M. Carbonell, J. A. Onieva, J. Lopez, and J. Zhou, "Modelo de Simulacion para la Estimacion de Parametros en los protocolos de no Repudio", *III Simposio Español de Comercio Electronico (SCE05)*, pp. 151-164, 2005.
NICS Lab. Publications: https://www.nics.uma.es/publications

TÍTULO: Modelo de simulación para estimación de parámetros en los protocolos de no repudio

Mildrey Carbonell, Jose A. Onieva, Javier Lopez
Departamento de Ciencia de la Computación, E.T.S. Ingeniería Informática.
Malaga, España
{mildrey,onieva,jlm }@lcc.uma.es

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

Resumen. El no repudio es un requisito de seguridad cuya importancia se ha hecho evidente con el crecimiento del comercio electrónico. Muchos protocolos se han desarrollado como solución a este requisito. La gran mayoría incluye en su especificación parámetros cuyos valores no son fáciles de especificar pues dependen de las condiciones reales de implementación del mismo como los tiempos límites, las características de la TTP, tiempo de publicación de las claves, etc. En este trabajo proponemos un modelo que nos ayudará en la estimación de esos parámetros basado en la simulación del escenario real. Para la explicación y prueba del modelo mostramos un conjunto de experimentos.

1 Introducción

El no repudio es un servicio de seguridad que asegura que partes involucradas en un protocolo no puedan negar su participación. Este concepto es esencial para diversas aplicaciones de Internet especialmente el comercio electrónico donde se requiere que las disputas entre cliente y vendedor sean resueltas por medio de evidencias digitales. Este servicio requiere a su vez de un comportamiento justo donde ninguna de las partes involucradas en el protocolo tome ventajas sobre la otra relacionándose así el no repudio con el servicio de intercambio justo.

La mayoría de las soluciones al no repudio se definen por medio de un protocolo que hace uso de una (tercera entidad confiable) como un intermediario confiable entre los participantes. En las primeras soluciones la TTP participaba en cada uno de los pasos del protocolo, almacenando todas las evidencias y provocando un cuello de botella en la comunicación. La primera solución que disminuyó la utilización de la misma fue presentada por Zhou y Gollmann en [1]. A partir de entonces la implementación del no repudio está más cercana a la realidad y cada vez se desarrollan más protocolos que representan escenarios extendiéndose incluso a ambientes multipartes y escenarios móviles.

Al mencionar los escenarios multipartes debemos hacer referencia a que los primeros esfuerzos comenzaron con los protocolos de intercambio justo [2,3]. En no repudio la primera solución lo constituye el trabajo de Markowitch y Kremer [4] de envío del mismo mensaje a varios receptores. Más tarde una extensión a diferentes mensajes fue presentado en [5], así como otras soluciones multipartes que se han desarrollado con posterioridad [6].

Hasta el momento aunque existen algunos esfuerzos por eliminar la participación de la TTP [7,8] inclusive en escenarios multipartes [9], hasta ahora, todas las propuestas exigen requerimientos muy fuertes y difíciles de obtener en escenarios reales. Por tal motivo, el papel de la TTP aun continua siendo esencial en los protocolos de no repudio.

Al estudiar los protocolos de no repudio encontramos diversos parámetros cuyo valor no es fácil de especificar pues dependen de las condiciones específicas de los escenarios de aplicación. Como podemos ver valores como: tiempos límites, las características idóneas de la TTP (como el número de conexiones simultáneas al servicio de *ftp,* capacidad de almacenamiento) y el número de orígenes / receptores que se puede asimilar bajo ciertas condiciones iniciales, etc dependen directamente del escenario.

En este trabajo proponemos un modelo de simulación para el cálculo de esos parámetros. Además destacamos, por medio de un ejemplo, su utilidad e integración con las condiciones reales de cada implementación. Se eligió para este ejemplo el protocolo Kremer and Markowitch, que se expone en el epígrafe 2, por ser el primer trabajo de no repudio multiparte desarrollado. En el epígrafe 3 se presentan especificaciones del modelo de simulación, las entidades y los eventos. Concluimos el trabajo mostrando en el epígrafe 4 experimentos de pruebas que muestran los resultados obtenidos con diversas características del escenario de aplicación.

2 Escenario multiparte uno-muchos con igual mensaje

El protocolo de no repudio de Markowicth [2] que a continuación mostramos constituye la primera generalización del no repudio a ambiente Multiparte. El mismo propone el uso de una única clave para enviar el mismo mensaje a un grupo de receptores. Esto provoca la creación de un solo mensaje cifrado C, una evidencia EOO, una Sub_k y una Con_k para cada ejecución del protocolo. En su diseño se presenta el problema de ejecución justa en caso de que no todos los receptores envíen evidencia de recibo del mensaje cifrado, solo un conjunto R' y al publicar la clave k en la TTP, todos puedan tomarla. La solución que se propone es publicar una encriptación en grupo [10] de la clave k ($E_P(k)$), donde el grupo P serán los R'.

