UNIVERSIDAD DE LA LAGUNA

Diseños de protocolos criptográficos: nuevas propuestas basadas en grafos

Autor: Hernández Goya, Candelaria

Director: Pino Caballero Gil

Departamento de Estadística, Investigación Operativa y Computación

UNIVERSIDAD DE LA LAGUNA
DEPARTAMENTO DE ESTADÍSTICA
INVESTIGACIÓN OPERATIVA Y COMPUTACIÓN

DISEÑO DE PROTOCOLOS CRIPTOGRÁFICOS: NUEVAS PROPUESTAS BASADAS EN GRAFOS

por
Candelaria Hernández Goya

MEMORIA PARA OPTAR AL GRADO DE
DOCTOR EN CIENCIAS MATEMÁTICAS
REALIZADA BAJO LA DIRECCIÓN DE LA
DRA. PINO CABALLERO GIL
UNIVERSIDAD DE LA LAGUNA
LA LAGUNA, TENERIFE
JULIO 2003

DRA. DÑA. PINO CABALLERO GIL, PROFESORA TITULAR DEL DEPARTAMENTO DE ESTADÍSTICA, INVESTIGACIÓN OPERATIVA Y COMPUTACIÓN DE LA UNIVERSIDAD DE LA LAGUNA

CERTIFICO:

Que la presente memoria titulada, **Diseño de Protocolos Criptográficos: Nuevas Propuestas Basadas en Grafos** ha sido realizada bajo mi dirección por la Licenciada M. Candelaria Hernández Goya, y constituye su Tesis para optar al grado de Doctora en Matemáticas.

Y para que conste, en cumplimiento de la legislación vigente y a los efectos que haya lugar, firmo el presente en

La Laguna, a 5 de Mayo de 2003

Fdo.: Pino Caballero Gil

A mis padres

Índice

Índice de figuras

Agradecimientos

En primer lugar me gustaría aprovechar esta irrepetible oportunidad para agradecer a la Dra. Pino Caballero Gil toda la ayuda prestada y poner de manifiesto lo mucho que ha significado para mí poder trabajar con ella. No sólo tengo que agradecerle el haberse prestado a dirigir esta tesis, sino también su continuo apoyo y esfuerzo. Además debo decir que ha sido un privilegio poder compartir su contagioso afán de superación. Sin duda es una de las personas de las que más he aprendido y espero poder seguir haciéndolo.

El aliento y los consejos que la Dra. Amparo Fúster Sabater me ha transmitido en múltiples ocasiones me han resultado de inestimable valía en muchos momentos difíciles.

También debo agradecer la ayuda, atención, consejos y sobre todo el valioso tiempo que el Dr. Josep Domingo i Ferrer me brindó durante mi estancia en Tarragona así como en posteriores y gratas coincidencias.

Termino agradeciendo a mis hermanos y mis padres el cariño y la comprensión que me han brindado, sobre todo durante este periodo en que todas mis energías y tiempo los he dedicado a la realización del trabajo presentado en esta memoria. Espero poder compensarles algún día.

La Laguna, Tenerife — Candelaria Hernández Goya

Mayo, 2003

Resumen

El objetivo primordial del presente trabajo es destacar el papel que la Teoría de Grafos puede jugar en el entorno de la criptografía moderna. Con ese fin se propone el uso de dicha disciplina como prontuario de problemas base, y se introduce una nueva metodología para el diseño de protocolos criptográficos.

Entre los protocolos estudiados destacan los de transferencias inconscientes y compromisos de bits por ser considerados primitivas esenciales para el diseño de esquemas criptográficos más complejos. Asimismo se proponen nuevas alternativas para los prácticos protocolos bipartitos de firmas de contratos y lanzamientos de monedas. En todos los casos mencionados los algoritmos diseñados se basan en problemas difíciles de grafos, como el del isomorfismo, el del circuito hamiltoniano o el del conjunto independiente.

Especial atención se presta al problema de la identificación fuerte, proponiéndose varias soluciones que combinan elementos tales como las demostraciones de conocimiento nulo, las contraseñas de un sólo uso, la criptografía de clave pública, y distintos problemas difíciles de la Teoría de Grafos.

Un segundo objetivo de la presente memoria consiste en investigar la posibilidad de diseñar nuevos protocolos criptográficos a partir de problemas combinatorios que sean difíciles en media. Concretamente se proponen una demostración de conocimiento nulo y un algoritmo multipartito de reparto de secretos, ambos basados en un problema clasificado como difícil según el análisis del caso medio, el conocido como Problema Distribucional de la Representación de Matrices.

Tras la descripción de cada una de las diferentes alternativas propuestas se analizan en detalle cuestiones tales como las hipótesis necesarias para garantizar la seguridad, la especificación formal de los pasos a desarrollar, y la comprobación de la correctitud de los mismos, prestando especial atención al comportamiento ante ciertos intentos de fraude. Asimismo se proporcionan estudios comparativos entre las diferentes propuestas en términos de parámetros tales como el ancho de banda requerido y la complejidad de las operaciones a realizar. Por último se incorporan algunas de las implementaciones realizadas.

Prólogo

Si se echara una mirada hacia atrás en el tiempo y se intentara localizar las primeras referencias dedicadas al estudio de los protocolos criptográficos se llegaría, probablemente, hasta finales de los años setenta. Sin embargo, las bases teóricas de diseño y aplicación de los protocolos que se tratan en esta memoria son tan antiguas como los comienzos de la Inteligencia Artificial o la Teoría de Juegos. Sirva de muestra "el juego de la imitación"definido por Alan Turing en 1950 [Tur00], descrito brevemente a continuación.

En dicho juego existen tres participantes, una mujer ($\mathcal{A}$), un hombre ($\mathcal{C}$) y un interrogador ($\mathcal{B}$), pudiendo éste último pertenecer a cualquiera de los dos sexos. Durante el desarrollo del juego se exige que el interrogador $\mathcal{B}$ se encuentre en una habitación distinta de la ocupada por $\mathcal{A}$ y $\mathcal{C}$. En la descripción del juego, Turing diferencia los objetivos de cada uno de los participantes, siendo algunos de ellos contrapuestos entre sí: el del interrogador $\mathcal{B}$ consiste en identificar cuál es la mujer y cuál el hombre, el de $\mathcal{C}$ es intentar que $\mathcal{B}$ se equivoque en la identificación, mientras que $\mathcal{A}$ intenta ayudar al interrogador en su tarea. Para conseguir su objetivo, $\mathcal{B}$ puede realizar preguntas a ambos participantes $\mathcal{A}$ y $\mathcal{C}$.

Se puede plantear una analogía entre el juego anteriormente descrito y un protocolo de identificación. En primer lugar, un protocolo de identificación se desarrolla normalmente en un entorno distribuido donde se supone la existencia de tres participantes, siendo uno de ellos ($\mathcal{C}$) deshonesto. Además, el protocolo en cuestión utiliza la técnica de reto-respuesta ya que el verificador $\mathcal{B}$ realiza una serie de cuestiones que deben ser respondidas por la participante legítima $\mathcal{A}$ y su adversario $\mathcal{C}$.

Tal y como se deduce del título del artículo [Tur00] ("Can a machine think?") la finalidad del autor es analizar la cuestión de si las máquinas son capaces de pensar, y para ello en el trabajo sustituye dicha pregunta por la siguiente "What will happen when a machine takes the part of $\mathcal{A}$ in this game?". Esta pregunta se puede equiparar a la que habitualmente se realiza en la demostración formal de la corrección de los

protocolos de Conocimiento Nulo. En dicha situación la pregunta concreta es: ¿cómo se desarrollaría el protocolo si la participante $\mathcal{A}$ fuera sustituida por un simulador? La respuesta a esta cuestión se estudia habitualmente mediante el denominado paradigma de la simulación.

A continuación se describe brevemente la organización de la presente memoria.

En primer lugar se introducen los conceptos básicos y herramientas necesarias en el capítulo 1, con el objeto de que la memoria sea autocontenida.

Posteriormente, en el segundo capítulo, se estudian en detalle un conjunto de protocolos cuyo factor común es su pertenencia a la categoría de protocolos bipartitos. Entre estos destacan especialmente, dada su consideración como primitivas de diseño en el campo de los protocolos criptográficos, la Transferencia Inconsciente y el Compromiso de Bits, para los cuales se presentan nuevas aportaciones basadas en grafos.

En el siguiente capítulo se analiza en profundidad una familia de protocolos bipartitos con aplicaciones en identificación y en verificación de la honestidad de participantes, las conocidas como Demostraciones de Conocimiento, incluyéndose también en este caso la descripción y análisis de nuevos algoritmos y esquemas de identificación basados en grafos.

En el cuarto capítulo se proponen dos nuevos protocolos criptográficos, uno bipartito y otro multipartito, basados ambos en un mismo problema combinatorio clasificado como difícil según el análisis del caso medio, el conocido como Problema Distribucional de la Representación de Matrices.

Finalmente, en el capítulo 5 se destacan aquellos resultados más significativos obtenidos en la memoria, y se mencionan algunas líneas de investigación abiertas, que han ido surgiendo durante el desarrollo del presente trabajo.

La Laguna, Tenerife — Candelaria Hernández Goya
Mayo, 2003

Capítulo 1

Preliminares

La criptografía ha sido sin duda una de las disciplinas científicas más dinámicas y de mayor crecimiento en los últimos 30 años. La necesidad de establecer y mantener comunicaciones seguras usando canales de comunicación cada vez menos confiables ha conseguido que este área se convierta en algo mucho más complejo que un conjunto de esquemas de cifrados. De hecho, en muchos casos, un esquema de cifrado es solamente un elemento más de un criptosistema formado por innumerables capas o "compartimentos estancos", teniendo todos ellos la función de garantizar la seguridad de la información asociada a ese nivel. Con este propósito han ido apareciendo gran cantidad de trabajos en los que se proponen, estudian y comparan multitud de protocolos criptográficos, siendo cada uno de ellos diseñados "ad hoc" para resolver problemas muy concretos.

1.1. Cuestiones de Protocolos

Al hacer un recorrido por las diferentes descripciones de protocolos que se han propuesto, la primera conclusión extraída es probablemente que el diseño de cada uno de ellos viene justificado por la necesidad de trasladar muchas de las situaciones de la vida cotidiana al mundo digital. La segunda conclusión es quizás, la gran dificultad que supone el resolver dichas situaciones en este nuevo entorno manteniendo satisfechas

las exigencias de seguridad.

1.1.1. Conceptos Básicos y Notación

El concepto de *Protocolo Criptográfico* es uno de los que más ha evolucionado dentro de la Criptografía. Pruebas de ello son las diferentes acepciones que han aparecido en la bibliografía. Una de las primeras aproximaciones a este término aparece en [Mer83], donde Merrit define protocolo criptográfico como un algoritmo distribuido de comunicaciones, cuyo diseño implica una serie de transacciones, y que es seguro siempre y cuando se mantengan los requisitos de seguridad de las comunicaciones.

A partir de 1986 comienzan a aparecer definiciones más elaboradas y formales que la anterior, como por ejemplo la que se incluye a continuación, aportada por Goldreich, Micali y Wigderson en [GMW87].

Definición 1.1.1. Protocolo según [GMW87]
Un protocolo es una función multiargumento f con las siguientes propiedades:

1. *Corrección*: El protocolo permite a cada participante obtener el valor de la función sobre argumentos repartidos entre todas las partes. Es decir, la salida de la función ($f(x_1, x_2, \ldots, x_n)$) es pública.

2. *Privacidad*: Cualquier información calculada eficientemente después de la participación en el protocolo, puede calcularse también a partir de su entrada y salida locales. Es decir, no se vierte información sensible.

Esta definición es modificada posteriormente en [Gol01b], donde se adapta explícitamente para el caso particular de protocolos bipartitos. Aquí se presenta una extensión para el caso general de forma que la función que modela el protocolo no es sólo multiargumento, como sucedía con la definición anterior, sino que además es multidimensional respecto a la imagen, es decir, la salida que obtiene cada uno de los participantes no tiene por qué coincidir.

Definición 1.1.2. Protocolo según [Gol01b]

Un protocolo con n participantes queda definido especificando un proceso que puede ser aleatorio. Dicho proceso asocia a un conjunto de n entradas (una por cada participante) n salidas, por lo que puede ser representado mediante una función multiargumento y multidimensional, denotada por f. Esta función está definida sobre el conjunto de todas las posibles n-uplas o vectores de longitud n de cadenas binarias que determinan la entrada de cada participante al comenzar el protocolo. A cada n-upla $X = (x_1, x_2, \ldots, x_n)$, la función mencionada asocia como imagen una nueva n-upla $f(X) = (f_1(X), f_2(X), \ldots, f_n(X))$ formada por las cadenas binarias que determinan la salida obtenida por cada una de las partes una vez finalizado el protocolo.

$$f : \{0,1\}^* \times \{0,1\}^* \times \overbrace{\cdots}^{n} \times \{0,1\}^* \mapsto \{0,1\}^* \times \{0,1\}^* \times \overbrace{\cdots}^{n} \times \{0,1\}^*$$

$$X = (x_1, x_2, \ldots, x_n) \hookrightarrow f(x_1, x_2, \ldots, x_n) = f(X) = (f_1(X), f_2(X), \ldots, f_n(X))$$

En las secciones dedicadas a cada uno de los protocolos tratados se puede obtener la definición concreta del mismo usando esta aproximación. Por otra parte, la definición general de protocolo por la que se ha optado en esta memoria se basa en el concepto de algoritmo y en la definición de protocolo siguiente, dada por Schneier [Sch93].

Definición 1.1.3. Protocolo criptográfico

Los protocolos criptográficos son algoritmos utilizados por un conjunto de participantes con una meta común, en un entorno distribuido inseguro de desconfianza mutua y respecto a terceras partes, y tales que verifican los dos requisitos siguientes:

- *Principio de Kerckhoffs*: Todos los participantes en un protocolo deben conocer su descripción y sus posibles resultados, y deben estar de acuerdo en su utilización.
- *Tolerancia a fallos*: Debe estar previsto el comportamiento de los participantes ante cualquier posible situación.

Otras definiciones propuestas para el concepto de protocolos establecen un vínculo entre la Teoría de Juegos y la Criptografía. Dicha conexión [Gol01b], [FW93], [KR02] permite analizar la seguridad de un protocolo estudiando las estrategias que un participante malicioso (ver sección 1.1.5) puede desarrollar con el objeto de deteriorar el protocolo. La primeras referencias a la relación entre las dos disciplinas anteriores estudian la implementación de protocolos que permiten la implementación de juegos de cartas en entornos distribuidos [Cré86a], [Cré86b], [CK93]. En [DST00] se presenta una solución al problema de la Teoría de Juegos consistente en determinar la existencia de juegos bipersonales en los que las ganancias obtenidas por los participantes sean comparables a las obtenidas en el caso de usar una tercera parte de confianza (ver sección 1.1.2) que coordine sus estrategias. Para obtener una solución definen un protocolo criptográfico probabilista e iterativo que usa una demostración de conocimiento nulo.

A menudo en los modelos formales se añaden suposiciones de limitación referentes a la capacidad de cálculo de los participantes. También normalmente el diseño del algoritmo se basa en intercambios sucesivos de mensajes, implicando esto la necesidad de compartir un canal fluido de comunicaciones entre los integrantes del protocolo. Más adelante se tratará en detalle el tema de los modelos formales.

Si se profundiza en la visión de los protocolos como algoritmos distribuidos es necesario estudiar la eficiencia de los mismos. En este caso, la eficiencia suele estudiarse en función de la complejidad de los cálculos a realizar por cada una de las partes, la posibilidad de realizar algunos cálculos off-line, el número de mensajes intercambiados y el ancho de banda necesario durante el desarrollo de los mismos, [MOV96]. La elección de estos parámetros de comparación se debe a las habituales restricciones de memoria y de flujo de información existentes en los entornos distribuidos.

Otra cuestión a estudiar siguiendo el punto de vista de la Teoría de la Computación es el modelo computacional subyacente. Cada participante se identifica con una

máquina de Turing interactiva (ver apartados 1.2 y 3.1) con cinta de entrada privada de sólo lectura y cinta de salida de sólo escritura, de manera que todas las máquinas comparten una cinta de entrada común de sólo lectura y una cinta de salida de sólo escritura. Según este planteamiento todo protocolo criptográfico especifica un programa para cada máquina, de forma que al final de la ejecución, cada máquina tendrá en su cinta de salida su mensaje de salida particular, y todos los programas trabajan coordinados. Además se suele suponer que son máquinas de Turing probabilistas con potencia de cálculo limitada polinomialmente en algunos casos.

En las descripciones de protocolos bipartitos incluidas en esta memoria se identifica a los participantes por los nombres de Alicia y Bernardo (Alice y Bob en la bibliografía en inglés), aunque normalmente se usa la notación abreviada $\mathcal{A}$ y $\mathcal{B}$ respectivamente.

Se debe aclarar también que en todos los análisis de complejidad incluidos se ha supuesto que la generación de números aleatorios se puede desarrollar en tiempo constante.

1.1.2. Clasificación

El ámbito de los protocolos criptográficos es lo suficientemente amplio como para hacer necesario el establecimiento de criterios de clasificación de los mismos. Así, si se atiende al número de participantes que intervienen se puede distinguir entre protocolos *bipartitos* y *multipartitos*. En los primeros, el número de partes directamente implicadas se limita a dos, mientras que en los otros sólo existe la restricción sobre la finitud de dicho número.

Otra clasificación se puede realizar en función de si existe un intercambio recíproco de información entre los participantes o si por el contrario, están establecidas las figuras de emisor y receptor. Siguiendo este razonamiento, realizamos la siguiente diferenciación:

- si durante el desarrollo del algoritmo un usuario tiene el papel de emisor (probador en las Demostraciones de Conocimiento) y su contrario intenta obtener la entrada del primero o una transformación de ésta, se trata de un *protocolo unilateral* o *unidireccional*. Esto es lo que sucede en protocolos tales como la transferencia inconsciente, las demostraciones de conocimiento nulo, etc...

- si por el contrario, todos los participantes juegan idénticos papeles siendo emisor y receptor a la vez se habla de *protocolo multilateral*, *multidireccional* o *de intercambio mutuo* (bilateral o bidireccional en el caso de los bipartitos). Esta situación se produce en protocolos tales como la firma de contratos, el lanzamiento de monedas, etc...

En el caso de los protocolos unilaterales la definición 1.1.2 a través de la función descrita resulta más sencilla puesto que podemos suprimir la salida asociada al emisor.

Al igual que sucede en el entorno de los algoritmos, también en el caso de los protocolos se puede distinguir entre protocolos *deterministas* (elecciones electorales, reparto de secretos, etc...) o *probabilistas* (demostraciones de conocimiento nulo, transferencia inconsciente, etc...). En este último conjunto se incluyen aquellos protocolos en los que la intervención explícita del azar modifica el desarrollo del protocolo, pudiendo incluso darse el caso de que el mismo falle, con una probabilidad insignificante.

Una familia de protocolos sobre los que existe cierta reticencia son los *protocolos arbitrados*. En éstos existe un ente externo (no toma parte directamente en el desarrollo del protocolo), cuya finalidad es posibilitar el intercambio de información certificada entre los participantes. Es lo que normalmente se denomina Tercera Parte de Confianza (TTP, Trusted Third Party). En el diseño de protocolos criptográficos se suele evitar su presencia puesto que conlleva tener que depositar la confianza de los usuarios en una entidad externa, además de poderse convertir en cuello de botella tanto para las comunicaciones como para las computaciones a realizar durante

el desarrollo del protocolo. Sin embargo, se ha demostrado formalmente la necesidad de estos entes no sólo para funciones relacionadas con la firma digital sino en otros contextos aún más particulares como por ejemplo en el trabajo [EY80b] que demuestra la no existencia de protocolos deterministas que no contemplen una TTP para la firma de contratos en entornos distribuidos.

Uno de los criterios de clasificación más significativos utilizado en esta memoria es el que distingue los protocolos criptográficos de propósito general (primitivas criptográficas) de los de propósito particular. En el capítulo 2 se presta especial atención al primer grupo por ser las primitivas criptográficas piedras angulares de la materia tratada en este trabajo.

1.1.3. Antecedentes

Uno de los primeros protocolos propuestos fue el diseñado por Needham y Schroeder [NS78] en 1978. Su finalidad consiste en que dos participantes, con la ayuda de una tercera parte de confianza, se intercambien una clave secreta para conseguir establecer un canal de comunicaciones seguro. A partir de ese momento comienzan a surgir una serie de trabajos en los que se proponen nuevos protocolos con finalidades muy diferentes como es el caso de [SRA78] donde se especifica el primer protocolo que permite jugar al póquer en un entorno distribuido. Para esta situación se han planteado alternativas de solución basadas en técnicas criptográficas diferentes, algunas de ellas se pueden encontrar en [FM85], [Cré86a], [KKOT90]. En [Blu82a] se propone un protocolo que permite a dos usuarios obtener el resultado de la tirada de una moneda a través de un canal de comunicaciones. En la descripción del anterior protocolo aparece por primera vez un compromiso de bits. El mismo año, Even en [Eve82] describe una solución determinista para el problema de la firma de contratos digitales en la que interviene una TTP. Para evitar el uso de estas entidades el

mismo autor ([EGL82]) plantea una solución probabilista esta vez basada en el concepto de transferencia inconsciente (sección 2.1). Es necesario destacar también el trabajo [Rab81] en el que se introduce la noción de transferencia inconsciente, uno de los cimientos en el diseño de protocolos generales a la que se le presta especial atención en el capítulo 2 de esta memoria. Otros trabajos en los que se intentan establecer criterios para el diseño y análisis de los protocolos son por ejemplo, [MLM82], [BKP86], [BO85] en los que se comienza a formalizar la noción de protocolo y los criterios que determinan la seguridad de los mismos (sección 1.1.6). Concretamente en [BO85] se establecen condiciones que caracterizan a los protocolos bipartitos en los que es posible la la autenticación de los mensajes intercambiados entre los participantes, mientras que en el artículo [BKP86], se incluye una descripción detallada de un modelo de computación teórico sobre el que se da una definición formal de la noción de protocolo atendiendo a la interacción entre los participantes. Finalmente se publican otros trabajos en los que se plantean fallos particulares de algunas de esas propuestas, como es el caso de [Cop86], [DS81], [BAN90], [DGB87].

