DELP: Dynamic Epistemic Logic for Security Protocols*

Ioana Leustean  
Faculty of Mathematics and Computer Science  
University of Bucharest  
Academiei 14, 010014 Bucharest, Romania  
ioana.leustean@unibuc.ro

Bogdan Macovei  
Research Center for Logic, Optimization and Security (LOS)  
Department of Computer Science  
Faculty of Mathematics and Computer Science  
University of Bucharest  
Academiei 14, 010014 Bucharest, Romania  
bogdan.macovei@s.unibuc.ro

Abstract—The formal analysis of security protocols is a challenging field, with various approaches being studied nowadays. The famous Burrows-Abadi-Needham Logic was the first logical system aiming to validate security protocols. Combining ideas from previous approaches, in this paper we define a complete system of dynamic epistemic logic for modeling security protocols. Our logic is implemented, and few of its properties are verified, using the theorem prover Lean.

Index Terms—Dynamic Epistemic Logic, BAN Logic, Security protocols, Lean Theorem Prover

I. INTRODUCTION

This paper presents DELP, a dynamic epistemic logic for analysing security protocols. In order to define our logic, we combine the epistemic approach to authentication from [5], the expectation semantics from [8] and the operational semantics for security protocols from [7].

Our main contributions are: (i) the definition of DELP as a sound and complete system with respect to an expectation semantics representing the adversary knowledge; (ii) the implementation of DELP in the theorem prover Lean. Consequently, using Lean: (iii) we defined translations in DELP for a few inference rules of the Burrows-Abadi-Needham (BAN) logic [4] and we proved their soundness, (iv) we defined the Needham-Schroeder authentication protocol as a theory in DELP and we verified a few security claims.

Section II presents the Needham-Schroeder security protocol and recalls the formal approaches from [5], [8] and [7]. In Section III we define the system DELP and we prove its properties. Section IV contains the Lean implementation of DELP. Few deduction rules of the BAN Logic are defined in DELP and their soundness is proved using the Lean implementation. In Section V we study the Needham-Schroeder authentication protocol using DELP and its Lean implementation. The last section contains conclusions and further developments.

II. PRELIMINARIES: FORMAL ANALYSIS OF SECURITY PROTOCOLS

A security protocol is defined as a set of rules and conventions that determine the exchange of messages between two or more agents in order to implement a security service. The protocol must be unambiguous and must allow the description of several roles, so that an agent can perform a certain role at a certain protocol round. An example of a security protocol, which we will mention and use in this paper, is the Needham-Schroeder protocol.

A. The Needham-Schroeder symmetric key protocol for key exchange

The protocol specification for three agents is as follows:

\[
\begin{align*}
A & \rightarrow S : A, B, N_a \\
S & \rightarrow A : \{N_a, B, K_{ab}, \{K_{ab}, A\}_{K_{ba}}\}_{K_{as}} \\
A & \rightarrow B : \{K_{ab}, A\}_{K_{as}} \\
B & \rightarrow A : \{N_b, K_{ab}\}_{K_{ab}} \\
A & \rightarrow B : \{N_b - 1\}_{K_{ab}}
\end{align*}
\]

A step-by-step description of the protocol is:

1) Alice initiates the connection with the Server, sending who she is, with whom she wants to communicate and a nonce;
2) the Server sends - encrypted with the common key between Alice and Server - the nonce generated by Alice, the identity of Bob and the communication key between Alice and Bob, to which is added a message that only Bob can decrypt (being encrypted with the communication key between Bob and Server), which contains the communication key shared by Alice and Bob; in this way, Alice cannot read the message sent by Server to Bob;
3) Alice sends Bob the message that it could not decrypt, received from the Server;
4) Bob decrypts the message, and sends Alice a nonce encrypted with the common key between Alice and Bob;
5) Alice receives Bob’s message, decypts it, and resends it, applying a simple function to it - in this case, it decrements it. This step is useful in two situations: it is a first protection on a reply attack and it shows that the agents are still alive in the session.

* All authors contributed equally to this work.
B. BAN Logic

We will briefly present the BAN logic, based on [4]. The mathematical system contains the following sets: a set of participating agents in communication protocol sessions - named, generally, using capital letters of the beginning of the alphabet (A, B, ...), a set of keys - named, generally, $K_{a,b}$ for the public key between agents $A$ and $B$, $K_a$ for $A$'s public key and $K_a^{-1}$ for $A$'s secret key, and a set of messages - named, generally, using capital letters of the end of the alphabet (X, Y, ...). An encrypted message is denoted by writing $\{X\}_k$, meaning that the message $X$ is encrypted with the key $k$.

The specific formulas introduced in BAN logic are the following:

- $P \equiv X$: the agent $P$ believes the message $X$;
- $P \prec X$: the agent $P$ sees or receives $X$;
- $P \sim X$: the agent $P$ once said or sends $X$;
- $P \Rightarrow X$: the agent $P$ controls $X$ or have jurisdiction over $X$;
- $\#(X)$: $X$ is a nonce;
- $P \dashv\vdash Q$: the agents $P$ and $Q$ shares the communication key $k$;
- $k \rightarrow P$: $k$ is $P$’s public key;
- $\{X\}K$: $X$ is encrypted with the key $k$;
- $\prec X$: $X$ is encrypted with the common secret $Y$.

In the sequel we recall only two deductions rules, we refer to [4] for the full deduction system.

The **Message Meaning Rule**, formally defined by

$$
\frac{P \equiv Q \prec K \quad P \prec \{X\}_K}{P \equiv Q \sim X}
$$

(1)

can be read as follows: if agent $P$ believes that he has a communication key $K$ with agent $Q$, and agent $P$ receives a message $X$ encrypted under $K$, then $P$ believes that the encrypted message was sent by $Q$.

