8								
		3	6					
	7			9		2		
	5				7			
				4	5	7		
			1				3	
		1					6	8
		8	5				1	
	9					4		

Demostraciones de Nula Divulgación: o cómo convencer a alguien que mi sudoku tiene solución, sin revelarla

Alejandro Hevia

Profesor Asistente Departamento de Ciencias de la Computación, Universidad de Chile. Ph.D. Computer Science, University of California, San Diego (2006); Ingeniero Civil en Computación, Universidad de Chile (1998). Director del Grupo de Respuesta a Incidentes de Seguridad Computacional, CLCERT. Líneas de Investigación: Criptografía Aplicada, Seguridad Computacional.

ahevia@dcc.uchile.cl

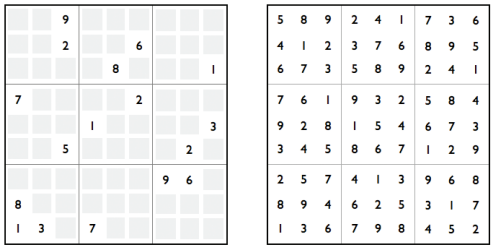
En este artículo presentaremos un concepto aparentemente paradójico pero extremadamente poderoso surgido del corazón de la computación teórica: el de las demostraciones de Nula Divulgación. Mostraremos cómo aplicarlo a los juegos de sudoku, lo que analizaremos está lejos de ser un mero juego. Veremos además cómo la Nula Divulgación permite el surgimiento de un amplio espectro de aplicaciones modernas para preservar la privacidad de las personas, en particular, aplicaciones que demuestran convincentemente la posesión de un dato secreto sin revelarlo.

LA HISTORIA

Como buenos fanáticos del sudoku (ver Figura 1), Alicia y Roberto rutinariamente intercambian puzles de sudoku para resolver. En una oportunidad, Alicia encuentra un puzle sudoku fascinante, el cual decide compartir con Roberto y pedirle que lo resuelva. Roberto, quien no quiere perder el tiempo, sospecha que el puzle no tiene solución por lo que desafía a Alicia a demostrarle que sí la tiene. Claramente, Alicia podría darle la respuesta a Roberto, quien la comprobaría y listo. Funciona, pero ¡le quitaría toda la diversión a resolver el puzle! ¿Cómo podemos hacerlo mejor? ¿Cómo puede Alicia convencer a Roberto que el puzle tiene solución sin revelársela? Para resolver este problema, debemos primero discutir el concepto de demostración[1].

TEOREMAS Y DEMOSTRACIONES

¿Qué exactamente le pide Roberto a Alicia al solicitarle una demostración? Una demostración matemática es típicamente una lista de pasos lógicos, algo que Alicia puede escribir y enviar a Roberto. Tal experiencia está tan arraigada en la comunidad matemática que rara vez nos preguntamos qué hay detrás. Al examinarlo en su manifestación más simple, la demostración de un teorema en matemáticas en realidad está formada por dos componentes: un teorema o aseveración lógica a demostrar (digamos x) y la demostración misma (un texto, digamos w). En nuestro caso, llamemos Z al conjunto de todos los puzles sudoku que tienen solución. Dado un puzle x, si Alicia puede dar la demostración w = "la solución del puzle por filas es 5, 8, ..., 2", entonces Roberto puede convencerse que $x \in Z$ simplemente ejecutando un procedimiento que chequee que la solución w es válida, un proceso razonablemente rápido.



		9						
		2			6			
				8				1
7					2			
			1					3
		5					2	
						9	6	
8								
1	3		7					

5	8	9	2	4	1	7	3	6
4	1	2	3	7	6	8	9	5
6	7	3	5	8	9	2	4	1
7	6	1	9	3	2	5	8	4
9	2	8	1	5	4	6	7	3
3	4	5	8	6	7	1	2	9
2	5	7	4	1	3	9	6	8
8	9	4	6	2	5	3	1	7
1	3	6	7	9	8	4	5	2

Figura 1 • El Sudoku es un tipo de puzle que consiste en completar una matriz de 9 filas y 9 columnas con números del 1 al 9. El juego parte con la matriz sólo parcialmente llena (unas 18 a 20 celdas, como se ve en el tablero de la izquierda) y el problema consiste en completar las celdas restantes con valores tales que toda fila y toda columna contenga números distintos, así como cada una de las 9 submatrices disjuntas de tamaño 3x3 que particionan la matriz (como se ve en el tablero de la derecha).

Así, no es difícil convencerse de que una demostración matemática cualquiera, tal como la hemos descrito, puede verse como un proceso: dado un teorema x, el demostrador (Alicia) le envía un sólo mensaje –la demostración w– al verificador (Roberto), quien lo revisa en forma rápida e independiente, para emitir una decisión: la demostración es válida o no. En Teoría de la Computación, el conjunto de todos los problemas para los cuales pueden darse demostraciones cortas, eficientemente chequeables como la anterior, se denomina NP.

