On the Automation of GNY Logic

Anish Mathuria* Reihaneh Safavi-Naini* Peter Nickolas

Centre for Computer Security Research
Department of Computer Science
University of Wollongong, NSW 2522
email: {g9312439, rei, peter}@cs.uow.edu.au

August 1994

Abstract

The Gong, Needham and Yahalom (GNY) logic is a belief-based logic for analyzing cryptographic protocols. It offers significant advantages over its predecessor, the Burrows, Abadi and Needham (BAN) logic, which has been successful in formally discovering flaws and redundancies in several well-known protocols. Manual analysis of protocols using the GNY logic, however, is cumbersome, as the logic has a large set of inference rules.

This report proposes a modified GNY logic, and describes the implementation of a protocol analysis tool based on that logic. The modifications are introduced to ensure that the logical statements derivable from a given protocol are all deducible in a finite number of steps. The nature of the modifications is such that no useful inferences are lost. The tool offers a facility to automatically generate proofs of protocol goals. It also generates a logical description of the state attained after every protocol step, thus allowing the stepwise development of protocols. The tool has proved useful in mechanically verifying the need for several inference rules which are absent from the original GNY logic.

Keywords: Cryptographic logics, protocol verification, protocol security, authentication protocols, key distribution protocols.

1 Introduction

The rapid proliferation of distributed computing systems has lead to an increased dependence on cryptographic techniques for protecting information transmitted over insecure channels. These techniques are used to counter the threats posed by passive eavesdroppers or active intruders interfering with the message transmission to meet malicious ends. Cryptographic protocols can be used to attain a multitude of security objectives like entity authentication, key distribution, message authentication, etc. (cf., e.g., [1]). It is well recognized that such protocols can be subverted without directly compromising the underlying cryptosystem [2, 3]. These protocols are susceptible to various kinds of attacks which lead to a breakdown of the security of the entire system. Examples of protocols which were argued to be correct by *ad hoc* techniques, but later found to contain flaws, abound in the literature [4, 5, 6]. It is therefore essential to employ systematic techniques for ensuring that protocols meet their desired goals under reasonable assumptions about the operating environment.

Formal analysis of protocols using a belief–based logic was proposed in the pioneering work of Burrows, Abadi and Needham [5, 7]. The BAN logic provides a language for specifying and analyzing protocols at a level adequate to successfully demonstrate the existence of flaws in

*Support for this work was provided in part by the University of Wollongong Computer Security Technical and Social Issues research program, Australian Research Council (ARC) small grant, and ARC grants A49131885 and A49030136.

several existing protocols. It has also revealed subtle differences between different protocols with seemingly identical requirements and goals. The logic makes some implicit assumptions about the working of protocols. Some of these are standard cryptographic assumptions such as perfect encryption[1], whereas others such as the assumption of sufficient redundancy in encrypted messages are less common, and are very often advantageous to specify explicitly. The BAN logic also fails to accommodate many of the wide range of cryptographic techniques available in designing protocols, and as a result the class of protocols that can be analyzed using the logic is limited. For example, the logic cannot be used to analyze protocols which are based on one-way hash functions instead of encryption (cf., e.g., [8]). The shortcomings of the BAN logic have lead to several extensions being proposed [9, 10, 11, 12, 13]. Each of these extensions attempts to improve upon the BAN logic by introducing additional primitives and rules. A prominent extension of the BAN logic is the logic proposed by Gong, Needham and Yahalom [9]. Their extension, known as the GNY logic, operates at a finer level of detail than its predecessor, and also extends the class of protocols that can be analyzed using the logic. It does this by introducing several new constructs and a large number of rules to formalize some of the implicit assumptions of the BAN approach, and to capture the wide variety of techniques that can be employed in protocols.

The simplicity and effectiveness of the BAN logic and its extensions has resulted in the development of several tools for machine-aided analysis of protocols by using BAN or BAN-like logics [14, 15, 16, 17]. The appeal of tools for mechanical validation is clear, but such tools can also assist in examining the role played by protocol messages and assumptions in attaining the desired goal. In addition, the tools can also be used to verify proofs of protocol goals which are obtained by manually applying the logic. Manual analysis of protocols using the GNY logic, however, is unwieldy, as the logic has more than forty inference rules.

In this report we describe a tool for automating GNY logic analysis of protocols. To obtain this automation we modify the set of inference rules of the logic in such a way that the logical statements that can be derived from a given protocol are finite in number, and are therefore derivable in a finite number of steps. The goal of the original logic is to deduce the possessions gained by every principal executing the protocol from the messages sent to him, and his beliefs about the sender that can be deduced from the messages sent. The modifications we carry out discard those inferences which do not contribute towards this goal, so no useful inferences are lost by the modified logic. We implement the modified logic to establish a tool for protocol verification using the logic. The tool provides a protocol-independent inference engine which generates all the logical statements that can be derived from a given protocol specification consisting of the statements representing the protocol messages and initial assumptions. A proof explanation facility allows proofs of protocol goals to be obtained mechanically. The tool also generates the intermediate states attained after each step of the protocol, thus providing an environment for stepwise development of protocols. The modified logic includes several new rules from an extension of the GNY logic in [10]. These rules are clearly required during protocol analyses using the GNY logic, but are nonetheless absent in [9]. The tool has proved useful in verifying the need for the inference rules we add to the GNY logic. It has also enabled us to detect a problem with the GNY protocol parser [9, p. 238] used for transforming a conventional protocol description into the logic.

The outline of the report is as follows. Section 2 gives a brief overview of the BAN logic and describes the GNY extensions to BAN. Section 3 discusses our modifications to the GNY logic and outlines a proof of finiteness of derivations in the modified logic. Section 4 describes the Prolog implementation of a tool based on the modified logic. We end the report in section 5 by illustrating the the use of the tool in analyzing an example protocol.

[1] Failures arising out of direct cryptanalysis of ciphertext are not addressed.

2 Cryptographic logics: BAN and GNY

The GNY logic is essentially a verification logic for cryptographic protocols. It is an extension of the BAN logic, and adopts the basic notational framework of BAN. Below we review the main features of the BAN logic, and then discuss how the GNY logic attempts to improve upon its predecessor.

2.1 BAN logic

The BAN logic deals with three types of objects: *principals*, *keys*, and *nonces*. Principals represent entities (people, computers, services, etc.) who execute protocols for attaining security goals like distribution of session keys, etc. A protocol specifies a procedure for exchanging messages between principals and is described by a list of steps of the form: $A \rightarrow B : M$, which denotes that principal A sends the message M to principal B. Protocol messages typically contain encrypted and unencrypted parts, and keys are used to perform encryption and decryption. Certain initial conditions are assumed to hold at the start of a protocol run, and a successful execution of the protocol is intended to result in the protocol goal being achieved. For example, a typical authenticated key distribution protocol is often arranged in such a way that two principals, each of whom initially shares a secret key with a trusted server only, are able to establish a shared session key generated by the server for communicating with each other. Nonces are fresh quantities, such as random numbers, typically used for verifying that messages received were sent after the start of the protocol run. Time is divided into two epochs: *past* and *present*. A message sent before the start of the protocol run is said to belong to the *past* epoch. No guarantees can be obtained from secrets distributed in such messages, since they could have been compromised in the past.

In the BAN logic, protocol messages are represented by logical formulae, also referred to as *statements*. A complete description of the constructs and inference rules of the logic can be found in [5]. Here we present the main formulae and logical rules.