The **Jurisdiction rule**, formally defined by

$$
\frac{P \equiv Q \Rightarrow X \quad P \equiv Q \equiv X}{P \equiv X}
$$

(2)

can be read as follows: if agent $P$ believes that agent $Q$ has jurisdiction over a message $X$ and, furthermore, agent $P$ believes that $Q$ believes $X$, then $P$ believes $X$.

C. An approach based on epistemic logic

In this subsection, we recall the main ideas from [5], and we refer to [9] for a comprehensive presentation of dynamic epistemic logic.

In this paper, there are defined $K$ (the set of communication keys), $N$ (the set of nonces), $T$ (the set of plain texts) and $\Phi$ (the set of formulas). The BNF specification of the language is:

$$
s ::= s \mid x
$$

$$
m ::= t \mid k \mid n \mid i \mid (m_1, m_2) \mid \{m\}_k \mid \varphi
$$

$$
\varphi ::= p \mid \text{sent}(s) \mid \text{recv}(s) \mid \text{extract}(m) \mid \sim \varphi \mid \varphi_1 \land \varphi_2 \mid K_{i\varphi} \mid \varphi \land \varphi_2 \mid K_{i\varphi} \mid \varphi \land \varphi_2 \mid K_{i\varphi} \mid \varphi \land \varphi_2 \mid K_{i\varphi} \mid \varphi \land \varphi_2
$$

where $p$ is an atomic formula, $i$ is an arbitrary agent, $m$ is an arbitrary message, $t \in T$, $k \in K$, $n \in N$, $\alpha \in [0, 1]$ a probability, $s$ a string, $x$ a variable over strings and $\varphi \in \Phi$.

For semantics, the models are

$$
I = (R, \pi, C, \{\mu_C\}_{C \in C})
$$

where $R$ is a protocol rounds system, $\pi$ is an evaluation function, $C$ is a partition of $R$, and for every $C \in C$, the measure $\mu_C$ is the distribution probability over rounds in $C$.

The inductive interpretation of formulas in these models are:

- $(I, r, m) \models \varphi \iff \pi_{r(m)}(p)$ is true
- $(I, r, m) \models \sim \varphi \iff (I, r, m) \not\models \varphi$
- $(I, r, m) \models \varphi_1 \land \varphi_2 \iff (I, r, m) \models \varphi_1$ and $(I, r, m) \models \varphi_2$
- $(I, r, m) \models K_{i\varphi} \iff$ for all $(r', m') \sim_{r} (r, m)$, we have $(I, r', m') \models \varphi$
- $(I, r, m) \models \varnothing \varphi \iff (I, r, m + 1) \models \varphi$
- $(I, r, m) \models \Box \varphi \iff m = 0$ or $(I, r, m - 1) \models \varphi$
- $(I, r, m) \models \Box \varphi \iff$ for all $m' \geq m, (I, r, m') \models \varphi$
- $(I, r, m) \models \square \varphi \iff$ for all $m', m \leq m', (I, r, m') \models \varphi$
- $(I, r, m) \models \mu_{r, m,}((r', m') \mid (I, r', m') \models \varphi) \cap K_{i\varphi} \cap (C)(r) \geq \alpha$
- $(I, r, m) \models \exists x \varphi \iff$ exists $s$ string, $(I, r, m) \models \varphi[s/x]$

D. An approach based on operational semantics

From [7], the main point of interest is the terms deduction system. In this formal system we have terms (roles, messages, keys and nonces), variables over Var, Fresh and Role sorts, functions symbols (in Func), the protocols specifications and a labeled transition system for the execution of the protocols.

Having $\Gamma$ a knowledge set, the term deduction rules are:

- if $t \in \Gamma$, then $\Gamma \vdash t$
- $\Gamma \vdash t_1$ and $\Gamma \vdash t_2$ if and only if $\Gamma \vdash (t_1, t_2)$
- if $\Gamma \vdash t$ and $\Gamma \vdash k$, then $\Gamma \vdash \{k\}_t$
- if $\Gamma \vdash \{t\}_k$ and $\Gamma \vdash k^{-1}$, then $\Gamma \vdash t$
- if $\Gamma \vdash t_i$, $1 \leq i \leq n$, then $\Gamma \vdash f(t_1, t_2, ..., t_n)$, where $f$ is a function symbol of Func, with the arity $n$.

E. An approach based on expectation models

In this subsection, we will present the main results of [8], that we will use in the next section to prove the completeness theorem of our system.

In this paper there are introduced two sets, $I$ - the set of agents and $P$ - the set of formulas. For interpreting formulas there are used Kripke models, $M = (S, \sim, V)$, where $S$ is the
set of accessible world, \(\sim\) is the accessibility relation between worlds and \(V\) is the evaluation function, \(V : P \rightarrow P(S)\).

There are an action set - \(\Sigma\) - and a language of observations - \(L_{\text{obs}}\). The BNF grammar of the actions is:

\[
\pi ::= \delta | \varepsilon | \alpha | \pi \cdot \pi | \pi + \pi | \pi^* \tag{3}
\]

where \(\delta\) is an empty set of observations, \(\varepsilon\) is the empty string and \(\alpha \in \Sigma\).