En 1983 aparece uno de los primeros trabajos [Mer83] en el que se trata el tema de los protocolos criptográficos como conjunto de algoritmos. En dicho trabajo, además de relacionar los conceptos de algoritmo y protocolo criptográfico, se proponen también dos aspectos a vigilar en la verificación de la seguridad de cualquier protocolo criptográfico. Dichos aspectos son la identificación explícita de las hipótesis criptográficas usadas, y la determinación de que cualquier ataque llevado a cabo con éxito sobre el protocolo en cuestión requiere la violación de alguna de esas hipótesis. Los criterios anteriores son fundamentales en el diseño de protocolos probablemente seguros. Este grupo se caracteriza por el hecho de que la seguridad de los mismos descansa sobre hipótesis relacionadas con la Complejidad Computacional. Por lo general, esto está motivado por el uso de problemas base pertenecientes a las clases de complejidad NP y $NP - completa$ (ver sección 1.2).

En el trabajo [GMW87], aparte de definir el concepto de protocolo (definición 1.1.1) los autores aportan un teorema que establece la existencia de protocolos correctos y tolerantes a fallos destinados al cómputo de una determinada función para cualquier problema descrito a través de la máquina de Turing asociada a la función mencionada.

Más recientemente, en [Cré97] se presentan propuestas para las primitivas de transferencia inconsciente y compromiso de bits (tratadas extensamente en el capítulo 2 de esta memoria) usando en su diseño un canal simétrico binario con cierto error asociado mejorando de esta forma alternativas anteriores en las que la complejidad de las comunicaciones (ver 1.2.4) era mayor. En la línea de definir criterios de seguridad para protocolos multipartitos surge el trabajo [Can00], en el que además se incluye un breve recorrido por las diferentes definiciones de seguridad que se han sugerido en trabajos previos.

Una gran parte de los criptosistemas actuales hacen uso de la problemas intratables de la Teoría de Números [Bac89] como así atestiguan los innumerables trabajos que usan por ejemplo el problema del logaritmo discreto (ver 1.3.2) o el problema de la factorización. En el desarrollo del presente trabajo de investigación se propondrán diferentes problemas combinatorios para ser utilizados como base, siendo la mayoría de ellos pertenecientes a la Teoría de Grafos, incluyendo el capítulo 4 alguna otra herramienta combinatoria. Con el fin de ilustrar la riqueza que el uso de las técnicas combinatorias proporciona al diseño de herramientas criptográficas, seguidamente se comentan brevemente algunas referencias en las que se ha usado la Teoría de Grafos en Criptografía. Una curiosa referencia es [FK94] donde se hace un recorrido por diferentes criptosistemas y protocolos criptográficos con la intención de introducir diferentes herramientas combinatorias en el currículo educativo. En el campo del control de accesos y más concretamente en las soluciones que la criptografía visual propone para el reparto de secretos [ABSS96] es común representar las estructuras de acceso (conjunto

de participantes legítimos con la información suficiente para acceder a la información protegida) utilizando grafos no dirigidos. También en el campo de las funciones hash surgen alternativas relacionadas con esta herramienta combinatoria. En particular [Zem91] propone una función hash específicamente diseñada para firmar textos de longitudes grandes basándose su construcción en propiedades de los grafos de Cayley. En [BM94] se usan los grafos acíclicos para modelar un criptosistema asimétrico general y la combinación del mismo con una función hash. Esta construcción le permite formalizar el concepto de esquemas de firmas. En la sección 1.3 se introducen otras referencias en las que se utilizan grafos para realizar construcciones similares a las utilizadas en la generación de instancias para las implementaciones desarrolladas en la presente tesis. En el apartado de protocolos bipartitos es imprescindible citar el trabajo [GMR85] (por la relación directa con el capítulo 3) en el que sientan las bases de las demostraciones de conocimiento nulo y se dan ejemplos de las mismas basados en problemas tales como el isomorfismo de grafos, su complementario y el problema de la 3-coloración.

1.1.4. Propiedades

Aunque cada protocolo se diseña para resolver una situación particular, debe garantizarse el cumplimiento de ciertas propiedades generales [MOV96] que se irán describiendo a continuación. En la figura 1.1, se recogen cuestiones fundamentales que deben ser garantizadas antes y/o durante y/o después de la ejecución, dependiendo en cada caso de la finalidad del protocolo en cuestión.

Sin embargo, hay cuatro de los objetivos mencionados en la figura 1.1 que debido a su importancia engloban en cierta medida al resto. Estos son la *privacidad/confidencialidad*, la *integridad*, la *autenticación* y el *no repudio*. Esto queda reflejado en el hecho de que en el estándar ISO 7498-2 se especifica una descripción general de dichos objetivos, que allí se denominan servicios de seguridad OSI.

Privacidad/ Confidencialidad:	Mantener la información secreta salvo para los individuos o sistemas autorizados
Integridad:	Garantizar que la información no ha sido alterada por medios no autorizados o desconocidos
No repudio:	Prevenir el rechazo de compromisos o acciones previas
Identificación:	Corroborar la identidad de una entidad
Autenticación:	Corroborar la fuente de información
Autorización:	Traspasar a otra entidad permiso oficial para hacer algo
Validación:	Aportar medios que permitan autorizar en tiempo real el uso o manipulación de información o recursos
Control de accesos:	Restringir el acceso a recursos de entidades con los privilegios adecuados
Certificación:	Respaldar la información mediante una entidad confiable
Registro de tiempo:	Almacenar el tiempo de creación o existencia de la información
Atestiguar:	Verificar la creación o existencia de la información por una entidad diferente a la entidad creadora
Recibo:	Avisar de que la información ha sido recibida
Confirmación	Avisar de que los servicios han sido proporcionados
Propietario:	Proporcionar un medio por el que se le entrega a una entidad los derechos legales para usar o transferir un recurso
Anonimato:	Esconder la identidad de una entidad que participa en un determinado proceso
Revocación:	Retirar la certificación o autorización

Figura 1.1: Objetivos generales de los protocolos

Garantizar la privacidad de la información transferida durante el desarrollo de

cualquier comunicación es uno de los principales retos con los que se enfrenta cualquier protocolo. Además, la importancia de la misma es tan manifiesta que se recoge explícitamente en la propia Declaración de los Derechos Humanos y en la definición de protocolo criptográfico, así como en la siguiente definición que se usará en el resto de la memoria para cada especificación de nuevas propuestas.

Definición 1.1.4. Protocolo Privado
Se dice que un protocolo criptográfico es *privado*, o preserva la privacidad, si la información que obtiene un participante deshonesto al intervenir en el desarrollo del protocolo puede ser obtenida por él mismo sin necesidad de participar.

La integridad de la información requiere también especial atención, sobre todo debido a que puede ser incluso más perjudicial recibir de una supuesta fuente confiable una información que ha sido alterada, que no recibir ninguna. Esta característica debe garantizarse en todas las etapas que conlleva el diseño de un protocolo.

La autenticación es uno de los principales paradigmas en criptografía. Normalmente dicha propiedad está vinculada a la integridad ya que es frecuente la verificación de ambos objetivos en dos facetas, una relacionada con la autenticidad de la fuente emisora del mensaje y otra con la integridad del propio mensaje. Un ejemplo distinto de protocolo que tiene como una de sus principales aplicaciones la autenticación de la fuente es la identificación mediante demostraciones de conocimiento nulo, estudiadas extensamente en el Capítulo 3.

El no repudio por parte de una fuente de información ya emitida es objetivo también de protocolos particulares como la firma de contratos, por ejemplo. En la mayoría de casos dicha propiedad se garantiza mediante el uso de firmas digitales.

A las cuatro importantes propiedades anteriores se debe sumar el anonimato, dado que en muchos casos el mantener en secreto la identidad del interlocutor o usuario es crucial. Una situación que refleja la importancia de esta propiedad es

la recomendación de mantener separada la identidad del titular de una tarjeta de crédito de los movimientos realizados por la misma, recomendación que se convierte en requerimiento cuando se diseñan protocolos de dinero digital.

Una vez establecidos el escenario y algunos de los objetivos que deben satisfacer los protocolos criptográficos, se describe a continuación la propiedad de corrección por ser, al igual que sucede con la privacidad, de insoslayable verificación en toda nueva propuesta.

Definición 1.1.5. Protocolo Correcto
Se dice que un protocolo criptográfico es *correcto* si se desarrolla de manera satisfactoria para todas las partes implicadas cuando todos los participantes actúan honestamente.

Cramer propone en [Cra99] una definición alternativa a la anterior añadiendo una segunda propiedad, la imparcialidad. En este caso un protocolo se dice que es correcto si ninguna de las partes acepta un resultado falso, y se dice que es imparcial, si todos obtienen el resultado esperado una vez finalizada la ejecución del mismo.

Finalmente, las características comunes a todas las propuestas incluidas en la presente memoria se pueden resumir en los siguientes puntos:

- Todos los protocolos pertenecen a la clase de los *protocolos probablemente seguros* [MOV96] puesto que están basados en problemas considerados difíciles en la jerarquía de la Complejidad Computacional. Dicha noción fue introducida por Goldwasser y Micali en [GM84] pero no con la finalidad del diseño de protocolos sino como concepto relacionado con los cifrados asimétricos probabilistas, el primero de los cuales es introducido en dicho trabajo. La clave del planteamiento presente en esta memoria es reducir el protocolo a una primitiva de manera que para comprobar la seguridad del protocolo basta con demostrar que la única

manera de romperlo es rompiendo la primitiva subyacente. Esto permite asegurar que para que un adversario tenga éxito a la hora de realizar alguno de los ataques conocidos debe ser capaz de resolver el problema de decisión asociado.

- Se minimiza, y en muchos casos incluso se anula, el intercambio de información previa necesario entre los participantes.
- No se requiere el uso de cifrados, aunque podrían utilizarse en varias propuestas de esquemas de compromiso de bits.
- Su ejecución se realiza en tiempo real (on-line).
- Existe interacción entre los usuarios.

Una cuestión que no se debe dejar de lado es la aleatoriedad ya que es uno de los cimientos del diseño de protocolos y en general de la mayoría de las técnicas criptográficas. La mayoría de los protocolos analizados pertenecen al conjunto de los probabilistas puesto que durante el desarrollo de la mayoría se realizan elecciones aleatorias de algunos elementos. Es por esto por lo que la aleatoriedad juega un papel tan importante. También se usará en otras vertientes como por ejemplo en la generación de los grafos que determinan las instancias de los problemas base en varios casos, tal y como se planteará en la sección 1.3.

El paradigma de la Aleatoriedad ha sido abordado desde tres puntos de vista principales en las Ciencias de la Computación. Dichos puntos de vista comprenden la Teoría de la Información, [Sha48], la Teoría de la Complejidad de Kolmogorov [Lee90] y la propuesta por [GM84] relacionada con la Criptografía. En esta última visión la aleatoriedad se relaciona directamente con los recursos computacionales disponibles para los observadores obteniéndose la definición incluida a continuación.

Definición 1.1.6. Distribución seudoaleatoria
Una distribución se dice seudoaleatoria si no existe ningún procedimiento eficiente

(algoritmo probabilista polinomial) capaz de distinguir dicha distribución de la distribución uniforme.

La necesidad de la aleatoriedad en Criptografía, y en particular en el diseño de protocolos, es diferente las necesidades de que puedan tener otras disciplinas, como pueden ser la Simulación o el Análisis de Algoritmos, puesto que la impredecibilidad pasa a ser una de las características primordiales para solventar los ataques de los que un criptosistema puede ser objeto.

Las aplicaciones criptográficas que usan fuentes de aleatoriedad son innumerables, desde la generación de contraseñas para la implementación de esquemas de control de acceso, la generación de claves para esquemas de cifrado hasta llegar a la materia estudiada en este trabajo, el diseño de protocolos. En esta disciplina el aleatorizar el intercambio de información entre los participantes es extremadamente importante y se utiliza como técnica de enmascaramiento de información en muchos protocolos. Muestra clara de esta aplicación son los protocolos basados en métodos de reto-respuesta. Especialmente destaca el que la propiedad de solidez, y por tanto la privacidad, en las demostraciones de conocimiento nulo se base en elecciones aleatorias realizadas por los dos participantes.

En la bibliografía se pueden encontrar algunos trabajos en los que se estudia la relación entre aleatoriedad y protocolos particulares, como es el caso de [DC95] en el que se estudia el efecto de la reutilización de los retos en las sucesivas iteraciones de demostraciones de conocimiento nulo proponiendo técnicas que permiten dicha reutilización.

Una dificultad importante con la que se debe contar a la hora de realizar propuestas de nuevos protocolos es el garantizar que la simultaneidad durante la ejecución de los mismos no perjudique la seguridad de los mismos. Una manera común de solventarla es usando en la etapa de modelización teórica modelos síncronos.

1.1.5. Tipos de Adversarios y Modelos

Una de las cuestiones a tener en cuenta en el diseño de los protocolos tratados en esta memoria es el modelo formal que subyace, entendiendo por modelo formal las condiciones teóricas en las que se entiende que el protocolo en cuestión es seguro. De tal manera, se distinguen los protocolos seguros ante participantes semi-honestos y los protocolos seguros ante participantes maliciosos.

Definición 1.1.7. Modelo semi-honesto
Se dice que un participante es *semi-honesto* si sigue las reglas del juego pero intenta obtener toda la información que le sea posible a partir de los datos de entrada que reciba del resto de participantes.

Definición 1.1.8. Modelo malicioso
Se dice que un participante es *malicioso* si se desvía de manera arbitraria del desarrollo del protocolo en cuestión.

A partir de las definiciones anteriores se estudia la conservación de las propiedades de privacidad y la corrección de los diferentes protocolos en ambos modelos de adversarios. Así surge a definición de protocolo bipartito privado en el modelo semi-honesto incluida a continuación [GMW87].

Definición 1.1.9. Protocolo privado en el modelo semi-honesto
Un protocolo bipartito en el que participan $\mathcal{A}$ y $\mathcal{B}$ se dice que preserva la privacidad en el modelo semi-honesto si la usuaria $\mathcal{A}$ preserva la privacidad ante su contrario y viceversa.

Esta definición sólo establece la conservación de la privacidad respecto al comportamiento de los participantes que intervienen en el protocolo y no ante agentes externos al mismo, lo que implica la introducción de un concepto de privacidad más restrictivo para este modelo.

Definición 1.1.10. Protocolo de máxima privacidad
Un protocolo bipartito en el que participan $\mathcal{A}$ y $\mathcal{B}$ se dice que tiene máxima privacidad si la usuaria $\mathcal{A}$ preserva la privacidad respecto a cualquier usuario $\mathcal{B}^*$ polinomialmente acotado y además el usuario $\mathcal{B}$ tiene la capacidad análoga.

1.1.6. Metodología de Diseño

Los puntos a analizar antes, durante y después del diseño de un protocolo son innumerables. De ahí que definir explícitamente una metodología de diseño sea una tarea realmente difícil. Por ello se ha decidido diferenciar técnicas generales y un tanto abstractas, de aquellas otras útiles que se manejan en la fase práctica de diseño.

La herramientas genéricas del primer tipo disponibles para esta tarea se pueden clasificar en dos conjuntos, los principios generales de diseño y los métodos formales, siendo ambos complementarios. Uno de los primeros trabajos en los que aparecen explícitas algunas reglas de diseño es [Moo87].

En este caso se hace hincapié en la necesidad de utilizar descripciones precisas de los protocolos y de las hipótesis necesarias para su correcto funcionamiento. En este sentido también debemos hacer referencia al trabajo de Abadi y Needham [AN96] ya que en él se proponen dos principios fundamentales de diseño que se pueden resumir de la manera siguiente:

1. Cualquier mensaje intercambiado debe ser totalmente autocontenido, esto es, su interpretación queda totalmente definida por el contenido del mismo.

2. Las condiciones de un mensaje deben ser explícitas y de manera tal que cualquiera que revise el diseño del protocolo en cuestión pueda decidir si son aceptables o no.

Un enunciado alternativo para el primer principio podría resumirse diciendo que la descripción del funcionamiento del protocolo debe ser totalmente inteligible para

las partes que intervienen e incluso para cualquier individuo con acceso a tal descripción, de manera que la seguridad del protocolo reside en mecanismos particulares. Este enunciado hace recordar uno de los principios enunciados por Kerckhoffs, en el que afirmaba que la seguridad de un criptosistema debe residir en mantener a buen recaudo la clave y no en mantener en secreto la descripción del criptosistema.

Los dos principios anteriores son estudiados con mayor detalle y ampliados en un trabajo posterior [AN95]. Dichos principios no pueden considerarse condiciones suficientes para garantizar el correcto funcionamiento de los protocolos diseñados, sino más bien ideas a tener en cuenta sobre todo durante la fase de diseño.

En cuanto a los métodos formales se hace necesario mencionar el trabajo [But99] en el que se puede encontrar un estado del arte sobre las diferentes utilidades que los métodos formales aportan al diseño de protocolos en cada una de sus tres fases de especificación, construcción y verificación. Esta última faceta es en la que más se han realizado investigaciones, fruto de las cuales es la aparición de lógicas específicas (BAN) basadas en otras existentes para la verificación de algoritmos distribuidos en general. Es conveniente destacar que tampoco los métodos formales por sí solos sirven para garantizar la seguridad, tal y como han demostrado algunas de las críticas que han recibido [BGH+91] debido sobre todo a la idealización requerida de los protocolos analizados.

Considerando una visión más particular del diseño merecen mención específica dos técnicas de considerable importancia, sobre todo en el diseño de los protocolos bipartitos. Estas son la técnica de corte y elección, y de reto-respuesta.

La técnica de corte y elección (cut-and-choose) se puede describir fácilmente con un símil muy utilizado en la literatura existente sobre el tema. Supuesto que hay una tarta y dos niños (Alicia y Bernardo), ¿cómo es posible que realicen un reparto lo más equitativo posible?. La repuesta la da el siguiente algoritmo:

- *Corte*: Alicia divide el objeto en dos partes.

- *Elección*: Bernardo elige una de las supuestas mitades.

Esta técnica fue usada por primera vez en 1978 en el ámbito de la criptografía por Rabin, [Rab78], aunque también se ha usado más recientemente en el diseño de esquemas de dinero digital [CdBvH$^+$89], [Cha87]. Además, tal y como se puede comprobar en el apartado 3.2, es un ingrediente fundamental en las Demostraciones de Conocimiento Nulo. Su utilidad reside en el hecho de que actuando de esta forma, el individuo encargado de cortar la información no conoce cuál será la porción seleccionada por su contrario. De esta manera, si la división es fraudulenta o injusta Bernardo podrá definir una estrategia óptima seleccionando siempre la porción que más le interese.

Esta técnica, no sólo tiene aplicación en esta disciplina tal y como demuestran las múltiples referencias en el campo de la lógica [Zal99], donde se define como una herramienta matemática útil en la teoría de las definiciones y en la teoría de la estabilidad.

En cuanto a la técnica de reto-respuesta (challenge-response), se debe mencionar que donde es ampliamente usada es sobre todo en los esquemas de identificación (ver sección 3.5) y en el control de accesos a recursos informáticos. También consta de dos pasos (tal y como sugiere la gráfica de la figura 1.2) en los que se produce intercambio de información entre los implicados:

- *Reto*: Bernardo envía a Alicia una información aleatoria.
- *Respuesta*: Alicia utiliza la información secreta de la que dispone para responder al reto de Bernardo.

De nuevo se refleja en el esquema de reto-respuesta la importancia de la aleatoriedad, ya que los retos deben ser totalmente aleatorios para garantizar la impredecibilidad de las respuestas y por tanto reducir la posibilidad de que Bernardo sea objeto de estafa.

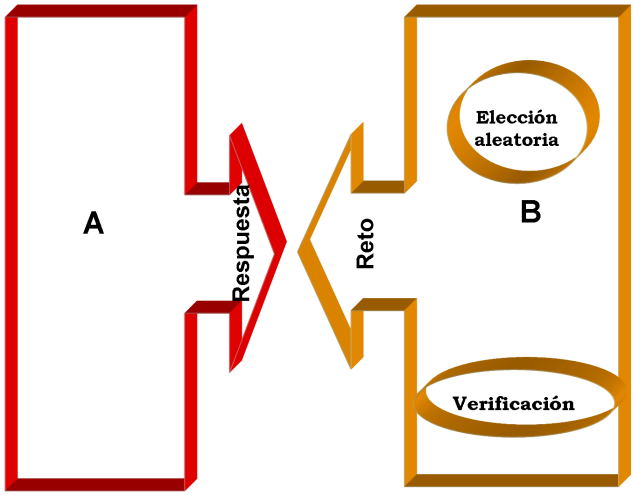

Figura 1.2: Esquema de reto-respuesta

1.2. Cuestiones de Complejidad

Sin duda alguna, la Complejidad Computacional es uno de los pilares de la Criptografía junto con la Teoría de la Información. La justificación de esta afirmación recae en el hecho de que detrás de la mayoría de criptosistemas subyace un problema tal que los recursos computacionales necesarios para su resolución son considerables, garantizándose así la seguridad del mismo. Por tanto, romper el criptosistema en cuestión acarrea una cantidad de recursos tal que el ataque se convierte en una tarea prácticamente inabordable.

De lo anterior se deduce la necesidad de que dicho problema esté catalogado en una de las clases superiores (en cuanto a dificultad) dentro de la jerarquía definida por la Complejidad Computacional. Por esta razón a continuación se incluyen las definiciones de los conceptos básicos de Complejidad Computacional necesarios para la posterior descripción de los protocolos.