Paréntesis: P versus NP

Es interesante notar que la clase NP de problemas recién descrita coincide con una clase importantísima de problemas usados en la práctica: aquellos donde verificar una solución del problema puede hacerse en forma eficiente, aún si no sabemos cómo encontrar una solución. Estos problemas aparecen por todos lados en ingeniería, especialmente al resolver problemas de optimización. Por ejemplo, el problema del Camino Hamiltoneano: dado x = "Un grafo G de n nodos", determinar si existe en G un camino que pase por todos los nodos exactamente una vez[2]. Claramente, si nos

[1] Este artículo está basado en una presentación de Mike Rosulek [6].

[2] Recordar que un grafo es simplemente un conjunto de nodos unidos por aristas. Un ejemplo clásico de grafo es un mapa de las ciudades y carreteras de un país: las ciudades son los nodos y las carreteras conectando las ciudades son las aristas. En dicho caso, un Camino Hamiltoneano es una ruta para visitar todas las ciudades pasando una y sólo una vez por cada ciudad.

dan como "pista" un camino (llamémosle *w*), podemos contestar rápidamente simplemente verificando si el camino dado pasa por cada nodo una y sólo una vez. Sin embargo, nadie sabe en la actualidad cómo encontrar eficientemente un camino para un grafo arbitrario. Como comparación, existe la clase de problemas (llamada *P*) que contiene a todos aquellos que pueden resolverse eficientemente sin necesidad de "pistas": por ejemplo, determinar si hay un camino entre dos nodos cualesquiera en un grafo puede hacerse rápidamente (con un algoritmo del tipo greedy, ver [1]). ¿Cómo se comparan P y NP? ¿Encontrar una solución es tan fácil como verificar si está correcta? Pareciera que no. Sin embargo, en estricto rigor, ¡no lo sabemos! Esta pregunta, la relación **P versus NP**, es probablemente el problema abierto más famoso de la Teoría de la Computación. De hecho, actualmente hay un premio, del Clay Mathematics Institute, de 1 millón de dólares para el primero que lo resuelva.

Existe una subclase de problema dentro de los problemas en NP denominados problemas NP-Completos. Estos problemas representan en algún sentido los problemas más difíciles de resolver en NP. Un ejemplo de éstos es justamente el problema del Camino Hamiltoneano. Los problemas NP-Completos tienen la propiedad que, si uno puede encontrar la solución de uno de ellos eficientemente, entonces podemos encontrar la solución de todos los problemas en NP eficientemente. Tal resultado, denominado el Teorema de Cook-Levin, es teóricamente impresionante, pues caracteriza los problemas más difíciles en la clase. Sin embargo, también es útil en la práctica. En la demostración de dicho teorema se entrega una manera explícita (la cual llamamos *t*) de transformar una instancia *x* de un problema *L* cualquiera en NP en una instancia $x'=t(x)$ del problema NP-Completo. Usando esta *función* podemos, por ejemplo, transformar el problema de demostrar que un texto cifrado *C* dado (calculado usando un sistema de clave pública) contiene un mensaje *M* igual a 0 ó 1, a demostrar que un cierto grafo $G = t(G)$ contiene un Camino Hamiltoneano. Esto es, si podemos demostrar que *G* tiene un camino, entonces estaremos demostrando que *C* codifica un mensaje igual a 0 ó 1. Quizás suene muy teórico, pero poder traducir la solución de un problema cualquiera (en NP) a encontrar la solución de un problema NP-Completo es extremadamente útil, como veremos al final.

DEMOSTRACIONES INTERACTIVAS: GENERALIZANDO LAS DEMOSTRACIONES MATEMÁTICAS

Volvamos al problema de Alicia. A priori, no se ve claro cómo podría ella demostrar que el sudoku tiene solución, sin revelar dicha solución; menos aún si Roberto debe enviarle un sólo mensaje. Aquí llega la interacción a nuestro auxilio.

Todo aquel que haya tenido que entender una demostración matemática se ha dado cuenta que no siempre es fácil convencerse de una demostración simplemente leyéndola. Mucho mejor es poder hacer preguntas a quien nos presenta la demostración, lo cual típicamente facilita comprenderla. Por lo mismo, ¿no sería "más fácil" si ahora le permitimos a Alicia y Roberto conversar? Esto es, ¿ayudaría si le permitimos a Alicia demostrar interactivamente a Roberto que su teorema es cierto? En esta conversación, Alicia y Roberto intercambian mensajes hasta que finalmente Roberto se detiene y decide si "le cree" a Alicia o no. Este proceso se denomina una demostración interactiva y veremos que efectivamente ayuda. En nuestro ejemplo del sudoku, Alicia y Roberto podrían usar la estrategia que llamaremos **Chequear-Fila-Luego-Columna:** Roberto podría pedirle a Alicia que le muestre una cierta columna específica (escogida por él) de la solución del sudoku, verificando que sólo contenga números distintos. Luego, Roberto podría repetir este proceso para otra fila y, si Alicia responde exitosamente a ambas preguntas, Roberto decide creerle. ¿Piensa usted que Roberto debiera estar convencido?