Notación

$\boldsymbol{X \Rightarrow \Pi}$ - X envía a todos los miembros de Π
$\boldsymbol{E_\pi()}$ - Esquema E de encriptación en grupo, que puede ser descifrado por cada $P \in \Pi$
$\boldsymbol{O}$ - El que envía los mensajes
$\boldsymbol{l = H(m,k)}$ – etiqueta del mensaje

t – Tiempo límite de entrega de la clave. No se publicará *k* un tiempo después de *t*
R - Conjunto de entidades receptoras
R' - Conjunto de entidades receptoras que envían evidencia de recibo
EOO = $S_O(f_{Eoo}, R, l, t, c)$ - Evidencia de origen del mensaje *c*
EOR $_i$ = S_{Ri} $(f_{Eor}; O; l; t; c)$ - Evidencia de recibo del mensaje *c*
Sub $_k$ = $S_O(f_{Sub}; R', l, t, E_{R'}(K))$ - Evidencia de entrega de la clave *k*
Con $_k$ =$S_{TTP}(fcon, O, R', l, t, E_{R'}(k))$ - Evidencia de confirmación de la clave *k*

Protocolo

1. $O \Rightarrow R : feoo, R, l, t, c, EOO.$
2. $R_i \rightarrow O: feor, O, R_i, l, EOR_i$ donde $R_i \in R$ and $i \in \{1, ...|R|\}$
3. $O \rightarrow TTP: fsub, R', l, t, E_{R'}(k), Sub_k$
4. $R'_i \leftrightarrow TTP: fcon, O, R', l, E_{R'}(k), Con_k$ donde $R'_i \in R' \ \forall i : 1<=i<=|R'|$
5. $O \leftrightarrow TTP: fcon, O, R', l, E_{R'}(k), Con_k$

En el paso 1 el origen *O* envía a todos los receptores *R* el mensaje *c* junto a la evidencia de origen. En el paso 2 algunos (o todos) los receptores envían la evidencia de recibo EOR_i. Seguidamente *O* envía la clave y la evidencia de envío Sub_k a la TTP en el paso 3 con el objetivo de obtener la evidencia de confirmación Con_k en el paso 5. En este protocolo se asume que el canal de comunicación entre *O* y *TTP* no está permanentemente interrumpido, por lo tanto, *O* podrá entregar la clave *k* a la TTP en un tiempo anterior a *t*. Luego los receptores podrán obtener la clave $E_{R'}(k)$ junto a la evidencia Con_k para culminar la prueba de origen del mensaje en el paso 5.

El tiempo *t* constituye un criterio de parada del protocolo en caso de la TTP recibir la clave después de *t*. A su vez es usado por los receptores para abortar la solicitud de clave a la TTP si pasado ese tiempo no está pública. Este valor *t* no está especificado en el protocolo y constituye uno de los ejemplos de parámetros que dependen de las características del escenario de aplicación. En la siguiente sección proponemos un modelo que ayudará a su estimación.

3 Modelo de simulación

El modelo de simulación que proponemos nos ayuda a estimar el tiempo *t* y a realizar diagnósticos sobre el protocolo de no repudio para detectar ineficiencias, a partir del muestreo de las variables críticas del sistema como: demora de los mensajes en la red (provocadas por la velocidad de la red, o por sistemas de protección como cortafuegos, esquemas de protección), limitaciones en las capacidades de la TTP, cantidad de entidades simultaneas a ser atendidas, tiempos límites previstos, entre otras. Una vez realizado el diagnóstico, la propia herramienta resulta útil en la identificación de las mejoras a las que es susceptible el sistema y sus variables de entrada.

3.1 Definición de los eventos y los problemas a solucionar

Utilizamos un modelo de simulación de eventos discretos [11] pues la generación y recepción de los mensajes son procesos asíncronos que involucran a un número finito de eventos. A continuación detallamos esos eventos.