El concepto de algoritmo es fundamental en la Teoría de la Complejidad. A grandes rasgos se puede afirmar que esta disciplina se encarga básicamente de analizar los recursos que un determinado algoritmo necesita para la resolución de un problema

determinado. Dichos recursos dependen del modelo computacional en que el algoritmo se basa, motivo por el cual se introduce también los modelos computacionales básicos en este apartado.

1.2.1. Conceptos Básicos en Complejidad Computacional

En la presente sección se introducen algunos conceptos fundamentales como los de problema de decisión, modelos computacionales y completitud. Para un estudio más profundo de esta materia se recomienda consultar [Lee90], [BDG88] y [GJ79] entre otros.

Definición 1.2.1. Problema
Un problema Π es un conjunto de pares ordenados (I, S) formados por cadenas binarias finitas. El primer elemento I se denomina *instancia* y el segundo S es una respuesta o solución para dicha instancia. Además toda cadena binaria finita es considerada como posible instancia de al menos uno de estos pares ordenados.

$$\Pi = \{(I, R)/I, R \in \{0,1\}^*\}$$

Definición 1.2.2. Problema de decisión
Un problema de decisión Π_D es un problema en el que la respuesta a cada instancia es un único valor binario $S \in \{0,1\}$.

La importancia de estos problemas es indiscutible ya que muchas de las clases de complejidad se definen explícitamente para este tipo de problemas.

Definición 1.2.3. Lenguaje
Un lenguaje L sobre el alfabeto binario $\Sigma = \{0,1\}^*$ se define como cualquier subconjunto de secuencias binarias $L \subseteq \{0,1\}^*$.

Definición 1.2.4. Lenguaje complementario
Para un lenguaje L definido sobre un alfabeto Σ, se define su lenguaje complementario

$\overline{L}$ como el conjunto de cadenas de Σ^* tal que no pertenecen a L ($\overline{L} = \Sigma^* - L$). Asimismo para una clase de lenguajes C, se define $co - C$ como $co - C = \{\overline{L} : L \in C\}$.

Dadas las definiciones anteriores, es fácil establecer una correspondencia entre lenguajes y problemas de decisión, y de hecho muchas veces se trabaja indistintamente con el lenguaje definido por un determinado problema de decisión o con el problema de decisión. Esta relación se utilizará en el capítulo 3 para la definición de las Demostraciones de Conocimiento Nulo.

Definición 1.2.5. Recurso
Si M es un método para resolver un problema Π y R es un recurso usado por ese método, el enfoque de la Teoría de la Complejidad clásica considera que la cantidad de recurso R necesitada por el método M es la función definida por $R_M : \mathbb{Z}^+ \rightarrow \mathbb{Z}^+$ donde $R_M(n) = \underset{\forall x \in \{0,1\}^* : |x| = n}{\text{máx}} \{\text{cantidad de recurso R usado por M(x)}\}$ y $M(x)$ denota la aplicación del método M sobre la entrada x.

A la hora de clasificar los métodos de resolución se establecen ciertas cotas sobre los recursos requeridos. Las más significativas se muestran a continuación:

Constante: Existe una constante k tal que $R_M(n) \leq k \forall n \geq 0$.

Logarítmico: Existe una constante positiva k tal que $\forall n \geq 0, R_M(n) \leq k \lg n$, es decir $R_M(n) = O(\lg n)$.

Poli-logarítmico: Existe una constante k tal que $R_M(n) \leq k \lg^k n, \forall n \geq 0$, es decir, $R_M(n) = O(\lg^k n)$.

Lineal: Existe una constante k tal que $R_M(n) \leq kn, \forall n \geq 0$, es decir, $R_M(n) = O(n)$.

Polinomial: Existe una constante k tal que $R_M(n) \leq kn^k, \forall n \geq 0$, es decir, $R_M(n) = O(n^k)$.

Exponencial: Existe una constante k tal que $R_M(n) \leq k2^{n^k}, \forall n \geq 0$, es decir, $R_M(n) = O(2^{n^k})$.

No acotado: La única acotación que se puede realizar en este caso es la finitud de la cantidad de recurso R necesaria para una instancia particular.

Normalmente se presta mayor atención a los recursos definidos por el tiempo y el espacio necesario para la computación. En el caso de la complejidad en tiempo ésta se expresa en términos del tamaño de la instancia a resolver. Concretamente, la función que determina la complejidad en tiempo de un algoritmo asocia a cada posible tamaño de la entrada la máxima cantidad de tiempo que necesita el algoritmo en cuestión para resolver cualquier instancia de ese tamaño.

1.2.2. Modelos Computacionales y Clases de Complejidad

Para poder estudiar la manera en que las medidas anteriores son definidas atendiendo a los recursos temporales se incluyen a continuación los tipos de máquinas formales (modelos computacionales subyacentes) que se utilizarán en posteriores definiciones.

Las Máquinas de Turing que se definen a continuación son el modelo estándar en la actual Complejidad Computacional. La variedad que se puede definir sobre el modelo básico siguiente es verdaderamente amplia aunque en este caso se presentan sólo los modelos básicos.

Definición 1.2.6. Máquinas de Turing (TM, Turing Machine)
En una Máquina de Turing la memoria está formada por una colección finita de cintas divididas en celdas. Cada celda posee capacidad para almacenar un símbolo del alfabeto considerado. Para cada cinta existe al menos un canal que conecta un control con la cinta. Estas son las llamadas cabezas de lectura/escritura. La máquina lee el conjunto ordenado de símbolos de las cintas de lectura actuales, y en función

de estos símbolos y del estado del control la máquina se produce la migración a un nuevo estado. La versión más simple de este modelo es la máquina de Turing con una única cinta infinita, una cabeza de lectura/escritura y un control. Dicha máquina se puede describir usando una 7-upla $M =< Q, \Sigma, \Gamma, \delta, B, q_0, q_f >$ en la que:

- Q es un conjunto finito de estados.
- Σ es el alfabeto de entrada, normalmente $\Sigma = \{0, 1\}$.
- Γ es el alfabeto de la cinta, contiene al alfabeto de entrada y un carácter especial B llamado blanco aunque también puede contener otros símbolos ($\{0, 1, B\}$).
- δ es la función de transición entre estados definida entre $Q \times \Sigma \rightarrow Q \times \Sigma \times \{Izquierda, Derecha, NoMovimiento\}$.
- $\mathcal{B}$ es el símbolo blanco.
- q_0 es el estado inicial.
- q_f es el estado final.

Al comienzo, la cabeza escanea la celda inicial. El contenido de las celdas contiguas a la derecha de ésta es la cadena de entrada (x) y el resto contiene el símbolo blanco. El control se encuentra en su estado inicial q_0. A partir de este momento la computación se realiza en pasos discretos en cada uno de los cuales, dependiendo del símbolo leído y del estado en el que se encuentra el control, las acciones que se pueden realizar consisten en escribir un símbolo del alfabeto de entrada Γ en la posición actual o desplazar la cabeza una posición a la derecha o a la izquierda. Las acciones anteriores son especificadas por su conjunto de instrucciones definido por δ. Cada instrucción tiene la forma (q, c, r, d, ϵ) donde $q, r \in Q$, (q es el estado actual y r el estado siguiente) $c, d \in \Gamma$ (c es el símbolo leído y d el que se escribirá a continuación) y $d \in \Gamma$ y

$\epsilon \in \{Izquierda, Derecha, NoMovimiento\}$, este último elemento indicará donde se realizará la escritura.

La importancia del concepto de máquina de Turing se debe a que posibilita la formalización de las medidas de complejidad en tiempo y espacio. Antes de especificar las variantes del modelo anterior se introduce en qué consiste el que una máquina de Turing reconozca un lenguaje. Se dice que una máquina de Turing reconoce un lenguaje L si ésta acepta todas las cadenas del lenguaje y rechaza el resto.

Máquina de Turing Determinista y Clase P

En el caso de las máquinas de Turing deterministas, la máquina más sencilla posee tres cintas, una semi-infinita solamente de lectura para recibir la entrada, una semi-infinita solamente de escritura que se usa para escribir la salida y otra de lectura-escritura.

Definición 1.2.7. Máquina de Turing Determinista (DTM, Deterministic Turing Machine)
La principal característica de una DTM es que dada una entrada es capaz de desarrollar una única sucesión de operaciones (es decir, para cada par de valores de q y c, mencionados en la definición de máquina de Turing, existe exactamente una instrucción (q, c, r, d, ϵ)) que puede terminar en un número finito de pasos o no. Si termina, la salida será la cadena más larga de caracteres comenzando en la celda donde la cabeza paró y recorriéndola hacia la derecha.

El tiempo de computación viene dado simplemente por el número de pasos realizados hasta que la máquina alcanza uno de los estados de parada. Por otra parte, el espacio es el número de celdas en la cinta de trabajo que fueron visitadas por la cabeza lectora durante la computación.

Ahora tiene sentido definir lo que se entiende por función de *complejidad tiempo* T_M demandado por una DTM M que es la función $T_M : \mathbb{Z}^+ \to \mathbb{Z}^+,\ n \to T_M(n)$,

donde $T_M(n)$ mide el máximo de los tiempos de computación de M sobre todas las cadenas de longitud n. A partir de esta función se definen las clases de complejidad. En general la clase de lenguajes cuyo tiempo de complejidad es como máximo $f(n)$ se denota por $DTIME(f(n))$.

Definición 1.2.8. Clase P (Polynomial)
Una DTM se dice que es polinomial si existe un polinomio f tal que $T_M(n) = O(f(n))$. Extendiendo esta noción a los problemas de decisión se dice que un problema de decisión Π_D es de complejidad polinomial si el lenguaje asociado al mismo ($L(\Pi_D)$) es reconocible por una DTM polinomial. La clase P denota al conjunto de problemas de decisión de complejidad polinomial.

Máquina de Turing no Determinista y Clase NP

En esta variante, la máquina posee diferentes alternativas en cada paso para realizar el movimiento siguiente. Por tanto, para la misma entrada y el mismo estado se pueden tener diferentes procesamientos y salidas. Entonces la única diferencia respecto a una DTM se encuentra en la función de transición, ya que ésta deja de ser una función en el sentido matemático, para pasar a ser una relación.

Definición 1.2.9. Máquina de Turing no Determinista ($NDTM$, Non Deterministic Turing Machine)
Dado un estado $q \in Q$ y un símbolo del alfabeto de entrada $a \in \Sigma$, la función de transición de estados de una $NDTM$ $\delta(q, a)$ consiste en el conjunto formado por todas las ternas (r, d, ϵ) que representan movimientos válidos. La máquina selecciona en cada paso una cualquiera de las instrucciones disponibles.

Una manera de modelar su comportamiento es asociar a todo el proceso de computación un árbol en el que los nodos están etiquetados por una configuración de la máquina, entendiendo por configuración la especificación del estado actual, la posición de la cabeza en la cinta y el contenido de la propia cinta. De esta manera,

cada nodo tiene tantos hijos como instrucciones diferentes pueden ejecutarse para esa configuración y cada hijo es la configuración resultante al ejecutar la instrucción correspondiente. La máquina acepta la entrada cuando el árbol contiene cualquier configuración en la que la máquina se detiene. En este caso la cinta de salida contiene la cadena que representa la respuesta afirmativa. El tiempo usado queda determinado por la profundidad del árbol asociado, y el espacio viene dado por el máximo número de celdas usadas calculado sobre todas las configuraciones del árbol.

A continuación se aclaran algunas cuestiones sobre la relación entre los dos modelos formales de computación previamente introducidos. La primera es el hecho de que toda DTM puede definirse como un caso particular de $NDTM$. Además, al contrario de lo que sucedía con las DTM, las no deterministas son meros modelos teóricos que sirven para la descripción y formulación de problemas pero no para su resolución. Otra cuestión es que no se considera asociada ninguna distribución de probabilidad sobre el conjunto de movimientos posibles, ya que esto es propio de las máquinas de Turing probabilistas que se definirán más adelante.

Cuando se restringe esta idea al conjunto de problemas de decisión se dice que una $NDTM$ M resuelve el problema de decisión Π en tiempo $f(n)$, donde $f : \mathbb{Z}^+ \to \mathbb{Z}^+$, si dicha máquina acepta todas las cadenas del lenguaje asociado $L(\Pi_D)$ en tiempo $f(n)$. En general, las clases de complejidad formadas por los problemas de decisión resolubles por una $NDTM$ en tiempo $f(n)$ se denotan por $NTIME(f(n))$.

Que un lenguaje (problema) sea reconocible por una $NDTM$ en un tiempo determinado está intrínsecamente relacionado con la dificultad de demostrar que una determinada cadena pertenece ese lenguaje. Por consiguiente, las $NDTM$ nos permiten clasificar los problemas de decisión no tanto por su dificultad sino por la del proceso de verificación asociado al mismo. El nuevo planteamiento ahora es el siguiente, dada una solución candidata a un problema, ¿qué cantidad de recursos requiere

el comprobar si dicha candidata es realmente una solución?. De hecho se puede demostrar que la clase de problemas de decisión resolubles por una $NDTM$ polinomial coincide con el conjunto de problemas de decisión para los que existe un procedimiento de verificación en tiempo polinomial.

Definición 1.2.10. Clase NP (Non-deterministic Polynomial)
La clase de problemas de decisión resolubles por una $NDTM$ polinomial coincide con el conjunto de problemas de decisión para los que existe un procedimiento de verificación en tiempo polinomial, y se conoce como la clase NP.

Es necesario destacar que cuando se dice que un problema pertenece a esta clase lo único que se caracteriza es la facilidad del procedimiento de verificación y no el nivel de dificultad del proceso de resolución.

Es fácil establecer que $DTIME(f(n)) \subset NTIME(f(n))$ puesto que como ya se había avanzado, una DTM puede verse como un caso particular de $NDTM$. Para el caso particular de P y NP la conjetura de que $P \neq NP$ es fundamental para muchos criptosistemas.

Una manera alternativa de definir la clase NP es considerarla como el conjunto de lenguajes que admiten una certificación corta para su pertenencia a la clase. Además dado este certificado, denominado testigo, la pertenencia al lenguaje se puede verificar en tiempo polinomial. Formalmente se dice que un lenguaje L_0 perteneciente a la clase $DTIME(g(n))$ es un testigo de longitud $f(n)$ para un lenguaje L si y sólo si dada un cadena x de L existe una cadena y tal que $\mid y \mid < \mid f(x) \mid$ definida sobre el mismo alfabeto de manera que al concatenar ambas cadenas, la resultante pertenece al lenguaje L_0.

Otro tipo importante de problemas son los problemas de búsqueda. En este caso se trata no sólo de averiguar si existe solución sino además, en caso de que exista

encontrarla. Muchos problemas de la clase NP poseen un problema de búsqueda asociado que puede ser resuelto de manera "sencilla" si se sabe cómo resolver el problema de decisión. Algo parecido sucede en algunos casos con los problemas de optimización consistentes en determinar la mejor solución respecto determinado criterio. Esta vez si se conoce un algoritmo polinomial para resolver el problema de decisión, la resolución del problema de optimización se puede determinar en tiempo polinomial combinando dicho algoritmo con una búsqueda binaria.

Clases NP-Completa y NP-dura

Una clase de problemas que es necesario destacar en la presente memoria es la clase $NP - completa$.

Definición 1.2.11. Clase $NP - Completa$
La clase $NP - Completa$ está formada por todos los problemas de decisión que pertenecen a NP y que además poseen la característica de que todo problema NP es polinomialmente reducible a ellos.

Definición 1.2.12. Reducción polinomial
Se dice que un problema Π_1 es polinomialmente reducible a el problema Π_2, si existe una función polinomial $f : D_{\Pi_1} \rightarrow D_{\Pi_2}$ tal que $\forall I \in D_{\Pi_1}$, se tiene $I \in Y_{\Pi_1} \Leftrightarrow f(I) \in Y_{\Pi_2}$, donde $D_{\Pi_i, i=1,2}$ denota al conjunto de instancias del problema correspondiente y Y_{Π_i} está formado por el conjunto de instancias con respuesta afirmativa.

Otro conjunto de problemas particularmente interesante por su dificultad son los siguientes.

Definición 1.2.13. Clase $NP - dura$
La clase de problemas que, aún no perteneciendo a la clase NP, permiten que todo problema NP sea reducible a ellos, recibe el nombre de clase $NP - dura$.

Máquina de Turing Probabilista y Clases de Complejidad Probabilistas

Tal y como se ya se mencionó, cuando se asocia una distribución de probabilidad sobre el conjunto de posibles movimientos, se habla de Máquinas de Turing Probabilistas.

Definición 1.2.14. Máquina de Turing Probabilista (PTM, Probabilistic Turing Machine)
Una máquina de Turing probabilista es una $NDTM$ tal que para todo estado y para todo elemento del alfabeto de entrada se define una distribución de probabilidad sobre las alternativas disponibles para cada paso siguiente.

Una PTM se puede restringir, sin pérdida de generalidad, a una $NDTM$ con dos alternativas posibles en cada paso, de manera que en cada paso puede seleccionar cada una de ellas con probabilidad 1/2. Se puede decir por tanto que en cada paso la acción a realizar después es seleccionada de manera equivalente a lanzar una moneda.

En general, el tiempo de computación depende de la distribución de probabilidad definida por lo que será una variable aleatoria, lo que implica la necesidad de definir las clases de complejidad en función del tiempo esperado.

Definición 1.2.15. Clase PP (Probabilistic Polynomial)
La clase probabilista polinomial es la clase de todos los lenguajes L para los que existe una PTM, M tal que las probabilidades de reconocer una cadena del lenguaje, y de rechazar las cadenas no pertenecientes al lenguaje son estrictamente mayores que 1/2, todo ello en tiempo esperado polinomial. Una manera más formal de expresarlo es definiendo $\chi(x) = \begin{cases} 1 & x \in L \\ 0 & x \notin L \end{cases}$ de forma que

$$PP = \{L \subseteq \{0,1\}^* / \exists M \ PTM \text{polinomial t.q. } \forall x \ Prob(M(x) = \chi(x)) > 1/2\}.$$

Los algoritmos que responden a esta definición son conocidos como algoritmos de Monte-Carlo. Esta clase de problemas en realidad no interesa demasiado en el

contexto de la memoria debido a que la diferencia entre lo que sucede con las cadenas que pertenecen realmente al lenguaje y las que no es prácticamente inexistente. Sin embargo, las siguientes clases si tienen utilidad en algunas aplicaciones concretas de demostraciones interactivas.

Definición 1.2.16. Clase BPP (Bounded Probabilistic Polynomial)
La clase de tiempo acotado probabilista polinomial está formada por todos los lenguajes de la clase PP para los que existe una PTM polinomial que acepta todas las cadenas del lenguaje en cuestión con probabilidad estrictamente mayor o igual que $2/3$ y análogamente rechaza las cadenas que no pertenecen al lenguaje con probabilidad acotada inferiormente por $2/3$. Siguiendo la notación usada en la definición anterior

$$BPP = \{L \subseteq \{0,1\}^* / \exists M \text{ PTM polinomial t.q. } \forall x \; Prob(M(x) = \chi(x)) \geq 2/3\}.$$

La desigualdad anterior se puede desglosar de la siguiente manera, si $x \in L \Rightarrow Prob(M(x) = 1) \geq 2/3$ y si $x \notin L \Rightarrow Prob(M(x) = 1) < 1/3$, lo que significa que la máquina se equivoca a veces, pero la mayoría de veces proporciona una respuesta correcta. Normalmente para referirse a las máquinas asociadas a la clase BPP se usa la expresión máquinas con error bilateral (two-sided error machines). Aprovechando esto la ejecución se puede repetir un número determinado de veces para la misma cadena x garantizándose así el decrecimiento de la probabilidad de error. Se puede resumir esta idea diciendo que BPP es la clase de lenguajes para los que existe una PTM cuyo tiempo de ejecución está acotado por un polinomio, que acepta la mayoría de cadenas, y tal que existe bastante diferencia entre la probabilidad de que acepte una cadena del lenguaje y la probabilidad de que acepte otra que no lo es.

La apreciación anterior implica de alguna forma que la clase BPP es más robusta que la clase PP. A continuación se introduce una nueva clase definida también sobre PTM.

Definición 1.2.17. Clase RP (Randomized Polynomial)
La clase polinomial aleatorizada está formada por los lenguajes para los que existe una PTM polinomial tal que todas las cadenas del lenguaje son aceptadas con probabilidad estrictamente superior a $1/2$ y las no pertenecientes al mismo son rechazadas con probabilidad 1. Esto se puede formular de la siguiente manera:

$$RP = \{L \subseteq \{0,1\}^* / \exists M \text{ PTM polinomial t.q. } \forall x \in L,$$

$$Prob(M(x) = \chi(x)) \geq 1/2 \ \wedge \ \forall x \notin L, \ Prob(M(x) = \chi(x)) = 1\}.$$

A las máquinas capaces de reconocer este tipo de lenguajes se las denomina máquinas con error unilateral (one-sided error machine). Los algoritmos pertenecientes a esta clase son más potentes que los pertenecientes a la clase BPP puesto que repitiendo su ejecución k veces se obtiene un algoritmo con probabilidad de error acotada por 2^{-k}.

Definición 1.2.18. Clase ZPP (Zero-error Probabilistic Polynomial)
La clase polinomial con probabilidad de error nulo está formada por los lenguajes para los que existe una PTM M tal que $\forall x, Prob[M(x) = \perp] \leq 1/2$ y $\forall x, Prob[M(x) = \chi(x) \ ó \ M(x) = \perp] = 1$, donde $\perp$ simboliza la respuesta de la máquina cuando no es capaz de tomar una decisión sobre la pertenencia o no al lenguaje de la cadena examinada.

La idea de esta clase es que la máquina M nunca da una respuesta incorrecta, pero puede suceder que no pueda decidir sobre el problema en cuestión con probabilidad acotada por $1/2$. Esta clase se corresponde con los algoritmos denominados de Las Vegas. Al igual que sucedía en las clases anteriores, a las máquinas asociadas a esta clase se les reserva un nombre relacionado con el tipo de error que pueden cometer al dar una respuesta, en este caso máquinas con error nulo (zero error machines).

