A Case Study on Automation of Verification Logics

Reiner Dojen
Data Communication Security Laboratory
University of Limerick
Limerick
Ireland
reiner.dojen@ul.ie

Tom Coffey
Data Communication Security Laboratory
University of Limerick
Limerick
Ireland
tom.coffey@ul.ie

***Abstract* – Formal logics are increasingly used to verify correctness of theories and designs in engineering. However, the process of logic-based verification is complex, tedious and prone to error. This is a serious issue, as a single mistake can render the result of the verification useless. Automated techniques reduce the potential for human errors during verification. This paper presents the theoretical concept of Layered Proving Trees, for the automation of logic-based verification. Verification tools based on Layered Proving Trees result in comparatively simple - but powerful – systems. Empirical result on the performance of a prototype implementation of a layered proving trees verification tool are presented.**

I INTRODUCTION

Formal logics have played an important role in many research fields. In recent years they are increasingly used in engineering to verify the correctness of theories and designs, for example in:

- specification and verification of imperative programming languages.
- distributed computing, concurrency and process control.
- circuit design and VLSI.
- verification of protocols.

In general, logic-based verification follows the structure as outlined in Fig. 1.

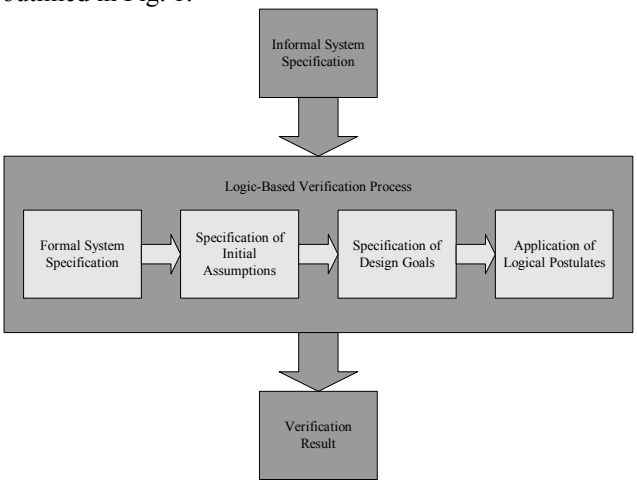

Figure 1: Logic-Based Verification

While logics for verifications purposes ("verification logics") are powerful techniques for establishing that a design meets its initial specification, the manual application of verification logics often requires in-depth expertise. Further, the process is complex, tedious and prone to error. This is a serious issue, as a single mistake during any stage of the verification process can render the result of the verification useless. Automated techniques reduce the potential for human errors during verification, while simultaneously removing the need of in-depth knowledge of the employed verification logic.

A Automation of Verification Logics

The application of the postulates of a verification logic is tedious and error prone. Hence, the use of software to automatically apply the axioms of the logic has the potential to offer significant benefits to the system designer [1]. Several possible sources of error are automatically removed, as an automated system will:

- not make implicit assumptions,
- not take shortcuts,
- ensure thorough and unambiguous use of the postulates,
- not make implicit assumptions about failed goals,
- allow redundant assumptions to be identified easily.

Also, the effort involved in protocol verification can be considerably reduced, since familiarity with the axioms is no longer required. The time taken to perform the verification is greatly reduced as software can automatically verify a system in minutes while a similar manual proof often requires hours or days.

Most current attempts to automate verification logics make use of generic theorem provers, such as HOL [2], Coq [3], Isabelle [4] and PVS [5]. This involves translating the used verification logic into the language of the theorem prover and providing some means to specify the protocol to be verified. The built-in reasoner of the theorem prover will attempt to establish whether the verification goals are a logical consequence of the initial assumptions and the system specification in the used verification logic. However, these tools often require large resources to operate, mainly due to their ability to prove generic theorems. Further, such theorem provers will usually only return results of the performed proofs, but will not provide any details of how this is achieved

On the other hand, it is realised that logic-based verification is a restricted theorem-proving problem. Therefore a specialised theorem prover could result in a more efficient automated proving system.

