EXPERIENCES OF USING THE PEPA PERFORMANCE MODELLING TOOLS WITH A NON-REPUDIATION PROTOCOL

Yishi Zhao
Nigel Thomas

School of Computing Science
Newcastle University, UK
Email: {Yishi.Zhao | Nigel.Thomas}@ncl.ac.uk

ABSTRACT

In this paper, we described a experience of using PEPA eclipse plug-in tool to specify a functional-equivalent representation of a non-repudiation protocol. The model is specified using Markovian process algebra PEPA. The basic model suffers from the well known state space explosion problem when tackled using *Continues Time Markov Chain* analysis. In order to modelling in a scalable way, *functional rates* has been adopted to avoid a unintended system behaviour. The *functional rates* have been specified in a *CMDL* (Chemical Model Definition Language) format which equivalently generated from the PEPA model by *PEPA eclipse plug-in*. This representation has been converted back to PEPA expression, and analyzed numerically.

KEYWORDS

PEPA, PEPA eclipse plug-in, Functional Rates, Non-repudiation

INTRODUCTION

Functional rates have been utilized to eliminate functional equivalent PEPA components to avoid state space explosion, [1]. This kind of specification is currently only supported by IPC (International PEPA Compiler).

In this paper, we demonstrate an alternative way of solving PEPA model with *functional rates* using the PEPA eclipse plug-in. This tool cannot directly analyze a PEPA model with functional rates, but an equivalent *CMDL* format can be generated that can be solved by fluid flow analysis (based on ODE) or stochastic simulation directly with the tool. We apply this method to a non-repudiation protocol, in which the service behaviours are not able to be defined in PEPA in a scalable way without *functional rates*.

The purpose of performance modelling security protocols is investigating the trade-off between security and performance. It is clear that in order to add more functionality to a system that more execution time is required. However, in the case of security, the benefit accrued from any additional overhead is not easy to quantify and so it is very hard for the performance engineer to argue that a particular performance target should take precedence over a security goal. There have been efforts made by both the security and performance communities to address aspects of this problem [2, 3, 4]. A *Key Distribution Centre* (key exchange protocol) has been studied in our previous work, which shows the possibility of modelling by the stochastic process algebra PEPA and analysis by several alternative techniques [5, 6, 7].

The remainder of this paper is organised as follows. In the next section we introduce the non-repudiation protocol to be modelled. This is followed by a brief review of the stochastic process algebra PEPA and PEPA eclipse plug-in. We then introduce the process of composing a PEPA model with *functional rates* and generating the equivalent CDML format model, followed by numerical results. Finally some conclusions are drawn and areas of further work described.

NON-REPUDIATION PROTOCOL

A non-repudiation service will prevent either of the principals involved from denying the contract after the agreement. The protocol depicted here were proposed by Zhou and Gollmann [8] and use a non-repudiation server, known as a *Trusted Third Party* (TTP).

- L: a unique label chosen by TTP to identify the message M
- T_s : the time that TTP received A's submission
- T_d : the time that TTP delivered and available to B
- $NRO = sS_A(f_{NRO}, TTP, B, M)$: non-repudiation of origin for M
- $NRS = sS_D(f_{NRS}, A, B, T_s, L, NRO)$: non-repudiation of submission of M
- $NRR = sS_B(f_{NRR}, TTP, A, L, NRO)$: non-repudiation of receiving a message labelled L
- $NRD = sS_D(f_{NRD}, A, B, T_d, L, NRR)$: non-repudiation of delivery of M

In this protocol, A sends the plaintext (M) and a non-repudiation origin (NRO) to the trusted third part (TTP), and then fetches the time of receiving (T_s) and non-repudiation of submission (NRS) from a public area, after TTP has published this information. The TTP tells B it received M from A by sending the NRO. B generates a non-repudiation of receiving for TTP following. Finally, B and A can fetch M and the time of delivery (T_d), with other non-repudiation evidence, from the public area, after the TTP has published.