- El origen envía mensajes a los receptores (***evento 1: generación de mensajes, evento 2: llegada de mensajes a R***)
- El origen espera por las evidencias *EOR* (***evento 3. Llegada de EOR a O***) y luego pasado un tiempo t_1 de espera por las respuestas de los R envía la solicitud de publicación de la clave a la TTP (***evento 4: Llegada de solicitud de publicación de clave a la TTP***).
- Si es posible realizar la conexión *ftp* con la TTP y la misma tiene suficiente capacidad de almacenamiento, la clave es publicada y luego *O* se desconecta (***evento 6: desconexión ftp de O***); si no es posible *O* intentará más tarde conectarse.(***evento 5: reintento de solicitud de publicación de clave***)
- Después que la clave es publicada, *O* y los *R* comienzan el proceso de solicitud de la clave junto a la evidencia *Con* (***evento 7: Solicitud de evidencia Con por O)(evento 8: Solicitud de evidencia Con por R***)
- Si es posible realizar la conexión *ftp*, se abre una conexión para solicitar la clave junto a la evidencia *Con* (***evento 9: Conexión de O para solicitud de evidencia Con, evento 10: Conexión de R para solicitud de evidencia Con***). Luego la entidad involucrada verifica la clave en la TTP y se desconecta (***evento 14: Desconexión ftp de O, evento 15: Desconexión ftp de R***).
- Si la *ftp* está saturada, las entidades intentan conectarse más tarde (evento ***11: Reintento de O de solicitud de evidencia Con, evento 12: Reintento de R de solicitud de evidencia Con***). La clave es mantenida pública en la base de datos de TTP hasta el tiempo límite t_2.(***evento 13: Borrado de la clave en la TTP***)

En un escenario real, la TTP necesita procesar varias ejecuciones del protocolo con diferentes orígenes. En este modelo el criterio de parada será el tiempo.

A continuación presentamos dos de los problemas estudiados con este modelo. El primero hace un análisis más pasivo de las condiciones reales del escenario, mientras que el segundo necesita de diversos experimentos para realizar las estimaciones que brinden una mejor solución.

P1: Estimación de los tiempos límites *t* (que *O* envía a los R en el primer paso del protocolo) y tiempo t_2 (que mantendrá la TTP la clave publicada) , dado como entrada los valores de un escenario real de aplicación.

P2: Estimación de las características eficientes en la TTP (número de conexiones simultaneas al servicio de *ftp*, capacidad de almacenamiento) sin incrementos en el tiempo límite *t* y manteniendo todas las búsquedas de clave y evidencia *Con* exitosas.

Las entidades que se define en el modelo de simulación son las tres entidades participantes en el protocolo origen, receptor y TTP, la entidad mensaje que es creada por los orígenes y la entidad simulador encargada de los datos generales del modelo que no pertenecían a ninguna de las anteriores.

- La entidad **simulador** (*S*) recibe como entrada los datos referentes a cantidad de orígenes y receptores que se simularan junto a las distribuciones que siguen la generación de mensajes, su trasmisión, los envío de EOR y las conexiones con TTP para publicar o buscar evidencias. También recibe el tiempo final de la simulación que constituye el criterio de parada de la misma. En el proceso de simulación se encarga de mantener el listado de eventos y el tiempo actual.
- La entidad **Mensaje** *(M)* almacena todos los datos referentes a su tiempo de creación, estado en que se encuentra dentro del protocolo (enviándose a R, esperando por EOR, intentando publicarse, publicado, borrado, etc.), número de evidencias de recibo enviadas y tiempo inicial de búsqueda de *Con*. A su vez recoge variables de salida esenciales para los estudios estadísticos a realizar una vez culminada la simulación como los tiempos de demora publicación y de solicitud de *Con* y sus cantidades de reintentos tanto por los orígenes como receptores.
- La entidad **Origen** *(O)* recibe como entrada los datos específico a los tiempos entre sucesivos reintentos de publicación y de búsqueda de la evidencia *Con*. Mantendrá durante todo el proceso de simulación la lista de mensajes que ha generado. Como salida brindará el numero de claves que ha publicado exitosamente y la cantidad de éxitos o fallos en la búsqueda de la evidencia *Con.*
- La entidad **Receptora (***R***)** recibe como entrada el tiempo entre sucesivos reintento de búsqueda de la evidencia *Con*. Como salida brinda el número de mensajes recibidos , así como la cantidad de éxitos o fallos en la búsqueda de la evidencia *Con*.
- La entidad **TTP** recibe como entrada los datos referentes a su capacidad de almacenamiento y de conexión, junto al tiempo de publicación de la llave. Mantendrá durante el proceso un contador de las entidades conectadas y el espacio de almacenamiento ocupado. Como salida brindará el número de mensaje cuya clave fue publicada, el total de solicitudes *Con* exitosas y fallidas de *O* y *R*, así como el número de reintentos de publicación y de búsqueda de la evidencia *Con* .

Las variables de las entidades se especifican más detalladamente en el ANEXO 1.

3.2 Eventos del modelo de simulación

El modelo fue implementado utilizando programación orientada a objeto en Delphi 6.0. A continuación realizamos una breve explicación de los eventos más importantes del modelo y su interrelación. Una descripción más detallada de se muestra en el ANEXO 2.