The observations set is denoted by \(L(\pi)\) and is inductively defined as:

\[
\begin{align*}
L(\delta) &= \emptyset \tag{4} \\
L(\varepsilon) &= \{\varepsilon\} \tag{5} \\
L(\alpha) &= \{\alpha\} \tag{6} \\
L(\pi \cdot \pi') &= \{wv | w \in L(\pi) \text{ and } v \in L(\pi')\} \tag{7} \\
L(\pi + \pi') &= L(\pi) \cup L(\pi') \tag{8} \\
L(\pi^*) &= \{\varepsilon\} \cup \bigcup_{n>0} (L(\pi \cdots \pi)) \tag{9}
\end{align*}
\]

An epistemic model defined with this observations is an epistemic expectation model \(M = (S, \sim, V, Exp)\), where \(Exp : S \rightarrow L_{\text{obs}}\) is a function that maps every state from \(S\) to an observation \(\pi\) for which \(L(\pi) \neq \emptyset\). The logical formulas are defined using the following BNF description:

\[
\varphi ::= p \neg \varphi | \varphi \wedge \psi | K_i \varphi | [\pi] \varphi \tag{10}
\]

where \(p \in P\), \(i \in I\) and \(\pi \in L_{\text{obs}}\).

An important result from this paper is the \textit{bisimilarity}; a binary relation \(R\) between two epistemic expectations models \(M = (S, \sim, V, Exp)\) and \(N = (S', \sim', V', Exp')\) is called bisimilarity if for every \(s \in S\) and \(s' \in S'\), if we have \((s, s') \in R\), then:

**Propositional invariance:** \(V(s) = V'(s')\) \tag{11}

**Observation invariance:** \(L(Exp(s)) = L(Exp(s'))\) \tag{12}

**Zig:** \(s \sim_t t \in M \implies \exists t' \in N\) \tag{13}

such that \(s' \sim_t t'\) and \(tRt'\)

**Zag:** \(s' \sim_t^i t' \in N \implies \exists t \in M\) \tag{14}

such that \(s \sim_t^i t\) and \(tRt'\)

The article also introduces the \textit{bisimilarity invariance}: for two epistemic states \(M, s\) and \(N, s'\), the following two statements are equivalent:

\[
i) \quad M, s \leftrightarrow N, s' \tag{15}
\]

\[
ii) \quad \text{for all } \varphi : M, s \models \varphi \leftrightarrow N, s' \models \varphi \tag{16}
\]

**Updated models.** Let \(w\) be an observation over \(\Sigma\), and \(M = (S, \sim, V, Exp)\) an epistemic expectation model. The, the updated model is denoted with \(M[w = (S', \sim', V', Exp')]\), where \(S' = \{s \mid L(Exp(s) - w) \neq \emptyset\}\), \(\sim' = \sim \circ 1 \times 1 \circ \sim\), \(V' = V|_{S'}\) and \(Exp'(s) = Exp(s) - w\), where \(\pi - w = \{v \mid vw \in L(\pi)\}\).

**Temporal models.** Let \(M = (S, \sim, V, Exp)\) be an epistemic expectation model. Then the temporal model is called \(ET(M)\) and is defined as \(ET(M) = (H, \rightarrow_a, \sim_a', V')\), where

\[
H = \{(s, w) \mid s \in S, w \in \varepsilon \text{ or } w \in L(Exp(s))\}, \quad (s, w) \rightarrow_a (t, v) \iff s = t \text{ and } v = wa, a \in \Sigma, \quad (s, w) \sim_a (t, v) \iff s \sim_t t \text{ and } w = v \text{ and } p \in V'(s, w) \iff p \in V(s).
\]

Using temporal models, it is proved in this paper that \(M, s \models \varphi \iff ET(M), (s, \varepsilon) \models E_{F D L} \varphi\), so the system is complete by the completeness of dynamic epistemic logic.

### III. DELP - Dynamic Epistemic Logic for Protocols

In order to define our system, we firstly recall the \textit{dynamic epistemic logic} [9], \textit{Dynamic epistemic logic} is a dynamic logic [6] to which is added the knowledge operator \(K\) from \textit{epistemic logic}. There are two sets, \(I\) - the set of programs, and \(\Phi\) - the set of formulas, with \(\Pi_0\) - set of atomic programs, and \(\Phi_0\) - set of atomic formulas. The language is described using the following BNF:

\[
\varphi ::= p \neg \varphi | \varphi \wedge \psi | K_i \varphi | [\pi] \varphi \tag{17}
\]

where \(p \in \Phi_0\), \(\varphi \in \Phi\), \(i\) is an arbitrary agent and \(\alpha \in I\).

The evaluation models are \textit{Kripke models} \(M = (R, \sim, V)\), where \(R\) is the finite set of accessible worlds, \(\sim\) is the accessibility relationship between worlds, and \(V\) is the evaluation function from dynamic logic: for a formula \(\varphi \in \Phi\), \(V(\varphi) \subseteq R\), and for a program \(\alpha \in I\), \(V(\pi) \subseteq R \times R\).

Interpretation of formulas in this models are inductively defined as:

\[
M, s \models p \iff v \in V(s) \tag{18}
\]

\[
M, s \models \varphi \wedge \psi \iff M, s \models \varphi \text{ and } M, s \models \psi \tag{19}
\]

\[
M, s \models \neg \varphi \iff M, s \not\models \varphi \tag{20}
\]

\[
M, s \models K_i \varphi \iff \text{for all } t \text{ such that } s \sim_t t, \quad \text{we have } M, t \models \varphi \tag{21}
\]

\[
M, s \models [\alpha] \varphi \iff \text{for all } t \in R \text{ such that } (s, t) \in V(\alpha), \quad \text{we have } M, t \models \varphi \tag{22}
\]

We also have the following operators for programs:

\[
V(\alpha_1 \cup \alpha_2) = V(\alpha_1) \cup V(\alpha_2) \tag{23}
\]

\[
V(\alpha_1; \alpha_2) = V(\alpha_1) \circ V(\alpha_2) \tag{24}
\]

\[
V(\alpha^*) = \bigcup_{n \geq 0} V(\alpha)^n \tag{25}
\]

