protocol using FDR", *Software-Concepts and Tools*, Vol. 17, No. 3, pp. 93-102, 1996.

[7] A.W. Roscoe, "Modelling and Verifying Key-Exchange Protocols using CSP and FDR", *Proceedings of 8th IEEE Computer Security Foundations* Workshop, pp. 98-107, 1995.

[8] J.K. Millen, S.C. Clark and S.B. Freedman, "The Interrogator: Protocol Security Analysis", *IEEE Transactions on Software Engineering*, Vol. 13, No. 2, pp. 274-288, 1987.

[9] S.T. Eckmann and R.A. Kemmerer, "Inatest: An Interactive Environment for Testing Formal Specifications", *ACM SIGSOFT Software Engineering Notes*, Vol. 10, No. 4, pp. 17-18, 1985.

[10] J. Scheid and S. Holtsberg, "Ina Jo Specification Language Reference Manual", Technical Report, Paramax Systems Corporation, Unisys Company, 1992.

[11] Catherine Meadows, "The NRL Protocol Analyzer: An Overview", *The Journal of Logic Programming*, Vol. 26, No. 2, pp. 113-131, 1996.

[12] B. Blanchet, "Automatic Verification of Correspondences for Security Protocols", *Journal of Computer Security*, Vol. 17, No. 4, pp. 363-434, 2009.

[13] C.J.F. Cremers, "The Scyther Tool: Verification, Falsification, and Analysis of Security Protocols", *Proceedings of 20th International Conference Computer Aided Verification*, pp. 414-418, 2008.

[14] Luca Vigano, "Automated Security Protocol Analysis with the AVISPA Tool 1", *Electronic Notes in Theoretical Computer Science*, Vol. 155, pp. 61-86, 2006.

[15] S. Meier, B. Schmidt, C. Cremers and D.A. Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols", *Proceedings of International Conference on Computer Aided Verification*, pp. 696-701, 2013.

[16] B. Barras, S. Boutin, C. Cornes, J. Courant, J.-C. Filliatre, E. Gimenez, H. Herbelin, G. Huet, C. Munoz and C. Murthy, "The Coq Proof Assistant Reference Manual: Version 6.1", Available at: https://hal.archives-ouvertes.fr/inria-00069968/document.

[17] Y. Bertot and P. Casteran, "*Interactive Theorem Proving and Program Development*", 1st Edition, Springer, 2004.

[18] Lawrence C. Paulson, "Isabelle: The Next 700 Theorem Provers", *Logic and computer science*, Vol. 31, pp. 361-386, 1990.

[19] L.C. Paulson, "*Logic and Computation: Interactive Proof with Cambridge LCF*", Cambridge University Press, 1990.

[20] P. Martin Lef, "Intuitionistic Type Theory", *Naples: Bibliopolis*, 1984.

