M. Carbonell, J. M. Sierra, J. A. Onieva, J. Lopez and J. Zhou, "Estimation of TTP Features in Non-repudiation Service", *7th International Conference on Computational Science and Its Applications (ICCSA07)*, vol. 4706, pp. 549-558, 2007.
http://doi.org/10.1007/978-3-540-74477-1_51
NICS Lab. Publications: https://www.nics.uma.es/publications

Estimation of TTP features in Non-repudiation service*

Mildrey Carbonell,

University of Carlos III Madrid
{mcarbone@inf.uc3m.es}

Jose A. Onieva†, Javier Lopez

Computer Science Department, E.T.S. Ingeniería Informática
University of Malaga, Spain
{onieva,jlm,galpert}@lcc.uma.es

Jianying Zhou

Institute for Infocomm Research, Singapore
jyzhou@i2r.a-star.edu.sg

Abstract. In order to achieve a high performance in a real implementation of the non-repudiation service it is necessary to estimate timeouts, TTP features, publication key time, number of originators and recipients, and other relevant parameters. An initial work of the authors focused on a basic event-oriented simulation model for the estimation of timeouts. In the actual work, we present a set of extensions to that basic model for the estimation of the TTP features (storage capacity and ftp connection capacity). We present and analyze the new and valuable results obtained.

1 Introduction

Most of the non-repudiation services solutions have been defined by means of a protocol using a Trusted Third Party (TTP). First solutions made use of the TTP in each of the steps of the protocol involving a high risk of communication bottleneck. Nevertheless, Zhou and Gollmann presented in [1] a protocol where the TTP intervenes during each execution as a "low weight notary". Other optimistic solutions, like [2], use an off-line TTP. They assume that the participating entities have no malicious intentions and that the TTP need to be involved if there is an exception in the protocol execution. There are solutions that eliminate the TTP's involvement [3]. However, these ones need a strong requirement like same computational power in all involved party or many rounds in the protocol execution. Therefore, the role of the TTP is essential for practical non-repudiation protocols, in one or another way.

* This work was partially supported by ASPECTS-M Project (Reference Models for Secure Architectures in Mobile Electronic Payments), CICYT-2004.
† The author has been funded by the Consejeria de Innovacion, Ciencia y Empresa (Junta de Andalucia) under the III Andalusian Research Plan.

Non-repudiation protocols include parameters whose values depend on the real conditions of each scenario or application. Some of these parameters are: timeouts, number of originators and receivers involved in the protocol, features of the TTPs (e.g., number of concurrent ftp connections and storage capacity), etc.

In a recent work [4], we have demonstrated how event-oriented simulation can be considered as a tool to estimate the timeouts of non-repudiation protocols. In this work we propose an extension of the simulation model and tests to estimate the appropriate values of the TTP features (storage capacity and ftp connection capacity). Additionally, this work pretends to further show the convenience of using the simulation techniques as a supporting tool for the correct implementation of non-repudiation protocols while, at the same time, proving the efficiency and fairness of the protocols in real scenarios.

The rest of the paper is organized as follows. In section 2, we present the specification of the event-oriented simulation model. In section 3, we present the new problems to solve with the model, and new entities variables. In section 4, we describe the principal simulation events related to the TTP process. Section 5 analyzes the results of the different tests. Finally, section 6 concludes the paper.

2 Simulation Model of the multi-party protocol

In order to show the results of our simulation model, we will make use a multi-party scenario because the set of events that take place in that type of scenario is, obviously, more complex than in a two-parties one. The first effort to generalize non-repudiation protocols to multi-party scenarios was presented in [5][6]. Next, other multi-party scenarios were presented in [7] [8]. We decide to use the first protocol [5]. This multi-party scenario is based on the existence of one originator (*O*) and several recipients (*R*). The protocol uses the same key *k* for each recipient, such that, an encrypted message *c*, evidence of origin (*EOO*), evidence of submission (Sub_k) and evidence of confirmation (Con_k) are generated for each protocol run. To ensure the fairness of the protocol, the key is only revealed to those recipients *R'* that replied with evidence of receipt (EOR_i). We use the same event-oriented simulation model like in the previous work [4]. Following we describe the steps and events of the protocol (fig 1).