Para responder, necesitamos una definición un poco más precisa de lo que es una demostración interactiva. Un protocolo o sistema de demostración interactiva para un problema *L* es un par de algoritmos, uno para el demostrador (*P* por *Prover* en inglés) y otro para el verificador (*V*), los cuales para poder capturar la idea de una demostración, deben satisfacer dos propiedades: correctitud y robustez. Un protocolo es correcto si para toda aseveración *x* verdadera (esto es, $x \in Z$), P siempre o con alta probabilidad convence a *V*. Un protocolo es robusto si ningún demostrador, aún uno malicioso que ejecuta un programa distinto a *P*, puede convencer a un *V* honesto que una cierta aseveración falsa, $x \notin L$, excepto con una probabilidad minúscula.

Claramente, el protocolo **Mostrar-La-Solución** descrito al comienzo, donde Alicia simplemente revela la solución a Roberto, satisface las dos condiciones: una Alicia (en el rol de *P*), honesta siempre, puede convencer a un Roberto (en el rol de *V*) honesto si es que el puzle sudoku exhibido tiene una solución; por otro lado, si el puzle no tiene solución, Alicia no tiene ninguna posibilidad de exhibir una solución que convenza a Roberto.

Por otro lado, el protocolo **Chequear-Fila-Luego-Columna** anterior no satisface la definición pues una Alicia (en el rol de *P*) que no siga las reglas puede engañar a Roberto (en el rol de *V*). Esto porque, aun si el sudoku no tiene solución, en general no es demasiado difícil completar una fila arbitraria y luego otra columna arbitraria para que tengan cada una sólo números distintos. Más adelante mostraremos no sólo cómo corregir este problema, sino cómo evitar que Alicia tenga que mostrar parte o toda la solución a Roberto.

TIRANDO MONEDAS

Un aspecto fundamental de las demostraciones interactivas es su utilización de la aleatoriedad, lo cual es mejor graficado a través de una historia. El famoso matemá-

tico Ronald Fisher [2] contaba la historia de Muriel, una dama de sociedad, quien le aseguraba que el orden en la combinación de los ingredientes en la preparación de un té con leche afectaba su sabor. Según Muriel, un té vertido sobre leche sabe distinto a leche vertida sobre té. Más aún, Muriel le aseguraba poder distinguirlos. Intrigado, Ronald prepara el siguiente experimento para determinar si creerle a Muriel o no.

1) Primero, en privado y lejos de la vista de Muriel, el matemático tira una moneda. Si es cara pone primero té y luego leche en una taza. Si es sello, lo hace al revés.
2) Luego, le entrega la taza a Muriel, quien la prueba e intenta adivinar.
3) Ronald y Muriel repiten el proceso anterior n veces.
4) Si Muriel acierta todas las n veces, Ronald acepta y cree que Muriel tiene la habilidad señalada.

Claramente, si no hay diferencia en los distintos tipos de té con leche, Muriel debiera adivinar una iteración con probabilidad 1/2. Por otro lado, si Muriel puede adivinar debiera responder correctamente siempre (o casi siempre), lo cual nos dice que el protocolo es correcto.

¿Por qué, de acertar Muriel todas las veces esta demostración, debiera convencer a Ronald? Es interesante notar que la decisión de poner primero leche o té en la taza es aleatoria e independiente de Muriel, lo cual es fundamental para argumentar la robustez del protocolo: la probabilidad de Muriel de adivinar las n veces sin tener la habilidad es minúscula (esto es, a lo más 2^{-n}). Luego, el protocolo es efectivamente una demostración interactiva. Notemos un aspecto interesante aquí: luego de ser convencido, Ronald no puede convencer a otros *ni transmitir su convencimiento*. Todo lo que tiene Ronald ahora son sus notas (por ejemplo, "en la primera iteración vertí leche y luego té, Muriel adivinó. En la segunda, vertí té y luego leche, Muriel adivinó..." etc.), las cuales no convencen a nadie pues pudieran haber sido falsificadas. Esto es algo contraintuitivo; convencer a alguien no es lo mismo que revelar información "útil", un punto que desarrollamos en detalle más adelante.

Paréntesis: ¿cuán grande es la clase de los problemas demostrables interactivamente? Sorprendentemente, el simple cambio de agregar interactividad a una demostración tiene un efecto inmenso en el tipo de problemas que podemos demostrar. Es posible mostrar que *IP*, la clase de todos los problemas que admiten demostraciones interactivas donde el verificador (Alicia) es eficiente, contiene no sólo a la clase NP sino a problemas mucho más difíciles de resolver. Por ejemplo, contiene problemas que requieren tiempo exponencial. En la práctica esto significa que, por ejemplo, hay problemas que no sabemos cómo verificar una solución eficientemente, pero para los cuales ¡*sí* podemos demostrar que existe una solución! Un ejemplo de ello es el problema de demostrar que dos grafos del mismo número de nodos NO son isomorfos, esto es, demostrar que uno de ellos no es simplemente una permutación de los nodos del otro. No es ni siquiera claro qué *solución o pista* uno pudiera recibir para poder resolver el problema. De hecho, se desconoce si este problema está en NP, se conjetura que no.

