A THEORETICAL FRAMEWORK FOR MODELLING AND SIMULATING SECURITY PROTOCOLS

FRANTZ O. IWU and RICHARD N. ZOBEL

Department of Computer Science
University of Manchester
Oxford Road, Manchester, M13 9PL
United Kingdom
E-mail: {iwuo, rzobel}@cs.man.ac.uk

Abstract:
The aim of this paper is to present an approach to describe cryptographic protocols using agent-based simulation. This provides a framework to understand and model protocol behaviour and interaction in a simulation environment. Simulation techniques in the past have proven to be useful especially in areas where it is critical for testing to be carried out. This allows the designer to determine the correctness and efficiency of a design before the real system is constructed and deployed. Hence, an attempt to use this approach in testing the correctness of cryptographic protocols is promising.

Keywords:
Security, Protocols, Agent-Based Simulation

1. INTRODUCTION

Cryptographic protocols are designed to provide security services. Research has shown that a good number of these protocol are flawed. One reason for these failures, is primarily the lack of proper universally accepted technique and methodology for describing and analysing these protocols. Several successful attacks against cryptographic protocols, which exist in academic literatures show that weaknesses are not due to the underlying cryptographic algorithms but are as a result of logical errors. To deal with these problems, several methods have been proposed. These include methods based on specification languages and verification tools [Varadharajan, 1990], modal logic, expert systems, algebraic reasoning, and model-based approaches [Nieh, 1992; Gong, 1990; Burrows, 1990].

In this paper an approach to reasoning about the security of a protocol, which involves the use of agent models to characterises how principals interact is described. Furthermore, it describes how messages are sent and received, what messages a particular agent can assemble and transmit, the actions an agent can perform at a particular time and the use of simulation framework in modelling the activities of these agents. These characterisations form the bases for asking security related questions such as: what are the possibilities, given all possible situation and interactions, of security compromises. First, a conceptual model of the system needs to be designed, which describes how agents may communicate within a simulation environment and formalised using the Discrete Event Simulation (DEVS) formalism [Zeigler, 2000]. DEVS describes the autonomous and dynamic behaviour of agents and how agents react and generate input and output events at the atomic and coupling levels. Second, there is a need to design a simulation model, which enables agents to react and respond to events such as an intruder activity. Finally, the Needham-Schroeder and DSE protocols are considered using this approach.

2. AGENT-BASED SYSTEMS

Research and development of agent-based systems as a solution for various problem domains are rapidly increasing. Agents are in fact a key contributing technology for the Internet and World Wide Web and can be classified in several dimensions. The concept of deliberative agents was derived from the deliberative thinking paradigm in which agents hold an internal reasoning model from which it can make decisions to meet set goals. These kinds of agents are found in the area of artificial intelligence, psychology, cognitive sciences where agents have been modelled with personality traits and passion for decision-making [Baillie, 2002; Schmidt, 2002]. Conversely, reactionary agents do not have any internal symbolic model but make decisions based on stimulus or reaction from its environment. Agents can be classified according to their attributes such as autonomy, learning ability, interaction and cooperation. An important attribute of an agent is its ability to take initiatives and learn from past experience as it reacts and/or interacts within or outside its environment.

3. AGENT FRAMEWORK

A cryptographic protocol is considered to include a set of agents and channels of communication.

 ISBN 3-936150-25-7

These agents interact with each other according to some predefined rules and processing messages sent and received via the communication channels. A channel is an abstraction of the communication facility that has certain constraints. Each agent is an autonomous and reactionary entity capable of performing a sequence of operations (events) on messages. The agent formalism is characterised by the tuple modelled at the atomic level.

$$\Sigma_{Agent} = (X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta)$$

$X = \{x_1, x_2, ...x_n\}$ is a non empty set of input events.
$S = \{s_1, s_2, ...s_n\}$ is a non empty set of allowable states.
$Y = \{y_1, y_2, ...y_n\}$ is a non empty set of output events.
$\delta_{int} : S \rightarrow S$ An internal state transition function describing the behaviour of a Finite State Automaton.
$\delta_{ext} : Q * X \rightarrow S$: An external state transition function describing reaction of the agent to external events, where $Q = \{(s , e) \mid s \varepsilon S, 0 \leq e \leq ta (s)\}$.

$\lambda : S \rightarrow Y$: An output function which maps the internal agent state to the output set. Output events can only be generated at the time of internal transition.
$ta : S \rightarrow$ Time: This represents the time the agent stays in a particular state before transiting to the next sequential state.

