A Novel Approach to the Automation of Logic-Based Security Protocol Verification

REINER DOJEN and TOM COFFEY
Data Communications Security Laboratory
Department of Electronic and Computer Engineering
University of Limerick,
IRELAND

Abstract:- Secure communications over insecure networks relies on the security of cryptographic protocols. Formal verification is an essential step in the design of security protocols. In particular logic-based verification has been shown to be effective and has discovered a number of protocol flaws. However, manual application of the deductive reasoning process is complex, tedious and prone to error. This paper introduces a novel approach to the automation of such a deductive reasoning process. This new approach results in a comparatively simple – but powerful – proving system for logic based security protocol verification.

Key-Words:- Automated Security Protocol Verification, Cryptographic Protocols

1. Introduction

Security protocols are used to achieve secure communications over insecure networks by employing cryptographic ciphers. The security of such ciphers is an essential pre-condition for protocol security. However, a discussion on their cryptographic strength is outside the scope of this paper, but can be found in papers such as [1].

The design of cryptographic protocols is a complex and error-prone process. This is particularly evident from the surprisingly large number of published protocols which have later been found to contain various flaws [2], [3], [4]. Formal verification is an imperative step in the design of security protocols [5]. Existing formal methods include state machine approaches, algebraic term rewriting, theorem proving techniques and modal logics [6]. Formal verification using logics has been shown to be effective and has discovered a number of flaws in protocols previously considered secure. Such logic-based techniques involve a process of deductive reasoning, where the desired protocol goals are deduced by applying a set of axioms and inference rules to the assumptions and message exchanges of the protocols. However, manual application of the deductive reasoning is complex, tedious and prone to error. Further, a single mistake during this process can render the result of the verification useless. Automated techniques reduce the potential for human errors during verification [7].

This paper introduces layered proving trees as a novel technique for the automation of logic-based verification of cryptographic protocols. While most current attempts to automate the application of logics are based on general theorem proving techniques, it is realised that logic based verification is a highly restricted theorem proving problem. The presented layered proving tree technique takes these restrictions into account, resulting in a comparatively simple - but powerful - proving system.

2. Logic-Based Verification of Security Protocols

Logic-based formal verification involves the following steps:

1. Formalization of the protocol messages
2. Specification of the initial assumptions
3. Specification of the protocol goals
4. Application of the logical postulates

Formalization of the protocol messages involves specifying the protocol in the language of the logic by expressing each protocol message as a logical formula. The initial assumptions state the beliefs and possessions of protocol principals at the beginning of a protocol run and the protocol goals formalise the desired beliefs and possessions of principals after a successful protocol run. The objective of the logical analysis is to verify whether the protocol goals can be derived from the initial assumptions and the formalised protocol by applying the logical postulates. If so, the protocol is successfully verified; otherwise, the verification fails.

2.1 Automated Logic-Based Verification

As outlined above 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 protocol designer. 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 protocol verification is greatly reduced as software can automatically verify a protocol in minutes while a similar manual proof often requires hours or days.

Current automated verification tools, such as TAPS [8], AAPA2 [9] and PIL/SETHEO [10], are based on general theorem provers. Others, such as SPEAR II [11] and Hauser-Lee [12], are based on declarative programming languages.

While verification tools based on such systems offer some advantages to the protocol verifier, they often suffer the following common limitations:

- Lack of Feedback: The presentation of the verification results is often highly cryptic and hard to decode. This results in poor support for identifying reasons for failed verifications.
- Inability to Trace Verification: The performed verification is not accessible after completion of the verification. Only the verification results can be reviewed, but not the decisions that make up the verification.
- Use of improper Logics: Verification logics are usually geared towards manual application. Many approaches to the automation of their application modify the logic to facilitate this automation. Commonly, no analysis of the correctness of the modified logic is presented.