This paper presents the theoretical concept of Layered Proving Trees, for the automation of logic-based

verification. Verification tools based on Layered Proving Trees result in comparatively simple - but powerful – systems. Empirical result on the performance of a prototype implementation of a layered proving trees verification tool are presented.

II THE CONCEPT OF LAYERED PROVING TREES

Logic-based verification is a specialised case of theorem proving with several restrictions. *Layered Proving Trees* are now introduced as a novel approach to the automation of the application of logical postulates for logic-based protocol verification. The use of Layered Proving Trees results in a comparatively simple, yet powerful, system for the automation of verification logics. Layered Proving Trees offer a number of advantages over other automation approaches:

- Simplicity – the Layered Proving Tree technique is quite simple, therefore reducing the number of potential error-sources. The main requirement of the technique is the ability to unify terms. This is essentially a pattern-matching problem, which is easily automated.
- Traceability – as the entire proof is stored in the Layered Proving Tree, all "decisions" made can be easily traced after completion of the proof. This results in superior feedback to the verifier, particularly in case of a failed verification. Further, this feedback can assist in studying the application of verification logics.
- Exhaustiveness – Many verification logics contain redundant postulates. Hence, multiple ways often exists to prove the same goal. The Layered Proving Tree technique can find all possible proving traces.
- Accuracy of Modelling - the Layered Proving Tree technique is able to accurately model verification logics.

A Properties of Verification Logics

Many verification logics are restricted forms of predicate logic with the following properties:

1. The domain is finite: the number of constants and variables is restricted.
2. Rather than using generic natural deduction rules, a specialized set of inference rules is provided.
3. All the provided postulates follow the structure $\pi_1 \wedge \pi_2 \wedge ... \wedge \pi_n \Rightarrow \gamma$: from the conjunct validity of the pre-requisites π_1 to π_n the validity of the conclusion γ can be inferred.

These restrictive features of verification logics facilitate automation. The main task in the proving process is the unification of statements with conclusions of the postulates.

B Layered Proving Tree Construction

The concept of Layered Proving Trees and its application to the automation of security protocol verification has been introduced by Dojen and Coffey in [6]. The basic idea is to create a tree representing a verification of a system in the logic *L*. Every node in this tree contains a statement, corresponding to a goal (or sub-goal) of manual verification. Nodes in the tree are expanded by application of some postulate of the verification logic *L*. Further, each node has an associated truth-value, which indicates whether the statement of the node has been proven or not.

Layered proving trees can be operated in two modes: In mode 1, only the question if there exists any successful verification is of interest. Mode 2 allows an exhaustive search for all possible ways of verifying the system in question. This is done by terminating extension of nodes within the tree only when no further *L* postulates can be applied to any leaf. Examination of the resulting layered tree will reveal all possible verifications in *L*. Such a tree is called an exhaustive layered proving tree.

Links between nodes are associated with the logical connectives AND (AND-link) or OR (OR-link). The truth-value of a node can only change through truth-values of children becoming true. If a node is connected to its children with an AND-link, it only becomes true if all children have truth-value true. On the other hand, if a node is connected to its children with an OR-link, it becomes true if any of the children has a truth-value true. Eventually, a state is reached where either the root of the tree becomes true or no *L* postulate can be applied to any leaf-node of the tree. In the former case the verification is successful, in the latter the verification failed.

It is worth noting, that all links in the same level are associated with the same connective. Further, a level with AND-links is always followed immediately by a level with OR-links and vice versa. Hence, the tree can be considered to consists of alternating layers, which are associated with AND or OR connectives. Figure 2 depicts this structure of a layered proving tree. As it can be seen in this figure, the root node contains the statement "System is Verified" with default truth-value false. The nodes in layer 1 (the immediate children of the root) model the goals of the system and are connected to the root via AND-links. These first two layers form the initial layered proving tree. This initial tree is created directly from the formal system specification, which is inputted to the system. All nodes in any layer beyond layer 1 are created by the automated system. This is done by alternating expansion steps and truth-value propagation steps.

