Zhao Y, Thomas N.

Performance Modelling of Optimistic Fair Exchange.

***In: 23rd International Conference on Analytical & Stochastic Modelling Techniques & Applications (ASMTA)*. 2016, Cardiff: Springer.**

Performance modelling of optimistic fair exchange

Yishi Zhao[1] and Nigel Thomas[2]

[1] Faculty of Information Engineering, China University of Geosciences (Wuhan), China. yishi.zhao@gmail.com
[2] School of Computing Science, Newcastle University, UK. nigel.thomas@ncl.ac.uk

Abstract. In this paper we explore the overhead introduced by secure functions in considering a case study in non-repudiation. We present a model of an optimistic fair exchange protocol specified using the Markovian process algebra PEPA and present results derived using a fluid approximation and stochastic simulation. This system poses an interesting performance problem in that the degree overhead of the protocol is depended on the degree of misbehaviour by the participants.

1 Introduction

The security of modern computer and communication systems is a major concern for governments, organisations and individuals, resulting in a significant effort to ensure, and prove, that systems remain secure and data remains private. However, it is also essential that security measures do not impose excessive constraints on the user which then encourage subversion of those measures in order to make the system more usable. It should therefore be clear that any security measure that degrades usability is undesirable. However, all security measures will entail some additional work being undertaken which will impose a performance overhead. It is therefore essential that this overhead is understood, measured and minimised. In many practical situations there may be a choice of methods, such as varying protocols, algorithms or parameters, which could be employed. Changing the choice of method could have a potentially significant impact on the system performance without degrading the security. In other situations methods can be modified, e.g. by changing a key length, block size, padding or key refresh rate, which might improve performance at the cost of some level of security, thus giving a security performance trade-off [17]. However, quantifying this trade-off is not always possible, due to a lack of quantitative methods for evaluating system security.

In this paper we consider a type of non-repudiation protocol known as an *optimistic non-repudiation protocol*, which utilises a trusted third party when errors occur. This leads to an interesting performance problem where the capacity of the system at a given time is determined by the degree of misbehaviour. This paper is organised as follows. In the next section we explore performance modelling of security protocols and focus on the use of the sctochastic process

algebra PEPA [8]. The main focus of the paper is then covered in Section 3, with a case study in secure e-commerce. Finally we make some concluding remarks and observations.

2 Performance models of secure systems

A greater level of understanding of secure system performance can be gained by specifying and analysing a performance model. One objective of such analysis is to understand the trad-off that may be formed between competing requirements for greater security and acceptable performance under variable load. The notion of the performance security trade-off has been investigated by Wolter and Reinecke [17]. In order to formalise a trade-off there needs to be measures of both security and performance whose relative values can be contrasted. Wolter and Reinecke take the view that security can be considered as a form of reliability problem; thus the system may be considered to suffer security failures and may be subsequently recovered. The performance security trade-off is then characterised by the performance of such a system under different levels of threat.

Researchers have used a number of modelling approaches to analyse the performance of secure systems. Cho *et al* [5] used stochastic Petri nets (SPN) to investigate the potential attack in Dynamic Group Communication system (DGCs) to explore how the different rekeying methods affect both security and performance of the whole system, and optimised those methods with appropriate parameters. Wang *et al* [16] formulated a queueing model for three types of attack on email systems, analysing the model for performance, dependability and information leakage. Other notable queueing models include El-Hadidi *et al* [6, 7], who evaluated the performance of the Kerberos protocol in an distributed environment and Liu *et al* [10], who studied an authentication protocol. Recently a queueing model has been proposed by Meng *et al* [11] which explores the relationship between encryption key refresh rate and vulnerability of the communication channel to attack.

As the correctness of security protocols is often undertaken using process algebra [1, 13], it is a natural step to investigate temporal properties of protocols using stochastic process algebra (SPA). The advantage of using a formal specification for such models is that it is possible to check specific properties to ensure that the model correctly depicts behaviour which is essential to the security of the system. Thus a formal performance model and a formal security model of a given system can be shown to exhibit equivalence, giving the system designed some reassurance that the performance behaviour is valid. A process algebra allows detailed behaviour to be modelled and has the potential to be modified automatically through model transformations to facilitate alternative forms of analysis.

One of the earliest stochastic process algebra models of a secure system was that proposed by Buchholtz *et al* [4] concerning the so-called *wide mouthed frog protocol*. The purpose of this model was to investigate the potential vulnerability of the protocol to timing attack. Thomas [14] also used stochastic process

algebra, in a peformability study of a secure e-voting system. The analysis of the model did not scale well as the number of voters was increased, hence it was necessary to develop simplified models to support the analysis of larger scale systems. The issue of scale was a significant feature of three stochastic process algebra models of Internet worm attacks, proposed by Bradley *et al* [2]. To consider scalable analysis, a fluid flow approximation based on ordinary differential equations (ODEs) [9] was employed to analyse the models. This kind of analysis approximates the original discrete state space into continuous states, and it is able to cope with models of 10^{10000} states and beyond. ODE analysis was also employed in our previous work on a *Key Distribution Centre* (key exchange protocol). This work demonstrates a rigorous approach to specifying, modifying and analysing stochastic process algebra models of security protocols by several alternative techniques [15, 18, 20].

