Computación segura multiparte coútil para cálculo de funciones arbitrarias

Jesús A. Manjón
Universitat Rovira i Virgili
Departament d'Enginyeria Informàtica i Matemàtiques
CYBERCAT-Center for Cybersecurity Research of Catalonia
UNESCO Chair in Data Privacy
Avda. Països Catalans 26, Tarragona
jesus.manjon@urv.cat

Josep Domingo-Ferrer
Universitat Rovira i Virgili
Departament d'Enginyeria Informàtica i Matemàtiques
CYBERCAT-Center for Cybersecurity Research of Catalonia
UNESCO Chair in Data Privacy
Avda. Països Catalans 26, Tarragona
josep.domingo@urv.cat

***Resumen*—En la computación segura multiparte (MPC), varias entidades llevan a cabo conjuntamente un cálculo sobre sus respectivos valores de entrada manteniéndolos privados en todo momento. Aunque la MPC existe desde 1980, solamente el desarrollo reciente de compiladores de propósito general ha permitido el cálculo de funciones arbitrarias. Aun así, el uso de esta clase de compiladores requiere un esfuerzo y una habilidad notables por parte de los programadores: el código original se traduce a circuitos booleanos o aritméticos, lo que impone restricciones en el uso de iteraciones y llamadas recursivas, con las consiguientes dificultades de implementación. Para salvar este escollo, presentamos un sistema que permite la ejecución de un protocolo MPC sobre una función arbitraria expresada en código ordinario sin restricciones. Nuestro sistema evita la vinculación entre cada uno de los participantes y sus correspondientes entradas y salidas. Nuestro método se basa en un canal coútil anónimo y un mecanismo de reputación descentralizada, hace un uso limitado de criptografía y no necesita que los participantes sean semihonestos: basta con que sean racionales y actúen en interés propio, lo que incluye a participantes racionalmente maliciosos (que atacan el sistema si ello les beneficia). Nuestros experimentos demuestran que las reputaciones capturan correctamente el comportamiento de los participantes y aseguran la obtención de resultados correctos a los participantes con alta reputación.**

***Index Terms*—Computación multiparte, coutilidad, privacidad, computación por pares**

I. Introducción

En la computación segura multiparte (MPC), varias entidades llevan a cabo conjuntamente un cálculo sobre sus respectivos valores de entrada manteniéndolos privados en todo momento. Desde la aparición de los primeros protocolos en los años 1980s ([1], [2], [3], [4]) y hasta hace poco, sólo existían protocolos MPC prácticos para ciertas aplicaciones específicas. De hecho, la aparición de compiladores de propósito general para calcular funciones arbitrarias es un avance reciente (véase [5] para un estado del arte de esta clase de compiladores). La mayoría de compiladores traducen el código de la computación a realizar a un circuito booleano o aritmético y emplean primitivas criptográficas como la compartición de secretos, la transferencia olvidadiza o los circuitos distorsionados (*garbled circuits*), entre otras, para generar el protocolo MPC.

Como se comenta en [5], el uso de los compiladores más avanzados requiere un gran esfuerzo y habilidad de programación, debido a las limitaciones inherentes de la representación en forma de circuito, principalmente en lo que respecta a las iteraciones y llamadas recursivas (que deben ser acotadas, pues hay que "aplanarlas").

En este artículo proponemos un protocolo MPC con las siguientes características:

- Es de propósito general, en el sentido de que funciona para cualquier tipo de cálculo, sin importar cuán complicado sea;
- Usa criptografía solamente para garantizar la autenticación y la confidencialidad de las comunicaciones;
- Supone la existencia de una comunidad de pares (P2P), en la que cada participante acumula una reputación de manera descentralizada;
- Se basa en la utilización de un canal anónimo coútil que garantiza que las entradas y los resultados, aun siendo visibles para algunos participantes, no pueden vincularse inequívocamente a los participantes a quienes corresponden;
- Proporciona resultados correctos y exactos siempre y cuando los participantes sean racionales, es decir, movidos por su propio interés. Esto no excluye a participantes racionales maliciosos, que podrían convertirse en atacantes si ello les beneficiase. Es decir, el modelo racional que adoptamos és más general que el modelo semihonesto pero menos que el modelo malicioso.

En la sección II presentamos el concepto de coutilidad y el sistema de reputaciones coútil utilizado. En la sección III detallamos los protocolos que conforman el sistema. La sección IV introduce un análisis de coutilidad. La sección V presenta nuestro trabajo experimental. Finalmente, en la sección VI resumimos las conclusiones del artículo.

