Decision Procedures for the Analysis of Cryptographic Protocols by Logics of Belief*

David MONNIAUX
Laboratoire d'Informatique de l'École Normale Supérieure †
&
SRI International, Computer Science Laboratory ‡

Abstract

Belief-logic deductions are used in the analysis of cryptographic protocols. We show a new method to decide such logics. In addition to the familiar BAN logic, it is also applicable to the more advanced versions of protocol security logics, and GNY in particular; and it employs an efficient forward-chaining algorithm the completeness and termination of which are proved. Theoretic proofs, implementation decisions and results are discussed.

1 Introduction

Cryptographic protocols are specifications for sequences of messages to be exchanged by machines on a possibly insecure network, such as the Internet, to establish private or authenticated communication. These protocols can be used to distribute sensitive information, such as classified material, credit card numbers or trade secrets, or to authenticate information.

Many cryptographic protocols have been found to be flawed; that is, there exists a way for an intruder that has gained partial or total control over the communication network and is able to read, suppress and forge messages to trick the communicating machines into revealing some sensitive information or believing they have an authenticated communication, whereas they are actually communicating with the intruder. Several tools and techniques have therefore been devised for analyzing and verifying the security of cryptographic protocols.

A common feature of these techniques, including ours, is that they address the design of the protocol rather than the strength of the underlying cryptographic algorithms, such as message digests or encryption primitives. For instance, it is assumed that one may decrypt a message encrypted with a public key only when possessing the corresponding private key.

Three main categories of approaches have been proposed:

Belief logics represent reasoning such as "if a message arrives encrypted with a key known only to me and machine M, and I did not send it originally, then it must have been sent by M". The first of these was the so-called BAN logic from Burrows, Abadi and Needham [3], which was followed by more expressive and elaborate extensions such as GNY (Gong, Needham and Yahalom [6, 5]) and SVO (Syverson and van Oorschot [17, 18]). One limitation of these logics is the need to annotate the protocols with logical assertions that are assumed to represent the intent of the sender of the message, as well as logical assumptions on the secrecy or freshness of certain pieces of information. Also, they cannot verify secrecy.

Partial model checking of a semantic model is a stronger method. Here a limited but wide set of possible attacks is generated and systematically tried against the protocol. The hope is that this set is wide enough so that any attack will be detected. Efficient implementations have been devised [10, 13, 9].

Theorem proving of properties of a semantic model is the strongest method. Formal proofs are sought for safety theorems on the protocols (in a certain model of what a protocol is). There are two approaches:

- using a generic proof assistant with specific procedures [14];
- using a specific analyzer like the NRL Protocol Analyzer [11].

Of course, a clear definition of the model that is considered is needed, and subtle points can have dramatic

*This work was partially funded by NSF grant CCR-9509931
†ENS-LIENS, 45, rue d'Ulm , 75230 PARIS cédex 5, FRANCE
‡SRI-CSL, 333 Ravenswood Ave., Menlo Park, CA 94025-3493, USA

$$\dfrac{\dfrac{A \triangleleft \star \{N_a\}_{-K_b}\,(2) \quad A \ni +K_b\,(a) \quad A \models \overset{+K_b}{\mapsto} K_b\,(b) \quad A \models \phi(N_a)\,(c)}{A \models B \mid\sim N_a}\ \mathsf{I4} \quad A \models \sharp(N_a)\,(d)}{A \models B \ni N_a}\ \mathsf{I6}$$

Figure 1. A derivation in GNY logic.

effects on whether some methods are suitable or not, as shown in [7]. More specifically, the rewriting systems that very often quotient the space of messages have to be taken into account in the design of the proof method [7, 12]. Rewriting systems also have to be taken into account in analyses by GNY logic and will be dealt with in this paper.

The three approaches have increasing strength, but also roughly increasing cost, human-directed theorem proving being very costly in human time and model checking in computation times. All approaches can be effectively used in combination, starting with belief logics, which, although the weakest, are simple to use and can be very effective in detecting certain flaws. Their simplicity and low cost would be enhanced if their analysis could be automated with complete reliability; this is such a complete, provably reliable automation that we are presenting in this paper.

The purpose of this paper is to present a new method for automating belief-logic deductions. It is applicable to the more advanced versions of protocol security logics, and GNY in particular; and it employs an efficient forward-chaining algorithm the completeness and termination of which are proved.

There have been other efforts to automate belief-logic security analyses [15, 8, 1, 2]. Their approaches have been less satisfactory since they were either incomplete or nonterminating [15] or limited to BAN logic [8]; completeness and nontermination have sometimes not been explicitly addressed [1, 2]. Our method is complete, meaning that if a formula is provable from a set of hypotheses in a logic where our method is applicable, our method will find a suitable derivation (proof tree); and it is terminating, meaning that when the formula is not provable, our method will stop and give a negative answer. Also, our method yields results in a matter of seconds, comparable to the times of the incomplete procedure of [15].

1.1 A simple example of belief-logic analysis

1	$A \rightarrow B$	N_a
2	$B \rightarrow A$	$\{N_a\}_{-K_b}$
3	$A \rightarrow B$	$\{K_{ab}\}_{+K_b}$

This is to be understood as: in step 1, A sends a newly generated number N_a to B; B answers with the encryption of N_a by its private key $-K_b$; A answers with the encryption of a newly generated session key K_{ab} with B's public key $+K_b$.