Expansion steps extend the layered proving tree by matching the statements in the nodes against conclusions of the logical postulates. All matching postulates are added to the tree, thus expanding the layered proving tree. This expansion models the application of logical postulates in a backward proof and inserts the pre-requisites of the used postulate(s) as new sub-goals that need to be proven in the further verification process. Each expansion step is followed by a truth-value propagation step. Truth-value propagation steps ensure that the truth-value of any node reflects the current state of the verification process.

Continuing in this way, eventually either of two possible states will be reached: either the root evaluates to true or no node can be further expanded. In the former case the verification is successful and the system can be considered correct within the scope of the used verification logic and

with respect to the initial assumptions and stated goals. In the latter case, the verification failed. Examination of the layered proving tree reveals potential problems with the system or identifies missing assumptions. The identified problems should be addressed and the verification should be repeated. Re-design and verification are performed iteratively until a verifiably correct system design is reached

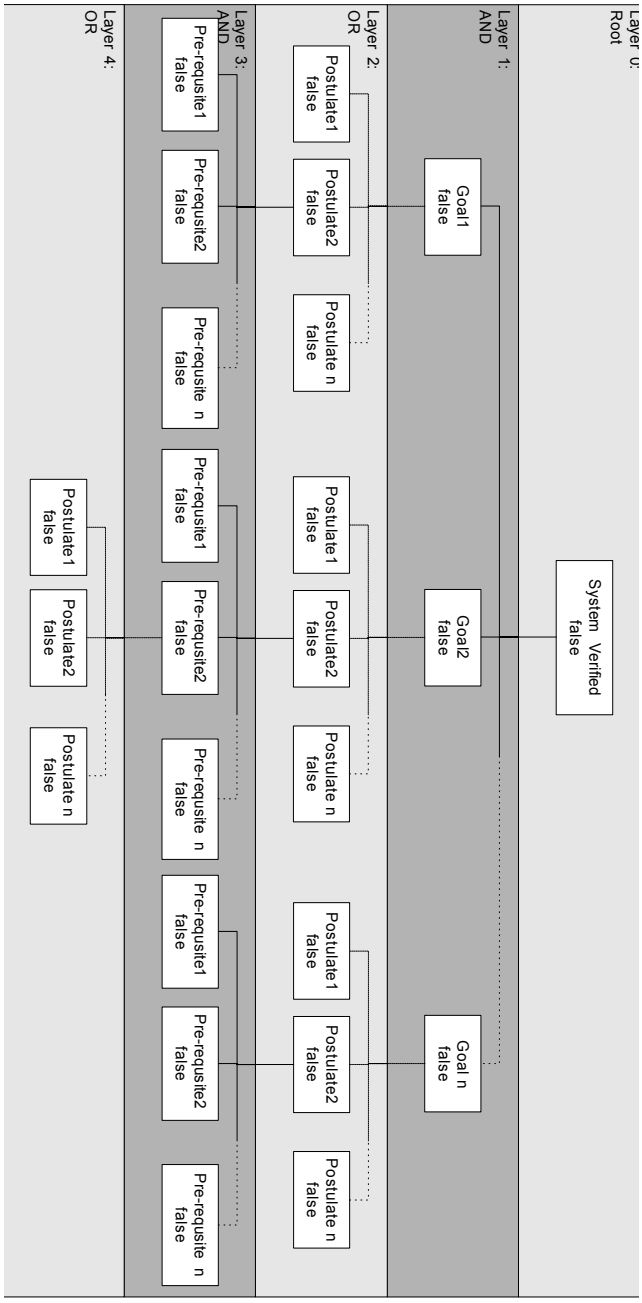

Figure 2: Structure of Layered Proving Tree

Correctness of the layered proving tree approach can be established by proving the correctness and completeness of the technique with respect to manual verification. However, due to space limitations, these proofs are not included here.