Cryptographic protocols are designed to establish and authenticate communication between entities. Entities in cryptographic protocols are formally called principals and are assumed to have unique identities. A good number of cryptographic protocols require an authentication server or a certification authority to provide keys and certificates to enable secure communication between two or more principals. These properties and more are clearly identifiable in agents and to describe these properties in DEVS, the formalism for a coupled model needs to be introduced. The coupled model describes how to integrate all three agents as identified in the DSE protocol (discussed in subsequent sections) forming a larger model as shown below.

$$\prod_{AgentS} = (X, Y, M, \Upsilon_{eic}, \Upsilon_{eoc}, \alpha, select)$$

$X = \{x_1, x_2, ...x_n\}$ is a non empty set of inputs to the coupled model agent S.
$Y = \{y_1, y_2, ...y_n\}$ is a non empty set of outputs to the coupled model agent S.
$M = \{m_1, m_2, ...m_n\}$ is a non empty set of unique component references.
$\Upsilon_{eic} \subseteq \prod_{AgentS}.input * \Sigma_{Agent}.input$: An external input coupling relation.
$\Upsilon_{eoc} \subseteq \Sigma_{Agent}.output * \prod_{AgentS}.output$: An external output coupling relation.
$\alpha \subseteq \Sigma_{Agent}.output * \Sigma_{Agent}.input$: An internal coupling relation.
$select : 2^M \rightarrow M$: Tie breaking selector.

A multiple state transition is one of the problems associated with coupling concurrent and sequential components in discrete simulation systems. This may lead to instability if it occurs at the same simulation time. In order to deal with this problem a selection criteria is defined, which determines the component's transition that is priority. This is also applicable to agents described in the simulation model where select represents a tiebreak. Select chooses a unique agent from any non-empty subset E of M where E corresponds to the set of all agents having simultaneous state transitions.

4. THE SIMULATION MODEL

In order to explain the development of the model using the agent definition described above, a simple cryptographic protocol simulation model is presented. The model is intended to convey the session key K_{ab} and data, from agent A to agent B whilst keeping it secret from other agents on the network.

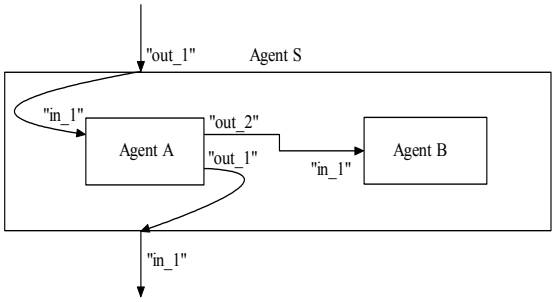

Figure 1: A Simple Cryptographic Protocol Model

Agent A makes contact with agent S, who provides A with the session key K_{ab} and a secret containing the session key K_{ab} but encrypted with B's key. Agent A then sends the secret to agent B who then decrypts the secret and stores the session key. Figure 1 shows the input and output ports of agent A. The input port "in_1" of agent A is for receiving messages from agent S. The output port "out_2" is used for sending the messages containing the session key K_{ab} to agent B and output port "out_1" of agent A is for making initial contact with agent S. A formal description of the model is specified using the atomic DEVS as shown below.

$$\Sigma_{AgentA} = (X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta)$$

$X = \{\text{“in_1”}\}$
$Y = \{\text{“out_1”}, \text{“out_2”}\}$
$S = \{idle, sent, recvd, accpt\}$.
δ_{int} (idle) = (make_INI_REQ, sent)
λ (idle) = “out_1” = msg

δ_{ext} ((-,-), "in_1") = (recv_MSG, recvd))
δ_{int} (cond ≠ False, accpt)) = (send_MSG, accpt)
λ (cond ≠ False, accpt)) = "out_2" = msg
ta(S) = time value

The notion of time cannot be effectively predicted due to factors such as network latency, bandwidth, encryption/decryption algorithms etc. The variable msg is an address location used to store received messages from agent S and cond is a conditional variable, which indicates success or failure of processed messages. Messages are transmitted either as plain text or cipher text depending on the protocol being simulated.

In figure 2, a state trajectory is given for the agent A model. It shows that the agent made an internal transition from idle to sent state. The agent remains in the idle state until the time ta (idle) elapses. When this occurs an output $y_1 = \lambda$(idle) is generated and the state is transited to the sent state. The agent remains in this autonomous mode until it receives an external event. This event does not give rise to an output.