To illustrate how belief-logic deductions work, we first show the "idealized" version of the protocol in GNY:

1	$B \triangleleft N_a$
2	$A \triangleleft \star \{N_a\}_{-K_b}$
3	$B \triangleleft \star \left(\{K_{ab}\}_{+K_b} \rightsquigarrow A \overset{K_{ab}}{\longleftrightarrow} B \right)$

The star means that the following term was not originated by the party who receives it. The statement after the wavy arrow in 3 is an annotation meaning that K_{ab} is intended to be a shared secret key for use between A and B.

We also need some assumptions, written as follows in GNY:

(a) $A \ni +K_b$ (A possesses $+K_b$)

(b) $A \models \overset{+B}{\mapsto} B$ (A believes that $+K_b$ is B's public key)

(c) $A \models \phi(N_a)$ (A believes N_a to be recognizable; that is, if A sees a message field that is supposed to be N_a, A can check whether it is or not)

(d) $A \models \sharp(N_a)$ (A believes N_a to be fresh; that is, to have been used for the first time in this run of the protocol)

Using GNY logic, we can derive the conclusion $A \models B \ni N_a$ (see fig. 1). This means that from the fact, coming from protocol step 2, that A sees the message consisting of the encryption of N_a by the private key for the public/private couple of keys K_b, from assumption a (A possesses the corresponding public key), from assumption b (B uses the private key of the couple K_b) and from assumption c (A believes that it can recognize N_a), we deduce using rule I4 that A is entitled to believe that B once said N_a. Then, using assumption d, which is that N_a is fresh (has never been used in another session before), we deduce that A is entitled to believe that B possesses N_a. See [6] for a complete list of the GNY inference rules and their designations.

The final goal of the protocol might be to cause B to believe that A believes that K_{ab} is the shared key ($A \models A \overset{K_{ab}}{\longleftrightarrow} B$). However, message 3 could have been forged by any intruder possessing $+K_b$, which is realistic since it is a public key, replacing K_{ab} by any key of his choice. The

logic (correctly) fails to conclude that the protocol accomplishes this goal: this goal has no derivation in GNY logic from the above set of hypotheses.

2 Our method

The purpose of our method is, given a set of hypotheses Γ and a would-be conclusion t, to decide whether t is derivable from Γ in GNY logic. Additionally, if it is derivable, a derivation is output; if it is not, an attempt is made to suggest some additional hypotheses to the user.

2.1 Vocabulary

We place ourselves in a free algebra of terms. By a "rule", we mean a finite set of terms with variables $\{\mathcal{H}_1, \ldots, \mathcal{H}_n\}$ called the premisses and a term with variables $\mathcal{C}$ call the conclusion, traditionally represented as

$$\frac{\mathcal{H}_1 \quad \cdots \quad \mathcal{H}_n}{\mathcal{C}}$$

Applying that rule to hypotheses $H_1, \ldots, H_n$ to yield the conclusion C means finding a common unifier for $\{(H_1, \mathcal{H}_1), \ldots, (H_n, \mathcal{H}_n), (C, \mathcal{C})\}$. A derivation is a tree whose nodes are such applications, with its root the conclusion and its leaves the hypotheses.

There are two traditional methods to test whether a formula t admits a derivation from a set of hypotheses Γ in a system of rules $\vdash$ (which we note by $\boldsymbol{\Gamma \vdash t}$):

Forward chaining, that is starting from the hypotheses Γ, apply all the possible deduction rules to deduce new formulas, then start again with the union of the hypotheses and the new formulas, until the formula t is discovered;

Backward chaining, that is, starting from the purported conclusion, find all the rules and all the instantiations of the variables in them that yield that conclusion, then try recursively to prove the hypotheses of each of these rules with each of the instantiations; it is a kind of search with backtracking.

Both these methods suffer from the following problems:

- Both methods may never terminate if $\Gamma \nvdash t$.
- Some rules maybe unsuitable for a given method: for a forward (resp. backward) chaining method, if some variables in the conclusion (resp. premisses) are not found in the premisses (resp. conclusion), then the system has to instantiate them with all possible terms. In the opposite case, we say that the rule is **suitable for forward (resp. backward) chaining**.

There are two problems with the systems of rules like GNY:

- There exist infinite derivations;
- There are rules that are unsuitable for forward-chaining and rules that are unsuitable for forward-chaining.

Nevertheless, we will give a characterization of a class of systems of rules, to which GNY belongs, that are equivalent to systems suitable for forward-chaining; moreover, we have a convenient criterion ensuring termination.

2.2 The transformation

If we straightforwardly (and naively) implement the rules of BAN or GNY logic in a general-purpose automatic theorem prover, as it has been done [15], the prover is likely to search an infinite space of possible proofs, which means that the system doesn't terminate when the conclusion is not provable. Furthermore, even in cases of termination, the computation time might be prohibitive, due to lots of unfruitful search.

Our idea is that, given a logic of belief for authentication (abiding by certain restrictions given in the definitions below), we can construct another logic in which for any Γ and t the search space in which to look for a derivation of t from the set of hypotheses Γ is finite. The implementation just has to enumerate exhaustively this search space and test whether it contains such a derivation.

2.2.1 Classes of rules and normal derivations

We deal with two classes of rules[1]:

- **composition rules**, in which all the variables of the premisses are found in the conclusion (these rules are suitable for backward-chaining);
- **decomposition rules**, in which all the variables of the conclusion are found in the premisses (these rules are suitable for forward-chaining); for these rules, we distinguish the **principal premisses** (which can be one or more) and the optional **auxiliary premisses**; the variables in the auxiliary premisses are a subset of these in the principal premisses.

