A Graphical Environment for the Facilitation of Logic-Based Security Protocol Analysis

E. Saul[a] A.C.M Hutchison[b]

DNA Research Group, University of Cape Town, South Africa
[a]esaul@cs.uct.ac.za, [b]hutch@cs.uct.ac.za

Abstract

The development of cryptographic logics to analyze security protocols has provided one technique for ensuring the correctness of security 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 a graphical tree-based specification environment which can be used to easily construct GNY statements using contextualized pop-up menus. The interface which we describe helps to move logic-based analysis out of the world of academia and into the mainstream market.

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. As a result of this fact, cryptographic logics have been developed to aid in determining whether protocols actually fulfil their intended goals. Using logics to analyze security protocols has a number of advantages:

- The use of logics forces protocol designers to explicitly state the security assumptions which they have made and will require after the protocol has executed.
- Reasoning with logics makes designers think about the use for which each component is intended, thus minimizing redundancy.
- Cryptographic logics can also be used to explicitly bind the evolution of beliefs in a protocol session to message contents, number of messages and message rounds, thus helping to determine the minimum number of messages required to achieve a given set of beliefs and possessions.

Analysis using logics was first popularized in 1989 by the BAN modal logic [2]. BAN and other logic systems have successfully been used to reveal flaws in protocols that were previously accepted as correct [4]. A popular successor of BAN is GNY [5].

However, it is commonly acknowledged that analysis using a modal logic such as 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. The opportunity exists to support analysis efforts by guiding the process from an appropriate starting state to the required final state, if it is achievable. Such a system would aid in making 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.

The aim of this paper is to describe a graphical environment which can be used to specify GNY statements that form the initial assumptions and final goals for a given protocol. The catalyst for this work was the development of a security protocol engineering and analysis resource (SPEAR) [3], which currently carries out automated BAN analysis. The SPEAR project aims to provide automated GNY analysis in its second iteration, however experience during the development of the tool revealed the need to implement techniques which would make it easier for a protocol designer to perform a security analysis.

The remainder of this paper is organized as follows. In Section 2 we present a brief overview of the GNY logic, while Section 3 presents a brief background on logic-based analysis tools. Section 4 describes the GUI-based GNY specification environment which has been developed. We close off by describing future work to be carried out in Section 5 and end with a conclusion in section 6.

2 GNY Concepts and Notation

In this section the basic notions underlying the GNY reasoning process will be introduced. We will first briefly describe the notion of a formula and formula extensions, after which we will describe how formulae and GNY operators can be combined to form statements. Finally, we present a brief description of how a GNY analysis is normally conducted.

2.1 Formulae

A *formula* in a protocol description is a name referring to a bit string which would have a particular value in a session. This is analogous to a variable identifier in a programming language. Principals often exchange formulae to express their current beliefs or transfer information. Beliefs are described by statements, introduced in the next section. Let C range over all statements and let X be a formula. Then $X \rightsquigarrow C$ is also a formula, more specifically, a formula with an *extension*. Statement C is the extension and is considered an integral part of the formula.

Essentially, an extension to a formula is a formal specification which dictates that a principal should proceed to send a formula only if certain conditions hold. For example, the formula $X \rightsquigarrow C$ can only be sent by a principal if he believes in the validity of the statement C. A formal specification such as this helps to eliminate ambiguity as these conditions are often only expressed verbally in traditional protocol specifications. Having accepted that a formula is genuine, the recipient can choose to believe that the formula's extension holds, if he trusts the sender's competence and honesty.

2.2 Statements

A basic *statement* reflects some property of a formula, typically reflecting a relation between a principal and a formula. Let P and Q range over principals. The following are statements:

$P \triangleleft X$: P *is told* a formula, X, possibly after performing some computation, such as decryption. Thus, the formula being told is itself, or some computable content thereof.

$P \triangleleft *X$: P is told a formula, X, which he is *not* the first to convey in the current session of the protocol, though he could have transmitted it in a previous session. *Also,* it is the first time that P receives X in the current session.

$P \ni X$: P *possesses* formula X. P is able to repeat this formula in future messages of the current session. At a particular stage of a session, P possesses all the formulae that he has been told, all the formulae he started the session with, and all the ones that he generated during the current session. In addition, P possesses all the formulae that are computable from formulae he already possesses.