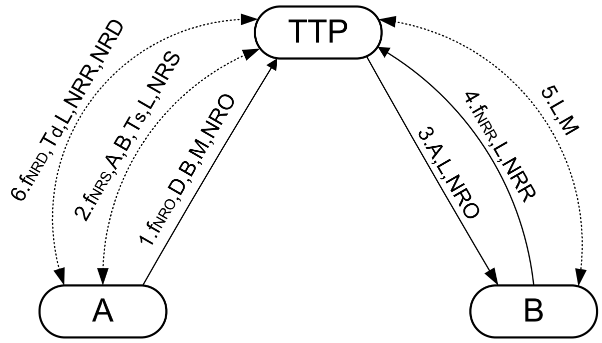

Figure 1: Non-repudiation protocol invented by Zhou&Gollmann

$$\begin{array}{rl}
(request) & 1.A \rightarrow TTP: \; f_{NRO}, TTP, B, M, NRO \\
(publish1\& & \\
getByA1) & 2.A \leftrightarrow TTP: \; f_{NRS}, A, B, T_s, L, NRS \\
(sendB) & 3.TTP \rightarrow B: \; A, L, NRO \\
(sendTTP) & 4.B \rightarrow TTP: \; f_{NRR}, L, NRR \\
(publish2\& & \\
getByB) & 5.B \leftrightarrow TTP: \; L, M \\
(publish2\& & \\
getByA2) & 6.A \leftrightarrow TTP: \; f_{NRD}, T_d, L, NRR, NRD
\end{array}$$

PEPA AND PEPA ECLIPSE PLUG-IN

A formal presentation of PEPA is given in [9], in this section a brief informal summary is presented. PEPA, being a Markovian Process Algebra, only supports actions that occur with rates that are negative exponentially distributed. Specifications written in PEPA represent Markov processes and can be mapped to a continuous time Markov chain (CTMC). Systems are specified in PEPA in terms of *activities* and *components*. An activity (α, r) is described by the type of the activity, α, and the rate of the associated negative exponential distribution, r. This rate may be any positive real number, or given as unspecified using the symbol $\top$.

The syntax for describing components is given as:

$$P ::= (\alpha, r).P | P + Q | P/L | P \underset{\mathcal{L}}{\bowtie} Q | A$$

The component $(\alpha, r).P$ performs the activity of type a at rate r and then behaves like P. The component $P + Q$ behaves either like P or like Q, the resultant behaviour being given by the first activity to complete.

The component P/L behaves exactly like P except that the activities in the set L are concealed, their type is not visible and instead appears as the unknown type τ.

Concurrent components can be synchronised, $P \underset{\mathcal{L}}{\bowtie} Q$, such that activities in the cooperation set L involve the participation of both components. In PEPA the shared activity occurs at the slowest of the rates of the participants and if a rate is unspecified in a component, the component is passive with respect to the activities of that type. $A \overset{def}{=} P$ gives the constant A the behaviour of the component P. The shorthand $P||Q$ is used to denote synchronisation over no actions, i.e. $P \underset{\emptyset}{\bowtie} Q$. We employ some further shorthand that has been commonly used in the study of large parallel systems. We denote $A[N]$ to mean that there are N instances of A in parallel, i.e. $A|| \ldots ||A$, but we are not concerned with the state of each individual component, rather the number of components in each state. As such, when using this representation, we would not distinguish between $A||A'$ and $A'||A$.

In this paper we consider only models which are cyclic, that is, every derivative of components P and Q are reachable in the model description $P \underset{\mathcal{L}}{\bowtie} Q$. Necessary conditions for a cyclic model may be defined on the component and model definitions without recourse to the entire state space of the model.

The *PEPA eclipse plug-in* is developed by researchers based in Edinburgh University [10, 11]. The tool can be used to conduct steady state and transient analysis of PEPA models through solving the CTMC, stochastic simulation and fluid flow analysis (based on ODEs). PEPA models with *functional rates* that are used in this paper are not directly supported by current version of *PEPA eclipse plug-in*, but an equivalent CMDL file (Chemical Model Definition Language) can be generated in which functional rates can be specified and analyzed.