NULA DIVULGACIÓN

Al demostrar una aseveración, una persona puede o no estar revelando información. En el caso de los puzles de sudoku, Alicia está revelando parte o toda la solución a Roberto, lo cual a ambos les gustaría evitar. Más generalmente, existen muchos escenarios donde queremos poder demostrar algo, un teorema (una propiedad de un cierto objeto) sin revelar el *cómo* podemos demostrarlo. Por ejemplo, al conectarse remotamente a un servidor, un usuario debe autentificarse remotamente, usualmente vía *demostrar* que conoce su contraseña, ¿y cómo lo hace? Simplemente la revela al servidor. Lamentablemente, esto puede causar que la contraseña se filtre y afecte la seguridad del servidor.

¿Puede un usuario demostrar que conoce una contraseña sin revelarla? ¿Puede Alicia demostrar que su puzle tiene solución sin revelarla?

Información y aprender

Lo que buscamos es un protocolo de demostración interactiva que permita demostrar una aseveración sin revelar "demasiada información", de hecho, sin "revelar ninguna otra información más allá de que la propiedad demostrada se tiene". Pero, ¿qué significa revelar información? Una manera de responder esto consiste en preguntarse qué puede hacer un verificador (por ejemplo Roberto), después de interactuar con el demostrador (por ejemplo Alicia) en la demostración de una aseveración x. Lo que sea que Roberto ha aprendido sobre x puede verse como una función f sobre x, la cual Roberto no podía calcular antes de la demostración, pero sí puede después de ella (ver Figura 2). Claramente, la única diferencia entre antes y después son los mensajes que recibió de Alicia durante la demostración. En el protocolo **Mostrar-la-Solución** el verificador Roberto puede calcular la función $f(x)$= "la solución del puzle x" puesto que de hecho recibe tal información de Alicia durante la demostración.

Si algo es calculable por mí solo, es que ya lo sabía

Ahora bien, si Roberto puede generar por sí solo una transcripción de todos los mensajes que recibiría en una demostración con Alicia *pero sin interactuar con ella*, entonces podemos decir que Roberto no ha aprendido nada. ¿Por qué? Si Roberto puede calcular por sí

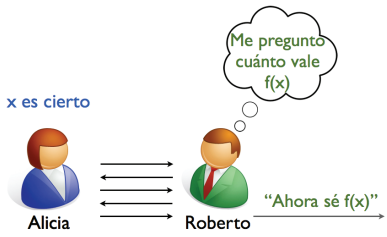


Figura 2 • Una interacción entre Alicia y Roberto donde éste último aprende algo.

solo dichos mensajes, entonces tiene todo lo necesario para calcular $f(x)$, pero ahora sin haber interactuado con Alicia. Luego, la interacción con Alicia no aporta nada extra para calcular $f(x)$, lo cual significa que Roberto no aprende nada nuevo. Conceptualmente, un protocolo de demostración interactiva (esto es, un par de algoritmos para el demostrador P y para el verificador V) es de *Nula Divulgación* (o *Zero-Knowledge* en Inglés) si existe una manera, un algoritmo, S, para producir una transcripción de los mensajes intercambiados entre P y V en una conversación, pero donde:

- El algoritmo S nunca conversa con el demostrador P, sólo con V, y
- La transcripción "luce igual", esto es, *indistinguible*, de una transcripción real de la conversación entre el verificador V y el demostrador P.

Como *bonus*, notemos que un protocolo de Nula Divulgación no permite al verificador V convencer a un tercero de lo demostrado por el demostrador P, puesto que la transcripción de la conversación misma con P –el único registro de la interacción con P– puede ser falsificada por el mismo V y, por ende, no puede significar evidencia alguna para un tercero (ver Figura 3).

Un primer ejemplo

Un ejemplo simple de protocolo de Nula Divulgación es el protocolo descrito anteriormente, diseñado por Ronald Fisher para permitir a Muriel demostrar su supuesta habilidad para diferenciar los tipos de té. La conversación entre Ronald y Muriel está contenida en el registro del matemático donde indica, para cada iteración, la moneda tirada (cara o sello), y si Muriel adivinó o no. Claramente, dicha tabla puede ser generada sin nunca interactuar con Muriel. No sólo Ronald no aprende nada, sino que un tercero no puede ser convencido de las supuestas habilidades de Muriel si la única evidencia de Ronald al respecto es sólo una tabla que cualquiera podría generar por sí solo. Ahora bien, este protocolo es algo artificial pues el demostrador necesita una habilidad algo extraña, la cual no sabemos si existe. Necesitamos un mejor ejemplo, y qué mejor que un protocolo para demostrar que un sudoku tiene solución. Este protocolo necesita el equivalente a una caja fuerte pero en el mundo digital.