3 Optimistic Fair Exchange

The case study in this section concerns a type of non-repudiation protocol known as an *optimistic non-repudiation protocol*, which utilises a trusted third party when errors occur. This leads us to model the protocol in two ways: with misbehaviour and without. We employ a modelling form in which a server has been considered as several threads, with each thread associated with a customer. Hence, the service rate of the server becomes a function of the number of threads. In the next subsection a specification of the basic version (no misbehaviour) of the e-commerce protocol is given. The subsequent section then introduces the PEPA model of this basic version of the protocol, followed by numerical results. After that, an extended version (with misbehaviour) of the e-commerce protocol is described, with the PEPA model, then some numerical results.

3.1 An e-commerce protocol (basic)

This e-commerce protocol is an optimistic non-repudiation protocol, which adopts an offline TTP (*Third Tust Party*) not only to ensure fair exchange, but also to minimize the workloads from TTP server. Following the formal description in [12], the basic protocol (without misbehaviour of any principals) is illustrated below:

There is a set environment before the protocol operates, in which C (*Customer*) opens an account with B (*Bank*) and M (*Merchant*) registers with the TTP (*Trusted Third Party*). The protocol is then covered in six steps:

1. **C selects a product to purchase** (*download*). The customer chooses a product, and downloads it from the Internet merchant. However, this e-product has been encrypted, and so the customer cannot acquire the product without a decryption key. This product can be used for validation later.
2. **C and M agree upon a price for the product** (*agreePrice*). Several messages may be exchanged between C and M in this step.

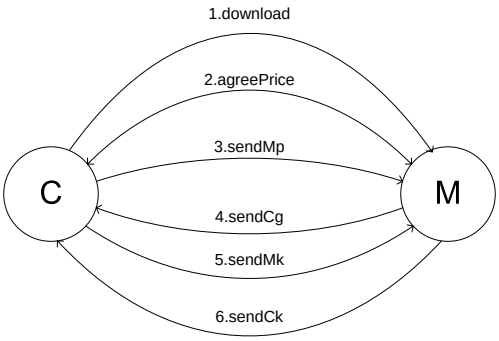

Fig. 1. The basic protocol

3. **C sends PO** (*purchase order*) to M (*sendMp*). The customer sends three elements to the merchant:*(a)* the purchase order; *(b)* a digitally signed cryptographic checksum of the PO; and *(c)* the PT (*Payment Token*).
4. **M sends encrypted product to C or abort the transaction** (*sendCg or sendCabort*). the merchant checks the purchase order which was received at the last step: if the merchant is not satisfied, then an abort message is sent to C; otherwise, the following is sent to C: *(a)* a signed cryptographic checksum of the purchase order; *(b)* encrypted product; *(c)* signed cryptographic checksum of the encrypted product; *(d)* encrypted random number; and *(e)* signed cryptographic checksum of the encrypted random number.
5. **C sends payment token decryption key to M or abort the transaction** (*sendMk or sendMabort*). C checks the message from M, if it is an abort, then abort the transaction. Otherwise, C attempts to validate the product. C sends M a signed abort message if the product has failed to be validated; otherwise, sends the payment token decryption key and a signed cryptographic checksum of the encrypted product decryption key.
6. **M sends product decryption key to C or terminates the transaction** (*sendCk*). If M receives an abort message from C, it terminates the transaction. Otherwise, if the received PT decryption key works, M sends the following to C:*(a)* the product decryption key; *(b)* signed cryptographic checksum of the encryption product decryption key; *(c)* the multiplicative inverse of a random number; *(d)* signed cryptographic checksum of the encrypted multiplicative inverse of the random number.

To address the performance aspects of this protocol, this illustration focuses on the behaviour. Therefore the security details (which would be crucial to a security evaluation) have been eliminated from the description above. The original paper [12] gives a more detailed version.