Como conclusión se puede decir que las cuatro clases de complejidad definidas para las máquinas de Turing aleatorias tienen en común la necesidad de tiempo polinomial,

Clase	$x \in L$	$x \notin L$
PP	$Prob(aceptar\ x) > 1/2$	$Prob(rechazar\ x) > 1/2$
BPP	$Prob(aceptar\ x) > 2/3$	$Prob(rechazar\ x) > 2/3$
RP	$Prob(aceptar\ x) > 1/2$	$Prob(rechazar\ x) = 1$
ZPP	$Prob(aceptar\ x) > 1/2$	$Prob(rechazar\ x) > 1/2$
	$Prob(rechazar\ x) = 0$	$Prob(aceptar\ x) = 0$

Figura 1.3: Clases de complejidad para algoritmos aleatorios

pero las probabilidades de aceptar y rechazar cadenas, tal y como se puede apreciar en la figura 1.3, establecen las diferencias entre ellas.

El siguiente teorema establece algunas de las relaciones más interesantes entre las clases definidas.

Teorema 1.2.1. *$P \subseteq ZPP \subseteq RP \subseteq BPP \subseteq PP$ y $RP \subseteq NP$.*

Otra máquina de Turing a la que haremos referencia en el Capítulo 3 es la Máquina de Turing Oráculo definida a continuación.

Definición 1.2.19. Máquina de Turing Oráculo (OTM, Oracle Turing Machine)
Una Máquina de Turing Oráculo (OTM) funciona como una máquina de Turing normal salvo que ésta puede escribir una cadena z y preguntar si dicha cadena está en el lenguaje reconocido por el oráculo. La máquina obtiene la respuesta correcta en un paso y puede distinguir su computación en función de la respuesta obtenida.

La definición anterior se refiere tanto a a máquinas de Turing deterministas como no deterministas. En ambos casos, la máquina consta de una cinta oráculo adicional y semi-infinita. El uso de esta nueva cinta se alterna entre modo de sólo lectura y modo de sólo escritura. Se usan sobre todo para extender la potencia de las máquinas de Turing a la hora de estudiar la dificultad comparativa de ciertos problemas. Asociado a cualquier máquina oráculo existe un problema (lenguaje) particular Y que puede ser resuelto (reconocido) por ella sin costo alguno. Mientras la cinta oráculo está en modo de escritura, la máquina de Turing puede pasar al estado de interrogación en

cualquier momento. En ese caso, el contenido y de la cinta oráculo es reemplazado en el siguiente paso por una cadena b tal que $(y, b) \in Y$ y la cinta pasa a ser sólo de lectura. Por tanto preguntar al oráculo es como invocar a una subrutina para resolver Y de forma que el tiempo que tarda en resolverlo no se tiene en cuenta.

Hasta el momento todas las definiciones incluidas hacen referencia a los recursos temporales necesarios para la resolución de los problemas mencionados. A continuación se listan algunas definiciones relacionadas con la jerarquía de clases de complejidad cuando las limitaciones se producen en el espacio necesitado. Desde este punto de vista, se define la complejidad espacio de una Máquina de Turing de la siguiente manera:

Definición 1.2.20. Complejidad en Espacio de una Máquina de Turing
Sea M una Máquina de Turing. Si para todo $n \in \mathbb{N}$ toda secuencia de movimientos legales asociados a una entrada x de longitud n usa como máximo $s(n)$ celdas de la cinta de trabajo (no se incluyen las celdas de la cinta de lectura) entonces se dice que M tiene complejidad en espacio $s(n)$.

Partiendo de la definición anterior se pueden describir las siguientes clases de complejidad:

- $DSPACE[s(n)]$ denota la clase de lenguajes aceptados por una DTM cuya complejidad en espacio es $s(n)$.
- $NSPACE[s(n)]$ se refiere a la clase de lenguajes aceptados por una $NDTM$ en espacio $s(n)$.
- $PSPACE$ es la clase de complejidad definida por $\bigcup_{c \in \mathbb{N}} DSPACE[n^c]$, es decir, la clase de aquellos lenguajes que son reconocibles en espacio polinomial.

La figura 1.4 muestra algunas de las relaciones entre las distintas clases de complejidad según la jerarquía anteriormente mencionada.

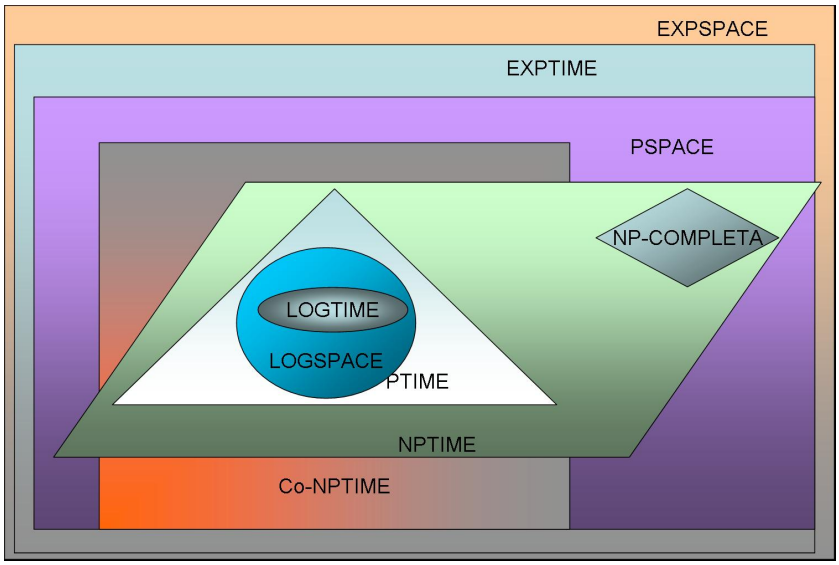

Figura 1.4: Relación entre clases de complejidad

Una de las finalidades principales de la disciplina estudiada en esta sección es, como ya se adelantó, establecer medidas sobre la cantidad de recursos (normalmente tiempo y espacio) utilizados por determinado método de resolución. Generalmente, estas medidas se establecen en función de la cantidad máxima de recurso utilizada para resolver instancias de longitud fija n. Es decir, se utiliza el caso peor como patrón de medida, sin embargo en ciertos casos esta consideración no resulta adecuada. Por ejemplo, es bastante común identificar todo problema perteneciente a la clase $NP - completa$ como un problema difícil, pero cuando se analizan para entradas aleatorias muchos de estos problemas se convierten en fáciles. Es por esto que en criptografía el análisis de complejidad del caso medio juega un papel relevante. En el apartado que se presenta a continuación se incluyen brevemente los conceptos básicos de esta teoría con el objetivo de describir posteriormente la justificación de la elección de un problema base sobre el que se apoyan dos de los protocolos propuestos. Para discusiones más completas sobre el tema se recomienda consultar [Lev86], [VL88].

1.2.3. Análisis del Caso Medio

En este tipo de análisis no se trabaja únicamente con las instancias de problemas, sino que además de esto es necesario definir una distribución de probabilidad sobre el conjunto de instancias. Por eso se usa el término *problema distribucional* para referirse al par ordenado formado por un problema de decisión Π_D y una distribución de probabilidad μ definida sobre las instancias del mismo problema.

De esta forma el tiempo asociado a un algoritmo pasa a ser una variable aleatoria, definiéndose como medida comparativa el tiempo medio de un algoritmo que viene dado por el promedio de los tiempos que tarda dicho algoritmo en resolver todas las posibles instancias de longitud n bajo el modelo de probabilidad definido por μ. La expresión asociada es $T_{A_\mu} = \sum_{\{x \in \Sigma^* \setminus |x|=n\}} T_A(x)\mu(x)$.

Este tipo de análisis depende en gran medida de la elección de la distribución de probabilidad, por lo que lo se hace imprescindible establecer cuáles son los modelos de probabilidad eficientes. Una posible elección que suele facilitar el análisis posterior es la distribución uniforme, siendo éste el modelo fundamental utilizado en esta memoria.

Los orígenes del enfoque del caso medio se deben a Levin, [Lev86], [VL88]. En el primero de estos trabajos se establecen las definiciones básicas y se demuestra la existencia de problemas intratables bajo este análisis. Partiendo de la idea anterior se traslada la jerarquía de la Complejidad Computacional clásica definiendo la clase distribucional análoga a NP como la formada por problemas de decisión pertenecientes a la clase NP emparejados con funciones de distribución cuya función de densidad es susceptible de ser calculada en tiempo polinomial. Estas funciones de distribución reciben el nombre de funciones de distribución $P - computables$.

Existe otra clase de distribuciones, denominadas *P-muestreables*, también importantes ya que se refieren a aquellas para las que existe un algoritmo eficiente de generación de instancias, bajo dicha distribución. Existe una valiosa relación entre ambos

conjuntos de distribuciones cuya demostración se incluye en el trabajo [BDCGL92]. Dicha relación establece que toda distribución $P - computable$ es $P - muestreable$, mientras que su recíproco no se verifica.

De modo equivalente también se define la clase correspondiente a problemas distribucionales polinomiales en media, denotada como clase $Media - P$. Para un problema Π_D perteneciente a esta clase existe un algoritmo que lo resuelve en tiempo t y una constante $\epsilon > 0$ tal que se verifica la expresión $\sum_{x \in \Sigma_*} \mu'(x) \frac{t(x)^\epsilon}{|x|} < \infty$, siendo μ' la función de densidad asociada a μ. Una cuestión que permanece abierta también en este caso es si $Dist - NP \subseteq Media - P$, [BDCGL92].

Otro concepto importante y necesario para trasladar la idea de completitud a la jerarquía del caso medio es el de las reducciones aleatorias de Turing. En este caso las reducciones definidas deben ser eficientemente computables, obtener resultados válidos (instancias positivas de un problema deben corresponder a instancias positivas del problema al que se reduce) y además deben preservar la distribución de probabilidad. Todo esto hace que la demostración de la pertenencia de un determinado problema a la clase $DistNP - completa$ se complique. Sin embargo, se ha demostrado la existencia de problemas pertenecientes a la clase anterior bajo la distribución de probabilidad uniforme [VL88], [Wan97].

Otro resultado destacable en esta teoría es el obtenido por Impagliazzo y Levin [IL90] según el cual para problemas de búsqueda asociados a la clase NP las distribuciones $P - muestreables$ no generan instancias más difíciles que las obtenidas a partir de la distribución uniforme. La versión para problemas de decisión de este resultado fue obtenida posteriormente en, [BDCGL92]. Un catálogo de problemas $DistNP - completos$ se encuentra en [Wan97]. Entre estos problemas destacan el problema de la teselación, por ser el primero para el que se obtuvo una demostración formal de pertenencia a esta clase, y la representación de matrices usado como base en los protocolos presentados en el capítulo 4.

La mayoría de los protocolos existentes, incluidos los aquí propuestos, fundamentan su seguridad en el uso de problemas de búsqueda asociados a problemas de decisión. Dado que los problemas distribucionales de búsqueda y decisión son equivalentes en cuanto a la jerarquía deducida de este nuevo análisis, [BDCGL92], la utilización del análisis de la complejidad del caso medio proporciona una ventaja sobre la complejidad de caso peor.

1.2.4. Complejidad de las Comunicaciones

Dado que los algoritmos estudiados son en realidad protocolos se hace necesario comentar brevemente el modelo propuesto para su estudio en la llamada Complejidad de las Comunicaciones. El objetivo primordial de esta disciplina es analizar la cantidad de información transferida entre dos entidades que participan en un proceso conjunto persiguiendo una meta común. Lo que sucede realmente en dichas situaciones es que ninguno de los participantes pueden llevar a cabo de manera aislada las operaciones necesarias para resolver la situación en cuestión debido a la falta de información, información que sí posee su contrario.

El modelo básico para el análisis en este sentido es aquél en el que intervienen dos participantes, suponiéndose además que ambos poseen capacidades computacionales ilimitadas, y el problema que deben resolver se puede definir como el cómputo de una función predefinida sobre la entrada. Para precisar el concepto de protocolo según esta teoría es menester aclarar la notación utilizada. X e Y representan el dominio de cada uno de los participantes respectivamente. La función que deben computar se representa por $f(x, y)$.

Según la notación anterior un protocolo P con dominio $X \times Y$ es un algoritmo determinista en el que en cada paso se especifica el bit que va a ser enviado de un participante a otro. La salida del algoritmo, denotada por $P(x, y)$ es la salida de cada participante al final del protocolo $\forall x \in X,\ y \in Y :\ P(x, y) = f(x, y)$.

La complejidad de las comunicaciones para un protocolo determinista P, denotada por $CC(P)$, es el máximo número de bits transferidos por el protocolo.

La definición anterior se puede trasladar al caso probabilista si se permite que el protocolo sea incorrecto para una pequeña fracción de entradas. Para formalizar este concepto se debe tener en cuenta que ambos participantes tienen acceso a una sucesión de bits aleatorios y que la salida del protocolo depende de dichas cadenas.

Otra cuestión interesante que se puede distinguir es si la cadena aleatoria de cada participante es conocida por su contrario (modelo público) o si estas permanecen privadas (modelo privado). Debido a que ambos modelos son susceptibles de transformación recíproca [Gol99a], la definición de protocolo probabilista se restringe al modelo público.

La capacidad de aleatorización disponible para cada usuario se refleja como la existencia de una distribución de probabilidad uniforme según la cual se obtienen las cadenas que cada usuario usa.

Considerando todo lo anterior, se define un protocolo probabilista como un algoritmo que obtiene una cadena según una distribución de probabilidad μ y luego actúa como en el caso determinista. La cadena mencionada es común a ambos participantes. Además se debe verificar que $\forall x \in X,\ y \in Y\ Prob(P(x,y) = f(x,y)) \geq 1 - \epsilon$. En este caso se trata de un $\epsilon - protocolo$.

La complejidad de las comunicaciones de un protocolo probabilista P sobre la entrada (x, y) es el máximo número de bits transferidos por el protocolo para cualquier elección inicial de las cadenas aleatorias de cada participante. Generalizando, la complejidad de un protocolo probabilista P es la máxima complejidad de las comunicaciones sobre todas las entradas posibles.

Es importante considerar que el número de bits transferidos pueden variar para una misma entrada, dependiendo de la cadena aleatoria seleccionada. Esto se puede apreciar más claramente en los protocolos descritos en el capítulo 3.

En el siguiente apartado se enuncia cada uno de los problemas base que se usan en el resto de la memoria además de la información necesaria para ubicarlos en la jerarquía de la Complejidad Computacional.

1.3. Cuestiones de Teoría de Grafos

En el desarrollo de los protocolos propuestos en la presente memoria se ha usado la Teoría de Grafos como base. En esta sección se introducen aquellos conceptos de esta teoría necesarios para la especificación de los protocolos. Cuestiones que se tratan en sucesivos apartados son los grafos aleatorios, la generación de instancias aleatorias de los problemas utilizados (descritos en el segundo apartado) y la comprobación eficiente de propiedades en grafos.

Los nuevos métodos propuestos en esta memoria están basados en problemas planteados sobre grafos simples (grafos no dirigidos, sin aristas múltiples ni bucles) $G = (V, E)$ con n vértices. Este tipo de grafos pueden ser unívocamente descritos por los elementos situados por encima de la diagonal principal de su matriz de adyacencia, lo que permite representar estos elementos como una cadena binaria facilitando así la identificación de una propiedad de dicho grafo como un lenguaje binario.

Es conveniente indicar que no es ésta la primera vez que se emplean grafos como herramienta en criptografía. Merrit en [Mer83] plantea modelar los protocolos de reconstrucción de claves a través de grafos no dirigidos y con etiquetas en sus aristas, de manera que el papel de los participantes (procesadores) es representado por los vértices, las aristas conectan procesadores que se comunican y las etiquetas consisten en los mensajes intercambiados. Igualmente en [Fra93] aparece la Teoría de Grafos con un propósito muy particular, modelar la situación en la que la privacidad de una red está amenazada por un adversario hostil que es capaz de desplazarse por el sistema, atacando y espiando diferentes nodos cada vez. En 1991 aparecen resultados un poco más cercanos a los presentados en este trabajo. Broder, Frieze y Shamir [BFS91]

utilizan el problema del circuito hamiltoniano para definir un esquema de firmas. En su construcción parten de un circuito hamiltoniano al que sucesiva y aleatoriamente se le van añadiendo aristas. Esta construcción es un caso particular del método general propuesto más adelante denominado método constructivo de generación de instancias (ver 1.3.3). También Kučera en 1992 [Kuč92] plantea el uso del problema del conjunto independiente en un grafo como base para el diseño de un esquema de cifrado probabilista. Una de las principales aportaciones de la presente memoria es la propuesta de la Teoría de Grafos como fuente general de problemas base para el diseño de diversos protocolos criptográficos.

1.3.1. Grafos Aleatorios

El trabajo de Erdös [Erd59] fue uno de los primeros en los que se fusiona la Teoría de Grafos con la Probabilidad. En él se introducen los conceptos básicos a partir de los que se construyen dos de los modelos de grafos aleatorios más extendidos. Dichos modelos son los denotados por $G_{n,m}$ y $G_{n,p}$. En el primer caso se trata de grafos con n nodos y m aristas generadas por muestreo uniforme en el conjunto de todas las aristas posibles, mientras que en el segundo modelo se generan grafos con n vértices tales que la inclusión de cada arista se acepta con una probabilidad p.

A continuación se describe brevemente el tratamiento que reciben los grafos vistos como variables aleatorias en el modelo $G_{n,p}$.

Se parte del espacio muestral Ω_n definido como el conjunto de todos los grafos con n vértices, siendo la cardinalidad de dicho espacio 2^N, con $N = \begin{pmatrix} n \\ 2 \end{pmatrix}$.

A su vez se usa la probabilidad p referente a la existencia de aristas ya mencionada para definir la función de probabilidad asociada al espacio muestral Ω_n. Dicha función de probabilidad coincide con la obtenida al realizar N experimentos de Bernoulli, facilitando así el estudio desde el punto de vista estadístico. El número medio de aristas en un grafo de n vertices viene dado por la expresión $pn(n-1)/2$

Esta formalización permite estudiar las funciones umbral, noción descrita en el trabajo [Pal96] pero previamente introducida por Erdös y Rényi [PR59]. Las funciones umbral estudian la curva asociada a la función de probabilidad definida por la variable aleatoria que describe el comportamiento de una propiedad cualquiera de los grafos aleatorios en el modelo $G_{n,p}$. Dicha curva se estudia en función de la probabilidad p de manera que permite establecer valores umbrales de probabilidad para la verificación de propiedades.

Otro concepto que surge a través de esta aproximación es el de propiedad monótona. Se dice que una determinada propiedad P es monótona si al añadir aristas a un grafo que posee la propiedad P, ésta se sigue preservando. Uno de los resultados más destacables relacionado con las propiedades monótonas asegura la existencia de funciones umbral para toda propiedad monótona.

Otras ventajas asociada a la sencillez de este modelo es la posibilidad de estudiar algunas invariantes como es el caso del grado mínimo de los vértices o también el establecer resultados relacionados con la probabilidad de verificación de una determinada propiedad. Relacionado con este último tema es común el estudio del comportamiento asintótico, en función del número de vertices, de las propiedades en este modelo. Si la probabilidad anterior tiende a uno para determinada propiedad se dice que dicha propiedad es verificada casi seguro por cualquier grafo generado según $G_{n,p}$.

A pesar de las diferencias subyacentes al modelo de construcción se debe destacar que ambos modelos generan grafos con propiedades asintóticas similares para determinadas relaciones entre p y m, ($p = m/\begin{pmatrix} n \\ 2 \end{pmatrix}$) pero como se puede deducir de los comentarios anteriores el modelo $G_{n,p}$ es más sencillo de analizar.

En la siguiente sección se describen resultados relacionados con la eficiencia de algoritmos que resuelven los problemas allí descritos sobre el modelo $G_{n,m}$.

Algunos estudios experimentales tales como [Wal01] han demostrado que los grafos generados por los modelos previamente mencionados no poseen todas las características de los grafos asociados a problemas reales por lo que han surgido modelos alternativos con la finalidad de obtener una mejor modelización. Sin embargo, sí son modelos adecuados para el estudio de la eficiencia de algoritmos y para el diseño de los protocolos que se describirán en los capítulos sucesivos [BC94].

En este trabajo se usan grafos aleatorios para definir instancias difíciles de problemas usados posteriormente como información base en el desarrollo de los diferentes protocolos. Esto comporta el análisis de la idoneidad de los generadores en base a la dificultad de la instancia generada y no en base a otros criterios como los sugeridos en [Wal01], donde se realiza un estudio de diferentes modelos en función de las capacidades que poseen para representar situaciones del mundo real.

1.3.2. Catálogo de Problemas Utilizados

En este apartado se describen los enunciados de los problemas utilizados en los protocolos descritos y estudiados a lo largo de la memoria [CHB02a]. Para cada uno de estos problemas se indica cuál es la clase de la jerarquía de la complejidad computacional a la que pertenece [GJ79], junto con resultados conocidos para el caso de grafos aleatorios. Se debe mencionar que en todas las propuestas presentadas en esta memoria se usan los problemas de búsqueda asociados a las descripciones proporcionadas en la presente sección.

Los comentarios descritos en los apartados siguientes, referentes a la dificultad de los problemas usados cuando se plantean sobre grafos aleatorios se encuentran en [FM97] y deben tenerse en cuenta a la hora de facilitar los parámetros de entrada en la generación de instancias asociadas a los diferentes protocolos propuestos.

Problema del Isomorfismo de Grafos (GI, Graph Isomorphism)

El enunciado de este problema como problema de decisión se puede definir de la siguiente manera. Dada una instancia formada por dos grafos $G_0 = (V_0, E_0)$ y $G_1 = (V_1, E_1)$ la cuestión es si existe o no una función biyectiva $f : V_0 \longrightarrow V_1$ tal que $\{u, v\} \in E_0$ si y sólo si $\{f(u), f(v)\} \in E_1$.