EVENTO 1: **Generación de mensajes (*O*: origen)** Constituye el evento de partida del modelo de simulación. Constantemente se están generando mensajes para ser enviados a los receptores siguiendo la distribución dada como entrada al modelo. En su implementación genera los próximos eventos de ***Llegada de mensajes a R (O,M,R_i)*** para cada receptor R_i y el próximo evento de ***Generación de mensaje***.
EVENTO 2: Llegada de mensajes a *R* (*O*: origen , *M*: mensaje, *R*: receptor) Este evento actualiza la cantidad de mensajes recibidos por *R* y seguidamente genera un evento de ***Llegada de EOR a O (M,R)*** utilizando la distribución de demora de mensaje entre *O* y *R*.
EVENTO 3: Llegada de *EOR* a *O* (*M*: Mensaje, *R*: receptor)

Este evento actualiza la cantidad de respuesta de *EOR* del mensaje y los tiempos de espera por *EOR,* este dato es de esencial importancia en la salida del modelo. Genera un evento de ***Llegada de solicitud de publicación de clave a la TTP (O, M, TTP)*** usando la distribución de demora de mensajes entre *O* y la *TTP*
EVENTO 4: Llegada de solicitud de publicación de clave a la TTP (*O*: Origen, *M*: mensaje, *TTP*: tercera parte confiable) Este evento permite conectarse y publicar la clave si hay capacidad de conexión y de almacenamiento en la TTP, generando luego un evento de ***Desconexión ftp de O (O,M, TTP)*** usando la distribución de demora de conexión en la TTP. En caso de no poder conectarse se genera un evento de ***Reintento de solicitud de publicación de clave (O,M)*** usando el tiempo entre reintentos sucesivos de la entidad *O*
EVENTO 6: Desconexión ftp de *O* (*O*: origen, *M*: mensaje, *TTP*: tercera parte confiable) Este evento permite actualizar el tiempo de demora de publicación y la cantidad de llaves que han tenido éxito en la publicación. Genera los eventos ***Solicitud de evidencia Con por O (M), Solicitud de evidencia Con por R (M)*** que dan paso a al proceso de búsqueda de evidencia de publicación del protocolo. A su vez genera el evento de ***Borrado de la clave en la TTP (TTP,M)*** usando el tiempo que se le asignó a la publicación de la llave
EVENTO 9: Conexión de *O* para solicitud de evidencia *Con* (*O*: origen, *M*: mensaje, *TTP*: tercera parte confiable*)* Este evento analiza la posibilidad de conexión en la TTP, si es posible genera ***Desconexión FTP de O (O,M,TTP)*** usando la distribución de conexión en la TTP. Si no es posible genera un evento de ***Reintento de O de solicitud de evidencia Con (M.)***
EVENTO 14: Desconexión FTP de *O*(*O*: origen, *M*: mensaje, *TTP*: Tercera parte confiable**)** Este evento actualiza los datos de solicitudes de *Con* exitosas o fallidas, así como el tiempo de obtención de estas evidencias.

4. Análisis de resultados

Para comprobar el modelo implementamos un ejemplo genérico del protocolo con varias cantidades de entidades involucradas y con una generación de mensajes entre ½ hora a 1 hora. Se realizaron más de 100 ejecuciones obteniéndose los siguientes datos iniciales de distribuciones:

- La distribución que sigue el tiempo de demora de paquetes en la red entre *O* y *R*, *O* y *TTP*, *R* y *TTP* es una distribución uniforme entre 10ms1 y 17ms.
- La distribución de los tiempos de análisis del mensaje recibido por los *R* y la creación del paquete de respuesta de evidencia de recibo a los Orígenes.(Distribución uniforme entre 15s y 20s)
- La distribución que siguen los tiempos de demora de conexión de los *O* en la TTP para publicar la clave (Distribución uniforme entre 30s y 50s).
- La distribución que siguen los tiempos de demora de conexión de los *R* y de los *O* en la TTP para buscar la evidencia de publicación de la clave (*Con*). (Distribución uniforme entre 25s y 35s)

[1] Milisegundos.

Se realizaron luego diversas ejecuciones del modelo de simulación (bajo la técnica de réplicas independientes) para obtener soluciones a los problemas planteados en la sección 3.1 (P1, P2). A continuación describimos la nomenclatura que se usará en las tablas de datos de los experimentos.