Cajas fuertes digitales

En el mundo físico, una manera de enviar un número sin revelarlo es meter un papel con el número escrito en una caja fuerte, la cual es cerrada y luego enviada por encomienda, sin adjuntar la llave. En el mundo digital, su análogo se denomina un compromiso o *commitment* (en inglés). Podemos crear y abrir commitments. Dado un valor v, para crear un commitment c con v adentro, calculamos c = *commitment* (v ; r) donde r es un valor aleatorio. Luego, el commitment c se envía al receptor. Con ello, quien envía sabe el valor v y conoce la llave r, mientras que el receptor sólo ve la caja "por fuera", esto es c, lo cual no revela v. Sólo quien envía puede abrir un commitment, esto es, convencer a alguien que el valor almacenado en c es v. En el mundo físico, esto se haría simplemente enviando al receptor la llave de la caja fuerte por correo. En el mundo digital, el valor r juega el rol de la clave y es lo que se envía. Los commitments son diseñados con dos propiedades análogas a su contraparte física: (1) el receptor de un commitment c no puede saber qué valor "contiene" el commitment, y (2) una vez recibido el commitment, quien lo envió no puede convencer al receptor que el valor almacenado es distinto a v, esto es, no puede abrirlo a un valor distinto al almacenado originalmente en él. Claramente el receptor de la caja fuerte al no poder abrir la caja no sabe qué valor contiene, y quien la envío no puede ya cambiar el valor almacenado dentro de la caja fuerte una vez que dicha caja está en posesión del receptor. Un esquema de encriptación de clave pública puede usarse como commitment si quien envía el commitment es quien genera el par de claves pública y privada, siempre que sea posible para el receptor verificar que la clave pública fue generada correctamente. Para más detalles y ejemplos, ver [5].

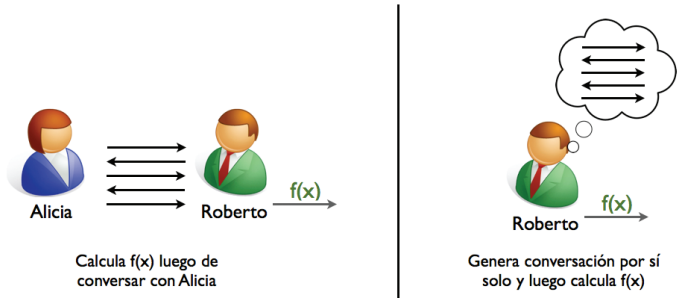


Figura 3 • En Nula Divulgación, cualquiera sea lo que Roberto puede calcular después de ver la transcripción de la conversación con Alicia, lo puede calcular sin conversar con Alicia.

DEMOSTRACIÓN DE NULA DIVULGACIÓN PARA SUDOKU

Mostraremos ahora cómo Alicia usando commitments puede demostrar a Roberto que su sudoku tiene solución sin revelarla[3]. En el siguiente protocolo, Alicia actúa como el demostrador (P) y Roberto como el verificador (V). Ambos conocen el puzle sudoku sin resolver (llamémosle x):

1.- Alicia: escoge una permutación secreta al azar π de{1,...,9} (por ejemplo, $\pi(1)=$, $\pi(2)=3$, etc.), la cual aplica a cada uno de los valores en las celdas de la solución.

[3] Este protocolo fue propuesto originalmente en [3], donde se puede ver más detalles y un análisis más formal.

Ello produce una matriz del mismo tamaño, llamémosla x', con todos los valores re-etiquetados: donde había un 1, ahora hay un 5, donde había un 2 hay ahora un 3, etc. (ver Figura 4a). Alicia entonces crea commitments para cada una de las celdas de la nueva tabla x', formando una tabla de 9x9 con 81 commitments, la cual llamamos C. Finalmente, Alicia envía C a Roberto.

2.- Roberto: desafía a Alicia, escogiendo al azar una y sólo una de las siguientes opciones:

(a) Revelar fila: en esta opción, Roberto escoge una fila al azar y le pide a Alicia que le revele los valores (abra los commitments) para toda esa fila.
(b) Revelar columna: en esta opción, Roberto escoge una columna y le pide a Alicia que le revele los valores para toda esa fila.
(c) Revelar submatriz: en esta opción, Roberto escoge al azar una de las 9 submatrices de 3x3 del tablero, y le pide a Alicia revelarla.
(d) Revelar posiciones iniciales: en esta opción, Roberto pide a Alicia revelar los valores correspondientes a todas las posiciones "llenas" en el sudoku original x.

3.- Alicia: revela lo solicitado por Roberto en el paso anterior.

4.- Roberto: revisa que los valores revelados por Alicia sean consistentes con la solución de un sudoku: si es una fila, columna o submatriz, todos sus valores deben ser distintos, del 1 al 9. Si Alicia debe revelar los valores para las posiciones "llenas" del sudoku original, entonces Roberto debe verificar que los valores revelados sean consistentes con una *permutación* de los valores reales del sudoku inicial x. Por ejemplo, si donde había un 1 en x ahora hay un 5, Roberto debe chequear que en toda otra celda donde había un 1 ahora hay un 5. También debe chequear que los valores revelados para todas las celdas que tenían valores distintos en x siguen siendo distintos (ver Figura 4b).

