Automated Detection of Confidentiality Goals*

Anders Moen Hagalisletto
Norwegian Computing Center
Postbox 1080 Blindern, 0316 Oslo, Norway
Email:anders.moen@nr.no

Abstract

The security goals of an authentication protocol specify the high level properties of a protocol. Despite the importance of goals, these are rarely specified explicitly. Yet, a qualified analysis of a security protocol requires that the goals are stated explicitly. We propose a novel approach to find confidentiality goals in an automated way, based only on the protocol specification. The benefits of the method are: (i) Manual specification of goals is replaced by fully automated methods, (ii) the algorithm constructs the entire protection domain of a protocol, that is, all private and shared secrets, and (iii) the goal of an attack can be found, explaining which compromised entities are shared between the attacker and the honest principals.

Keywords: Security protocols, formal specification, automated refinement, security goals.

1 Introduction

To know if a protocol is secure one must know what it aims to achieve. Security goals state this explicitly. Security goals for protocols conjecture the confidentiality, integrity or authenticity of particular entities, such as: A session key must be kept secret between two parties, a nonce must not be tampered with, or the agent Alice should ensure that she really communicates with Bob.

Therefore, security goals are used as the main guiding principles in the design and analysis of a protocol. In practice, security goals have rarely been specified explicitly, neither for commercial [5] nor academic protocols [2]. The application of formal methods in order to analyse security protocols is about to change this. Initiated by the ground-breaking work of Gavin Lowe [7], and the analysis tools developed the last decade [1, 3], security goals of several classical protocols were specified explicit. Protocol analysis takes a security goal $\mathscr{G}$ of a protocol P as input and returns an output: "yes the protocol fulfills the goal" or "no, it is not secure, the goal $\mathscr{G}$ can be broken by the attack description $\mathscr{A}$".

Security goals are still specified manually, none of the current tools like for instance OFMC[1], CL-Atse[12], or Athena[11], provide automated support finding them. Analysis tools are rarely used in industry as we have experienced while analysing protocols for Norwegian enterprises.

The formalization of security goals have been a matter of careful investigation, confidentiality could be given a successful set theoretic interpretation, while authentication and integrity is tied to the notion of execution as shown in [8]. The current paper can be summarized in one single question:

> *What security properties might the designers have had in mind, while defining the protocol?*

We show that the detected goals includes all goals defined in the literature. Moreover, the output of our algorithm can be used to gain rapidly understanding of new protocols and the composition of several complex protocols.

The remainder paper is organized as follows: In Section 2, we first present a language for specification of *intended* protocols, and then show how to specify *attack* descriptions. We present an agent model that gives an interpretation of the roles played in the protocol, as agents possessing beliefs.Protocols, keys, timestamps, nonces, and cipher-texts are all beliefs possessed by an agent. In Section 3, we present the method for automated detection of confidentiality goals, by using the Needham Schroeder public key protocol to illustrate

*This work has been funded by Norwegian Research Council as part of the RSE-SIP in the VERDIKT programme. The author would like to thank Wolfgang Leister, Lothar Fritsch, Truls Fretland, Åsmund Skomedal, Jørn Inge Vestgaarden, and Dag Normann for comments on earlier drafts versions of this paper.

[1] `http://www.inf.ethz.ch/personal/moederss/research/`

each step in the algorithm. In Section 4, the algorithm is deployed on a large collection of protocol specifications. Finally we evaluate the approach and point to future work.

2 Protocol specification

Consider the Needham Schroeder authentication protocol for public keys [9]. The overall goal of the protocol is to provide mutual authentication both for the initiator Alice and the responder Bob. The initial assumption of the protocols that the agents possess their own private keys and the required public keys. The means to achieve this goal is by using freshly generated nonces to be kept secret by encrypting them using the two agents associated public keys. The authentication part of the protocol is given as follows, only involving Alice (A) and Bob (B):

$$\begin{array}{lll} (P_1) & A \longrightarrow B: & E(K_B : N_A, A) \\ (P_2) & B \longrightarrow A: & E(K_A : N_A, N_B) \\ (P_3) & A \longrightarrow B: & E(K_B : N_B) \end{array}$$

The following low level confidentiality goals are involved in this protocol: the nonces should not be disclosed to other agents than Alice and Bob, formalized by Secret($\{A, B\}, \{N_A, N_B\}$), meaning that the group consisting of Alice and Bob are the only agents sharing the beliefs about the nonces N_A and N_B. The candidates to confidentiality goals for the protocol is divided into three groups, the private beliefs of Alice and Bob only contain their respective private keys: K_A^{-1} and K_B^{-1}, and finally the group consisting of both agents consist of the two nonces, and the public keys.

2.1 Formal specification language

We define a language for specifying security protocols $\mathscr{L}_P$ as follows: $\mathscr{L}_P$ consists of terms of five sorts, *agents*, *nonces*, *time-stamps*, *keys*, and natural numbers. The *agent-names* are typically written "Alice", "Bob", "Server", *agent-variables* are written x, y, $x_1, x_2, \ldots$, and *agent-terms* a, b, c, $\ldots$, or in the general style t^A, t_1^A, t_2^A, $\ldots$, indicating that the terms have the *agent sort*. Variables for the new sorts are labeled with x^N, x^T, and x^K respectively, when their sorts are emphasized. Constants include *protocol names* μ, μ_1, μ_2, *encryption methods* for cryptography s (symmetric) and a (asymmetric), in addition to the indicators i (private) and u (public). There are function symbols for nonces $\mathsf{n}(N, a)$, time-stamps $\mathsf{stamp}(t^T, t^A)$, keys $\mathsf{key}(\mathsf{s}, a, b, x^K)$, $\mathsf{key}(\mathsf{a}, \mathsf{i}, a, x^K)$, $\mathsf{key}(\mathsf{a}, \mathsf{u}, a, x^K)$, and for the concatenation of protocol names.