3.2 Misbehaviour

The protocol specification above is a basic version, which operates without misbehaviour of any participants. It is necessary to investigate the performance of the TTP to observe how the protocol reacts to potential misbehaviour by participants. Following [12], several misbehaviours have been introduced as follows:

M behaves improperly:

- M receives the payment token decryption key in step 5, but does not send the correct product decryption key in step 6.
  1. C sends a record of the exchange to the TTP (*sendTPall*).
  2. TTP asks M to send the correct decryption key and start a timer (*notifyM1*).
  3. M send the correct key to the TTP or has no response (*sendTPk1* or *tiemout1*).
  4. if M sends the correct key, the TTP forwards the key to C; if not, the TTP sends a decryption key (which was registered before this exchange) to C and takes appropriate action against M (*sendCkbyTP1* or *sendCkbyTP2, takeactionM*).
- M receives the payment token decryption key in step 5, but disappears without sending the product decryption key.
  1. C's timer expires (*noresponsedelay*).
  2. C sends a record of the exchange to the TTP (*sendTPall*).
  3. TTP asks M to send the correct decryption key and starts a timer (*notifyM2*).
  4. M has no response (*timeout2*).
  5. TTP sends a decryption key (which was generated before this exchange) to C and takes appropriate action against M (*sendCkbyTP2, takeactionM*).
- M claims that it did not send the correct decryption key because it has not received payment.
  1. M sends the reason that he did not receive proper payment (*sendTPreason*).
  2. M still needs to send product decryption key to the TTP (*sendTPk2*).
  3. Once the TTP receives the product decryption key from M, he sends appropriate decryption key to M and C (*sendMkbyTP1, sendCkbyTP3*).

C behaves improperly: M received the payment decryption key from the TTP again after he claims the wrong key in first instance. However, he still can not decrypt the payment by the key again:

1. Notify TTP of the failure of using the payment decryption key again (*sendTPnoti*).
2. TTP gets in touch with Bank to obtain a new key (*getkfromB*).
3. Sends the new key to M (*sendMkbyTP2*).

Once again, the description above is mainly about behaviour, in order address performance and more detailed security content has been described in [12]. The terms in the brackets after each item with bold font are the action name we have used in the PEPA model below. Moreover, we would like to propose three performance questions for this extended protocol as well as our previous case studies: "how many clients can a given TTP configuration support?", "how much service capacity must we provide at a TTP to satisfy a given number of clients?" and "what is the maximum rate at which keys can be refreshed before the TTP performance begins to degrade?" These questions are answered through numerical results following the model specification.

3.3 PEPA model

A PEPA model of the protocol incorporating misbehaviour can be specified as follows:

$$
\begin{aligned}
CT_0 &\stackrel{def}{=} (download, r_d).CT_1 \\
CT_1 &\stackrel{def}{=} (agreePrice, r_a).CT_2 \\
CT_2 &\stackrel{def}{=} (sendMp, r_{smp}).CT_3 \\
CT_3 &\stackrel{def}{=} (sendCg, f_1).CT_4 + (sendCabort, f_2).CT_7 \\
CT_4 &\stackrel{def}{=} (sendMk, r_{smk}).CT_5 + (sendMabort, r_{sma}).CT_8 \\
CT_5 &\stackrel{def}{=} (sendCk, f_3).CT_6 + (noresponsedelay, r_n).CT_{14} \\
CT_6 &\stackrel{def}{=} (work, r_w).CT_0 + (sendTPall, r_{stp}).CT_9 \\
CT_7 &\stackrel{def}{=} (sendMabort, r_{sma}).CT_8 \\
CT_8 &\stackrel{def}{=} (work, r_w).CT_0 \\
CT_9 &\stackrel{def}{=} (notifyM1, r_1).CT_{10} \\
CT_{10} &\stackrel{def}{=} (sendTPk1, f_7).CT_{11} + (timeout1, r_{10}).CT_{12} \\
CT_{11} &\stackrel{def}{=} (sendCkbyTP1, r_3).CT_8 \\
CT_{12} &\stackrel{def}{=} (sendCkbyTP2, r_4).CT_{13} \\
CT_{13} &\stackrel{def}{=} (takeactionM, r_6).CT_8 \\
CT_{14} &\stackrel{def}{=} (sendTPall, r_{stp}).CT_{15} \\
CT_{15} &\stackrel{def}{=} (notifyM2, r_2).CT_{16} \\
CT_{16} &\stackrel{def}{=} (sendTPreason, f_4).CT_{17} \\
&\quad + (timeout2, r_{11}).CT_{12} \\
CT_{17} &\stackrel{def}{=} (sendTPk2, f_5).CT_{18} \\
CT_{18} &\stackrel{def}{=} (sendMkbyTP1, r_7).CT_{19} \\
CT_{19} &\stackrel{def}{=} (sendCkbyTP3, p * r_5).CT_{20} \\
&\quad + (sendCkbyTP3, (1-p) * r_5).CT_8
\end{aligned}
$$