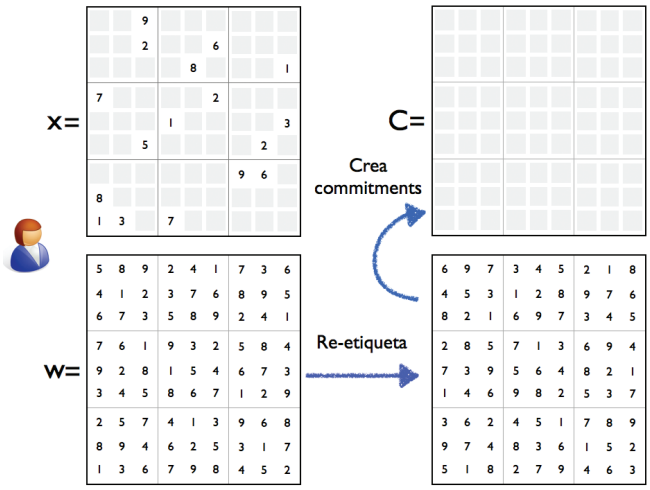


Figura 4a • Protocolo de Nula Divulgación para sudoku, paso 1.

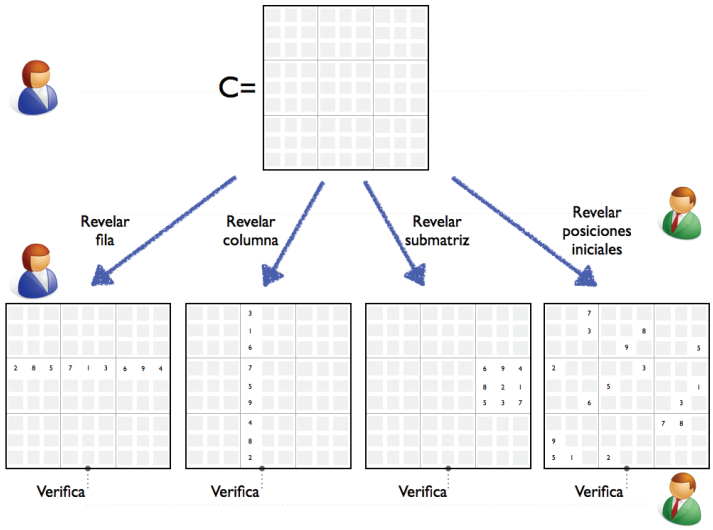


Figura 4b • Protocolo de Nula Divulgación para sudoku, pasos 2, 3 y 4.

5.- Roberto y Alicia: repiten todos los pasos anteriores n veces.

Primero notemos que hay un total de 28 posibles preguntas que Roberto puede hacer en el paso 2. Esto porque puede preguntar una fila específica (opción (a), 9 posibilidades, una por fila) o columna específica (opción (b), 9 posibilidades) o submatriz específica (9 posibilidades) o simplemente preguntar por los valores en las posiciones originales (1 posibilidad).

Si Alicia conoce una solución, puede almacenarla en forma permutada en los commitments, y siempre contestar exitosamente cualquier pregunta de Roberto en el paso 3. Así, luego de iterar n veces, Roberto queda convencido. Pero, ¿y si Alicia está engañando a Roberto y simplemente no existe una solución para el sudoku? Es fácil ver que ninguna selección de valores para los commitments puede lograr convencer ante todas y cada una de las 28 posibles preguntas de Roberto. Supongamos Alicia intenta demostrar a Roberto que un x sin solución realmente la tiene. Entonces, lo mejor que puede hacer es intentar adivinar la pregunta que Roberto le hará en el paso 2 y preparar commitments con valores que "funcionen" para dicha opción (por ejemplo si apuesta que Roberto escogerá

la opción (b), esto es preguntará por los valores de una columna específica, Alicia podría poner valores distintos en todas las columnas). Sin embargo, claramente esta estrategia no puede funcionar para una o más de las otras opciones y Alicia será descubierta (no podrá responder) si le preguntan por ellas. Una manera de verlo es la siguiente: si no hay solución, no importa qué valores Alicia ponga en los commitments, hay al menos una de las 28 preguntas específicas que requiere revelar valores de los commitments que son inconsistentes con una solución de sudoku válida. Con probabilidad al menos 1/28, Roberto escogerá dicha pregunta exponiendo a Alicia como mentirosa. Esto implica que, en principio, con probabilidad al menos 27/28 (aproximadamente 96.4%) Alicia podría contestar a Roberto correctamente en el paso 3. Pero, ¿no es eso es muy alto? Sí, pero es sólo para una iteración. Basta recordar que los pasos 1 a 3 se realizan varias veces, n veces, en forma independiente, usando permutaciones y commitments nuevos cada vez. Así, la probabilidad que Alicia engañe a Roberto en todas las n iteraciones es a lo más $(27/28)^n \approx (1/2)^{0{,}05n}$. Un cálculo simple nos muestra que si $n = 2500$ esta probabilidad se convierte en aproximadamente $2^{-125} \approx 1/10^{38}$, esto es, 0,...(37 ceros)...2. ¡Esta probabilidad es bajísima! De hecho, es menor que la probabilidad de ganarse cinco veces seguidas el premio mayor del Loto en Chile[4]. Si Alicia puede hacer esto, ¡sería millonaria y poco incentivo tendría para engañar a Roberto!

