Model checking security protocols:
a multi-agent system approach

Wojciech Penczek\textsuperscript{1}

\textit{a joint work with Alessio Lomuscio}\textsuperscript{2}

\textsuperscript{1}ICS PAS, Poland

\textsuperscript{2}Imperial College London, UK

WG2.2 Meeting, Torino, September 2007
1 Introduction
   - Related Work
   - Aim of our Work

2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems

3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$

4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK

5 Conclusions
Outline

1 Introduction
   - Related Work
   - Aim of our Work

2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems

3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$

4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK

5 Conclusions
Outline

1 Introduction
   - Related Work
   - Aim of our Work

2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems

3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$

4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK

5 Conclusions
Outline

1 Introduction
   - Related Work
   - Aim of our Work

2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems

3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$

4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK

5 Conclusions
Outline

1 Introduction
   - Related Work
   - Aim of our Work
2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems
3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$
4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK
5 Conclusions
Outline

1 Introduction
   - Related Work
   - Aim of our Work
2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems
3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$
4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK
5 Conclusions
Related Work

A. Armando et al.,
The AVISPA Tool for the automated validation of internet security protocols and applications, CAV 2005.

D.A. Basin, S. Modersheim, and L. Vigano

A. Armando and L. Compagna.

M. Burrows, M. Abadi, R. Needham William
A logic of authentication.
IEEE Computer Society Press, 1996
Related Work

A. Armando et al.,
The AVISPA Tool for the automated validation of internet security protocols and applications, CAV 2005.

D.A. Basin, S. Modersheim, and L. Vigano

A. Armando and L. Compagna.

M. Burrows, M. Abadi, R. Needham William
A. Armando et al.,
The AVISPA Tool for the automated validation of internet security protocols and applications, CAV 2005.

D.A. Basin, S. Modershein, and L. Vigano
OFMC: A symbolic model checker for security protocols.

A. Armando and L. Compagna.
SAT-based model-checking for security protocols analysis.

M. Burrows, M. Abadi, R. Needham William
A logic of authentication.
IEEE Computer Society Press, 1996
A. Armando et al.,
The AVISPA Tool for the automated validation of internet security protocols and applications, CAV 2005.

D.A. Basin, S. Modersheim, and L. Vigano

A. Armando and L. Compagna.

M. Burrows, M. Abadi, R. Needham William
Outline

1 Introduction
   - Related Work
   - Aim of our Work

2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems

3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$

4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK

5 Conclusions
Aim of our Work

To define a **semantics for temporal/epistemic logic** inspired on some of the ideas above.

To integrate the lazy intruder model with a temporal/epistemic logic and pair this with a highly-efficient bounded model checking algorithm.

To work on a fully-fledged specification language involving temporal/epistemic operators as opposed to simply checking reachability of states.
To define a **semantics for temporal/epistemic logic** inspired on some of the ideas above.

To **integrate the lazy intruder model** with a temporal/epistemic logic and pair this with a highly-efficient bounded model checking algorithm.

To work on a **fully-fledged specification language** involving temporal/epistemic operators as opposed to simply checking reachability of states.
Aim of our Work

To define a **semantics for temporal/epistemic logic** inspired on some of the ideas above.

To integrate the lazy intruder model with a temporal/epistemic logic and pair this with a highly-efficient bounded model checking algorithm.

To **work on a fully-fledged specification language involving temporal/epistemic operators** as opposed to simply checking reachability of states.
Aim of our Work

To define a **semantics for temporal/epistemic logic** inspired on some of the ideas above.

To **integrate the lazy intruder model with a temporal/epistemic logic** and pair this with a highly-efficient bounded model checking algorithm.