II. Coutilidad y reputación descentralizada coútil

En términos de teoría de juegos, un protocolo es autoimpuesto (*self-enforcing*) si, en cada paso del protocolo, ejecutar el siguiente paso sin desviarse es un equilibrio del subjuego restante. Por otra parte, un protocolo Π es coútil [6] si y sólo si cumple las tres condiciones siguientes: (i) Π es autoimpuesto; (ii) la utilidad conseguida por un agente que participa en el protocolo es más grande que la que conseguiría sin participar; (iii) no existe otro protocolo Π' que aporte una utilidad mayor al conjunto de agentes ni a un agente concreto.

La primera condición asegura que, si un agente decide participar en el protocolo, no se desvíe en su ejecución. La segunda condición es necesaria para asegurar que participar en el protocolo es atractivo para todos los agentes. Finalmente, la tercera puede ser reformulada en términos de teoría de juegos diciendo que el protocolo es Pareto-óptimo.

En [7] se presenta una adaptación coútil del protocolo de gestión descentralizada de reputaciones EigenTrust [8]. Más allá de su co-utilidad, este protocolo posee las siguientes características: descentralización, pseudonimato de los participantes, coste computacional moderado, adecuada gestión de los nuevos participantes (para disuadir que participantes con baja reputación salgan y entren de nuevo al sistema con nuevas identidades) y resistencia a las manipulaciones de la reputación (como la autopromoción o la difamación). En nuestro artículo usaremos una versión simplificada de dicha gestión descentralizada de reputaciones.

III. Sistema coútil para computación multiparte

III-A. Participantes y modelo de seguridad

Los participantes en nuestro sistema son nodos en una red P2P que pueden asumir uno o más de los siguientes roles:

- Los *clientes* son los participantes que deciden llevar a cabo una computación conjunta, y que aportan valores de entrada privados y obtienen resultados privados. Como el objetivo principal de la MPC es obtener resultados correctos, suponemos que los clientes aportarán siempre entradas correctas.
- Los *operarios* llevan a cabo los cálculos requeridos por los clientes.
- Los *mensajeros* reciben mensajes de los clientes y los reenvían a otros mensajeros, o bien directamente a los operarios.
- Los *gestores de responsabilidad* son los encargados de gestionar la reputación de los otros nodos de la red. A cada nodo P (sea cliente, operario o mensajero) se le asignan de manera pseudoaleatoria M gestores de responsabilidad, evitando de esta manera que un nodo P pueda seleccionar a sus gestores de responsabilidad.

En nuestro sistema no se pueden vincular de manera inequívoca los valores de entrada ni los resultados a sus correspondientes clientes, aunque es cierto que tanto entradas como resultados pueden ser visibles para otros participantes. Nuestra garantía de privacidad se basa en dicha invinculabilidad (*unlinkability*) más que en la confidencialidad. Por ello, *si las entradas o los resultados son tales que no es aceptable revelarlos ni tan sólo de forma no vinculable o tales que sus mismos valores o formatos los hacen vinculables a ciertos clientes, entonces nuestro sistema no debe usarse.*

Suponemos que los participantes son *racionales*: siempre y cuando se les proporcionen los incentivos necesarios, los participantes asumirán y cumplirán correctamente sus roles asignados en el protocolo, aunque se mostrarán interesados en conocer las entradas y los resultados de clientes específicos. Por tanto, hemos diseñado los protocolos de manera que los participantes racionales no tengan incentivos reales para desviarse en su ejecución. Sin embargo, puede haber una minoría de participantes maliciosos que se comporten de manera *irracional* y que se desvíen de los protocolos aunque ello les perjudique también a ellos.

III-B. Requisitos

El objetivo principal de un cliente racional es obtener un resultado correcto de una computación conjunta manteniendo tanto sus valores de entrada como el resultado privados ante el resto de participantes. Para llevar a cabo este objetivo, los protocolos han sido diseñados de manera que sea imprescindible para el cliente tener una alta reputación. Por tanto, el incentivo que tiene un participante para cooperar en el funcionamiento del sistema es incrementar su reputación, para así poder convertirse en un cliente exitoso.

Para que la reputación sea efectiva, debe cumplir los siguientes requisitos:

- *Recompensa*. Si un operario lleva a cabo una computación correcta para un cliente, su reputación debe aumentar.
- *Castigo*. Si un operario lleva a cabo de manera incorrecta una computación para un cliente, su reputación debe disminuir. Igualmente, si un cliente se desvía de los protocolos, su reputación debe disminuir.
- *Recompensa probabilística*. Un participante que actúa de mensajero debe tener una cierta probabilidad de ver aumentada su reputación.
- *Utilidad*. Tener una reputación alta debe ser atractivo para todos los participantes. Concretamente, cuanto más alta sea su reputación, más fácil debe ser para un cliente conseguir un resultado correcto a la vez que preserva su privacidad.