Generalizaciones

Recientemente [7] fue demostrado que el problema de decidir si una instancia de sudoku tiene solución es NP-Completo. En teoría, esto nos permite demostrar en Nula Divulgación que cualquier problema L en NP tiene solución, en dos pasos: (1) transformamos L en una instancia de sudoku, y (2) demostramos en Nula Divulgación que existe una solución para este sudoku usando el protocolo anterior. En otras palabras, para demostrar que un problema cualquiera (en NP) tiene solución sin revelarla, nos basta transformarlo matemáticamente en una instancia de sudoku, y luego jugar el rol de Alicia más arriba. ¡Quién dijo que la teoría era aburrida!

DEMOSTRACIÓN DE CONOCIMIENTO

Usando un protocolo de Nula Divulgación, Alicia puede entonces demostrarle a Roberto que el sudoku en cuestión tiene solución. ¿Puede entonces Roberto concluir que Alicia sabe la solución? No en general. Conceptualmente, una demostración de Nula Divulgación permite demostrar que una solución existe, pero nada más. Es posible que el demostrador pudiera realizarla sin necesariamente conocer una solución[5]. Sin embargo, hay muchas situaciones en las cuales esto no es suficiente, lo que importa es si el demostrador conoce una *solución* al problema. Por ejemplo, cuando un servidor desea autentificar a una persona, lo importante es verificar que ésta conoce la contraseña almacenada en el servidor. Al servidor le interesa verificar si el usuario la sabe, no si existe (pues siempre existe). En nuestro caso, quizás Roberto sólo desea resolver sudokus que Alicia ha resuelto, por lo que le gustaría tener certeza que ella sabe la solución. Efectivamente, el protocolo de la sección anterior sí garantiza que Alicia sabe la solución, lo cual demostraremos a continuación. Primero necesitamos preguntarnos qué significa "saber".

¿Cómo podemos definir matemáticamente que Alicia "conoce" o "sabe" la solución? Pese a que hemos llamado Alicia al demostrador, típicamente quienes realizan las demostraciones son programas, no personas. ¿Qué significa entonces que un programa *conozca* un valor o que una máquina *sabe* algo?

Interrogando al demostrador

La respuesta a esta interrogante es simple pues evita contestar la pregunta respecto a qué es conocer algo. En vez de eso, decimos que una demostración interactiva garantiza que el demostrador "sabe" una solución w para un problema x dado, si podemos "extraer" la solución del demostrador, simplemente haciéndole las preguntas adecuadas. En otras palabras, si un demostrador P^* convence a V con alta probabilidad, entonces existe un algoritmo extractor E que calcula el valor w sólo a partir de interactuar con P^*. Pero, ¿qué impide que cualquier verificador pueda usar el algoritmo E y robarle la solución al demostrador P^*? Después de todo, queremos un protocolo que sea una demostración de conocimiento y –al mismo tiempo– de Nula Divulgación. La respuesta a esta aparente contradicción es clara si pensamos qué significa exactamente tener un demostrador que nos convence. Un demostrador exitoso P^* es simplemente un programa ejecutado sobre una entrada x, el cual puede usar una cierta aleatoriedad (lanzar monedas o bits aleatorios que usa durante su ejecución). El algoritmo E puede entonces invocar o ejecutar a P^* varias veces sobre la misma entrada x y sobre la misma aleatoriedad r. Eso es algo que ningún verificador típicamente puede hacer en una ejecución real. Un demostrador no permitirá al verificador dictar la aleatoriedad a usar por el demostrador; tampoco ejecutará el mismo protocolo más de una vez con el mismo verificador.

[4] En el juego de Loto en Chile, el premio mayor lo obtiene quien acierta a los números en seis bolitas seleccionadas de un total de 41. La probabilidad de adivinar la combinación ganadora es de 1/4.496.388.

[5] Esto es en general para un protocolo de Nula Divulgación arbitrario, y no es el caso para el protocolo anterior para sudoku, el cual sí garantiza que Alicia sabe la solución, como veremos más adelante.