$$
\begin{aligned}
CT_{20} &\stackrel{def}{=} (sendTPnoti, f_6).CT_{21} \\
CT_{21} &\stackrel{def}{=} (getkfromB, r_9).CT_{22} \\
CT_{22} &\stackrel{def}{=} (sendMkbyTP2, r_8).CT_8 \\
TP &\stackrel{def}{=} (notifyM1, r_1).TP + (notifyM2, r_2).TP \\
&\quad +(sendCkbyTP1, r_3).TP \\
&\quad +(sendCkbyTP2, r_4).TP \\
&\quad +(sendCkbyTP3, r_5).TP \\
&\quad +(takeactionM, r_6).TP \\
&\quad +(sendMkbyTP1, r_7).TP \\
&\quad +(sendMbyTP2, r_8).TP \\
&\quad +(getKfromB, r_9).TP \\
&\quad +(timeout1, r_{10}).TP + (timeout2, r_{11}).TP \\
System &\stackrel{def}{=} TP[K] \underset{\mathcal{L}}{\bowtie} CT_0[N]
\end{aligned}
$$

Where,

$$
\begin{aligned}
\mathcal{L} = \{&notifyM1, notifyM2, sendCkbyTP1, sendCkbyTP2, sendCkbyTP3, \\
&takeactionM, timeout1, sendMkbyTP1, sendMkTP2, getKfromB, \\
&timeout2\}
\end{aligned}
$$

$$
\begin{aligned}
r_1 &= r_{nm1} \frac{CT_9}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right) \\
r_2 &= r_{nm2} \frac{CT_{15}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right) \\
r_3 &= r_{scktp1} \frac{CT_{11}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right) \\
r_4 &= r_{scktp2} \frac{CT_{12}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right) \\
r_5 &= r_{scktp3} \frac{CT_{19}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right) \\
r_6 &= r_{ta} \frac{CT_{13}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right) \\
r_7 &= r_{smktp1} \frac{CT_{18}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right) \\
r_8 &= r_{smktp2} \frac{CT_{22}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right) \\
r_9 &= r_{kb} \frac{CT_{21}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right) \\
r_{10} &= r_{t1} \frac{CT_{10}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right)
\end{aligned}
$$

$$r_{11} = r_{t2}\frac{CT_{16}}{\sum waitingJobs_{TP}} min\left(\sum waitingJobs_{TP}, TP\right)$$

$\sum waitingJobs_{TP} = \sum_{\forall i} CT_i(t),\ i \in \{9, 15, 11, 12, 19, 13, 18, 22, 21, 10, 16\}$.
if $N = 1$:

$f_1 = r_{scg}$, $f_2 = r_{sca}$, $f_3 = r_{sck}$, $f_4 = r_{stpr}$, $f_5 = r_{stpk}$, $f_6 = r_{stpno}$, $f_7 = r_{stpk}$,
if $N \neq 1$:

$f_1 = \frac{r_{scg}}{CT_3+1}$, $f_2 = \frac{r_{sca}}{CT_3+1}$, $f_3 = \frac{r_{sck}}{CT_5+1}$, $f_4 = \frac{r_{stpr}}{CT_{16}+1}$, $f_5 = \frac{r_{stpk}}{CT_{17}+1}$, $f_6 = \frac{r_{stpno}}{CT_{20}+1}$, $f_7 = \frac{r_{stpk}}{CT_{10}+1}$.

Following [19], a form of functional rates has been applied to avoid over estimating the value of rates of cooperation actions, which are denoted by $r_i, i = 1, 2, \cdots, 11$. Each of these functions describes the actual service rate if there is one job in the system(r_{nm1}, r_{nm2}, r_{scktp1}, r_{scktp2}, r_{scktp3}, r_{ta}, r_{smktp1}, r_{smktp2}, r_{kb}, r_{t1} and r_{t2}), or as a proportion of the number of waiting jobs (at TTP) of each type ($CT_i/\sum waitingJobs_{TP}, i = 9, 15, 11, 12, 19, 13, 18, 22, 21, 10, 16$) and the times of service ($min(TP, \sum waitingJobs_{TP})$, which allocates each service with respect to its job type to eliminates the potential race.

3.4 Numerical results

Figure 2 compares the average number of waiting customers at TTP against initial population of customers calculated by ODEs [9] and stochastic simulation [3]. The queue length increases when more clients are involved in the system. However, it is not difficult to spot that the two curves seems to keep a constant error when N is larger than 120. This phenomenon does not follow the ODE's normal excellent accuracy when N is very large. To investigate more deeply, we find that the population of behaviours after CT_{12} is actually very small, due to the race that between action *sendTPk* and *timeout1* in component CT_{10}, and also between action *sendTPreason* and *timeout2*. In the case where $N = 240$, it is a simple matter to calculate the functional rate of *sendTPk* $f_7(N = 240) \approx 0.85397$, and the functional rate of *timeout1* $r_{10}(N = 240) \approx 0.0020397$. The large difference also exists between *sendTPreason* and *timeout2*. About 400 times difference causes just a few components evolving to CT_{12} and its further (evolving) behaviours. Thus, $N = 240$ still cannot be considered as a large scale system with the current set of rates. That explains why the two methods do not converge when $N = 240$. Nevertheless, the two curves will converge eventually in some (extremely large) value of N. To take a further experiment, we set r_{t1} and r_{t2}, the original rates of *timeout1* and *timeout2*, to 200, and keep all other rates unchanged. This is in order to switch more clients to the behaviours after CT_{12}. Still in case of $N = 240$, $L(ODE) \approx 99.9595$ and $L(SS) \approx 100.0637$, illustrating the argument above.