MODEL CONSTRUCTION

Initial Model

We intuitively begin by define the behaviour of a pair of principals as:

$$\begin{aligned}
A0 &\overset{def}{=} (request, r_{t1}).A1 \\
A1 &\overset{def}{=} (publish1, r_{p1}).A2 \\
A2 &\overset{def}{=} (getByA1, r_{ga1}).A3 \\
A3 &\overset{def}{=} (sendB, r_b).A4 \\
A4 &\overset{def}{=} (publish2, r_{p2}).A5
\end{aligned}$$

$$
\begin{aligned}
A5 &\overset{def}{=} (getByA2, r_{ga2}).A6\\
A6 &\overset{def}{=} (work, r_w).A0\\[1em]
B0 &\overset{def}{=} (sendB, r_b).B1\\
B1 &\overset{def}{=} (sendTTP, r_{t2}).B2\\
B2 &\overset{def}{=} (publsih2, r_{p2}).B3\\
B3 &\overset{def}{=} (getByB, r_{gb}).B4\\
B4 &\overset{def}{=} (work, r_w).B0\\[1em]
TTP &\overset{def}{=} (publish1, r_{p1}).TTP\\
&\quad\ +(publish2, r_{p2}).TTP\\
&\quad\ +(sendB, r_b).TTP\\[1em]
System &\overset{def}{=} TTP[K] \underset{\mathcal{L}}{\bowtie} (A0||B0)[N]
\end{aligned}
$$

Where, $\mathcal{L} = \{publish1, publish2, sendB\}$.

In order simplify the model specification and analysis, we combine A and B into a new component called AB, using a process referred to as *partial evaluation* [12]. Following description can be obtained as a strong equivalent representation of above one with N pairs of principals.

$$
\begin{aligned}
AB0 &\overset{def}{=} (request, r_{t1}).AB1\\
AB1 &\overset{def}{=} (publish1, r_{p1}).AB2\\
AB2 &\overset{def}{=} (getByA1, r_{ga1}).AB3\\
AB3 &\overset{def}{=} (sendB, r_b).AB4\\
AB4 &\overset{def}{=} (sendTTP, r_t2).AB5\\
AB5 &\overset{def}{=} (publish2, r_{p2}).AB6\\
AB6 &\overset{def}{=} (getByA2, r_{ga2}).AB7\\
&\quad\ +(getByB, r_{gb}).AB8\\
AB7 &\overset{def}{=} (getByB, r_{gb}).AB9\\
AB8 &\overset{def}{=} (getByA2, r_{ga2}).AB9\\
AB9 &\overset{def}{=} (work, r_w).AB0;\\
TTP &\overset{def}{=} (publish1, r_{p1}).TTP\\
&\quad\ +(publish2, r_{p2}).TTP\\
&\quad\ +(sendB, r_b).TTP\\
System &\overset{def}{=} TTP[k] \underset{publsih1, publish2, sendB}{\bowtie} AB0[N]
\end{aligned}
$$

AB_0 to AB_9 in the above PEPA model denote the different behaviours of the AB component, and its evolution along the sequence of prescribed actions in the protocol. The choice from AB_6 to AB_7 and AB_8 means step 5 and step 6 in the protocol can happen in any order. The $work$ action is used to define that B can do something with the plaintext (M) that was sent by A after he has obtained it, before returning to the state AB_0 to make a new request again, which forms a working cycle to investigate the steady state. Here, we assume multiple TTP servers specified in the model are able to share a common memory.

In this TTP component representation, this naive model gives rise to a race between $publish1$, $publish2$ and $sendB$ in PEPA, which does not capture the intended behaviour of actual system. To avoid this unwanted race, a reasonable solution is adjusting the three rates associated with it, that should depend on the proportion each job that request service of $publish1$, $publish2$ and $sendB$ respectively. In other words, the rates are dependent on the current state of system using so-called *functional rates*. The procedure of specifying*functional rates* is illustrated in next subsection.