We consider a partition of the rules between these two classes. The intuition is that composition rules introduce

[1] This partition of the set of rules into two classes is very similar to that of [8], where they are called respectively *growing* and *shrinking rules*. This is also similar to the introduction and elimination rules of intuitionistic logic; see [4, p. 75]. Our theorem on normal derivations is thus similar to the normalization theorem of intuitionistic logic or Gentzen's Hauptsatz [4, p. 105].

constructors, like a pair, and decomposition rules break these constructors, as in taking the first projection of a pair.

Of course, several partitions abiding by the above criteria can exist; the choice for the partition that is implemented has to take into account issues like termination and complexity. For termination purposes, one may use a well-founded ordering so that for any decomposition rule, there exists at least one premiss so that the conclusion is strictly less than that premiss.

Informally, a decision procedure would have to backward-chain on the composition rules and to forward-chain on the decomposition rules.

We define the following criterion (called the **normal derivation criterion**) on systems of rules: we require that if there exists a derivation of $\Gamma \vdash t$ then there must also exist a **normal derivation** of $\Gamma \vdash t$. A derivation Δ of a conclusion $\Gamma \vdash t$ is said to be **normal** if there is no composition rule to be used as the root rule of the sub-derivation for a principal premise of a decomposition rule: for any decomposition rule used in Δ:

$$\frac{\begin{array}{cccccc}\vdots & & \vdots & & \vdots & \\ \overline{P_1}\ r_1 & \cdots & \overline{P_n}\ r_n & A_1 & \cdots & A_m\end{array}}{C}\ r$$

where the P_i are the principal hypotheses and the A_j the auxiliary hypotheses, none of the rules $r_1, \ldots, r_n$ is a composition rule. In the opposite case, we say that there is a **cut** at r.

Informally, that means that it must be possible to derive anything that is derivable without having to compose something and decompose it afterwards; for instance, in

$$\dfrac{\dfrac{\begin{array}{cc}\vdots\ \alpha & \vdots\ \beta \\ P \ni X & P \ni Y\end{array}}{P \ni (X,Y)}\ \mathsf{P2}}{P \ni X}\ \mathsf{P3}$$

we compose a pair just to decompose it afterwards. This is a like a cut in logic and poses difficulty for automatic proof search because it requires invention (in the above example, the system would have had to guess a suitable value for Y).[2] Of course, the above derivation can be replaced by the following one:

$$\begin{array}{c}\vdots\ \alpha \\ P \ni X\end{array}$$

The possibility of such replacements is what we express in our criterion.

We will only deal with systems matching the normal derivation criterion. GNY logic doesn't match the criterion, that's why we'll have to supplement it with a few rules, yielding an equivalent system that matches the criterion.

2.2.2 The "goal" symbol

We introduce a special symbol, **goal**. $\boxdot X$ means that "we would like to compose X". We use it to internalize our decision strategy, which is that composition rules will only be applied when their conclusion is to be composed.

Our transformation turns the composition rule

$$\frac{\mathcal{H}_1 \cdots \mathcal{H}_n}{\mathcal{C}}$$

into a pair

$$\frac{\boxdot\mathcal{C} \quad \mathcal{H}_1 \cdots \mathcal{H}_n}{\mathcal{C}} \qquad \frac{\boxdot\mathcal{C}}{\boxdot\mathcal{H}_1 \cdots \boxdot \mathcal{H}_n}$$

and adds to the decomposition rule

$$\frac{\mathcal{D}_1 \cdots \mathcal{D}_m \quad \mathcal{K}_1 \cdots \mathcal{K}_n}{\mathcal{C}},$$

where the one or more $\mathcal{D}_i$ are principal premisses and the zero or more $\mathcal{K}_i$ are auxiliary premisses, the triggering rule

$$\frac{\mathcal{D}_1 \cdots \mathcal{D}_m}{\boxdot\mathcal{K}_1 \cdots \boxdot \mathcal{K}_n}.$$

Theorem 1 *For all system of rules $\vdash$ matching the conditions for the above transformation, calling $\vdash'$ the system of rules resulting from the transformation of $\vdash$, we have: for all set of hypotheses Γ and formula t,*

$$\Gamma \vdash t \iff \Gamma, \boxdot t \vdash' t;$$

furthermore, in that case a derivation of $\Gamma, \boxdot t \vdash' t$ can be obtained by forward-chaining.

The second claim is evident, given the structure of the rules in $\vdash'$ (all the rules in $\vdash'$ are suitable for forward-chaing). The right-to-left direction of the first claim comes from the fact that the rules whose conclusions are non-goal formulas are rules from the original system, weakened by adding extra hypotheses. The other direction is a consequence of the following lemma:

[2] A stronger constraint is that when a composition rule is used to produce a principal premise of a decomposition rule, both rules can be removed (i.e., it is never necessary to compose some object to split it up afterwards); doing so until impossible yields a *normal derivation*. This is the case for BAN logic and the simple system of rules expressing the knowledge that is derivable from some piece of data by pairing, projection, encryption and decryption. However, this constraint is unnecessarily strong, and for instance fails to capture logics where this condition is not true for all composition-decomposition pairs, but where the remaining pairs can be transformed into a decomposition rule, such as GNY logic (see part 3.2). We call that last condition the **strong normal derivation criterion**.