$P \mid\sim X$: P *once conveyed* formula X. X can be a formula explicitly exchanged or some computable content of a formula. Thus, a formula can also be exchanged implicitly.

$\sharp(X)$: Formula X is *fresh*. A principal should believe that a formula originated by another principal is fresh if it has been constructed after the occurrence of some fresh event. A principal believes anything he has originated to be fresh if he cannot have chosen the same formula for the same purpose before.

$\phi(X)$: Formula X is *recognizable*. A principal would believe X to be recognizable if he has certain expectations about the value or structure thereof. He may recognize a particular value, a particular structure or other forms of redundancy. In either case, he may not possess part or all of the formula.

$P \overset{S}{\longleftrightarrow} Q$: S is a suitable *secret* for P and Q. These entities may use S to prove their identities to each other. They may also use it as, or derive from it, an encryption key K to communicate, denoted as $P \overset{K}{\longleftrightarrow} Q$. This notation is symmetrical.

$\overset{+K}{\mapsto} Q$: $+K$ is a suitable *public key* for Q. The matching secret key is given by $-K$.

The only default assumption which we require is that S, K or $-K$ will never be discovered by any principal except the legitimate owners or principals which the owners trust. In the latter case, the trusted principals should never use S, K or $-K$ as a proof of identity or as an encryption key to communicate. Continuing, the following are also statements:

$P \propto X$: P is eligible to convey formula X. P holds the relevant possessions and beliefs. This notation is used to detect inconsistencies in the protocol description.

$P \dashv (X)$: P is *not the first* principal to originate formula X. This formula must first be generated and conveyed by another principal.

Statements are often associated with individual principals to specify their states. Let C range over statements. The following are also statements:

(C_1, C_2): The conjunction of two statements. Conjunctions represent sets and have properties such as associativity and commutativity.

$P \mid\equiv C$: P believes that statement C holds. $P \mid\equiv$ is considered an empty statement.

$P \mid\equiv Q \mid\Rightarrow C$: P believes that Q has *jurisdiction* over statement C. He believes that Q has authority on C and should be trusted in this respect.

$P \mid\equiv Q \mid\Rightarrow Q \mid\equiv *$: P believes that Q has *jurisdiction* over all his beliefs. P considers Q to be competent and honest.

2.3 Conducting an 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 that contains a secret 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 [6]. 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 [1]. 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 Specifying GNY Statements

When using GNY to carry out an analysis, many individuals struggle to understand the notation and formulate syntactically and semantically correct statements. Also, after specifying a set of initial beliefs and possessions, one is often left with a collection of cryptic looking statements that have no structure or organization of any form.

No tools currently exist which allow a user to specify GNY statements in an intuitive graphical interface. The tools which do exist require that the user enter GNY statements in a textual specification environment. This is far from ideal as it often results in users becoming bogged down in GNY syntactical and semantic issues. A graphical environment supporting the creation of GNY statements should fulfil the following criteria:

- The user should not be expected to remember the GNY syntax and symbols. However, he should have a working knowledge of GNY and be acquainted with the associated semantics.
- The representation of GNY beliefs and possessions should be as concise as possible, yet it should be possible to easily view all of the defined statements.
- Beliefs and statements which have been specified should be structured and organized so that it is easy to locate them for later modification or referral.
- The specification environment must allow all possible GNY statements to be constructed. However, no syntactically incorrect statements should be permitted.
- The efficiency and effectiveness of an individual using the environment should not be hampered or restrained in any way. The specification environment should also be intuitive and simple to use, requiring minimal keystrokes and mouse clicks.

To represent GNY statements, we have chosen to make use of a tree-based view with pop-up menus being used to add components, principals and belief categories. A prototype is illustrated in Figure 1. This approach imposes a hierarchical structure on the GNY statements and makes the representation of these statements as concise as possible. A tree-based view combined with pop-up menus also aids users in specifying GNY beliefs and possessions by ensuring that they do not have to remember any cryptic GNY syntax or symbols.

4.1 The Interface

The tree-based interface allows a user to construct GNY statements by using only a pointing device such as a mouse. A principal having no beliefs or possessions contains two empty panels, one representing his belief set and the other his possession set. These panels are populated by the user through selections from various contextualized pop-up menus.

Categories which reflect different types of beliefs are added to the belief panel. A belief category exists for each type of GNY statement. Each of these categories can contain principal or component nodes or even further belief categories, depending on their type. Belief categories that only contain principal or component nodes will contain subtrees with a fixed depth, while those which can contain further belief categories will have subtrees with variable depths. Possessions are simply added to the possession panel for each principal.