Functional Rates Specification

In this subsection we describe a *CDML (Chemical Model Definition Language)* model generated from above PEPA model by the PEPA eclipse plug-in. This specification contains rate functions and is able to be analyzed by the fluid flow approach (based on ODEs) or stochastic simulation, supported by the tool. The following *CMDL* model is generated by eclipse plug-in and modified with *functional rates*.

$$
\begin{array}{ll}
//Rates & //Population\ sizes\\
rb = 1.0; & AB0 = N;\\
rga1 = 1.0; & AB1 = 0;\\
rga2 = 1.0; & AB2 = 0;\\
rgb = 1.0; & AB3 = 0;\\
rp1 = 1.0; & AB4 = 0;\\
rp2 = 1.0; & AB5 = 0;\\
rt1 = 1.0; & AB6 = 0;\\
rt2 = 1.0; & AB7 = 0;\\
rw = 0.01; & AB8 = 0;\\
 & AB9 = 0;\\
 & TTP = K;
\end{array}
$$

$$
\begin{array}{l}
//Reactions\\
getByA1, AB2 \rightarrow AB3, rga1;\\
getByA21, AB6 \rightarrow AB7, rga2;\\
getByA22, AB8 \rightarrow AB9, rga2;\\
getByB1, AB6 \rightarrow AB8, rgb;\\
getByB2, AB7 \rightarrow AB9, rgb;\\
publish1, TTP + AB1 \rightarrow TTP + AB2, rx1;\\
publish2, TTP + AB5 \rightarrow TTP + AB6, rx2;\\
request, AB0 \rightarrow AB1, rt1;\\
sendB, TTP + AB3 \rightarrow TTP + AB4, rx3;\\
sendTTP, AB4 \rightarrow AB5, rt2;\\
work, AB9 \rightarrow AB0, rw;
\end{array}
$$

Where,

$r_{x1} = [rp1 * AB1 * ((AB1 + AB3 + AB5)^{-1}) * min(TTP, AB1 + AB3 + AB5)]$

$r_{x2} = [rp2 * AB5 * ((AB1 + AB3 + AB5)^{-1}) *$

$min(TTP, AB1 + AB3 + AB5)]$
$r_{x3} = [rb * AB3 * ((AB1 + AB3 + AB5)^{-1}) * min(TTP, AB1 + AB3 + AB5)]$

This *CMDL* format of the model is composed of *Rates*, *Population sizes* and *Reactions* parts. The *Rates* section is exactly the same as that specified in the PEPA model. The *Population sizes* contains the initial population of all derivatives and components. In our scenario, there are N client pairs, which haven't started any behaviours at the initial stage, represented by $AB_0 = N$, other derivatives have no population. K is the population of TTP all the time as no derivatives associated with it. The most important and main section of *CMDL* definition is $Reactions$, in which system behaviours defined as all actions name, individual state transitions and their rates.

In the *CMDL* model, we specify the functional rates for each cooperation under action $publish1$, $publish2$ and $sendB$ by r_{x1}, r_{x2} and r_{x3}, respectively, instead of r_{p1}, r_{p2} and r_b. Each of these functions describes the actual service rate if there is one job in the system(r_{p1}, r_{p2} or r_b), or as a proportion of the number of waiting jobs of each type ($ABi*((AB1+AB3+AB5)^{-1}), i = 1, 3, 5$) and the times of service ($min(TTP, AB1 + AB3 + AB5)$), which allocates each service with respect to its job type to eliminates the potential race. Although the PEPA model with function rates cannot be recognised by *PEPA eclipse plug-in*, it is still necessary to write it down as a formal specification of the protocol:

$$\begin{aligned}
AB_0 &\stackrel{def}{=} (request, r_{t1}).AB_1\\
AB_1 &\stackrel{def}{=} (publish1, r_{x1}).AB_2\\
AB_2 &\stackrel{def}{=} (getByA1, r_{ga1}).AB_3\\
AB_3 &\stackrel{def}{=} (sendB, r_{x3}).AB_4\\
AB_4 &\stackrel{def}{=} (sendTTP, r_{t2}).AB_5\\
AB_5 &\stackrel{def}{=} (publish2, r_{x2}).AB_6\\
AB_6 &\stackrel{def}{=} (getByB, r_{gb}).AB_7\\
&\quad +(getByA2, r_{ga2}).AB_8\\
AB_7 &\stackrel{def}{=} (getByA2, r_{ga2}).AB_9\\
AB_8 &\stackrel{def}{=} (getByB, r_{gb}).AB_9\\
AB_9 &\stackrel{def}{=} (work, r_w).AB_0\\
TTP &\stackrel{def}{=} (publish1, r_{x1}).TTP\\
&\quad +(publish2, r_{x2}).TTP\\
&\quad +(sendB, r_{x3}).TTP\\
System &\stackrel{def}{=} TTP[K] \underset{publish1, publish2, sendB}{\bowtie} AB_0[N]
\end{aligned}$$