$P \mid\equiv X$	P *believes* X. P believes that X is true.
$P \triangleleft X$	P *sees* X. P has received a message containing X.
$P \mid\sim X$	P *once said* X. P has sent a message containing X.
$P \Rightarrow X$	P has *jurisdiction* over X. P is a trusted or delegated authority on X.
$\sharp(X)$	X is *fresh*. No message sent in the past contained X.
$P \overset{K}{\leftrightarrow} Q$	P and Q *share* key K. No one except either P or Q or someone they trust can know K.

The inference rules of the logic capture commonly accepted principles used in designing protocols. Below we give three such rules which are central to the logic.

- *Message-meaning* – The message-meaning rule for shared keys states that the identity of the sender of an encrypted message can be deduced from the encryption key used:

$$\frac{P \mid\equiv P \overset{K}{\leftrightarrow} Q,\ \ P \triangleleft \{X\}_K \text{ from } R}{P \mid\equiv Q \mid\sim X}$$

  The side condition $P \neq R$ guarantees that P did not send $\{X\}_K$ himself.

- *Nonce-verification* – The nonce-verification rule states that the sender of a fresh message must believe what is said:

$$\frac{P \mid\equiv \sharp(X),\ \ P \mid\equiv Q \mid\sim X}{P \mid\equiv Q \mid\equiv X}$$

- *Jurisdiction* – The jurisdiction rule states that one must believe what a trusted principal believes in:

$$\frac{P \mid\equiv Q \Rightarrow X,\ \ P \mid\equiv Q \mid\equiv X}{P \mid\equiv X}$$

A protocol to be analyzed is first transformed into an idealized version to include any implicit information conveyed by protocol messages. The basic idea behind idealization can be illustrated as follows. Consider the protocol step $S \rightarrow A : \{N_a, B, K_{ab}\}_{K_{as}}$, in which a key server S distributes a session key K_{ab} to A for talking to B. Here K_{as} is a key shared by A and S, and N_a is A's nonce, used by him to verify that the message is not a replay. This message implies that S asserts K_{ab} to be a good session key for A and B, and this assertion is represented explicitly in the idealized version of the step as: $A \triangleleft \{N_a, A \overset{K_{ab}}{\leftrightarrow} B\}_{K_{as}}$. Initial assumptions, such as those stating the goodness of keys shared initially between a server and various principals, are then made, and the inference rules are applied to determine if the logical statements describing the goal of the protocol are derivable. The inability to derive those statements very often suggests implicit assumptions in the protocol. As protocol flaws are typically manifested as implicit assumptions which are clearly dubious when made explicit, the logic enables flaws to be detected by forcing all the assumptions to be made explicit. Examples of BAN logic analysis of several well-known protocols can be found in [5]. The usefulness of the logic has made it a widely used formal method for analyzing cryptographic protocols.

2.2 GNY logic

The GNY logic [9] is more expressive than the BAN logic. It makes explicit some of the assumptions which are implicit in the BAN logic. For example, the BAN logic assumes that encrypted messages provide sufficient redundancy; this may not be always true, and it is often advantageous to specify the presence of redundancy explicitly. A large number of protocol failures have resulted from implicit assumptions which are unreasonable in practice. For example, the Needham-Schroeder protocol [1] makes the unreasonable assumption that one of the principals' simply assume that the message containing the session key is freshly generated [4, 5]. Furthermore, it is also useful to make explicit any assumptions which more accurately describe the conditions required for the correct functioning of protocols. The notions made explicit by the GNY logic provide a step in this direction. The logic also has new notation to represent functions other than encryption, such as decryption and one-way hash functions, and new notation for denoting private keys associated with public keys. Many new rules are introduced by the logic, which reflect the wide variety of cryptographic techniques that can be employed in constructing protocols. For example, the message interpretation rule I3 [9, p. 247] captures identity corroboration using one-way hash functions. The additional notation and rules collectively yield an increase in reasoning power, and also broaden the class of protocols that can be analyzed using the logic. Here we discuss the three most notable notions which can be represented explicitly in the GNY logic.

2.2.1 Possession

The notion of possession allows reasoning at a finer level of detail than the BAN logic. This notion is implicit in many BAN rules. For example, in the message decryption rule for shared keys [5, p. 7]:

$$\frac{P \mid\equiv P \overset{K}{\leftrightarrow} Q,\ \ P \triangleleft \{X\}_K}{P \triangleleft X}$$

it is implicitly assumed that P has K. Possessing a key or message is quite distinct from believing anything about it. The construct $P \ni X$, meaning P *possesses* X, provides a means to represent this in the GNY logic. The GNY logic further differentiates between the bit strings which protocol messages contain and the assertions implicitly conveyed by them. The former are represented by *formulas* (typically denoted by X, Y, etc.) and the latter by *statements* (typically denoted by C, D, etc.) of the logic. Message extensions are used to denote the implicit contents of messages. The formula $X \rightsquigarrow C$ signals that the extension C is a precondition for conveying X.

2.2.2 Recognizability

The BAN message-meaning rules for shared and public keys assume that there is sufficient redundancy in the message space so that the decryption of a ciphertext with the correct key results in a recognizable message. The GNY logic does not assume this, and introduces a new statement $P \mid\equiv \phi(X)$, meaning P believes that X is *recognizable*, which expresses P's expectations about X before actually receiving it in a message. The logical use of this addition is made by including recognizability premises in several of the message interpretation rules of the logic [9, p. 246].

2.2.3 Honesty

The BAN logic assumes that the principals executing a protocol are *trustworthy*. This is clear from the nonce-verification rule wherein a principal who has recently said a message is assumed to believe in the message. To make the notion of trustworthiness explicit, the GNY logic introduces the statement $P \mid\equiv Q \Rightarrow Q \mid\equiv *$, which means that P believes Q to be *honest* and *competent*. The logical use of this addition is made by including honesty premises in the jurisdiction rules J2 and J3 [9, p. 240], thereby allowing the outcome of a protocol to be determined when not all principals are trustworthy.

In addition, the logic reformulates the BAN syntax to correspond more closely to the way protocol analysis is carried out in the logic. The logic also dispenses with the separate BAN notation for shared keys and shared secrets, and provides a common syntax, $P \overset{S}{\leftrightarrow} Q$, with the generic semantics that S is a suitable secret for P and Q. A complete list of the logical notation and rules of the GNY logic can be found in [9].

2.2.4 Protocol analysis

The first step in a protocol analysis using the GNY logic, like BAN, is to produce a logical description of the protocol from the conventional notation used for expressing the protocol. Unlike in BAN idealization, cleartext is retained in GNY idealized messages. This enables reasoning about possessions of principals executing the protocol. As noted earlier, any implicit information conveyed by a protocol message is represented logically by attaching an extension to the formula representing the message. In addition, for every message sent to a principal, the *not–originated–here* marker $*$ is prefixed to those parts of the message which are not included in any message sent earlier by the principal. The BAN logic assumes that principals can recognize messages which were generated by themselves. This is evident from the side condition in the BAN message-meaning rule for shared keys. The GNY not-originated-here attribute gives the logical ability to formally express this requirement for messages generated in the current run of a protocol. For example, the protocol step used earlier in section 2.1 to illustrate BAN idealization would typically be idealized in GNY as: $A \triangleleft *\{N_a, B, *K_{ab}\}_{K_{as}} \rightsquigarrow S \mid\equiv A \overset{K_{ab}}{\leftrightarrow} B$. A parser algorithm which carries out insertion of the not–originated–here marker is given in [9, p. 238].