To **work on a fully-fledged specification language involving temporal/epistemic operators** as opposed to simply checking reachability of states.
Outline

1 Introduction
   - Related Work
   - Aim of our Work

2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems

3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for \( \mathcal{L} \)

4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK

5 Conclusions
Dolev Yao Model

Principles of Dolev Yao Model

- Intruder in full control of the communication channel.
- Intruder stops and copies all messages and forwards them as he sees they fit.
- Intruder composes messages with keys and nonces and routes them as he sees it.
- Perfect encryption: Intruder cannot decrypt/compose messages without having the correct key.
### Interpreted Systems versus Security Terminology

<table>
<thead>
<tr>
<th>Interpreted systems</th>
<th>Security terminology</th>
</tr>
</thead>
<tbody>
<tr>
<td>Agents</td>
<td>Participants, Intruders</td>
</tr>
<tr>
<td>Actions</td>
<td>Send/Receipt of Messages</td>
</tr>
<tr>
<td>Protocol</td>
<td>Protocol Moves</td>
</tr>
</tbody>
</table>
### Security-specialised symbols and their interpretation

<table>
<thead>
<tr>
<th>Symbol</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a$</td>
<td>constant agent $a$</td>
</tr>
<tr>
<td>$A$</td>
<td>variable agent $A$</td>
</tr>
<tr>
<td>$k_a$</td>
<td>constant key $k$ related to constant agent $a$</td>
</tr>
<tr>
<td>$k_A$</td>
<td>(variable) key related to variable agent $A$</td>
</tr>
<tr>
<td>$K_a$</td>
<td>variable key related to constant agent $a$</td>
</tr>
<tr>
<td>$K_A$</td>
<td>variable key related to variable agent $A$</td>
</tr>
<tr>
<td>$n_a$</td>
<td>constant nonce $n$ related to constant agent $a$</td>
</tr>
<tr>
<td>$n_A$</td>
<td>(variable) nonce related to variable agent $A$</td>
</tr>
<tr>
<td>$N_a$</td>
<td>variable nonce related to constant agent $a$</td>
</tr>
<tr>
<td>$N_A$</td>
<td>variable nonce related to variable agent $A$</td>
</tr>
</tbody>
</table>
Definition

Messages

A message $msg$ is defined by the following grammar:

$$msg ::= a \mid A \mid n \mid N \mid k \mid K \mid (msg)_k \mid (msg)_K \mid msg \cdot msg,$$

where $a$ ($A$) is a constant (variable) principal,

$n$ ($N$) is a constant (variable) nonce,

$k$ ($K$) is a constant (variable) key.

The symbol $\cdot$ represents concatenation between messages.
Definitions

Letters

A letter is a tuple \( \langle @s, @r \rangle, msg \rangle \) where

- \( @s \) is the sender’s address,
- \( @r \) is the receiver’s address, and
- \( msg \) is the content of the letter \( lt \).

Header

\( (@s, @r) \) is called the header of \( \langle (@s, @r), msg \rangle \).

Example

\( \langle (@A, @b), n_A \rangle \) represents a (variable) letter referring to a message from variable sender \( A \) to constant principal \( b \) in which the content is a variable nonce that depends on the sender.
Local States

Definition

A local state for an agent $i \in Ag$ is a 6-tuple

$$l_i=(Ag_i, N_i^o, N_i^f, K_i, id_i, lt_i),$$

where

- $Ag_i \subseteq Ag$ is a set of agents known to agent $i$,
- $N_i^o$ is an ordered set of nonces that have been seen by agent $i$. Each nonce in $N_i^o$ is present in some tuple in $lt_i$.
- $N_i^f$ is a set of fresh nonces available to agent $i$.
- $K_i$ is a set of keys known to agent $i$.
- $id_i$ is the number of sessions either completed or currently running in which agent $i$ has participated.
- $lt_i \subseteq (lt, id)^*$ is a sequence of pairs of letters and sessions identifiers for the protocols sessions the agent has actively participated in.
Definitions

Global States

A global state $g = (l_1, \ldots, l_n)$ is a n-tuple of local states for all agents under consideration.

Initial Global State

The initial global state $g^0 = (l_1, \ldots, l_n)$ is a tuple, where

- $l_i = (A_{g_i}, \emptyset, N_i^f, K_i, 0, \epsilon)$ for all $i = 1, \ldots n$
- $\bigcap_{i=1}^{n} N_i^f = \emptyset$ (i.e., the sets of fresh nonces are disjoint).
Outline

1 Introduction
   - Related Work
   - Aim of our Work
2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems
3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$
4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK
5 Conclusions
State Transformer Rules

Definition

For each step $t$ in a protocol under analysis, we consider state transformer rules of the form

$$(pre(t), post(t))$$

- $G$ - a set of the global states,
- $pre(t)$ - $t$-preconditions on $G$,
- $post(t)$ - $t$-postconditions on $G$.