[21] J.C. Mitchell, M. Mitchell and U. Stern, "Automated Analysis of Cryptographic Protocols using Murphi", *Proceedings of IEEE Symposium on Security and* Privacy, pp. 141-151, 1997.
[22] D.X. Song, S. Berezin and A. Perrig, "Athena: A Novel Approach to Efficient Automatic Security Protocol Analysis", *Journal of Computer Security*, Vol. 9, No. 1-2, pp. 47-74, 2001.
[23] F.J. Thayer, J.C. Herzog and J.D. Guttman, "Strand Spaces: Why is a Security Protocol Correct?", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 160-171, 1998.
[24] E.M. Clarke, S. Jha and W.R. Marrero, "Verifying Security Protocols with Brutus", *ACM Transactions on Software Engineering and Methodology*, Vol. 9, No. 4, pp. 443-487, 2000.
[25] G. Lowe, "Casper: A Compiler for the Analysis of Security Protocols", *Journal of Computer Security*, Vol. 6, No. 1-2, pp. 53-84, 1998.
[26] C.A.R. Hoare, "*Communicating Sequential Processes*", Prentice-Hall, 1985.
[27] A. W. Roscoe, "*Model-Checking Csp*", Prentice-Hall, 1994.
[28] D. Longley and S. Rigby, "An Automatic Search for Security Flaws in Key Management Schemes", *Computers and Security*, Vol. 11, No. 1, pp. 75-89, 1992.
[29] C. Meadows, "Formal Verification of Cryptographic Protocols: A Survey", *Proceedings of 4th International Conference on the Theory and Applications of Cryptology*, pp. 135-150, 1994.
[30] R.A. Kemmerer, C. Meadows and J.K. Millen, "Three System for Cryptographic Protocol Analysis", *Journal of Cryptology*, Vol. 7, No. 2, pp. 79-130, 1994.
[31] M. Tatebayashi, N. Matsuzaki and D.B. Newman, "Key Distribution Protocol for Digital Mobile Communication Systems", *Proceedings of Conference on the Theory and Application of Cryptology*, pp. 324-334, 1990.
[32] C.A. Meadows, "Analyzing the Needham-Schroeder Public Key Protocol: A Comparison of Two Approaches", *Proceedings of European Symposium on Research in Computer Security*, pp. 351-364, 1996.
[33] C. Meadows, "Analysis of the Internet Key Exchange Protocol using the NRL Protocol Analyzer", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 216-231, 1999.
[34] S. Delaune, S. Kremer, M.D. Ryan and G. Steel, "A Formal Analysis of Authentication in the TPM", *International Workshop on Formal Aspects in Security and Trust*, pp. 111-125, 2011.
[35] M. Abadi, N. Glew, B. Horne, and B. Pinkas, "Certified Email with A Light On-Line Trusted Third Party: Design and Implementation", *Proceedings of the 11th International Conference on World Wide Web*, pp. 387-395, 2002.
[36] M. Abadi and B. Blanchet, "Computer-Assisted Verification of a Protocol for Certified Email", *Science of Computer Programming*, Vol. 58, No. 1, pp. 3-27, 2005.
[37] W. Aiello, S.M. Bellovin, M. Blaze, R. Canetti, J. Ioannidis, A.D. Keromytis and O. Reingold, "Just Fast Keying: Key Agreement in A Hostile Internet", *ACM Transactions on Information and System Security*, Vol. 7, No. 2, pp. 242-273, 2004.
[38] M. Abadi, B. Blanchet and C. Fournet, "Just Fast Keying in the Pi Calculus", *ACM Transactions on Information and System Security*, Vol. 10, No. 3, p. 1-9, 2007.
[39] M. Kallahalla, E. Riedel, R. Swaminathan, Q. Wang, and K. Fu, "Plutus: Scalable secure file sharing on untrusted storage", Available at: https://www.usenix.org/legacy/event/fast03/tech/full_papers/kallahalla/kallahalla_html/main.html
[40] B. Blanchet and A. Chaudhuri, "Automated Formal Analysis of a Protocol for Secure File Sharing on Untrusted Storage", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 417-431, 2008.
[41] Pascal Lafourcade, Vanessa Terrade and Sylvain Vigier, "Comparison of Cryptographic Verification Tools Dealing with Algebraic Properties", *Proceedings of International Workshop on Formal Aspects in Security and Trust*, pp. 173-185, 2010.
[42] Nitish Dalal, Jenny Shah, Khushboo Hisaria and Devesh Jinwala, "A Comparative Analysis of Tools for Verification of Security Protocols", *International Journal of Communications, Network and System Sciences*, Vol. 3, No. 10, pp. 779-787, 2010.
[43] The Avispa Project, Available at: http://www.avispa-project.org/, Accessed on 2015.
[44] B. Schmidt, S. Meier, C. Cremers and D. Basin, "Automated Analysis of Diffie-Hellman Protocols and Advanced Security Properties", *Proceedings of 25th IEEE Computer Security Foundations Symposium*, pp. 78-94, 2012.
[45] S. Meier, B. Schmidt, C. Cremers and D. Basin, "The TAMARIN Prover for the Symbolic Analysis of Security Protocols", *Proceedings of the 25th International Conference on Computer Aided Verification*, pp. 696-701, 2013.
[46] B. Schmidt, R. Sasse, C. Cremers and D. Basin, "Automated Verification of Group Key Agreement Protocols", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 179-194, 2014.
[47] D.A. Basin, C.J.F. Cremers, T.H. Kim, A. Perrig, R. Sasse and P. Szalachowski, "ARPKI: Attack Resilient Public-Key Infrastructure", *Proceedings of Conference on Computer and Communications Security*, pp. 382-393, 2014.
[48] T.H. Kim, C. Basescu, L. Jia, S. B. Lee, Y. Hu and A. Perrig, "Lightweight Source Authentication and Path Validation", *Proceedings of ACM SIGCOMM Conference*, pp. 271-282, 2014.
[49] F. Zhang, L. Jia, C. Basescu, T.H. Kim, Y. Hu and A. Perrig, "Mechanized Network Origin and Path Authenticity Proofs", *Proceedings of Conference on Computer and Communications Security*, pp. 346-357, 2014.
[50] J. Aransay, C. Ballarin and J. Rubio, "A Mechanized Proof of the Basic Perturbation Lemma", *Journal of Automated Reasoning*, Vol. 40, No. 4, pp. 271-292, 2008.
[51] C. Pusch, "Verification of Compiler Correctness for the WAM", *Proceedings of International Conference on Theorem Proving in Higher Order Logics*, pp. 347-361, 1996.
[52] Vitaly Shmatikov and John C. Mitchell, "Analysis of a Fair Exchange Protocol", *Proceedings of Network and Distributed System Security Symposium*, pp. 1-6, 2000.