Se cree que el problema del isomorfismo de grafos no pertenece ni a la clase P, ni a la clase $NP - Completa$. Su clasificación permanece abierta, aunque su resolución parece relativamente sencilla para grafos aleatorios o grafos relacionados estructuralmente hasta tal punto que para algunas clases especiales de grafos se han desarrollado algoritmos polinomiales. Por ejemplo, para grafos planares y grafos intervalos es resoluble en tiempo polinomial [GJ79]. Sin embargo, en general no es sencillo seleccionar la heurística adecuada para intentar una instancia concreta del problema ni para descubrir si determinado grafo pertenece a dichas clases especiales.

Finalmente, aunque encontrar instancias difíciles para este problema no es una tarea que pueda considerarse fácil, lo que sí es factible es transformar instancias que en principio pueden ser fáciles para obtener así instancias difíciles [For96a]. Una idea clave introducida en el trabajo anterior es el uso de grafos regulares disminuyendo la cantidad de similitudes seleccionando dos aristas aleatoriamente e intercambiando los vértices que las determinan teniendo cuidado de no modificar el grado de dichos vértices con esta operación. Una vez hecho esto, el problema del isomorfismo definido sobre la instancia resultante es considerablemente más difícil. Esta idea es utilizada en las implementaciones de varios protocolos propuestos en esta memoria basados en el problema del isomorfismo de grafos con el objeto de garantizar la dificultad del problema base.

Problema del Conjunto Independiente (IS, Independent Set)

En este caso se trata de responder a la pregunta siguiente: dado un grafo $G = (V, E)$ y un entero positivo k, '¿contiene G un subconjunto $V' \subseteq V$ tal que $V' \geq k$ y de manera que ningún par de vértices de V' estén unidos por una arista de E?'.

Este problema de decisión pertenece a la clase $NP - completa$ en el caso general. A continuación se enumeran algunos de los casos más significativos para los que dicho problema es resoluble en tiempo polinomial. Esto sucede para grafos bipartitos, grafos arista, grafos cordales y para todos aquellos grafos en los que los grados de todos sus vértices no es mayor que 2.

En cuanto a este problema en el entorno de los grafos aleatorios existen resultados bien conocidos que establecen una cota para la probabilidad de que un grafo generado según el modelo $G_{n,p}$ contenga un conjunto independiente. Concretamente $\forall n, k : 2 \leq k \leq n$ la probabilidad de que un grafo aleatorio G contenga un conjunto independiente de cardinal mayor o igual que k está acotada según la siguiente expresión: $Prob[\exists I \subseteq G, |I| \geq k] \geq \left(\begin{array}{c} n \\ k \end{array} \right) q^{\left(\begin{array}{c} k \\ 2 \end{array} \right)}$

Esta información debe tenerse en cuenta a la hora de generar los ejemplos a usar en los protocolos puesto que, si se escogen los parámetros de manera que dicho suceso ocurre con una probabilidad demasiado pequeña, podría incrementarse la probabilidad de que un adversario deshonesto cometiera una estafa.

Se ha estudiado el problema del conjunto independiente máximo sobre grafos aleatorios generados según el modelo $G_{n,p}$ comparando el tamaño del conjunto independiente generado por determinadas heurísticas voraces con la solución exacta. Sin embargo, lo que se puede afirmar es que no se conoce un algoritmo polinomial que determine un conjunto independiente cuyo tamaño sea al menos $(1/2 + \delta)\alpha_n$, siendo α_n el tamaño del conjunto independiente máximo y δ una constante positiva.

Se ha demostrado que este problema juega un papel importante en el diseño de

canales de transmisión [FW00]. La versión de búsqueda del conjunto independiente maximal se ha usado para implementar un esquema de cifrado probabilista [Kuč92]. Un método muy útil para esconder un conjunto independiente en un grafo resistente a aproximaciones heurísticas generales usado en las implementaciones realizadas en esta memoria se describe en [BC94].

Un problema que está directamente relacionado con el anterior es el del subgrafo completo maximal (clique). Dicho problema consiste en determinar la existencia de un subconjunto de vértices del grafo de entrada en el que cualquier pareja de vértices de este subconjunto están unidos por una arista. La relación mencionada es la siguiente: Dado un conjunto de vértices V' que determinan un conjunto independiente en un grafo $G = (V, E)$, ese mismo conjunto V' determina un subgrafo completo en el grafo complementario de G. (El grafo complementario de un grafo dado se define como $G^c = (V, E^c)$, siendo $E^c = \{(u, v)|u, v \in V : (u, v) \notin E\}$.

Problema de Circuito Hamiltoniano (HC, Hamiltonian Circuit)

El problema de decisión del circuito Hamiltoniano, consiste en responder a la pregunta de si un grafo de entrada G posee un circuito simple que contenga a todos los vértices, entendiendo por circuito simple aquél que no repite los vértices.

También este problema es $NP - completo$, incluso para grafos planares y grafos bipartitos. Por el contrario es resoluble en tiempo polinomial cuando no tiene vértices de grado mayor que dos.

Los estudios realizados sobre grafos aleatorios generados según $G_{n,m}$ incluyen un algoritmo determinista cuya complejidad es $O(n^3 \log n)$ y que con ciertas modificaciones es capaz de encontrar un circuito Hamiltoniano en $G_{n,1/2}$.

En cuanto al modelo $G_{n,p}$ se da demostrado la existencia de un algoritmo que en tiempo esperado lineal encuentra un circuito Hamiltoniano para los casos en que $p \geq n^{1/3}$.

Problema de la Coloración (GC, Graph Colorability)

Los datos asociados a la definición de este problema son un grafo $G = (V, E)$ y un entero k, tal que $k \leq \mid V \mid$, y la cuestión a resolver consiste en determinar una función $f : V \longrightarrow \{1, 2, \ldots, k\}$ que verifique que $f(u) \neq f(v), \forall \{u, v\} \in E$.

Este problema es polinomialmente resoluble cuando k es 2, pero es $NP-completo$ para cualquier valor de k fijo mayor que 3. Para el caso particular $k = 3$ está demostrado que pertenece a la clase polinomial si el grafo es planar y no posee vértices de grado mayor que 4. Finalmente se ha demostrado que el caso general $k > 3$ es polinomialmente resoluble para grafos que no poseen vértices cuyo grado sea mayor que 3.

Para el entorno de los grafos aleatorios se ha estudiado el problema de determinar la coloración mínima distinguiendo el caso de grafos densos y dispersos. En el primer conjunto, al igual que sucedía en el caso del problema del conjunto independiente, se ha estudiado el comportamiento de diversas heurísticas voraces concluyendo que la mayoría de ellas usan aproximadamente el doble de colores de los realmente necesarios. Además se conoce una cota inferior para el número cromático de un grafo G generado según el modelo $G_{n,p}$. Esta cota válida para cualquier valor de p y $\epsilon > 0$ viene dada por la expresión siguiente: $\chi(G) > \frac{\log(1/q)}{2+\epsilon} \cdot \frac{n}{\log n}$.

Problema del Logaritmo Discreto (DL, Discrete Logarithm)

El problema del logaritmo discreto se describe de la siguiente manera: Dado un número primo p, un generador de $\mathbb{Z}_p^*$ β, y un elemento $\alpha \in \mathbb{Z}_p^*$, se trata de encontrar el entero x, $0 \leq x \leq p-2$, tal que $\alpha^x \equiv \beta (mod\ p)$.

El mejor algoritmo para resolver el DL requiere un tiempo esperado subexponencial que viene expresado en función del tamaño del número primo p utilizado por $O(L[p, \sqrt{2}, 1/2]) = O(exp(\sqrt{2}\sqrt{\ln p \ln \ln p}))$.

No existe demasiada información sobre para qué valores de p este problema resulta

fácil de resolver. Se suele recomendar no utilizar valores de dicho parámetro tales que los factores primos de $p-1$ estén acotados [Bac89].

1.3.3. Generación de Instancias

La generación de instancias difíciles de resolver es una materia imprescindible en cualquiera de los apartados que comprende la Criptografía. No obstante no son muchos los trabajos desarrollados en este sentido. En el ámbito de la complejidad del caso medio se tienen los trabajos publicados [IL90] y [VL88] (ver sección 1.2.3). Algunos resultados teóricos en sintonía con la generación y el uso de instancias difíciles en criptografía aparecen en [AAB$^+$89]. Los resultados allí expuestos establecen la relación entre la existencia de generadores de instancias difíciles y la jerarquía de la Complejidad Computacional.

En este subapartado se incluyen las descripciones detalladas de las implementaciones desarrolladas en el lenguaje ANSI C (ver Apéndice A) para la generación de instancias, junto con la descripción de los formatos de los ficheros de salida utilizados en la implementación de los protocolos propuestos a lo largo de la presente memoria.

Se estudia aquí cómo se ha llevado a cabo la generación de grafos aleatorios distinguiendo el caso en el que sólo se genera un grafo cualquiera, de aquel en el que se genera un grafo incluyendo una solución de un determinado problema.

Generador de grafos aleatorios

Los grafos $G = (V, E)$ utilizados en este trabajo son todos grafos simples y no dirigidos. Cuando se trata de generar uno de estos grafos usamos como punto de partida el modelo $G_{n,p}$. Es decir, se supone que el usuario fija el número n de vértices del grafo y a partir de ahí se utiliza un registro de desplazamiento con realimentación no lineal adecuado para generar las aristas como pares de números enteros aleatorios en el rango $\{1, 2, \ldots, n\}$ en función de una probabilidad también proporcionada como

entrada.

En el caso en el que el protocolo requiera la generación de un grafo que contenga una solución a un determinado problema se proponen a continuación [CH01] tres alternativas generales. La semejanza entre ellas radica en el uso del modelo $G_{n,p}$ como base y en el hecho de que todas comienzan con la definición, bien sea aleatoria o bien sea por parte del usuario, de los elementos que formarán parte de de la solución. Por contra, las diferencias aparecen a la hora de decidir cómo comienza la construcción. Una vez se ha elegido un problema P, un número de vértices n y una solución S, el usuario $\mathcal{A}$ construye un grafo G donde S es una solución del problema P. Los métodos de construcción de la instancia responden a las siguientes opciones:

1. Método Constructivo (CM, Constructive Method): consiste en partir del grafo nulo N_n e ir añadiendo aristas generadas aleatoriamente siempre que con ello de manera que no se violen las restricciones del problema asociado. Este es el método básico que se ha usado en las implementaciones aquí incluidas.

2. Método Destructivo (DM, Destructive Method): consiste en partir del grafo completo K_n e ir eliminando aristas escogidas al azar siempre que con ello no se violen las restricciones del problema asociado.

3. Método de Inserción (IM, Insertion Method): este método es una mixtura de los dos anteriores puesto que el proceso en este caso consiste en generar en principio un grafo aleatorio $G_{n,m}$ o $G_{n,p}$, y posteriormente, y sólo si es necesario, calcular cuál es el menor conjunto de aristas a incluir y/o a eliminar para que una solución válida quede insertada en el grafo. Para realizar esta construcción se puede usar la definición de distancia entre grafos que aparece en el ámbito de la comprobación de propiedades en grafos [Ron01]. La cuantificación de distancia se realiza comparando el número de posiciones de la estructura de datos

seleccionada para la representación (matriz de adyacencia o lista de adyacencia) que difieren al comparar el grafo de entrada con otro grafo de las mismas dimensiones para el que se conoce una solución.

Este paso de construcción es probablemente el más costoso del protocolo. Por ello debe ser eficientemente implementado para controlar el cómputo off-line. En las implementaciones prácticas se ha optado por usar como formato de los ficheros que contienen las instancias es el utilizado en $DIMACS$ Challenge 1993, descrito a continuación.

Es imprescindible la presencia de una línea reservada a la descripción del tamaño del grafo con el formato *p aristas NUMERONODOS NUMEROARISTAS*. La descripción de cada una de las aristas aparece una única vez en líneas diferentes, con el formato *e VERTICE1 VERTICE2*.

1.3.4. Comprobación de Propiedades

Recientemente se han desarrollado estudios teóricos amplios [Gol02a], [Ron01], [GT01], [Gol97] sobre la formalización de técnicas generales que permitan la comprobación de determinadas propiedades en grafos intentando lograr el equilibrio entre la eficiencia y la exactitud. Estos métodos están especialmente indicados cuando el tamaño de la instancia correspondiente es considerable.

La toma de decisiones se realiza a partir de una función distancia definida sobre el conjunto de aristas del grafo en cuestión. Esta distancia proporciona información sobre el número de aristas que habría que modificar en este grafo para que la propiedad cuestionada fuese verificada, de manera que si dicha distancia supera un valor umbral que depende del número de vértices, se concluye que el grafo no posee la propiedad evaluada.

La manera general de proceder está directamente relacionada con la estructura utilizada para la representación del grafo a examinar. Así, si se utiliza la matriz de

adyacencia, se selecciona aleatoriamente un subconjunto pequeño de vértices y se estudia si el subgrafo inducido mantiene la propiedad estudiada y en caso afirmativo se concluye que el grafo original también la verifica. En caso de que el grafo sea disperso y se opte por representarlo a través de su lista de adyacencia, se proponen métodos basados en búsquedas exhaustivas locales puesto que la selección aleatoria de subgrafos en este escenario puede conllevar la generación de subgrafos no representativos. Se suelen ejecutar como primera aproximación, utilizando la salida de dicho procedimiento para decidir posteriormente si se ejecuta el procedimiento determinista o no.

La representación utilizada en las implementaciones de los protocolos aportados en este trabajo se basa en la mayoría de casos en el modelo de la matriz de adyacencia puesto que generalmente se necesita conocer si determinados vértices son adyacentes. Además se trabaja con grafos de tamaño considerable por lo que la técnica tratada en este apartado tiene aplicaciones en este ámbito. Las aplicaciones a considerar tienen tres vertientes. La primera es la posibilidad de acometer de esta manera los procedimientos de verificación. La segunda es su utilización en ataques desarrollados por los usuarios. Por último, cabe la posibilidad de controlar si el comportamiento de los participantes en un determinado protocolo es honesto o no, verificando si la información suministrada (en este caso los grafos) tiene las propiedades requeridas. No se ha optado por esta forma de proceder en los procedimientos de verificación para no introducir una fuente de incertidumbre adicional en el desarrollo de la verificación (se debe recordar que los protocolos estudiados son probabilistas). Además, el uso de problemas de la clase NP garantiza que el procedimiento de verificación es polinomial. En cuanto al desarrollo de ataques basados en la comprobación de las propiedades de los grafos utilizados se debe puntualizar que realmente se usan problemas de búsqueda y no problemas de decisión.

Capítulo 2

Protocolos Bipartitos

Los protocolos bipartitos han sido probablemente los estudiados con mayor intensidad y profundidad en los últimos años. Este tipo de protocolos tiene la finalidad de construir soluciones a problemas en los que solamente se ven envueltos dos participantes, siendo algunos de ellos generalizables a un entorno más complejo con un número mayor de participantes. Tal y como se ha venido haciendo a lo largo del Capítulo 1, en éste se aludirá a dichos participantes usando los nombres de Alicia ($\mathcal{A}$) y Bernardo ($\mathcal{B}$).

En la mayoría de las alternativas analizadas en esta sección aparecen dos herramientas fundamentales ya descritas en el capítulo anterior: la técnica de corte y elección y la de reto-respuesta. También características comunes a la mayoría de los algoritmos descritos en este capítulo son la interacción y la necesidad de que su ejecución sea en tiempo real (on-line) para su correcta implementación.

Se comenzará estudiando dos de las principales primitivas criptográficas, la transferencia inconsciente (sección 2.1) y el compromiso de bits (sección 2.2). En la presente memoria se usa el término primitiva criptográfica de forma restringida para referirse a aquellos basados en operaciones básicas de propósito general y que se utilizan para la construcción de protocolos más complejos, [Bel98]. Sin embargo, hay que aclarar que en el marco de la criptografía general en muchos casos se utiliza el mismo vocablo

para referirse a los esquemas de cifrado y firma digital.

No existen muchos trabajos en la bibliografía que traten el tema de los protocolos bipartitos como conjunto, [Mer83], aunque sí se pueden encontrar muchos trabajos en los que se estudian concretamente las dos primitivas criptográficas mencionadas (transferencia inconsciente y compromiso de bits), [CGT95], [Cré93]. Algunos trabajos importantes en los que se establecen protocolos basados en ellas y se analizan las relaciones existentes entre los mismos son [BVV84], [KKMO94]. Es de destacar el análisis de Cramer, [Cra99] en el que se hace un completo recorrido por las distintas definiciones de transferencia inconsciente, y se relaciona este protocolo con el problema de la computación bipartita segura. Aparte de esto, también existen algunos libros que engloban un mayor conjunto de protocolos destacando la importancia de estas primitivas, [Sch93], [GB01].

En cada una de las siguientes secciones se proponen nuevas alternativas para cada una de las dos primitivas criptográficas y algunas de sus variantes, así como para otros protocolos bipartitos más complejos tales como la firma de contratos y el lanzamiento de monedas.

2.1. Transferencia Inconsciente

En esta sección se proponen nuevos algoritmos basados en grafos para la principal primitiva del diseño de protocolos criptográficos, conocida como transferencia inconsciente, así como para las principales variantes de su definición básica.

Definición 2.1.1. Transferencia Inconsciente (OT, Oblivious Transfer)
El protocolo conocido como de transferencia inconsciente, también llamado en ocasiones transferencia trascordada, resuelve la siguiente situación: $\mathcal{A}$ conoce un secreto que desea transferir a $\mathcal{B}$ de manera probabilista, y verificando las dos propiedades descritas a continuación:

- *Significación*: $\mathcal{B}$ obtiene el secreto con probabilidad 1/2.
- *Inconsciencia*: $\mathcal{B}$ es consciente con toda certeza de si recibió el secreto o no, mientras que $\mathcal{A}$ sólo puede adivinar con probabilidad 1/2 si la transferencia tuvo éxito.

Las propiedades anteriores se pueden entender como adaptaciones del concepto de corrección y privacidad, de obligado cumplimiento en todo protocolo (ver 1.1.1), para este protocolo particular.

A continuación se describen algunas de las variantes más interesantes de dicho protocolo. La primera de ellas es la denominada Transferencia Inconsciente 1-2.

Definición 2.1.2. Transferencia Inconsciente 1 de 2 ($OT1-2$, Oblivious Transfer 1-out-of-2)
En una $OT1-2$ la usuaria $\mathcal{A}$ dispone de dos secretos de los cuales, al final de la ejecución del protocolo $\mathcal{B}$ debe haber obtenido exactamente uno de ellos sin que $\mathcal{A}$ sepa cuál.

Definición 2.1.3. Transferencia Inconsciente 1 Seleccionada de 2 ($OT1C-2$, Oblivious Transfer 1 Chosen from 2)
Si $\mathcal{B}$ no realiza su elección de manera totalmente aleatoria, se habla de *transferencia inconsciente seleccionada 1 de 2*.

La primera variante se puede utilizar cuando $\mathcal{A}$ posee dos secretos y $\mathcal{B}$ desea obtener uno de de ellos sin permitir que $\mathcal{A}$ conozca cuál de ellos es. La diferencia con una $OT1C-2$ consiste en que en este último caso se entiende que $\mathcal{B}$ está particularmente interesado en uno de los dos secretos. En estos dos protocolos las propiedades mencionadas para la definición de transferencia inconsciente general se corresponden con las mostradas a continuación:

- *Significación*: $\mathcal{B}$ obtiene exactamente uno de los dos secretos.

- *Inconsciencia*: $\mathcal{B}$ es consciente de cuál es el secreto recibido, pero $\mathcal{A}$ no.

La generalización de la $OT1C-2$ a un número mayor de secretos se conoce habitualmente como *venta de secretos*.

Haciendo uso de la definición formal 1.1.2 de protocolo incluida en el capítulo 1, la función que modela estas variantes se puede expresar de la siguiente manera:

$$f(b_0, b_1, s) = (0, b_s) \text{ donde } b_s = \begin{cases} b_0 & si\ s = 0 \\ b_1 & si\ s = 1 \end{cases},\ b_0,\ b_1 \in \{0, 1\}$$

La variante que se describe a continuación fue definida por Crépeau y otros en [CGT95] y tiene aplicaciones directas en la computación multipartita segura. Este protocolo hace uso del concepto de $OT1C-2$ y del de compromiso de bits que se verá en la próxima sección, obteniendo así un nuevo protocolo que fusiona los dos anteriores. Su objetivo es desarrollar una transferencia inconsciente a partir de que ambos usuarios están comprometidos con determinados valores.

Definición 2.1.4. Transferencia Inconsciente Comprometida (COT, Committed Oblivious Transfer)
Al comienzo del protocolo la usuaria $\mathcal{A}$ posee dos bits con los que se ha comprometido, a_0 y a_1, mientras que el usuario $\mathcal{B}$ está comprometido con el bit b. Una vez desarrollado el protocolo, $\mathcal{B}$ queda comprometido con el bit a_b, sin tener información alguna sobre el otro bit $a_{\bar{b}}$ (siendo $\bar{b} = b + 1(\mod 2)$) y sin que $\mathcal{A}$ reciba información alguna acerca el bit recibido.

2.1.1. Estado del Arte

Fue Rabin en 1981 quien primero propuso, en su trabajo [Rab81], la definición de este protocolo, convirtiéndose con el tiempo en primitiva indispensable [Kil88] para otros protocolos importantes que resuelven problemas tales como el lanzamiento de monedas, la firma de contratos y el correo certificado. De esta manera, el poder de este protocolo radica en su relación con el problema de la computación multipartita

segura [Cra99], [Fra93] y la multitud de aplicaciones que posee. En un trabajo de 1982 Even y otros [EGL82] van más allá incluyendo en la definición las tres propiedades siguientes:

1. El usuario $\mathcal{B}$ debe poder comprobar el contenido del secreto una vez transferido.

2. Si la usuaria $\mathcal{A}$ es honesta, la probabilidad a priori de que $\mathcal{B}$ obtenga el secreto es 1/2, y la probabilidad a posteriori de que $\mathcal{A}$ sepa si $\mathcal{B}$ obtuvo el secreto es también 1/2.

