Verificación Formal de la Equidad de un Protocolo de Firma de Contratos mediante Colored Petri Nets

M. Magdalena Payeras-Capellà, Macià Mut-Puigserver, Andreu P. Isern-Deyà, Josep L. Ferrer-Gomila, Llorenç Huguet-Rotger

Universitat de les Illes Balears, Spain
{mpayeras, macia.mut, andreupere.isern, jlferrer, lhuguet}@uib.es

Resumen. Un protocolo de firma de contratos es un protocolo de intercambio equitativo de valores en el que las partes intercambian sus firmas sobre el contrato. Existen diversos protocolos de firma de contratos, usualmente acompañados de una verificación informal. En este artículo utilizamos un modelo basado en Colored Petri Nets para verificar formalmente la equidad de un protocolo de firma de contratos existente. En el análisis automático hemos modelado el protocolo y los roles de los firmantes, la tercera parte de confianza, firmantes maliciosos así como el rol de un usuario intruso. El análisis ha permitido detectar tres casos en los que el protocolo genera evidencias contradictorias y, finalmente, se ha definido el comportamiento de un árbitro que permite resolver el intercambio de forma equitativa incluso ante la presencia de evidencias contradictorias.

Palabras Clave: Protocolo de firma de contratos, Verificación formal, Petri Nets

1. Introducción

La firma de contratos es un caso particular de un protocolo de intercambio equitativo, en el que dos o más firmantes se comprometen a firmar un contrato previamente acordado, con la condición que a su finalización, todos los firmantes tengan el contrato firmado o ninguno de ellos lo tenga. Por tanto, un protocolo equitativo siempre debe proporcionar un trato igualitario a todos los participantes. Además, debe generar evidencias que prueben el comportamiento de los usuarios, para que, en caso de disputa, un árbitro externo pueda evaluar las evidencias y tomar una decisión sin ambigüedades.

Los protocolos de firma de contratos normalmente hacen uso de una Tercera Parte de Confianza (TTP) que ayuda a los usuarios a finalizar satisfactoriamente el intercambio. La TTP puede involucrarse en diferente medida en la ejecución del protocolo, actuando en cada paso, o bien solo interviniendo cuando los usuarios lo requieran. Usando este último tipo de TTP, se han definido varios protocolos eficientes y optimistas [1] que requieren solo tres mensajes para finalizar su ejecución. El protocolo de Micali [2] y el protocolo FPH [3] cumplen estas dos propiedades.

Con el objetivo de verificar formalmente un protocolo de intercambio equitativo, se han ido utilizando técnicas manuales, complicadas de aplicar y que deben ser adaptadas en cada caso, como el uso de Strand Spaces [4]. Recientemente, Sornkhom y Permpoontanalarp [6] han aplicado un método automatizado para analizar la seguridad del protocolo de Micali mediante el uso de Coloured Petri Nets (CPN) [7]. El método permite la demostración de vulnerabilidades del protocolo de Micali descubiertas anteriormente, además de detectar dos nuevos ataques al protocolo de Micali.

En este artículo vamos a aplicar un nuevo modelo de verificación formal y automatizado de protocolos de firma de contratos, basado en [6], que permitirá analizar con mayor rapidez y generalización las propiedades de los protocolos de intercambio equitativo. Como primer paso, mediante la aplicación de dicho modelo vamos a verificar formalmente la equidad del protocolo FPH.

El artículo se organiza de la siguiente forma: la sección 2 resume el protocolo FPH y sus características de seguridad. La sección 3 incluye la descripción del modelo de simulación utilizando CPNs. La sección 4 presenta el análisis de la equidad del protocolo y, finalmente, la sección 5 incluye las conclusiones y describe algunas líneas de trabajo futuro.

2. Protocolo de Firma de Contratos FPH

2.1. Propiedades Ideales del Protocolo de Firma de Contratos

Las soluciones prácticas para firma de contratos requieren de la existencia y la posible participación de una TTP. Para que el protocolo sea eficiente, generalmente se desean conseguir tres objetivos:

1. Reducir la participación de la TTP.
2. Reducir el número de mensajes intercambiados.
3. Minimizar la necesidad de operaciones costosas y de almacenamiento de gran volumen de información cuando la TTP deba actuar.

El primer objetivo es alcanzado en varias soluciones optimistas [1, 8, 9, 10], en donde la TTP solo actúa en caso de conflicto entre las partes para garantizar la equidad del protocolo. En referencia al número de mensajes intercambiados entre dos partes se establece que el mínimo son tres para las propuestas optimistas [1].

Además de los anteriores objetivos, un protocolo tiene que proporcionar evidencias a las partes implicadas que prueben, a la finalización del intercambio, si el contrato ha resultado firmado. Otras propiedades adicionales que deberían ser cumplidas por protocolos optimistas son [9, 10]:

- **Efectividad**. Si las partes implicadas actúan correctamente obtendrán los elementos deseados sin la participación de la TTP.
- **Equidad**. Ningún participante debe obtener una situación de ventaja en ninguna de las etapas de la ejecución del protocolo.

- **Asincronía.** Los participantes pueden decidir cuando finalizar la ejecución del protocolo, sin límites de tiempo.
- **No repudio**. Los participantes no pueden negar sus acciones.
- **Verificabilidad de la TTP**. Si la TTP actúa incorrectamente, todas las partes afectadas deben ser capaces de demostrarlo.

2.2 Descripción del Protocolo de Firma de Contratos FPH

El protocolo asume que Alice y Bob se han puesto de acuerdo en el texto del contrato *C* antes del inicio del intercambio. A continuación firman dicho contrato usando el protocolo. El canal utilizado para el envío de información entre los firmantes es un canal inseguro, por lo que no puede asumirse que los mensajes enviados a través de este canal llegan a su destinatario. El canal entre los firmantes y la TTP es un canal elástico, es decir, los mensajes llegaran en algún momento a su destinatario aunque el instante de llegada no puede ser predecido.