Definition 1 *$\mathscr{L}_P$ is the smallest language such that:*

(i) Each of the following atomic formulas are in $\mathscr{L}_P$

ε	the empty sentence
$a = b$	equality
$\mathsf{Agent}(a)$	a is an agent
$\mathsf{isKey}(k)$	k is a key
$\mathsf{isNonce}(\mathsf{n}(N, a))$	$\mathsf{n}(N, a)$ is a nonce
$\mathsf{Time}(\mathsf{stamp}(N, a))$	$\mathsf{stamp}(N, a)$ time stamp
$\mathsf{playRole}(a, x, \mu)$	a plays the x-role in protocol μ
$\mathsf{role}(a)$	a is a role in a protocol

(ii) If φ, ψ, $\xi^{\mathcal{T}}$, $\xi^{\mathcal{A}}$, $\xi^{\mathcal{S}} \in \mathscr{L}_P$, then so are;

$\neg\varphi$, $\varphi \to \psi$	propositional logic
$\mathsf{Transmit}(a, b, \varphi)$	a sends the message φ to b
$\mathsf{Bel}_a(\varphi)$	a believes φ
$\varphi \,\mathcal{U}\, \psi$	φ holds until ψ holds
$\forall x\, \varphi$	first order quantification
$\forall X\, \varphi$	second order quantification
$E[k : \varphi]$	encrypt φ using key k
$D[k : \varphi]$	decrypt φ using key k
$\mathsf{Enforce}_{t^A}(\varphi)$	enforce agent t^A to do φ
$\mathsf{protocol}[\mu, N, \xi^{\mathcal{T}}, \xi^{\mathcal{A}}, \xi^{\mathcal{S}}, \Phi]$	protocol operator

As usual $\wedge$, $\vee$ and $\leftrightarrow$ are definable using $\neg$ and $\to$ and $\top = \varepsilon \to \varepsilon$. The operator *before* $\mathscr{B}$ is definable from $\mathcal{U}$ by $\varphi \,\mathscr{B}\, \psi = \neg(\neg\varphi \,\mathcal{U}\, \psi)$. An event is the minimal unit in a protocol specification. The basic examples of events are transmissions or the empty event ε. A protocol is a chain of events between agents. A *chain of events* is a sentence of the form $\varphi_1 \,\mathscr{B}\, \varphi_2 \wedge \varphi_2 \,\mathscr{B}\, \varphi_3 \wedge \ldots \wedge \varphi_{n-1} \,\mathscr{B}\, \varphi_n$, where each φ_i is an event. We let $\Phi = \varphi \,\mathscr{B}\, [\Phi']$ denote a chain of events written by recursion, hence φ is a single event and Φ' chain of events. The sentence $\mathsf{protocol}[\mu, N, \xi^{\mathcal{T}}, \xi^{\mathcal{A}}, \xi^{\mathcal{S}}, \Phi]$ reads "protocol named μ with session number N with the total roles $\xi^{\mathcal{T}}$, the agent specific roles $\xi^{\mathcal{A}}$, the start-roles $\xi^{\mathcal{S}}$ and the protocol body Φ".

We say that a *textbook protocol* is a valid protocol where each single event is of the form $\mathsf{Transmit}(x_j, x_k, \xi)$, where x_j, x_k are agent-variables and ξ is a sentence in $\mathscr{L}_P$. Figure 1 shows the textbook specification of the Needham Schroeder public key protocol. The protocol body consists of three transmission sentence, and ends with the empty event.

An *attack protocol* is a protocol P, such that some of the transmissions contains occurrences of *impersonation terms* $\mathbb{im}(I, A)$ either as the sender or the receiver of a transmission. The term $\mathbb{im}(I, A)$ reads "I impersonates as A". If t is an agent variable or agent name, then $\underline{t} = \overline{t} = t$. If $t = \mathbb{im}(t', t'')$, then $\underline{t} = t'$ and $\overline{t} = t''$.

$\mathsf{protocol}[\mathrm{NeedhamPK}, N, \mathsf{role}(x) \wedge \mathsf{role}(y), \mathsf{role}(x) \wedge \mathsf{role}(y), \mathsf{role}(x),$

$\mathsf{Transmit}(x, y, E(\mathsf{key}(\mathsf{a}, \mathsf{u}, y) \ : \ \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)))$
$\mathscr{B}\ \mathsf{Transmit}\ (y, x, E(\mathsf{key}(\mathsf{a}, \mathsf{u}, x) \ : \ \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))))$
$\mathscr{B}\ \mathsf{Transmit}\ (x, y, E(\mathsf{key}(\mathsf{a}, \mathsf{u}, y) \ : \ \mathsf{isNonce}(\mathsf{n}(z_2, y))))$
$\mathscr{B}\ \varepsilon]$

Figure 1. Needham Schroeder public key protocol.

2.2 Confidentiality

Security properties can be expressed within quantified epistemic logic: we say that a fact φ is *contingent secret* for a group of agents G, written $\mathrm{CSecret}(G, \varphi)$, iff every agent not in the group does not possess φ, formally:

$$\mathrm{Secret}(G, \varphi) \stackrel{\mathrm{def}}{\longleftrightarrow} \forall x(\mathsf{Agent}(x) \wedge x \notin G \rightarrow \neg\, \mathsf{Bel}_x\,(\varphi)).$$

A fact is *private* for an agent a if the group G is a singleton set, that is $\mathrm{Private}(a, \varphi) \stackrel{\mathrm{def}}{\longleftrightarrow} \mathrm{Secret}(\{a\}, \varphi)$.