The deductive system contains all instances of propositional tautologies to which are added the following axioms:

\[
K_a(\varphi \rightarrow \psi) \rightarrow (K_a \varphi \rightarrow K_a \psi) \tag{26}
\]

\[
K_a \varphi \rightarrow \varphi \tag{27}
\]

\[
K_a \varphi \rightarrow K_a K_a \varphi \tag{28}
\]

\[
\neg K_a \varphi \rightarrow K_a \neg K_a \varphi \tag{29}
\]

\[
[a(\varphi \rightarrow \psi) \rightarrow ([a] \varphi \rightarrow [a] \psi)] \tag{30}
\]

\[
[a(\varphi \wedge \psi) \leftrightarrow [a] \varphi \wedge [a] \psi] \tag{31}
\]

\[
[a \cup \beta] \varphi \leftrightarrow [a] \varphi \wedge [\beta] \varphi \tag{32}
\]

\[
[a; \beta] \varphi \leftrightarrow [a][\beta] \varphi \tag{33}
\]
Deductive rules are *modus ponens, generalization* from dynamic logic and *necessity* from epistemic logic:

\[(MP) \frac{\varphi \rightarrow \psi}{\varphi} ; \ (GEN) \frac{\varphi}{[a]\varphi} ; \ (NEC) \frac{\varphi}{K_i\varphi}\]

This system is known as the PA-system in [9], and it is proved sound and complete [9, p. 187-188].

A. DELP

In this subsection we define DELP, a logic based on dynamic epistemic logic, enriched with a set of actions collected during the execution of the protocol and a grammar for messages, together with a system of deduction for knowledge based on actions.

1) Syntax: Let \( \text{Agent} \) be the set of agents and let \( \text{Func} \) be a set of (encryption) functions. We consider the sets \( \Phi \) and \( \Pi \) like in dynamic epistemic logic, with \( \Phi_0 \) the set of atomic formulas, and \( \Pi_0 \) defined by

\[\Pi_0 := \{ \text{send}_i, \text{recv}_i \}_{i \in \text{Agent}}\]

The elements of \( \Pi_0 \) are protocols actions: we read \( \text{send}_i \) as "the agent \( i \) sends" and we read \( \text{recv}_i \) as "the agent \( i \) receives".

In the following we define messages and formulas. In a security protocol, a message contains clear texts, keys, nonces, and agents identities. The possible operations are messages concatenation and messages encryption. Following [7], the grammar for messages is:

\[m ::= \text{text}(m) \mid \text{key}_m(i, j) \mid \text{nonce}(m) \mid \text{agent}(i) \]

where \( i, j \in \text{Agent} \) and \( f \in \text{Func} \). In the sequel we will use \( t \) for texts, \( k \) for keys, \( n \) for nonces and \( i, j \) for agents. Based on [7], we define the following deductive system on messages:

\[\frac{\text{nonce}(m)}{\text{key}_m(i, j) \text{ nonce}(m)} \quad \frac{\text{key}_m(j, i)}{m_1 \text{ nonce}(m)} \quad \frac{m_2 \text{ nonce}(m)}{(m_1, m_2)}\]

\[\frac{t}{\{t\}_k} \quad \frac{k}{\{t\}_k} \quad \frac{t, t_1, t_2, \ldots, t_n}{f(t_1, t_2, \ldots, t_n)}\]

Finally, we are able to define the DELP formulas:

\[\varphi ::= p \mid \neg \varphi \mid \varphi \rightarrow \varphi \mid K_i\varphi \mid [a]\varphi \mid @\mu\]

2) Semantics: The models that we use are Kripke models like in dynamic epistemic logic, \( \mathcal{M} = (R, \sim, V) \) which we extend with \( \text{Exp} \) set, a knowledge set with information collected from protocol runs.

**Definition 1.** Let \( \mathcal{M} = (R, \sim, V, \text{Exp}) \) be a DELP model, where

1) \( R \) is the finite set of accessible worlds;

2) \( \sim := \bigcup_{i \in \text{Agent}} \sim_i \) represents the accessibility relationship between worlds, based on epistemic relation;

3) \( V \) is the evaluation function from dynamic logic: \( V(\varphi) \subseteq R \) for any \( \varphi \in \Phi \), and \( V(\alpha) \subseteq R \times R \), for any \( \alpha \in \Pi \);

4) \( \text{Exp} \) is the knowledge set: for any \( s \in R \), \( \text{Exp}(s) \) represents the set of all knowledge inferred up to \( s \)-th round of the protocol;

5) for any agent \( i \), \( V(\text{send}_i) \subseteq \sim_i \) and \( V(\text{recv}_i) \subseteq \sim_i \).