El iniciador, *A*(lice), y el destinatario, *B*(ob), intercambiaran las evidencias de no repudio directamente. Solo en el caso de que no puedan obtener el elemento esperado de la otra parte los firmantes contactaran con la *TTP* (*T*), iniciando los subprotocolos *cancel* o *finish*. La notación y los elementos utilizados en la descripción del protocolo se resumen en la tabla siguiente:

NOTACIÓN	
X, Y	concatenación de dos mensajes *X* e *Y*
H(X)	Función de hash unidireccional resistente a colisiones del mensaje *X*
$\mathbf{S_i(X)}$	Firma digital del mensaje *X con la clave privada del usuario i* (utilizando una función de hash, *H*(), para crear el resumen de *X*)
i → j: X	El usuario *i* envía el mensaje *X* al usuario *j*

ELEMENTOS	
M	mensaje que contiene el contrato a firmar; el contrato especifica la identidad del iniciador, *A*(lice), y la del destinatario, *B*(ob)
$\mathbf{h_A = S_A(M)}$	Firma de *A* sobre el contrato *M*
$\mathbf{h_B = S_B(M)}$	Firma de *B* sobre el contrato *M*
$\mathbf{ACK_A = S_A(h_B)}$	Firma de *A* sobre h_B; reconocimiento de que *A* sabe que el contrato ha sido firmado. Parte de la evidencia de no repudio (NR) para *B*
$\mathbf{ACK_T = S_T(h_B)}$	Firma de la *TTP* sobre h_B; reconocimiento equivalente al que *A* debería haber enviado
$\mathbf{h_{AT} = S_A[H(M), h_A]}$	evidencia de que *A* ha solicitado la intervención de la *TTP*
$\mathbf{h_{BT} = S_B[H(M), h_A, h_B]}$	evidencia de que *B* ha solicitado la intervención de la *TTP*

El subprotocolo de *intercambio* es el siguiente:

1. A → B:	M, h_A
2. B → A:	h_B
3. A → B:	ACK_A

Si se completa la ejecución del protocolo, el iniciador *A* poseerá la evidencia de no repudio (NR), h_B, y el receptor *B* tendrá la evidencia de no repudio, h_A y ACK_A, por lo que el protocolo es efectivo. Si no es el caso, *A* o *B*, o ambos, deberán resolver la situación no equitativa iniciando los subprotocolos *cancel* o *finish*, respectivamente. Si *A* afirma que no ha recibido el mensaje 2 de *B*, *A* puede iniciar el subprotocolo de cancelación siguiente:

	1'. A → T:	$H(M), h_A, h_{AT}$
IF (*finished = verdadero*)	2'. T:	recupera h_B
	3'. T → A:	h_B, ACK_T
ELSE	2''. T → A:	S_T("*cancelled*", h_A)
	3''. T:	almacena *cancelled = verdadero*

La *TTP* verificará la corrección de la información proporcionada por *A*. Si es errónea la *TTP* enviará un mensaje de error a *A*. Si es correcta, procederá de una de dos formas posibles. Si la variable *finished* es *verdadero*, significa que *B* había contactado previamente con la *TTP* (ver párrafo siguiente). La *TTP* ha proporcionado evidencia de NR a *B*, ACK_T, y ahora debe proporcionar la evidencia correspondiente a *A*. La TTP recupera el elemento almacenado, ACK_T, y lo envía a *A*, junto con un elemento que pruebe su implicación en la resolución de la disputa, h_B'. Si *B* no ha contactado con la *TTP* anteriormente, la *TTP* enviará un mensaje a *A* para cancelar el intercambio, y almacenará esta información (*cancelled = verdadero*) para satisfacer futuras peticiones de *B*. En cualquier caso, la situación final es equitativa.

Si *B* afirma que no ha recibido el mensaje 3, *B* puede iniciar el siguiente subprotocolo *finish*:

	2'. B → T:	$H(M), h_A, h_B, h_{BT}$
IF (*cancelled = verdad*)	3'. T → B:	S_T("*cancelled*", h_B)
ELSE	3''. T → B:	ACK_T
	4''. T:	almacena *finished = verdad.* y h_B

La TTP también verificará la veracidad de la información proporcionada por *B*. Si ésta no es correcta la TTP enviará un mensaje de error a B. Si, en cambio, es correcta,

la TTP procederá de una de dos formas posibles. Si la variable *cancelled* es *verdadero*, significa que *A* ha contactado previamente con la TTP (ver párrafo anterior). La TTP ha enviado el mensaje a *A* para cancelar la transacción, y ahora debe enviar un mensaje similar a *B*. Si *A* no ha contactado con la TTP anteriormente, la TTP enviará el elemento de NR, ACK_T, a *B*. En este caso la *TTP* almacenará la evidencia NR, h_B, y asignará el valor *verdadero* a la variable *finished*, para satisfacer peticiones futuras de *A*. de nuevo, en cualquier circunstancia, el intercambio se encuentra en una situación equitativa.

Como conclusión, el protocolo es equitativo y no se han utilizado límites temporales, por lo que el protocolo es asíncrono.

2.3 Análisis Informal de la Equidad y el No Repudio en el Protocolo de Firma de Contratos FPH

Después de una ejecución completa del protocolo (con o sin la intervención de la TTP), pueden aparecer disputas entre los firmantes. Nos podemos encontrar con dos tipos de disputas: el repudio de *A* (*B* afirma que el contrato está firmado) y el repudio de *B* (*A* afirma que el contrato está firmado).