The remaining steps to be followed during GNY analysis are similar to those in BAN. A GNY protocol analysis can also be extended to perform two consistency checks on the protocol. The *possession consistency* check requires that a principal includes only those formulae he possesses before sending a message containing any of them. The *belief consistency* check requires that a message extension include only those beliefs which are held by the sender before the message is sent. (See [18] for a detailed discussion.) The consistency checks, however, cannot be carried out within the GNY logic itself, and can be done only by relying on common knowledge about the checks. Hence, we do not attempt to include the checks while automating the logic.

3 Automating GNY logic

The inherent appeal in using BAN and GNY logics lies in their simplicity and effectiveness in analyzing cryptographic protocols. The logics can be systematically applied to understand the working of protocols. Very often a protocol analysis using the logics reveals missing assumptions or deficiencies in the protocol. This can lead to the assumptions or the original protocol being revised and the inference rules being reapplied to determine if the desired goal is then attainable. The process of applying and reapplying the inference rules, however, is often tedious and error-prone to do by hand. This has lead to the development of a number of tools for automating this task for BAN or modified versions of BAN [14, 15, 16, 17]. Such tools can also help determine the sufficiency or necessity of protocol assumptions or steps in achieving the desired goals [17]. To the best of our knowledge, no similar attempt to automate the GNY logic is reported yet. The possibility of missing inferences increases in logics like GNY, which has more than forty inference rules. Moreover, the GNY logic operates at a finer level than its predecessor, so proofs of protocol goals in the logic typically tend to be much longer than their BAN counterparts.

Our main aim in automating the logic is to be able to mechanically determine whether one or more statements describing the goal of a protocol are derivable from a given set of assumptions. On the other hand, it is also desirable to generate all statements that are derivable for a given protocol. This allows us to analyze the state of the principals after the execution of each protocol step. Such analysis is of value not only in designing optimal protocols, but also enables direct comparisons between different protocols in terms of the states attained by the principals involved. We therefore adopt a forward–chaining strategy in automating the logic. This involves repeated application of the inference rules of the logic to the set of statements consisting of the idealized protocol, initial assumptions, and derived statements, until all statements derivable are generated. However, many of the inference rules of the logic, as presented in [9], are unsuitable for forward–chaining. The problem is clear just from the freshness rule F1 [9, p. 237]:

$$\text{F1} \quad \frac{P \mid\equiv \sharp(X)}{P \mid\equiv \sharp(X, Y)},$$

which states that the freshness of a concatenated message can be deduced from the freshness of any of the concatenates[2]. Here, Y can be any formula, so one can indefinitely apply this rule to generate new beliefs in the freshness of formulas containing X.

The set of inference rules can, however, be modified in such a way that the set of statements derivable from a given protocol is finite.

3.1 Modifying the GNY postulate set

We now describe the modifications to the set of inference rules of the GNY logic, which we make in order to obtain finiteness of derivations. The modified set of rules is given in appendix A, and a proof of finiteness of derivations for this rule set is outlined in section 3.2. In addition, we add several new rules which are clearly required during protocol analyses using the logic, but which are nonetheless absent in [9].

3.1.1 Modifying existing rules

In addition to F1, several other freshness and recognizability rules cause problems by repeated application. To overcome this problem, we include an additional premise of the form $P \ni X$ in every freshness and recognizability rule with a conclusion of the form $P \mid\equiv \sharp(X)$ or $P \mid\equiv \phi(X)$. The resulting rules ensure that a principal can obtain a belief in the freshness or recognizability of a formula only if he possesses the formula. The modified freshness and recognizability rules are listed in appendices A.3 and A.4, respectively. Below we only discuss the rationale behind the modifications to the freshness rules; the recognizability rules can be dealt with similarly.

[2]The rule has one more conclusion; nonetheless, the one shown suffices to illustrate the problem.

The basic purpose in applying the logic is to reason about a principal's possessions and beliefs about others, that can be derived from messages received by the principal. Since reasoning about a principal's possessions does not depend on his beliefs, the conclusion obtained by applying a freshness rule is of no practical value if it does not affect his beliefs about others. The rule which enables a principal P to obtain beliefs from messages received by him is the jurisdiction rule J2, which has a premise of the form $P \mid\equiv Q \mid\sim (X \rightsquigarrow C)$. This premise reflects the requirement that P can only obtain beliefs from messages sent by some well–known principal Q, and appears as a conclusion of the message interpretation rules I1, I2, I3, and I4 [9, pp. 246–247]. Therefore, the statement $P \mid\equiv \sharp(X)$ is of significance in deriving P's beliefs about others only if it appears as a premise in one of these rules. Out of these rules, only I1, I2, and I3 have a freshness premise. Further, the premise set of each of these rules implies that P possesses each formula occurring in the freshness premise of the rule. We formally state and prove this property below.

> *Write $\mathcal{S} \vdash C$ to denote the derivability of statement C from a set of statements $\mathcal{S}$. Let $I \in \{\mathrm{I}_1, \mathrm{I}_2, \mathrm{I}_3\}$, and let $\mathcal{C}$ denote the premise set of I. If $P \mid\equiv \sharp(X_1, \ldots, X_m)$ is the freshness premise of I, then $\mathcal{C} \vdash P \ni X$ for every $X \in \{X_1, \ldots, X_m\}$.*

Proof: We give below the proof for I1; proofs for I2 and I3 can be carried out similarly. Note that the freshness premise in I1 is of the form $P \mid\equiv \sharp(X, K)$. Since $P \mid\equiv \sharp(X, K)$ is used in this rule to *denote* $P \mid\equiv \sharp(X)$ or $P \mid\equiv \sharp(K)$ [9, p. 245], we first replace I1 by two equivalent rules:

$$\mathrm{I1}' \quad \frac{P \triangleleft *\{X\}_K,\ P \ni K,\ P \mid\equiv P \overset{K}{\leftrightarrow} Q,\ P \mid\equiv \phi(X),\ P \mid\equiv \sharp(X)}{P \mid\equiv Q \mid\sim X,\ P \mid\equiv Q \mid\sim \{X\}_K,\ P \mid\equiv Q \ni K}$$

$$\mathrm{I1}'' \quad \frac{P \triangleleft *\{X\}_K,\ P \ni K,\ P \mid\equiv P \overset{K}{\leftrightarrow} Q,\ P \mid\equiv \phi(X),\ P \mid\equiv \sharp(K)}{P \mid\equiv Q \mid\sim X,\ P \mid\equiv Q \mid\sim \{X\}_K,\ P \mid\equiv Q \ni K}$$

It is easy to see that in both I1′ and I1″, P possesses the formula appearing in the freshness premise:

1. I1′: $P \ni X$ follows from the first two premises by applying T1, T3, and P1 respectively.
2. I1″: $P \ni K$ holds as a premise, trivially.

□

We also modify the rule R6 which states that if $P \ni H(X)$, then $P \mid\equiv \phi(X)$. This rule is unreasonable as it can give rise to unsound conclusions. For example from just $P \ni X$, we can derive $P \mid\equiv \phi(X)$ by applying rules P4 and R6. We propose the introduction of an additional premise $P \mid\equiv \phi(H(X))$ to capture the condition required for the rule's conclusion to hold.

3.1.2 Adding new rules

We add three new rules all of which enable dropping of extensions attached to formulae. These rules formalize rather trivial inferences which are obviously correct, but are nevertheless required during protocol analyses. The first two, labeled as T7 and I8 in appendix A, are found in an extension of the GNY logic [10]; the role played by these rules should be intuitively clear. Surprisingly, the rule I9 we add is absent in both [9] and [10].

$$\mathrm{I9} \quad \frac{P \mid\equiv Q \mid\sim X \rightsquigarrow (C, C')}{P \mid\equiv Q \mid\sim X \rightsquigarrow C}$$

What I10 does is to enable splitting of message extensions which are essentially conjunctions of one or more beliefs. The logical use of the conclusion of the above rule is made in the jurisdiction rule J2 where it appears as a premise. Moreover, the conclusion of J2 itself appears as a premise in J1 [9, p. 240]. Also, the rule J1 has a premise: $P \mid\equiv Q \Rightarrow C$, which reflects P's trust in Q on C. Since P may trust Q differently on C and C', the rule I9 allows us to proceed with derivation of P's beliefs from a formula conveyed by Q, containing both C and C' in the extension attached to the formula.