2.3 Agent model

An operational semantics for protocol specifications and attacks has been defined in [4], that showed how agents process messages as part of a protocol session. We introduce the *parallel operator* $\parallel$, and use the notation $o_1 \parallel o_2 \parallel \ldots \parallel o_n$, to express that the agents $o_1, o_2, \ldots, o_n$ coexists concurrently. A rewrite rule $t \longrightarrow t'$ can be interpreted as a local transition rule allowing an instance of the term t to evolve into the corresponding instance of the pattern t'. Each rewrite rule describe how a part of a configuration can evolve in one transition step. A *configuration* is a snapshot of a dynamic system evolving. The parallel operator is an abelian monoid over the set of configuration with an identity element, the empty configuration, denoted $\mathfrak{E}$. The agent is represented by the following structure:

$$< \underbrace{\mathsf{id}}_{\text{agent name}} \ | \ \underbrace{\mathsf{bel}}_{\text{set of sentences}} >$$

where id denotes the *identity* the agent (variable), while bel denotes its current *set of beliefs*. The beliefs contain the free variables of a agent that occurs in the specification: In the case of the Needham Schroeder specification in Figure 1, an agent possess concrete beliefs involving the agent variables x, y, and z in addition to the nonce variables z_1 and z_2.

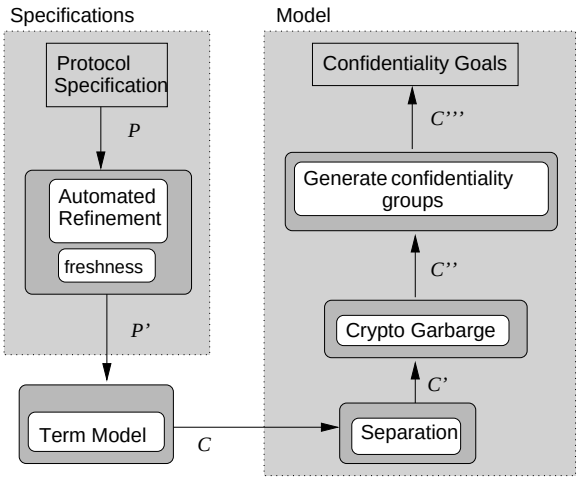

Figure 2. The process of constructing goals.

3 Automated construction of goals

The process of constructing goals is described by the following process, depicted in Figure 2. Initially a plain textbook specification P taken as input to the automated refinement module. Then an agent model C is constructed that consists of agents with beliefs derived from the refined specification. The model C does not distinguish between elements in the messages that are encrypted or publicly accessible. The separation module introduces a designated agent called *Public*, traverses the textbook specification and moves the appropriate beliefs from the agents and into *Public*, if the elements can be obtained by a third party wiretapping the communication. The resulting configuration C' contains a agent representative for each role in the protocol specification in addition to the eavesdropping agent *Public*. The configuration C' is thus a separation of the agents' potential secrets and the unprotected entities in the protocol session. Some of the beliefs in the latter configuration are not potential candidates for security goals, this includes public keys and cipher text elements. Hence, before building the final goals, the *cryptographic garbage* is removed from the beliefs of the agents, resulting in a rather focused configuration C''. The final step in the process is the construction of *confidentiality groups*. This algorithm constructs agents representing beliefs that are intended to be private, beliefs shared by given pair agents, beliefs shared by given triples of agents, etc. The confidentiality goals can be extracted directly from the agents in the resulting in the configuration C'''.

$\mathsf{protocol}[\mathrm{NSPKIrefined}, 0,$
$\quad \mathsf{role}(x) \wedge \mathsf{role}(y) \qquad \mathsf{role}(x) \wedge \mathsf{role}(y), \mathsf{role}(x),$
$\mathsf{Enforce}_x(\mathsf{Bel}_x(\mathsf{newNonce}(\mathsf{n}(z_1, x))))$ (1)
$\mathscr{B}\ \mathsf{Bel}_x(\mathsf{Agent}(x))$ (2)
$\mathscr{B}\ \mathsf{Bel}_x(\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, y)))$ (3)
$\mathscr{B}\ \mathsf{Enforce}_x(\mathsf{Bel}_x(E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)]))$ (4)
$\mathscr{B}\ \mathsf{Bel}_x(\mathsf{Agent}(y))$ (5)
$\mathscr{B}\ \mathsf{Transmit}(x, y, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)])$ (6)
$\mathscr{B}\ \mathsf{Bel}_y(\mathsf{Agent}(x))$ (7)
$\mathscr{B}\ \mathsf{Enforce}_y(\mathsf{Bel}_y(\mathrm{Trust}(y, x,$
$\quad E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)])))$ (8)
$\mathscr{B}\ \mathsf{Bel}_y(\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, y)))$ (9)
$\mathscr{B}\ \mathsf{Bel}_y(\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{i}, y)))$ (10)
$\mathscr{B}\ \mathsf{Enforce}_y(\mathsf{Bel}_y(D[\mathsf{key}(\mathsf{a}, \mathsf{i}, y) :$
$\quad E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)]]))$ (11)
$\mathscr{B}\ \mathsf{Bel}_y(\mathsf{isNonce}(\mathsf{n}(z_1, x)))$ (12)
$\mathscr{B}\ \mathsf{Enforce}_y(\mathsf{Bel}_y(\mathsf{newNonce}(\mathsf{n}(z_2, y))))$ (13)
$\mathscr{B}\ \mathsf{Bel}_y(\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, x)))$ (14)
$\mathscr{B}\ \mathsf{Enforce}_y(\mathsf{Bel}_y($
$\quad E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))]))$ (15)
$\mathscr{B}\ \mathsf{Transmit}(y, x,$
$\quad E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))])$ (16)
$\mathscr{B}\ \mathsf{Enforce}_x(\mathsf{Bel}_x(\mathrm{Trust}(x, y,$
$\quad E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))])))$ (17)
$\mathscr{B}\ \mathsf{Bel}_x(\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, x)))$ (18)
$\mathscr{B}\ \mathsf{Bel}_x(\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{i}, x)))$ (19)
$\mathscr{B}\ \mathsf{Enforce}_x(\mathsf{Bel}_x(D[\mathsf{key}(\mathsf{a}, \mathsf{i}, x) :$
$\quad E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))]]))$ (20)
$\mathscr{B}\ \mathsf{Bel}_x(\mathsf{isNonce}(\mathsf{n}(z_2, y)))$ (21)
$\mathscr{B}\ \mathsf{Enforce}_x(\mathsf{Bel}_x(E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_2, y))]))$ (22)
$\mathscr{B}\ \mathsf{Transmit}(x, y, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_2, y))])$ (23)
$\mathscr{B}\ \mathsf{Enforce}_y(\mathsf{Bel}_y(\mathrm{Trust}(y, x,$
$\quad E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_2, y))])))$ (24)
$\mathscr{B}\ \mathsf{Enforce}_y(\mathsf{Bel}_y(D[\mathsf{key}(\mathsf{a}, \mathsf{i}, y) : E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) :$
$\quad \mathsf{isNonce}(\mathsf{n}(z_2, y))]]))$ (25)
$\mathscr{B}\ \varepsilon]$

Figure 3. Automated refinement.