Un árbitro externo (que no forma parte del protocolo) tiene que evaluar las evidencias proporcionadas por las partes para resolver estos dos tipos de disputas. Como resultado, el árbitro determinará quien dice la verdad. El árbitro tiene que saber quien actúa como originador (información que aparece en el contrato).

En caso de repudio de *A*, *B* afirma que ha recibido la firma sobre el contrato *M* de *A* y debe proporcionar la siguiente información al árbitro: *M*, h_A y ACK_A o ACK_T. El árbitro comprobará si h_A es la firma de *A* sobre *M*, y si es correcta el árbitro asumirá que *A* ha enviado su firma a *B*. Entonces el árbitro comprobará si ACK_A es la firma de *A* sobre h_B, o comprobará si ACK_T es la firma de la TTP sobre h_B. Si se supera la verificación, el árbitro asumirá que *A* o la TTP había enviado un reconocimiento a *B*. Entonces el árbitro dará la razón a *B*. Si no es así, y una o ambas de las comprobaciones previas falla, el árbitro rechazará la demanda de *B*. Si la evidencia aportada por *B* prueba que tiene la razón, y *A* presenta el mensaje $PR_T[H(\text{“cancelled”}, h_A)]$, entonces la TTP o *A* actuaron de forma incorrecta.

En caso de repudio de *B*, *A* afirma que *B* ha firmado el contrato *M*. Debe proporcionar la siguiente información al árbitro: *M* y h_B. El árbitro comprobará si h_B es la firma de *B* sobre *M*, y si es correcta el árbitro asumirá que *B* ha recibido *M* y h_A, y que se ha comprometido a obtener el reconocimiento, ACK_A o ACK_T. Si las verificaciones previas fallan, el árbitro rechazará la demanda de *A*. Si la verificación es positiva, el árbitro deberá interrogar a *B*. Si *B* proporciona un mensaje *cancel*, significa que *B* ha contactado con la TTP, y que la TTP observó que *A* ha ejecutado el subprotocolo *cancel*. Por esta razón la TTP envía un mensaje *cancel* a *B*. Ahora se ha demostrado que *A* ha intentado engañar. Por tanto, el árbitro rechazará la demanda de *A*, y el árbitro dará la razón a *B*. Si *B* no puede proporcionar el mensaje *cancel*, el árbitro dará la razón a *A*. Como conclusión, el protocolo satisface los requisitos de equidad y no repudio.

Este análisis informal no cubre todas las situaciones derivadas de la ejecución del protocolo. Puede ser completado mediante una verificación formal del protocolo

(incluída en la sección 4) utilizando el modelo basado en Petri Nets descrito en la sección 3.

3. Descripción del Modelo Usado para la Verificación Formal de Protocolos de Intercambio Equitativo

3.1 Colored Petri Nets

CPN (Coloured Petri Nets) [7] es un lenguaje de modelado que combina las características y propiedades de las Petri Nets ordinarias con un lenguaje de programación de alto nivel llamado ML (Modeling Language) [7]. Las Petri Nets aportan las primitivas para la interacción de procesos, mientras que por su parte, el lenguaje de programación proporciona la definición de los tipos de datos y la posibilidad de manipularlos. CPN puede ser usado como un método formal de análisis de sistemas distribuidos y protocolos de comunicaciones. Una red CPN es un modelo ejecutable que representa los estados de un sistema a lo largo de una línea temporal. Una CPN contiene cuatro tipos de componentes:

- **Estados.** Representan el estado del sistema en un instante determinado. Los estados cambian a partir de la activación de las transiciones.
- **Transiciones.** Se corresponden con una acción que hace cambiar el estado del sistema.
- **Arcos.** Son los enlaces entre los estados y las transiciones.
- **Color sets.** Son testimonios que van pasando a través de los estados y las transiciones, y que tienen un determinado valor, llamado color.

El estado global del sistema modelado, para cada instante después de la generación de un evento, determina lo que se llama *marking*. Un *marking*, por tanto, es equivalente a una fotografía del estado del sistema tras cada evento que hace cambiar el mismo. Una de las herramientas que implementan CPN es CPNTools [7], la cual es usada en el desarrollo del presente trabajo. Se trata de una herramienta que permite construir CPN de forma rápida y visual. Una vez el modelo está diseñado, se puede someter a simulación, proceso mediante el cual se genera el *state space*. *State spaces* es el conjunto de *markings* del sistema entre el evento inicial y el final. Por tanto, mediante este procedimiento se obtiene una definición completa del comportamiento del sistema a lo largo de su ejecución.

3.2 Suposiciones Iniciales y Metodología

Para usar CPN con el objetivo de modelar un protocolo, hay que definir una serie de consideraciones generales: cada participante posee un identificador único, cada participante conoce, como paso previo a la ejecución del protocolo, las claves públicas del resto de partes firmantes, los algoritmos criptográficos usados son seguros, los mensajes enviados entre la TTP y cualquier participante siempre son entregados al destino sin modificaciones (canal elástico).

La metodología a seguir para construir el modelo y analizar la equidad del protocolo es la siguiente:

1. **Construir el modelo**: Declarar los *color sets* para representar los mensajes y los elementos del protocolo. Crear la red de alto nivel para modelar el conjunto del protocolo incluyendo los participantes. Crear la red a nivel de entidad para modelar el comportamiento interno de cada participante. Crear el nivel de proceso para cada nivel de entidad. Declarar funciones y variables que serán usadas en el modelo.
2. **Generar el *state space***: definir el *marking* inicial para cada participante. Generar el *state space* del modelo usando CPNTools.
3. **Crear las funciones** para buscar estados de ataque.
4. **Extraer escenarios de ataque** mediante la evaluación de los *markings* obtenidos.