Key restriction

For any rule to be triggered we need both the sender and the receiver to be in the correct protocol state for the transition to occur.
Lazy D-Y Interpreted System

Definition

\( P \) - a security protocol, \( PV \) - a set of propositional variables.
A model \( M_P = (G, g^0, \Pi, \sim_1, \ldots, \sim_n, V) \) is a \( n + 4 \)-tuple, where:

- \( G \) - the set of global states reachable from \( g^0 \),
- \( g^0 \) - an initial state for the system,
- \( \Pi = \bigcup_{g \in G} \Pi_{DY}(g) \), where \( \Pi_{DY}(g) \subset \Pi(g) \) is the set of all paths starting at \( g \) compliant with the Lazy D-Y conditions,
- \( \sim_i \subseteq G \times G \) - an epistemic relation for agent \( i \) defined by \( g \sim_i g' \) iff \( l_i(g) = l_i(g') \), where \( l_i : G \to L_i \) returns the local state of agent \( i \) given a global state.
- \( V : G \times PV \to \{ true, false \} \) - an interpretation for the propositional variables \( PV \) in the language.
Conditions on $M_P$

Key assumptions in generating the model include:

The model $M_P$ satisfies the following conditions:

- Agents have **perfect recall**: following receipt of a message agents add the message to their local state by pairing it with an appropriate session identifier.

- Every message sent by a principle is **intercepted** by the intruder, who records it in its local state.

- Upon receipt of messages all principals and the intruder immediately **decode** any messages and submessages providing they have the key to do so.

- No agent (nor the intruder) can decode messages without the correct key.
Generating the rules

Honest-send-i-A $\rightarrow B$

- **Preconditions:**
  If $i = 1$, then $A$ has not yet sent a message of step 1 to $B$. If $i \geq 2$, then $A$ has received a message from $B$ of step $i-1$ and has not yet replied to $B$.

- **Postconditions:**
  The local states of $A$ and $\iota$ ($B$ if $A = \iota$) are updated according to the message sent by $A$. If $B = \iota$, then $id_B := id_B + 1$. If $i = 1$, then $id_A := id_A + 1$. 
Generating the rules - cont’d

Fake-send-i-ι(A) → B

- **Preconditions:**
  A message of step $i$ is composable by $\iota$ and acceptable by $B$, i.e., $B$ has sent a message of step $i - 1$ (if $i \geq 2$) to $\iota$ and has not received a reply.

- **Postconditions:**
  The local states of $\iota$ and $B$ are updated according to the message sent by $\iota$. If $i = 1$, then $id_B := id_B + 1$. 
Generating the rules - cont’d

\(\iota\)-forward (step \(i\): \(A \rightarrow B\))

- **Preconditions:**
  A message of step \(i\) sent by \(A\) was intercepted by \(\iota\) and not yet received by \(B\).

- **Postconditions:**
  The local state of \(B\) is updated according to the message intercepted by \(\iota\).
<table>
<thead>
<tr>
<th>Section</th>
<th>Outline</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Introduction</td>
</tr>
<tr>
<td></td>
<td>Related Work</td>
</tr>
<tr>
<td></td>
<td>Aim of our Work</td>
</tr>
<tr>
<td>2</td>
<td>Semantics</td>
</tr>
<tr>
<td></td>
<td>Dolev-Yao Model</td>
</tr>
<tr>
<td></td>
<td>Lazy D-Y Interpreted Systems</td>
</tr>
<tr>
<td>3</td>
<td>Temporal Logic of Knowledge</td>
</tr>
<tr>
<td></td>
<td>Syntax</td>
</tr>
<tr>
<td></td>
<td>Semantics</td>
</tr>
<tr>
<td></td>
<td>BMC for $\mathcal{L}$</td>
</tr>
<tr>
<td>4</td>
<td>Example: NSPK</td>
</tr>
<tr>
<td></td>
<td>Rules for NSPK</td>
</tr>
<tr>
<td></td>
<td>Run of NSPK</td>
</tr>
<tr>
<td></td>
<td>Execution of NSPK</td>
</tr>
<tr>
<td>5</td>
<td>Conclusions</td>
</tr>
</tbody>
</table>
Temporal Logic of Knowledge