Having this models, we can interpret \( @\mu \) formula as:

\[\mathcal{M}, s \models @\mu \iff \mu \in \text{Exp}(s)\]

The other formulas have the interpretation from the dynamic epistemic logic:

\[\mathcal{M}, s \models p \iff v \in V(s)\]

\[\mathcal{M}, s \models \varphi \land \psi \iff \mathcal{M}, s \models \varphi \quad \text{and} \quad \mathcal{M}, s \models \psi\]

\[\mathcal{M}, s \models \neg \varphi \iff \mathcal{M}, s \models \varphi\]

\[\mathcal{M}, s \models K_i\varphi \iff \text{for all } t \text{ such that } s \sim_i t, \text{ we have } \mathcal{M}, t \models \varphi\]

\[\mathcal{M}, s \models [a]\varphi \iff \text{for all } t \in R \text{ such that } (s, t) \in V(\alpha), \text{ we have } \mathcal{M}, t \models \varphi\]

3) Deductive system: The deductive system contains all instances of propositional tautologies to which are added the following axioms from dynamic epistemic logic:

\[K_a(\varphi \rightarrow \psi) \rightarrow (K_a\varphi \rightarrow K_a\psi)\]

\[K_a\varphi \rightarrow \varphi\]

\[K_a\varphi \rightarrow K_aK_a\varphi\]

\[\neg K_a\varphi \rightarrow K_a\neg K_a\varphi\]

\[\neg [a]\varphi \rightarrow \neg [a]\neg \varphi \rightarrow [a]\varphi \rightarrow [a][\alpha] \psi\]

\[\neg [a; \beta]\varphi \rightarrow [a][\alpha] \varphi \land [a][\alpha] \psi\]

In addition, we have the following specific axiom, that is necessary to have a correspondence between states; if the agent \( i \) performs an action within the protocols (sends or receives a message), then he knows the message:

\[[\text{send}_i]@m \lor [\text{recv}_i]@m \rightarrow K_i@m\]

The soundness of this system is given by the soundness of the dynamic epistemic logic [9, p. 187-188], and all that remains for us to prove is the soundness of the specific axiom.

**Lemma 1.** Axiom \([\text{send}_i]@m \lor [\text{recv}_i]@m \rightarrow K_i@m\) is sound.

**Proof.** Let \( \mathcal{M} = (R, \sim, V, \text{Exp}) \) be a DELP model and \( s \in R \) an arbitrary state.

\[\mathcal{M}, s \models [\text{send}_i]@m \iff \text{for all } t \text{ such that } (s, t) \in V(\text{send}_i), \text{ we have that } \mathcal{M}, t \models @m\]
but $V(\text{send}_i) \subseteq \sim_i$, so
\[ \mathcal{M}, s \models [\text{send}_i]@m \iff \text{for all } t \text{ such that } (s, t) \in \sim_i, \]
we have that $\mathcal{M}, t \models @m$
\[ \iff \mathcal{M}, s \models K_i@m \]
\hfill $\square$

4) Completeness: In order to prove the completeness of DELP, we follow ideas from [8] and general results from dynamic epistemic logic.

**Definition 2. [Restricted model]** Let $\mu$ be a message and $\mathcal{M} = (R, \sim, V, \text{Exp})$ a DELP model. Then, the restricted model is defined as
\[ M|_\mu = (R', \sim', V', \text{Exp}') \]
where $R' = \{ s | \text{Exp}(s) - \mu \neq 0\}$, $\sim' = \sim |_{R' \times R'}$, $V' = V|_{R'}$, and $\text{Exp}'(s) = \text{Exp}(s) - \mu$.

**Definition 3. [Temporal model]** Let $\mathcal{M} = (R, \sim, V, \text{Exp})$ be a DELP model. We define
\[ ET(\mathcal{M}) = (H, \rightarrow, \sim', V') \]
where
- $H = \{ (s, m) | s \in R, m \in \text{Exp}(s) \}$;
- $(s, m) \rightarrow (s', m')$ if and only if $s = s'$ and $\{ m \} \models m'$ using the deduction system (37);
- $(s, m) \sim' (s', m')$ if and only if $s \sim s'$ and $m \equiv m'$ where $\equiv$ is the logic equivalence;
- $p \in V'(s, m)$ if and only if $p \in V(s)$.

Having $\mathcal{N} = ET(\mathcal{M})$ a temporal model, we inductively define the following interpretation of formulas:
\[ \mathcal{N}, w \models p \iff p \in V(w) \]
\[ \mathcal{N}, w \models \neg \varphi \iff \mathcal{N}, w \not\models \varphi \]
\[ \mathcal{N}, w \models \varphi \land \psi \iff \mathcal{N}, w \models \varphi \text{ and } \mathcal{N}, w \models \psi \]
\[ \mathcal{N}, w \models K_i \varphi \iff \text{for all } v \in \mathcal{N}, \text{ if } w \sim_i v, \]
then $\mathcal{N}, v \models \varphi$
\[ \mathcal{N}, w \models [\alpha] \varphi \iff \text{for all } \mu \in \text{Exp}(\alpha), \]
$w \rightarrow v$ implies $\mathcal{N}, w \models \varphi$

**Definition 4. [Bisimilarity]** Based on [8, Def. 11], we have that the binary relation $\rho \subseteq M \times N$, for two DDEL models $\mathcal{M} = (R, \sim, V, \text{Exp})$ and $\mathcal{N} = (R', \sim', V', \text{Exp}')$ is called bisimilarity if for any $v \in R$ and $v' \in R'$, if we have $v \rho v'$, then:

- **Propositional invariance**
\[ V(v) = V'(v') \]

- **Observation invariance**
\[ \text{Exp}(v) = \text{Exp}(v') \]

- **Zig** $v \sim_i w \in \mathcal{M} \implies \text{exists } w' \in \mathcal{N}$
\[ \text{such that } v' \sim_i w' \text{ and } w \rho w' \]

- **Zag** $v' \sim_i w' \in \mathcal{N} \implies \text{exists } w \in \mathcal{M}$
\[ \text{such that } v \sim_i w \text{ and } w \rho w' \]

**Theorem 1. [Bisimilarity invariance]** For two DDEL states $\mathcal{M}, v$ and $\mathcal{N}, v'$, the following two statements are equivalent:
\[ (i) \mathcal{M}, v \leftrightarrow \mathcal{N}, v' \]
\[ (ii) \text{for all } \varphi: \mathcal{M}, v \models \varphi \iff \mathcal{N}, v' \models \varphi \]

The proof is the same as [8, Prop. 12].