3.3 Descripción del Modelo

El modelo, basado en lo planteado por Sornkhom y Perpoontanalarp [10], define cuatro participantes: A(lice), B(ob), I(ntruso) y TTP. Mientras la TTP es estrictamente honesta, las otras partes pueden seguir un rol malicioso. *A* y *B*, mediante sus roles maliciosos (A_m y B_m, respectivamente), pueden parar el intercambio o contactar con la TTP en diferentes etapas del protocolo, distintas a las definidas por el mismo, para intentar engañar a la otra parte. *I* es un participante malicioso que puede actuar como observador, capturando y almacenando los mensajes en tránsito, pero además puede realizar otras tareas: eliminar, reenviar o modificar mensajes en tránsito transmitidos por cualquier participante.

Para modelar los eventos de parada del intercambio realizados por cualquier participante malicioso (esto es, A_m, B_m o *I*), el modelo define un mecanismo para informar sobre estos eventos a la otra parte involucrada. Cuando ocurre un evento de este tipo, inmediatamente se envía un mensaje desde el participante que realiza la eliminación del mensaje o la parada del intercambio a las otras partes implicadas. Con este procedimiento, se evita tener que usar nociones de tiempo (*timeout*) en el modelo. Cuando un mensaje de evento es recibido, el participante puede contactar con la TTP o bien parar el intercambio.

Otra consideración importante atañe al tipo de canal de comunicaciones usado para el intercambio de mensajes entre cualquier participante y la TTP, el cual por definición es elástico (ver 2.1).

Con los datos proporcionados, creamos un escenario para ser usado en el modelado de un protocolo utilizando diferentes sesiones de ataque, donde cada una de ellas puede involucrar un iniciador (*A* o A_m) y un receptor (*B* y B_m). Notar que *I* y la TTP están presentes implícitamente en cada sesión. Por tanto, se pueden generar cuatro sesiones distintas combinando los roles de los participantes: (*A*, *B*), (A_m, *B*),(*A*, B_m) y (A_m, B_m) donde (X, Y) se corresponde con el participante que es el iniciador (X) y el receptor (Y), respectivamente. En este artículo no se consideran sesiones en paralelo, donde las partes maliciosas pueden participar en múltiples sesiones concurrentes, dejando esta tarea para próximos trabajos.

La arquitectura del modelo está claramente definida en tres grandes bloques o niveles: alto nivel, nivel de entidad y nivel de proceso. Todos los mensajes enviados

por cualquier participante están compuestos por los identificadores de la fuente y el destino, así como el mensaje de protocolo incluido como *payload*.

El esquema del nivel alto (Figura 1) enseña la interacción entre todos los participantes involucrados en el protocolo, así como el flujo de mensajes entre los mismos. En el nivel alto se pueden consultar los contenidos de la base de datos de cada participante, las cuales contienen los mensajes de protocolo intercambiados. Finalmente, se puede observar y controlar el contenido de la sesión. La variable controla el rol de los participantes y cuáles son utilizados en la ejecución del protocolo (rol honesto o malicioso). Además, en la Figura 1 se puede observar como todos los mensajes son interceptados por *I* en su tránsito entre los participantes. La notación que aparece en la figura es la notación propia de las Colored Petri Nets, i aparece descrita en la documentación de las mismas.

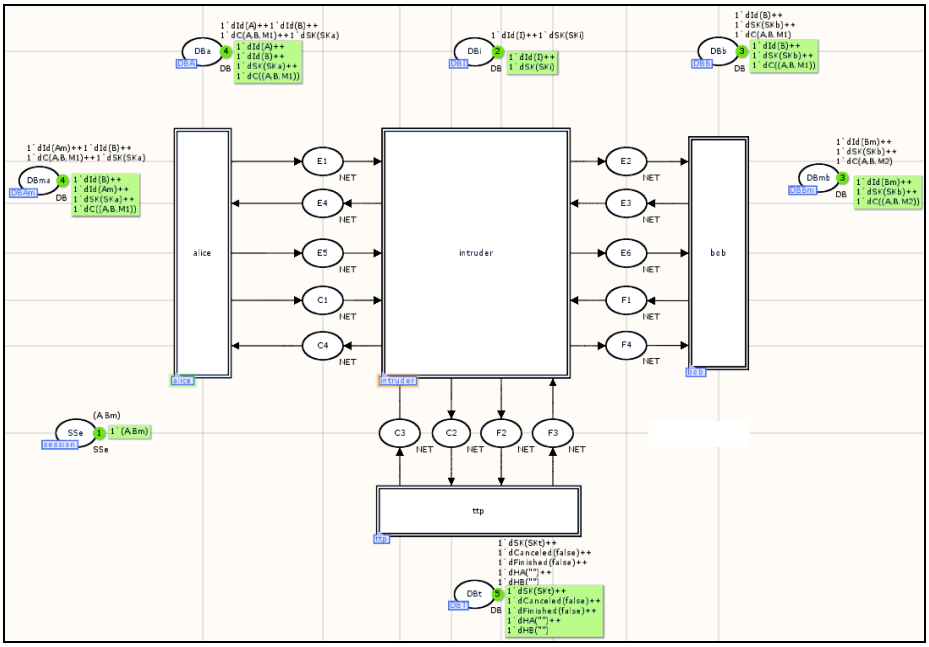

Figura 1. Esquema de Alto nivel.