The originator *O* multicasts to all recipients *R* the *EOO* corresponding to the encrypted message *c* in **step 1** <**event 1, 2** >. Next, the originator waits for *EOR* in **step 2** <**event 3** > and then send a key publication request to the TTP in **step 3** <**event 4** >. If the TTP has enough connections and storage capacity, it publishes the key and *O* is disconnected <**event 6** >, otherwise *O* will try the request later <**event 5**>. Once the key is published, the originator and the recipients can start *Con* requests in **step 4** and **step 5** <**event 7**>. If allowed by FTP resources, the TTP opens one connection for a *Con* request <**event 9, 10** >. Afterwards, the entity involved verifies the key of the message and outputs an affirmative or negative response to the request. Finally, the entity is disconnected **<event 14, 15** >. If FTP resources are overloaded, the involved entity should retry the connection later **<event 11, 12** >. The key is maintained publicly in the TTP’s database until timeout t1 <**event 13** >. When all involved entities have verified the key, the protocol run finishes (step 4 and step 5).

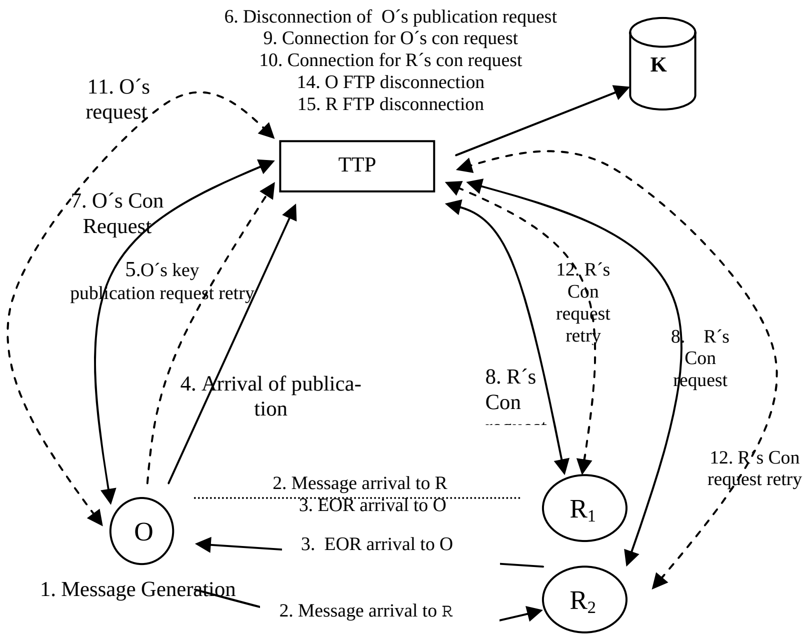

Fig. 1: Simulation Model

2 New Simulation problems and entities

Using the same model we have realized that it is important to reduce the key storage time in the TTP, as well as, to eliminate the unsuccessful confirmation requests while the number of originators and recipients increase or the TTP features (number of concurrent ftp connections and storage capacity) change. It is relevant to note that the elimination of unsuccessful confirmation requests guarantees a fair execution of the protocol.

For this reason, we study two new problems. Firstly, we will denote **P1** *as the problem of the estimation of efficient features in TTP without increasing the timeout t and while keeping all Con requests successful*. Similarly, we will denote **P2** *as the problem of calculating the estimated number of simultaneous originators and recipients without any changes in the critical parameters (TTP features, timeout t) while keeping all Con requests successful.*

In comparison with our previous work, we have not defined new entities for the model. However, we have added new variables that will allow us to use and store the values necessary for the new estimations. Following we describe all the entities including some important previous and new variables.