3. Las estafas cometidas por la usuaria $\mathcal{A}$ son detectadas por $\mathcal{B}$ con probabilidad 1/2.

A pesar de ser uno de los primeros protocolos propuestos, su importancia para el diseño de otros protocolos más complejos hace que sea un tema de investigación aún abierto como lo demuestra la aparición de publicaciones recientes en el tema como [MZV02], [NNPV02], [NP01], en las que se plantean nuevos protocolos de transferencia inconsciente con diferentes aplicaciones. En los dos primeros trabajos citados anteriormente se plantean nuevos esquemas de Transferencia Inconsciente 1 de 2 mientras que [NP01] se centra en presentar alternativas que mejoren la eficiencia de las propuestas existentes y para ello hace uso de las funciones hash.

En su trabajo, Rabin además de introducir el concepto propuso un protocolo utilizando los problemas de la factorización y de los restos cuadráticos. Ambos problemas pertenecen a la Teoría de Números y los problemas de decisión a ellos asociados se conjeturan pertenecientes a la clase NPI. En [BPT84] se propone una alternativa de OT que solventa algunas dificultades que presentaba la propuesta de Rabin basada también en el problema de la factorización. Fisher, Micali y Rackof [FMR96] ponen de manifiesto que en el protocolo de Rabin un participante deshonesto que intervenga en el papel de $\mathcal{B}$ puede estafar a su contrario si selecciona la información a enviar de

manera determinada. En este mismo trabajo se propone una manera de resolver este problema.

Es conveniente indicar que en algunos trabajos como [Cra99] se utiliza la variante $OT1-2$ como definición básica de transferencia inconsciente, mientras que en otros como [EGL82] no sucede así. En cualquier caso, Crepeau [Cré87], [Cré90] demostró que ambas nociones son equivalentes puesto que son mutuamente susceptibles de ser simuladas mediante la definición alternativa.

Brassard y otros en [BCR86] definen la idea de *all-or-nothing disclosure*, concepto equivalente a la noción de $OT1C-2$, planteando una solución basada también en el problema de los restos cuadráticos. Posteriormente, se demuestra la equivalencia entre las definiciones de OT existentes utilizando para ello una generalización de este concepto [Cré87]. Aunque la mayoría de protocolos de OT propuestos hacen uso de la interacción entre los participantes, también se han diseñado versiones no interactivas, ejemplo de ello es la propuesta que aparece en [BM89]. En este protocolo se hace uso de la infraestructura de clave pública planteando además como aplicación su uso en las demostraciones de conocimiento nulo no interactivas que permiten la participación de múltiples probadores y verificadores.

Otro ejemplo de la aplicación del protocolo de transferencia inconsciente es la construcción de protocolos de compromiso de bits. Un ejemplo concreto de esta construcción se plantea en [dB90]. Sin embargo, un año más tarde se demostró que este protocolo poseía un fallo puesto que permite que la usuaria $\mathcal{A}$ elija la información que determina cuál será el bit seleccionado por $\mathcal{B}$. También han surgido diferentes propuestas para este protocolo en el entorno de la Criptografía Cuántica, [Cré93]. Beaver en [Bea95] plantea modificaciones sobre esquemas de OT existentes con el objetivo de mejorar la eficiencia reduciendo la necesidad de computaciones en tiempo real. También la Teoría de la Codificación ha servido como base en el diseño de esta primitiva. Concretamente Crépeau y otros [CGT95] usan propiedades de los códigos

correctores de errores para proponer un protocolo de COT. Un algoritmo de OT cuya implementación se basa en cualquier esquema de clave pública puede encontrarse en [Pfl96]. Por último un interesante campo de aplicación de los protocolos de OT es la protección de la propiedad intelectual, tal y como se observa en una propuesta concreta que se encuentra en [DF99].

2.1.2. Esquema General

En la mayoría de los protocolos de OT existentes, el secreto a transferir es una solución a un determinado problema difícil. Estos algoritmos probablemente seguros se pueden describir según el esquema general $OT - GS$ (Oblivious Transfer-General Scheme) presentado en la figura 2.1, [HC03a]. Este esquema unilateral probabilista

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ una bipartición de una instancia del problema base $\{P_0, P_1\}$.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ la imagen obtenida a través de una función unidireccional h de un elemento, seleccionado al azar, perteneciente a uno de los dos subconjuntos de la partición anterior $b_j \in P_j, j \in \{0, 1\}$, $h(b_j)$.

Respuesta: $\mathcal{A} \longrightarrow \mathcal{B}$ la solución del problema definido por su elección aleatoria de un elemento de uno de los dos conjuntos, $a_i \in P_i, i \in \{0, 1\}$ y la información enviada por $\mathcal{B}$, $Sol(a_i, h(b_j))$.

Verificación: La solución secreta es transferida con éxito a $\mathcal{B}$ dependiendo de la coincidencia o no de las elecciones de ambos participantes.

Figura 2.1: Esquema OT-GS

verifica las propiedades de significación e inconsciencia ya mencionadas. Ambos participantes obtienen la salida de la función f que modela el protocolo, por lo que la corrección queda garantizada cuando ambas partes actúan en el modelo semi-honesto. Con respecto a la tolerancia a fallos, en el caso de que $\mathcal{A}$ trate de transferir una solución secreta inexistente, el arbitraje de una TTP o el uso de una Demostración de

Conocimiento Nulo (ver Capítulo 3) se puede incorporar para garantizar la corrección. En cuanto a la privacidad, se debe destacar que después de la participación de $\mathcal{B}$ en el protocolo no existe manera alguna de que pueda obtener el secreto de $\mathcal{A}$ a menos que éste se lo haya transferido. Esto sucede gracias a que $\mathcal{B}$ posee recursos computacionales acotados polinomialmente, con lo que le resulta imposible resolver la instancia del problema base. Además, $\mathcal{A}$ no puede averiguar la elección que $\mathcal{B}$ ha realizado secretamente, con lo que no podrá determinar si $\mathcal{B}$ obtuvo la solución secreta o no.

Las adaptaciones concretas realizadas sobre este esquema en las propuestas descritas en secciones posteriores basadas en grafos se pueden resumir de la manera siguiente. La fase de configuración requiere la generación de un grafo suficientemente grande y adecuado como instancia difícil del problema elegido como base de forma que garantice la privacidad del protocolo. En la fase de reto, el usuario $\mathcal{B}$ debe generar una permutación sobre el conjunto de vértices, consiguiendo con ella construir un grafo isomorfo al de partida. En cuanto a la respuesta de $\mathcal{A}$ que implica la solución de un problema de grafos, ésta es factible gracias a alguna hipótesis acerca de su poder computacional. Finalmente para obtener un proceso de verificación adecuado se deben especificar claramente cuáles son los cálculos a realizar por $\mathcal{B}$ en esta última fase, dada su capacidad limitada de cómputo.

2.1.3. Algoritmo OT-GI

Nuestra primera propuesta de *OT* [CHB02b] hace uso del problema del isomorfismo de grafos descrito en el apartado 1.3.2 como único problema base. El objetivo del algoritmo $OT - GI$ (Oblivious Transfer-Graph Isomorphism) presentado en la figura 2.2 consiste en transferir con probabilidad $1/2$ un isomorfismo secreto g, definido entre dos grafos preseleccionados, G_0 y G_1.

El proceso utilizado por $\mathcal{A}$ para obtener la pareja anterior de grafos es totalmente

constructivo en el sentido de que el proceso consiste en que $\mathcal{A}$ genera un grafo aleatorio, y a partir de él genera otro, tal y como se describió en el apartado 1.3.2, para que el nuevo grafo resultante verifique las propiedades correspondientes del isomorfismo y que el problema del isomorfismo resulte difícil de resolver para la instancia generada.

Este esquema se puede clasificar como ilimitado-limitado, ya que se asume que la capacidad computacional de $\mathcal{A}$ le permite resolver el problema del isomorfismo para cualesquiera copias isomorfas de G_0 y G_1, mientras que las capacidades de $\mathcal{B}$ están polinomialmente limitadas. El hecho de utilizar hipótesis de este tipo, es decir relacionadas con las habilidades computacionales de los participantes, es inevitable tal y como se justifica en [Cra99]. De acuerdo con la notación del esquema general

Configuración: $\mathcal{A} \longrightarrow B$ los grafos G_0 y G_1.

Reto: $\mathcal{B} \longrightarrow A$ un grafo H, isomorfo a uno de los originales, $G_j, j \in \{0,1\}$.

Respuesta: $\mathcal{A} \longrightarrow B$ el isomorfismo definido entre H y uno de los grafos originales, G_i, $i \in \{0,1\}$.

Verificación: Si $i \neq j$, el secreto es transferido a $\mathcal{B}$, en caso contrario no se transfiere información sensible.

Figura 2.2: Algoritmo OT-GI

$OT-GS$ (ver figura 2.1) se deben aclarar algunas correspondencias:

- La partición de la instancia del problema en $\{P_0, P_1\}$ viene dada en este caso por los grafos $\{G_0, G_1\}$.
- La elección aleatoria de $\mathcal{B}$ se corresponde con el isomorfismo existente entre $h_j = G_j \sim H, j \in \{0,1\}$ donde H es un grafo isomorfo a G_0 y a G_1 que representa $h(b_j)$ en el esquema anterior.
- El elemento $Sol(a_i, h(b_j)), i \in \{0,1\}$, se refiere al isomorfismo $h_i = G_i \sim H$.

Es fácil observar que al acabar la transmisión sólo en el caso de que i no coincida con j, $\mathcal{B}$ podrá determinar el isomorfismo secreto g a través de la composición entre h_0 y h_1. Para ilustrar el desarrollo del protocolo se muestra un ejemplo del planteamiento del mismo en un gráfico (ver figura 2.3).

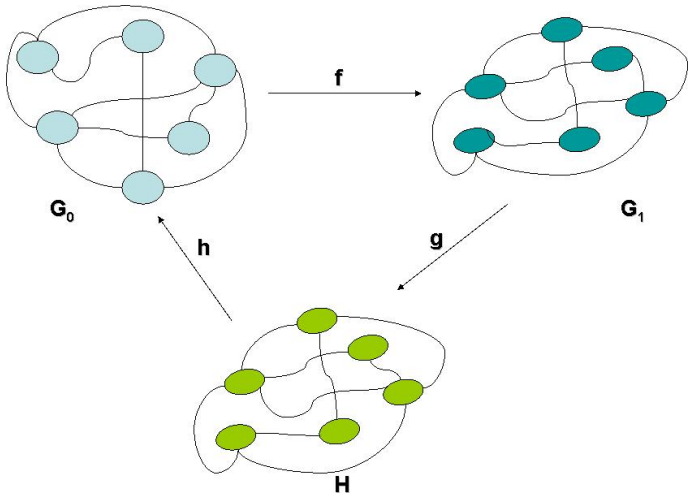

Figura 2.3: Ejemplo de Algoritmo OT-GI

La ejecución correcta del protocolo sólo depende de las elecciones aleatorias hechas por los usuarios, de manera que si ambos poseen generadores seudoaleatorios correctos, el desarrollo del protocolo resulta satisfactorio para ambas partes implicadas. A continuación se demuestra la verificación de las propiedades de corrección y privacidad bajo el modelo semi-honesto de participantes.

Teorema 2.1.1. *El algoritmo $OT-GI$ es una Transferencia Inconsciente.*

Demostración. El algoritmo descrito satisface los dos requisitos de la definición de transferencia inconsciente. Primero, la propiedad de significación se cumple porque la probabilidad de que las elecciones aleatorias de $\mathcal{A}$ y $\mathcal{B}$ (i y j) no coincidan es $Prob(i \neq j) = 1/2$, y sólo en tal caso $\mathcal{B}$ puede obtener la composición de los dos

isomorfismos por él conocidos tras el tercer paso del protocolo $G_0 \sim H$ y $H \sim G_1$ para construir el isomorfismo secreto $G_0 \sim G_1$. Respecto a la propiedad de inconsciencia, ésta también se verifica puesto que $\mathcal{A}$ no puede averiguar la elección secreta de $\mathcal{B}$, $j \in \{0, 1\}$ a partir del grafo $H \sim G_j$, ya que G_0 y G_1 son isomorfos. □

A continuación se analiza la tolerancia a fallos describiéndose lo que sucede cuando uno de los participantes actúa de manera deshonesta.

- Estafa de la usuaria $\mathcal{A}$: Que $\mathcal{A}$ no sepa resolver realmente el problema del isomorfismo para todos los grafos isomorfos a G_0 y G_1 y/o que G_0 y G_1 no sean isomorfos es fácilmente detectable por $\mathcal{B}$ ya que la función h_i recibida en el paso de respuesta ni coincidiría con h_j, ni le permitiría obtener g mediante la composición con h_j. Por otra parte, puesto que $\mathcal{A}$ no es capaz de determinar cuál fue el grafo seleccionado por $\mathcal{B}$ para construir H, tampoco sabrá si éste coincide con el elegido por él en el reto planteado.

- Estafa del usuario $\mathcal{B}$: Si el grafo que $\mathcal{B}$ construye no es isomorfo a ninguno de los de partida, $\mathcal{A}$ lo detecta ya que sabe resolver el problema del isomorfismo.

2.1.4. Algoritmo OT-GP

El esquema de la sección anterior puede ser generalizado a una transferencia inconsciente donde el secreto es una solución a un problema difícil cualquiera denotado por P en un grafo G, $Sol_P(G)$ (ver Figura 2.4) [HC02]. De nuevo la corrección del algoritmo OT-GP (Oblivious Transfer-Graph Problem) recae sobre la hipótesis de que los recursos computacionales disponibles para $\mathcal{B}$ son limitados, pero por el contrario, las habilidades computacionales de $\mathcal{A}$ le permiten resolver el problema P en todo grafo isomorfo al grafo de partida G. Entre los problemas más adecuados para los protocolos propuestos en este y el siguiente apartado destacan los problemas $NP - completos$ mencionados en el apartado 1.3.2.

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ el problema P, el grafo G y dos copias isomorfas G_0 y G_1.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ un grafo H, isomorfo a uno de los grafos originales $G_j, j \in \{0,1\}$.

Respuesta: $\mathcal{A} \longrightarrow \mathcal{B}$ la solución $Sol_P(H)$ y uno de los isomorfismos $G_i \sim G, i \in \{0,1\}$.

Verificación: Si $i = j$, la solución secreta es transferida a $\mathcal{B}$, en otro caso no.

Figura 2.4: Algoritmo OT-GP

El algoritmo $OT - GP$ puede describirse formalmente realizando las siguientes correspondencias con la notación usada en el esquema general $OT - GS$ de la figura 2.1:

- La partición $\{P_0, P_1\}$ viene descrita por ambos grafos $\{G_0, G_1\}$.
- La elección de $\mathcal{B}$, b_j es el isomorfismo $G_j \sim H, j \in \{0,1\}$, y el reto $h(b_j)$ es el grafo H.
- La respuesta de $\mathcal{A}$, $Sol(a_i, h(b_j))$, consiste en la solución $Sol_P(H)$ y el isomorfismo $G_i \sim G, i \in \{0,1\}$.

En la figura 2.5 se puede apreciar esquemáticamente un ejemplo de este protocolo.

Teorema 2.1.2. *El algoritmo $OT - GP$ es una Transferencia Inconsciente.*

Demostración. La probabilidad de que $\mathcal{A}$ y $\mathcal{B}$ seleccionen el mismo grafo es $Prob(i = j) = 1/2$. Así, dado que $\mathcal{B}$ puede obtener $Sol_P(G)$ a partir de $G_j \sim H$, $G_i \sim G$ y $Sol_P(H)$ si y sólo si $i = j$, la propiedad de significación se cumple. Con respecto a la inconsciencia, $\mathcal{A}$ no puede averiguar la elección realizada por $\mathcal{B}$, $j \in \{0,1\}$ a partir del grafo $H \sim G_j$, ya que G_0 y G_1 son isomorfos. □

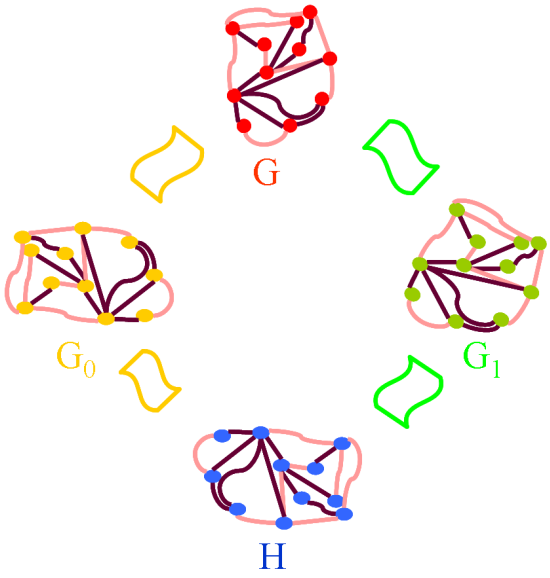

Figura 2.5: Ejemplo de Algoritmo OT-GP

2.1.5. Algoritmos OT1-2-GI y OT1C-2-GI

Los algoritmos descritos en apartados anteriores se corresponde con la versión más sencilla de la definición de Transferencia Inconsciente. Como ya se ha mencionado en la introducción de esta sección existen dos variantes especialmente interesantes denominadas transferencia inconsciente 1 de 2 ($OT1-2$) y transferencia inconsciente 1 seleccionado de 2 ($OT1C-2$).

El esquema $OT-GS$ continúa siendo válido para los algoritmos $OT1-2$ y $OT1C-2$ propuestos a continuación, realizando algunas ligeras modificaciones en la etapa de verificación donde se entiende que esta vez siempre un secreto es recibido por $\mathcal{B}$.

Los algoritmos mostrados en las figuras 2.6 y 2.8 describen respectivamente una $OT1-2$ y una $OT1C-2$ para el isomorfismo de grafos. En ambos casos, los dos secretos son los isomorfismos $G_0 \sim H$ y $H \sim G_1$, donde G_0, G_1 y H son grafos isomorfos generados por $\mathcal{A}$ [HC03a]. La implementación es posible bajo la hipótesis

de que $\mathcal{A}$ es es capaz de resolver el problema del isomorfismo para cualquier copia isomorfa de $G_0,\ G_1$ y H, mientras que $\mathcal{B}$ debe ser polinomialmente acotado. En el

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ los grafos $G_0,\ G_1$ y H.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ los grafos $H_0,\ H_1$ (en orden aleatorio) y H', isomorfos respectivamente a los grafos originales $G_0,\ G_1$ and H.

Respuesta: $\mathcal{A} \longrightarrow \mathcal{B}$ el isomorfismo $H_i \sim H'$, $i \in \{0, 1\}$.

Verificación: $\mathcal{B}$ calcula el isomorfismo secreto $G_i \sim H$ correspondiente a la composición de los tres isomorfismos $G_i \sim H_i, H_i \sim H'$ y $H' \sim H$.

Figura 2.6: Algoritmo OT1-2-GI

algoritmo $OT1-2-GI$ la notación se entiende de la siguiente manera:

- La partición de la instancia del problema $\{P_0, P_1\}$ viene determinada por ambos isomorfismos secretos $G_0 \sim H$ y $G_1 \sim H$.
- La elección aleatoria de $\mathcal{B}$, b_j está definida por los tres isomorfismos $G_0 \sim H_0$, $G_1 \sim H_1$, y $H \sim H'$. Además, el reto $h(b_j)$ está formado por los tres grafos $H_0,\ H_1$ y H'.
- La respuesta $Sol(a_i, h(b_j))$ es el isomorfismo $H_i \sim H'$.

El esquema contenido en la figura 2.7 ilustra un ejemplo de este protocolo.

Teorema 2.1.3. *El algoritmo $OT1-2-GI$ es una Transferencia Inconsciente 1 de 2.*

Demostración. La propiedad de significación se verifica debido a que $\mathcal{B}$ obtiene de $\mathcal{A}$ el isomorfismo $H_i \sim H'$ a través del algoritmo, el cual junto con dos de los tres isomorfismos generados por él, $G_0 \sim H_0$, $G_1 \sim H_1$ y $H' \sim H$, le permiten calcular

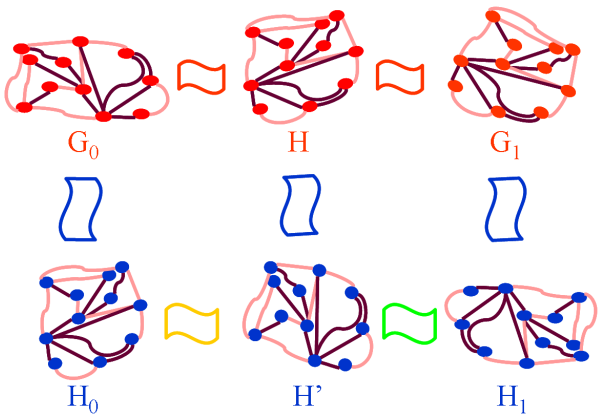

Figura 2.7: Ejemplo de Algoritmo OT1-2-GI

exactamente uno de los dos isomorfismos secretos $G_i \sim H$. La propiedad de inconsciencia se verifica ya que los grafos H_0 y H_1 se envían en orden aleatorio por lo que $\mathcal{A}$ no es capaz de averiguar qué grafo G_0 ó G_1 usó $\mathcal{B}$ para generar cada uno. □

En el caso de la transferencia inconsciente 1 seleccionado de 2 el algoritmo se simplifica por la elección no aleatoria por parte de $\mathcal{B}$ del isomorfismo secreto que quiere recibir. La descripción del algoritmo se puede ver en la figura 2.8. Siguiendo la

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ los grafos G_0, G_1 y H.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ un grafo H', isomorfo a uno de los grafos originales, $G_i, i \in \{0, 1\}$.