III IMPLEMENTING LAYERED PROVING TREES

When Layered Proving Trees are implemented for an individual verification logic, the following issues need to be addressed:

a) Grammar and unification of terms: A grammar must be developed to implement the statements that model the system, the initial assumptions and the design goals. Procedures must be provided to unify statements in this grammar.
b) Structure of postulates: Most verification logics are developed for manual application. Hence, the structure of the postulates is often optimised towards manual application. The structure of the postulates needs to be adopted for automated application. However, it is important to ensure that the implemented logic is equivalent to the original logic.
c) Termination: Verification logics sometimes include postulates that can cause infinite loops in a Layered Proving Tree. In this case, special rules must be included in the application of Layered Proving Trees to prevent non-termination of the tree construction.

The authors have implemented a Layered Proving Tree based on the logic by Gong, Needham and Schröder [7], which has been developed for security protocol verification. The resulting verification tool takes a formal protocol specification as input. The formal protocol specification includes the protocol steps, the initial assumptions and the goals of the protocol. Figure 3 outlines the structure of the tool.

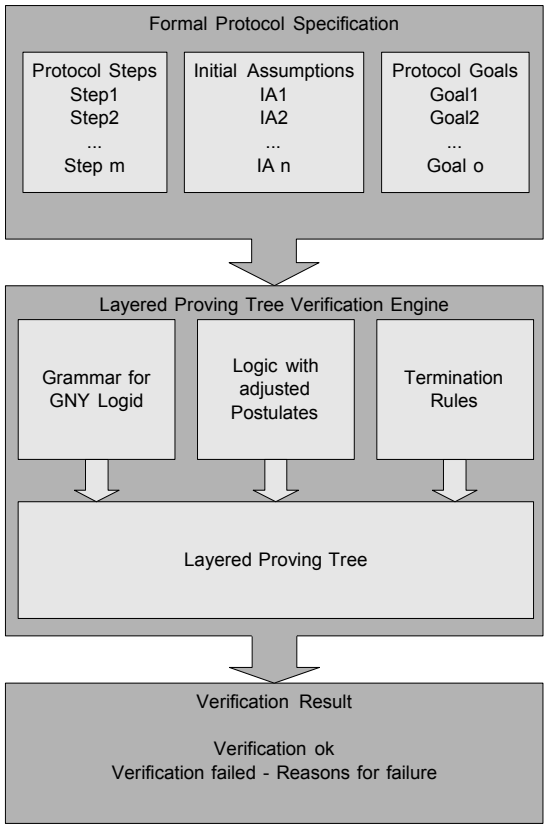

Figure 3: Structure of Layered Proving Tree Tool

IV EMPIRICAL RESULTS OF LAYERED PROVING TREE IMPLEMENTATION

This section presents experimental results obtained from the implementation of the Layered Proving Tree technique outlined above. The resulting prototype has been applied to a range of security protocols, such as the Needham-Schröder symmetric key and public key protocols [8], the corrected Needham-Schröder symmetric key [9] the BCY protocol [10], the CDF-BCY protocol [11], the ASK protocol [12], the CDF-ASK protocol [13] and the ASPeCT protocol [14]. The outcome of the automated verification matched those of manual verification. These results, in conjunction with the performed proofs in this paper, provide strong confidence in the correctness and effectiveness of the Layered Proving Tree approach.

A Verification of the ASK Protocol using Layered Proving Trees

The ASK protocol [12] was among the first to use elliptic curve encryption to achieve "Authentication and Key Agreement". This technique is especially appropriate for use in wireless communications, as the computational burden on the different communicating parties is not symmetric. Hence, the computational demands placed on the mobile device can be kept low. A further advantage of elliptic curve encryption is its comparably small key-size. This helps to reduce the bandwidth requirement compared to protocols based on conventional public key crypto-systems.