The average number of waiting customers at the merchant is presented in Figure 3. Generally, the more customers involved, the more customers that will be waiting at the merchant. However, the results calculated by ODEs and stochastic simulation do not converge. This is caused by the same reason as discussed

above. When the TTP is working for misbehaviour cases, it is become very busy (if there is only one TTP server, as in our model) and most of the customers are waiting for the TTP. Therefore, the scale of the queue length at the merchant remains very small. This is why results of ODE and stochastic simulation did not converge here.

The total average number of waiting customers with and without misbehaviour have been compared in Figure 4. Under the same rates for each relevant actions and the same involved number of customers, far more customers are waiting in a situation of misbehaviour, especially, when N is very large. This is an intuitive and expected result, because customers who encounter misbehaviour have recourse to the TTP for help, and then wait at the TTP for a resolution. Hence, it is clear that misbehaviours reduce the performance of the whole sys-

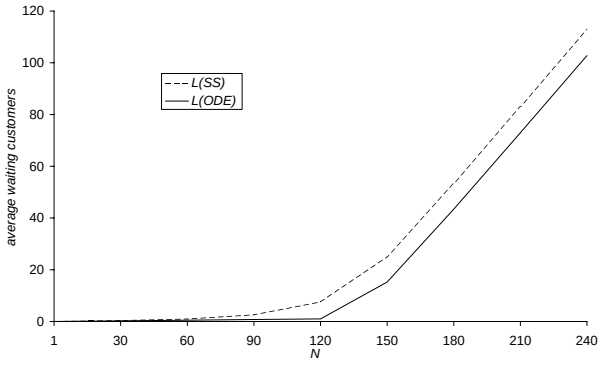

Fig. 2. Average number of waiting customers at TTP varied with population size calculated by ODEs and stochastic simulation, $p = 0.5$, $r_w = 0.01$ and all other rates are 1.

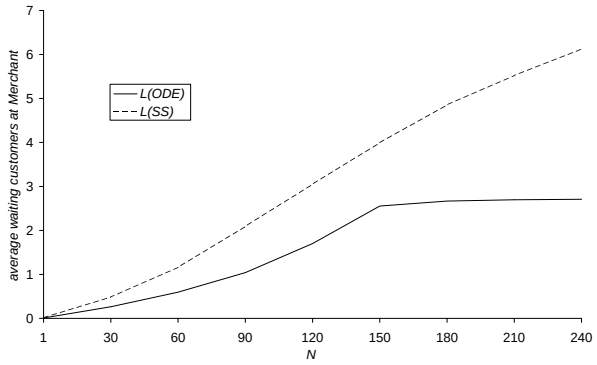

Fig. 3. Average number of waiting customers at merchant varied with population size calculated by ODEs and stochastic simulation, $p = 0.5$, $r_w = 0.01$ and all other rates are 1.

tem, and also demonstrates that this kind (optimistic) non-repudiation protocol could perform much better than those that always employ an on-line TTP.

Figure 5 shows the average response time for the merchant at different actions. Overall, if we increase total number of clients in the system, the merchant takes longer to process each individual request. However, the response time increases slowly, and that is caused by the queue length which has been shown in Figure 3. Following our functional rate definitions for the merchant (f_i), it is intuitively understood that queue length and response time should have the same increasing ratio. Moreover, more customers waiting for action *sendCg* and *sendCabort* than others, this gives longer a response time for these two actions.

We experiment to increase the capacity of the TTP to twice that shown before (2), and plot the results for average response time for the TTP in all actions and the merchant in action *sendCg* in Figure 6 and Figure 7. From Figure 6, it is clear that the response time for customers waiting at the TTP is smaller if the TTP

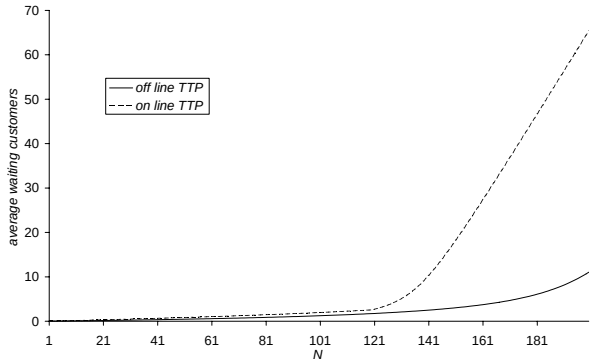

Fig. 4. Average number of waiting customers with and without TTP varied with population size calculated by ODEs, $p = 0.5$, $r_w = 0.01$ and all other rates are 1.