Respuesta: $\mathcal{A} \longrightarrow \mathcal{B}$ el isomorfismo $H \sim H'$.

Verificación: $\mathcal{B}$ calcula el isomorfismo secreto $G_i \sim H$ usando la composición de ambos isomorfismos $G_i \sim H'$ y $H' \sim H$.

Figura 2.8: Algoritmo OT1C-2-GI

notación general, el algoritmo $OT1C-2-GI$ puede ser definido como sigue:

- La instancia del problema $\{P_0, P_1\}$ esta determinada por los isomorfismos $\{G_0 \sim H, G_1 \sim H\}$.
- La elección de $\mathcal{B}$, b_j está formado por el isomorfismo $G_i \sim H'$ y el reto $h(b_j)$ es el grafo H'.
- La respuesta de $\mathcal{A}$, $Sol(a_i, h(b_j))$, es el isomorfismo $H \sim H'$.

Teorema 2.1.4. *El algoritmo $OT1C-2-GI$ es una Transferencia Inconsciente 1seleccionado de 2.*

Demostración. La propiedad de significación se verifica porque $\mathcal{B}$ obtiene de $\mathcal{A}$ el isomorfismo $H \sim H'$ a través del algoritmo, lo que le permite obtener el isomorfismo que desee $G_i \sim H$ por medio del isomorfismo conocido por él $G_i \sim H', i \in \{0,1\}$. Además, $\mathcal{B}$ no recibe información alguna acerca del otro isomorfismo $G_{\bar{i}} \sim H'$. La propiedad de inconsciencia se satisface puesto que $\mathcal{A}$ no puede averiguar qué grafo G_i, $i \in \{0,1\}$, fue usado por $\mathcal{B}$ para generar H'. □

Para contrastar el funcionamiento de este algoritmo se puede consultar la figura 2.9 incluida a modo de ejemplo.

2.1.6. Algoritmo OT1-2-GP y OT1C-2-GP

En la generalización descrita en la figura 2.10 se muestran una transferencia inconsciente 1 de 2 (entendiendo excluido el comentario entre paréntesis) y una transferencia inconsciente 1 seleccionado de 2 (incluyendo dicho comentario), en las que los dos secretos de $\mathcal{A}$ son sendas soluciones de un problema difícil P en dos grafos diferentes G_0 y G_1. Estos grafos no deben ser isomorfos pero sí deben tener las propiedades susceptibles de ser comprobadas en tiempo polinomial, [Gol97], [GT01], [Ron01], [Gol02a].

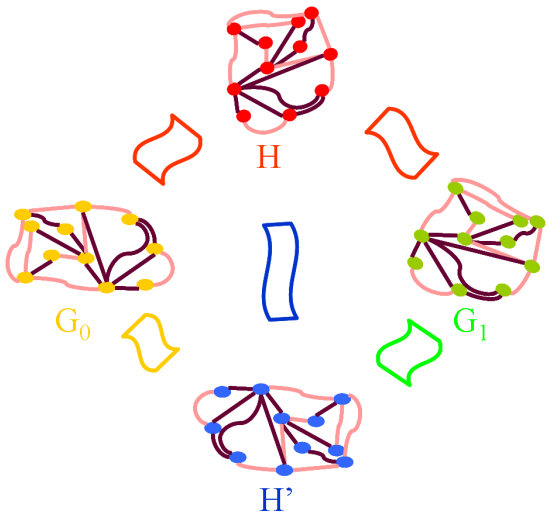

Figura 2.9: Ejemplo de Algoritmo OT1C-2-GI

Ambos participantes deben tener recursos computacionales polinomialmente acotados pero $\mathcal{A}$ debe tener la capacidad de resolver P en toda copia isomorfa de G_0 y G_1, anque no el isomorfismo. La única diferencia entre las dos versiones de transferencia inconsciente reside en la etapa en la que se fija el reto. En el caso de $OT1C-2$, $\mathcal{B}$ debe indicar a $\mathcal{A}$ el grafo en el que desea la solución de P (en la descripción del protocolo aparece destacado entre paréntesis), mientras que en el algoritmo $OT1-2$ $\mathcal{A}$ selecciona el grafo aleatoriamente [HC03a].

La definición formal de ambos algoritmos es como sigue:

- La partición $\{P_0, P_1\}$ viene dada por las dos soluciones $\{Sol_P(G_0), Sol_P(G_1)\}$.
- Las elecciones de $\mathcal{B}$, b_j están definidos por los isomorfismos $H_i \sim G_i$, para $i \in \{0,1\}$, mientras que el reto $h(b_j)$ está representado por $H_i, i \in \{0,1\}$.
- La respuesta de $\mathcal{A}$, $Sol(a_i, h(b_j))$ está formada por la solución $Sol_P(H_j)$ y el isomorfismo $G_j \sim H_j$.

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ el problema P y los grafos G_0 y G_1.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ dos grafos H_0, H_1, isomorfos respectivamente a los grafos originales G_0, G_1, en orden aleatorio (y un bit $j \in \{0, 1\}$).

Respuesta: $\mathcal{A} \longrightarrow \mathcal{B}$ la solución $Sol_P(H_j)$.

Verificación: $\mathcal{B}$ determina la solución $Sol_P(G_j)$ a partir de $Sol_P(H_j)$ y el isomorfismo $G_j \sim H_j$.

Figura 2.10: Algoritmos OT1-2-GP y OT1C-2-GP

$\mathcal{A}$ realmente no sabe cuál de los grafos originales es isomorfo al que $\mathcal{B}$ eligió en la etapa de selección del reto ya que para saberlo tendría que ser capaz de resolver el problema del isomorfismo y se supone que no lo es. Además $\mathcal{B}$ no tiene más que usar el isomorfismo correspondiente para determinar cuáles son los elementos de la solución en uno de los grafos de entrada.

Teorema 2.1.5. *Los algoritmos $OT1-2-GP$ y $OT1C-2-GP$ son respectivamente una Transferencia Inconsciente 1 de 2 y 1 seleccionado de 2.*

Demostración. La significación es verificada en ambos casos ya que $\mathcal{B}$ obtiene de $\mathcal{A}$ la solución $Sol_P(H_j)$ que, combinada con el isomorfismo $G_j \sim H_j$, le permite calcular la solución secreta $Sol_P(G_j)$, donde j es aleatoriamente seleccionado por $\mathcal{A}$ en el caso de la $OT1-2-GP$, y en el caso de $OT1C-2-GP$ es fijado por $\mathcal{B}$. Además, $\mathcal{B}$ no recibe información alguna sobre la otra solución $Sol_P(G_{\bar{j}})$. También la inconsciencia se mantiene puesto que los grafos H_0 and H_1 son enviados en orden aleatorio, de forma que $\mathcal{A}$ es incapaz de averiguar cuál es el isomorfo a G_0 y cuál a G_1 debido a que ambos grafos poseen idénticas invariantes polinomialmente comprobables. □

Los ataques que pueden realizar los participantes, suponiendo el modelo malicioso, durante la ejecución de la propuesta se muestran a continuación.

- Ataque de $\mathcal{A}$: Si $\mathcal{A}$ no sabe resolver el problema P en los grafos isomorfos a G y a H, no podrá enviar una solución válida a $\mathcal{B}$, cuando éste se la requiera, por lo que $\mathcal{B}$ descubrirá el engaño. Otro problema surge si $\mathcal{A}$ sólo sabe resolver el problema en uno de los grafos de partida (supongamos sin pérdida de generalidad que conoce la solución en G). Si se estuviera ante esta situación y $\mathcal{B}$ solicitara que $\mathcal{A}$ resolviera el problema en un grafo isomorfo a G, la estafa no sería detectada. Sin embargo si $\mathcal{B}$ hiciera la otra elección, si lo detectaría. En el caso del algoritmo $OT1-2-GP$ dicha elección es aleatoria. En cualquiera de los casos, si $\mathcal{B}$ elige al azar se concluye que la estafa es detectada con probabilidad 1/2.

- Ataque de $\mathcal{B}$: Si los grafos iniciales G y H son isomorfos y $\mathcal{B}$ conoce el isomorfismo que los relaciona, obtendrá la solución del problema en ambos grafos. Para solventar esta dificultad se recomienda incluir alguna garantía de que los grafos no son isomorfos, pero de forma que ambos grafos tengan las características necesarias para que determinar el isomorfismo no sea fácil (en [For96a] se aconseja el uso de grafos regulares). Otro posible ataque consistiría en construir grafos no isomorfos a los de partida, pero esto sería fácilmente detectado por $\mathcal{A}$ cuando intentara resolver el problema en el grafo elegido por $\mathcal{B}$ en el paso de configuración.

El protocolo propuesto en este apartado se puede generalizar fácilmente para resolver el paradigma de la venta de secretos. Para ello se utilizan l grafos $G_1, G_2, \ldots, G_l$ en lugar de un único G y se construye un grafo isomorfo a cada uno de ellos. Luego se utiliza el esquema $OT-GP$ sobre cada una de las parejas G_i, H_i correspondientes. Así, al final del protocolo $\mathcal{B}$ obtiene la solución al problema planteado en un grafo $G_x, x \in \{1, 2, \ldots, l\}$ sin que $\mathcal{A}$ sepa de cuál de ellos se trata.

2.1.7. Complejidad de los Algoritmos

En el algoritmo $OT - GI$ la generación del grafo G_0 se realiza teniendo en cuenta que se debe garantizar la dificultad del problema del isomorfismo por lo que se generará un grafo regular sobre el que luego se realizará un intercambio entre dos aristas siguiendo las indicaciones que se mencionan en [For96a]. La generación de G_0 se puede desarrollar en orden $O(n^2)$, y a partir de este grafo se genera G_1 usando una permutación, por lo que la construcción de G_1 se desarrolla en orden $O(m)$. De esta forma la generación de la entrada del protocolo lleva un costo total para $\mathcal{A}$ de $O(n^2)$ operaciones elementales.

Entre las tareas que debe realizar $\mathcal{A}$, la determinación del isomorfismo entre H y uno de los grafos G_i es posible gracias a la hipótesis asociada al protocolo.

En cuanto a las acciones a realizar por $\mathcal{B}$, éstas son la construcción del grafo H, (de orden $O(m)$), así como del isomorfismo secreto a partir de la composición del isomorfismo utilizado por él para la generación de H y el facilitado por $\mathcal{A}$ (también de $O(m)$). En conclusión, la complejidad asociada a los procedimientos desarrollados por $\mathcal{B}$ son de orden $O(m)$.

Como medida de la complejidad de las comunicaciones necesarias para el desarrollo correcto del protocolo se calcula el número de bits que es necesarios transferir. En este caso sólo es necesario el intercambio de los dos grafos de entrada, el grafo que constituye el reto de $\mathcal{B}$ y un isomorfismo. Además el isomorfismo puede interpretarse como una permutación sobre el conjunto de vértices, por lo que se puede codificar como un vector de longitud n. Así concretamente, el número de bits transferidos es del orden de $O((m+n)log\ n)$.

Si se realiza una comparación con el protocolo de Rabin se puede observar que las operaciones necesarias en dicho protocolo comienzan con la necesidad de generar números enteros primos de más de cien dígitos, y una vez obtenido su producto debe ser transferido. Además se maneja una potencia de otro número de la misma longitud

que el anteriormente transferido. Por tanto se puede afirmar que la complejidad de las operaciones anteriores es indudablemente superior a las realizadas en las propuestas descritas.

En el algoritmo $OT - GP$ la construcción del conjunto de grafos de entrada a desarrollar por $\mathcal{A}$ es similar a la del algoritmo analizado anteriormente. La única diferencia es que esta vez se deben generar tres grafos isomorfos conteniendo una solución de determinado problema. De esta manera, la complejidad del algoritmo depende implícitamente del problema seleccionado. Suponiendo que el primer grafo se pueda generar en orden $O(n^2)$, la generación de los tres grafos de entrada se podría desarrollar en orden $O(n^2)$. Otra acción asignada a $\mathcal{A}$, cuya complejidad también depende del problema concreto es la construcción de la solución del problema base en el grafo construido por $\mathcal{B}$, que es posible gracias a la hipótesis relacionada con sus capacidades computacionales.

Los procedimientos asociados a $\mathcal{B}$ sólo requieren capacidad de cálculo acotada polinomialmente puesto que:

- la generación del grafo isomorfo H se desarrolla en orden $O(m)$,
- el cálculo de la composición del isomorfismo inverso que usó para la generación de H con el facilitado por $\mathcal{A}$ se realiza en orden $O(n)$,
- la comprobación del isomorfismo usa orden $O(m)$,
- la transformación de la solución depende de la cardinalidad de la misma así como de sus características, siendo su verificación polinomial en cualquier caso puesto que se usan problemas $NP - completos$.

Por tanto, se puede concluir que los procedimientos asociados a $\mathcal{B}$ son de orden $O(m)$.

El número de bits intercambiados entre los participantes también depende de la naturaleza del problema usado como base. Por ejemplo, para el caso en el que cualquier solución está determinada por un subconjunto de vértices, se tiene que la transferencia de los grafos conlleva $8m \cdot \log n$ bits, y la transferencia del isomorfismo y solución implica $2n \cdot \log n$ bits, por lo que la transferencia total de datos es del orden $O((m+n) \cdot \log n)$.

El análisis de la complejidad del protocolo anterior se puede extender para las dos nuevas propuestas $OT-1-2-GI$ y $OT1C-2-GI$. De esta forma, la complejidad de los procedimientos asociados a la usuaria $\mathcal{A}$ en ambos protocolos es del orden $O(n^2)$, suponiendo que sus capacidades computacionales le permiten resolver el problema del isomorfismo para cualquier grafo isomorfo a los de partida generados por ella.

De nuevo sucede que las acciones del usuario $\mathcal{B}$ se desarrollan en tiempo polinomial, y concretamente en orden $O(m)$, siendo lo más costoso la generación de los grafos isomorfos.

Durante el desarrollo del protocolo $OT-1-2-GI$ se hace necesario el envío entre los participantes de $(12m+n) \cdot \log n$ bits, mientras que en el segundo caso se intercambian en total $(8m+n) \cdot \log n$ quedando la complejidad de las comunicaciones acotada por $O((m+n) \cdot \log n)$ en ambos casos.

En el caso de los algoritmos $OT1-2-GP$ y $OT1C-2-GP$, la complejidad de los procedimientos a desarrollar por ambos participantes dependen del problema asociado al protocolo. La fase de generación de los grafos de entrada a desarrollar por la usuaria $\mathcal{A}$ consiste en determinar dos grafos regulares no isomorfos con una solución del problema base insertada en cada uno de ellos. Esta tarea se puede realizar generando los dos grafos regulares con las soluciones insertadas, y luego comprobar que no son isomorfos. Por tanto dicha tarea se puede desarrollar en orden $O(n^2)$. Para obtener la solución en uno de los grafos generados por $\mathcal{B}$, es necesario hacer uso de la hipótesis relacionada sus capacidades computacionales.

Las acciones a desarrollar por $\mathcal{B}$ son la generación de los retos (de orden $O(m)$), y la transformación de la solución (cuya complejidad depende básicamente de las características de la misma). Si por ejemplo la solución consiste en un subconjunto de vértices, dicha transformación necesita tiempo lineal. Resumiendo, el usuario $\mathcal{B}$ sólo requiere capacidad computacional acotada polinomialmente para tomar parte en el presente protocolo.

Con respecto a la complejidad de las comunicaciones, el envío de los grafos de entrada conlleva $4m \cdot \log n$ bits, el reto planteado por $\mathcal{B}$ implica $2m \cdot \log n$ bits, y la solución del problema en el grafo auxiliar consiste en $k \cdot \log n$ bits, en el caso de que sea un conjunto de vértices de cardinalidad k. Para este último cálculo se puede usar una codificación del problema usado como base que utilice enteros como índices. En definitiva, las comunicaciones que requiere el protocolo son del orden $O((m+n)\cdot\log n)$.

2.2. Compromiso de Bits

Los protocolos unilaterales analizados en esta sección configuran la segunda piedra angular del área de los protocolos criptográficos bipartitos.

Definición 2.2.1. Compromiso de Bits (BC, Bit Commitment)
Los algoritmos denominados compromisos de bits tienen por objetivo permitir que una usuaria $\mathcal{A}$ se comprometa frente a otro usuario $\mathcal{B}$ con cierto valor binario, de forma que $\mathcal{A}$ no pueda modificarlo, y además $\mathcal{B}$ no pueda descubrir el valor hasta que $\mathcal{A}$ abra el compromiso. Estas dos propiedades se explicitan a continuación detalladamente.

- *Inalterabilidad*: $\mathcal{A}$ no puede modificar el bit comprometido una vez ha sido enviado el testigo del compromiso a $\mathcal{B}$.

- *Ilegibilidad:* $\mathcal{B}$ no puede obtener ni el valor comprometido ni información alguna sobre el mismo hasta que $\mathcal{A}$ lo abra.

Debido a las propiedades mencionadas en la definición anterior en algunos trabajos [Wig01] se hace referencia a este protocolo determinista con el término de sobre digital, ya que sus características son análogas a las de un sobre común.

La primera condición es equivalente a la propiedad de corrección ya mencionada en la definición de protocolo criptográfico, siendo también denominada por algunos autores como propiedad de *vinculación* de los compromisos de bits. La segunda condición se corresponde con la propiedad de privacidad, y en este ámbito algunas veces se usa el término de propiedad de *ocultación* para referirse a ella.

De acuerdo con la definición original donde el secreto comprometido es un único bit, un esquema de BC puede ser considerado como una correspondencia sobreyectiva definida sobre un dominio extenso (normalmente un conjunto de instancias de un problema base) y cuyo conjunto imagen está formado por el conjunto binario $\{0, 1\}$. Así, se entiende comprometido el bit de salida de la correspondencia mediante un elemento aleatorio del correspondiente conjunto preimagen. Desde este punto de vista estos esquemas pueden ser considerados como un caso especial de función hash dada la necesaria unidireccionalidad y la reducción drástica del tamaño del conjunto de elementos del dominio. Además nótese que la propiedad de vinculación de los esquemas de BC implica que la correspondencia subyacente sea una función. Por otro lado, un BC posee la propiedad de ocultación si las distribuciones del conjunto preimagen asociado al cero y del conjunto preimagen asociado al uno al cero y los asociados al uno son indistinguibles para $\mathcal{B}$.

A continuación se enumeran de forma resumida, las propiedades a verificar para comprobar la tolerancia a fallos cuando un nuevo esquema de compromiso de bits es propuesto.

- Cualquiera de los dos posibles valores binarios puede comprometerse de manera que sean totalmente indistinguibles para el usuario $\mathcal{B}$.

- La manera de abrir el compromiso es totalmente independiente del contenido del mismo, es decir la apertura no debe permitir la modificación del contenido.

- De la apertura de varios compromisos no se deduce información alguna que facilite a $\mathcal{B}$ la apertura de posteriores compromisos.

La mayoría de esquemas propuestos, son probablemente seguros ya que se supone que $\mathcal{B}$ está acotado polinomialmente y que $\mathcal{A}$ conoce una solución secreta del problema difícil que usa para comprometer un bit i.

2.2.1. Estado del Arte

El primer algoritmo de BC fue definido por Blum en 1982 [Blu82a]. Desde entonces son muchos los algoritmos basados en varias herramientas criptográficas típicas como funciones hash, cifrados de clave secreta, generadores seudoaleatorios, logaritmos discretos o restos cuadráticos que han sido propuestos. Los esquemas de compromiso de bits han demostrado ser muy útiles como bloques básicos en el diseño de protocolos criptográficos más complejos. Es por esto por lo que se pueden considerar la segunda primitiva en importancia en el diseño de protocolos. Además también en este caso, como con las transferencias inconscientes, se ha demostrado formalmente [OVY92] que estos esquemas pueden usarse como primitivas para el diseño de cualquier protocolo bipartito, tal y como confirman los ejemplos de casos prácticos que serán descritos en secciones posteriores.

Existen múltiples trabajos que relacionan BC y ZKP (ver capítulo 3). Un ejemplo de esta relación son los trabajos de Damgård [Dam89] y [OOF92]. En el primero de estos artículos se usa la existencia de demostraciones de conocimiento nulo para garantizar la existencia de BC. En [OOF92] se apunta la posibilidad de usar un protocolo de compromiso de BC descrito en [Nao89] (basado en el uso de generadores aleatorios) para evitar la utilización fraudulenta de ejecuciones ZKP consistentes en

que dos usuarios $\mathcal{A}$ y $\mathcal{B}$ conspiren contra un tercer participante $\mathcal{C}$ usando para ello una ejecución previa desarrollada entre ellos. Más recientemente han aparecido trabajos [DN02], [CF01] en los que se estudia la posibilidad de ejecutar concurrentemente algunos de los protocolos de BC existentes con vistas a garantizar la propiedad de conocimiento nulo en las ZKP cuando éstas se usan también concurrentemente.

Brassard y Crépeau presentan en [BC90] un interesante recorrido por los diferentes protocolos de BC que se habían propuesto hasta ese momento basados en mecánica cuántica.

Además de los generadores aleatorios y la computación cuántica también las funciones hash han contribuido a la construcción de protocolos de BC [HM96], generándose con ellas esquemas de fácil implementación.

Incluso en el ámbito de las firmas digitales se ha demostrado la utilidad de los protocolos de BC. Muestra de ello es la construcción de firmas tolerantes a fallos (fail-stop signatures) a partir de BC que se establece en [DPP93]. Este tipo de firmas permiten distinguir el caso de que el sistema haya sido roto debido a la revelación de información por parte del individuo que firma el mensaje, o bien ha sido la participación de un agente externo lo que ha puesto en evidencia el sistema.

