USING GYPSIE, GYNGER AND VISUAL GNY TO ANALYZE CRYPTOGRAPHIC PROTOCOLS IN SPEAR II

Elton Saul
Data Network Architectures Laboratory
University of Cape Town, South Africa
esaul(@cs.uct.ac.za

Andrew Hutchison
Data Network Architectures Laboratory
University of Cape Town, South Africa
hutch@cs.uct.ac.za

Abstract The development of cryptographic logics to analyze security protocols has provided one technique for ensuring the correctness of these protocols. However, it is commonly acknowledged that analysis using a modal logic such as GNY tends to be inaccessible and obscure for the uninitiated. In this paper we describe the SPEAR II graphically-based security protocol engineering environment that can be used to easily conduct GNY analyses. SPEAR II consists of three primary components: a protocol specification environment (GYPSIE), a GNY statement construction interface (Visual GNY) and a Prolog-based GNY analysis engine (GYNGER). In contrast to other tools, SPEAR II offers a multi-dimensional approach to protocol engineering, integrating protocol design and analysis into one consistent and unified interface. The interface and techniques used within this tool are built on the foundation of previous experience with the original SPEAR tool and GNY analysis research. We also show how the SPEAR II tool is used to conduct a GNY analysis and how it distances protocol engineers from any associated syntactical issues, allowing them to focus more on the associated semantics and distil the critical issues that arise. By freeing individuals to focus on an analysis, instead of hampering them with the necessary syntax, we can ensure that the fundamental concepts and advantages related to GNY analysis are kept in mind and applied as well.

Keywords: Security Protocol Modelling, Protocol Engineering, GNY Logic

1. INTRODUCTION

Analysis methods for cryptographic protocols have predominantly focused on detecting information leakage, rather than determining whether a protocol attains its stated goals. However, security protocols often fall short of achieving their intended objectives, usually for very subtle reasons [2]. As a result of this fact, cryptographic logics have been developed to aid in determining whether protocols actually fulfil their intended goals [5].

The inherent appeal in using modal logics stems from their simplicity and effectiveness for analyzing cryptographic protocols. Logics can be systematically applied to reason about the working of protocols, often helping to reveal missing assumptions, deficiencies or redundancies. This can then lead to the protocol, the assumptions or the original goals being re-evaluated, after which the inference rules can be reapplied to determine whether the goals are attainable after these modifications have been made.

The BAN modal logic [1] popularized the notion of using logics to detect flaws and redundancies in protocols. It has been labelled as a success by many commentators [6,11,4] and has been used to find flaws in several protocols. BAN spawned the creation of a number of related logics, each of which has tried to improve on or add to its underlying premises. A popular descendant of BAN is GNY [7, 8].

However, due to the complexity of the GNY syntax, notation and inference rules, it is commonly acknowledged that analysis with GNY tends to be inaccessible and obscure for the uninitiated. Often it requires experience and insight to determine what the desirable and appropriate initial and final conditions for a given protocol should be. Also, the actual analysis phase during which inference rules are applied can be very tedious and error prone when carried out by hand. Thus, the opportunity exists to create tools that will support analysis efforts by guiding the process from an appropriate starting state to the required final state. Such a system would help to make the rigorous analysis of security protocols more accessible and thus contribute to the overall security level of cryptographic protocols that currently exist and are being designed.

In this paper we will describe the SPEAR II cryptographic protocol analysis tool which we have developed. SPEAR II allows an individual to easily conduct a GNY-based protocol analysis using an intuitive visual interface. This interface offers a multi-dimensional approach to cryptographic protocol analysis, unifying the design and analysis phases of protocol engineering into one consistent and easy-to-use system. We

will elaborate on the core components of SPEAR II and show how they help to distance protocol engineers from the syntactical element of GNY analysis, allowing them to focus more on the associated semantics and distil the critical issues which arise.