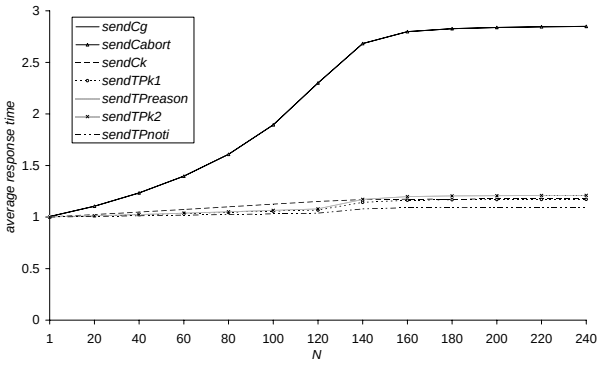

Fig. 5. Average response time at merchant varied with population size calculated by ODEs, $p = 0.5$, $r_w = 0.01$ and all other rates are 1.

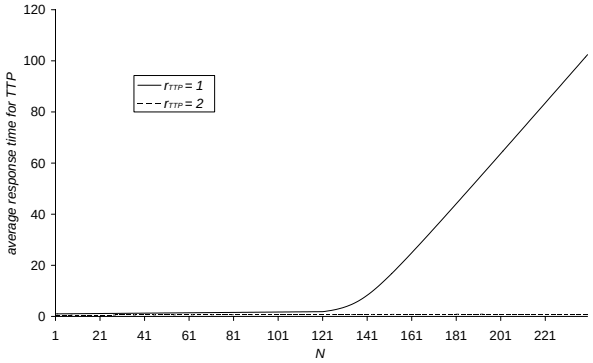

Fig. 6. Average response time for TTP varied with population size calculated by ODEs, $p = 0.5$, $r_w = 0.01$ and all other rates are 1 except for r_{TTP}.

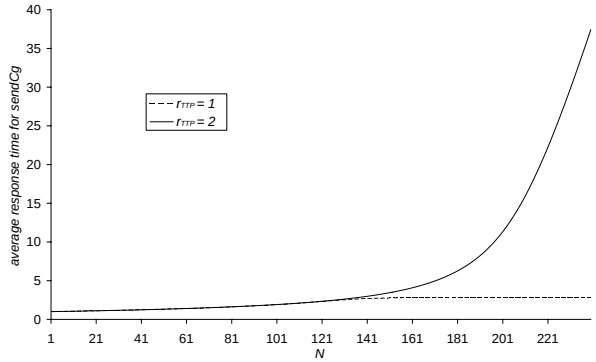

Fig. 7. Average response time for sendCg varied with population size calculated by ODEs, $p = 0.5$, $r_w = 0.01$ and all other rates are 1 except for r_{TTP}.

is more powerful. Nevertheless, the average response time for customers waiting at the merchant for action *sendCg* increases if we double the TTP's capacity, since the throughput from the TTP is obviously greater. A quicker response from the TTP means that the number of customers waiting at misbehaviour stage decreases. Under the same total number of clients, more customers go to the normal stage without misbehaviour. Consequently, the number of customers (CT_3) waiting for action *sendCg* increases, and the average response time for these customers takes longer.

Finally, we plot the proportion of satisfied customers (been served) in Figure 8. Generally, the proportion decreases for both case ($r_{TTP} = 1$ and $r_{TTP} = 2$) if more customers come to the system. The two curves are very close before the point, $N = 120$, and both keep a very high percentage of satisfied customers in this area. After that point, those percentages start to go down clearly. However,

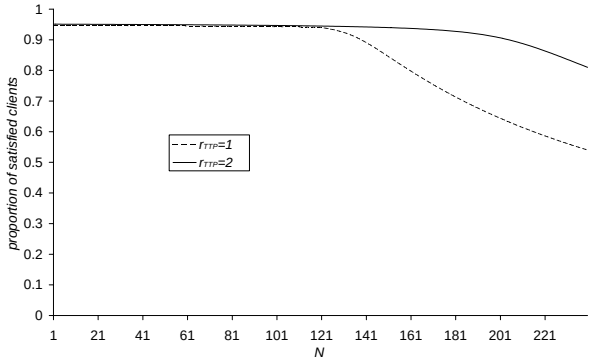

Fig. 8. Proportion of satisfied customers varied with population size calculated by ODEs, $p = 0.5$, $r_w = 0.01$ and all other rates are 1 except for r_{TTP}.

the proportion for $r_{TTP} = 1$ drops more quickly than the other, and it becomes 50% when $N = 240$, while the percentage for $r_{TTP} = 2$ is still above 80%.