**Theorem 2. [Completeness]** Let $\mathcal{M} = (R, \sim, V, \text{Exp})$ be a DDEL model, $\varepsilon$ the initial knowledge and $\varphi \in \Phi$ a formula. Then
\[ \mathcal{M}, v \models \varphi \iff ET(\mathcal{M}), (s, \varepsilon) \models \varphi \]

**Proof.** We follow the proof from [8, Prop. 14]. The boolean and epistemic cases are immediate from the temporal model construction. For $\varphi := [\alpha] \psi$ we assume that $\mathcal{M}, v \models [\alpha] \psi$, but $ET(\mathcal{M}), (v, \varepsilon) \not\models [\alpha] \psi$. Then, exists $m \in \text{Exp}(v)$ such that $ET(\mathcal{M}), (v, m) \not\models \psi$. From the construction of $ET(\mathcal{M})$, the definition of worlds is $H = \{ (s, m) | s \in R, m \in \text{Exp}(s) \}$, so $m \in \text{Exp}(v)$. But $m$ is a message, then exists the restricted model $M|_m$. From bisimilarity, we have that $ET(M|_m), (v, \varepsilon)$ is bisimilar with $ET(M), (v, m)$. Then $ET(M|_m), (v, \varepsilon) \not\models \psi$. From the induction hypothesis, we have $\mathcal{M}, v \models \neg \psi$, which contradicts $\mathcal{M}, v \models [\alpha] \psi$. \hfill $\square$

We have that the DDEL system is complete.

**IV. IMPLEMENTATION IN LEAN**

In this section we will present the implementation of our system in Lean [1] prover assistant based on [2], and then we will prove the corectness of BAN deduction rules in DDEL.

**A. Language**

To implement DDEL, we have the following inductive types:

1. For messages:

   **inductive** message (\(\sigma : \mathbb{N}\) : Type
   
   \begin{verbatim}
   l null : \sigma \rightarrow message
   l nonc : message \rightarrow message
   l keys : message \rightarrow message \rightarrow message \rightarrow message
   l enec : message \rightarrow message \rightarrow message
   l decr : message \rightarrow message \rightarrow message
   l tupl : message \rightarrow message \rightarrow message
   \end{verbatim}

2. For programs:

   **inductive** program (\(\sigma : \mathbb{N}\) : Type
   
   \begin{verbatim}
   l skip : program
   l secv : program \rightarrow program \rightarrow program
   l reun : program \rightarrow program \rightarrow program
   l send : message \rightarrow program
   l recv : message \rightarrow program
   \end{verbatim}

3. For formulas:

   **inductive** form (\(\sigma : \mathbb{N}\) : Type
   
   \begin{verbatim}
   l atom : fin \sigma \rightarrow form
   l botm : form
   l impl : form \rightarrow form \rightarrow form
   l know : message \rightarrow form \rightarrow form
   l pro : program \rightarrow form \rightarrow form
   l mesg : message \rightarrow form
   l and : form \rightarrow form \rightarrow form
   l or : form \rightarrow form \rightarrow form
   \end{verbatim}
We make the following notations:

- **notation** \( p \overset{\text{impl}}{\Rightarrow} q := \text{form} \).\text{impl} p q
- **notation** \( p \overset{\land}{\Rightarrow} q := \text{form} \).\text{and} p q
- **notation** \( p \overset{\lor}{\Rightarrow} q := \text{form} \).\text{or} p q
- **notation** \( \langle K \rangle \text{\ m \ i} := \text{form} \).\text{know} m p
- **notation** \( p \overset{\alpha}{\Rightarrow} q := \text{form} \).\text{prog} \alpha q
- **notation** \( \langle \{ \rangle \langle m \rangle \rangle \) := \text{message} . \text{tupl} m n
- **notation** \( \langle \langle m \rangle \langle \rangle \rangle \langle k \rangle := \text{message} . \text{enct} m k

B. Deductive system

In order to be able to check security properties using DEXP, we have two add two deduction hypotheses that help us specify symmetric key protocols:

\[
\begin{align*}
\forall m_k \land \forall k \text{key}(i,j) \Rightarrow \text{send}_i \otimes m_k \lor \text{send}_j \otimes m_k \\
\forall k \text{key}(i,j) \Rightarrow K \otimes k \land K \otimes k
\end{align*}
\]

Observation 1. The first deduction hypothesis of the system represents a rule of how the participating agents; its need is highlighted in the modeling of the BAN logic: if there is an encrypted message with the communication key \( k \), and the communication key \( k \) is a key known to the agents \( i \) and \( j \), then the message is transmitted by only one of them.

Observation 2. The second deduction hypothesis is a rule for modeling symmetric key protocols: if the \( k \) key is a communication key between \( i \) and \( j \), then each of them knows it.