III-C. Computación multiparte de próposito general en el modelo racional: protocolo principal

Para facilitar la comprensión de nuestros protocolos, describimos brevemente como funcionaría una versión básica de los mismos. Unos cuantos nodos de la red P2P deciden participar en una computación conjunta y toman el papel de clientes. Cada cliente P selecciona de manera secreta un operario P_w (que no conoce la identidad de P). Todos los operarios reciben las entradas de todos los clientes a través de un canal coútil anónimo que garantiza que los valores no puedan ser vinculados a sus correspondientes clientes. Una vez que todas las entradas han llegado a todos los operarios, cada cliente P envía a su operario P_w la computación que ha de llevar a cabo y una clave que será usada por P_w para enviar el resultado de vuelta a P a través de un canal anónimo inverso. La idea de utilizar un canal anónimo en MPC fue por primera vez propuesta en [9]; la novedad en nuestro artículo reside en que el canal anónimo no depende de una entidad centralizada y es racionalmente sostenible.

En el modelo racional los participantes pueden desviarse de su comportamiento esperado si no se les incentiva correctamente. Por ejemplo, un operario podría devolver un resultado aleatorio sin llevar a cabo realmente una computación que podría ser costosa o un mensajero podría no reenviar los mensajes recibidos, con el objetivo en ambos casos de ahorrar sus propios recursos. Para superar este problema, añadimos dos mecanismos: (i) redundancia para detectar una computación incorrecta; (ii) reputación descentralizada para recompensar las computaciones y los reenvíos de mensaje correctos, y para

castigar el mal comportamiento. Las reputaciones de todos los participantes son *públicas*.

Protocol 1: MPC RACIONAL DE PROPÓSITO GENERAL

1 Los clientes $ID_1, \ldots, ID_m$ entre un conjunto de n participantes (donde m, n son públicos y $4 \leq m \leq n$) se conocen mutuamente y deciden llevar a cabo una computación conjunta $(O_1, O_2, \ldots, O_m) = C(I_1, I_2, \ldots, I_m)$, donde la entrada I_i y el resultado O_i han de ser privados para ID_i;
2 **for** $i = 1$ **to** m **en paralelo do**
3 El cliente ID_i usa un pseudónimo P_i;
4 P_i "poda" el código C y se queda con la parte C_i que calcula O_i, es decir, $O_i = C_i(I_1, I_2, \ldots, I_m)$;
5 P_i selecciona secretamente como operarios a r nodos $P_{i_1}, \ldots, P_{i_r}$ escogidos aleatoriamente entre los $\kappa_i > r$ nodos con la reputación más cercana a g_i y los recién llegados;
6 **for** $l = 1$ **to** n **do**
7 P_i ejecuta C-FWD-CH$(P_i, PK_l(I_i||nonce_i), PK_l(nil), P_l)$;
8 **for** $k = 1$ **to** r **do**
9 P_i ejecuta $O_{i,k} =$ C-FWD-CH$(P_i, PK_{i_k}(K_i), PK_{i_k}(C_i), P_{i_k})$;
10 **forall the** *gestores de responsabilidad* AM *de* P_i **do**
11 **if** P_i *no muestra a* AM *el recibo de recompensa recibido del primer mensajero* **then**
12 AM asigna una reputación local $\ell_{AM,i} = 0$ a P_i;
13 Sea O_i el valor más frecuente en $\{O_{i,1}, \ldots, O_{i,r}\}$;
14 **for** $k = 1$ **to** r **do**
15 **if** $O_{i,k} = O_i$ *AND* $O_i \neq nil$ **then**
16 P_i asigna $\ell_{ii_k} = 1$;
17 **else**
18 P_i asigna $\ell_{ii_k} = 0$;
19 Ejecutar el protocolo que actualiza las reputaciones globales públicas g_i de cada nodo P_i.

El protocolo 1 es una versión detallada del protocolo explicado anteriormente extendida con redundancia (cada cliente escoge a r operarios diferentes) y reputación descentralizada. Cada cliente P envía sus entradas cifradas a los operarios P_w a través del canal anónimo ejecutando el protocolo 2. Posteriormente les envía por el mismo canal la computación C_i que han de llevar a cabo y la clave K con la que cifrar el resultado, y queda a la espera. Tras recibir los resultados de todos los operarios, el cliente demuestra a sus gestores de responsabilidad que ha recompensado al primer mensajero, compara estos resultados entre sí y considera el mayoritario como el resultado final. Finalmente, recompensa o castiga a los operarios según el resultado recibido.