The ASK protocol contains a number of well-known flaws [13][15]. Figure 4 shows the results of a Layered Proving Tree verification of the ASK protocol. These can be browsed to reveal the verification in detail. Thus, all decisions made can be easily traced and viewed after completion of the verification. These results match those identified by manual verification [13].

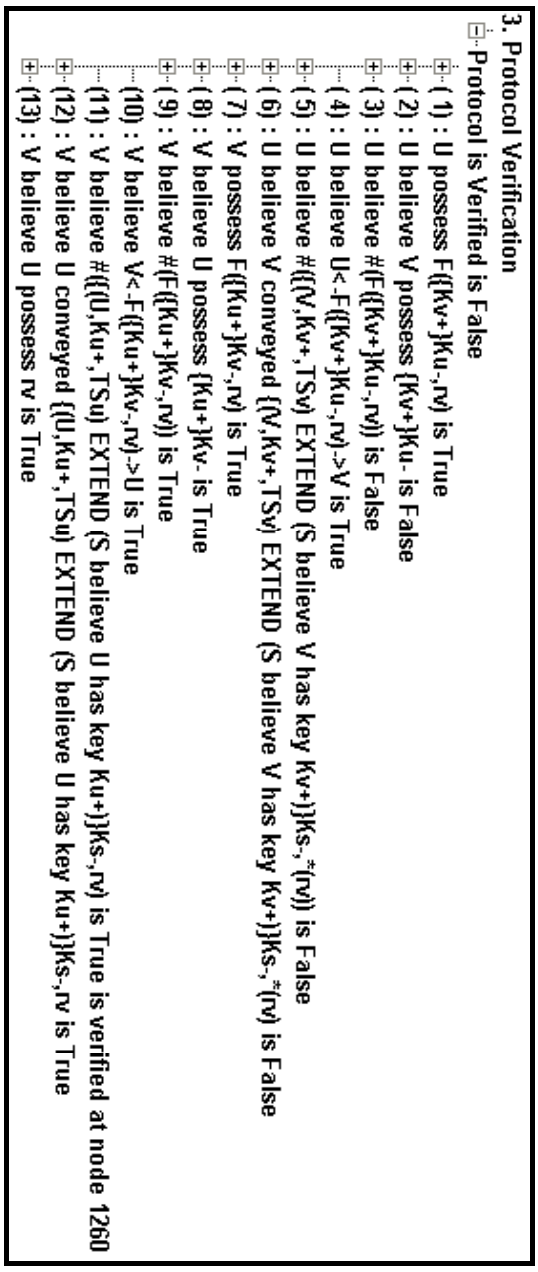

Figure 4: ASK Verification using a Layered Proving Tree

Coffey, Dojen and Flanagan presented a verifiably correct protocol called CDF-ASK [13], which is based on the ASK protocol. Figure 5 presents the results of the Layered Proving Tree verification of the CDF-BCY protocol. These results also match the manual verification.