The introduced technique of layered proving trees will allow the creation of verification tools that do not suffer these limitations. Figure 1 outlines the structure of such a verification tool. The tool takes a formal protocol specification – consisting of the formalised protocol steps, the initial assumptions and the protocol goals – and a formalised logic as input and, after performing the verification automatically, outputs the verification result. The main operation in the automated verification is the matching of goals/sub-goals against the conclusions of the logical postulates.

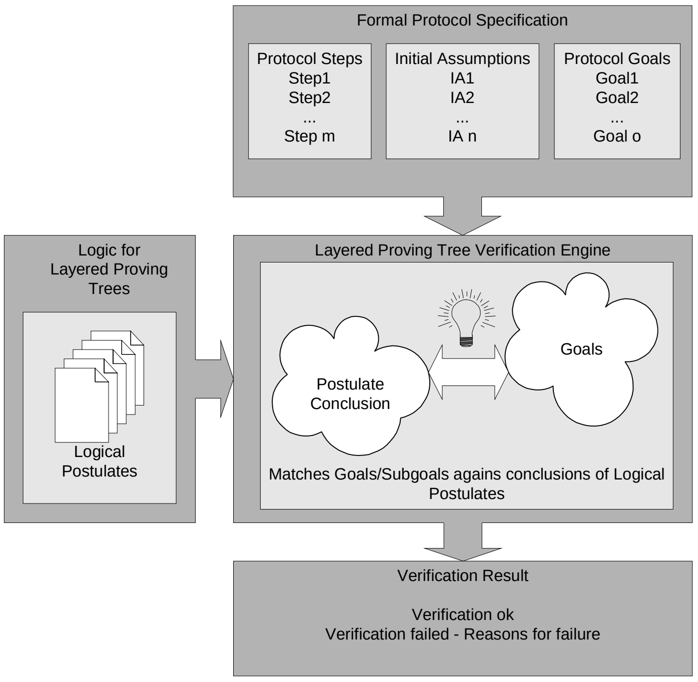
Formal Protocol Specification
Protocol Steps
Step1
Step2
...
Step m
Initial Assumptions
IA1
IA2
...
IA n
Protocol Goals
Goal1
Goal2
...
Goal o
Logic for Layered Proving Trees
Logical Postulates
Layered Proving Tree Verification Engine
Postulate Conclusion
Goals
Matches Goals/Subgoals agains conclusions of Logical Postulates
Verification Result
Verification ok
Verification failed - Reasons for failure

Figure 1: Structure of Automated Verification Tool

3. A Novel Approach to the Automation of Logic-Based Security Protocol Verification

The basic idea of layered proving trees is to create a tree representing a *L*-based verification of a security protocol. 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 of the protocol is of interest. Mode 2 allows an exhaustive search for all possible ways of verifying the protocol 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 “Protocol is Verified” with default truth-value false. The nodes in layer 1 (the immediate children of the root) model the goals of the protocol 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 protocol 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. Truth-value propagation steps ensure that the truth-value of any node reflects the current state of the verification process.

In detail, the expansion of the layered proving tree is done as follows: Select any leaf-node in the tree that has truth-value false and call it l. Find all matching postulates of the used verification logic. For each of these matching postulates add a child-node l_m to l, connected with an OR-link. This models the possibility of proving a goal through alternative postulates. Each of the l_m is expanded by nodes l_p, corresponding to the pre-requisites of the associated postulate. These l_p are connected to the l_m via AND-links, thus modeling the requirement of pre-requisites being proven in order to prove the conclusion of postulates. This finalises a single expansion step. Each expansion step is followed by a truth-value propagation step.

In a truth-value propagation step the truth-value of nodes is re-evaluated by examining the truth-value of child-nodes. If all children connected to a node with AND-links have become true, the parent-node will also become true. If children are connected to a parent-node with OR-links, a single child being true will evaluate the parent node also to true. Further, for all leave-nodes it will be checked whether the associated statement has been proven or not. In the former case, the node will also be marked true.