The remainder of this paper is organized as follows. In Section 2 we give a brief introduction to GNY analysis. Section 3 gives an overview of some tools that can be used for logic-based analysis. In Section 4 we will describe all of the current SPEAR II components, and then show how the SPEAR II environment is used in an analysis in Section 5. We conclude in Section 7.

2. PRINCIPLES OF GNY ANALYSIS

An analysis with GNY is very similar to one carried out with BAN. However, one significant improvement of GNY over BAN is that it defines an abstract 'protocol parser' which helps to derive a form of the protocol more suitable for manipulation. The major steps carried out before analyzing a security protocol with GNY are enumerated below:

1 Any implicit information conveyed by a protocol formula is represented logically by the attachment of an extension to the formula.

2 A star is placed in front of all formulae containing secrets that the receiving principal is *not* the first to convey in the current session of the protocol. The star also indicates that it is the first time that the receiving principal receives the formula in the current session.

3 The initial belief and possession sets of each principal are constructed. The possession set consists of all formulae available to the principal, while the belief set includes the current beliefs of the principal.

4 The desired final possession and belief sets for each principal are specified based on the design goals of the protocol.

Once these steps have been performed, an analysis can proceed. Each analysis essentially consists of deriving a series of assertions, each assertion being obtained by the application of the GNY inference rules to the assertions already contained within the belief and possession sets of a principal. After each assertion is derived, it is added to either the belief or possession set of the relevant principal. Once the analysis is complete the belief and possession sets will contain the final state of each principal after the protocol has run to completion. This information can then be compared to the desired final conditions to determine whether the protocol has achieved its intended goals.

3. LOGIC-BASED ANALYSIS TOOLS

A number of tools exist to carry out automated logic-based analyses. However, the interface to these tools is often textual, and in cases where a GUI is used to define the protocol to be analyzed, the GNY logic statements are still defined using textual commands.

Convince is an automated toolset that facilitates the modelling and analysis of cryptographic protocols [9]. A protocol is specified by using an integrated commercial GUI system, however GNY statements which are used for analysis must still be defined through textual annotations.

A Prolog-based analysis tool was created to facilitate in a GNY analysis [12]. However, this tool again makes use of textual input schemes which are then analyzed by the Prolog program.

The SPEAR multi-dimensional protocol analysis tool allows a user to specify a protocol in an intuitive graphical environment [3]. Logic-based analysis is conducted using BAN. However, even though the tool has a GUI for defining the protocol, BAN statements have to be constructed in a textual form. Primitive assistance is provided when constructing BAN statements by providing the user with a list of operators and operands which can be added to the current BAN statement.

4. THE SPEAR II FRAMEWORK

The SPEAR II framework aims to provide a unified graphically-based environment within which security protocols can be specified, analyzed and then implemented. A major aim of the framework is to ensure speed and ease of use, but at the same time ensure that quality protocol engineering takes place. At present, the framework consists of three primary components, each of which will be described in the sections that follow.

4.1. GYPSIE

GYPSIE is a graphically-based cryptographic protocol specification environment [13]. Using GYPSIE, a designer can specify a security protocol by employing three basic components. All of these components are rendered on a design canvas in a style reminiscent of SDL and MSCs. The graphical representation of each of these components has been selected to provide the most intuitive and simplistic representaticn of the real-world analogues. These canvas components are listed below:

- *Principals,* which send and receive messages.
- *Messages,* which contain formulae such as nonces and hashes.