El nivel de entidad muestra con mayor detalle el flujo de trabajo de los participantes. En la Figura 2(a) se puede ver el nivel de entidad de *A* y sus dos roles. Las transiciones TA_1 a TA_4 se corresponden a su rol honesto, mientras que las transiciones TAm_1 a TAm_4 pertenecen al rol malicioso. La primera transición de *A*, TA_1 y TAm_1, genera el primer mensaje del protocolo y lo envía a *B*. Las transiciones TA_2 y TAm_2 reciben y verifican el segundo mensaje enviado por *B* y le retornan el tercer mensaje. TA_3 y TAm_3 tienen la responsabilidad de contactar con la TTP usando el subprotocolo de cancelación, y las ultimas transiciones, TA_4 y TAm_4 reciben la respuesta desde la TTP. Notar que la selección de las transiciones que tienen que ser ejecutadas se lleva a cabo a través de la configuración de la sesión.

El nivel de entidad de *B*, como se puede ver en la Figura 2(b), como en el participante *A*, implementa el rol honesto (TB_1 a TB_3) y el malicioso (TBm_1 a TBm_3). TB_1 y TBm_1 reciben y verifican el primer mensaje del protocolo y además envían el segundo mensaje. Por su parte, TB_2 y TBm_2 reciben y verifican el tercer mensaje del protocolo, y si es necesario, implementan la lógica para contactar con la TTP. Finalmente, TB_3 y TBm_3 son las transiciones responsables de recibir la respuesta desde la TTP.

El nivel de proceso implementa todas las acciones que ejecutan los participantes y especifica la forma en la que las entidades se relacionan. Las acciones ejecutadas por cada proceso son atómicas, esto es, solo un proceso puede ser ejecutado al mismo tiempo y debe terminar antes que el siguiente empiece su ejecución. Esto puede modelarse por un *token*. Este es único y compartido por todos los participantes. Es capturado por cada parte cuando un proceso empieza su ejecución y es liberado cuando el mismo proceso termina. Además, cada proceso está controlado por un mecanismo de control de flujo, mediante el cual un *token* es transferido a través de los participantes para cada paso del protocolo. Este *token* controla el orden en el que deben ser ejecutadas las acciones. Por ejemplo, el mecanismo controla que la generación de un mensaje se realice después de haberse ejecutado el proceso de verificación del mensaje previamente recibido.

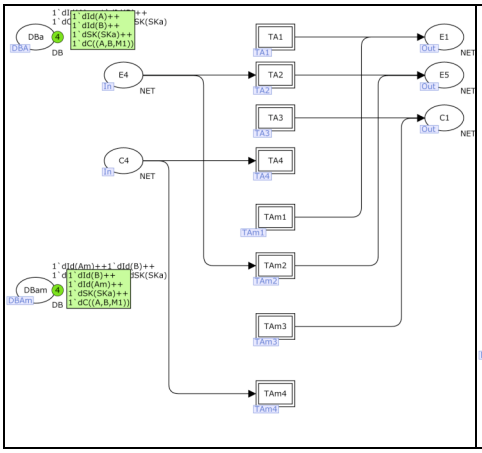

Fig. 2a. Nivel de entidad de *A*

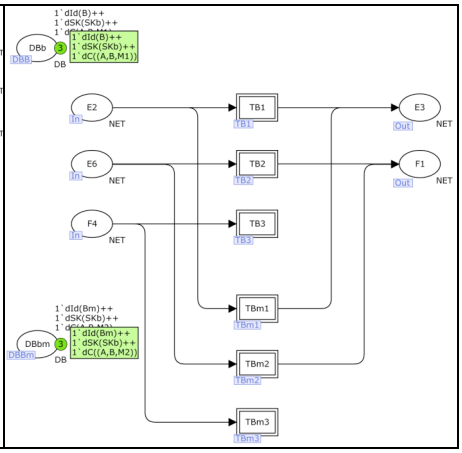

Fig. 2b. Nivel de entidad de *B*

3.4 Query functions

Para detectar y extraer escenarios de ataque a partir del *state space* generado por la utilidad, utilizamos un serie de funciones de búsqueda (Figura 3) para encontrar contenidos especiales en cada base de datos de los participantes. La función principal es *SearchCommitsTerminalNodes(ack,id),* donde *ack* es el elemento o compromiso que se quiere buscar en la base de datos del participante identificador por *id*. La función está construida usando una función estándar de la utilidad CPN-Tools, llamada *PredNodes(p1,p2,p3)*. El primer parámetro de esta función es otra función anidada llamada *SearchCommits(ack,id),* donde *ack* y *id* tienen el mismo uso que en

la función previa. Esta función es capaz de examinar los contenidos de la base de datos de *id* buscando el elemento *ack*. El segundo parámetro es usado para elegir solo los nodos que sean terminales, es decir, nodos hoja del árbol generado, los cuales contienen una ejecución completa del protocolo. El último parámetro, *NoLimit*, indica a la función que debe de examinar todos los nodos y devolver todos los resultados posibles.