Table 1: Simulator entity

Entity 1: Simulator (S)	
Variables	**Description**
Input variables	
FinalTime	Final simulation time
MsgGenDist	List of message generation distributions for each *O*
CommunicationOR , *CommunicationOTTP* *CommunicationRTTP*	Matrix of delay distributions of network messages, between *O* and *R*, between *O* and the TTP, between *R* and the TTP
EORsendDist	Delay distribution of the *EOR* message
PUBConnectionDist	Time distribution of O's connection to publish the key in the TTP
FTPConnectionDist	FTP connection time distribution of *O* and *R*
State variables	
CurrentTime	Current simulation time
LEntity	List of entities
LEvent	List of events

Table 2: Message entity

Entity 2: Message (M): This entity is created by originators.	
Variables	**Description**
State variables	
CreationTime	Creation time
State	States of the message: St1 : It is being sent to R St2: O is waiting for *EOR* St3: O is trying to publish the key in the TTP St4: The key has been published in the TTP St5: The key was deleted from the TTP
InitTime_O_Con	Initial time of O's *Con* request (step 5)
Output variables	
PubDelayTime	Key publication delay time
DelayTime_O_Con	Delay time of O's *Con* request (successful or not)
Nbr_O_Con_Retries	Number of O's *Con* request retries (step 5)
Status_O_Con	Boolean value: True if O's *Con* request was successful (step 5); False if O's *Con* request was not successful (step 5)
DelayTime_R_Con	List of delay times of R´s Co*n* requests (step 4)
WaitRTime	Total waiting time for all *EOR*
Nbr_R_Con_Retries	List of R´s *Con* requests retries (step 4)

Table 3: Originator entity

Entity 3: ORIGINATOR (O)

Variables	**Description**
Input variables	
Time_btw_PUBRetries	Time between successive retries of O’s key publication requests
Time_btw_FTPRetries	Time between successive retries of O’s *Con* requests
Output variables	
Nbr_Successful_Con	Number of successful *Con* requests
Nbr_Unsuccessful_Con	Number of unsuccessful *Con* requests
Average_Con_Time	Average *Con* request time (step 5) (it is calculated using DelayTime_O_Con in the message)

Table 4: Recipient entity

Entidad 4 : RECIPIENT (R)	
Variables	**Description**
Input variables	
Time_btw_FTPRetries	Time between successive retries of R’s *Con* requests
Output variables	
Nbr_ReceivedMsg	Number of received messages
Nbr_Successful_Con	Number of successful *Con* requests
Nbr_Unsuccessful_Con	Number of unsuccessful *Con* requests
Average_Con_Time	Maximum *Con* request time (step 4)
LUnsuccessfulMsg	List of messages which could not be retrieved

Table 5: TTP entity

Entity 5: TTP	
Variables	**Description**
Input variables	
Max_StorageKTime	Key storage time in the TTP
CapacPUBConnection	Publication connection capacity
CapacFTPConnection	FTP connection capacity
CapacStorage	Storage capacity measured in number of keys
State variables	
Current_ConnectedPUB	Current number of publishing connected entities
Current_ConnectedFTP	Number of FTP connected entities
CapacOccupied	Occupied storage key capacity
Output variables	
LPublicMsg	List of messages whose keys were published
Nbr_PUBMsg	Number of messages whose keys were published
Nbr_PUBRetries	Number of retries of O’s key publication request caused by the lack of TTP connection capacity
Nbr_PUBRetries_Str	Number of retries of O’s key publication request caused by the lack of TTP storage capacity
Nbr_O_Con_Retries	Number of retries of O’s *Con* request

Nbr_R_Con_Retries	Number of retries of R's *Con* request
Nbr_Successful_O_Con	Total number of successful O's *Con* requests
Nbr_Unsuccessful_O_Con	Total number of unsuccessful O's *Con* requests
Nbr_Successful_R_Con	Total number of successful R's *Con* requests
Nbr_Unsuccessful_R_Con	Total number of unsuccessful R's *Con* requests

4 List of Model Simulation Events

Our previous work described seven events (labelled 1 to 7) that were closely related to the estimation of the key publication delay time. In the work described in the present paper we introduce events that are related to all the protocols steps that make use of the TTP. Events 7 to 15 were defined, but neither fully specified nor elaborated in the model presented in our previous work. In fact, the problems **P1** and **P2** explained in the previous section have required the redefinition of some of the events in order to include the new variables.