3.1 Constructing models

From a textbook specification of an authentication protocol, an automated refinement algorithm, denoted $\Re^{\star}_{\mathrm{A}}$, can be defined [6]. The refinement of the Needham Schroeder specification P^{NS} given in Figure 1 is described in Figure 3, denoted $\Re^{\star}_{\mathrm{A}}(\mathsf{P}^{\mathrm{NS}})$. The refinement algorithm injects local assumptions about the message transmissions: Prior to the first message transmission (line 6), the fresh nonce is constructed by the command $\mathsf{Enforce}_x(\mathsf{Bel}_x(\mathsf{newNonce}(\mathsf{n}(z_1, x))))$ (1), the agent x is self-aware (2), x knows y's public key (3), and x encrypts the content by using the public key (4), and finally x believes that the receiver y is an agent (5). Hence the clauses (1-5) describes the assumptions required for agent x to send the first message. The receiver y can extract information from the message received, lines (7-12) contain the explicit assumptions about y's beliefs: Agent y learns that x is an agent (7), y is enforced to trust the content of the message (8), y knows both its public and private keys (9-10). With the appropriate keys at hand, agent y decrypts the received message (11), and obtain the nonce in clear text, as described in (12).

$\prec x \mid \mathsf{bel}_x = \{\mathsf{Agent}(x), \mathsf{Agent}(y),$
$\mathsf{isNonce}(\mathsf{n}(z_1, x)), \mathsf{isNonce}(\mathsf{n}(z_2, y)),$
$\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{i}, x)), \mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, x)), \mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, y)),$
$E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)],$
$E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))],$
$E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_2, y))],$
$\mathsf{Transmit}(x, y, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)]),$
$\mathsf{Transmit}(y, x, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))]),$
$\mathsf{Transmit}(x, y, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_2, y))])\} \succ$
$\quad \| \qquad \prec y \mid \mathsf{bel}_y = \{\mathsf{Agent}(x),$
$\mathsf{isNonce}(\mathsf{n}(z_1, x)), \mathsf{isNonce}(\mathsf{n}(z_2, y)),$
$\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{i}, y)), \mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, x)), \mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, y)),$
$E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)],$
$E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))],$
$E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_2, y))],$
$\mathsf{Transmit}(x, y, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)]),$
$\mathsf{Transmit}(y, x, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) : \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))]),$
$\mathsf{Transmit}(x, y, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) : \mathsf{isNonce}(\mathsf{n}(z_2, y))])\} \succ$

Figure 4. Model of the refined beliefs.

From a protocol specification a *model* can be constructed, the model contains agents representing each role in the protocol. The refined specification is traversed and the local assumptions are injected into the belief set of the appropriate agent.

$(i)\quad \mathcal{C}oll(\mathsf{protocol}[\mu, N, \mathsf{role}(a) \wedge \xi^{\mathcal{T}}, \xi^{\mathcal{A}}, \xi^{\mathcal{S}}, \Phi], \mathscr{C}) =$
$\qquad \mathcal{C}oll(\mathsf{protocol}[\mu, N, \xi^{\mathcal{T}}, \xi^{\mathcal{A}}, \xi^{\mathcal{S}}, \Phi],$
$\qquad\qquad \mathscr{C} \parallel \prec a \mid \mathsf{bel} := \varnothing \succ)$
$(ii)\quad \mathcal{C}oll(\mathsf{protocol}[\mu, N, \top, \xi^{\mathcal{A}}, \xi^{\mathcal{S}}, \Phi], \mathscr{C}) = \mathcal{C}oll(\Phi, \mathscr{C})$
$(iii)\quad \mathcal{C}oll(\mathsf{Transmit}(x, y, F)\,\mathscr{B}\,\Phi, \mathscr{C}) =$
$\qquad \mathcal{C}oll(\Phi, \mathrm{insert}(\underline{x}, \mathsf{Transmit}(x, y, F),$
$\qquad\qquad \mathrm{insert}(\underline{y}, \mathsf{Transmit}(x, y, F), \mathscr{C}))$
$(iv)\quad \mathcal{C}oll(\mathsf{Bel}_x(F)\,\mathscr{B}\,\Phi, \mathscr{C}) = \mathcal{C}oll(\Phi, \mathrm{insert}(x, F, \mathscr{C}))$
$((v)\quad \mathcal{C}oll(\varepsilon, \mathscr{C}) = \mathscr{C}$

The function $\mathrm{insert}(a, F, \mathscr{C})$, injects a sentence F into the belief set of the agent a (concrete name or variable), that occurs in the configuration $\mathscr{C}$.

The model represents a symbolic execution of the participants interpreted by their role in the protocol. The model is a *Henkin-model* in the logical sense [10], where variables are interpreted as variables in the model (there are no concrete agent names). This means that the names of the agents are the agent variables specified by the roles in the protocol, in clause (i) "empty" agents are constructed. When there are no more agents to construct (ii), the main body of the protocol is analyzed. A transmission from x to y is analyzed by adding the message $\mathsf{Transmit}(x, y, F)$ to both agents belief sets (iii). A belief assertion in the refined specification is inserted into the belief set of the agent (iv). Finally, the accumulated configuration is returned if we reach the end of the specification (v).