La función principal es usada para analizar la equidad del protocolo, aplicándola contra las bases de datos de los participantes del intercambio. La función devuelve una lista de nodos terminales, que contienen el compromiso buscado. El análisis de esta lista permitirá determinar si el intercambio es o no equitativo.

```
fun SearchCommits( ack:DB, id:Id ) : Node list
  = PredAllNodes(
      fn n =>
      let
          val dba = Mark.Top'DBa 1 n
          val dbb = Mark.Top'DBb 1 n
          val dbi = Mark.Top'DBi 1 n
          val dbam = Mark.Top'DBma 1 n
          val dbbm = Mark.Top'DBmb 1 n
      in
          if (id=A) then
             cf( ack , dba ) > 0
          else if (id=B) then
             cf( ack , dbb ) > 0
          else if (id=Am) then
             cf( ack, dbam ) > 0
          else if (id=Bm) then
             cf( ack, dbbm ) > 0
          else (* id=I *)
             cf( ack , dbi ) > 0
      end
  )
```

```
fun SearchCommitsTerminalNodes( ack:DB, id:Id ) : Node list
   = PredNodes ( (SearchCommits(ack,id)) ,
                 fn n => (Terminal n) andalso (FullyProcessed n),
                 NoLimit)
```

Figura 3. Funciones de búsqueda

4 Análisis Automático de la Equidad del Protocolo de Firma de Contratos FPH.

A continuación se describen algunas situaciones conflictivas del protocolo FPH en las que los firmantes disponen de evidencias contradictorias. Las evidencias generadas por este protocolo no son transferibles, y un árbitro debe contactar con ambos firmantes para resolver una disputa, conocer el estado final del intercambio y garantizar el no repudio. Esta propiedad ha sido descrita en [8] y recibe el nombre de *weak fairness*. En este análisis formal de la equidad del protocolo probaremos que el árbitro puede resolver todo tipo de situaciones conflictivas derivadas de la ejecución del protocolo.

En [4] se describió, juntoal protocolo FPH, una situación conflictiva en la que *A* podía obtener evidencia *NR* de *B* (h_B) y un mensaje *cancel* de *T*, mientras *B* obtenía la evidencia *NR* de *A* (h_A, ACK_A). *A* puede conseguirlo, por ejemplo, invocando el subprotocolo *cancel* después de la finalización del subprotocolo de *intercambio*. Parece que *A* puede afirmar que el contrato está firmado (*signed*) o no firmado (*cancelled*), dependiendo de su utilidad, mientras que *B* posee la evidencia NR que probará que el contrato está firmado (*signed*). Hemos detectado esta situación en el análisis automático y la hemos llamado *caso 1*.

Además, gracias al modelo utilizado, hemos descubierto dos situaciones conflictivas más. La primera (a la que llamaremos *caso 2*) se produce cuando un *A* malicioso invoca el subprotocolo *cancel* después de la finalización del subprotocolo de *intercambio* (como en el *caso 1*) y después un *B* malicioso lo ejecuta también. Parece que tanto *A* como *B* pueden afirmar que el contrato está firmado (*signed*) o no firmado (*cancelled*), dependiendo de su voluntad.

La última situación conflictiva que hemos detectado (*caso 3*) se alcanza cuando el intercambio se interrumpe después del segundo paso. En este caso *A* dispone de la evidencia de NR de *B* mientras *B* no dispone de la evidencia NR de *A*. Ambas partes pueden contactar con la TTP. Si *B* contacta en primer lugar, la TTP le enviará la evidencia NR y el contrato estará firmado. Por otra parte, si *A* contacta en primer lugar, la TTP cancelará el intercambio y entonces *A* tendría la evidencia *NR* de *B* (h_B) y un mensaje de cancelación (*cancel*) de *T*, mientras *B* obtendría el mensaje de cancelación (*cancel*).

Las tres situaciones conflictivas se pueden detectar utilizando el modelo que recrea una sesión formada por un iniciador malicioso o bien dos usuarios maliciosos actuando como iniciador y receptor (*Am*, *Bm*). De esta forma, todos los posibles comportamientos fraudulentos de ambas partes pueden ser contemplados. Usando las funciones (*query functions*) definidas sobre el modelo, como puede observarse en la figura 4, se ha realizado una búsqueda en las bases de datos de cada usuario para localizar las evidencias almacenadas. En este caso, se han buscado los mensajes segundo y tercero del subprotocolo de intercambio así como todas las respuestas recibidas de la *TTP*.

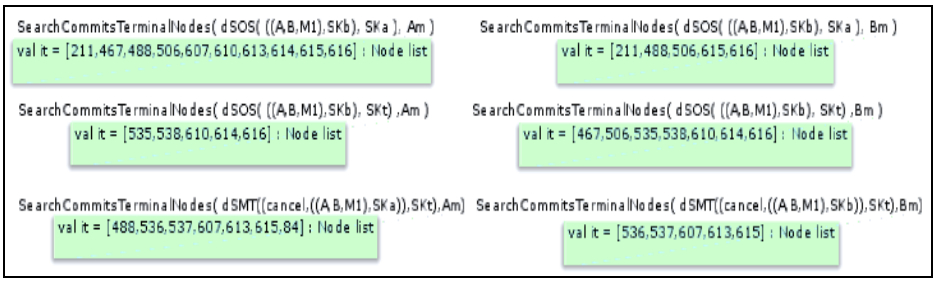

Figura 4. Resultados de la aplicación de las funciones de búsqueda en la sesión (A_m, B_m).

Si analizamos la lista de nodos obtenidos en cada ejecución de la función, podemos construir la tabla siguiente con las tres situaciones previamente descritas. Por cada caso detectado, señalamos el estado del contrato (firmado o cancelado) así como los contactos de las partes, honestamente o de forma maliciosa, (A_m o B_m), con la *TTP*.

Caso	Nodo	A_m tiene NR	B_m tiene NR	A_m contacta con la *TTP*	B_m contacta con la *TTP*
1	488	Firmado Cancelado	Firmado	Si (malicioso)	No
2	615	Firmado Cancelado	Firmado Cancelado	Si (malicioso)	Si (malicioso)
3	607	Firmado Cancelado	Cancelado	Si (malicioso)	Si (honesto)
3	613	Firmado Cancelado	Cancelado	Si (malicioso)	Si (honesto)

Tabla 1. Lista de nodos correspondientes a las tres situaciones con evidencias contradictorias.