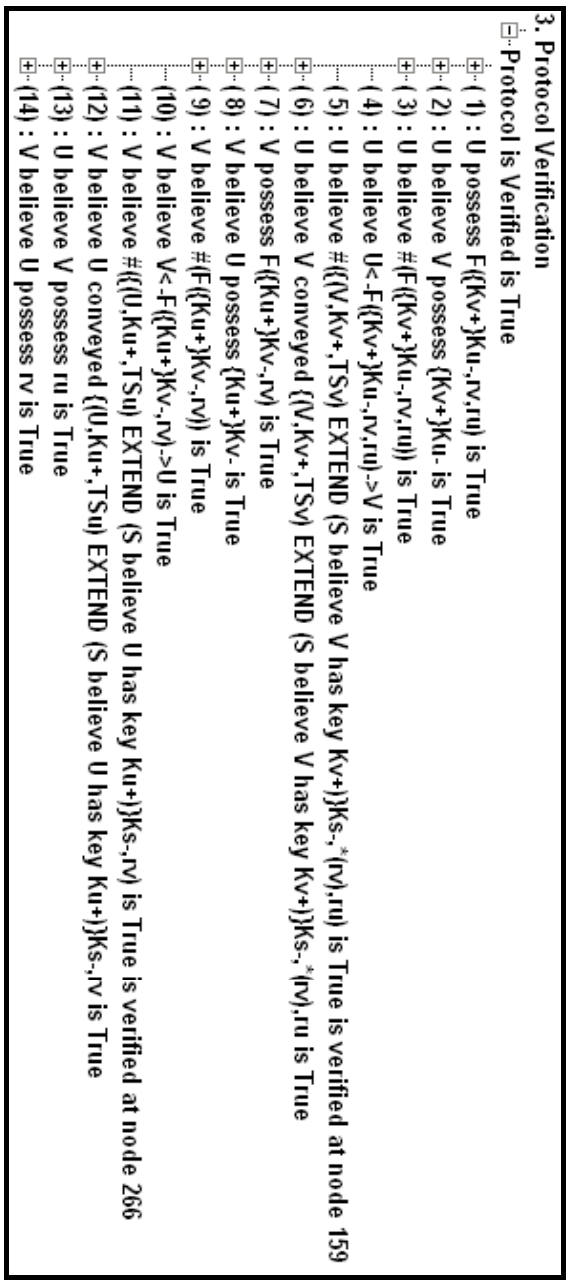

Figure 5: CDF-ASK Verification using a Layered Proving Tree

Figure 6 illustrates the traceability of the proof by showing an expanded view of part of the CDF-BCY protocol verification.

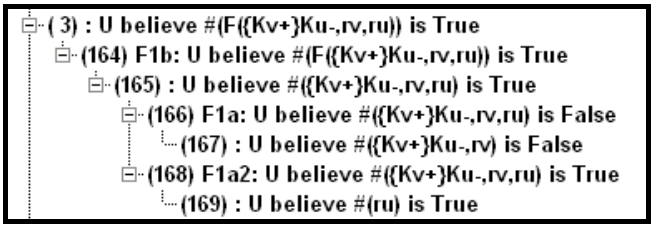

Figure 6: Expanded View of partial CDF-ASK Automated Verification

B Performance of the Layered Proving Tree TEchnique

The Layered Proving Tree technique forms a powerful and efficient way to automate logic-based verification of security protocols. A summary of the scalability and performance of the prototype implementation are presented in Table I. Theses results were obtained using a PC with a 1GHz Celeron processor and with 256MB RAM running Windows XP.

Among the performed verifications, the attempted verification of the BCY protocol (which fails due to weaknesses in the protocol) has the largest memory requirements (at 32MB) and takes approximately 3.8s to run. These results clearly identify the low requirements on memory and computational power of the Layered Proving Tree technique. Further, it shows that there is no (obvious) correlation between the number of steps in the protocol and the size (memory/nodes) of the corresponding Layered Proving Tree verification.

Table I: Performance Overview

Protocol	Step	Mem	Nodes	Goal	Result	Time
NS Conv. Key	5	10MB	330	8	Fail	0.5s
NS Pub. Key	7	6MB	134	8	Fail	0.2s
Corrected NS Conv. Key	7	9MB	266	8	Pass	0.4s
BCY	4	32MB	1395	12	Fail	3.8s
CDFBCY	4	22MB	851	12	Pass	2.7s
ASK	4	30MB	1290	13	Fail	2.8s
CDFASK	4	9MB	303	14	Pass	0.4s
ASPeCT	3	26MB	860	18	Fail	2.4s

V CONCLUSION

This paper discussed automation of logic-based verification. Layered Proving Trees were introduced as a novel approach to the automation of verification logics. The main advantages of Layered Proving Trees are their simplicity, traceability of verification results, ability to accurately model verification logics and the capability to perform an exhaustive search for different proof-traces.