If we run our algorithm on the refined specification in Figure 3, we obtain the model depicted in Figure 4.

The agents possess beliefs about the agents participating in the session, the appropriate public and private keys, ciphertexts, and even the messages. Since this configuration contains too much information, our next task is to eliminate beliefs superfluous to goal detection.

3.2 Separation of cipher-text

The elements in the model do not separate between the elements that can appear as plain-text and cipher text. The separation function emulates an execution, and moves the elements of a message that is publicly available into the agent denoted Public. Hence a textbook specification of a protocol (like the one given in Figure 1), is taken as input in addition to a configuration of agents with beliefs.

Definition 2 *Let Φ be a textbook specification of a protocol and $\mathscr{C}$ a configuration. Then separation $\mathit{Sep}(\Phi, \mathscr{C})$ is defined as follows:*

(i) $\mathit{Sep}(\varepsilon, \mathscr{C}) = \mathscr{C}$
(ii) $\mathit{Sep}((\mathsf{Transmit}(x, y, F))\,\mathscr{B}\,\Phi, \mathscr{C}) =$
$\quad \mathit{Sep}(\Phi, \mathit{Sep}(\mathsf{Transmit}(x, y, F), \mathscr{C}))$
(iii) $\mathit{Sep}((\mathsf{Enforce}_x(\mathsf{Bel}_x(F)))\,\mathscr{B}\,\Phi, \mathscr{C}) = \mathit{Sep}(\Phi, \mathscr{C})$
(iv) $\mathit{Sep}(\mathsf{Transmit}(x, y, F), \mathscr{C}) =$
$\quad \mathit{Sep}(F, \mathrm{insert}(\mathrm{Public}, \mathsf{Agent}(x) \wedge \mathsf{Agent}(y) \wedge$
$\quad\quad \mathsf{Transmit}(x, y, F), \mathscr{C}))$
(v) $\mathit{Sep}(F_1 \wedge F_2, \mathscr{C}) = \mathit{Sep}(F_1, \mathit{Sep}(F_2, \mathscr{C}))$
(vi) $\mathit{Sep}(E[K : F], \mathscr{C}) = \mathscr{C}$
(vii) $\mathit{Sep}(F, \mathscr{C}) = \mathrm{insert}(\mathrm{Public}, F, \mathscr{C})$
$\quad$ *if F is an elementary sentence.*

Each message transmission in the specification is analyzed sequentially (ii): For each message the content is analyzed: the message including the header information is public (iv) conjunctions are separated (v), cipher-text is left unchanged (vi), while occurrences of plain-text entities like nonces, timestamps, strings, etc., are public. In order to remove the beliefs that are public from the beliefs of agent x and y, the equation

$$\prec x \mid \mathsf{bel} \cup \{F\} \succ \| \prec \mathrm{Public} \mid \mathsf{bel} \cup \{F\} \succ = \prec x \mid \mathsf{bel} \succ \| \prec \mathrm{Public} \mid \mathsf{bel} \cup \{F\} \succ$$

always applies.

If we run the separation function on the automatically generated model of the Needham Schroeder protocol depicted in Figure 4 we obtain the separated model described in Figure 5. Both agent x and y possess the nonces, keys and ciphertexts from the basic model, but the transmissions and agent names are moved into the Public agent.

$$\begin{array}{l}
\quad \prec x \mid \mathsf{bel}_x = \{\mathsf{isNonce}(\mathsf{n}(z_1, x)), \mathsf{isNonce}(\mathsf{n}(z_2, y)), \\
\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{i}, x)), \mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, x)), \mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, y)), \\
E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) \ : \ \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)], \\
E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) \ : \ \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))], \\
E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) \ : \ \mathsf{isNonce}(\mathsf{n}(z_2, y))]\} \succ \\
\quad \| \quad \prec y \mid \mathsf{bel}_y = \{\mathsf{isNonce}(\mathsf{n}(z_1, x)), \mathsf{isNonce}(\mathsf{n}(z_2, y)), \\
\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{i}, y)), \mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, x)), \mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, y)), \\
E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) \ : \ \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)], \\
E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) \ : \ \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))], \\
E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) \ : \ \mathsf{isNonce}(\mathsf{n}(z_2, y))]\} \succ \\
\quad \| \quad \prec \mathrm{Public} \mid \mathsf{bel}_y = \{\mathsf{Agent}(x), \mathsf{Agent}(y), \\
\mathsf{Transmit}(x, y, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) \ : \ \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{Agent}(x)]), \\
\mathsf{Transmit}(y, x, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, x) \ : \ \mathsf{isNonce}(\mathsf{n}(z_1, x)) \wedge \mathsf{isNonce}(\mathsf{n}(z_2, y))]), \\
\mathsf{Transmit}(x, y, E[\mathsf{key}(\mathsf{a}, \mathsf{u}, y) \ : \ \mathsf{isNonce}(\mathsf{n}(z_2, y))])\} \succ
\end{array}$$

Figure 5. Model of the separated beliefs.