La idea general de los diferentes protocolos es que un nodo sólo interactuará con otros nodos de similar reputación (g_i). Por ejemplo, un operario sólo aceptará trabajar para un cliente cuya reputación sea superior o muy cercana a la suya, o un mensajero sólo aceptará reenviar un mensaje si éste proviene de otro nodo con una reputación superior o similar a la suya. De esta manera, los nodos con reputaciones bajas (producto de malas computaciones o mal comportamiento) acaban arrinconados sin posibilidad de interacción con otros nodos, y por tanto, sin posibilidad de recibir resultados correctos.

Nota 1 (Los recién llegados): Los nodos que entran en el sistema con posterioridad a su puesta en marcha lo hacen con una reputación global $g = 0$. Si no lleváramos a cabo ninguna acción especial, estos nodos no serían nunca seleccionados por los nodos honestos para ser sus operarios porque la diferencia entre las reputaciones de unos y otros es demasiado grande. Tan sólo podrían computar para los clientes maliciosos, que conseguirían computaciones correctas gracias a los recién llegados honestos que entran en el sistema. Por ello, cuando un nodo entra en el sistema, durante un número determinado de iteraciones se le permite ser escogido por los clientes como operario sea cual sea la diferencia entre reputaciones. De esta forma, el nodo podrá ganar reputación siempre y cuando ejecute las computaciones correctamente. No obstante, como contrapartida y para evitar que los nodos maliciosos encuentren ventajas en salir del sistema y volver a entrar con un pseudónimo diferente (blanqueo), un recién llegado no podrá ejercer él mismo como cliente y no obtendrá ningún tipo de resultado.

Nota 2 (El parámetro κ_i): El parámetro κ_i es secreto para cada cliente P_i. Si $\kappa_i \gg r$, entonces los operarios son escogidos entre un conjunto de nodos más grande, lo que hace más difícil para los operarios saber qué cliente les ha escogido. Sin embargo, por otro lado, escoger un valor de κ_i demasiado alto puede ser arriesgado para P_i porque podría significar que los operarios escogidos tienen una reputación demasiado elevada (y rechazarán trabajar para P_i) o demasiado baja (lo que significaría que no son completamente fiables).

Nota 3 (Compartir operarios): Una manera de reducir la computación y la comunicación del protocolo 1 al aplicar redundancia sería que los clientes compartiesen operarios. Por ejemplo, si cada cliente propusiera un operario, un grupo de m operarios estaría disponible para todos los clientes; para conseguir un nivel de redundancia similar, el protocolo 1 requeriría escoger a m^2 operarios en total. Sin embargo, compartir operarios tiene varias desventajas. Por un lado, todos los operarios llevarían a cabo las mismas computaciones C para todos los clientes. Por otro lado, los clientes se verían forzados a confiar en unos operarios que han sido seleccionados por otros clientes. Esto es particularmente desaconsejable si las reputaciones de los clientes son muy heterogéneas: los clientes con alta reputación pueden escoger a operarios con reputaciones más altas (y por tanto, más fiables) que los propuestos por clientes con reputaciones más modestas.

Nota 4 (Enviar a todos): En la línea 7 del protocolo 1 se puede apreciar que los valores de entrada son enviados a todos los participantes de la red. Esto es necesario para que todos los operarios reciban los valores de entrada de todos los clientes, sin obligar a los mismos a compartir entre ellos los operarios que ha escogido cada uno. De hecho, un nodo no sabe que es operario hasta que recibe el mensaje con la computación que ha de llevar a cabo (línea 9).

III-D. Protocolo para el envío de mensajes a través del canal coútil anónimo

El protocolo 2 corresponde al funcionamiento del canal coútil anónimo. Si un nodo recibe un mensaje, lo primero que hace es comprobar si se lo ha enviado un nodo con una reputación superior o similar a la suya (líneas 21, si el nodo es un mensajero, y 28, si el nodo es el operario destinatario del mensaje). Si la reputación del remitente era demasiado baja, el nodo descarta el mensaje. Si en cambio acepta el mensaje y no es el operario a quien iba dirigido, debe decidir si enviarlo directamente al mismo o reenviarlo a otro mensajero intermedio (línea 19). Si decide reenviarlo, lo hará a un nodo con una reputación superior o similar a la suya (ver nota 6).

Los operarios encargados de las computaciones irán acumulando las entradas de los diferentes clientes (línea 6), y ejecutarán la computación requerida cuando hayan recibido m entradas y C_i (línea 15). El resultado, cifrado con la clave K que les había enviado el cliente previamente, lo enviarán de vuelta al cliente a través del protocolo 3.