Language $\mathcal{L}$

$$\phi ::= sends_i(msg) | receives_i(msg) | has_i(k) | has_i(n) | \neg\phi \mid \phi \land \phi \mid \overline{K}_i\phi \mid EX\phi \mid E(\phi U \phi) \mid EG\phi,$$

where

- $sends_i(msg)$, $receives_i(msg)$, $has_i(k)$, $has_i(n) \in PV$,
- $msg$ is a constant message,
- $k \in \mathcal{K}_i$ is a key,
- $n \in \mathcal{N}_i^o$ is a nonce, and
- $PV$ is a set of propositional variables.
Semantics (1)

- \( g \models sends_i(msg) \) iff \(((@_i, @_j), msg), id)\) is an element of the sequence \(lt_i\) in agent \(i'\)s local state for some address \( @_j \) and session number \(id\),

- \( g \models receives_i(msg) \) iff \(((@_j, @_i), msg), id)\) is an element of the sequence \(lt_i\) in agent \(i'\)s local state \(l_i\) in \(g\), for some address \( @_j \) and session number \(id\),

- \( g \models has_i(n) \) iff \(n \in N_i^o\),

- \( g \models has_i(k) \) iff \(k \in K_i\),

- \( g \models \neg \phi \) iff not \( g \models \phi \)

- \( g \models \phi \land \psi \) iff \( g \models \phi \) and \( g \models \psi \),
Semantics (2)

- $g \models K_i \phi$ iff $(\exists g' \in G) \ g \sim_i g'$ and $g' \models \phi$,
- $g \models EX \phi$ iff $(\exists \pi \in \Pi_{DY}(g))$ s.t. $\pi(1) \models \phi$,
- $g \models EG \phi$ iff $(\exists \pi \in \Pi_{DY}(g))$ s.t. $(\forall k \geq 0) \ \pi(k) \models \phi$,
- $g \models E(\phi U \psi)$ iff $(\exists \pi \in \Pi_{DY}(g))$ $(\exists k \geq 0)$ s.t. $\pi(k) \models \psi$ and $(\forall 0 \leq j < k) \ \pi(j) \models \phi$. 
Outline

1. Introduction
   - Related Work
   - Aim of our Work

2. Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems

3. Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$

4. Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK

5. Conclusions
BMC for $L$

Translations

The propositional formula $[M\varphi,g^0]_k$, representing the $k$-paths in the $k$-model, is defined as follows:

$$[M\varphi,g^0]_k := I^0_g(w_{0,0}) \land \bigwedge_{j=1}^{f_k(\varphi) - 1} \bigwedge_{i=0}^{k-1} T(w_{i,j}, w_{i+1,j}),$$

where $w_{0,0}$ and $w_{i,j}$ for $0 \leq i \leq k$ and $1 \leq j \leq f_k(\varphi)$ are global state variables, and $T(w_{i,j}, w_{i+1,j})$ is a formula encoding the transition relation $T$ of $\Pi$.

$[\varphi]_{k^{[m,n]}}$ - the translation of $\varphi$ at $w_{m,n}$.

$M_k \models \varphi$ iff $[M\varphi,g^0]_k \land [\varphi]_{k^{[0,0]}}$ is satisfiable.
The NSPK protocol is defined by the following three steps:

1. \( A \rightarrow B: \{A, N_A\}_{K_B} \)

2. \( B \rightarrow A: \{N_A, N_B\}_{K_A} \)

3. \( A \rightarrow B: \{N_B\}_{K_B} \)

1. \( A \) (the Initiator) sends to \( B \) (the Responder) his identity \( A \) and a fresh nonce \( N_A \), both encrypted with \( B \)’s public key \( K_B \).

2. \( B \) responds to \( A \) with the nonce \( N_A \) and a fresh nonce \( N_B \), both encrypted with \( A \)’s public key \( K_A \).