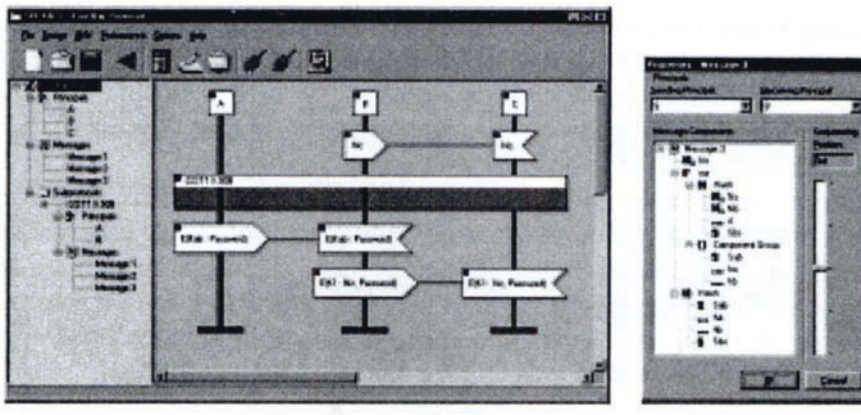

Figure 1 An illustration of the high-level protocol view on the left and the component view on the right.

- *Subprotocols,* which contain further principals and messages.

The high-level protocol view (illustrated in Figure 1) allows a designer to manipulate principals, messages and subprotocols that are found in the protocol model. Messages, principals and subprotocols can be dragged and dropped on the canvas and imported from other protocols which are included in the subprotocol hierarchy. Messages and subprotocols are ordered in time, based on when they are sent, received or called. All design operations can be carried out through the use of pull-down or pop-up menus, depending on the user's preference. The high-level view also has full undo and redo capabilities, and allows a protocol model to be saved, loaded or exported to a number of formats. It is also fully customizable and allows users to select the colours for each component, dragging styles and message display options.

The more detailed component view (shown in Figure 1) hierarchically displays the formulae within an individual message as a structured tree-view. Within this tree-view, formulae can be manipulated, edited, deleted, reordered and dragged and dropped onto encryption, function or grouping nodes. Thirteen primary formula types exist:

- *Non-terminal* formulae include functions, hashes, symmetric encryptions, public-key encryptions, private-key encryptions and groups.
- *Terminal* formulae include nonces, timestamps, shared secrets, symmetric keys, public keys, private keys and user-defined types.

Cut, copy and paste facilities exist within both the high-level and component views, allowing designers to copy or move formulae between

messages and subprotocols. Within the component view, tooltips aid users by displaying the contents of non-terminal formulae when hovering over them. In the high-level view, tooltips are used to display message formulae when hovering over a message, as the message component on the canvas may be shortened to save screen real-estate.

A number of other useful features are included in GYPSIE. The 'component tracker' allows a designer to highlight all of the places where a given formula appears on the design canvas. A navigation bar on the left side of the high-level view aids users in visualizing the structure of a given protocol. GYPSIE also includes the ability to calculate synchronous and optimal rounds [10] for the specified protocol.

4.2. GYNGER

GYNGER is a Prolog-based GNY analysis engine that uses forward-chaining techniques to derive all of the possible GNY statements applicable to a given protocol and set of initial assumptions. The analyzer is based on the one presented in [12], however, it implements more GNY inference rules (71 in total) and uses an improved syntax to facilitate the use of advanced GNY constructs.

To analyze a protocol with GYNGER, one must supply the protocol message steps and the initial belief and possession sets. Any target goals may also be specified. When the analyzer is invoked, the GNY inference rules are applied to the current set of GNY statements. When no more statements can be derived, the analysis process terminates. For those goals which were successful, a proof is generated listing all of the statements and inference rules used to derive the result. A list of all of the derived statements can also be generated.

4.3. VISUAL GNY

The Visual GNY environment (illustrated in Figure 2) makes use of structured trees to represent GNY statements [14]. For each principal within a given protocol specification, up to four sets of structured trees are created, two for the storage of initial beliefs and possessions, and another two for the storage of target beliefs and possessions. A further four sets of trees can be used to store the successful beliefs, successful possessions, failed beliefs and failed possessions for each principal upon the completion of a successful GNY analysis. A set of structured trees is also created for every formula that has extensions, these extensions being defined in the Visual GNY environment.