Where,
$r_{x1} = r_{p1}\frac{AB_1(t)}{AB_1(t)+AB_3(t)+AB_5(t)}min(AB_1(t) + AB_3(t) + AB_5(t), TTP(t))$,
$r_{x2} = r_{p2}\frac{AB_5(t)}{AB_1(t)+AB_3(t)+AB_5(t)}min(AB_1(t) + AB_3(t) + AB_5(t), TTP(t))$.
$r_{x3} = r_b\frac{AB_3(t)}{AB_1(t)+AB_3(t)+AB_5(t)}min(AB_1(t) + AB_3(t) + AB_5(t), TTP(t))$.

NUMERICAL RESULTS

In our previous work [7], an assumption of the same action name and the same rates has been made for $publish1$, $publish2$ and $sendB$. With functional rates a more general scenario can be investigated to observe any differences between these three types of TTP service.

Figure 2 shows the average queue length varied with number of customer involved in this non-repudiation system solved by ODE solution supported by the tool. The ODE solution is an approximation which is very accurate in the extremes ($N = 1$ or large N) but much less accurate at the point at which the angle of the plot changes (here around $N = 14$ or $N = 16$). The point of maximum error can be precisely predicted, following our previous research [6]. If an accurate solution is required at these points then stochastic simulation could be employed within the PEPA eclipse plug-in.

Obviously the queue length increases as more client pairs join the system for both cases. In the case of $r_{p1} = 0.5$ and $r_{p2} = 0.2$, number of waiting jobs is always larger than the other case, as the average service rate is lower. In addition, the queue length of this set of parameters increases faster, this because the slower server is proportionally more heavily loaded as demand increases.

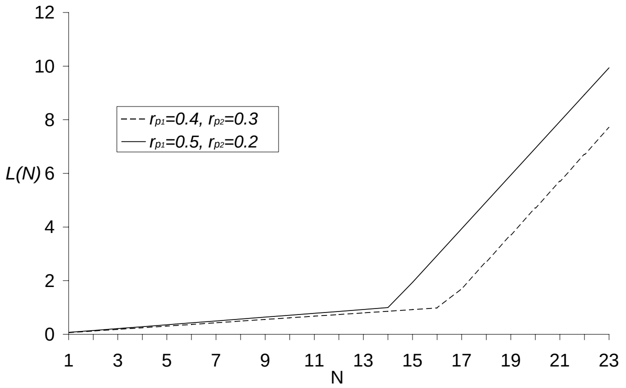

Figure 2: Average queue length varied with population size calculated by the ODE, $r_b = r_{t1} = r_{ga1} = r_b = r_{t2} = r_{gb} = r_{ga2} = 1, r_w = 0.01, K = 1$

Following the random observer principle of queueing theory (see [13] for example), the average response time can be calculated. If the random observer sees a free server, then the average response time will be the average service time, i.e. there is no queueing. However, if the random observer sees all the servers busy, then the average response time will be the average service time plus the time it takes for one server to become available (including scheduling the other jobs waiting ahead of the random observer).

$$
\begin{aligned}
W(N) &= \frac{1}{r_p}\ , \ L(N-1)+1 \leq K \\
W(N) &= \frac{1}{r_p} + \frac{L(N-1)+1-K}{Kr_p} \\
&= \frac{L(N-1)+1}{Kr_p}\ , \ L(N-1)+1 > K
\end{aligned}
$$

The above equations have been adopted to calculate average response time varied with system capacity by individual service behaviours in Figure 3. Here $W(1)$, $W(3)$ and $W(5)$ denotes the response times for the three responding actions by the TTP in the protocol, with the rates r_{p1},r_b and r_{p2} respectively. These are equivalent to the derivatives AB_1, AB_3 and AB_5 in the PEPA model. Clearly, the average response time for the first job type is slightly larger than third one (AB_5) and smaller than for the second job type (AB_3), because of the ratio between response time and responding rate. However, the average response time of all three job types grows at the same rate. The reason is obviously that the time for processing all the requests already within the queue is the same, only the time to process the arriving request differs. Thus, the difference between the response times of these three response actions is a constant.