Consider the GNY statement tree displayed in Figure 1. This tree contains beliefs for principal *A*. Eight

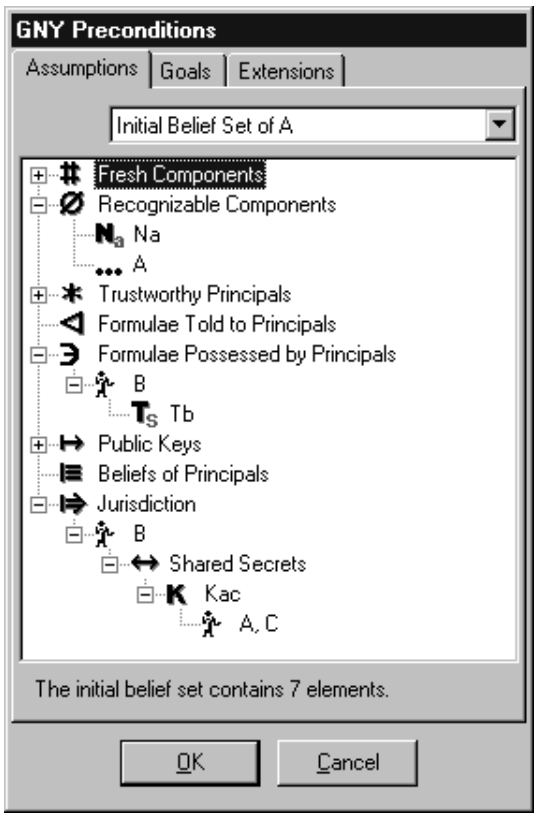

Figure 1: A prototype of the GUI-based tree-view used to represent GNY statements.

categories of beliefs have been specified, six of which are populated with at least one statement. The visible belief statements contained within this tree are as follows:

(1) $A \mid\equiv \phi(N_a)$
(2) $A \mid\equiv \phi(A)$
(3) $A \mid\equiv B \ni T_b$
(4) $A \mid\equiv B \mid\Rightarrow A \overset{K_{ac}}{\longleftrightarrow} C$

Nodes within the tree can be deleted or expanded. If a node containing children is deleted, then these children are removed from the tree as well. If a node contains children then a clickable token is displayed to the right thereof. Clicking on this token allows the node to be collapsed or expanded, thus allowing a user to control the amount of information which is be presented.

Often while working in a tree-based structure such as the one which we have described, a user might need to know what GNY statement is represented by a specific node. For this reason, a feature which displays the GNY statement represented by a given node through the use of tooltips has been added. When hovering over a node, the GNY statement which it represents is briefly displayed. For example, if the user were to hover a pointer over the node 'A,C' in Figure 1, then the text "A believes that B has jurisdiction over the suitability of the key K_{ac} shared between A and C" would be displayed.

4.2 Implementing Functionality

Pop-up menus are used to present a user with functions to perform on nodes and lists of components to add. Different pop-up menus are displayed according to the type of node which has been selected. Thus, a user does not have to use a keyboard and is not in danger of adding an incorrect node since he is guided through the statement construction process by the menu options presented to him.

At present, the prototype which has been implemented uses thirteen different pop-up menus. Some of these are specific to a given node type, while others are shared between nodes. For the sake of brevity, we will not describe all of these pop-up menus, but will present a brief sample of those which have been constructed:

- The 'Jurisdiction' and 'Beliefs of Other Principals' nodes share a pop-up menu which allows a user to add a principal that will serve as the target of the jurisdiction or belief statement. Two further options allow a user to expand all of the subnodes and to delete the node. The principal node which is created has an associated pop-up menu which allows a belief category to be added as a child-node. This menu also contains an option to delete the current node and to expand all subnodes. The beliefs root folder has a very similar pop-up menu which does not contain the delete command.

- The 'Recognizable Components' and 'Fresh Components' nodes share a pop-up menu which allows a user to add different types of components as children and to delete the node. The pop-up menu also contains a submenu that can contain suggestions for components to be added. These suggested components can be determined by the design environment based on the type of protocol being constructed. For example, the sender of a nonce always considers it to be fresh.