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

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.

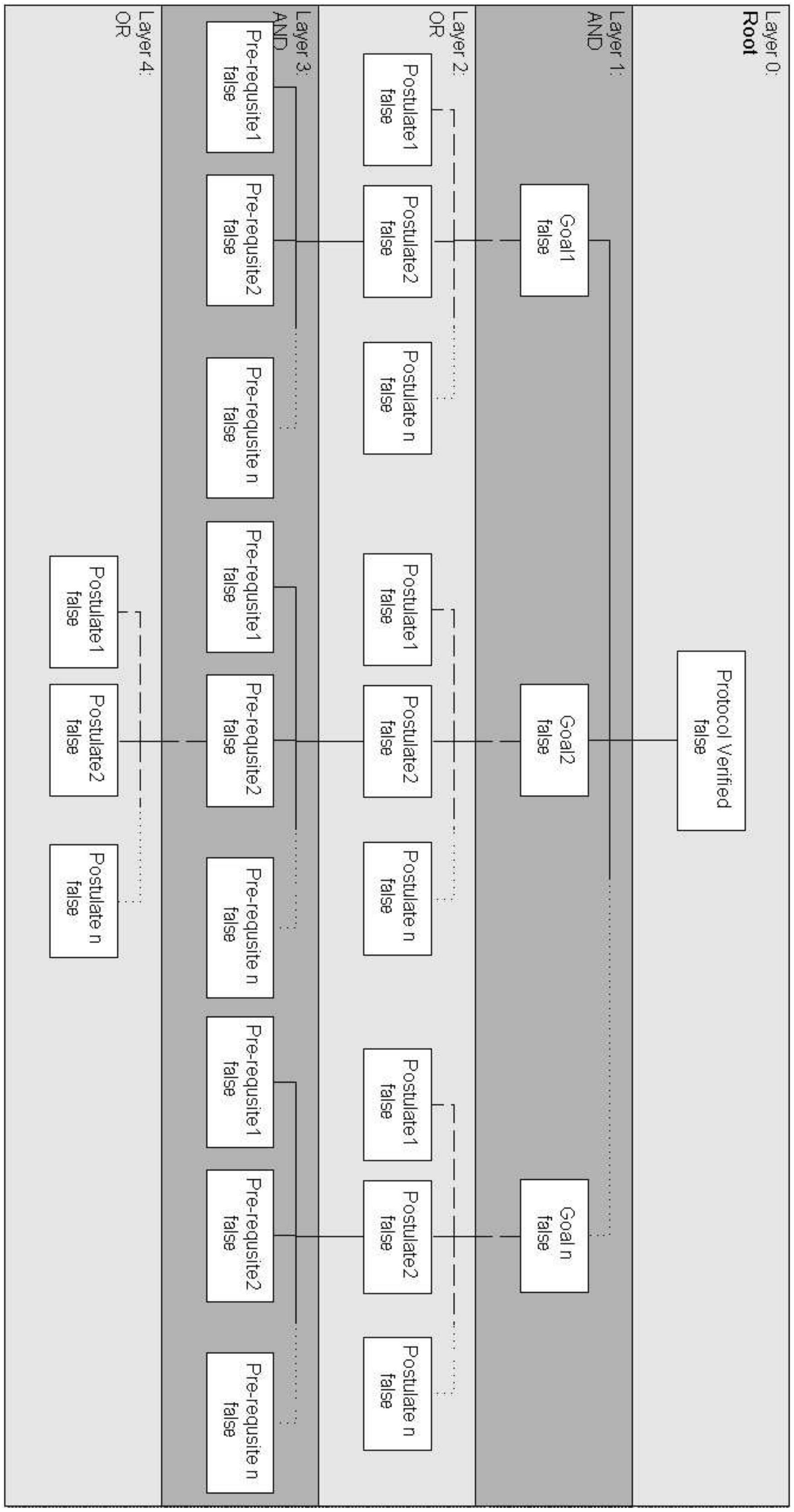

Figure 2: Layered Proving Tree Structure

4. Conclusions

Cryptographic protocols provide services for secure communications over insecure networks. However, experience has shown that the design of such protocols is highly complex. Often, subtle flaws in a protocol lead to weaknesses that can be exploited by an attacker. Formal verification provides a means to detect such flaws in a systematic and thorough way. Formal verification using logics has been shown to be effective and has discovered a number of flaws in protocols previously considered secure.