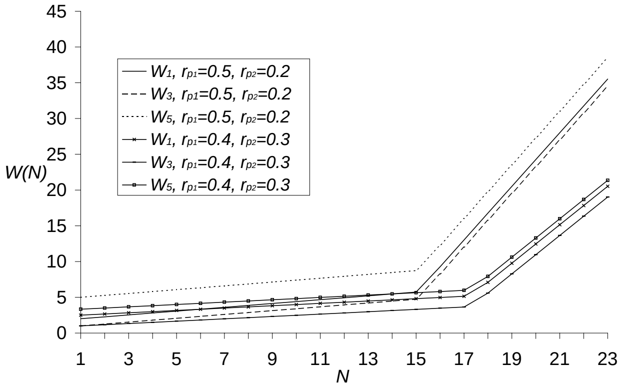

Figure 3: Average response time varied with population size calculated by the ODE, $r_b = r_{t1} = r_{ga1} = r_b = r_{t2} = r_{gb} = r_{ga2} = 1, r_w = 0.01, K = 1$

It is also interesting to note that the differences that occur as we alter the rate of the second and third response action. This difference between the two sets of curves is quite significant, far more so than we might naively expect. The initial stage ($N = 1 \sim 6$) of the average response time of the first type of jobs ($W(1)$) becomes larger as response rate decreases. Nevertheless, all three job types tend to respond quicker than the first set as N increases, because the average service time ($1/rp1 + 1/rb + 1/rp2$) is decreased, and the proportion of this type of request waiting at the TTP is smaller.
Multiple servers can be analyzed, as illustrated in Figure 4. Here, $L(1)$, $L(3)$ and $L(5)$ denote the queuing lengths for the three responding actions by the TTP in the protocol, corresponding to AB_1, AB_3 and AB_5 in the PEPA model. In each set of curves, a larger service rate results in a smaller number of waiting customers. Generally, there are fewer jobs waiting if more servers are being provided (obviously). Nevertheless, the number of the first type of waiting jobs (L1) with four TTP servers catches number of second type jobs with two TTP servers when $N = 145$, as they are the slowest and fastest one in each set respectively.

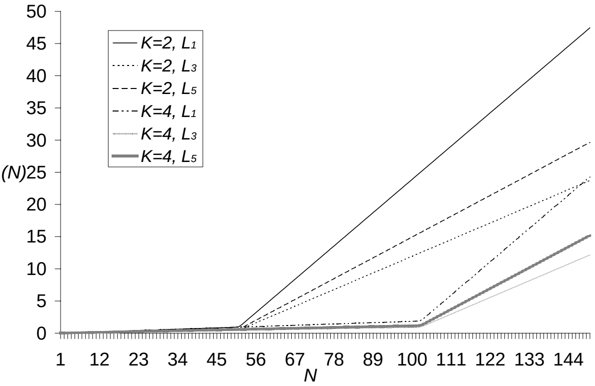

Figure 4: Average queue length varied with population size with different number of servers calculated by the ODE, $r_b = r_{t1} = r_{ga1} = r_b = r_{t2} = r_{gb} = r_{ga2} = 1, r_{p1} = 0.5, rp2 = 0.8, r_w = 0.01$

CONCLUSION AND FUTURE WORK

In this paper, we have showed how functional rates can deal with the unintended behaviours in the system, and introduced a novel approach to specify and analysis a PEPA model of a non-repudiation protocol with functional rates using PEPA eclipse plug-in. Although this tool cannot directly solve PEPA model with functional rates, a *CMDL* format model can be generated which can use functional rates and is supported by the tool with time series analysis. The PEPA eclipse plug-in includes ODE analysis and stochastic simulation, which are very scalable approaches, and applicable for a large class of models. Some numerical results have been presented which benefit from the functional rates specification.