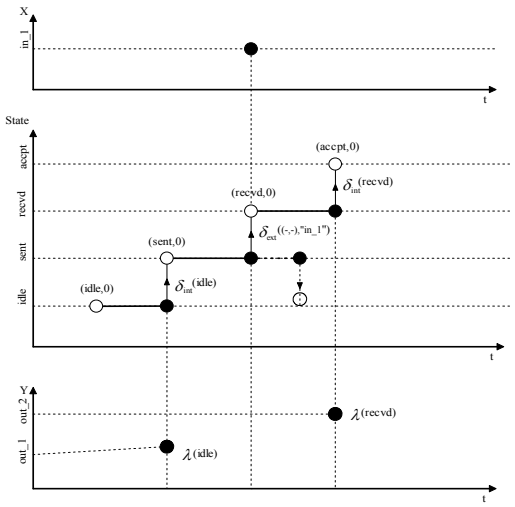

Figure 2: State Trajectory of Agent A

Once the agent A and agent B have been developed, the coupled agent S can be specified defining the required coupling relationship between all the agents in the model. The agent S model is formalised as follows.

$$\prod_{AgentS} = (X, Y, M, \Upsilon_{eic}, \Upsilon_{eoc}, \alpha, \text{select})$$

X = {"in_1","in_2"}
Y = {"out_1", "out_2"}
M = {Agent A, Agent B}
Υ_{eic} = {(Agent S.out_1, Agent A.in_1)}
Υ_{eoc} = {Agent A.out_1, Agent S.in_1)}
α = { AgentA.out_2, AgentB.in_1}
select : (Agent A, Agent B) → Agent A

5. NEEDHAM-SCHROEDER PROTOCOL

This protocol is the basis of many existing protocol designs today. It implements a symmetric mechanism and shares the common problem of key distribution. Here the client A makes the initial contact with the server S by sending a message consisting of its identity, the identity of client B and a randomly generated number N. The server S randomly generates a session key, which is shared between clients A and B and then encrypts a message containing the shared session key, the identity of client A with the session key it shares with client B.

Following from that the server encrypts another message containing the shared session key, the identity of client B, the random number generated by client A and an embedded encrypted message. The server eventually sends the encrypted message to client A, who with the knowledge of the server's shared key is able to decrypt the message. Client A subsequently sends the embedded encrypted message to client B, who is also able to decrypt it. Figure 3 illustrates the simulation model in the context of agent based framework. $N_b - 1$ in message 5 implies that the message is from A and not from B [Burrows et al, 1990].

message 1 A → S: A, B, N_a
message 2 S → A: $\{N_a, B, K_{ab}, \{K_{ab}, A\}_{Kbs}\} K_{as}$
message 3 A→ B: $\{K_{ab}, A\}_{Kbs}\}$
message 4 B → A: $\{N_b\}_{Kab}$
message 5 A → B: $\{N_b - 1\}_{Kab}$

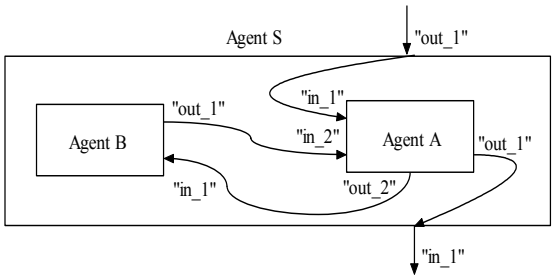

Figure 3: Needham-Schroeder Protocol Model

$$\prod_{AgentS} = (X, Y, M, \Upsilon_{eic}, \Upsilon_{eoc}, \alpha, \text{select})$$

X = {"in_1","in_2"}
Y = {"out_1", "out_2"}
M = {Agent A, Agent B}
Υ_{eic} = {(Agent S.out_1, Agent A.in_1)}
Υ_{eoc} = {Agent A.out_1, Agent S.in_1)}
α = { AgentA.out_2, AgentB.in_1}
α = { AgentB.out_1, AgentA.in_2}
select : (Agent A, Agent B) → Agent A

6. DSE PROTOCOL

The DSE protocol is based on shared and public key cryptography and is suitable for authenticating federates in HLA coupled distributed synthetic environment. Any number of federates can join or resign from the federation securely, hopefully, affecting the performance of the scheme minimally. The protocol is based on the plug and adaptor concept where plugs are attached to each federate model and the adaptor is attached to the RTI. This provides a platform for coordinated and secure communication between federates participating in the federation exercise. The adaptor S serves as an authentication server and a certificate authority providing various services to each federate model. Amongst other strengths, the protocol is designed to guard against a replay attack using synchronised clocks and randomly generated numbers.