In this sense, the following includes the more representative events closely related to the sequence of events of the TTP. We exclude the *O Con request* event due to the similarity with the *R Con* request. We can use *entity.variable* to refer to one variable of any of the entities. For every event, we describe the name and the input parameters (between brackets) followed by the description of the event using a simple pseudo-language. All variables used are defined in the Tables included in the previous section.

EVENT 4: Arrival of the publication request to TTP (O: originator, M: mesage, TTP: trusted third party)

If *TTP.Current_ConnectedPUB + 1 > TTP.CapacPUBConnection*
    Increase *TTP.Nbr_PUBRetries*
  Add the event ***O's key publication request retry (O,M)*** at time
      t = S.CurrentTime + O.Time_btw_PUBRetries
Else
    If *TTP.CapacOccupied + 1 >TTP. CapacStorage*
        Increase *TTP.Nbr_PUBRetries_Str*
        Add the event ***O's key publication request retry (O,M)*** at time
            t = S.CurrentTime + O.Time_btw_PUBRetries
    Else
        Increase *TTP.Current_ConnectedPUB*
        Add the event ***Disconnection of O's publication request (O,M, TTP)*** at time
            t = S.CurrentTime + Random value generated `with S.PUBConnectionDist`

EVENT 8: R's Con request (M: message)

Update *M.DelayTime_R_Con*$[R_i]$*=S.CurrentTime*
Add the event ***Connection for R's Con request (R,M,TTP)*** at time
  t = S.CurrentTime + Random value generated with *S.CommunicationRTTP(R)*

EVENT 10: Connection for R's Con request(R: recipient, M: message, TTP: trusted third party)

If *TTP.Current_ConnectedFTP + 1 > TTP.CapacFTPConnection*
Increase *TTP. Nbr_R_Con_Retries*
Increase *M. Nbr_R_Con_Retries[Ri]*
Add the event ***R's Con request retry (M)*** at time
t = S.CurrentTime + R.Time_btw_FTPRetries
Else
Increase *TTP.Current_ConnectedFTP*
Add the event ***R's FTP disconnection (R,M, TTP)*** at time *t = S.CurrentTime* + Random value generated with *S.FTPConnectionDist*

EVENT 12: R's Con request retry (M: message)

Add the event **Connection for *R's Con requests (R,M)*** at time *t = S.CurrentTime* + Random value generated with *S.CommunicationRTTP(R)*

EVENT 13: Key deletion in the TTP (M: message)

Change the state of the message *M.State=St5*
Decrease *TTP.CapacOccupied*

EVENT 15: R's FTP disconnection (R: recipient, M: message, TTP: trusted third party)

If M is in the list *TTP.LPublicMsg* and *M.State=St4*
Increase *TTP.Nbr_Successful_R_Con*
Add M to the list *R.LSuccessfulMsg*
Increase *R.Nbr_Successful_Con*
Else
Increase *TTP.Nbr_Unsuccessful_R_Con*
Add M to the list *R.LUnSuccessfulMsg*
Increase *R.Nbr_Unsuccessful_Con*
Update *M.DelayTime_R_Con[Ri] = S.CurrentTime - M.DelayTime_R_Con[Ri]*
Decrease *TTP.Current_ConnectedFTP*

5 Analysis of Results

In the following tests we used the same protocol implementation, same input distribution variables, and same notations like in the previous work:

- S.MsgGenDist = Uniform distribution between 30 and 60 minutes.
- *S.CommunicationOR, S.CommunicationOTTP, S.CommunicationRTTP* = uniform distribution between 10ms and 17ms.
- *S.EORsendDist* = Uniform distribution between 15ms and 20ms.
- *S.PUBConnectionDist* = Uniform distribution between 30ms and 50ms.
- *S. FTPConnectionDist* = Uniform distribution between 25ms and 35ms.