Lemma 2 *If $\Gamma \vdash t$ then $\boxdot t, \Gamma \vdash' t$; we then have $t \vdash' t$ if there exists a normal derivation $\Gamma \vdash t$ that does not end with a composition rule.*

The proof is by induction on the structure of a normal derivation of $\Gamma \vdash t$:

- for axioms, it is evident;
- if the bottom rule is a composition rule

$$\frac{\mathcal{H}_1 \cdots \mathcal{H}_n}{t},$$

because we have $\boxdot t$ we can apply the corresponding rule

$$\frac{\boxdot t \quad \mathcal{H}_1 \cdots \mathcal{H}_n}{t}$$

at the bottom of the derivations of $\boxdot \mathcal{H}_i, \Gamma \vdash' \mathcal{H}_i$ of the induction hypothesis; the $\boxdot \mathcal{H}_i$ are derived using

$$\frac{\boxdot \mathcal{C}}{\boxdot \mathcal{H}_1 \cdots \boxdot \mathcal{H}_n};$$

- if the bottom rule is a decomposition rule

$$\frac{\mathcal{D}_1 \cdots \mathcal{D}_m \quad \mathcal{K}_1 \cdots \mathcal{K}_n}{t}$$

then for any i the sub-derivation of $\mathcal{D}_i$ is normal and does not end with a composition rule (we act on a normal derivation); then we have $\Gamma \vdash' \mathcal{D}_i$; we then use the triggering rule

$$\frac{\mathcal{D}_1 \cdots \mathcal{D}_m}{\boxdot \mathcal{K}_1 \cdots \boxdot \mathcal{K}_n}$$

and the induction hypothesis to get the $\boxdot \mathcal{K}_i, \Gamma \vdash' \mathcal{K}_i$; we apply the original decomposition rule at the bottom of these.

3 Application to GNY logic

As the treatment of BAN logic by our method is very similar to that of GNY logic, and BAN is simpler, we deal here with GNY logic.

3.1 Definitions

The GNY logic [6], named after its authors, Li Gong [5], Roger Needham and Raphael Yahalom is a logic of belief similar to BAN, but which addresses some deficiencies in that latter one; among other things, it addresses unencrypted message parts and includes a notion of recognizability.

We use the rules from [6], plus the following rule, which was forgotten in the original paper

$$\frac{P \triangleleft (X \rightsquigarrow Y)}{P \triangleleft X}\ \mathsf{TZ}$$

that is, if a principal is told a message meaning something, it is told that message. This rule is necessary to do the analysis of the Needham-Shroeder protocol found in [6].

GNY logic has two peculiar features that need special treatment:

The rationality rule All the rules of the logic may be applied with any preceding chain of belief modality. That is, for all rule

$$\frac{\mathcal{H}_1 \quad \ldots \quad \mathcal{H}_n}{\mathcal{C}}$$

and new principal variable name P, we have a rule

$$\frac{P \models \mathcal{H}_1 \quad \ldots \quad P \models \mathcal{H}_n}{P \models \mathcal{C}}$$

We consider the closure of the rules by this meta-rule.

Rewrites The set of formulas is quotiented by the following rewrites (or simplifications):

- $\{\{X\}_K\}_K^{-1} \longrightarrow X$
- $\left\{\{X\}_{+K}\right\}_{-K} \longrightarrow X$

This system R is confluent and terminating, and we'll call the normal forms of formulas under these rewrites their **canonical forms**.

We will show in the next subsection that these features can be accommodated within our framework.

3.2 The set of rules and our additions

We first divide the rules of GNY logic into three categories:

Decomposition rules such as rule T3

$$\frac{\boxed{P \triangleleft \{X\}_K} \quad P \ni K}{P \triangleleft X}$$

The principal premiss is shown within a frame.

Composition rules such as rule P6

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

These rules make a syntactically defined measure (see tab. 1) decrease.

Equivalence rules Equivalence rules such as F5

$$\frac{\boxed{P \models \sharp(+K)}}{P \models \sharp(-K)}$$

are handled like composition rules.

t	$\mathcal{M}(t)$
Beliefs and conveyance	
$P \models X$	$\mathcal{M}(X)$
$P \triangleleft X$	$20 + \mathcal{M}(X)$
$P \ni X$	$\mathcal{M}(X)$
$P \sim X$	$3 + \mathcal{M}(X)$
Properties of formulas	
$\sharp(X)$	$2 + \mathcal{M}(X)$
$\phi(X)$	$\mathcal{M}(X)$
Keys and secrets	
$P \overset{S}{\longleftrightarrow} Q$	0
$\overset{+P}{\mapsto} P$	0
$+K$	0
$-K$	0
Message constructors	
$\{X\}_K$	$3 + \mathcal{M}(X) + \mathcal{M}(K)$
$\{X\}_K^{-1}$	$3 + \mathcal{M}(X) + \mathcal{M}(K)$
(X, Y)	$1 + \mathcal{M}(X) + \mathcal{M}(Y)$
$F(X, Y)$	$2 + \mathcal{M}(X) + \mathcal{M}(Y)$
$H(X)$	$2 + \mathcal{M}(X)$
Jurisdiction	
$P \Rrightarrow X$	$1 + \mathcal{M}(X)$
$P \Rrightarrow P \models \star$	0
$(X \rightsquigarrow C)$	$\mathcal{M}(X) + \mathcal{M}(C)$
Detection of replays	
$\star X$	$1 + \mathcal{M}(X)$
Proof direction	
$\boxdot X$	$1 + \mathcal{M}(X)$

Table 1. The measure $\mathcal{M}$ used, defined inductively on the formulas.