We define the following context, a set \( \Gamma \) of statements:

```lean
def ctx (\sigma : \text{N}) : \text{Type} := \text{set} (\text{form} \sigma)
```

The deductive system is:

- **inductive** proof (\sigma : \text{N}) : ctx \sigma \rightarrow \text{form} \sigma \rightarrow Prop
  - \text{ax} \{ \Gamma \} \{ \sigma \} \{ \text{p} \} \{ \text{h} \} \Rightarrow \text{proof} \Gamma \text{p}
  - \text{kand} \{ \Gamma \} \{ \text{i} \} \{ \text{message} \sigma \} \{ \text{p} \text{q} \} \Rightarrow \text{proof} \Gamma (((K \text{i}, \text{p}) \land (K \text{i}, \text{q}))) \Rightarrow ((K \text{i}, \text{p} \land \text{q}))
  - \text{ktruth} \{ \Gamma \} \{ \text{i} \} \{ \text{message} \sigma \} \{ \varphi \} \Rightarrow \text{proof} \Gamma ((K \text{i}, \varphi) \Rightarrow \varphi)
  - \text{kdist} \{ \Gamma \} \{ \text{i} \} \{ \text{message} \sigma \} \{ \varphi \} \{ \text{form} \sigma \} \Rightarrow \text{proof} \Gamma ((K \text{i}, (\varphi \Rightarrow \psi)) \Rightarrow ((K \text{i}, \varphi) \Rightarrow (K \text{i}, \psi)))
  - \text{progrdist} \{ \Gamma \} \{ \alpha \} \{ \text{program} \sigma \} \{ \varphi \} \{ \varphi \} \Rightarrow \text{proof} \Gamma ((\alpha(\varphi \Rightarrow \psi) \Rightarrow ((\alpha) \psi)) \Rightarrow ((\alpha) \psi))
  - \text{pdtruth} \{ \Gamma \} \{ \alpha \} \{ \text{program} \sigma \} \{ \varphi \} \{ \text{form} \sigma \} \Rightarrow \text{proof} \Gamma ((\alpha(\varphi) \Rightarrow \varphi) \Rightarrow ((\alpha) \varphi))
  - \text{honestyright} \{ \Gamma \} \{ \text{m} \text{k} \text{i} \} \{ \text{message} \sigma \} \Rightarrow \text{proof} \Gamma (((\kappa (\text{k}.\text{keys} i j j)) \land (\kappa (\text{\{m\} k)))) \Rightarrow ((\text{send} j (\text{m} i)))
  - \text{knowreceive} \{ \Gamma \} \{ \text{m} \text{i} \} \{ \text{message} \sigma \} \Rightarrow \text{proof} \Gamma (((\text{recv} i (\text{m} i)) \Rightarrow (K \text{i}, (\text{m} i)))
  - \text{knowsend} \{ \Gamma \} \{ \text{m} \text{i} \} \{ \text{message} \sigma \} \Rightarrow \text{proof} \Gamma (((\text{send} i (\text{m} i)) \Rightarrow (K \text{i}, (\text{m} i)))
```

C. BAN Rules Verification

In order to be able to verify the correctness of the BAN rules, we translate them our logic. We use the following correspondence:

1) formula \( i \equiv m \) is translated as \( K \otimes i m \) and it means \( i \) knows \( m \) in current state;
2) formula \( i \otimes m \) means that \( i \) receives \( m \) and is translated as \( \text{recv} i, \otimes m \);
3) formula \( i \rightarrow m \) is translated as \( \text{send} i, \rightarrow m \);
4) formula \( i \Rightarrow m \) means that \( i \) has jurisdiction over \( m \), so the agent knows \( m \) and \( m \) is true: \( K \otimes i m \Rightarrow m \);
5) formula \( i \leftarrow j \) is translated as \( \text{recv} i, \leftarrow j \);
6) formula \( i \leftarrow j \) is translated as \( \otimes i j \).

Now, we can prove that the translations in DEXP of the most important BAN inference rules (according to [5]) are sound. In the sequel, using Lean, we give the proofs only for the Message Meaning rule and for the Jurisdiction rule, few other rules are analysed in the Appendix.

**Lemma 2. The Message Meaning rule for shared key is a correct rule in the DEXP system.**

\( i \equiv j \leftarrow \ i \otimes \{m\} \)

**Proof.** We will prove this using Lean.

```lean
lemma MMSK_is_correct (\sigma : \text{N}) \{ \text{m} \text{k} \text{i} \text{j} : \text{message} \sigma \} \{ \Gamma : \text{ctx} \sigma \} \Rightarrow ((\sigma \rightarrow \Gamma) \Rightarrow (K \text{i}, (\text{send} j (\text{m} i)))) :=
\begin{align*}
\lambda \text{h} .  \text{h} \text{gen} \otimes \text{m} \text{p} \text{honestyright}
\end{align*}
```

A much easier demonstration is for the jurisdiction rule, because it uses the \( K \) operator distributivity over implication:

**Lemma 3. Jurisdiction rule is a correct rule in DEXP system.**

\( i \equiv j \Rightarrow m \quad i \equiv j \equiv m \)

**Proof.** We will prove this using Lean.
In order to prove some security properties, we recall the Lean protocol and we will implement the specification in Lean.

A. Protocol description in Lean

In this section we will analyze the Needham-Schroeder protocol and we will implement the specification in Lean, in order to prove some security properties. We recall the exchange of messages in Needham-Schroeder protocol:

\[
\begin{align*}
A \rightarrow S & : A, B, N_a \\
S \rightarrow A & : \{N_a, B, K_{ab}, \{K_{ab}, A\} K_{as}\} K_{as} \\
A \rightarrow B & : \{K_{ab}, A\} K_{bs} \\
B \rightarrow A & : \{N_b\} K_{ab} \\
A \rightarrow B & : \{N_b - 1\} K_{bs}
\end{align*}
\]

V. Needham-Schroeder PROTOCOL IMPLEMENTATION IN LEAN

In this section we will analyze the Needham-Schroeder protocol and we will implement the specification in Lean, in order to prove some security properties. We will formalize the specification in Delp and then we will implement every Delp formula in Lean.

First step: initialization

The initial knowledge of agents are:

\[
\begin{align*}
K_A(\@N_A \land \@key_{KAS}(A, S)) \\
K_S(\@key_{KAS}(A, S) \land \@key_{KBS}(B, S) \land \@key_{KAB}(A, B)) \\
K_B(\@key_{KBS}(B, S))
\end{align*}
\]

In Lean we have:

axiom NSimt (σ : N) { Γ : ctx σ } { A B S Na \\
Kab Kas Kbs : message σ } \\
: σ−Γ ⊢ (K A, ((i Na) \land (i Kas.keys A S))) \\
\land (K S, ((i Kas.keys A S) \land (i Kbs.keys B S)) \land (i Kab.keys A B))) \\
\land (K B, (i Kbs.keys B S)).

First round: exchange of messages between A and S

In Delp we have:

\[
\text{[send}_A]\text{[recv}_S]\text{[\@N}_A
\]

with the corresponding Lean implementation:

axiom NS1AtoS (σ : N) { Γ : ctx σ } { A S Na : message σ } \\
: σ−Γ ⊢ [send] A [recv S] (i Na).

Second round: exchange of messages between S and A

\[
\text{[send}_S]\text{[recv}_A]((\@\{N}_A K_{AS} \land \@\{key}_{KAB}(A, B) K_{BS}) (72)
\]

B. Verifying security properties of Needham-Schroeder

In order to prove some security properties, we must prove the following lemma that we will use further.

Lemma 4. Let 𝜑 be a set of statements, i and j two agents and 𝜑 a formula. Then 𝜑 ⊢ [send]i[recv]j[𝜑 implies 𝜑 ⊢ 𝐾j. 𝜑]

Proof. We will prove this lemma using Lean.

We can prove that the agent A knows the communication key between A and B.

Theorem 3. In Needham-Schroeder protocol, the agent A knows the communication key between A and B.

Proof. We will prove this theorem using Lean.
In a similar way, we can prove that also $B$ knows the communication key between $A$ and $B$.

**Theorem 4.** In Needham-Schroeder protocols, the agent $B$ knows the communication key between $A$ and $B$.

**Proof.** We will prove this theorem using Lean.