3.1.3 Deleting existing rules

We delete several possession rules (P2, P4, P6, P7 and P8 [9, pp. 244–245]) from the logic. Each of these rules can be applied indefinitely to produce what seem quite trivial new possessions. For example, the possession rule P6 for shared keys,

$$\text{P6} \quad \frac{P \ni K,\ P \ni X}{P \ni \{X\}_K,\ P \ni \{X\}_K^{-1}}$$

may be applied to an initial set containing $P \ni K$ and $P \ni X$ to indefinitely generate many new encryptions and decryptions. These rules are evidently useful in enforcing the possession consistency check, but of no use otherwise. As noted earlier we do not include the check in the automation, since the check is performed outside the logic.

We also delete the rationality rule, which states that if $\dfrac{C_1}{C_2}$ is a rule, then so is $\dfrac{P \models C_1}{P \models C_2}$, for any principal P. This rule of no practical value while analyzing protocols by using the GNY logic.

3.2 Finiteness of derivations

We sketch a proof, following [14], that for the modified rule set given in appendix A:

> *The statements derivable from a finite set of idealized protocol steps and initial assumptions are finite in number, and are therefore derivable in a finite number of steps.*

It is convenient to use the notation $(\mathcal{D}/E)$ to denote a generic inference rule, where $\mathcal{D}$ is the set consisting of the premises of the rule and E is the conclusion of the rule. (Rules with multiple conclusions are assumed decomposed in the obvious way into separate rules, each with a single conclusion.) Denote by $\mathcal{R}$ the modified set of rules. Then we define an operator ρ on sets of statements, as follows: for any set of statements $\mathcal{S}$,

$$\rho(\mathcal{S}) = \mathcal{S} \cup \{E :\ \text{there exists } (\mathcal{D}/E) \in \mathcal{R} \text{ such that } \mathcal{D} \subseteq \mathcal{S}\}.$$

Thus ρ returns $\mathcal{S}$ together with the statements derivable from $\mathcal{S}$ by applying the inference rules in $\mathcal{R}$ once. We outline an argument showing that there exists an n such that

$$\rho^n(\mathcal{S}) = \rho^\infty(\mathcal{S}),$$

where we use $\rho^\infty(\mathcal{S})$ to denote $\cup_{m=0}^{\infty} \rho^m(\mathcal{S})$.

The key step in the argument is the definition of a relation $\prec$ over the set of statements of the forms $P \ni X$, $P \triangleleft X$, and $P \models C$, in terms of six subsidiary relations $\prec_{\triangleleft}$, $\prec_{\ni}^{\triangleleft}$, $\prec_{\ni}$, $\prec_{\models}^{\ni}$, $\prec_{\models}^{\triangleleft}$ and $\prec_{\models}$, as follows:

$$\begin{array}{lll}
(1) & P \triangleleft X \prec P \triangleleft Y & \text{if } X \prec_{\triangleleft} Y \\
(2) & P \ni X \prec P \triangleleft Y & \text{if } X \prec_{\ni}^{\triangleleft} Y \\
(3) & P \ni X \prec P \ni Y & \text{if } X \prec_{\ni} Y \\
(4) & P \models C \prec P \ni X & \text{if } C \prec_{\models}^{\ni} X \\
(5) & P \models C \prec P \triangleleft X & \text{if } C \prec_{\models}^{\triangleleft} X \\
(6) & P \models C \prec P \models D & \text{if } C \prec_{\models} D
\end{array}$$

The definitions of the subsidiary relations are derived from suitably chosen classes of rules and are given below.

The definition of $\prec_{\triangleleft}$ is read off the *being-told* rules T1, T2, T3, T4, T5, T6 and T7 (see appendix A.1), where T2 and T5 are used in their two symmetrical forms, giving clauses as

follows:

$$\begin{array}{ll}
(1) & X \prec_{\triangleleft} *X \\
(2)(\mathrm{i}) & X \prec_{\triangleleft} (X,Y) \\
(\mathrm{ii}) & X \prec_{\triangleleft} (Y,X) \\
(3) & X \prec_{\triangleleft} \{X\}_K \\
(4) & X \prec_{\triangleleft} \{X\}_{+K} \\
(5)(\mathrm{i}) & X \prec_{\triangleleft} F(X,Y) \\
(\mathrm{ii}) & Y \prec_{\triangleleft} F(X,Y) \\
(6) & X \prec_{\triangleleft} \{X\}_{-K} \\
(7) & X \prec_{\triangleleft} X \rightsquigarrow C
\end{array}$$

The definition of $\prec^{\triangleleft}_{\ni}$ consists of a single clause which is read off the *possession* rule P1 (see appendix A.2):

$$(1) \quad X \prec^{\triangleleft}_{\ni} X$$

The definition of $\prec_{\ni}$ is read off the *possession* rules P3 and P5 (see appendix A.2), where both the rules are used in their two symmetrical forms, giving clauses as follows:

$$\begin{array}{ll}
(1)(\mathrm{i}) & X \prec_{\ni} (X,Y) \\
(\mathrm{ii}) & X \prec_{\ni} (Y,X) \\
(2)(\mathrm{i}) & X \prec_{\ni} F(X,Y) \\
(\mathrm{ii}) & X \prec_{\ni} F(Y,X)
\end{array}$$

The definition of $\prec^{\ni}_{\equiv}$ consists of two clauses. The first clause is read off each of the *freshness* rules F1′, F1″, F2′, F2″, F3′, F4′, F5′, F6′, F7′, F7″, F8′, F9′, F10′, and F11′ (see appendix A.3). The second clause is read off each of the *recognizability* rules R1′, R1″, R2′, R2″, R3′, R4′, R5′, and R6′ (see appendix A.4).

$$\begin{array}{ll}
(1) & \sharp(X) \prec^{\ni}_{\equiv} X \\
(2) & \phi(X) \prec^{\ni}_{\equiv} X
\end{array}$$

The definition of $\prec^{\triangleleft}_{\equiv}$ is read off the *message interpretation* rules I1, I2, I3, I4 and I5 (see appendix A.5), and the rules for *never-originated-here* messages I1′, I2′ and I3′ (see appendix A.7). Each rule contributes as many clauses to the definition as the number of conclusions in the rule, giving clauses as follows:

$$\begin{array}{lrll}
(1)(\mathrm{i}) & Q \mid\sim X & \prec^{\triangleleft}_{\equiv} & *\{X\}_K \rightsquigarrow C \\
(\mathrm{ii}) & Q \mid\sim \{X\}_K \rightsquigarrow C & \prec^{\triangleleft}_{\equiv} & *\{X\}_K \rightsquigarrow C \\
(\mathrm{iii}) & Q \ni K & \prec^{\triangleleft}_{\equiv} & *\{X\}_K \rightsquigarrow C \\
(2)(\mathrm{i}) & Q \mid\sim X & \prec^{\triangleleft}_{\equiv} & *\{X\}_{+K} \rightsquigarrow C \\
(\mathrm{ii}) & Q \mid\sim \{X\}_{+K} \rightsquigarrow C & \prec^{\triangleleft}_{\equiv} & *\{X\}_{+K} \rightsquigarrow C \\
(\mathrm{iii}) & Q \ni +K & \prec^{\triangleleft}_{\equiv} & *\{X\}_{+K} \rightsquigarrow C \\
(3)(\mathrm{i}) & Q \mid\sim X & \prec^{\triangleleft}_{\equiv} & *H(X) \rightsquigarrow C \\
(\mathrm{ii}) & Q \mid\sim H(X) \rightsquigarrow C & \prec^{\triangleleft}_{\equiv} & *H(X) \rightsquigarrow C \\
(4)(\mathrm{i}) & Q \mid\sim X & \prec^{\triangleleft}_{\equiv} & \{X\}_{-K} \rightsquigarrow C \\
(\mathrm{ii}) & Q \mid\sim \{X\}_{-K} \rightsquigarrow C & \prec^{\triangleleft}_{\equiv} & \{X\}_{-K} \rightsquigarrow C \\
(5)(\mathrm{i}) & Q \ni -K & \prec^{\triangleleft}_{\equiv} & \{X\}_{-K} \\
(\mathrm{ii}) & Q \ni X & \prec^{\triangleleft}_{\equiv} & \{X\}_{-K} \\
(6)(\mathrm{i}) & Q \mid\sim X & \prec^{\triangleleft}_{\equiv} & \{X\}_K \rightsquigarrow C \\
(\mathrm{ii}) & Q \mid\sim \{X\}_K \rightsquigarrow C & \prec^{\triangleleft}_{\equiv} & \{X\}_K \rightsquigarrow C \\
(7)(\mathrm{i}) & Q \mid\sim X & \prec^{\triangleleft}_{\equiv} & \{X\}_{+K} \rightsquigarrow C \\
(\mathrm{ii}) & Q \mid\sim \{X\}_{+K} \rightsquigarrow C & \prec^{\triangleleft}_{\equiv} & \{X\}_{+K} \rightsquigarrow C \\
(8)(\mathrm{i}) & Q \mid\sim X & \prec^{\triangleleft}_{\equiv} & H(X) \rightsquigarrow C \\
(\mathrm{ii}) & Q \mid\sim H(X) \rightsquigarrow C & \prec^{\triangleleft}_{\equiv} & H(X) \rightsquigarrow C
\end{array}$$

The definition of $\prec_{\equiv}$ is read off the *message interpretation* rules I6, I7, I8 and I9 (see appendix A.5), where I7 and I9 are used in their two symmetrical forms, and the *jurisdiction* rules J1, J2 and J3 (see appendix A.6), giving clauses as follows:

$$
\begin{array}{lrcl}
(1) & Q \ni X & \prec_{\equiv} & Q \mid\sim X \\
(2)(\mathrm{i}) & Q \mid\sim X & \prec_{\equiv} & Q \mid\sim (X, Y) \\
\quad (\mathrm{ii}) & Q \mid\sim X & \prec_{\equiv} & Q \mid\sim (Y, X) \\
(3) & Q \mid\sim X & \prec_{\equiv} & Q \mid\sim X \rightsquigarrow C \\
(4)(\mathrm{i}) & Q \mid\sim X \rightsquigarrow C & \prec_{\equiv} & Q \mid\sim X \rightsquigarrow (C, C') \\
\quad (\mathrm{ii}) & Q \mid\sim X \rightsquigarrow C & \prec_{\equiv} & Q \mid\sim X \rightsquigarrow (C', C) \\
(5) & C & \prec_{\equiv} & Q \mid\equiv C \\
(6) & Q \mid\equiv C & \prec_{\equiv} & Q \mid\sim (X \rightsquigarrow C) \\
(7) & Q \mid\equiv C & \prec_{\equiv} & Q \mid\equiv Q \mid\equiv C
\end{array}
$$

This completes the definitions of the six subsidiary relations. Of the six relations, the most critical in the analysis are $\prec_{\triangleleft}$, $\prec_{\ni}$ and $\prec_{\equiv}$. It is easy to see that each of these three relations is well-founded. Consider the definition of the first of the three, $\prec_{\triangleleft}$: the formula on the left in each clause is syntactically shorter than the formula on the right, so there cannot be infinite descending chains with respect to $\prec_{\triangleleft}$. Well-foundedness of $\prec_{\ni}$ and $\prec_{\equiv}$ is equally easily proved. (Well-foundedness of the other three subsidiary relations $\prec_{\ni}^{\triangleleft}$, $\prec_{\equiv}^{\ni}$ and $\prec_{\equiv}^{\triangleleft}$ is not required.)

We can now see from the definition of the relation $\prec$ above that $\prec$ is also well-founded. In fact, it is easy to see that any infinite descending chain with respect to $\prec$ must contain an infinite descending chain of statements of one of the three forms $P \ni X$, $P \triangleleft X$, or $P \mid\equiv C$. (Knowledge of the definitions of the subsidiary relations is not required for the proof.) But a chain of one of those forms must have arisen from an infinite descending chain with respect to one of the relations $\prec_{\triangleleft}$, $\prec_{\ni}$ and $\prec_{\equiv}$, and this is impossible.

A further property of all six subsidiary relations is that they are *finitary*; that is, given any statement C of the form $P \ni X$ or $P \triangleleft X$ or $P \mid\equiv C'$, the set of statements $\{D : D \prec C\}$ is finite. This is easy to see in the case of the relation $\prec_{\triangleleft}$, and is equally easily proved for the other five relations. It follows straightforwardly that $\prec$ is also finitary. Since $\prec$ is well-founded as well as finitary, it follows by König's Lemma that for any C the set of statements $\{D : D \prec^* C\}$, where $\prec^*$ denotes the transitive and reflexive closure of $\prec$, is finite as well.

Now the definitions of the subsidiary relations and of $\prec$ are constructed so as to give a straightforward guarantee that for each rule $(\mathcal{D}/E) \in \mathcal{R}$, there exists $C \in \mathcal{D}$ such that $E \prec C$; that is, in each rule the conclusion is smaller, with respect to $\prec$, than at least one of the premises. Therefore, if $\mathcal{S}$ is the set of idealized steps and initial assumptions of a protocol, and $C \in \rho^{\infty}(\mathcal{S})$, then there exists $S \in \mathcal{S}$ such that $C \prec^* S$. Hence

$$\rho^{\infty}(\mathcal{S}) \subseteq \bigcup_{S \in \mathcal{S}} \{C : C \prec^* S\},$$

and since the right-hand side here is a finite union of finite sets, given our assumption that $\mathcal{S}$ is finite, $\rho^{\infty}(\mathcal{S})$ is finite, as was required.

4 Implementing the tool

We now describe an implementation of a tool based on the modified set of rules given in appendix A. What we require to begin with is the ability to represent logical constructs in the implementation language. The tool should basically provide an inference engine which can process an input specification describing the idealized protocol and initial assumptions to produce the complete set of logical statements derivable from the specification. We also need a facility to extract proofs of the statements derived. Apart from obtaining assurance about protocol goals, such proofs enable us to determine the role played by various assumptions and protocol messages

in attaining the desired goals. We discuss below the implementation of such a tool written in Prolog. We find Prolog suitable as an implementation language as it is easy to represent the logical constructs directly by Prolog structures.