3. \( A \) sends back to \( B \) the nonce \( N_B \) encrypted with the key \( K_B \).
Outline

1. Introduction
   - Related Work
   - Aim of our Work
2. Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems
3. Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$
4. Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK
5. Conclusions
Rule $T_1$: Honest Send 1. $(A \rightarrow B)$

**Preconditions:**
$(((\@_A, \@_B), (A, N_A)_{k_B}), Id'_A) \notin L_A,$

**Postconditions:**
If $A \neq \iota$, then
$L'_A = L_A \circ (((\@_A, \@_B), (A, N_A)_{k_B}), Id_A + 1) \circ \{N_A\} \circ \{Id_A + 1\},$
$L'_\iota = L_\iota \circ (((\@_A, \@_B), (A, N_A)_{k_B}), Id_A + 1)$ if $B \neq \iota$, and
$L'_\iota = L_\iota \circ (((\@_A, \@_B), (A, N_A)_{k_B}), Id_\iota + 1) \circ \{N_A\} \circ \{Id_\iota + 1\}$
if $B = \iota$, where $N_A = \text{first}(N^f_A)$.

If $A = \iota$, then
$L'_A = L_A \circ (((\@_A, \@_B), (A, N_A)_{k_B}), Id_A + 1) \circ \{N_A\} \circ \{Id_A + 1\},$
$L'_B = L_B \circ (((\@_A, \@_B), (A, N_A)_{k_B}), Id_B + 1) \circ \{N_A\} \circ \{Id_B + 1\},$
where $N_A \in \{\text{First}(N^f_\iota)\} \cup N^o_\iota$.
$L'_A = L_A \circ c$ denotes the update of the set of local states for variable agent $A$ by means of the component $c$. 
Rule $T_2$: Fake Send 1. $(\tau(A) \rightarrow B)$

Preconditions:

$(((\tau_l, \tau_B), (A, N_l)_{k_B}), Id) \notin L_B,$

Postconditions:

$L'_B = L_B \circ (((\tau_l, \tau_B), (A, N_A)_{k_B}), Id_B + 1) \circ \{N_A\} \circ \{Id_B + 1\},$

$L'_l = L_l \circ (((\tau_l, \tau_B), (A, N_l)_{k_B}), Id_l + 1) \circ \{N_l\} \circ \{Id_l + 1\},$

where $N_l \in \{\text{First}(N^f_l)\} \cup N^o_l$, $N_A = N_l$, and $A \neq \tau$. 
Consider 3 agents (2 participants $a$ and $b$ communicating in the presence of an intruder $\iota$).

A run begins at an initial global state $g^0 = (l_a^0, l_b^0, l_\iota^0)$, where $l_j^0 = (\{a, b, \iota\}, \emptyset, \mathcal{N}_j^f, \{k_a, k_b, k_\iota, k_{j-1}\}, 0, \epsilon)$, for $j \in \{a, b, \iota\}$.

$a$ is initiating an NSPK exchange with $\iota$ thinking he is an honest participant.
A run of NSPK (2)

1.1 honest - send - 1.a → τ

\[ l'_a = l_a \circ (((@_a, @_l), (a, n_a)_{k_l}, 1) \circ \{ n_a \} \circ \{ 1 \}, \]
\[ l'_l = l_l \circ (((@_a, @_l), (a, n_a)_{k_l}, 1) \circ \{ n_a \} \circ \{ 1 \}, \]
where \( n_a = First(N^f_a) \).

2.1 fake - send - 1.ι(a) → b.

\[ l'_b = l_b \circ (((@_b, @_l), (a, n_a)_{k_b}, 1) \circ \{ n_a \} \circ \{ 1 \}, \]
\[ l'_l = l_l \circ (((@_l, @_b), (a, n_a)_{k_l}, 2) \circ \{ 2 \}, \]

2.2 honest - send - 2.b → ι(a)

\[ l'_b = l_b \circ (((@_b, @_l), (n_a, n_b)_{k_a}, 1) \circ \{ n_b \}, \]
\[ l'_l = l_l \circ (((@_b, @_l), (n_a, n_b)_{k_l}, 2), \]
where \( n_b = First(N^f_b) \).
A run of NSPK (2)