3.3 Removing unresolved crypto entities

The separated model contains beliefs that are not candidates for confidentiality goals. Unresolved cipher texts, public keys, and entire messages are cryptographic "garbage", that should be removed. This is taken care of by the garbage-collector for cryptographic entities:

$$\begin{array}{l}
\mathit{cryptoGarb}(\mathfrak{E}) = \mathfrak{E} \\
\mathit{cryptoGarb}(\prec x \mid \mathsf{bel} \succ \| \mathscr{C}) = \\
\prec x \mid \mathit{cryptoGarb}(\mathsf{bel}) \succ \| \mathit{cryptoGarb}(\mathscr{C}) \\
\mathit{cryptoGarb}(\varnothing) = \varnothing \\
\mathit{cryptoGarb}(\mathsf{bel} \cup \{E[K \ : \ F]\}) = \mathit{cryptoGarb}(\mathsf{bel}) \\
\mathit{cryptoGarb}(\mathsf{bel} \cup \{D[K \ : \ F]\}) = \mathit{cryptoGarb}(\mathsf{bel}) \\
\mathit{cryptoGarb}(\mathsf{bel} \cup \{\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{u}, x))\}) = \mathit{cryptoGarb}(\mathsf{bel}) \\
\mathit{cryptoGarb}(\mathsf{bel} \cup \{\mathsf{Transmit}(a, b, \varphi)\}) = \mathit{cryptoGarb}(\mathsf{bel}) \\
\text{else } \mathit{cryptoGarb}(\mathsf{bel} \cup \{F\}) = \{F\} \cup \mathit{cryptoGarb}(\mathsf{bel})
\end{array}$$

After the application *cryptoGarb*, the model contains only realistic candidates for secrets. Hence the model described in Figure 5 results in a real candidate for confidentiality goals in the Needham Schroeder protocol:

$$\begin{array}{l}
\prec x \mid \mathsf{bel}_x = \{\mathsf{isNonce}(\mathsf{n}(z_1, x)), \mathsf{isNonce}(\mathsf{n}(z_2, y)), \\
\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{i}, x))\} \succ \\
\| \quad \prec y \mid \mathsf{bel}_y = \{\mathsf{isNonce}(\mathsf{n}(z_1, x)), \mathsf{isNonce}(\mathsf{n}(z_2, y)), \\
\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{i}, y))\} \succ \\
\| \quad \prec \mathrm{Public} \mid \mathsf{bel}_y = \{\mathsf{Agent}(x), \mathsf{Agent}(y)\} \succ
\end{array}$$

The remaining task is to find the groups of agents and the subsets of credentials that share common secrets.

3.4 Confidentiality groups

A security protocol partitions the participants into temporary protection domains. A *temporary protection domain* (TPD) is the largest group of agents sharing a set of secrets. The smallest TPD is the agent's private beliefs, while the largest TPD is secrets shared by

every participant in the protocol. If there are n roles in a protocol P, then the maximal number of groups that share secrets is $2^n - 1$. A group is represented as a agent, its name is the collection of names from its constituents: If x and y are names of groups then $x^\frown y$ denotes their composition, hence a new group is constructed of two agents $\prec x \mid \mathsf{bel}_x \succ$ and $\prec y \mid \mathsf{bel}_y \succ$, the composition of names and an operator f on the two sets of beliefs:

$$\prec x^\frown y \mid f(\mathsf{bel}_x, \mathsf{bel}_y) \succ$$

The name composition operator is defined as an abelian monoid, with the empty name ϵ as the identity element. The main function for finding confidentiality groups is given by the following definition:

Definition 3 *Let $\mathscr{C}$ denote a configuration, then*

$$\begin{array}{ll}
(i) & \mathit{Groups}(\mathscr{C}) = \\
 & \quad \mathit{privateBeliefs}(\mathscr{C}) \parallel \mathit{Groups}(\mathscr{C}, 0, \#(\mathscr{C}), \mathscr{C}) \\
(ii) & \mathit{Groups}(\mathscr{C}, N, N, \mathscr{C}') = \mathfrak{E} \\
(iii) & \mathit{Groups}(\mathscr{C}, N, M, \mathscr{C}') = \\
 & \mathit{removeEmpty}(\mathit{disjoint}(\mathit{shared}(\mathscr{C}, \mathscr{C}'))) \\
 & \parallel \mathit{Groups}(\mathscr{C}, N+1, M, \mathit{removeEmpty}(\mathit{shared}(\mathscr{C}, \mathscr{C}')))
\end{array}$$

The collection function for groups can be explained as follows: A configuration of agents $\mathscr{C}$ is taken as input, and then the private beliefs are constructed in addition to the initial call on the recursive *Groups*. The collection function takes four parameters, $\mathit{Groups}(\mathscr{C}, N, M, \mathscr{C}')$, the initial configuration of agents $\mathscr{C}$, the current cardinality of groups N, the total number of roles M, and the current collection of groups $\mathscr{C}'$ with shared beliefs. Thus initially (i), the function *Groups* is called with the basic agents $\mathscr{C}$, the current size of the group (which is equal to zero since no groups have been formed yet), the maximal size of a potential group - the number of agents in the collection $\#(\mathscr{C})$, and the basic current collection of groups $\mathscr{C}$ - which is just a collection of singleton groups consisting of the basic agents. The remaining sub-functions are defined as follows:

$$\begin{array}{l}
\mathit{shared}(\mathscr{C}, \mathscr{C}') = \{\prec x^\frown y \mid \ \mathsf{bel}_x \cap \mathsf{bel}_y \succ \mid \\
\qquad\qquad \prec x \mid \mathsf{bel}_x \succ \in \mathscr{C} \wedge \prec y \mid \mathsf{bel}_y \succ \in \mathscr{C}'\} \\
\mathit{disjoint}(\mathscr{C}) = \{\prec x \mid \mathsf{bel} \succ \mid \\
\qquad\qquad \mathsf{bel} = \mathsf{bel}_x \setminus \bigcup_{y \neq x \wedge \prec y \mid \mathsf{bel}_y \succ \in \mathscr{C}} \mathsf{bel}_y\} \\
\mathit{removeEmpty}(\mathscr{C}) = \{\prec x \mid \mathsf{bel}_x \succ \in \mathscr{C} \mid \mathsf{bel}_x \neq \varnothing\}
\end{array}$$

The function *shared* takes two collections of agents and finds all pairwise unions of groups, where the belief set of the joint group is the intersection of the base groups, *disjoint* takes a set of agents eliminates the beliefs of each agent shared by another agent, while eliminates every agent that has empty beliefs.