Five tabbed panes are used to guide a user through the process of specifying initial GNY assumptions, goals and analyzing a protocol. Within

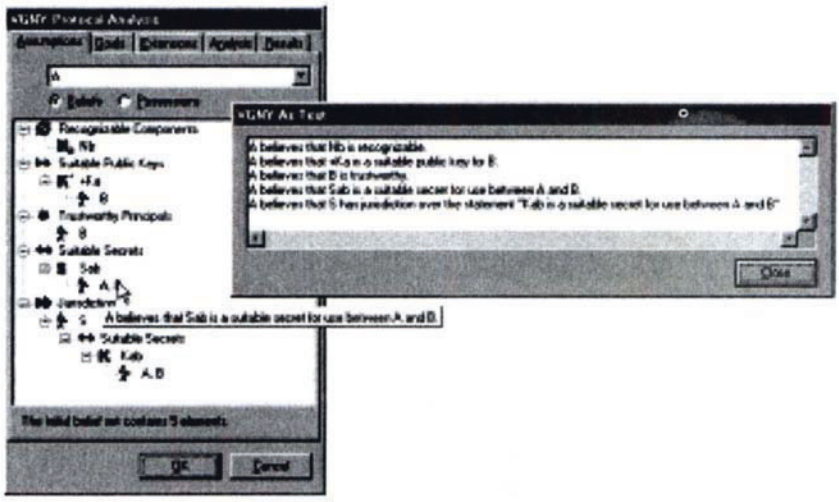

Figure 2 The Visual GNY environment.

each of these tabbed panes, a drop-down combo-box and a selection of radio buttons are used to select the appropriate set of structured trees to modify or view. The currently selected set of structured trees is displayed in a tree-view component centered within the client area of the tabbed pane. Changing either the combo-box or radio button selection changes the set of structured trees being displayed in the tree-view. A label situated below the tree-view indicates the number of GNY statements represented by the set of structured trees displayed in the tree view. All interaction with the structured tree takes place through pop-up menus that are dynamically constructed depending on the selected tree node.

The pop-up menus that are used in the Visual GNY environment are constructed dynamically so as to guide a user as she constructs the structured tree representation of a given GNY statement. Commands on a pop-up menu present a user with a choice of GNY statement types, principal names and message formulae to include in a structured tree. The structured trees are always ensured of being syntactically correct as the pop-up menus used in their construction reveal only the commands applicable to the currently selected node.

The Visual GNY environment attempts to structure and organize GNY analyses. The tabbed panes give an indication of the information required for an analysis, and are roughly laid out in the order that this information would be supplied. Nodes within a given structured tree can be expanded or collapsed as required. If a node contains children then