The issues arising when implementing Layered Proving Trees for an individual verification logic are outlined. A prototype, based on the GNY logic for security protocol verification, was implemented and tested against a range of security protocols, some of which contain well-known flaws. The automated verifications obtained from the prototype implementation corresponded to the manual verifications, including detection of all known flaws in the analysed protocols. A summary of the scalability and performance issues of the prototype implementation was presented. These results clearly identified the low requirements on memory and computational power of the Layered Proving Tree technique.

In conclusion, the presented proofs and empirical results provide strong confidence in the correctness and effectiveness of the Layered Proving Tree approach to model the process of logic-based verification.

VI. ACKNOWLEDGEMENTS

This work was funded by the Irish Research Council for Science, Engineering and Technology (IRCSET) - Basic Research Award SC02/237.

VII REFERENCES:

[1] Coffey, T., Dojen, R. and Flanagan, T., "On the Automated Implementation of Modal Logics used to Verify Security Protocols", in *Proceedings of ISICT'03 (Invited Workshop on Network Security and Management at the International Symposium on Information and Communication Technologies),* Dublin, Ireland, 24-26 September 2003, pp.324-347

[2] Gordon, M. and Melham, T.F., *Introduction to HOL: A theorem proving environment for Higher Order Logic*, Cambridge University Press, 1993

[3] Gerard Huet, Amy Felty, B. Werner, H Herbelin, and Gilles Dowek, "Presenting the system Coq, version 5.6", in *Proceedings of the Second Workshop on Logical Frameworks,* 1991

[4] Paulson, L.C., *Isabelle: The next 700 Therorem Provers,* Logic and Computer Science, Piergiorgi Odifreddi (ed.), Academic Press, pp.361-386, 1991.

[5] S. Owre, S. Rajan, J.M. Rushby, N. Shankar, and M.K. Srivas "PVS: Combining specification, proof checking, and model checking". *Computer-Aided Verification, CAV '96*, Volume 1102 of Lecture Notes in Computer Science, pages 411-414

[6] Dojen, R. and Coffey, T., "A Novel Approach to the Automation of Logic-Based Security Protocol Verification", *WSEAS Transactions on Information Science and Applications*, Issue 5, Volume 1, November 2004, pp. 1243-1247

[7] Li Gong, Roger Needham, and Raphael Yahalom, "Reasoning about belief in cryptographic protocols," in *Proceedings of IEEE Computer Security Symp. on Security and Privacy,* 1990, pages 234-248,

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

[9] Needham, R.M. and Schroeder, M.D., "Authentication Revisited." *ACM Operating System Reviews, Vol.21, No.1*, 1987, p. 7

[10] M.J. Beller, L.-F. Chang and Y. Yacobi, "Privacy and authentication on a portable communications system", *IEEE Journal on Selected Areas in Communications*, vol. 11, No. 6, 1993, pp. 821-829.

[11] Coffey, T., Dojen, R. and Flanagan, T., "Formal Verification: An Imperative Step in the Design of Security Protocols", *Computer Networks Journal*, Elsevier Science, Vol. 43, No. 5, 2003, pp.601-618

[12] Aydos, M., Sunar, B., and Koc, C. K., "An Elliptic Curve Cryptography based Authentication and Key Agreement Protocol for Wireless Communication." *In: Proceedings of DIAL M 98,* 1998

[13] Coffey, T., Dojen, R. and Flanagan, T., "On Different Approaches to Establish the Security of Cryptographic Protocols", *Proceedings of SAM'03 (Conference on Security and Management), Vol. II,* 2003, Pages 637-643

[14] Horn, G. and Preneel, H., "Authentication and Payment in Future Mobile Systems", *Journal of Computer Security*, 2000, pp. 183-207

[15] G. Horn, K. Martin, and C. Mitchell, "Authentication Protocols for Mobile Network Environment Value-added Services", *IEEE Transactions on Vehicular Technology*, Vol. 51, 2002