There is still a limitation, in that the average response time calculation with different service rates at the TTP with multiple servers cannot be calculated exactly. In this case, in order to obtain the response time, the time it takes for one TTP server to become available should be calculated first. However, in FCFS queueing this requires us to know the queued order of the requests, which is clearly infeasible. We can only obtain the response time for three responding rates when there is a single TTP server. Thus, the waiting time for an arriving request is the time for a single TTP server to respond to all the requests in the queue, which does not require any knowledge about the order in which requests are queued. This remains issues of further investigation.

The work presented here forms part of an ongoing investigation into techniques for modelling and performance analysis of security protocols. The motivation for this work is the need to be able to investigate the trade-off that often exists between providing a secure environment and one that meets its temporal requirements. In the continuation of this investigation we will consider further protocols with more complex behaviour, e.g. multiple authentication parties and broadcast mechanisms.

REFERENCES

[1] J. Hillston and L. Kloul, A Functional Equivalent Component Based Simplification Technique for PEPA Models, in: *3rd European Performance Engineering Workshop*, pp.16-30, LNCS 4054, Springer-Verlag, 2006.

[2] S. Dick and N. Thomas, *Performance analysis of PGP*, in: F. Ball (ed.) *Proceedings of 22nd UK Performance Engineering Workshop*, Bournemouth University, 2006.

[3] W. Freeman and E. Miller, An Experimental Analysis of Cryptographic Overhead in Performance-critical Systems, *Proceedings of the 7th International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems*, IEEE Computer Society, 1999.

[4] C. Lamprecht, A. van Moorsel, P. Tomlinson and N. Thomas, Investigating the efficiency of cryptographic algorithms in online transactions, *International Journal of Simulation: Systems, Science & Technology*, 7(2), pp 63-75, 2006.

[5] Y. Zhao and N. Thomas, Approximate solution of a PEPA model of a key distribution centre, in: *Performance Evaluation - Metrics, Models and Benchmarks: SPEC International Performance Evaluation Workshop*, pp. 44-57, LNCS 5119, Springer-Verlag, 2008.

[6] N. Thomas and Y. Zhao, Fluid flow analysis of a model of a secure key distribution centre, in: *Proceedings of the 24th UK Performance Engineering Workshop*, Imperial College London, 2008.

[7] N. Thomas and Y. Zhao, Mean value analysis for a class of PEPA models, in: *6th European Performance Engineering Workshop*, pp.59-72, LNCS 5652, Springer-Verlag, 2009.

[8] J. Zhou and D. Gollmann, Observation on Non-repudiation, in: *Advances in Cryptology-ASIACRYPT'96*, pp.133-144, LNCS 1163/1996, Springer-Verlag, 1996.

[9] J. Hillston, A Compositional Approach to Performance Modelling, pp.141, Cambridge University Press, 1996.

[10] M. Tribastone, The PEPA Plug-in Project, in: *Proceedings of the 4th International Conference on the Quantitative Evaluation of SysTems (QEST)*, pages 53-54, IEEE, 2007.

[11] http://eclipse.org.

[12] A. Clark, A. Duguid, S. Gilmore and M. Tribastone, Partial Evaluation of PEPA Models for Fluid-Flow Analysis, in: *Computer Performance Evaluation: 5th European Performance Engineering Workshop*, LNCS 5261, Springer-Verlag, 2008.

[13] I. Mitrani, Probabilistic Modelling, Cambridge University Press, 1998.

BIOGRAPHY

Yishi Zhao is currently completing his Ph.D. in the School of Computing Science at Newcastle University. His area of investigation is the performance modelling and analysis of security protocols using stochastic process algebra. Yishi was awarded an M.Sc. in Computing Science with Distinction from Newcastle University in 2006.

Nigel Thomas is a Reader in the School of Computing Science at Newcastle University. He joined the Newcastle University in January 2004 from the University of Durham, where he had been a lecturer since 1998. His research interests lie in performance modelling, in particular Markov modelling through queueing theory and stochastic process algebra. Nigel was awarded a Ph.D. in 1997 and an M.Sc. in 1991, both from the Newcastle University.