Input variables

- NO, NR: Number of originators and Number of recipients
- C: TTP storage capacity measured in number of keys (*TTP. CapacStorage*)
- FTP: FTP connection capacity (*TTP. CapacFTPConnection*)
- TS: Key storage time in the TTP (*TTP.Max_StorageKTime*)

– RO, RR:Time between successive retries of O´s Con request (*O. Time_btw_FTPRetries*) of R´s Con request(R. Time_btw_FTPRetries)

Output variables

– NM: Number of generated messages in the experiment $\sum_{i=1}^{NO} O_i.Nbr\_Msg$
– MP: Number of messages whose keys were published on the TTP (*TTP.Nbr_PUBMsg*)
– CPC, CPA:Number of retries of O´s key publication request caused by the lack of TTP connection capacity (*TTP. Nbr_PUBRetries*) and request caused by the lack of TTP storage capacity (*TTP.Nbr_PUBRetries_Str*)
– CRO:Number of retries of O´s Con request (*TTP.Nbr_O_Con_Retries*)
– CRR:Number of retries of R´s Con request(*TTP.Nbr_R_Con_Retries*)
– SO: Number of successful O´s Con requests (*TTP.Nbr_Successful_O_Con*)
– SR: Number of successful R´s Con requests (*TTP.Nbr_Successful_R_Con*)
– UO: Number of unsuccessful O´s Con requests (*TTP.Nbr_UnSuccessful_O_Con*)
– UR: Number of unsuccessful R´s Con requests (*TTP.Nbr_UnSuccessful_R_Con*)
– ERT: Average waiting time of all EOR

$$\frac{\sum_{i=1}^{NO}\left(\frac{\sum_{j=1}^{Oi.Nbr\_Msg} Mj.WaitRTime}{NM}\right)}{NO}$$

– PKT: Average key publication delay time

$$\frac{\sum_{i=1}^{NO}\left(\frac{\sum_{j=1}^{Oi.Nbr\_Msg} Mj.PubDelayTime}{NM}\right)}{NO}$$

We have performed different tests (table 1) that have helped us to obtain good results for problem **P1**, and efficient conditions for the protocol operation.

– **Test 1 (A,B)** : In this test we use small increments for values of C and TS, and this has resulted in little changes in the unsuccessful Con requests (UO, UR).
– **Test 2 (D,E)**: In this test we increase the values of C, FTP and TS. The result is that we get a significant reduction of UO, UR.
– **Tets 3 (F):** In this test we increase the value of TS and obtain a reduction of UO and UR. However, the key has been published for long time (1 hour), and this is not a good solution for the Internet case. When analyzing the results, we deduct that an increment in the capacity of ftp connections (FTP) is necessary.
– **Test 4 (G)**: In this test we increase FTP. The result is that the unsuccessful Con requests become 0. Additionally, the number of retries of Con requests becomes 0 too. This guarantees a better execution of the protocol in a real scenario because of the reduction in the number of messages in the network.

– **Test 5 (H,I,J):** In this test we perform some estimations of the appropriate TS value. The result is that we get the best solution with a value of TS=50 seconds.
– **Test 6 (K):** In this test, we decrease the FTP value. The result is that we get unsuccessful Con requests again, what proves that the most appropriate value for FTP is 9000 key capacity.