- The 'Public Keys' and 'Shared Secrets' nodes share a very similar structure. The only difference is that the pop-up menu for the 'Public Keys' node allows a user to add a public key as a child-node, while the 'Shared Secrets' pop-up menu presents a user with a list of shared-secrets. Both menus allow for the deletion and expansion of the node to reveal all of the subnodes.

- Within the 'Public Keys' node, public keys which the user has added are listed. The principals who trust in the suitability of one of these public keys can then be added as children to that key's node using a suitable pop-up menu. In the case of shared-secrets the principle is similar. However, the issue is also slightly more complicated, as a secret is shared between *two* principals and pop-up menus are not really suitable to specify this fact. Instead, a pop-up menu is used to open a modal dialog which allows a user to select the two principals sharing the secret.

Because the task of adding a GNY statement is not always a single-step operation, half-completed beliefs could

reside in the tree. In this case, the system alerts the user to this fact when closing the dialog and requests that the statements be completed. Automatic deletion of incomplete statements is not appropriate, as a user may have merely been distracted and forgot to complete a given statement.

4.3 Completeness of the Specification

At this point we will informally show that any GNY statement can be represented in the specification environment. We define the statement "$A \Rightarrow B \Rightarrow C$" to represent a subtree where A is the root, B is a child of A and C is the child of B. So, assume that P and Q range over principals, C ranges over statements, X ranges over formulae, $+K$ is a public key and S is a shared secret. The $\Box$ symbol ranges over all belief categories, while $\star$ ranges over all GNY operators. From the context, it will be clear what $\Box$ and $\star$ represent.

1. The eligibility, told, possession and conveyance statements, which all have the form $Q \star X$, can be represented in the tree as $\Box \Rightarrow Q \Rightarrow X$.

2. Freshness and recognizability statements have the form $\star(X)$ and can be displayed in the tree as $\Box \Rightarrow X$.

3. The belief in the suitability of a secret shared between P and Q can be represented in the tree as $\Box \Rightarrow S \Rightarrow (P, Q)$. Similarly, the belief in the suitability of a public key can be represented as $\Box \Rightarrow +K \Rightarrow Q$.

4. The fact that principal Q is considered to be honest and competent can simply be represented as $\Box \Rightarrow Q$ in the tree. Notice that the cryptic GNY syntax for this fact is totally removed.

5. Jurisdiction and belief statements both have the form $Q \star C$. Since C is a statement, it can be any one of those which we have already described in this list. To represent jurisdiction or belief we merely use the arrangement $\Box \Rightarrow Q \Rightarrow C$ in the tree.

Thus, all GNY statements can be represented in the tree view by placing nodes correctly within the tree. The enforcement of this correct placing is handled by pop-up menus and as a result only syntactically correct GNY statements can be generated.

5 Future Work

The specification environment which has been described in this paper forms part of a larger GNY analysis system which is currently being implemented. The primary aim of this system is to create an environment which will facilitate the rapid yet high-quality analysis of security protocols using the GNY modal logic. The specification environment will be used to define the initial beliefs and possessions for a given principal, as well as the target beliefs and possessions. Formula extensions will also be defined in this environment. Once initial beliefs, possessions and extensions have been defined, an analysis can take place and the system can then report on which of the target goals have been attained. The analysis will be conducted by passing information defined in the GNY specification environment to a Prolog-based analyzer.

6 Conclusion

Security logics such as GNY enable designers to prove that security protocols achieve their intended design goals. However, merely having the facility of security logics is not sufficient, since these logics should also be usable by a broad spectrum of protocol designers, even if they are not academically inclined.

We have created a GNY statement specification environment which uses a combination of a tree-based specification system and contextualized pop-up menus. This combination of GUI components allows a user to specify GNY statements without having to be acquainted with the associated syntax or notation. The tree-view also structures the GNY statements and allows a user to control the level of detail which is displayed, thus allowing him to focus on the core elements of a protocol analysis.

With an interface such as the one which we have created in place, the process of moving logic-based analysis out of the world of academia and into the mainstream market moves one step closer.

References

[1] A. Mathuria and R. Safavi-Naini and P. Nickolas. On the Automation of GNY Logic. 17(1):370 – 379, 1995.

[2] 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.

[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] 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.

[5] L. Gong. *Cryptographic Protocols for Distributed Systems*. PhD thesis, Cambridge University, United Kingdom, 1990.

[6] 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.