We then have to deal with the rewrites. These are problematic; for instance, we have

$$\dfrac{\dfrac{P \ni K \quad P \ni \{X\}_K}{P \ni \{\{X\}_K\}_K^{-1}}\ \mathsf{P6}}{P \ni X}\ \text{simplification}$$

That is, we sometimes have to construct the decryption of a piece of data just to apply a simplification rule. We add a rule P6' standing for the former derivation:

$$\dfrac{\boxed{P \ni \{X\}_K} \quad P \ni K}{P \ni X}$$

We add similar rules in similar other cases, making up the $\vdash_+$ derivation system. $\vdash_+$ does not include rewrites. $\vdash_+$ is therefore made up of the rules found in [6], plus the following decomposition rules:

$$\dfrac{\boxed{P \ni \{X\}_K} \quad P \ni K}{P \ni X}\ \mathsf{P6'}$$

$$\dfrac{\boxed{P \ni \{X\}_{+K}} \quad P \ni -K}{P \ni X}\ \mathsf{P8'}$$

$$\dfrac{\boxed{P \models \sharp(\{X\}_K)} \quad P \ni K}{P \models \sharp(X)}\ \mathsf{F2'}$$

$$\dfrac{\boxed{P \models \sharp(\{X\}_{+K})} \quad P \ni -K}{P \models \sharp(X)}\ \mathsf{F4'}$$

$$\dfrac{\boxed{P \models \phi(\{X\}_K)} \quad P \ni K}{P \models \phi(X)}\ \mathsf{R2'}$$

$$\dfrac{\boxed{P \models \phi(\{X\}_{+K})} \quad P \ni -K}{P \models \phi(X)}\ \mathsf{R4'}$$

If $\vdash$ is GNY logic, consisting of the rules and the rewrites, we then state the following:

Property 3 *For all set of formulas Γ and formula t, $\Gamma \vdash t$ if and only if $C(\Gamma) \vdash C(t)$ where $C(f)$ denotes the canonical form under the rewrites of a formula or set of formulas.*

The proof is by induction on the structure of a derivation of $\Gamma \vdash t$. All pairs of rewrites and rules are considered; the

only annoying cases are rules P6, P8, F2, F4, R2, R4; in these cases the sequence

$$\dfrac{\dfrac{\begin{matrix}\vdots\ \Delta_1 \\ H_1\end{matrix} \quad \cdots \quad \begin{matrix}\vdots\ \Delta_m \\ H_m\end{matrix}}{C'}\ R}{C}\ \text{rewrite}$$

gets collapsed into

$$\dfrac{\begin{matrix}\vdots\ \Delta_1 \\ H_1\end{matrix} \quad \cdots \quad \begin{matrix}\vdots\ \Delta_m \\ H_m\end{matrix}}{C}\ R'.$$

We also have the following important property

Property 4 *$\vdash_+$ matches the normal derivation criterion.*

Again, the proof is by induction on the structure of the derivation, by considering all pairs of composition rules and decomposition rules. All pairs of composition and decomposition rules that constitute a cut (as defined in 2.2.1) can be collapsed. For instance

$$\dfrac{\dfrac{\begin{matrix}\vdots\ \Delta_1 \\ P \ni X\end{matrix} \quad \begin{matrix}\vdots\ \Delta_2 \\ P \ni K\end{matrix}}{P \ni \{X\}_K}\ \textsf{P6} \quad \begin{matrix}\vdots\ \Delta_3 \\ P \ni K\end{matrix}}{P \ni X}\ \textsf{P6'}$$

is collapsed into

$$\begin{matrix}\vdots\ \Delta_1 \\ P \ni X\end{matrix}$$

3.3 Transformation into a forward-chaining system

We apply our transformation (section 2.2) to the $\vdash_+$ system of rules, getting the $\vdash'_+$ system of rules.

The measure $\mathcal{M}$ we've defined (see tab. 1) over the formulas of this logic, then satisfies the following strict decreasing property:

Property 5 *For each rule*

$$\frac{\mathcal{H}_1 \cdots \mathcal{H}_n}{\mathcal{C}}$$

of $\vdash'_+$, except the equivalence rules, we have:

$$\mathcal{M}(\mathcal{C}) \leq \max\{\mathcal{M}(\mathcal{H}_1), \ldots, \mathcal{M}(\mathcal{H}_n)\} - 1.$$

This property is very important, since it ensures the finiteness of the search space, as it is explained below, and thus the termination of the method. We chose the measure *ad hoc* for this.

At first sight, it would seem that because of the "rationality rule", an infinite number of rules exist in GNY; that is, for any rule

$$\frac{H_1 \ldots H_n}{C}$$

there exists the infinity of rules

$$\frac{S.H_1 \ldots S.H_n}{S.C}$$

where S is a sequence of "believes" modalities $P_i \models \ldots$. But in a given problem, only a finite number of these rules needs to be considered. This is a consequence of the above property: since the belief modalities weigh in the measure of the formulas, and the measure of the formulas always decreases in the derivation, then we can bound the length of the chains of belief modalities to be taken into account.

Theorem 6 *The following strategy:*

- *rewrite (by the rewriting system R) until normalization all formulas in $\Gamma \cup \{t\}$; we will assume in the next step that these formulas are normalized;*
- *initialize the system with a set $\Gamma \cup \{\boxdot t\}$;*
- *iteratively, apply all the possible rules to the formulas in the set and add the consequences to the set, until a fixpoint is reached;*
- *test whether t is in the set*

decides whether $\Gamma \vdash t$.