Nomenclatura

NO: Número de orígenes (*S.Origenes)*
NR: Número de receptores (*S.Receptores)*
C: Capacidad de almacenamiento de llave en la TTP (*TTP. CapAlmacenamiento*)
FTP: Capacidad de conexión FTP (*TTP. CapConexion_FTP*)
TS: Tiempo de publicación de la clave (*TTP.Max_TiempoBD_k*)
RO: Tiempo entre sucesivos reintentos de solicitud de *Con* por *O* (*O. TReintFTP*)
RR: Tiempo entre sucesivos reintentos de solicitud de *Con* por *R* (*R. TReintFTP*)
NM: Cantidad de mensajes generados en el experimento. $\sum_{i=1}^{NO} O_i.NMsj$
MP: Cantidad de mensajes publicados en la TTP. (*TTP.N_MsjPUB*)
CPC: Cantidad de reintentos de publicación que realizaron los *O* por falta de capacidad de conexión en la TTP. (*TTP. NReintPUB*)
CPA: Cantidad de reintentos de publicación por falta de capacidad de almacenamiento en la TTP. (*TTP.NDemPUB_Alm*)
CRO: Cantidad de reintentos de *O* en la búsqueda de la evidencia *Con*. (*TTP.NReint_O_Con*)
CRR: Cantidad total de reintentos en la búsqueda de la evidencia Con. (*TTP.NReint_R_Con*)
EO: Cantidad de veces que los *O* tuvieron éxito en la recogida de la evidencia *Con*. (*TTP.N_Exitosos_O_Con*)
ER: Cantidad total de veces que los *R* tuvieron éxito en la recogida de la evidencia *Con*. (*TTP.N_Exitosos_R_Con*)
FO: Fallos de los *O* en la recogida de evidencia *Con*.(*TTP.N_NoExitosos_O_Con*)
FR: Fallos de los *R* en la recogida de evidencia *Con*. (*TTP.N_NoExitosos_R_Con*)
PER: Promedio de espera de los *O* por *EOR* de todos los *R*.

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

PP – Promedio de tiempo de demora de publicación de las claves en la TTP.

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

Grupo 1: Experimentos para solucionar el P1 (estimación del tiempo *t***)**

Este experimento requiere de la entrada de los datos del escenario y a partir de esto se estima el valor apropiado para *t* (**PP**) *y* t_1 (**PER**).

	Variables de entrada						
	NO	**NR**	**C**	**FTP**	**TS**	**RO**	**RR**
A	300	30	10500	9000	1min	20s	20s
B	5000	30	10500	9000	2min	20s	20s
C	10000	10	10500	9000	1min	20s	20s

Variable de salida										Tiempos límites	
NM	**MP**	**CPC**	**CPA**	**CRO**	**CRR**	**EO**	**ER**	**FO**	**FR**	**PER**	**PP**
4672	4669	0	0	0	0	4668	140041	0	0	10.75s	50.85s
76885	76833	0	0	0	0	76816	2304481	0	0	11.93s	51.97s
157850	157775	2000	0	0	0	157739	1577370	0	0	10.50	60.20s

La simulación en estos casos arrojó que los orígenes no deben esperar más de:

Caso A: 10.75s, **Caso B:** 11.93s, **Caso C:** 10.50s

El tiempo de publicación de clave será

Caso A: 50.85s, **Caso B:** 51.97s**, Caso C: 60.20s**

Grupo 2: Experimentos para solucionar P2 (Estimación de las características eficientes en la TTP).

Como este grupo de experimentos requiere de variaciones en diversos parámetros de entrada para hacer estimaciones eficientes en los valores buscados, dividimos los experimentos es grupos de variaciones. Aunque se realizaron muchos más experimentos mostramos aquellos que consideramos más significativos para la comprensión del uso del modelo.

Variación 1: Realizamos pequeños incrementos en los valores de **C** y **TS** y esto resulta en pocas variaciones en las solicitudes fallidas de *Con* (**FO, FR**)

	Variable de entrada						
	NO	**NR**	**C**	**FTP**	**TS**	**RO**	**RR**
A	300	30	100	140	1/2min	20s	20s
B	300	30	450	140	3min	20s	5s

Variable de salida									Tiempo límites		
NM	**PER**	**CPC**	**CPA**	**CRO**	**CRR**	**EO**	**ER**	**FO**	**FR**	**PER**	**PP**
4712	10.67s	388	4895	3990	80388	3601	121540	1040	14011	10.67s	72.96s
4720	10.79s	869	0	3530	80502	4015	125108	879	12560	10.79s	54.77s

Variaciones 2: Incrementos en los valores **C** producen reducciones en los valores **FO** y **FR** pero aun no desaparecen totalmente.

	Input variables						
	NO	**NR**	**C**	**FTP**	**TS**	**RO**	**RR**
D	300	30	3500	3000	5min	20s	20s
E	300	30	4000	3000	5min	20s	5s

Output variables										Timeouts	
NM	**PER**	**CPC**	**CPA**	**CRO**	**CRR**	**EO**	**ER**	**FO**	**FR**	**PER**	**PP**
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

Variaciones 3: Incrementos en el valor **TS** provoca mayor reducción en **FO** y **FR**. En cambio, la clave necesita estar publica demasiado tiempo siendo no apropiado para escenarios reales.