Table 1: Result of the test

	Input variables						
	NO	**NR**	**C**	**FTP**	**TS**	**RO**	**RR**
A	300	30	**100**	140	**1/2min**	20s	20s
B	300	30	**450**	140	**3min**	20s	20s
D	300	30	**3500**	**3000**	**5min**	20s	20s
E	300	30	**4000**	**3000**	**20min**	20s	20s
F	300	30	4000	3000	**1h**	20s	20s
G	300	30	1500	**9000**	30min	20s	20s
H	300	30	1500	9000	**1min**	20s	20s
I	300	30	1500	9000	**1/2min**	20s	20s
J	300	30	1500	9000	**50s**	20s	20s
K	300	30	1500	**8000**	1/2min	20s	20s

Output variables									**Timeouts**		
NM	**ERT**	**CPC**	**CPA**	**CRO**	**CRR**	**SO**	**SR**	**UO**	**UR**	**ERT**	**PKT**
4712	10.67s	388	4895	3990	80388	3601	121540	**1040**	**14011**	10.67s	72.96s
4720	10.79s	869	0	3530	80502	4015	125108	**679**	**10560**	10.79s	54.77s
4628	4621	0	0	2220	7985	4220	133068	**302**	**5200**	10.75s	51.10s
4655	4652	0	0	2160	9239	4385	133461	**254**	**4099**	10.66s	50.40s
4593	4587	0	0	1990	6523	4387	132864	**140**	**3732**	10.72s	50.77s
4734	4733	0	0	**0**	**0**	4732	141930	**0**	**0**	10.62s	50.86s
4672	4669	0	0	0	0	4668	140041	**0**	**0**	10.75s	50.85s
4737	4734	0	0	0	0	4730	141916	**3**	**74**	10.75s	50.56s
4646	4641	0	0	0	0	4639	139140	**0**	**0**	10.85s	50.94s
4706	4703	0	0	**12**	**378**	4584	140498	**70**	**412**	10.77s	50.87s

We have used experimental values in order to avoid high time in the simulation execution.

6 Conclusions and future works

It is widely known that the role of the TTP is essential for many Internet security protocols. On the other hand, we know that most of non-repudiation protocols include parameters whose values are not easy to specify, and some of those parameters are directly related to the TTP. In a previous work we demonstrated how event-oriented simulation can be considered as a tool to estimate the timeouts of non-repudiation protocols.

In this work we have proposed an extension of the simulation model in order to estimate the appropriate values of the parameters for an efficient use of a TTP in non-repudiation protocols. The model has been proved with some tests that have helped, as we have shown throughout the paper, to estimate the most appropriate values for the simulated parameters.

At this moment we are working on new tests for the simulation of optimistic non-repudiation protocols, where taking into consideration the storage capacity of the TTP is not an essential issue.

References

1. J. Zhou and D. Gollmann, “A fair non-repudiation protocol”, IEEE Symposium on Research in Security and Privacy, pages 55-61, Oakland, CA, May 1996.
2. O. Markowitch, S. Kremer, “Optimistic non-repudiable information exchange”, In J. Biemond, editor 21st Symp. On Information Theory in the Benelux, pages 139-146, Wassenaar (NL), May 25-26 2000.
3. O. Markowitch and R. Yves, “Probabilistic non-repudiation without trusted third party”, Second Conference on Security in Communication Networks. Amalfi, Italy, September 1999.
4. M. Carbonell, J. Onieva, J. Lopez, J. Zhou, D. Galpert, “Simulation Model for the Estimation of timeouts in non-repudiation protocols”, ICCSA Workshop on Internet Communications Security, pages 903-914, LNCS 3043, May 2004.
5. O. Markowitch and S. Kremer, “A multi-party non-repudiation protocol”, Fifteenth IFIP International Information Security Conference, pages 271-280, Beijing, China, August 2000.
6. O. Markowitch and S. Kremer, “A multi-party optimistic non-repudiation protocol”, Third International Conference on Information Security and Cryptology, pages 109-122, Seoul, Korea, December, 2000.
7. J. Onieva, J. Zhou, and J. Lopez. “Non-repudiation Protocols for Multiple Entities”. Computer Communications, 27(16):1608—1616, Elsevier, October 2004.
8. J. Onieva, J. Zhou, and J. Lopez and M. Carbonell. "Agent-Mediated Non-repudiation Protocols". Electronic Commerce Research and Applications, 3(2):152--162, Elsevier, Summer 2004.