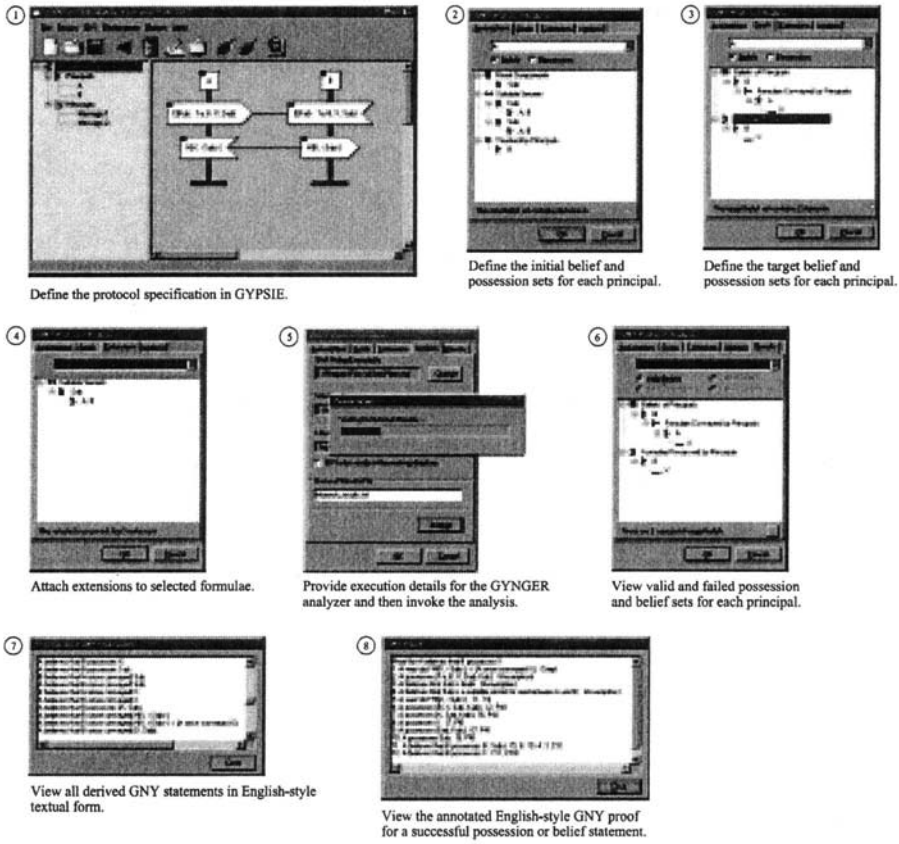

Figure 3 Steps undertaken when conducting a GNY protocol analysis.

a clickable token is displayed to its left. Clicking on this token allows the node to be collapsed or expanded, thus allowing a user to control the amount of information which is presented. In this way the level of detail provided by the interface can be varied appropriately.

GNY statements created within the Visual GNY environment can be exported to an English-style textual format, a mathematical-style LaTeX format, or to GYNGER-compatible Prolog statements. When hovering a mouse pointer over the tree node that terminates a given GNY statement in a structured tree, a tooltip containing the English-style text representing the statement is displayed. Besides this aid, a user can also view all of the statements contained in a given tree-view as English-style text in a pop-up window. This feature is activated from the pop-up menus.

5. GNY ANALYSIS WITH SPEAR II

In Figure 3 we sketch the steps that are undertaken during a typical analysis session. Such a session normally begins by specifying the principals, messages and formulae of the protocol in question within the GYPSIE specification environment. Once this phase has been completed, the Visual GNY environment is invoked and the initial assumptions and goals of each principal are specified as required. Extensions are also appended to formulae. Once all of the necessary preconditions have been defined, details such as the location of the Prolog interpreter, the location of the GNY rules Prolog source, working directories and output files are defined within the *Analysis* tabbed pane. Upon the initiation of the analysis process, the structured GNY trees are all translated into a GYNGER-compatible Prolog syntax, the GNY protocol parser is invoked, and the analyzer is then called with the relevant parameters. The Visual GNY environment monitors the analysis thread, and when it is complete, retrieves the results from the output files, parses these results, and then constructs the appropriate structured trees to display in the *Results* pane. Proofs and the list of all derived statements are also stored .

As we can see, a typical analysis session is very visual, with the graphical environment being used as much as possible to aid and guide the user. The *Results* tab is only displayed if an analysis has been conducted, and is hidden if any deletions are made from the GNY preconditions. To view the proof for a valid target goal, one merely needs to right-click on the terminal node of the statement's structured tree representation and then select the *View GNY Proof* menu item. To view all of the GNY statements derived during the most recent analysis, the button in the lower right of the *Analysis* tabbed pane is pressed. All of the constructed GNY statements and analysis results are saved together with the GYPSIE protocol specification. The analysis results are also saved to an output file defined within the *Analysis* tabbed pane. The undo and redo feature within GYPSIE is very useful for protocol analysis, since it allows a user to conduct analyses on variations of the same protocol. For instance, an analysis can be conducted with a certain formula contained within the protocol messages. This formula can then be deleted and another analysis conducted, with the two results being compared at the end. If the first results are better, then the deletion of the formula can be undone. In this way, we can determine whether a given formula is redundant with respect to its effect on helping to achieve the protocol goals.