Como se puede observar en la tabla 2, la utilización del modelo basado en Petri Nets permite localizar las situaciones relativas a los tres casos conflictivos, en los que A_m y B_m disponen de evidencias contradictorias. Se han localizado cuatro posibles escenarios dado que el *caso 3* puede aparecer en dos ocasiones.

Podemos observar que el *caso 1* se produce en el nodo 488, cuando A_m obtiene la evidencia NR de B_m (entonces *A* dispone de la evidencia que demuestra que el contrato esta firmado (*signed*). Sin embargo *A* contacta con la *TTP* con el objetivo de cancelar el intercambio. Este es un comportamiento malicioso ya que A_m no debería contactar con la TTP para cancelar un intercambio que ya ha sido finalizado. La *TTP* envía a A_m el mensaje *cancel* y a partir de este momento A_m puede afirmar que el contrato está firmado (*signed*) o no firmado (*cancelled*). B_m recibe la evidencia NR de A_m y no necesita contactar con la TTP.

En el *caso 2*, correspondiente al nodo 615, el subprotocolo de intercambio finaliza satisfactoriamente para ambas partes, pero A_m contacta con la TTP, después de transferir la evidencia NR a B_m, para obtener el mensaje *cancel*. En el momento en el que B_m recibe la evidencia NR de A_m también contactará con la TTP y obtendrá un mensaje *cancel*. Por tanto, A_m y B_m se comportan incorrectamente ya que han contactado con la TTP en situaciones en las que no deberían hacerlo.

El *caso 3* se detecta en dos ocasiones durante la simulación del protocolo. En ambos escenarios A_m obtiene la evidencia NR de B_m pero B_m nunca recibe el tercer mensaje del subprotocolo de intercambio. En cada escenario, A_m ejecuta maliciosamente el subprotocolo de cancelación y como resultado recibe un mensaje *cancel* de la *TTP*. En el primer escenario, correspondiente al nodo 607, A_m decide, maliciosamente, detener el intercambio y por ello no envía el tercer mensaje a B_m. Por otra parte, el nodo 613 es el resultado de un evento de tipo *drop* sobre el tercer mensaje realizado por un usuario intruso, *I*. Desde el punto de vista de B_m, ambos escenarios son el mismo caso y por tanto contactará con la TTP para resolver la situación obteniendo un mensaje *cancel* para cada escenario.

Además de los casos conflictivos descritos anteriormente, existen otras situaciones, también detectadas por el modelo, en las que no se generan evidencias contradictorias, aunque las partes pueden disponer de pruebas repetidas al haber contactado con la TTP una vez finalizado el intercambio. La siguiente tabla muestra los escenarios en los que las pruebas generadas son complementarias pero no contradictorias:

Nodo	A_m tiene NR	B_m tiene NR	A_m contacta TTP	B_m contacta TTP
84	Cancelado	Ninguna	Si (honesto)	No
211	Firmado	Firmado	No	No
467	Firmado	Firmado	No	Si
506	Firmado	Firmado Firmado (TTP)	No	Si
535	Firmado (TTP)	Firmado (TTP)	Si (honesto)	Si (honesto)
536	Cancelado	Cancelado	Si (honesto)	Si (honesto)
537	Cancelado	Cancelado	Si (honesto)	Si (honesto)
538	Firmado (TTP)	Firmado (TTP)	Si (honesto)	Si (honesto)
610	Firmado Firmado (TTP)	Firmado (TTP)	Si (malicioso)	Si (honesto)
614	Firmado Firmado (TTP)	Firmado (TTP)	Si (malicioso)	Si (honesta)
616	Firmado Firmado (TTP)	Firmado Firmado (TTP)	Si (malicioso)	Si (malicioso)

Tabla 2. Escenarios especiales pero sin evidencias contradictorias

Los casos más interesantes de entre los listados en la tabla 2 son los nodos 84 y 616. En el nodo 84, B_m no tiene ningún elemento de A_m ya que un intruso I ha ejecutado un evento de *drop* sobre el primer mensaje. B_m no puede ejecutar el subprotocolo de finalización porque no dispone de ningún elemento válido recibido de A_m. Por otra parte, A_m resuelve la situación anormal ejecutando el subprotocolo de cancelación que le permitirá obtener de la TTP un mensaje de cancelación. El segundo nodo, 616, es el caso en el que A_m y B_m actúan maliciosamente contactando con la TTP cuando el subprotocolo de intercambio finaliza satisfactoriamente. Se trata de un caso similar al del nodo 615, pero esta vez B_m contacta en primer lugar con la TTP, obteniendo la correspondiente evidencia de NR. Entonces, si A_m intenta cancelar, la TTP envía una evidencia de NR que probará que el contrato está firmado.

Para resolver las situaciones conflictivas descritas anteriormente, un arbitro debe siempre contactar con ambas partes antes de tomar una decisión, y en caso de encontrar evidencias contradictorias, hemos determinado que debe seguir el siguiente procedimiento de actuación:

- *Caso 1: A* puede afirmar que el contrato está firmado o no (*signed* o *cancelled*), pero *B* posee evidencia de NR que probará que el contrato está firmado. Si *A* intenta usar el mensaje que prueba que el intercambio ha sido cancelado estará demostrando que es un impostor. Por esta razón el árbitro deberá dar la razón a *B.*
- *Caso 2:* Como en el caso anterior, un árbitro deberá contactar con ambas partes en caso de evidencias contradictorias. Si *B* puede aportar las evidencias de NR que prueben que el contrato está firmado, el árbitro puede afirmar que el contrato está firmado, y si B demuestra que el contrato se canceló, el árbitro dará la razón a B y afirmará que el intercambio fue cancelado y el contrato no ha sido firmado. De éste modo, debido al hecho que *A* es siempre la primera parte en hacer trampas, si el árbitro da la razón siempre a B, el protocolo desalentará a *A* a actuar fraudulentamente.