3.5 Utility function of extended protocol

Consider the folloing utility function to answer our proposed performance questions for extended protocol.

$$C = c_1 L + c_2 K r_p \ , \ c_1, c_2 \geq 0 \tag{1}$$

Here, L denotes the average waiting customers at the non-repudiation sever (TTP), and K is number of servers. r_p is the response rate of the TTP. We assume the TTP server responds any type of jobs in the same rate here. C_1 and C_2 are cost rates, and they many depend on the type of system or quality of service agreement with customers.

Figure 9 shows the cost varied against the number of clients calculated by ODEs. Similar to the results of cost function in Chapter 3 and 4, more clients results in more waiting customers with fixed service capacity. Therefore, the total cost increases along with the cost of customer waiting goes up. Furthermore, it is a simple matter to find that the cost rises rapidly when N is around 130, and this is the maximum capacity that the TTP server can handle before performance start to significantly degrade.

Figure 10 presents the cost varied with number of TTP servers calculated by ODEs when total number of clients is 500. Again, customer waiting costs more in initial stage. Along with the system being given more servers, number of waiting clients is reduced. However, the cost of service dominate the total cost. The optimal point is around 2 in this case.

Figure 11 shows the cost varied with the rate of refresh key, r_w, calculated by ODEs. With fixed service capacity and total number of clients, more frequently refresh the session key results in more workload has been added in the system.

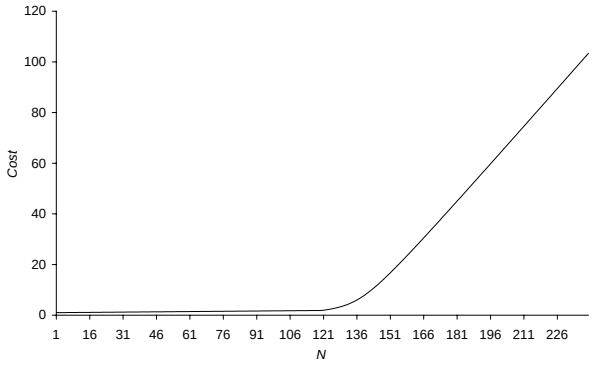

Fig. 9. Cost varied against the number of clients calculated by ODEs, $p = 0.5$, $K = 1$, $c1 = c2 = 1$, $r_w = 0.01$ and all other rates are 1.

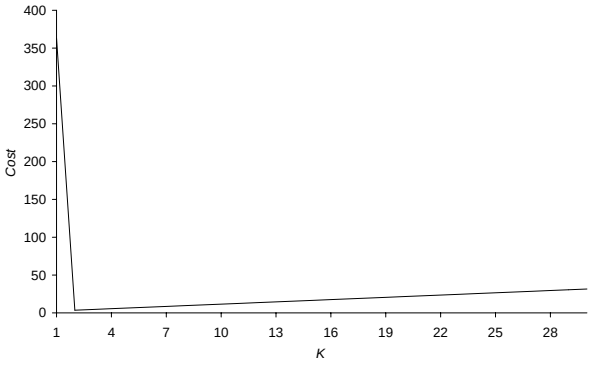

Fig. 10. Cost varied with number of TTP servers calculated by ODEs, $p = 0.5$, $N = 500$, $c1 = c2 = 1$, $r_w = 0.01$ and all other rates are 1.

Therefore, the cost of customer waiting increases. Similarly, we can easily find that the balance point between performance and security is around $r_w = 0.002$.

4 Conclusions

This case study has investigated an optimistic fair exchange protocol in an e-commerce environment. We model the protocol when the *third trust party* is online due to misbehaviour of one or more of the participants. According to the optimistic characteristic, the protocol can work in a lighter mode when there is no misbehaviour detected, where the TTP is not engaged. However this mode is much less interesting from a performance perspective. In this work, we consider that a merchant server consists of several threads; PEPA works well in this style of modelling. The ODE solution does not always coincide with stochastic simulation when N is very large. However, in this context, this large N only gives

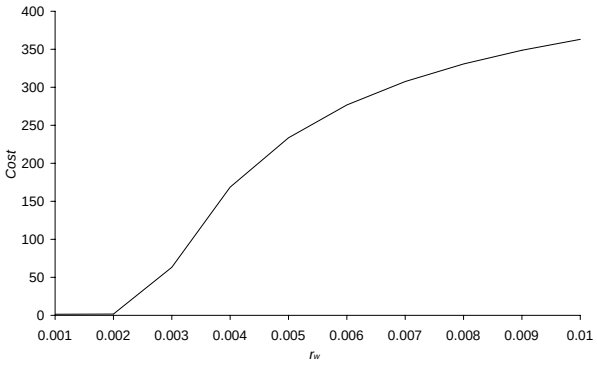

Fig. 11. Cost varied with rate of *work* calculated by ODEs, $p = 0.5$, $N = 300$, $c1 = c2 = 1$, all other rates are 1 except for r_w.

large scale for part of the derivatives, and they are still may converge under other rates. Despite this, the ODE solutions are shown to give a good indication of expected performance and can be derived extremely efficiently for large systems.