On the other hand, the manual process of logic-based verification is error-prone itself. Automation promises to reduce the number of potential error-sources, as it will ensure correct application of the verification logic. However, existing automated verification tools often suffer from a lack of feedback on the performed verification. Further, they usually do not allow the user to trace the verification path after completion. Also they often require modifications to established verification logics to facilitate automation.

This paper introduced the technique of layered proving trees, which can be used to implement verification tools for cryptographic protocols that do not suffer from the limitations above. The basic idea of a layered proving tree is to create a tree representing a *L*-based verification of a security protocol. 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*. Eventually, either the verification will be successful, or no postulate of the verification logic *L* can be further applied.

In conclusion, the layered proving tree approach provided a simple, but powerful, proving system for automated logic-based verification of cryptographic protocols.

References:

[1] Dojen, R. and Coffey, T., "Applying Conditional Linear Cryptanalysis to Ciphers With Key-Dependant operations", *WSEAS Transactions on Computers*, Vol. 5, No. 3, 2004, pp.1425-1430

[2] Burrows, M., Abadi, M. and Needham, R., "A Logic of Authentication," *ACM Operating Systems Review*, Vol. 23, No. 5, 1989, pp.1-13.

[3] Gong, L., Needham, R., and Yahalom, R., "Reasoning About Belief in Cryptographic Protocols," *Proceedings of the IEEE Computer Security Synopsis on Research in Security and Privacy*, Oakland, USA, May 1990, pp.234-248

[4] Coffey, T. and Saidha, P., "A Logic for Verifying Public Key Cryptographic Protocols", *IEE Journal Proceedings-Computers and Digital Techniques*, Vol. 144, No. 1, 1997, pp.28-32.

[5] 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

[6] Coffey, T., Dojen, R. and Flanagan, T., "On the Formal Verification of Cryptographic Protocols", *Proceedings of CSCC'03 (WSEAS International Conference on Circuits, Systems, Communciations and Computers),* Corfu, Greece, 7-10 July 2003

[7] Coffey, T., Dojen, R. and Flanagan, T., "On the Automated Implementation of Modal Logics used to Verify Security Protocols", *Proceedings of International Symposium on Information and Communication Technologies (Invited Workshop on Network Security and Managemen),* Dublin, Ireland, September 2003, pp.324-347

[8] Cohen, E., "First-Order Verification of Cryptographic Protocols", *Journal of Computer Security*, Vol. 11, No. 2, 2003, pp.189-216

[9] Brackin, S., "Automatically Detecting Most Vulnerabilities in Cryptographic Protocols," *Proceedings of DARPA Information Survivability Conference and Exposition*, Hilton Head, USA, January 2000, pp.222-236

[10] Schumann, J., "PIL/SETHEO: A Tool for the Automatic Analysis of Authentication Protocols," *Proceedings of International Conference on Computer Aided Verification*, Trento, Italy, July 1999, pp.500-504

[11] Saul, E. and Hutchison, A., "SPEARII: The Security Protocol Engineering and Analysis Resource", *Proceedings of South African Tele-communications, Networks and Applications Conference*, Durban, South Africa, September 1999, pp.171-177

[12] Hauser, R.C. and Lee, E.S., "Verification and Modelling of Authentication Protocols", *Proceedings of Second European Symposium on Research in Computer Security*, Toulouse, France, November 1992, pp.141-154