4.1 Formulas and statements

The basic building blocks of protocol messages are formulas, which typically contain constants like principal names, keys, nonces, etc. We typically represent these constants by one or more lowercase letters. For example, a session key K_{ab} for principals A and B is denoted by the Prolog atom `kab`. The remaining formulas like concatenation, encryption, functions, etc. are represented by Prolog structures chosen to represent their typographical counterparts wherever possible. We also use the structure `ext(X, nil)` to represent a formula X without any extension. The statements of the logic are similarly represented by appropriately named Prolog structures:

Formula	*Structure*
(X, Y)	`[X,Y]`
$\{X\}_K$	`encrypt(X, shared(K))`
$\{X\}_K^{-1}$	`decrypt(X, shared(K))`
$\{X\}_{+K}$	`encrypt(X, public(K))`
$\{X\}_{-K}$	`decrypt(X, private(K))`
$H(X)$	`h(X)`
$F(X_1, \ldots, X_n)$	`f(X1, ..., Xn)`
$*X$	`star(X)`
$X \rightsquigarrow C$	`ext(X, C)`
X	`ext(X, nil)`

Statement	*Structure*
$P \triangleleft X$	`told(P, X)`
$P \ni X$	`possesses(P, X)`
$P \mid\sim X$	`conveyed(P, X)`
$P \models \sharp(X)$	`believes(P, fresh(X))`
$P \models \phi(X)$	`believes(P, recognizes(X))`
$P \models Q \overset{S}{\leftrightarrow} R$	`believes(P, secret(Q, S, R))`
$P \models \overset{+K}{\mapsto} Q$	`believes(P, public(K, Q))`
$P \models C$	`believes(P, C)`
$P \models Q \Rightarrow C$	`believes(P, controls(Q, C))`
$P \models Q \Rightarrow Q \models *$	`believes(P, honest(Q))`
C_1, C_2	`[C1, C2]`

The translations given in the above tables enable us to represent any formula or statement in the GNY syntax by its Prolog counterpart. For example, the idealized protocol step given earlier in section 2.2.4 is typically represented by the Prolog structure:

```
told(a, ext(star(encrypt([na, b, star(kab)], kas), believes(s, secret(a, kab, b)))))
```

4.1.1 Derived statements

Apart from the ability to represent logical constructs in Prolog syntax, we also need to maintain derivation information of statements obtained by applying the inference rules. The predicate `fact/3` which defines an inference step is used for this purpose. It takes the form: `fact(Index, Stat, reason(PremIs, Rule))`. The integer argument `Index` is used to index instances of `fact/3`. The second argument `Stat` is bound to a derived statement. In the last argument, `PremIs` is a list containing the indices of premises used in deriving `Stat` by an application of rule `Rule`. For uniformity, the statements describing the idealized protocol and assumptions are also represented as instances of `fact/3`. The list `PremIs` is empty if `Stat` represents either an idealized step or assumption, and `Rule` is then `'Step'` or `'Assumption'`.

4.2 Inference engine

To begin analyzing a given protocol, the instances of `fact/3` representing the idealized protocol and assumptions are loaded from an input file. Forward-chaining is then performed by a predicate `forward/1`, which repeatedly applies all inference rules until no new statements can be derived:

```
forward(Cycle) :-
    Cycle > 0,
    done,
```

```
    write('Analyzed in '), write(Cycle), write(' cycles'), nl.

forward(Cycle) :-
    apply_rules,
    NextCycle is Cycle + 1,
    forward(NextCycle).
```

The goal `done` succeeds when no new statements can be derived. The new statements that can be derived from the statements stored in the database are generated by the goal `apply_rules`, which invokes the inference rules of the logic.

4.2.1 Logical rules

The representation of the inference rules is best explained by means of an example — the being–told rule T1 is defined by the following clause for `told/2`:

```
told(told(P, X), reason([I], 'T1')) :-
        fact(I, told(P, star(X)), _).
```

4.3 Proof Explanation

A stepwise explanation of the inferences leading to a derived statement can be obtained by invoking the goal `explain_proof(Stat)`. This goal calls a predicate `proof/2` to build a list containing indices of facts used in deriving `Stat`.

```
proof(Index, ProofDup) :-
    fact(Index, _, reason(Premises, _)),
    append(Premises, ProofOfPremises, ProofDup),
    prooflist(Premises, ProofOfPremises).

prooflist([], []).
prooflist([Premise|Premises], TotalProof) :-
    proof(Premise, ProofOfPremise),
    prooflist(Premises, ProofOfPremises),
    append(ProofOfPremise, ProofOfPremises, TotalProof).
```

The proof list returned by `proof/2` is then further processed to generate a formatted proof. An example proof is shown in the next section.

5 Using the tool

We now demonstrate the use of the tool by analyzing a voting protocol. This protocol was analyzed in [9], and we use it to illustrate how the tool helped us in detecting a problem with the protocol parsing scheme therein. The idealized version of the protocol produced by the GNY parser is [9, p. 239] [3]:

1. $P_i \triangleleft *N_q$
2. $Q \triangleleft *P_i, *N_i, *V_i, *H(N_q, *<S_i>, V_i)$
3. $P_i \triangleleft *R, *H(N_i, <S_i>, R)$

Here S_i is a secret shared by P_i and Q only, and N_i and N_q are nonces generated by P_i and Q respectively. The following assumptions are presumed to hold at the beginning of the protocol:

$$P_i \ni S_i; \quad P_i \ni N_i; \quad P_i \models Q \overset{S_i}{\leftrightarrow} P_i; \quad P_i \models \sharp(N_i)$$
$$Q \ni S_i; \quad Q \ni N_q; \quad Q \models Q \overset{S_i}{\leftrightarrow} P_i; \quad Q \models \sharp(N_q)$$

It is straightforward to convert the idealized protocol and assumptions given above into Prolog syntax by using the translations defined in section 4.1. For example, we define the clause:

[3] $<S>$ is used as a mnemonic aid for distinguishing a secret S used for identification purposes [9, p. 246].

```
fact(2, told(q, [star(pi), star(ni), star(vi), ext(star(h([nq, star(si), vi])), nil)]),
reason([], 'Step')).
```

to represent step 2 of the idealized protocol. The set of facts representing the idealized protocol and the initial assumptions is then loaded into the analyzer to generate all the logical statements derivable:

```
| ?- analyze(voting).
Analyzed in 4 cycles
```

The database of generated facts can now simply be queried to determine whether a particular goal statement is attained or not. For example, the statements $Q \mid\equiv P_i \mid\sim V_i$ and $P_i \mid\equiv Q \mid\sim R$ are expected to hold at the end of the protocol [9, p. 239]. The following queries can be used to see whether the protocol attains these statements:

```
| ?- fact(I, believes(q, conveyed(pi, vi)), Rule).
no
| ?- fact(I, believes(pi, conveyed(q, r)), Rule).
I = 37
Rule = reason([33],I7);
no
```

The output of the above queries show that $Q \mid\equiv P_i \mid\sim V_i$ is not attained whereas $P_i \mid\equiv Q \mid\sim R$ is. This result contradicts the manual analysis carried out by GNY [9, p. 239] who are able to derive both these statements. We now explain the discrepancy behind this.