Nota 5 (El parámetro de flexibilidad δ): En el protocolo 2, un nodo P_j no descarta un mensaje recibido de un nodo P_i, siempre y cuando la reputación de P_i (g_i) sea al menos $g_j - \delta$, siendo δ una pequeña cantidad de reputación que introduce flexibilidad en la interacción. Un valor δ muy elevado no sería aceptable desde un punto de vista racional: los nodos de alta reputación tienen poco que ganar aceptando mensajes o computando para nodos con una reputación muy inferior a la suya (p.e. si estos nodos de baja reputación son clientes, podrían no recompensarles posteriormente por la computación efectuada).

Nota 6 (La función SELECT*):* En el protocolo 2, la función $P_t = \text{SELECT}(g_s, g_d)$ es ejecutada por un nodo P_s para seleccionar a otro nodo P_t como mensajero, con el fin de que reenvíe el mensaje hacia el operario P_d. Hay diversas maneras de llevar a cabo esta selección. Sin embargo, la opción racional para P_s es escoger a un mensajero P_t con una reputación suficiente para que P_d no descarte el mensaje si P_t se lo envía directamente. Por tanto, si la reputación de P_s es $g_s \geq g_d - \delta$, P_s puede seleccionar aleatoriamente cualquier nodo con una reputación entre $[g_d - \delta, g_s + \delta]$: cualquier mensajero P_t con una reputación de al menos $g_d - \delta$ no correrá el riesgo de ver su mensaje descartado por el operario P_d. En cambio, otros mensajeros P_t con reputaciones superiores a $g_s + \delta$ descartarían directamente los mensajes de P_s. Por otro lado, si $g_s < g_d - \delta$, P_s ha de escoger el nodo con la máxima reputación posible que no exceda de $g_s + \delta$, porque cualquier otro nodo con un reputación superior descartaría directamente su mensaje.

III-E. Protocolo para el envío de resultados a través del canal inverso

Una vez se ha llevado a cabo una computación, un operario envía el resultado de vuelta a través del protocolo 3 (C-REV-CH), ya que, al no conocer la identidad del cliente, no puede utilizar el canal anónimo. Se recorre el camino inverso hasta que el resultado llega al cliente P_{prev} (que es quien puede descifrar el resultado al conocer la clave K) y éste recompensa al primer mensajero P_s, que a su vez le entrega al cliente el recibo que ha de mostrar a sus gestores de responsabilidad en la línea 11 del protocolo 1.

Protocol 2: C-FWD-CH($P_s, msg, Ecomp, P_d$)

1 Parámetro $p \in [0, 1]$;
2 **if** $P_s = P_d$ **then**
3 P_d descifra $comp = SK_d(Ecomp)$;
4 **if** $comp = nil$ **then**
5 P_d descifra $I||nonce = SK_d(msg)$;
6 **if** P_d *no ha recibido previamente* $nonce$ **then** P_d añade I a $Ilist$;
7 **else**
8 **if** $comp =$ *"refuse"* **then**
9 P_d asigna $out := nil$;
10 **else**
11 P_d espera hasta que $Ilist$ contiene m valores de entrada diferentes;
12 P_d computa $out := comp(Ilist)$;
13 P_d vacía $Ilist$;
14 P_d recupera $K = SK_d(msg)$;
15 P_d ejecuta C-REV-CH($P_d, E_K(out), Ecomp, P_{prev}$);
16 **else**
17 **if** P_s *es el origen de* msg **then** $p_{forward} = 1$ **else** $p_{forward} = p$;
18 P_s computa $decision =$ Bernoulli($p_{forward}$);
19 **if** $decision = 1$ **then**
20 P_s envía $(msg, Ecomp)$ a $P_t = \text{SELECT}(g_s, g_d)$;
21 **if** *la reputación de* P_s *es al menos* $g_t - \delta$ **then**
22 P_t ejecuta C-FWD-CH($P_t, msg, Ecomp, P_d$);
23 **else**
24 P_t descarta $(msg, Ecomp)$;
25 **else**
26 P_s envía directamente $(msg, Ecomp)$ a P_d;
27 P_d descifra $comp = SK_d(Ecomp)$;
28 **if** $comp \neq nil$ *y la reputación de* P_s *es menor que* $g_d - \delta$ **then**
29 P_d asigna $Ecomp := PK_d$("refuse");
30 P_d ejecuta C-FWD-CH($P_d, msg, Ecomp, P_d$);

Protocol 3: C-REV-CH($P_s, E_K(out), Ecomp, P_{prev}$)