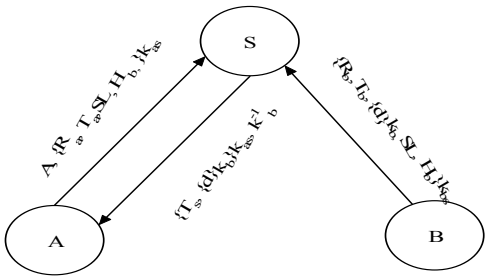

Figure 4: The DSE Authentication Protocol Structure

The description of the protocol is given below, with two federates attached to Plugs A and B as shown in figure 4. K_{as} and K_{bs} represent the shared keys for Plugs A and B, K_b^{-1} and K_b represent public and private keys for Plug B. R_a and R_b are random numbers generated by federates A and B. T_s, T_a and T_b are timestamps of Plugs A, B and adaptor S. SL represents the security level and finally d is the data in the form of updates. For the sake of clarity, the federate attached to Plug A will be referred to as federate A and the federate attached to Plug B as federate B. Federate B sends an encrypted message consisting of R_b, T_a and the object handle of the instance whose attributes need to be updated with the new attribute value d. If federate A subscribed to this attribute, once it has been registered and updated by B, it sends its identity along with an encrypted message consisting of a randomly generated number R_a, a timestamp T_a, and the object handle of the instance whose attributes have been updated to S. S confirms that the message is timely, and R_a checked against existing random generated numbers. S then generates a timestamp T_s, and forwards both the encrypted attribute values (data) and the certificate of B containing the public key of B to A who then extracts and verifies the public key. If the process is successful, the message is decrypted and the data is reflected and updated as shown in figure 4.

message 1 $B \rightarrow S$: $B,\{R_b, T_b, \{d\}k_b, SL, H_b\}k_{bs}$
message 2 $A \rightarrow S$: $A, \{R_a, T_a, SL, H_b\}k_{as}$
message 3 $S \rightarrow A$: $\{T_b, \{d\}k_b\}k_{as}, K_b^{-1}$

6.1 DSE Protocol Simulation Model

In this protocol agent B makes contact with agent S, if successful data is transferred. Agent A wishes to have access to the data transferred by agent B. To do so agent A must make contact with agent S passing on its identity and authentication details, which are verified. If successful agent S responds with the requested data encrypted with the shared key. The input and output ports of the agents are shown in figure 5 and the formal description given below.

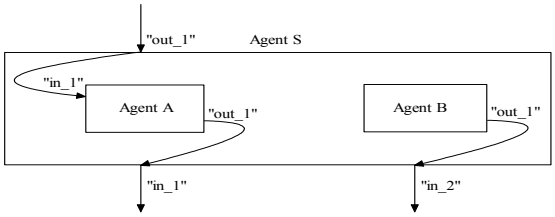

Figure 5: DSE Simulation Protocol Model

$\Sigma_{AgentA} = (X, S, Y, \delta_{int}, \delta_{ext}, \lambda, ta)$

X = {in_1"}
Y = {"out_1"}
S ={idle, sent, recvd, accpt}.
δ_{int} (idle) = (make_INI_REQ, sent)
λ (idle) = "out_1" = msg
δ_{ext} ((-,-), "in_1") = (recv_MSG, recvd))
ta(S) = time value

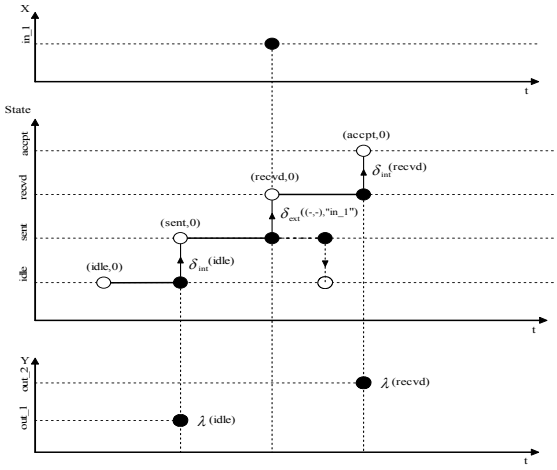

Figure 6: State Trajectory of Agent A in the DSE Model

The trajectory of agent A is shown in figure 6. Similarly, agent B can be developed and shown trivially. Once agent A and agent B have been

developed, the coupled model agent S can be specified. The agent S model formalisation is shown below.