[53] Vitaly Shmatikov and John C. Mitchell, "Finite-State Analysis of Two Contract Signing Protocols", *Theoretical Computer Science*, Vol. 283, No. 2, pp. 419-450, 2002.

[54] C. He and J.C. Mitchell, "Analysis of the 802.11i 4-way Handshake", *Proceedings of 3rd ACM Workshop on Wireless Security*, pp. 43-50, 2004.

[55] J.C. Mitchell, M. Mitchell and U. Stern, "Automated Analysis of Cryptographic Protocols using Mur/splphi/", *Proceedings of IEEE Symposium on Security and Privacy*, pp. 141-151, 1997.

[56] J.A. Clark and J.L. Jacob, "A Survey of Authentication Protocol Literature: Version 1.0", Available at: http://e-centre.mdx.ac.uk/staffpages/m_cheng/link/clarkjacob.pdf.

[57] B. Donovan, P. Norris and G. Lowe, "Analyzing a Library of Security Protocols using Casper and FDR", *Proceedings of Workshop on Formal Methods and Security Protocols*, pp. 36-43, 1999.

[58] G. Lowe and B. Roscoe, "Using CSP to Detect Errors in the TMN Protocol", *IEEE Transactions on Software Engineering*, Vol. 23, No. 10, pp. 659-669, 1997.

[59] L. Xinfeng, L. Xinghua, L. Jun and X. Junmo, "Specification and Verification of Cryptographic Protocols based on TCPL", *Proceedings of 4th International Conference on Computer Science and Education*, pp. 1216-1220, 2009.

[60] G. Denker, J.J. Garcia-Luna-Aveces, J. Meseguer, P.C. Olveczky, Y. Raju, B. Smith and C. Talcott, "Specification and Analysis of a Reliable Broadcasting Protocol in Maude", *Proceedings of 37th Annual Allerton Conference on Communication, Control, and Computation*, pp. 21-26, 1999.

[61] Bruno Blanchet, "CryptoVerif: Cryptographic Protocol Verifier in the Computational Model", Available at: http://prosecco.gforge.inria.fr/personal/bblanche/cryptoverif

[62] The Cpsa Package, Available at: https://hackage.haskell.org/package/cpsa.

[63] KISS (Knowledge In Security protocolS), Available at: http://www.lsv.ens-cachan.fr/~ciobaca/kiss

[64] Study on Cryptographic Protocols, Available at: https://www.enisa.europa.eu/publications/study-on-cryptographic-protocols.