	Input variables						
	NO	**NR**	**C**	**FTP**	**TS**	**RO**	**RR**
F	300	30	4000	3000	1h	20s	10min

Output variables										Timeouts	
NM	**PER**	**CPC**	**CPA**	**CRO**	**CRR**	**EO**	**ER**	**FO**	**FR**	**PER**	**PP**
4593	4587	0	0	1990	6523	4387	132864	140	3732	10.72s	50.77s

Con estos ejemplos se deduce que un aumento en la capacidad de conexión al servicio *ftp* es necesario

Variación 4: Aumento en la FTP da como resultado que **FO** y **FR** sean 0, garantizando la reducción de mensajes en la red en busca de las claves.

	Input variables						
	NO	**NR**	**C**	**FTP**	**TS**	**RO**	**RR**
G	300	30	1500	9000	30min	20s	1min

Output variables										Timeouts	
NM	**PER**	**CPC**	**CPA**	**CRO**	**CRR**	**EO**	**ER**	**FO**	**FR**	**PER**	**PP**
4734	4733	0	0	0	0	4732	141930	0	0	10.62s	50.86s

Variación 5: Ahora realizamos estimaciones del valor **TS** en busca de valores apropiados**,** encontrando como mejor solución **TS**=50s

	Input variables						
	NO	**NR**	**C**	**FTP**	**TS**	**RO**	**RR**
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

Output variables										Timeouts	
NM	**PER**	**CPC**	**CPA**	**CRO**	**CRR**	**EO**	**ER**	**FO**	**FR**	**PER**	**PP**
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

En los experimentos no trabajamos con valores altos de *O* y *R* pues deseábamos mostrar el uso del modelo y no realizar un experimento real cuyas estimaciones podrían ser muy costosas. Como el modelo puede ser usado para la estimación de diversos parámetros constituye una herramienta útil para el estudio de todos los valores que el desarrollador necesite estimar sin necesidad de esperar a la ejecución del mismo en un ambiente real.

5. Conclusiones

Un aspecto esencial para la adecuada implementación y ejecución del protocolo lo constituye el cálculo de los tiempos límites. En este artículo propusimos un modelo de simulación cuyo objetivo es la estimación de este tiempo límite así como otros parámetros esenciales en la implementación del protocolo. El mismo permite representar las características reales del escenario en que se aplicará. Se describe su uso y utilidad a través de diversos experimentos que además permitieron en la

validación del modelo. Los valores de entrada elegidos en los ejemplos no fueron muy elevados pues se pretendía dar una muestra de su uso y no solucionar un escenario específico cuyas estimaciones podrían ser muy costosas

Este modelo de simulación puede extenderse a otros protocolos de seguridad en escenarios de dos partes o multipartes. En trabajos futuros serán modelado otros protocolos cuya complejidad en eventos puede ser superior como no repudio con intermediario.

Referencias

1. J. Zhou and D. Gollmann. “A fair non-repudiation protocol”. Proceedings of 1996 IEEE Symposium on Research in Security and Privacy, pages 55-61, Oakland, CA, May 1996.
2. N. Gonzalez-Deleito and O. Markowitch. “An optimistic multi-party fair exchange protocol with reduced trust requirements”. Proceedings of 4th International Conference on Information Security and Cryptology, pages 258–267, Seoul, Korea, December 2001.
3. J. Kim and J. Ryou. “Multi-party fair exchange protocol using ring architecture model”. Proceedings of Japan-Korea Joint Workshop on Information Security and Cryptology, January 2000.
4. O. Markowitch and S. Kremer. “A multi-party non-repudiation protocol”. Proceedings of 15th IFIP International Information Security Conference, pages 271-280, Beijing, China, August 2000.
5. J. Onieva, J. Zhou, M. Carbonell, and J. Lopez. “A multi-party non-repudiation protocol for exchange of different messages”. Proceedings of 18th IFIP International Information Security Conference, Athens, Greece, May 2003.
6. J. Onieva, J. Zhou, M. Carbonell, and J. Lopez.. "Agent-Mediated Non-repudiation Protocols". Electronic Commerce Research and Applications, 3(2):152--162, Summer 2004.
7. 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.
8. O. Markowitch and R. Yves, “Probabilistic non-repudiation without trusted third party”, Second Conference on Security in Communication Networks. Amalfi, Italy, September 1999.
9. O. Markowitch and S. Kremer. “A multi-party optimistic non-repudiation protocol”. Proceedings of 3rd International Conference on Information Security and Cryptology, pages 109-122, Seoul, Korea, December, 2000.
10. G . Chiou and W. Chen. “Secure broadcasting using the secure lock”. IEEE Transaction on Software Engineering, Vol. 15, No. 8, August 1989.
11. J. Banks, J. Carson, and B. Nelson. “Discrete-event system simulation”. Prentice Hall, 2000.