1 P_s envía $(E_K(out), Ecomp)$ a P_{prev};
2 **if** P_{prev} *conoce la clave* K **then**
3 P_{prev} descifra out;
4 P_{prev} envía S_{prev}("P_{prev} recompensa a P_s") a P_s ;
5 P_s reenvía S_{prev}("P_{prev} recompensa a P_s") a los AM de P_{prev};
6 Los AM de P_{prev} ponen $\ell_{prev,s} = 1$;
7 P_s devuelve un recibo firmado S_s("P_s acusa recibo de recompensa de P_{prev}") a P_{prev};
8 **else**
9 Sea P_{prev2} el nodo del cual P_{prev} recibió el mensaje con la computación $Ecomp$;
10 P_{prev} ejecuta C-REV-CH($P_{prev}, E_K(out), Ecomp, P_{prev2}$).

Nota 7 (Recompensa únicamente para el primer mensajero): Como hemos comentado, en el protocolo C-REV-CH solamente se recompensa al primer mensajero. El protocolo que actualiza las reputaciones globales requiere que las reputaciones locales sean normalizadas previamente (ver [8]). Por tanto, si todos los mensajeros fueran recompensados por el cliente, el incremento de reputación de cada uno de los mensajeros sería mucho menor. Un mensajero que recibiera un mensaje msg lo enviaría siempre directamente a su destino final con el objetivo de minimizar el número de mensajeros para msg. Por ello, en la práctica, solo existiría un mensajero, que automáticamente sabría que el nodo que le ha enviado msg es el cliente, lo que rompería el anonimato del canal.

IV. Análisis de co-utilidad

En la presente sección argumentamos por qué el sistema formado por los protocolos presentados anteriormente es coútil, es decir, por qué será ejecutado sin desviaciones por participantes racionales:

- El objetivo de los clientes es ejecutar computaciones conjuntas y obtener sus correspondientes resultados correctos, a la vez que mantienen privados tanto sus valores de entrada como sus resultados. Por esta razón, podemos suponer que los clientes llevarán a cabo correctamente sus tareas en los protocolos 1, 2 y 3. Si un cliente no recompensara al primer mensajero, sería penalizado con una reducción de su reputación, lo que le haría más difícil obtener resultados correctos en sucesivas computaciones. También tiene sentido que los clientes escojan a un recién llegado como uno de sus r operarios redundantes: cuantos más nodos honestos participen, mayor resistencia a ataques tendrá el sistema. La redundancia de los operarios minimiza el riesgo en caso de que el recién llegado sea malicioso.
- Los mensajeros son esenciales para la ejecución del canal anónimo en los protocolos C-FWD-CH y C-REV-CH. Su incentivo es ser recompensados en el protocolo 3 como primer mensajero (*a priori* no saben si son o no son el primer mensajero) con un incremento de su reputación.
- De los operarios se espera que lleven a cabo correctamente la computación requerida en C-FWD-CH. Posteriormente, en C-REV-CH se espera que inicien el recorrido inverso del trayecto para devolver el resultado. Su incentivo es ser recompensados en el protocolo 1 si el resultado que devuelven es el mayoritario entre los devueltos por el conjunto de operarios de ese cliente.
- Los gestores de responsabilidad tiene un rol importante en los protocolos 1 y 3, y en el protocolo de cálculo de las reputaciones globales. En la descripción del modelo de seguridad (Sección III-A) suponíamos que los nodos era racionales, aunque no excluíamos la posibilidad de ataques racionales. Dado que los nodos interactuarán en sucesivas iteraciones, el interés racional de los gestores de responsabilidad será favorecer a los nodos que han llevado a cabo computaciones correctas, ya que ellos mismos pueden ser clientes en futuras rondas de computación. Por el otro lado, el hecho de que los M gestores de responsabilidad de un nodo sean escogidos de manera pseudoaleatoria dificulta el conflicto de interés y facilita la gestión honesta de la reputación de los nodos.

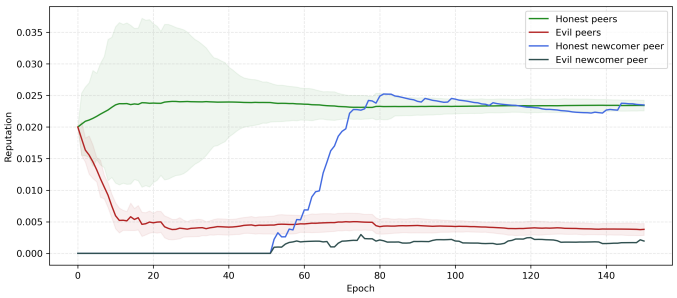


Figura 1. Evolución de las reputaciones de las diferentes clases de nodos en un escenario con el 20 % de nodos maliciosos

V. Resultados experimentales

En esta sección mostramos los resultados de los experimentos llevados a cabo con el sistema coútil propuesto. Consideramos *nodos honestos* (los que llevan a cabo todos los pasos de los protocolos correctamente) y *nodos maliciosos* (los que no).