6. EXPERIENCES WITH USING SPEAR II

Over the past few months we have conducted a number of usability and practical usage experiments with SPEAR II. The usability experiments tested the interface of the Visual GNY environment and the ease with which individuals work within the GYPSIE modelling environment. Besides these user experiments, we have also tested the GYNGER analyzer and used it to analyze a wide variety of authentication protocols, as well as information exchange protocols. Some of these protocols include, the Needham Schroeder and Voting Protocols [8], the Wide Mouth Frog, Yahalom and Kerberos Protocols [1], as well as a number of authentication protocols from [10]. All the results from our analyses worked out as expected and returned accurate results and proofs.

The Visual GNY experiments returned some interesting results. We tested the interface on fifteen fourth-year Computer Science students who had taken a course in network security and protocol analysis with GNY. The last time any of these students had used GNY was almost six months prior to the experiment. They were each asked to specify a set of GNY statements in both conventional mathematical syntax and structured tree notation using the Visual GNY environment. On average, they specified 78% of the GNY statements correctly using mathematical notation, and 98% correctly using the Visual GNY environment. Only substitution errors were made when using the Visual GNY environment, as syntactic correctness correctness is enforced by the interface. When we asked these students to translate Visual GNY and mathematical style statements into English-style text, they got approximately 85% correct on average. This demonstrates that Visual GNY does not improve the readability of GNY statements. However, this is not much of an issue, as the tooltips and 'View GNY Statements as Text' features both display the constructed GNY structured trees as English-style text.

Experiments conducted with the GYPSIE interface set out to test how individuals interacted with the environment and how effectively they could specify protocols. We made use of another batch of twenty fourth-year Computer Science students who had all studied network security techniques. They were asked to specify three protocols: a voting protocol, an authentication protocol and the Needham-Schroeder protocol. On average, 0.5 mistakes were made in the voting protocol, 0.20 in the authentication protocol, and 0.65 in the Needham-Schroder protocol. The average construction times were 300 seconds, 378 seconds, and 589 seconds respectively. As evidenced by these figures, the GYPSIE environment facilitates accurate protocol construction, and intuitively we can assume that it will make individuals more productive and more

effective than they would be in a text-based system, since they do not need to concern themselves with syntactical issues but can instead focus on the protocol at hand and its associated semantics.

7. CONCLUSION

Security protocol engineers need to be familiar with security protocol analysis techniques and must also be able to effectively put these into practice. However, to be useful an analysis method must also be usable. We cannot expect individuals to be able to readily recall the syntax associated with a modal logic such as GNY or the plethora of inference rules used in an analysis, as this syntactical knowledge is often forgotten after it has not been applied for a while. Instead, the associated semantic issues and an understanding of how an analysis occurs should be the focus of an individual's analysis arsenal, tools and reference material being used to fill in any syntactical gaps.

There are a number of tools that can be used to carry out automated GNY protocol analysis [12, 9]. However, an impediment to using most of these is the construction of the specification which describes the protocol messages, formulae, initial assumptions and target goals. Supplying this information is not always a simple and straight-forward task and its prompt, efficient and error-free delivery often depends on the type of software being used. For this reason, the use of software that helps to distance protocol engineers from the syntactical element of protocol analysis, allowing them to focus more on the underlying critical issues, should be encouraged.

A formal analysis method should not just be studied and forgotten. Instead, the security community should be encouraged to develop tools that facilitate and encourage its use by a broad spectrum of individuals. When creating such tools, we should bear in mind that they should promote information recall, not require it. A tremendous amount of research has been carried out on security protocol analysis techniques [6], but how much of this research actually gets used in the field by the engineers who work there? Let's not allow good techniques to go unused. By encouraging more protocol analysis techniques to be applied, we will encourage the development of more robust and secure protocols.