Anexo 1: Entidades del sistema

Entidad 1: Simulador (S): Variable del sistema de simulación

Variables de entrada

TFinal	Tiempo final de la simulación
Receptores	Número de receptores (R)
Origenes	Número de orígenes (*O*)
DGenMsj	Lista de distribución de generación de mensajes de cada *O*
DcomunicacionOR, DComunicacionOTTP, DComunicacionRTTP	Matrices de distribución de demora de los mensajes en la red entre (O / R), (O / TTP) y (R / TTP)

DEnvEOR	Distribución de demora de los envíos de *EOR*
DConexionPUB	Distribución de los tiempos de demora de conexión de los *O* para publicar la clave en la TTP
DConexionFTP	Distribución de los tiempos de conexión *ftp* de *O* y *R*
Variables de estado	
TAct	Tiempo actual de simulación
LEvent	Lista de eventos del modelo

Entidad 2: Mensaje (M): Esta entidad es creada por los orígenes

Variables de estado	
TiempoCreacion	Tiempo de creación
N_EOR	Número de *R* que han enviado *EOR*
TInic_O_Con	Tiempo inicial de la solicitud por *O* de la evidencia *Con*
Variables de salida	
TEspEOR	Tiempo total de espera por todos los *EOR*
TDemPUB	Tiempo de demora de publicación de la clave
NDemPUB	Número de solicitudes de publicación de la clave
TDem_O_Con	Tiempo de demora de solicitud de *Con* por *O*
NReint_O_Con	Número de reintentos de *O* en la solicitud de *Con*
Estado_O_Con	Este valor es verdadero si la solicitud de Con por *O* es exitosa y falso en caso contrario

Entidad 3: Origen (*O*)

Variables de entrada	
TReintPUB	Tiempo entre sucesivos reintentos de solicitud de publicación de clave por *O*
TReintFTP	Tiempo entre sucesivos reintentos de solicitud de *Con* por *O*
Variables de estado	
LMsj	Lista de mensajes generado por *O*
NMsj	Número de mensajes generado por *O*
Variables de salida	
N_MsjPUB	Número de claves publicadas
N_Exitosos_Con	Número de exitosas solicitudes de *Con*
N_NoExitosos_Con	Número de no exitosas solicitudes de *Con*

Entidad 4 : Receptora (R)

Variables de entrada	
TReintFTP	Tiempo entre sucesivos reintentos de solicitud de *Con*
Variables de estado	
LMsjRecibidos	Lista de mensajes recibidos
Output variables	
N_MsjRecibidos	Número de mensajes recibidos
N_Exitosos_Con	Número de exitosas solicitudes de *Con*
N_NoExitosos_Con	Número de no exitosas solicitudes de *Con*

Entidad 5: TTP

Variables de entrada	
Max_TiempoBD_k	Tiempo de almacenamiento de la clave en la TTP
CapConexion_PUB	Capacidad de conexión para publicación
CapConexion_FTP	Capacidad de conexión *ftp*
CapAlmacenamiento	Capacidad de almacenamiento en la TTP medido en número de claves
Variables de estado	
CntConectados_PUB	Número actual de entidades conectadas para publicar
CntConectados_FTP	Número de entidades conectadas por *ftp*
CapacOcupada	Capacidad de almacenamiento ocupada
Variables de salida	
N_MsjPUB	Número de mensajes cuya clave fue publicada
NReintPUB	Número de reintentos de publicación de la clave causados por capacidad de conexión
NDemPUB_Alm	Número de reintentos de publicación de la clave causado por capacidad de almacenamiento en la TTP
NReint_O_Con	Número de reintentos de *Con* por *O*
NReint_R_Con	Número de reintentos de *Con* por *R*
N_Exitosos_O_Con	Número total de exitosas solicitudes de Con por *O*
N_NoExitosos_O_Con	Número total de no exitosas solicitudes de *Con* por *O*

N_Exitosos_R_Con	Número total de exitosas solicitudes de *Con* por *R*
N_NoExitosos_R_Con	Número total de no exitosas solicitudes de *Con* por *R*

Anexo 2: Eventos del sistema. Se omiten algunos eventos de *R* cuya implementación es similares a los de *O*