Resultado esperado. Si nuestro sistema está bien diseñado, los nodos honestos deberían obtener una mayor proporción de resultados correctos que los nodos maliciosos. Además, la reputación de cada nodo debería estar muy correlacionada con su comportamiento y con la proporción de computaciones correctas que obtiene. La razón es que es probable que los nodos que llevan a cabo computaciones incorrectas como operarios acaben teniendo una baja reputación, y para un nodo con baja reputación es complicado encontrar (como cliente) a operarios o mensajeros fiables. Asimismo, un nodo honesto recién llegado debería tener a medio plazo una proporción de computaciones correctas parecida a la del resto de nodos honestos. En cambio, si el recién llegado se comportara de manera maliciosa, debería mantener una proporción baja de computaciones correctas, como el resto de nodos maliciosos de la red. De esta manera, un nodo malicioso con baja reputación no conseguiría ninguna ventaja si decide dejar el sistema y volver a entrar con un pseudónimo diferente.

Configuración experimental. Construimos una red P2P con $n = 50$ nodos y dejamos que evolucionase durante 150 iteraciones; en cada iteración se llevó a cabo una computación conjunta que implicó a $n = 10$ clientes escogidos aleatoriamente entre los 50 nodos. Cada cliente aportó un número como valor de entrada privado para la computación, que consistía simplemente en devolver la posición de ese valor en la ordenación de todos los valores de los clientes. En este caso, un nodo malicioso retornaba siempre un valor aleatorio en lugar del resultado real de la computación (con lo que mantenía un 10 % de posibilidades de acertar el resultado correcto). Definimos $r = 3$ (tres operarios redundantes para cada cliente), otorgamos una reputación inicial de $g = 1/n = 0{,}02$ a cada cliente, fijamos inicialmente $\delta = g * 0{,}75 = 0{,}015$ y un período para considerar a un nodo como recién llegado de 25 iteraciones.

En el experimento, el 20 % de los nodos eran nodos maliciosos y además introdujimos 2 nodos recién llegados

al sistema en la iteración 50, uno honesto y otro malicioso. La figura 1 muestra la evolución de la reputación de cada clase de nodos durante las 150 iteraciones. Las líneas indican la reputación media de los nodos de un tipo determinado, mientras que la región sombreada muestra la desviación estándar. Podemos observar como ya en las primeras iteraciones, mientras el comportamiento de los nodos está siendo modelado, la reputación de los nodos maliciosos va decreciendo rápidamente. Alrededor de la novena iteración ya hay una clara diferencia entre las reputaciones de los nodos honestos y las de los maliciosos. A partir de la iteración 50 aparece también la evolución tanto del recién llegado honesto (azul) como del malicioso (gris). Como durante las primeras 25 iteraciones tras su aparición, el resto de nodos les pueden escoger como operarios a pesar de no tener la reputación adecuada, el comportamiento de ambos nodos es modelado convenientemente. La reputación del recién llegado honesto va aumentando hasta situarse al nivel del resto de nodos honestos, mientras que la del recién llegado malicioso se queda al nivel del resto de nodos maliciosos de la red. Recordemos que, durante esta fase, los recién llegados sólo pueden ejercer de operarios y mensajeros, no de clientes, por lo que un nodo malicioso no obtendría ninguna ventaja si decide salir y entrar de nuevo al sistema con otro pseudónimo. Evitamos de este modo ataques de blanqueo.

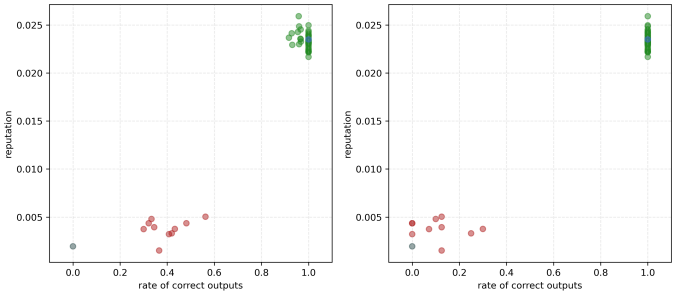


Figura 2. Proporción de computaciones correctas en función de la reputación de los diferentes nodos, con un 20 % de nodos maliciosos. A la izquierda, en las 150 iteraciones; a la derecha, en las últimas 50.