Finally, the TPD's of Needham Schroeder can be constructed by running *Groups* on the model from Section 3.3.

$$\begin{array}{l}
\prec x \mid \{\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{p}, x))\} \succ \\
\prec y \mid \{\mathsf{isKey}(\mathsf{key}(\mathsf{a}, \mathsf{p}, y))\} \succ \\
\prec x^\frown y \mid \{\mathsf{isNonce}(\mathsf{n}(z_1, x)), \mathsf{isNonce}(\mathsf{n}(z_2, y))\} \succ
\end{array}$$

To summarize the exposition so far, the process described in Figure 2 can be defined by the main *confidentiality goal detection* algorithm:

$$\begin{array}{ll}
\mathit{ConfGoals}(\mathsf{protocol}[\mu, N, \xi^{\mathcal{T}}, \xi^{\mathcal{A}}, \xi^{\mathcal{S}}, \Phi]) = & \\
\quad \mathit{Groups}(& (a) \\
\qquad \mathit{cryptoGarb}(& (b) \\
\qquad\quad \mathit{Sep}(\Phi, & (c) \\
\qquad\qquad \mathit{Coll}(& (d) \\
\qquad\qquad\quad \Re^{\star}_{\mathrm{A}}(\mathsf{protocol}[\mu, N, \xi^{\mathcal{T}}, \xi^{\mathcal{A}}, \xi^{\mathcal{S}}, \Phi])))))) & (e)
\end{array}$$

The protocol is refined (e), and the Henkin model is constructed (d). The plain text parts of the messages are removed from each of the agents, and then the cryptographic garbage is removed (b), in order to prepare for the final construction of the TPD's.

4 Deploying goal detection

In this section we present how goal detection apply to standard protocols, by giving a detailed analysis of the Kerberos protocol and a survey of applying the algorithm to the entire library of authentication protocols provided by Clark/Jacob. Finally we present an analysis of Lowe's attack on the Needham Schroeder public key protocol, and show that the method presented is even general enough to cover attack descriptions.

4.1 The Kerberos protocol (ver. 5)

The Kerberos protocol is widely used on several platforms. The protocol was originally designed to protect network services provided by Project Athena at Massachusetts Institute of Technology (MIT), but is recently used for a variety of purposes. Kerberos provides mutual authentication, both the user and the server verify each other's identity. The protocol is intricate, it uses two trusted servers, authentication server (A) and a ticket granting server (G), in addition to a standard client (C) and a service server (S). The protocol only uses symmetric keys - three keys should be established in advance: the key shared by the authentication server and the client K_{AC}, the one shared by the authentication server and the granting server K_{AG}, and finally between the granting server and the service

server K_{GS}. Two keys are freshly generated during the session, the keys shared by the client and the granting server K_{CG} and by the client and the service server K_{CS}. Additionally, the following two tickets are used:

$$T_1 = E[K_{AG} : C, G, K_{CG}, T_A^{\text{start}}, T_A^{\text{exp}}]$$
$$T_2 = E[K_{SG} : C, S, K_{CS}, T_G^{\text{start}}, T_G^{\text{exp}}]$$

The tickets contain initiation time T^{start} and expiration time T^{exp} that limit the interval over which the ticket is considered valid - one pair for both ticket generators. Additionally the main protocol includes two lifelines (T_C^{life} and T_C^{life2}) and two nonces (N_C and N'_C), each generated by the client. The specification of the protocol is as follows:

$K_1\ C \longrightarrow A : C, G, T_C^{\text{life}}, N_C$
$K_2\ A \longrightarrow C : C, T_1, E[K_{AC} : G, K_{CG}, T_A^{\text{start}}, T_A^{\text{exp}}, N_C]$
$K_3\ C \longrightarrow G : S, T_C^{\text{life2}}, N'_C, T_1, E[K_{CG} : C, T_C]$
$K_4\ G \longrightarrow C : C, T_2, E[K_{CG} : S, K_{GS}, T_G^{\text{start}}, T_G^{\text{exp}}, N'_C]$
$K_5\ C \longrightarrow S : T_2, E[K_{CS} : C, T'_C]$
$K_6\ S \longrightarrow C : E[K_{CS} : T'_C]$

Seven TPDs are induced by the Kerberos protocol, involving the following potential confidentiality goals:

$G_1 :$ Secret$(\{A, C\}, \{K_{AC}\})$
$G_2 :$ Secret$(\{A, G\}, \{K_{AG}\})$
$G_3 :$ Secret$(\{C, G\}, \{T_C\})$
$G_4 :$ Secret$(\{C, S\}, \{T'_C\})$
$G_5 :$ Secret$(\{G, S\}, \{K_{GS}\})$
$G_6 :$ Secret$(\{A, C, G\}, \{K_{CG}, T_A^{\text{exp}}, T_A^{\text{start}}\})$
$G_7 :$ Secret$(\{C, G, S\}, \{K_{CS}, T_G^{\text{exp}}, T_G^{\text{start}}\})$

The timestamps T_C is only shared by the authentication server A and the client while T'_C is only shared by A and the granting server G. The keys established before the protocol runs should be kept secret during the execution as specified in G_1, G_2, and G_5. The freshly constructed session keys belong to larger groups, respectively G_6, and G_7. Each group shares the ticket start time and expiration time originating from the ticket generated from the authentication server and the granting server respectively.

4.2 Investigating libraries of protocols

We investigated standard libraries, like the Clark/Jacob library [2] and compared it with the goals specified in the online libraries SPORE and AVISPA [2] using the module for automated construction of confidentiality goals. Table 1 reports data of the resulting configuration: the number of *groups* generated, the total number of cryptographic *keys* that should be kept secret, other *credentials*, like nonces, timestamps, strings, etc., and finally the number of rewrites performed by Maude to arrive at the final state.