Protocolo de Nula Divulgación y demostración de conocimiento para sudoku

Veremos ahora que el protocolo dado en la sección anterior es también una demostración de conocimiento. El extractor E en cuestión es simplemente un programa que actúa como Roberto pero que en el paso 2 pregunta primero por la opción (a), digamos la fila 1, y luego de recibir una respuesta correcta, *ejecuta P de nuevo, desde el comienzo con la misma aleatoriedad*, pero ahora le pregunta en el paso 2 por la fila 2 en la opción (a) (notemos que los commitments son iguales pues la aleatoriedad es la misma.) El algoritmo E puede repetir este proceso de "ejecutar de nuevo" varias veces, recuperando las respuestas para todas las opciones posibles: puede preguntar no sólo por todas las filas, sino por todas las columnas, todas las submatrices y todos los valores en las posiciones originales "llenas" del sudoku. Al obtener las respuestas correctas a *todas*[6] las preguntas posibles de Roberto en el paso 2, el extractor E puede deducir cuál fue la permutación usada por el demostrador y, a partir de los valores revelados de todas las filas (o columnas) obtener la solución w del sudoku original x. Esto nos demuestra que, si Alicia puede convencer a Roberto usando el protocolo de Nula Divulgación descrito anteriormente, entonces Alicia debe saber la solución. De hecho, incluso si ella no lo supiera (por ejemplo, si para convencer a Roberto usara algún algoritmo extraño, ultra sofisticado, que no necesitase la solución para funcionar), entonces ¡ella misma podría ejecutar E y "extraer de sí misma" la solución!

APLICACIÓN A VOTACIÓN ELECTRÓNICA

Concluimos el artículo comentando que las aplicaciones de los protocolos de Nula Divulgación son inmensas y variadas. Una en particular es su uso en votación electrónica. Efectivamente, es posible diseñar sistemas de votación electrónica en los cuales el voto de una persona es encriptado usando un esquema de encriptación de clave pública. Supongamos que la votación es de tipo plebiscito: el voto puede sólo ser 1 (a favor) o 0 (en contra). En estos sistemas el esquema de encriptación usado tiene una propiedad muy útil, es *homomórfico:* la multiplicación de todos los textos cifrados permite obtener un texto cifrado con la suma de los votos. Luego, para saber si se aprueba o no el plebiscito basta desencriptar el texto cifrado así calculado. Si el valor obtenido es un número mayor que la mitad de los votos, entonces el resultado final es a favor; si no, es en contra. Fácil, ¿no?

Hay un detalle crítico, sin embargo, para que el sistema anterior funcione. Si un votante malicioso puede encriptar un valor distinto a 0 ó 1 en su voto, digamos 100 entonces, su voto afectará el total en forma inapropiada: el efecto de su voto es equivalente a 100 votos a favor, algo claramente inaceptable. Para tener la certeza que el voto encriptado es sólo 0 ó 1 *sin revelar el voto específico* podemos utilizar demostraciones de Nula Divulgación. En principio, el problema que queremos resolver es demostrar que una cierta encriptación tiene como solución (su desencriptación) es 0 ó 1 y nada más, algo que se puede demostrar está en NP. Dado que el problema de determinar si un sudoku tiene solución es NP-Completo podemos transformar el problema anterior en una instancia de sudoku, demostrar en Nula Divulgación que conocemos una solución al puzle, y ello inmediatamente nos daría una demostración para el problema original, esto es, que el voto dado es válido. Más aún, dado que dicho protocolo es una demostración de conocimiento, entonces podemos tener la certeza que el votante sabe su voto, y por ende, es suyo (y no es simplemente una copia del texto cifrado usado por otra persona). Vale notar que, por cierto, ésta no es la única manera de dar un protocolo de Nula Divulgación para este problema, hay otros más eficientes e incluso no interactivos (por ejemplo [4]), pero ése es tema de otro artículo.

Es difícil pensar en una aplicación más práctica y con mayor impacto social que la votación electrónica. ¿Quién hubiera pensando que la Teoría de la Computación nos daría herramientas para fundar nuestra democracia en el mundo digital? BITS

Referencias

[1] T. H. Cormen, C. E. Leiserson, R. L. Rivest and C. Stein. Introduction to Algorithms (3. ed.). MIT Press, 2009.

[2] R.A. Fisher. Mathematics of a lady tasting tea. In J. R. Newman, editor, The World of Mathematics, Statistics and the Design of Experiments, volume 3, pages 1512 - 1521. Simon & Schuster, New York, 1956.

[3] Ronen Gradwohl, Moni Naor, Benny Pinkas, and Guy N. Rothblum. Cryptographic and physical zero-knowledge proof systems for solutions of sudoku puzzles. Theory Comput. Syst., 44(2):245 - 268, 2009.

[4] J. Groth. Non-interactive zero-knowledge arguments for voting. In proceedings of ACNS 05, LNCS series, pages 467 - 482. Springer-Verlag, 2005.

[5] Jonathan Katz and Yehuda Lindell. Introduction to modern cryptography. Chapman & Hall/CRC Cryptography and Network Security. Chapman & Hall/CRC, Boca Raton, FL, 2008.

[6] Mike Rosulek. Zero-knowledge proofs, with applications to sudoku and where's waldo? Presentación educacional, University of Montana, 2008.

[7] T. Seta T. Yato. Complexity and completeness of finding another solution and its application to puzzles. IEICE Trans. Fundam. Electron. Commun. Computer Science, E86-A(5):1052 - 1060, 2003.

[6] En estricto rigor, no es necesario obtener las respuestas para todas las preguntas, basta preguntar por todas las filas, o todas las columnas, o todas las submatrices para obtener todos los valores de la matriz.