- It **terminates**, since the set of generated formulas is finitely bounded and we stop once we don't add any new formula. This set is finite because we only build a bounded number of derivations:
  - there exists a finite number of rules that can be applied, since there is a finite number of rules in the logic without the rationality rule and there is a upper bound on the depth of the modality chains generated by the rationality rule;
  - the number of steps (that is, applied rules) in such a derivation is bounded because:
    - the number of non-equivalence steps is bounded, because of property 5;
    - the number of equivalence steps is bounded too, because equivalence classes are finite and our system cannot derive already derived formulas;
- It is **complete** and **correct**, following from property 3 and theorem 1 (which applies because of property 4).

The reader should not be confused by the fact that no computation of measure happens in our algorithm. The measure is solely a construct for the proof of termination of the algorithm.

4 A practical implementation

We implemented the aforementioned algorithm as a program in ML[3]. As we achieved adequate performance using relatively simple optimizations, discussed below, we didn't feel the need for more ambitious and complex implementation techniques, like BBDs.

4.1 Our algorithm

Our algorithm proceeds by iterations up to a fixpoint (see Fig. 2). The iterated procedure takes a current set S and outputs a set $S' \supset S$. How to implement the rationality rule ? We could of course compute the aforementioned (section 3.3) bound on the length of the belief modality chains to be considered. A more clever tactic, which is used in our implementation, is the following: consider $P = \{p \in S \mid \exists X\ p \models\!\!\models X \in S\}$; for each $p \in P$, consider $S_p = \{X \mid p \models\!\!\models X \in S\}$ and apply Z recursively, yielding S'_p; then $S' = S \cup W(S) \cup \{p \models\!\!\models X \mid p \in P \wedge X \in S'_p\}$ where $W(S)$ is the set of all conclusions that one may get by applying each rule to elements of S.

A side-effect of this transformation from $\vdash$ to $\vdash_+$ is that in the case of failure, the system may give some crude clues as what extra assumptions would have been needed: just collect the formulas f so that $f \neq t$ and $\boxdot f$ but not f have been derived; some further filtering may be necessary to produce a reasonably small number of hints for the user.

Extracting $\vdash_+$ proofs from the $\vdash'_+$ proofs (which contain "goal" operators) yielded by that techniques is simple, and our implementation does so before pretty-printing the proofs. It is easy to see that, given a proof of $\Gamma, \boxdot t \vdash'_+ t$, one can extract a proof of $\Gamma \vdash_+ t$ by pruning out all branches whose root is a "goal" formula. One can get $\vdash$ proofs, that is, pure GNY proofs, by simply expanding the rules we added to GNY to form $\vdash_+$ into GNY proof trees.

4.2 Avoiding combinatorial explosion

In this subsection, we do not consider the rationality rule, which is discussed in the previous subsection.

Implementing a working forward-chaining strategy for this problem is not as trivial as it seems. For instance, trying all possible rules in the fashion that to try a n-ary rule you match it against all the n-tuples of already derived formulas, until it ends, leads to prohibitive costs (in our case, $n = 6$, which makes the number of matchings grow in D^7, where D is the number of derivable formulas).

A first optimization we tried was based on the fact that one need not test all possible n-tuples, but only ones containing at least one "new" formula; that is, a formula made during the last application of the rules. This is not sufficient, since it reduces only to D^6; experimentally, this is far too slow.

Our implementation is based on the fact that to instantiate all the variables in a rule, you need not consider all the hypotheses; especially, in the modified versions of the composition rules, only one "goal" hypothesis suffices; in the decomposition and trigger rules, only the principal premisses are needed. Expensive exhaustive searches for the fully instantiated hypotheses are replaced by a much faster binary search.

Our algorithm is in Fig. 2.

To refine that algorithm even more, the pattern matching over d-tuples is replaced by d successive pattern matchings, and any failure of one the k searches ends the search process with a negative answer.

Such an algorithm is at most in $D^{d+1}(\log D)^k$, where $d = 2$ and $k = 4$ in the case of GNY logic.

Another obvious improvement is to consider only tuples of premisses where at least one is new, that is, has been added at the last iteration.

Obviously, more clever implementation techniques using, for instance, BDD-like data structures, could be more

do
  for all rule R having d principal premisses
    k auxiliary premisses
  do
    for all d-tuple D of "old" formulas
    do
      if D matches the principal premisses pattern
      then
        if the auxiliary premisses are in
          the "old" formulas
        then put the conclusions into
          the "new" formulas
  add the "new" formulas that were not already in
  the "old" formulas to the "old" formulas
until no more formulas are added

Figure 2. Practical GNY logic decision procedure.

[3]More precisely, it is written in Objective CAML, on which more information is available from `http://pauillac.inria.fr/ocaml`. A first ML program reads the set of GNY rules and outputs some other ML code expressing them, using the ML pattern matching facility. The resulting code, along with some parsing and control modules, is then compiled, resulting in an executable gnytool. gnytool takes an input of the protocol, the assumptions, and the purported conclusions, expressed in an ASCII syntax for GNY formulas, and outputs some pretty-printed LaTeX version of it, along with a pretty-printed version of the derivations produced for the proved conclusions. The total source code length is approximately 2700 lines long.

efficient. We didn't investigate these, since our implementation achieved good performances on realistic problems.

4.3 Example

We ran our automatic analyzer on, among other things, a version of the Needham-Shroeder protocol (tab. 2). This analysis took 0.3 second on a 400 Mhz Pentium-II machine[4], taking only 1.6 megabytes of memory, including the runtime system.