1.2 honest – send – 2.ι → a,
\[ l'_a = l_a \circ ((@_l, @_a), (n_a, n_b)_{k_a}, 1) \circ \{n_b\}, \]
\[ l'_\iota = l_\iota \circ ((@_l, @_a), (n_a, n_b)_{k_a}, 2). \]

1.3 honest – send – 3.a → ι,
\[ l'_a = l_a \circ ((@_a, @_l), (n_b)_{k_b}, 1), \]
\[ l'_\iota = L_\iota \circ ((@_a, @_l), (n_b)_{k_b}, 1). \]

2.3 fake – send – 3.ι(a) → b,
\[ l'_b = l_b \circ ((@_l, @_b), (n_b)_{k_b}, 1), \]
\[ l'_\iota = l_\iota \circ ((@_l, @_b), (n_b)_{k_b}, 2), \]
Outline

1 Introduction
   - Related Work
   - Aim of our Work
2 Semantics
   - Dolev-Yao Model
   - Lazy D-Y Interpreted Systems
3 Temporal Logic of Knowledge
   - Syntax
   - Semantics
   - BMC for $\mathcal{L}$
4 Example: NSPK
   - Rules for NSPK
   - Run of NSPK
   - Execution of NSPK
5 Conclusions
The above two interleaved sessions define the following execution:

\[ g_0 \xrightarrow{1.1} g_1 \xrightarrow{2.1} g_2 \xrightarrow{2.2} g_3 \xrightarrow{1.2} g_4 \xrightarrow{1.3} g_5 \xrightarrow{2.3} g_6. \]

**Correctness Criterion**

If \( b \) completes an execution started by \( a \) using nonce \( n_b \), then \( b \) and \( a \) know that \( n_b \) is a secret shared by \( a \) and \( b \) only (it is unknown to \( \iota \)).

**Correctness Formula**

\[
\varphi = AG((\text{has}_a(n_b) \land \text{has}_b(n_a) \land \text{send}_a((n_b)_{k_b}) \land \\
\text{receive}_b((n_b)_{k_b}) \Rightarrow (K_b(\neg \text{has}_\iota(n_b)) \land K_a(\neg \text{has}_\iota(n_b))).
\]
Verification with BMC

The run satisfies the negation of the correctness formula:

\[ EF(has_a(n_b) \land has_b(n_a) \land sends_a((n_b)_k_b) \land receives_b((n_b)_k_b) \land
\quad (\overline{K}_b(has_t(n_b)) \lor \overline{K}_a(has_t(n_b)))]. \]
We have taken inspiration from the ideas of the lazy-intruder model to implement LDYISs, a MAS based semantics for security protocols.

We have defined a general approach to transition rules that generate LDYISs runs on which a temporal-epistemic logic can be interpreted.

The semantics of LDYISs is immediately ready to be model checked by means of any SAT-based methods such as bounded model checking.

We are about to finish an implementation.
We have taken inspiration from the ideas of the lazy-intruder model to implement LDYISs, a MAS based semantics for security protocols.

We have defined a general approach to transition rules that generate LDYISs runs on which a temporal-epistemic logic can be interpreted.

The semantics of LDYISs is immediately ready to be model checked by means of any SAT-based methods such as bounded model checking.

We are about to finish an implementation.
Conclusions

- We have taken inspiration from the ideas of the lazy-intruder model to implement LDYISs, a MAS based semantics for security protocols.

- We have defined a general approach to transition rules that generate LDYISs runs on which a temporal-epistemic logic can be interpreted.

- The semantics of LDYISs is immediately ready to be model checked by means of any SAT-based methods such as bounded model checking.

- We are about to finish an implementation.
We have taken inspiration from the ideas of the lazy-intruder model to implement LDYISs, a MAS based semantics for security protocols.

We have defined a general approach to transition rules that generate LDYISs runs on which a temporal-epistemic logic can be interpreted.

The semantics of LDYISs is immediately ready to be model checked by means of any SAT-based methods such as bounded model checking.

We are about to finish an implementation.