EVENTO 1: Generación de mensajes (*O*: origen) Generar el mensaje en el tiempo *t=S.TAct* Incrementar *O.NMsj, M.TiempoCreacion = S.Tact, M.Estado = St1* For i = 1 to *S.Receptores* do *Adiciono el evento* ***Llegada de mensajes a R (O,M,R$_i$)*** *t=S.TAct + S.DComunicacionOR(O,Ri)* Adiciono *M* a la lista *O.LMsj* Adiciono el evento ***generación de mensaje(O)*** *t=S.TAct + S.DGenMsj(O)*
EVENTO 2: Llegada de mensajes a *R* (*O*: origen , *M*: mensaje, *R*: receptor) Incremento el número de mensajes recibidos *R.N_MsjRecibidos* Adiciono el evento ***Llegada de EOR a O (M,R)*** *t=S.TAct + S.DComunicacionOR(O,R) + S.DEnvEOR*
EVENTO 3: Llegada de *EOR* a *O* (*M*: Mensaje, *R*: receptor) Incremento *M.N_EOR* Si *M.N_EOR = S.Receptores* Actualizo *M.TEspEOR= S.TAct - M.TiempoCreacion* Adiciono el evento ***Llegada de solicitud de publicación de clave a la TTP (O, M, TTP)*** en el tiempo *t=S.TAct + S.DComunicacionOTTP(O)*
EVENTO 4: Llegada de solicitud de publicación de clave a la TTP (*O*: Origen, *M*: mensaje, *TTP*: tercera parte confiable) Si *TTP.CntConectados_PUB + 1 > TTP.CapConexion_PUB* Incremento *TTP.NReintPUB* ***Reintento de solicitud de publicación de clave (O,M)*** en el tiempo *t = S.TAct + O.TReintPUB* Sino Si *TTP.CapacOcupada + 1 >TTP. CapAlmacenamiento* Incremento *TTP.NReintPUB_Alm* Adiciono el evento ***Reintento de solicitud de publicación de clave (O,M)*** en el tiempo *t = S.TAct + O.TReintPUB* Sino Incremento *TTP.CntConectados_PUB* Adiciono el evento ***Desconexión ftp de O (O,M, TTP)*** en el tiempo *t = S.TAct + S.DConexionPUB*
EVENTO 5: Reintento de solicitud de publicación de clave(*O*: originator, *M*: mensaje) Adicionar el evento ***Llegada de solicitud de publicación de clave a la TTP (O, M)*** en el tiempo *t = S.TAct + S.DComunicacionOTTP(O)*
EVENTO 6: Desconexión ftp de *O* (*O*: origen, *M*: mensaje, *TTP*: tercera parte confiable) Actualizar *M.TDemPUB=S.TAct - M.TiempoCreacion* Incrementar *O.N_MsjPUB* Incrementar *TTP.N_MsjPUB* Incrementar *TTP.CapacOcupada* Decrementar *TTP.CntConectados_PUB* Cambiar el estado del mensaje *M.Estado=St4* Adicionar el evento ***Solicitud de evidencia Con por O (M)*** en el tiempo *t = S.TAct* Adicionar el evento ***Solicitud de evidencia Con por R (M)*** en el tiempo *t=S.TAct* Adicionar el evento ***Borrado de la clave en la TTP (TTP,M)*** en el tiempo *t = S.TAct + TTP. TiempoBD_k*
EVENTO 7: Solicitud de evidencia *Con* por *O* (*M*: mensaje) Actualizar *M.TInic_O_Con = S.TAct* Adicionar el evento ***Conexión de O para solicitud de evidencia Con, (O,M,TTP)*** en el tiempo *t=S.TAct +* valor aleatorio generado por *S.DComunicacionOTTP(O)*
EVENTO 9: Conexión de *O* para solicitud de evidencia *Con* (*O*: origen, *M*: mensaje, *TTP*: tercera parte confiable*)* Si *TTP.CntConectados_FTP + 1 > TTP. CapConexion_FTP* Incrementar *TTP.NReint_O_Con* Incrementar *M.NReint_O_Con* Adicionar el evento ***Reintento de O de solicitud de evidencia Con (M)*** en el tiempo *t = S.TAct + O.TReintFTP* Sino Incrementar *TTP.CntConectados_FTP* Adicionar el evento ***Desconexión FTP de O (O,M,TTP)*** en el tiempo *t = S.TAct + S.DConexionFTP*
EVENTO 13: Borrado de la clave en la TTP (*M*: mensaje) Cambio el estado del mensaje *M.Estado=St5*

Decremento *TTP.CapacOcupada*

EVENTO 14: Desconexión FTP de *O* (*O*: origen, *M*: mensaje, *TTP*: Tercera parte confiable**)**

Si *M* está en la lista *TTP.LMsj_PUB* y *M.Estado=St4*

Incremento *TTP.N_Exitosos_O_Con*

Sino

Incremento *TTP.N_NoExitosos_O_Con*

Actualizo *M.TDem_O_Con= S.CurrenTime - O M.TInic_O_Con*

Decremento *FTP.CntConectados_FTP*

Si *M.Estado_O_Con* = true

Incremento *O.N_NoExitosos_Con*

sino

Incremento *O.N_Exitosos_Con*