5.1 Modified parsing scheme

Note that GNY are basically led to the conclusion that $Q \mid\equiv P_i \mid\sim V_i$ by applying the message interpretation rule I3 to the second message. Since $Q \triangleleft *H(N_q, *<S_i>, V_i)$ (from step 2 by T2), the intended application of I3 is:[4]

$$\frac{Q \triangleleft *H(N_q, V_i, *<S_i>),\ \ Q \ni (N_q, V_i, *S_i),\ \ Q \mid\equiv Q \overset{*S_i}{\leftrightarrow} P_i,\ \ Q \mid\equiv \sharp(N_q, V_i, *S_i)}{Q \mid\equiv P_i \mid\sim (N_q, V_i, *<S_i>)}$$

The failure of the tool to draw the desired conclusion is now obvious. In all of the premises above, the secret S_i appears prefixed with a not–originated–here marker. For example, the third premise is $Q \mid\equiv Q \overset{*S_i}{\leftrightarrow} P_i$. We do have an assumption $Q \mid\equiv Q \overset{S_i}{\leftrightarrow} P_i$, but there is no way we can transform it into $Q \mid\equiv Q \overset{*S_i}{\leftrightarrow} P_i$ by using the logic, although GNY overlook this in their analysis. In [10], the protocol parsing scheme is modified in such a way that the insertion of the not–originated–here marker is possibly carried out only for formulas which are encryptions or decryptions or hash functions. The above problem would not arise if this modified parsing scheme is used. It is easy to see that in order to derive a principal's possessions and beliefs about others from messages received by him, it suffices to consider only encrypted or hashed formulas for insertion of $*$'s. Note that the only message interpretation rules with a $*$ prefixed to a told formula appearing in a premise are I1, I2, and I3. In each of these rules, since the formula to which $*$ is prefixed is either encrypted or hashed, we need consider only such formulas while inserting $*$'s. We therefore adopt the modified parsing scheme of [10] while analyzing protocols using the logic.

The desired statement is immediately derived once we alter the idealization of step 2 to reflect this:

```
fact(2, told(q, [pi, ni, vi, ext(star(h([nq, si, vi])), nil)]), reason([], 'Step')).
```

[4] The rule has one more conclusion which need not concern us here.

```
| ?- fact(I, believes(q, conveyed(pi, vi)), Rule).
I = 37
Rule = reason([31],I7);
no
```

The proof explanation facility can be used to obtain proofs of derived statements. For example, we obtain the following machine-generated proof of $Q \mid\equiv P_i \mid\sim V_i$:

```
| ?- explain_proof(believes(q, conveyed(pi, vi))).
1. told(q,[pi,ni,vi,ext(star(h([nq,si,vi])),nil)])  {Step}
2. told(q,vi)  {1, T2}
3. possesses(q,vi)  {2, P1}
4. possesses(q,si)  {Assumption}
5. possesses(q,nq)  {Assumption}
6. believes(q,fresh(nq))  {Assumption}
7. believes(q,secret(q,si,pi))  {Assumption}
8. told(q,ext(star(h([nq,si,vi])),nil))  {1, T2}
9. believes(q,conveyed(pi,[nq,si,vi]))  {8, 7, 6, 5, 4, 3, I3}
10. believes(q,conveyed(pi,vi))  {9, I7}
```

Each step of the proof is output on a new line and contains either an assumption or a conclusion drawn from statements in earlier lines. The inference rule and the line numbers of the premises used in deriving a statement are listed to the right of the statement.

We have further used the tool to carry out machine-aided GNY logic analysis of several other protocols given in [5, 9]. The tool has proved useful in mechanically verifying the need for the rules T7, I8, and I9, which are all absent from the original GNY logic.

References

[1] R. M. Needham and M. D. Schroeder, "Using Encryption for Authentication in Large Networks of Computers," *Communications of the ACM*, vol. 21, pp. 993–999, Dec. 1978.

[2] J. H. Moore, "Protocol Failures in Cryptosystems," *Proc. IEEE*, vol. 76, pp. 594–602, May 1988.

[3] G. J. Simmons, "Cryptanalysis and Protocol Failures," in *Proc. First ACM Conference on Computer and Communications Security*, Nov. 1993. Invited talk.

[4] D. E. Denning and G. M. Sacco, "Timestamps in Key Distribution Protocols," *Communications of the ACM*, vol. 24, pp. 533–536, Aug. 1981.

[5] M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication," Tech. Rep. 39, Systems Research Center, Digital Equipment Corporation, Palo Alto, California, Feb. 1990.

[6] M. Abadi and R. Needham, "Prudent Engineering Practice for Cryptographic Protocols," in *Proc. IEEE Symposium on Security and Privacy*, (Los Alamitos, California), pp. 122–136, IEEE Computer Society Press, may 1994.

[7] M. Burrows, M. Abadi, and R. Needham, "A Logic of Authentication," *ACM Trans. on Computer Systems*, vol. 8, pp. 18–36, Feb. 1990.

[8] L. Gong, "Using One-Way Functions for Authentication," *ACM Computer Communication Review*, vol. 19, pp. 8–11, Oct. 1989.

[9] L. Gong, R. Needham, and R. Yahalom, "Reasoning about Belief in Cryptographic Protocols," in *Proc. IEEE Symposium on Security and Privacy*, (Los Alamitos, California), pp. 234–248, IEEE Computer Society Press, 1990.

[10] L. Gong, *Cryptographic Protocols for Distributed Systems*. PhD thesis, Cambridge University, U.K., 1990.

[11] R. Kailar and V. D. Gligor, "On Belief Evolution in Authentication Protocols," in *Proc. IEEE Computer Security Foundations Workshop IV*, (Los Alamitos, California), pp. 103–116, IEEE Computer Society Press, 1991.

[12] K. Gaarder and E. Snekkenes, "Applying a Formal Analysis Technique to the CCITT X.509 Strong Two-Way Authentication Protocol," *Journal of Cryptology*, vol. 3, pp. 81–98, 1991.

[13] P. C. V. Oorschot, "Extending Cryptographic Logics of Belief to Key Agreement Protocols (Extended Abstract)," in *Proc. First ACM Conference on Computer and Communications Security*, pp. 232–243, Nov. 1993.

[14] U. Engberg, "Analyzing Authentication Protocols," Tech. Rep. TR DAIMI IR-97, Aarhus University, Denmark, 1990.

[15] E. A. Campbell and R. Safavi-Naini, "On Automating The BAN Logic of Authentication," in *Proc. 15th Australian Computer Science Conference (ACSC-15)*, 1992.

[16] R. C. Hauser and E. S. Lee, "Verification and Modelling of Authentication Protocols," in *Computer Security - ESORICS 92* (Y. Deswarte, G. Eizenberg, and J.-J. Quisquater, eds.), no. 648 in Lecture Notes in Computer Science, pp. 141–154, Springer-Verlag, 1992.

[17] A. Mathuria, R. Safavi-Naini, and P. Nickolas, "Exploring Minimal BAN Logic Proofs of Authentication Protocols," in *Proc. 10th International Conference on Information Security (IFIP SEC'94)*, May 1994. (To appear).

[18] L. Gong, "Handling Infeasible Specifications of Cryptographic Protocols," in *Proc. of IEEE Symposium on Security and Privacy*, (Los Alamitos, California), pp. 99–102, IEEE Computer Society Press, 1991.