$$\prod_{AgentS} = (X, Y, M, \Upsilon_{eic}, \Upsilon_{eoc}, \alpha, \text{select})$$

X = {“in_1”, “in_2”}
Y = {“out_1”}
M = {Agent A, Agent B}
Υ_{eic} = {(Agent S.out_1, Agent A.in_1)}
Υ_{eoc} = {Agent A.out_1, Agent S.in_1)}
α = { AgentB.out_1, AgentS.in_2}

6.2 Prototyping the Agent Model

The Simplex3 simulation system introduced by Schmidt [Schmidt, 2001] has been used to achieve the dynamics of the DEVS models. Each agent comprises various parts. The name, declaration and the dynamic part all make up the agent composition. Firstly, agents are identified by their names. The quantities and their properties are defined within the declaration part and the behaviour of the agent is described in the dynamic part. Agent behaviour could be modelled using differential equations, events or algebraic equations. However, in this case the behaviour of the agent is modelled using events.

6.3 High Level Component Agent_SIM

Three high level components have been defined, two for the atomic DEVS and the other for the coupled DEVS. Within the Simplex3 system, components can be composed of subcomponents and linked via connectors. Each agent is represented as an independent basic component and linked together in a high-level component Agent_SIM and described using the Simplex3 model description language. The Agent_SIM model describes a cycle in which agents can send and receive messages via input and output channels. On the start of the simulation, agent B creates and sends a message carrying data updates to agent S who then verifies and accepts the update.

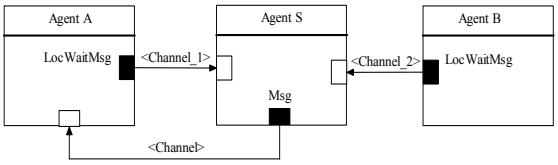

Figure 7: Relationship between Basic Agent Components

Agent A subscribes to data by sending a message comprising security information to agent S and if successful the data requested is sent to agent A. The dynamic behaviour of each agent describes the various events which can take place during the simulation run. Figure 7 shows the basic component agent A, agent B and agent S all linked together using component connections. Figure 8 shows the high level component connection representation of the agents. Messages are modelled using a mobile message component, as shown in figure 9, where a number of attributes are declared.

```
1: HIGH LEVEL COMPONENT Agent_SIM

2: SUBCOMPONENTS

3:    Agent_A,
4:    Agent_B,
5:    Agent_S

6: COMPONENT CONNECTIONS
7:    Agent_B.LocWaitMsg --> Agent_S.Channel_2;
8:    Agent_A.LocWaitMsg --> Agent_S.Channel_1;
9:    Agent_S.Msg --> Agent_A.Channel;

10: END OF Agent_SIM
```

Figure 8: High Level Component Agent_SIM

```
1: MOBILE COMPONENT Message

2: DECLARATION OF ELEMENTS

3:   STATE VARIABLES
4:   DISCRETE
5:      timestamp(REAL) :=0,  # timestamp
6:      randomnum(REAL):=0,   # random generated number
7:      data(REAL):=0,        # data updates
8:      handle (INTEGER):=0,  # data handle
9:      identity(INTEGER):=0  # federate identity

10: END OF Message
```

Figure 9: Mobile Component Message

7. EVALUATING THE PROTOCOL DESIGN

When examining the security of protocols, it is assumed that the underlying cryptographic mechanisms are secure. In evaluating the protocol an intruder does not necessary have to attack the underlying mechanism directly but rather attempt to subvert the protocol’s objective by defeating the manner in which such mechanisms are combined. Agents could be simulated with this capability and other known security breaches. This could provide for an effective security assessment of the protocol simulated.

The aim of the attack model is to provide some form of benchmark for testing the security of a protocol with the hope of uncovering any potential failures in the design. It is intended that the attack model is capable of performing a number of attack scenarios which include, first, the cases where the attack model is able to send messages but not read messages that are not addressed to it. Second, the attack model is able to send and read messages but not block messages and the last but not the least, the attack model could send, read and block messages but could not replace the blocked messages with other messages etc. Other capabilities that could be included in the attack model include the ability to break certain classes of cryptosystems. The overall attack model would provide the basis for simulating various attack scenarios and thus could reveal the requirements for a more secure protocol design.