4.3.1 Assumptions

A $P \ni K_{ps}$

B $P \ni N_p$

C $P \models P \overset{K_{ps}}{\longleftrightarrow} S$

D $P \models \sharp(N_p)$

E $P \models \phi(Q)$

F $Q \ni K_{qs}$

G $Q \ni N_q$

H $Q \models Q \overset{K_{qs}}{\longleftrightarrow} S$

I $Q \models \sharp(N_q)$

J $Q \models \phi(N_q)$

K $P \models S \models\!\!\Rightarrow P \overset{K}{\longleftrightarrow} Q$

L $P \models S \models\!\!\Rightarrow S \models \star$

M $P \models Q \models\!\!\Rightarrow Q \models \star$

N $Q \models S \models\!\!\Rightarrow P \overset{K}{\longleftrightarrow} Q$

O $Q \models S \models\!\!\Rightarrow S \models \star$

P $Q \models P \models\!\!\Rightarrow P \models \star$

Q $S \ni K_{ps}$

R $S \ni K_{qs}$

S $S \ni K$

T $S \models P \overset{K_{ps}}{\longleftrightarrow} S$

U $S \models Q \overset{K_{qs}}{\longleftrightarrow} S$

V $S \models P \overset{K}{\longleftrightarrow} Q$

4.3.2 Goals

- $P \models P \overset{K}{\longleftrightarrow} Q$
- $P \ni N_q$

4.3.3 Derivation

obtained automatically

$P \models P \overset{K}{\longleftrightarrow} Q$,

applying rule J1 to:

$P \models S \models\!\!\Rightarrow P \overset{K}{\longleftrightarrow} Q$,

from hypothesis **K**.

$P \models S \models P \overset{K}{\longleftrightarrow} Q$,

applying rule J3 to:

$P \models S \models S \models P \overset{K}{\longleftrightarrow} Q$,

applying rule J2 to:

$P \models S \models\sim \left(\star\{(K,P)\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q\right)$,

$\vdots \quad \vdots \quad \vdots \quad \vdots \quad \vdots$

This is an excerpt from the 100-line pretty-printed output[5] of the analysis.

4.4 Other examples

Following [3, part 4], we analyzed some properties of the Otway-Rees protocol; the analysis, on the same hardware, took 0.08 s.

A previous implementation using the same method, this time of BAN logic [3] inside the PVS theorem prover [16], using its forward-chaining capability, allowed us to check mechanically the results of [3], on:

- the Otway-Rees protocol,
- the Needham-Shroeder shared-keys protocol,
- the wide-mouthed frog protocol,
- the Kerberos protocol (let's note that there's a slight error in [3]: one has to add the reasonable hypothesis $A \models \sharp(N_a)$ to get $A \models B \models A \models A \overset{K_{ab}}{\longleftrightarrow} B$),
- the Needham-Shroeder public-key protocol (the rule

$$\frac{P \models \langle X \rangle_Y \quad P \models \sharp(Y)}{P \models \sharp(\langle X \rangle_Y)}$$

which is consistent with the informal semantics of the logic, has to be added to reach the conclusions given in [3]).

This implementation, being made in a general system, was of course significantly slower than the one we wrote for GNY, and the implementation of some of the GNY capabilities looked difficult in that context.

[4] Using `ocamlopt -inline 2` for compilation, with OCaml version 2.00

[5] The LaTeX macros used in the outputs of our implementation and in this paper are available from `http://www.csl.sri.com/~monniaux/download/ban-gny.sty.tar.gz`.

1	$P \rightarrow S$	$(\star P, \star Q, \star N_p)$
2	$S \rightarrow P$	$\left(\star\left\{\left(N_p, Q, \star K, \left(\star\left\{(K,P)\right\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q\right)\right)\right\}_{K_{ps}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q\right)$
3	$P \rightarrow Q$	$\left(\star\left\{(\star K, \star P)\right\}_{K_{qs}} \rightsquigarrow S \models P \overset{K}{\longleftrightarrow} Q\right)$
4	$Q \rightarrow P$	$\star\left\{\star N_q\right\}_K$
5	$P \rightarrow Q$	$\left(\star\left\{\star F\left(N_q\right)\right\}_K \rightsquigarrow P \models P \overset{K}{\longleftrightarrow} Q\right)$

Table 2. The Needham-Shroeder protocol steps, in GNY logic

4.5 Generalization

How are the above techniques generalizable to other similar logics? The general method we've given: test whether $\Gamma \vdash t$ by generating all $\vdash'$-conclusions from $\Gamma \cup \{\boxdot t\}$ and testing whether t belongs to them, works as long as:

- $\vdash$ fulfills the normal derivation criterion;
- for any Λ, the set of formulas that are $\vdash'$-derivable from Λ is finite.

This second condition is ensured by the existence of a well-founded preorder $\preccurlyeq$ so that

- for any formula X, $X \prec \boxdot X$;
- for any decomposition rule

$$\frac{\mathcal{D}_1 \cdots \mathcal{D}_m \quad \mathcal{K}_1 \cdots \mathcal{K}_n}{\mathcal{C}},$$

whose set of variables is Ξ, and any instanciation ξ, for any $F \in \{\mathcal{C}[\xi/\Xi], \boxdot\mathcal{K}_1[\xi/\Xi], \ldots, \boxdot\mathcal{K}_n[\xi/\Xi]\}$ there exists $H \in \{\mathcal{D}_1[\xi/\Xi], \ldots, \mathcal{D}_m[\xi/\Xi]\}$ so that $F \prec H$.

For GNY logic, we chose $x \preccurlyeq y \iff \mathcal{M}(x) \leq \mathcal{M}(y)$.