Thus, by leveraging specially developed tools and techniques, a large portion of the difficulties that individuals encounter when using formal methods can be resolved. The SPEAR II tool [1] is a graphically-based analysis environment within which GNY protocol analysis can be conducted. SPEAR II places a user-friendly front-end on the GNY analysis process, thus freeing individuals to focus more on an analysis and the

issues related thereto, instead of having them bogged down in syntax and tedious inference rule application. We hope to continue development of the SPEAR II framework by adding more analysis techniques and ensuring that these techniques can be used by protocol engineers when implementing and designing network security protocols.

Notes

1. Available from http://www.cs.uct.ac.za/Reaearch/DNA/SPEAR2.

References

[1] M. Abadi, M. Burrows, and R. Needham. A Logic of Authentication. In *Proceedings of the Royal Society,* Series A, 426, 1871, pages 233 – 271, December 1989.

[2] M. Abadi and R. Needham. Prudent Engineering Practice for Cryptographic Protocols. *IEEE Transactions on Software Engineering,* 22(1):6 – 15, January 1996.

[3] J.P. Beckmann, P. De Goede, and A.C.M. Hutchison. SPEAR: Security Protocol Engineering and Analysis Resources. In *DIMACS Workshop on Design and Formal Verification of Security Protocols.* Rutgers University, September 1997.

[4] J. Clark and J. Jacob. *A Survey of Authentication Protocol Literature: Version 1.0,* November 1997.

[5] V.D. Gligor, L. Gong, R. Kailar, and S. Stubblebine. Logics for Cryptographic Protocols – Virtues and Limitations. In *Proceedings of the Fourth IEEE Computer Security Foundations Workshop,* pages 219 – 226, Franconia, New Hampshire, October 1991. IEEE Computer Society Press.

[6] P. Georgiadis, S. Gritzalis, and D. Spinellis. Security Protocols Over Open Networks and Distributed Systems: Formal Methods for Their Analysis, Design and Verification. *Computer Communications,* 22(8):695 – 707, May 1999.

[7] L. Gong, R. Needham, and R. Yahalom. Reasoning about Belief in Cryptographic Protocols. In *Proceedings of the 1990 IEEE Symposium on Research in Security and Privacy,* pages 234 – 248, Oakland, California, 1990. IEEE Computer Society Press.

[8] L. Gong. *Cryptographic Protocols for Distributed Systems.* PhD thesis, University of Cambridge, April 1990.

[9] R. Lichota, G. Hammonds, and S.H. Brackin. Verifying the Correctness of Cryptographic Protocols using Convince. In *Proceedings of*

the Twelfth IEEE Computer Security Applications Conference, pages 117 – 128. IEEE Computer Society Press, 1996.

[10] L. Gong Lower Bounds on Messages and Rounds for Network Authentication Protocols. In *Proceedings of the 1st ACM Conference on Computer and Communications Security,* pages 26 – 37, Fairfax, Virginia, November 1993.

[11] C.A. Meadows. Formal Verification of Cryptographic Protocols: A Survey. In *Advances in Cryptology - Asiacrypt '94*, pages 133 – 150. Springer-Verlag, 1995.

[12] A. Mathuria, R. Safavi-Naini, and P. Nickolas. On the Automation of GNY Logic. In *Proceedings of the 18th Australian Computer Science Conference,* volume 17, pages 370 – 379, Glenelg, South Australia, February 1995.

[13] E. Saul and A.C.M. Hutchison. A Generic Graphical Specification Environment for Security Protocol Modelling. In *Proceedings of the Sixth Annual Working Conference on Information Security,* pages 31 1 – 320, Beijing, China, August 2000. Kluwer Academic Publishers.

[14] E. Saul and A.C.M. Hutchison. A Graphical Environment for the Facilitation of Logic-Based Security Protocol Analysis. *South African Computer Journal,* (26):196 – 200, November 2000.