A Modified GNY rules

A.1 Being-Told Rules

T1 $\dfrac{P \triangleleft *X}{P \triangleleft X}$

T2 $\dfrac{P \triangleleft (X,Y)}{P \triangleleft X}$

T3 $\dfrac{P \triangleleft \{X\}_K,\ P \ni K}{P \triangleleft X}$

T4 $\dfrac{P \triangleleft \{X\}_{+K},\ P \ni -K}{P \triangleleft X}$

T5 $\dfrac{P \triangleleft F(X,Y),\ P \ni X}{P \triangleleft Y}$

T6 $\dfrac{P \triangleleft \{X\}_{-K},\ P \ni +K}{P \triangleleft X}$

T7 $\dfrac{P \triangleleft X \rightsquigarrow C}{P \triangleleft X}$

A.2 Possession Rules

P1 $\dfrac{P \triangleleft X}{P \ni X}$

P3 $\dfrac{P \ni (X,Y)}{P \ni X}$

P5 $\dfrac{P \ni F(X,Y),\ P \ni X}{P \ni Y}$

A.3 Freshness Rules

F1′ $\dfrac{P \models \sharp(X),\ P \ni (X,Y)}{P \models \sharp(X,Y)}$

F1″ $\dfrac{P \models \sharp(X),\ P \ni F(X)}{P \models \sharp(F(X))}$

F2′ $\dfrac{P \models \sharp(X),\ P \ni K,\ P \ni \{X\}_K}{P \models \sharp(\{X\}_K)}$

F2″ $\dfrac{P \models \sharp(X),\ P \ni K,\ P \ni \{X\}_K^{-1}}{P \models \sharp(\{X\}_K^{-1})}$

F3′ $\dfrac{P \models \sharp(X),\ P \ni +K,\ P \ni \{X\}_{+K}}{P \models \sharp(\{X\}_{+K})}$

F4′ $\dfrac{P \models \sharp(X),\ P \ni -K,\ P \ni \{X\}_{-K}}{P \models \sharp(\{X\}_{-K})}$

F5′ $\dfrac{P \models \sharp(+K),\ P \ni -K}{P \models \sharp(-K)}$

F6′ $\dfrac{P \models \sharp(-K),\ P \ni +K}{P \models \sharp(+K)}$

F7′ $\dfrac{P \models \phi(X),\ P \models \sharp(K),\ P \ni K,\ P \ni \{X\}_K}{P \models \sharp(\{X\}_K)}$

F7″ $\dfrac{P \models \phi(X),\ P \models \sharp(K),\ P \ni K,\ P \ni \{X\}_K^{-1}}{P \models \sharp(\{X\}_K^{-1})}$

F8′ $\dfrac{P \models \phi(X),\ P \models \sharp(+K),\ P \ni +K,\ P \ni \{X\}_{+K}}{P \models \sharp(\{X\}_{+K})}$

F9′ $\dfrac{P \models \phi(X),\ P \models \sharp(-K),\ P \ni -K,\ P \ni \{X\}_{-K}}{P \models \sharp(\{X\}_{-K})}$

F10′ $\dfrac{P \models \sharp(X),\ P \ni X,\ P \ni H(X)}{P \models \sharp(H(X))}$

F11′ $\dfrac{P \models \sharp(H(X)),\ P \ni H(X),\ P \ni X}{P \models \sharp(X)}$

A.4 Recognizability Rules

R1′ $\dfrac{P \models \phi(X),\ P \ni (X,Y)}{P \models \phi(X,Y)}$

R1″ $\dfrac{P \models \phi(X),\ P \ni F(X)}{P \models \phi(F(X))}$

R2′ $\dfrac{P \models \phi(X),\ P \ni K,\ P \ni \{X\}_K}{P \models \phi(\{X\}_K)}$

R2″ $\dfrac{P \models \phi(X),\ P \ni K,\ P \ni \{X\}_K^{-1}}{P \models \phi(\{X\}_K^{-1})}$

R3′ $\dfrac{P \models \phi(X),\ P \ni +K,\ P \ni \{X\}_{+K}}{P \models \phi(\{X\}_{+K})}$

R4′ $\dfrac{P \models \phi(X),\ P \ni -K,\ P \ni \{X\}_{-K}}{P \models \phi(\{X\}_{-K})}$

R5′ $\dfrac{P \models \phi(X),\ P \ni X,\ P \ni H(X)}{P \models \phi(H(X))}$

R6′ $\dfrac{P \ni H(X),\ P \models \phi(H(X)),\ P \ni X}{P \models \phi(X)}$

A.5 Message Interpretation Rules

I1 $$\frac{P \triangleleft *\{X\}_K \rightsquigarrow C,\ P \ni K,\ P \models P \overset{K}{\leftrightarrow} Q,\ P \models \phi(X),\ P \models \sharp(X,K)}{P \models Q \mid\sim X,\ P \models Q \mid\sim \{X\}_K \rightsquigarrow C,\ P \models Q \ni K}$$

I2 $$\frac{P \triangleleft *\{X,<S>\}_{+K} \rightsquigarrow C,\ P \ni (-K,S),\ P \models \overset{+K}{\mapsto} P,\ P \models P \overset{S}{\leftrightarrow} Q,\ P \models \phi(X,S),\ P \models \sharp(X,S,+K)}{P \models Q \mid\sim (X,<S>),\ P \models Q \mid\sim \{X,<S>\}_{+K} \rightsquigarrow C,\ P \models Q \ni +K}$$

I3 $$\frac{P \triangleleft *H(X,<S>) \rightsquigarrow C,\ P \ni (X,S),\ P \models P \overset{S}{\leftrightarrow} Q,\ P \models \sharp(X,S)}{P \models Q \mid\sim (X,<S>),\ P \models Q \mid\sim H(X,<S>) \rightsquigarrow C}$$

I4 $$\frac{P \triangleleft \{X\}_{-K} \rightsquigarrow C,\ P \ni +K,\ P \models \overset{+K}{\mapsto} Q,\ P \models \phi(X)}{P \models Q \mid\sim X,\ P \models Q \mid\sim \{X\}_{-K} \rightsquigarrow C}$$

I5 $$\frac{P \triangleleft \{X\}_{-K},\ P \ni +K,\ P \models \overset{+K}{\mapsto} Q,\ P \models \phi(X),\ P \models \sharp(X,+K)}{P \models Q \ni (-K,X)}$$

I6 $$\frac{P \models Q \mid\sim X,\ P \models \sharp(X)}{P \models Q \ni X}$$

I7 $$\frac{P \models Q \mid\sim (X,Y)}{P \models Q \mid\sim X}$$

I8 $$\frac{P \models Q \mid\sim X \rightsquigarrow C}{P \models Q \mid\sim X}$$

I9 $$\frac{P \models Q \mid\sim X \rightsquigarrow (C,C')}{P \models Q \mid\sim X \rightsquigarrow C}$$

A.6 Jurisdiction Rules

J1 $$\frac{P \models Q \Rightarrow C,\ P \models Q \models C}{P \models C}$$

J2 $$\frac{P \models Q \Rightarrow Q \models *,\ P \models Q \mid\sim (X \rightsquigarrow C),\ P \models \sharp(X)}{P \models Q \models C}$$

J3 $$\frac{P \models Q \Rightarrow Q \models *,\ P \models Q \models Q \models C}{P \models Q \models C}$$

A.7 Never-Originated-Here Messages

I1′ $$\frac{P \triangleleft \{X\}_K \rightsquigarrow C,\ P \ni K,\ P \models P \overset{K}{\leftrightarrow} Q,\ P \models \phi(X),\ P \models \otimes(P)}{P \models Q \mid\sim X,\ P \models Q \mid\sim \{X\}_K \rightsquigarrow C}$$

I2′ $$\frac{P \triangleleft \{X,<S>\}_{+K} \rightsquigarrow C,\ P \ni (S,-K),\ P \models \overset{+K}{\mapsto} P,\ P \models P \overset{S}{\leftrightarrow} Q,\ P \models \phi(X,S),\ P \models \otimes(P)}{P \models Q \mid\sim (X,<S>),\ P \models Q \mid\sim \{X,<S>\}_{+K} \rightsquigarrow C}$$

I3′ $$\frac{P \triangleleft H(X,<S>) \rightsquigarrow C,\ P \ni (X,S),\ P \models P \overset{S}{\leftrightarrow} Q,\ P \models \phi(X,S),\ P \models \otimes(P)}{P \models Q \mid\sim (X,<S>),\ P \models Q \mid\sim H(X,<S>) \rightsquigarrow C}$$