```
theorem B_knows_Kab {σ : Ν} {Γ : c t x σ} {A B S Na Kab Kas Kbs : message σ} : σ ⊢ Γ ⊢ KB , ι (K a b. keys A B) :=
  kgen $ mp pdtruth
  $ mp honestyright
  $ andintro (mp ktruth $ B_knows_Kbs A B S Na Kab Kas Kbs)
  (mp ktruth $ secv_imp_knowledge $ NS3 AtoB A B S Kab Kbs).
```

We have now that $K_{ab}$ is a common secret between $A$ and $B$, but we cannot prove that we also have a mutual authentication. We know that $K_A @ key_K_{ab}(A, B) \land K_B @ key_K_{ab}(A, B)$, but we don’t know if $K_A K_B @ key_K_{ab}(A, B)$ and $K_B K_A @ key_K_{ab}(A, B)$.

**VI. CONCLUSION AND FURTHER WORK**

The system DEXP is closely related to the system POL (Public observation logic [5]), but it has a different semantics for $[\alpha]_{p}$: the updated models of POL are replaced by DEL models [9], while the set $Exp$ represents the "adversary knowledge" (defined as in the operational semantics from [7]) and not the "expected observations" (as in POL). Even if our system is simpler than the one from [5], we are able to translate BAN logic and to validate BAN inference rules.

Our work so far shows that DEXP is a good candidate for modelling and analysing security protocols. We are aiming to define a system that has a rigorous theoretical development: it is complete and all proofs are certified by Lean implementations.

At this stage we’ve already noticed that further refinements are needed: so far we used "knowledge" operators but, in order to increase our system expressiveness, we would like to model the epistemic "trust"; we also consider adding a temporal behaviour, in order to be able to model the property of freshness since, currently, we use a weaker variant, namely

the uniqueness on the system (nonce). Last but not least, we consider adding the probabilistic interpretation, following the initial idea from [5].

On the implementation side in Lean, we will add the proof for the completeness theorem and we will keep all the theoretical results automatically verified for any subsequent modification.

**REFERENCES**

[1] Avigad, Jeremy and de Moura, Leonardo and Kong, Soonho *Theorem Proving in Lean* https://leanprover.github.io/theorem_proving_in_lean/theorem_proving_in_lean.pdf, 2021

[2] Bentzen, Bruno. "A Henkin-style completeness proof for the modal logic S5." arXiv preprint arXiv:1910.01697 (2019).

[3] Blackburn, Patrick, Maarten De Rijke, and Yde Venema. Modal logic: graph. Durst. Vol. 53. Cambridge University Press, 2002.

[4] Burrows, Michael, Martin Abadi, and Roger Michael Needham. “A logic of authentication.” Proceedings of the Royal Society of London. A. Mathematical and Physical Sciences 426.1871 (1989): 233-271.

[5] Halpern, Joseph Y., Ron van der Meyden, and Riccardo Pucella. "An epistemic foundation for authentication logics." arXiv preprint arXiv:1707.08750 (2017).

[6] Harel, David, Dexter Kozen, and Jerzy Tiuryn. "Dynamic logic." Handbook of philosophical logic. Springer, Dordrecht, 2001. 99-217.

[7] Cremers, Cas, and Sjouke Mauw. "Operational semantics." Operational Semantics and Verification of Security Protocols. Springer, Berlin, Heidelberg, 2012. 13-35.

[8] V an Ditmarsch, Hans, et al. "Hidden protocols: Modifying our expectations in an evolving world." Artificial Intelligence 208 (2014): 18-40.

[9] Van Ditmarsch, Hans,Wiebe van Der Hoek, and Barteld Kooi. Dynamic epistemic logic. Vol. 337. Springer Science & Business Media, 2007.