En el gráfico izquierdo de la figura 2 vemos la proporción de computaciones correctas que han obtenido tanto los nodos honestos (en verde), como los maliciosos (rojo) y los recién llegados (en azul el honesto y en gris el malicioso) desde la primera iteración, en un experimento con la configuración explicada anteriormente. Podemos observar que la gran mayoría de las computaciones que han solicitado los nodos honestos han sido correctas. En cambio, los nodos maliciosos han recibido aproximadamente la mitad de sus computaciones de manera correcta, lo que cumple con lo esperado. De hecho, como se puede ver en el gráfico derecho de la figura 2, la reputación es mucho más decisiva en las últimas 50 iteraciones. En estas iteraciones, todas las computaciones solicitadas por los clientes honestos (con alta reputación) fueron correctas, lo que contrasta con la proporción de 10 % de computaciones correctas entre las solicitadas por los nodos maliciosos (con baja reputación). Por tanto, en cuanto el sistema se estabiliza, los nodos honestos obtienen siempre resultados correctos.

VI. Conclusiones

En este artículo hemos presentado un sistema P2P para computación multiparte que, con un uso muy moderado de la criptografía, asegura que no podrán vincularse inequívocamente los valores de entrada y los resultados con las partes correspondientes. La ventaja principal de este sistema es que es de propósito general y no hace uso de circuitos, por lo que funciona para cualquier tipo de cálculo expresado en código de alto nivel (sin restricciones de iteraciones o llamadas recursivas).

Como trabajo futuro prevemos ampliar el análisis de coutilidad y llevar a cabo más experimentos que nos permitan afinar la configuración inicial de los parámetros del sistema. La investigación futura se centrará en intentar reducir el coste de comunicaciones.

Agradecimientos

Agradecemos las subvenciones de los organismos siguientes: Comisión Europea (proyectos H2020-871042 “SoBigData++” y H2020-101006879 “MobiDataLab”), Generalitat de Catalunya (Premio ICREA Acadèmia al segundo autor) y MCIN/AEI /10.13039/501100011033 /FEDER, UE (proyectos RTI2018-095094-B-C21 y PID2021-123637NB-I00). Los autores pertenecen a la Cátedra UNESCO de Privacidad de datos, pero las opiniones en este artículo son suyas y no necesariamente compartidas por UNESCO o los organismos financiadores.

Referencias

[1] A. C. Yao: “Protocols for secure computations”, en *23rd Annual Symposium on Foundations of Computer Science - SFCS 1982*, pp. 160-164. IEEE, 1982.

[2] O. Goldreich, S. Micali, A. Wigderson: “How to play any mental game or a completeness theorem for protocols with honest majority”, en *19th Annual ACM Symposium on the Theory of Computing - STOC 1987*, pp. 218-229. ACM, 1987.

[3] M. Ben-Or, S. Goldwasser, A. Wigderson: “Completeness theorems for non-cryptographic fault-tolerant distributed computation”, en *20th Annual ACM Symposium on the Theory of Computing - STOC 1988*, pp. 1-10. ACM, 1988.

[4] D. Chaum, C. Crépeau, I. Damgård: “Multiparty unconditionally secure protocols”, en *20th Annual ACM Symposium on the Theory of Computing - STOC 1988*, pp. 11-19. ACM, 1988.

[5] M. Hastings, B. Hemenway, D. Noble, S. Zdancewic: “SoK: General purpose compilers for secure multiparty computation”, en *IEEE Symposium on Security and Privacy - SP 2019*, pp. 1220-1237. IEEE, 2019.

[6] J. Domingo-Ferrer, S. Martínez, D. Sánchez, J. Soria-Comas: “Co-utility: self-enforcing protocols for the mutual benefit of participants”, en *Engineering Applications of Artificial Intelligence*, vol. 59, pp. 148-158. 2017.

[7] J. Domingo-Ferrer, O. Farràs, S. Martínez, D. Sánchez and J. Soria-Comas: “Self-enforcing protocols via co-utile reputation management”, en *Information Sciences*, vol. 367-368, pp. 159-175. 2016.

[8] S. D. Kamvar, M. T. Schlosser, H. Garcia-Molina. “The EigenTrust algorithm for reputation management in P2P networks”, en *12th International Conference on World Wide Web*, pp. 640-651. ACM, 2003.

[9] Y. Ishai, E. Kushilevitz, R. Ostrovsky, A. Sahai: “Cryptography from anonymity”, en *47th Annual IEEE Symposium on Foundations of Computer Science - FOCS 2006*, pp. 239-248. IEEE, 2006.