[2] `http://avispa-project.org/`

Protocol name	Groups	Keys	Cred.	Rew.
Symmetric key				
ISO One pass unilateral	1	1	2	1293
ISO Two pass unilateral	1	1	1	1673
ISO Two pass mutual	1	1	4	2684
ISO Three pass mutual	1	1	3	3458
Non-reversible func.	2	2	4	3688
Andrew Secure RPC	1	2	3	3923
ISO One pass unilat. CCF	2	1	2	1134
ISO Two pass unilat. CCF	1	1	1	1673
ISO Two pass mutual CCF	1	1	2	2584
ISO Three pass CCF	1	1	2	3266
Andrew Secure	1	2	5	3922
Symmetric key with TTP				
Needham Schroeder sym.	4	3	2	6665
Denning Sacco	3	3	1	5074
Otway Rees	3	3	2	7910
Wide Mouthed Frog	3	3	2	3730
Yahalom	3	3	1	6758
Carlsen's Secret Key	1	2	0	7054
ISO Four pass	4	3	5	13655
ISO Five pass	4	3	5	14754
Woo Lam Pi	2	2	0	3801
Kerberos (ver. 5)	7	5	6	28167
Neuman Stubblebine	3	3	1	12694
Kehne Langendorfer	4	4	1	14205
Kao Chow	4	4	1	11051
Public key				
ISO PKI One pass	3	1	1	2164
ISO PKI Two pass	3	1	1	3375
ISO PKI Two pass mutual	3	1	2	6968
ISO Three pass	3	1	2	7705
ISO Two pass Parallel	3	2	2	11069
Bilateral Key Exchange	3	3	4	4094
Needham Schroeder PKI	4	3	2	9188
SPLICE/AS	5	5	4	16277
Denning Sacco PKI	5	5	2	9980
CCITT X.509	3	2	8	8702
Shamir Rivest Adelman	2	1	1	1789
Encrypted Key Exchange	2	3	2	4295
Davis Swick	4	4	4	11440

Table 1. Automated detection of goals in Clark/Jacob.

More goals than those occurring in the AVISPA library were found.

4.3 The attack on Needham Schroeder

The method proposed is general, and even covers attack descriptions. Gavin Lowe's attack on the authentication part of the Needham Schroeder protocol [7] can be specified as follows:

$$
\begin{array}{lcl}
(\alpha_1) & A \longrightarrow I & : E(PK_I : N_A, A) \\
(\beta_1) & I(A) \longrightarrow B & : E(PK_B : N_A, A) \\
(\beta_2) & B \longrightarrow I(A) & : E(PK_A : N_A, N_B) \\
(\alpha_2) & I \longrightarrow A & : E(PK_A : N_A, N_B) \\
(\alpha_3) & A \longrightarrow I & : E(PK_I : N_B) \\
(\beta_3) & I(A) \longrightarrow B & : E(PK_B : N_B)
\end{array}
$$

When automated goal-detection is applied to Lowe's attack, we obtain the following groups that share beliefs:

$$
\begin{array}{l}
\text{Private}(I, K_I^{-1}) \\
\text{Private}(A, K_A^{-1}) \\
\text{Private}(B, K_B^{-1}) \\
\text{Secret}(\{A, B, I\}, \{N_A, N_B\})
\end{array}
$$

Each agent possesses each private key only, including the attacker I. Lowe's attack is an attack on authenticity. Yet, within the confidentiality setting, we observere that the algorithm shows that the original TPD of the uncompromized session is extended to include the attacker. The difference to the intended specification of the Needham Schroeder protocol is that the confidentiality group consisting of Alice of Bob, sharing the two nonces N_A and N_B, is extended to include the attacker I too.

5 Conclusion

We have shown how to derive potential security goals directly from a specification, without any human interaction. The author is currently using the algorithm in enterprises, analysing large scale PKI applications with high level of trust and complex cryptographic protocols involved.

An obvious yet challenging extension of the method described in this paper is to find candidates for integrity and authenticity goals in an automated way.

References

[1] D. Basin, S. Mödersheim, and L. Viganò. OFMC: A symbolic model checker for security protocols. *International Journal of Information Security*, 4(3):181–208, June 2005.

[2] J. Clark and J. Jacob. A Survey of Authentication Protocol Literature, 1997. Version 1.0, Unpublished Report, University of York, `http://cs.york.ac.uk/~jac/papers/drareview.ps.gz`.

[3] B. Donovan, P. Norris, and G. Lowe. Analyzing a Library of Security Protocols using Casper and FDR. In *Proceedings of the Workshop on Formal Methods and Security Protocols*. IEEE Computer Society Press, 1999.

[4] A. M. Hagalisletto. Attacks are Protocols Too. In *Proceedings of The Second International Conference on Availability, Reliability and Security (IEEE ARES 2007)*, pages 1197 – 1206. Workshop WAIS 2007.

[5] A. M. Hagalisletto. Using the mobile phone in two-factor authentication. In E. Bajart, H. Muller, and T. Strang, editors, *UbiComp 2007 Workshop Proceedings*, pages 469–474. First International Workshop on Security for Spontaneous Interaction.

[6] A. M. Hagalisletto. Automated Refinement of Security Protocols. In *Workshop SSN in the Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium*, 2006.

[7] G. Lowe. Breaking and Fixing the Needham-Schroeder Public-Key Protocol Using FDR. In *Proceedings of the Second International Workshop on Tools and Algorithms for Construction and Analysis of Systems*, pages 147–166. Springer-Verlag, 1996.

[8] G. Lowe. A hierarchy of authentication specifications. In *PCSFW: Proceedings of The 10th Computer Security Foundations Workshop*. IEEE Computer Society Press, 1997.

[9] R. M. Needham and M. D. Schroeder. Using Encryption for Authentication in Large Networks of Computers. *Commun. ACM*, 21(12):993–999, 1978.

[10] J. R. Shoenfield. *Mathematical Logic*. Association for symbolic Logic, 1967. Reprint 2000 of original manuscript.

[11] D. X. Song, S. Berezin, and A. Perrig. Athena: A Novel Approach to Efficient Automatic Security Protocol Analysis. *Journal of Computer Security*, 9(1/2):47–74, 2001.

[12] M. Turuani. The CL-Atse Protocol Analyser. In *RTA*, pages 277–286, 2006.