5 Conclusions

We have a general framework for deciding authentication logics, encompassing BAN and GNY logics. This framework is flexible and likely to handle other similar logics.

Particularizing our method on well-known authentication logics, we implemented a decision procedure for BAN logic in a general-purpose forward-chaining system, which we used to mechanically check the results in the reference paper on this logic, and we implemented a special-purpose decision procedure for GNY logic. This latter system decided GNY problems found in related papers using little time (a fraction of a second) and memory space.

In our opinion, this tool can be integrated in a general environment for the analysis of cryptographic protocols, as a fast analysis that, applied first, before more expensive methods, can help detecting some bugs in protocol specifications, particularly unwanted assumptions.

Acknowledgements

We wish to thank Jonathan K. Millen for his valuable help in making the paper clearer, John Rushby for inviting us to his group at SRI International, the OCaml team and all the people working on LaTeX packages and related tools.

References

[1] Stephen H. Brackin. Deciding cryptographic protocol adequacy with HOL. In *IWHOLTP: 8th International Workshop on Higher Order Logic Theorem Proving and Its Applications*, volume 971 of *Lecture Notes in Computer Science*. Springer-Verlag, 1995.

[2] Stephen H. Brackin. Automatic formal analyses of cryptographic protocols. In *NISSC: 19th National Information Systems Security Conference*, pages I.181 – 193, Baltimore, MD, October 1996. National Institute of Standards and Technology and National Computer Security Center.

[3] Michael Burrows, Martin Abadi, and Roger Needham. A logic of authentication. Technical Report 39, Digital Equipment Corporation, Systems Research Centre, February 1989.
`http://gatekeeper.dec.com/pub/DEC/SRC/research-reports/abstracts/src-rr-039.html`

[4] J.Y. Girard. *Proofs and Types*. Cambridge University Press, 1990. Translated and with appendices by Paul Taylor, Yves Lafont.

[5] Li Gong. *Cryptographic Protocols for Distributed Systems*. PhD thesis, University of Cambridge, Cambridge, England, April 1990.
`http://java.sun.com/people/gong/papers/phd-thesis.ps.gz`

[6] Li Gong, Roger Needham, and Raphael Yahalom. Reasoning about belief in cryptographic protocols. In *IEEE Symposium on Research in Security and*

Privacy, pages 234–248, Oakland, California, May 1990. IEEE Computer Society, IEEE Computer Society Press.
http://java.sun.com/people/gong/papers/gny-oakland.ps.gz

[7] James W. Gray and John McLean. Using Temporal Logic to Specify and Verify Cryptographic Protocols (progress report). In *America in the Age of Information, National Science and Technology Council Committee on Information and Communications Forum*, Bethesda, 1995.
http://www.itd.nrl.navy.mil/ITD/5540/publications/CHACS/1995/1995mclean-NSTCCIC.pdf

[8] D. Kindred and J. M. Wing. Fast, automatic checking of security protocols. In *Second USENIX Workshop on Electronic Commerce*, pages 41–52, Oakland, California, November 1996. USENIX.
http://www-cgi.cs.cmu.edu/afs/cs.cmu.edu/project/venari/www/usenix96-submit.html

[9] G. Lowe and B. Roscoe. Using CSP to detect Errors in the TMN Protocol. *IEEE Transactions on Software Engineering*, 23(10):659–669, 1997.

[10] W. Marrero, E.M. Clarke, and S. Jha. Model checking for security protocols. Technical Report CMU-SCS-97-139, Carnegie Mellon University, May 1997.

[11] Catherine Meadows. The NRL Protocol Analyzer: An Overview. *Journal of Logic Programming*, 1995. To appear.
http://www.itd.nrl.navy.mil/ITD/5540/publications/CHACS/1995/1995meadows-toappearJLP.ps

[12] J. K. Millen and H-P. Ko. Narrowing terminates for encryption. In *Computer Security Foundations Workshop*, 1996.
http://www.csl.sri.com/~millen/narr_term.ps

[13] J.C. Mitchell, M. Mitchell, and U. Stern. Automated analysis of cryptographic protocols using murphi. In *IEEE Symp. Security and Privacy*, pages 141–153, Oakland, 1997.
ftp://theory.stanford.edu/pub/jcm/papers/murphi-protocols.ps

[14] Lawrence C. Paulson. Proving properties of security protocols by induction. In *10th Computer Security Foundations Workshop*, pages 70–83. IEEE Computer Society Press, 1997.
http://www.cl.cam.ac.uk/users/lcp/papers/protocols.html

[15] Johann Schumann. Automatic verification of cryptographic protocols with SETHEO. In W. McCune, editor, *14th Conference on Automated Deduction (CADE)*, volume 1249 of *Lecture Notes in Computer Science*. Springer-Verlag, 1997.

[16] SRI International, Computer Science Laboratory. *Overview of the PVS Verification System.*
http://www.csl.sri.com/pvs/overview.html

[17] P. Syverson. Adding time to a logic of authentication. In *1st ACM Conference on Computer and Communications Security*, pages 97–101, 1993.
http://www.itd.nrl.navy.mil/ITD/5540/publications/CHACS/1993/1993syverson-ccs.ps

[18] P. Syverson and P. C. van Oorschot. On unifying some cryptographic protocol logics. In *1994 IEEE Computer Society Symposium on Research in Security and Privacy*, pages 14–28, May 1994.
http://www.itd.nrl.navy.mil/ITD/5540/publications/CHACS/1994/1994syverson-sp.ps