7.1 Attack Model Description

In this section, an abstract simulator is considered. It describes some of the capabilities of the attack model for example impersonation and message interception where the attack agent attempts to play the role of the sender or the receiver as well as modify message content. Attack actions could be defined as events engaged by an attacker, which affect messages sent and received by legitimate participants in the protocol. Hence, an attack capability could be defined as a set of actions an attacker is able to perform. These actions and events are described in the model. The simulation starts when the model receives the SIMULATE message shown below, and stops when $tAttack_{agent} = \infty$

when receive (SIMULATE, t)
send (START, t) to $agent_{attack}$
while ($tAttack_{agent} \neq \infty$) do
"Intercept messages transmitted"
send (MSG, $tAttack_{agent}$) to $agent_{attack}$
"Message modification"
send (MSG, tN_{child}) to $agent_S$
endWhile

The attack simulator is necessary to drive the model. The variables t_L and t_N hold the time of last time, and the time for the next transition. This method sets the partial state to s{o} and the value e{o} is interpreted as the time elapsed in the current state.

when receive (START, t)
$t_L \leftarrow t \leftarrow e\{o\}$
$s \leftarrow t \leftarrow s\{o\}$
$t_N \leftarrow t_L \leftarrow ta\{s\}$
end
when receive (MSG, t)
if $t \neq t_N$ then return endif
intercepted $\leftarrow$ message $t \leftarrow$ MSG
$s \leftarrow \delta(s), s \leftarrow t \leftarrow s\{o\}$
$t_L \leftarrow t, t_N \leftarrow t_L \leftarrow ta\{s\}$
end
when receive (MSG, t)
if $t \neq t_N$ then return endif
message $t \leftarrow$ MSG
$s \leftarrow \delta(s), s \leftarrow t \leftarrow s\{o\}$
$t_L \leftarrow t, t_N \leftarrow t_L \leftarrow ta\{s\}$
end

8. SUMMARY

A possible approach, which utilises the merits of both agent-based and simulation technologies for analysing cryptographic protocols has been proposed. This approach is based on simulating an environment appropriate for describing the DSE protocol as well as other known protocols. The environment allows agents to interact amongst themselves and also react to external activities such as an intruder attack. In addition, an attack model was described and introduced, with a number of attack capabilities, in the simulation environment to provide a test bed for examining security flaws in the protocol simulation.

9. REFERENCES

Baillie P. 2002. "An Agent with a Passion for Decision Making." In Proc. of Agents in Simulation Workshop III Passau, Germany, University of Passau, April.

Burrows M, Abadi M, and Needham R. 1990. "A Logic for Authentication" SCR Research Report 39, Digital Equipment Corporation, February.

Gong L, Needham R and Yaholm R. 1990. "Reasoning About Belief in Cryptographic Protocols." In Proceeding of IEEE Symposium on Research in Security and Privacy, Pages 234-248, Oakland California.

Iwu, F.O. and Zobel R.N. 2002. "Network Attack Profiling: Using Agents-Based Simulation to Gather Forensic Information" In Proc. of Agents in SimulationWorkshop III Passau, Germany, University of Passau, Passau, April.

Nieh B and Tavares S. 1992. "Modelling and Analysing Cryptographic Protocols using Petri Nets." In Proceedings of AUSCRYPT.

Schmidt B. 2002. "How to give Agents a Personality." In Proc. Of Agents in Simulation Workshop III Passau, Germany, University of Passau, Passau.

Schmidt B. 2001. "The Art of Modelling and Simulation: Introduction to Simulation Systems Simplex3." Book, SCS-European Publishing House Erlanger.

Varadharajan V and Shankaran R. 1990. "The use of Formal Description Technique in the Specification of Authentication Protocols." In Proceedings of Computer Standards and Interfaces.

Zeigler B. and et al. 2000. "Theory of Modelling and Simulation." Book, Academic Press, Inc. January.

10. BIOGRAPHY

FRANTZ IWU is a research associate at the University of York. He obtained his MSc in Advanced Computer Science and has recently completed his PhD studies at Manchester University. He is a member of the British Computer Society.

RICHARD ZOBEL He is a former Chairman of the United Kingdom Simulation Society (UKSim), Former Secretary of the European Federation of Simulation Societies (EUROSIM), and is a European Director of SCSI, the Society for Computer Simulation International. His current research work concerns distributed simulation for non-military applications, issues of verification and validation of re-useable simulation models and security for distributed simulation under commercial network protocols. He is now semi retired, but remains very active.