To date our analysis has focussed on identifying and employing efficient solution methods. There is considerable scope for further work to investigate the relationship between formal security models and formal performance models. The ultimate goal would be to create a system which could automatically produce analysable performance models from security models. However, the choice of security solution, driven by the performance security trade-off should always remain an expert task.

References

1. C. Bodei, M. Buchholtz, M. Curti, P. Degano, F. Nielson, H. Nielson and C. Priami, Performance evaluation of security protocols specified in LySa, *Electronic Notes in Theoretical Computer Science*, **112**, 2005.
2. J. Bradley, S. Gilmore and J. Hillston, Analysing distributed Internet worm attacks using continuous state-space approximation of process algebra models, *Journal of Computer and System Sciences*, **74**(6), 2008.
3. J. Bradley, S. Gilmore and N. Thomas, Performance analysis of Stochastic Process Algebra models using Stochastic Simulation, in: *Proceedings of 20th IEEE International Parallel and Distributed Processing Symposium*, IEEE, 2006.
4. M. Buchholtz, S. Gilmore, J. Hillston and F. Nielson, Securing statically-verified communications protocols against timing attacks, *Electronic Notes in Theoretical Computer Science*, **128**(4), Elsevier, 2005.
5. J. Cho, I. Chen and P. Feng, Performance analysis of dynamic group communication systems with intrusion detection integrated with batch rekeying in Mobile Ad Hoc Networks, in: *Proceddings of the 22nd International Conference on Advanced Information Networking and Applications*, IEEE, 2008.
6. M. El-Hadidi, N. Hegazi and H. Aslan, Performance analysis of the Kerberos protocol in a distributed environment, in: *Proceedings of the 2nd IEEE Symposium on Computers and Communications*, IEEE, 1997.

7. M. El-Hadidi, N. Hegazi and H. Aslan, Performance evaluation of a new hybrid encryption protocol for authentication and key distribution, in: *Proceedings of the International Symposium on Computers and Communications*, IEEE, 1999.
8. J. Hillston, *A Compositional Approach to Performance Modelling*, Cambridge University Press, 1996.
9. J. Hillston, Fluid Flow Approximation of PEPA models, in: *Second International Conference on the Quantitative Evaluaiton of Systems*, pp. 33-43, IEEE Computer Society, 2005.
10. W. Liu, L. Yang, Q. Li, H. Dai and B. Hou, Performance analytic model for authentication mechanism, in: *Proceedings of the International Conference on Networking, Sensing and Control*, pp. 1097-1102, 2008.
11. T. Meng, Q. Wang and K. Wolter, Security and Performance Tradeoff Analysis of Mobile Offloading Systems under Timing Attacks, in: *Proceedings of the 12th European Performance Engineering Workshop*, Springer, 2015.
12. I Ray and I. Ray, An Optimistic Fair Exchange E-commerce Protocol with Automated Dispute Resolution, in: *Proceedings of the First International Conference on Electronic Commerce and Web Technologies*, pp. 84-93, LNCS 1875, Springer, 2000.
13. P. Ryan, S. Schneider, M. Goldsmith, G. Lowe and B. Roscoe, *Modelling and Analysis of Security Protocols*, Addison Wesley, 2000.
14. N. Thomas, Performability of a secure electronic voting algorithm, *Electronic Notes in Theoretical Computer Science*, **128**(4), pp: 45-58, 2005.
15. N. Thomas and Y. Zhao, Fluid flow analysis of a model of a secure key distribution centre, in: *Proceedings 24th Annual UK Performance Engineering Workshop*, Imperial College, 2008.
16. Y. Wang, C. Lin and Q. Li, Performance analysis of email systems under three types of attacks, *Performance Evaluation*, **67**(6), 2010.
17. K. Wolter and P. Reinecke. Performance and security tradeoff, in: *Formal methods for quantitative aspects of programming languages*, pp. 135-167, LNCS 6154, Springer, 2010.
18. 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, 2008.
19. Y. Zhao and N. Thomas, Comparing Methods for the Efficient Analysis of PEPA Models of Non-repudiation Protocols, in: *Proceedings of the 15th International Conference on Parallel and Distributed Systems*, pp. 821-827, IEEE, 2009.
20. Y. Zhao and N. Thomas, Efficient solutions of a PEPA model of a key distribution centre, *Performance Evaluation*, **67**(8), pp. 740-756, 2010.