- *Caso 3:* En este caso, de nuevo *A* ha actuado fraudulentamente, y si el árbitro da la razón a *B*, afirmará que el contrato no está firmado ya que el intercambio fue cancelado.

Como conclusión, hemos detectado la situación conflictiva definida en [4] y hemos descubierto dos casos adicionales. Todos los casos son debidos al comportamiento fraudulento de *A*. Para resolver estas situaciones, un árbitro debe contactar siempre con ambas partes y en caso de conflicto siempre dará la razón a *B*. De este modo, el protocolo será equitativo en todos los casos y además las partes no se verán motivadas a comportarse fraudulentamente sabiendo que ello puede ser perjudicial para sus intereses.

5. Conclusiones y Trabajo Futuro

En este artículo se describe como hemos analizado formalmente, usando un mecanismo automático basado en Petri Nets, un protocolo de firma de contratos eficiente, el protocolo FPH [4], una de las soluciones existentes en la que las partes intercambian únicamente tres mensajes.

Para la evaluación del protocolo FPH se ha utilizado un modelo que asume que todos los firmantes pueden ser deshonestos e incluso que un intruso puede atacar también el intercambio. De esta forma el modelo puede ser utilizado para probar la resistencia del protocolo frente a ataques habituales en este tipo de protocolos.

Para la evaluación formal de la equidad del protocolo se han evaluado todas las posibles situaciones derivadas de la intervención de usuarios maliciosos. Se ha demostrado que en todos los casos el intercambio finaliza siempre proporcionando un trato justo a todos los firmantes.

En la realización de la demostración se ha detectado que existen tres casos en los que, aunque el intercambio sea equitativo, uno de los firmantes (o ambos) pueden poseer evidencias contradictorias (que permitan demostrar que el contrato esta firmado o que no está firmado). Por esta razón, aunque el intercambio sea equitativo, no puede decirse que las pruebas generadas sean transferibles. Ambas partes deben ser interrogadas por un árbitro para que se conozca el estado final del intercambio.

Finalmente, hemos generado las reglas de comportamiento del arbitro para que se mantenga la equidad incluso cuando detecte la existencia de evidencias contradictorias.

La creación de un modelo de análisis automático ha permitido verificar una propiedad del protocolo. De modo similar el modelo puede utilizarse para detectar vulnerabilidades frente a ataques. Utilizando el modelo, en un futuro podrán analizarse otras propiedades del protocolo, como la verificabilidad de la tercera parte de confianza, y se podrán modelar protocolos más complejos como la versión multiparte del protocolo de firma de contratos. De forma paralela se seguirá trabajando en la mejora del modelo para permitir nuevos escenarios de ataque, como los ataques confabulados utilizando datos de sesiones diferentes, y para permitir más opciones de control sobre el comportamiento del usuario intruso.

Referencias

1. N. Asokan, M. Schunter and M. Waidner: "Optimistic protocols for fair exchange"; *4th ACM Conference on Computer and Communications Security*, pp. 7-17, 1997.
2. N. Asokan, V. Shoup and M. Waidner: "Asynchronous Protocols for Optimistic Fair Exchange"; *IEEE Symposium on Research in Security and Privacy*, pp. 86-99, 1998.
3. Bao, F., Wang, G., Zhou, J., and Zhu, Z.: "Analysis and Improvement of Micali's Fair Contract Signing Protocol", *Proceedings of The 9th Australasian Conference on Information Security and Privacy*, 2004: 176-187.
4. Ferrer-Gomila, J.L., Payeras-Capellà, M., Huguet-Rotger, L.: "Efficient Optimistic *N*-Party contract Signing Protocol", *Information Security Conference. 4th International Conference, ISC'01*, Lecture Notes in Computer Science 2200, pàgines 394-407, Springer Verlag, 2001.
5. Ferrer-Gomila, J.L., Payeras-Capellà, M., Huguet-Rotger, L.: "Optimallity in Asynchronous Contract Signing Protocols", *Trust and Privacy in digital Bussiness, TrustBus'04*, Lecture Notes in Computer Science 3184, pàgines 200-208, Springer Verlag, 2004.
6. Garay, J.A.; Jakobsson, M., and MacKenzie, P.: "Abuse-Free Optimistic Contract Signing"; *CRYPTO'99*, LNCS 1666, Springer Verlag, pp. 449-466, 1999.
7. Jensen, K., Kristensen, L.M., and Wells, L., "Coloured Petri Nets and CPN Tools for Modeling and Validation of Concurrent Systems", *International Journal on Software Tools for Technology Transfer*, 9(3): 213-254, 2007.
8. Kremer, S., Markowitch, O., Zhou, J.: "An Intensive Survey of Fair Non-repudiation Protocols", *Computer Communications*, Vol. 25, 1606-1621, 2002
9. Micali, S.: "Simple and Fast Optimistic Protocols for Fair Electronic Exchange", *Proceedings of 21st Symposium on Principles of Distributed Computing,* 2003: 12-19.
10. Sornkhom, P. and Permpoontanalarp, Y.: Security analysis of Micali's fair contract signing protocol by using coloured Petri nets", *9th ACIS Int. Conference on Software Engineering, Artificial Intelligence, Networking, and Parallel/Distributed Computing*, 329-334, 2008.
11. J. Zhou, R. Deng and F. Bao: "Some Remarks on a Fair Exchange Protocol"; *PKC 2000*, LNCS 1751, Springer Verlag, pp. 46-57, 2000.