En el libro [Dam99] se hace un largo recorrido por diferentes implementaciones de esta primitiva mostrando la relación directa entre estos protocolos y las demostraciones de conocimiento nulo. En [BMSW00] se puede encontrar un análisis de protocolos de BC considerados incondicionalmente seguros en los que participan tres usuarios. Uno de estos participantes juega el papel de TTP encargándose de arbitrar inicializando el protocolo con el envío de cierta información a los otros dos.

2.2.2. Esquema General BC

En la figura 2.11 se propone un esquema general de esta primitiva basado en la técnica de corte-elección. De nuevo el esquema general $BC - GS$ [CHB02b] está dividido en cuatro fases claramente diferenciables. Existe una etapa de configuración en la que se genera una partición (corte) de una instancia de un problema difícil. A continuación se define y transfiere el testigo del compromiso a partir de la salida de una función unidireccional aplicada sobre uno de los elementos que definen la partición previamente mencionada. En la fase de apertura del compromiso se envía a la parte contraria el argumento usado para la definición del mismo. Llegados a este punto, sólo resta que el receptor compruebe que la información recibida en este paso se corresponde con la obtenida en el paso del compromiso. La propiedad de inalte-

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ una partición de un instancia del problema de entrada $\{P_0, P_1\}$.

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ el testigo $h(a_i)$ obtenido a través de una función unidireccional h sobre un elemento aleatorio $a_i \in P_i$, siendo $i \in \{0, 1\}$ el bit comprometido.

Apertura: $\mathcal{A} \rightarrow B$ el elemento a_i.

Verificación: $\mathcal{B}$ comprueba la información recibida descubriendo el bit comprometido i.

Figura 2.11: Esquema BC-GS

rabilidad o vinculación es verificada por el esquema general $BC - GS$ ya que si $\mathcal{A}$ modifica el contenido, el fraude es detectado por $\mathcal{B}$ en la fase de verificación. Por otro lado, la propiedad de ilegibilidad u ocultación se garantiza a través de la transformación unidireccional usada en la fase de compromiso. También se verifican las tres propiedades específicas de BC ya que es posible comprometer cualquiera de los dos valores de i, y la elección de la función unidireccional h garantiza la no modificación

y el no descubrimiento de información sensible por parte de $\mathcal{B}$.

Tal y como se puede observar, en los dos esquemas generales $OT-GS$ y $BC-GS$ hay muchas coincidencias. Sin embargo, en este último el papel de $\mathcal{B}$ es pasivo porque su participación se limita a comprobar la información recibida en el último paso de verificación. Así, el esquema $BC-GS$ puede ser considerado un protocolo no interactivo debido a que todas las comunicaciones son unidireccionales y se producen en la dirección de $\mathcal{A}$ hacia $\mathcal{B}$.

A continuación se describen algunas propuestas nuevas de BC basadas en grafos. En cada una de ellas la fase de configuración requiere de la elección por parte de $\mathcal{A}$ de grafos lo suficientemente grandes como para garantizar la propiedad de ocultación.

2.2.3. Algoritmo BC-GI

La siguiente propuesta de BC se basa de nuevo en el problema del isomorfismo, esta vez junto con su problema complementario, el no isomorfismo de grafos [CHB02b]. En el caso de que el secreto de $\mathcal{A}$ sea un bit i, entonces el algoritmo $BC-GI$ mostrado en la figura 2.12 puede usarse para comprometerse con él usando para ello dos grafos no isomorfos G_0 y G_1, que tienen idénticas invariantes polinomialmente comprobables. La correspondiente formalización del algoritmo $BC-GI$ de acuerdo con el esquema

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ los grafos G_0 y G_1.

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ un grafo H_i, isomorfo a uno de los grafos originales, $G_i, i \in \{0,1\}$.

Apertura: $\mathcal{A} \longrightarrow \mathcal{B}$ el isomorfismo $H_i \sim G_i$.

Verificación: $\mathcal{B}$ comprueba el isomorfismo recibido y descubre el bit i comprometido.

Figura 2.12: Algoritmo BC-GI

$BC-GS$ es como sigue:

- El problema de entrada se divide en $\{P_0, P_1\} = \{G_0, G_1\}$.
- El testigo $h(a_i)$ es una copia isomorfa H_i de uno de los grafos originales $G_i, i \in \{0, 1\}$, correspondiente al bit comprometido y la elección aleatoria de $\mathcal{A}$, a_i es el isomorfismo correspondiente $H_i \sim G_i$.

Para ilustrar el funcionamiento del algoritmo $BC - GI$ puede observarse la figura 2.13.

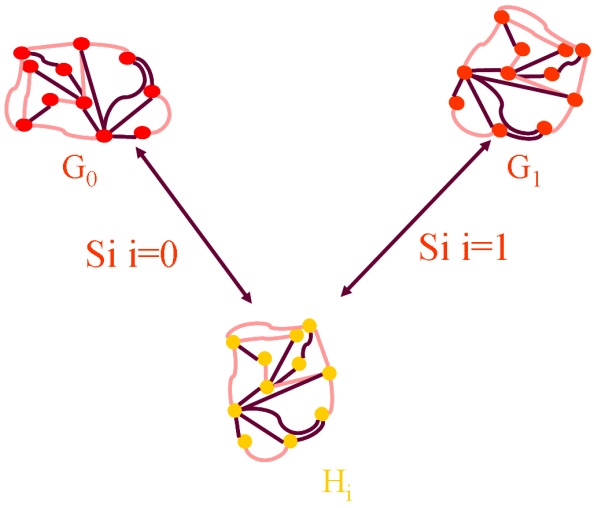

Figura 2.13: Ejemplo de Algoritmo BC-GI

Teorema 2.2.1. *El algoritmo $BC - GI$ es un Compromiso de Bits.*

Demostración. La propiedad de vinculación está garantizada por el no isomorfismo entre G_0 y G_1. Después de que se ha enviado H_i $\mathcal{A}$ no puede modificar el bit comprometido puesto que esto conlleva el definir un isomorfismo entre dos grafos no isomorfos. También el algoritmo verifica la propiedad de ocultación porque se supone que $\mathcal{B}$ es incapaz de resolver el isomorfismo $H_i \sim G_i$ ni el problema del no isomorfismo entre H_i y $G_{\bar{i}}$. □

A continuación se describen los ataques de los que puede ser objeto esta propuesta según el modelo malicioso.

- Ataque cometido por $\mathcal{A}$: Nótese que en el algoritmo anterior los grafos utilizados deben ser no isomorfos, pero a la vez deben tener idénticas propiedades susceptibles de ser comprobadas en tiempo polinomial. El objetivo de la anterior condición no es otro que el que dichos grafos sean indistinguibles para $\mathcal{B}$, ya que esto garantiza el secreto del bit i hasta el momento de la apertura. Por tanto, es de vital importancia que $\mathcal{A}$ y $\mathcal{B}$ acuerden conjuntamente los grafos utilizados en el protocolo, puesto que de no ser así $\mathcal{A}$ podría escoger G_0 y G_1 isomorfos y entonces $\mathcal{B}$ tendría problemas para detectar la estafa. Otra posible solución al problema mencionado, es que $\mathcal{A}$ dé una prueba del no isomorfismo en el paso de apertura, pero en ese caso los grafos no serían reutilizables. La opción menos recomendable debido a las dificultades que conlleva es la de encomendar la elección de dichos grafos a una TTP que arbitra de esa forma el protocolo.

- Ataque cometido por $\mathcal{B}$: La intervención de $\mathcal{B}$ en el protocolo es prácticamente pasiva ya que se limita a comprobar la información recibida, por lo que no cabe la posibilidad de que cometa estafa. Además, determinar cuál es el grafo G_i de partida isomorfo a H_i es impracticable en tiempo polinomial para el usuario $\mathcal{B}$ por lo que éste no tiene información suficiente que pueda utilizar para romper el compromiso. No obstante, es aconsejable que en cada una de las ejecuciones independientes del esquema de compromiso desarrolladas, al menos el grafo H sea renovado para evitar la acumulación de información sensible por parte de $\mathcal{B}$.

2.2.4. Algoritmo BC-GP

Para el caso de un problema P de la Teoría de Grafos que posea una única solución, se puede llevar a cabo una adaptación del esquema $BC-GS$ bastante sencilla. En este caso, para comprometer una solución única, el grafo que determina dicha solución puede ser usado como testigo [CHB02b].

Por otra parte, también es factible el desarrollo de una extensión del algoritmo $BC-GI$ para comprometer el valor de un bit i a través de un problema $NP-completo$. Este nuevo algoritmo mostrado en la figura 2.14 está basado en la utilización de una función hash criptográfica h libre de colisiones, y del problema complementario a P (denotado como $co-P$). Este último problema debe estar clasificado también como un problema intratable al igual que P, aunque en cualquier caso no será $NP-completo$ (a menos que $NP = co-NP$). La correspondencia entre el esquema $BC-GS$ y el algoritmo $BC-GP$ es como sigue:

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ el grafo G y los dos problemas P y $co-P$.

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ el resumen $h(Sol_{problem}(G))$, donde $problem = \begin{cases} \text{P} & \text{si i} = 0 \\ \text{co-P} & \text{si i} = 1 \end{cases}$.

Apertura: $\mathcal{A} \longrightarrow \mathcal{B}$ la solución $Sol_{problem}(G)$.

Verificación: $\mathcal{B}$ comprueba la validez de la solución recibida y descubre el valor del bit secreto de $\mathcal{A}$, i.

Figura 2.14: Algoritmo BC-GP

- La partición $\{P_0, P_1\}$ esta definida por $\{P, co-P\}$.
- El testigo $h(a_i)$ es obtenido mediante el resumen de una solución de P ó de $co-P$ en G, $h(Sol_{problem}(G))$, donde $problem = P$ o $co-P$ dependiendo del bit comprometido, y la elección aleatoria de $\mathcal{A}$, a_i es $Sol_{problem}(G)$.

También en este caso se ilustra gráficamente el funcionamiento del algoritmo mediante la figura 2.15.

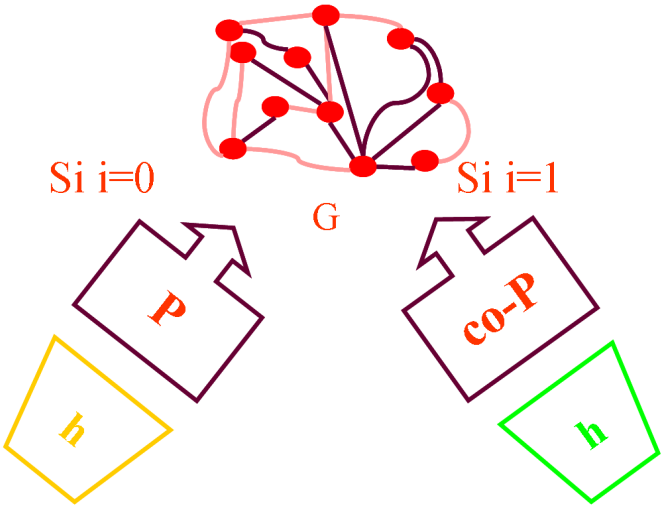

Figura 2.15: Ejemplo de Algoritmo BC-GP

Teorema 2.2.2. *El algoritmo $BC-GP$ es un Compromiso de Bits.*

Demostración. El esquema es vinculante debido a que la función hash usada h es una función libre de colisiones, y a la imposibilidad de que una solución de un problema sea a su vez solución del problema complementario. También el esquema estudiado posee la propiedad de ocultación ya que h es unidireccional y además $\mathcal{B}$ no posee las capacidad computacional suficiente para resolver P ni $co-P$. □

2.2.5. Complejidad de los Algoritmos

La construcción de la entrada en el primer protocolo de compromiso de bits propuesto es equivalente a la realizada por $\mathcal{A}$ en el protocolo $OT1-2-GP$, necesitando $O(n^2)$ operaciones elementales. La generación del compromiso en este caso consiste en la construcción de un grafo isomorfo por lo que su complejidad es de orden $O(m)$. Por

tanto, $\mathcal{A}$ desarrolla su participación en el protocolo con una complejidad del orden $O(n^2)$.

Por otra parte, puesto que el papel de $\mathcal{B}$ se limita a comprobar el isomorfismo recibido, las capacidades computacionales que necesita para su intervención en el protocolo son de orden $O(m)$.

Toda la transferencia de información en los protocolos de compromiso de bits es unidireccional ya que tiene lugar desde $\mathcal{A}$ hacia $\mathcal{B}$, siendo el número de bits transferidos $(6m + n) \cdot \log n$, por lo que el orden de la complejidad de las comunicaciones es $O((m + n) \cdot \log n)$.

En el algoritmo $BC - GP$ la complejidad de los cálculos a desarrollar por ambos participantes necesita la especificación de la función hash que se ha decidido utilizar, siendo la generación del grafo de partida (de orden $O(n^2)$) lo más costoso para la usuaria $\mathcal{A}$.

También para acotar la capacidad de cálculo de $\mathcal{B}$ se necesita la especificación del problema usado como base, puesto que debe verificar la correspondencia y validez de la solución entregada por $\mathcal{A}$. En todo caso, debido al uso de problemas $NP-completos$, dicha verificación se puede desarrollar en tiempo polinomial.

2.3. Firma de Contratos

El diseño de protocolos de firma de contratos resulta de una clara y directa aplicación tanto de OT como de BC por lo que en esta sección se presta especial atención a este interesante tipo de protocolos bilaterales.

Definición 2.3.1. Firma de Contratos (CS, Contract Signing)
Los protocolos bipartitos denominados de Firma de Contratos se caracterizan por estar diseñados para que dos usuarios, $\mathcal{A}$ y $\mathcal{B}$ firmen simultáneamente un contrato a través de una red de comunicaciones de forma que ninguno pueda obtener la firma del

otro sin haber firmado el contrato, garantizando además que ninguno pueda repudiar su propia firma. Para ello se deben verificar las propiedades siguientes:

- *Vinculación*: Durante la ejecución del protocolo, cada participante puede alcanzar un punto partir del cual ambos firmantes quedan comprometidos a finalizar el protocolo.

- *Imparcialidad*: Ambas firmas son infalsificables, pudiéndose comprobar este hecho por cada uno de los participantes.

Ambas propiedades de vinculación e imparcialidad garantizan la tolerancia a fallos del protocolo en el modelo semi-honesto.

En general los protocolo de CS usan esquemas de BC y se basan en OT, y ambas firmas son fraccionadas enviándose de forma alternativa.. Concretamente cualquier protocolo de OT puede ser adaptado a un esquema de CS sin más que aplicar el algoritmo de OT sucesivamente, y considerar el contrato firmado al final de la ejecución si ambos usuarios logran conocer la información secreta del otro.

Se ha demostrado formalmente que es imposible diseñar un protocolo determinista que no requiera el arbitraje de una TTP. Por tanto, el diseño de un protocolo de CS independiente de TTP ha de ser probabilista y basarse en un proceso de aleatorización. Los protocolos de CS aleatorizados más conocidos se basan en herramientas criptográficas tales como cifrados de Clave Secreta o de Clave Pública.

Por otra parte, existe una relación directa entre los protocolos de CS y los conocidos como Intercambio de Secretos y Correo Certificado ya que los tres protocolos son reducibles unos a otros.

Definición 2.3.2. Intercambio de Secretos (SE, Secret Exchange)
Un protocolo de Intercambio de Secretos permite a dos usuarios $\mathcal{A}$ y $\mathcal{B}$ que disponen cada uno de un secreto, el intercambiarlos de manera que posteriormente pueden verificar que obtuvieron el secreto correcto.

El caso de el correo certificado presentado a continuación pertenece al conjunto de protocolos unilaterales.

Definición 2.3.3. Correo Certificado (CM, Certified Mail)
Un protocolo de Correo Certificado garantiza que una usuaria $\mathcal{A}$ puede enviar a $\mathcal{B}$ un mensaje de forma que éste último no tenga acceso al contenido a menos que antes envíe un acuse de recibo a $\mathcal{A}$.

Los tres casos de CS, SE y CM pueden resolverse fácilmente usando OT de forma iterativa. Así, concretamente esta misma idea puede aplicarse fácilmente con las propuestas de OT realizadas en este trabajo basada en grafos.

2.3.1. Estado del Arte

Probablemente el algoritmo propuesto por Rabin [Rab81] para resolver el problema de CS sea el más conocido. Está basado en el uso iterativo del protocolo de Rabin para transferencia inconsciente, de forma que el contrato se considera firmado si al final ambos conocen la factorización secreta del otro. También se han propuestos protocolos de CS deterministas basados en Clave Pública con la intervención de una TTP [Eve82]. Posteriormente se demostró la imposibilidad de definir CS deterministas sin la presencia de TTP [EY80a], con lo que comenzaron a surgir propuestas probabilistas, como la incluida en [EGL82]. El principal inconveniente de las propuestas probabilistas es la necesidad de incrementar el número de mensajes intercambiados para conseguir un nivel de confianza suficiente, con lo que la complejidad de las comunicaciones se ve incrementada. Una solución a este problema, que aparece en [ASW97] considera el uso de TTP solamente en caso de haberse detectado alguna dificultad.

Las relaciones existentes entre los protocolos de CS, CM y SE se establecieron en [BVV84]. En este trabajo se especifican la reducciones a aplicar para transformar uno de estos protocolos en cualquiera de los otros dos, indicando también cuál es el nivel de confianza que adquieren los participantes después de intervenir en el protocolo.

En [Ted85] se estudian y proponen maneras concretas de garantizar la propiedad de vinculación en estos protocolos, haciendo que ambos participantes acuerden previamente el orden y tamaño de las fracciones de información intercambiadas. Para un análisis detallado y formal de los protocolos de CM se recomienda consultar [PSW00].

Una nueva visión de los protocolos de CS planteada desde el punto de vista de la Teoría de Juegos se puede encontrar en [KR02].

2.3.2. Algoritmo CS-GI

El protocolo aquí formulado [CHB03a] hace uso de la transferencia inconsciente como primitiva. El problema utilizado como base es nuevamente el problema del isomorfismo. Se supone que $\mathcal{A}$ sabe resolver el problema del isomorfismo para todos los grafos isomorfos a los grafos de partida por ella seleccionados y la misma hipótesis se necesita para $\mathcal{B}$ referida a dos grafos seleccionados por este último. Los grafos fijados tanto por $\mathcal{A}$ como por $\mathcal{B}$ son parejas de grafos isomorfos entre si, ($G_0^{\mathcal{A}}$ y $G_1^{\mathcal{A}}$) y ($G_0^{\mathcal{B}}$ y $G_1^{\mathcal{B}}$), y los secretos que están en juego son los isomorfismos. Todas las parejas de grafos, una vez generadas, se hacen públicas. Ambos firmantes $\mathcal{A}$ y $\mathcal{B}$ deben ejecutar el protocolo varias veces e independientemente sobre el mismo número de parejas de grafos. En la figura 2.16 se encuentra una descripción más detallada de la propuesta de firma de contratos basada en el isomorfismo de grafos. Se considera que el contrato está firmado cuando ambos logran recibir todos o bien un porcentaje predeterminado de los isomorfismos secretos del otro. Por tanto, tras el intercambio bilateral de isomorfismos, aquellos que han sido recibidos del otro participante configuran una firma válida del contrato asociada a dicho participante. El proceso descrito debe repetirse un número suficiente de veces sobre cada par de grafos de manera que al terminar, la probabilidad de que los isomorfismos secretos no hayan sido mutuamente intercambiados sea realmente muy pequeña, reduciéndose de esta manera la probabilidad de que uno de los participantes estafe a su contrario. En tal caso, la incertidumbre

Configuración:

- $\mathcal{A} \longrightarrow \mathcal{B}$ dos grafos G_0^A y G_1^A isomorfos.
- $\mathcal{B} \longrightarrow \mathcal{A}$ dos grafos G_0^B y G_1^B isomorfos.

Compromiso:

- $\mathcal{B} \longrightarrow \mathcal{A}$ el grafo $H_i^A, i \in \{0,1\}$ isomorfo a uno de los grafos originales, G_i^A.
- $\mathcal{A} \longrightarrow \mathcal{B}$ el grafo $H_j^B, j \in \{0,1\}$ isomorfo a uno de los grafos originales, G_j^B.

Apertura:

- $\mathcal{A} \longrightarrow \mathcal{B}$ el isomorfismo entre H_i^A y $G_{i'}^A, i, i' \in \{0,1\}$.
- $\mathcal{B} \longrightarrow \mathcal{A}$ el isomorfismo entre H_j^B y $G_{j'}^B, j, j' \in \{0,1\}$.

Verificación: $\mathcal{A}$ y $\mathcal{B}$ comprueban la validez del isomorfismo recibido por cada uno de ellos.

Figura 2.16: Algoritmo CS-GI

respecto al estado del otro participante garantiza la propiedad de vinculación, y con ella la corrección del protocolo.

En cuanto a la tolerancia a fallos en el modelo malicioso, el ataque más básico que ambos pueden cometer es generar grafos no isomorfos a los de partida, pero esto será detectado inevitablemente por su contrario.

En este protocolo ambos usuarios deben realizar los mismos procedimientos, puesto que es totalmente simétrico, y además ambos hacen uso de la misma hipótesis. Concretamente en cada iteración, ambos realizan la generación de la correspondiente pareja de grafos (de orden $O(n^2)$), generan un grafo isomorfo a uno de los grafos asociados a su contrario (de orden $O(m)$), y verifican el isomorfismo construido (de orden $O(m)$). Por tanto, si el número de iteraciones se denota por l, los procedimientos de ambos usuarios son de orden $O(l \cdot n^2)$.

Es preceptivo que los grafos usados por $\mathcal{A}$ y $\mathcal{B}$ no sean isomorfos puesto que en caso contrario las hipótesis permitirían que cualquiera de los participantes pueda obtener el isomorfismo de su contrincante sin necesidad de terminar el protocolo. Para solventar este problema se puede simplemente usar grafos con distinto número de vértices y aristas para cada participante. En este caso y denotando por m_0, n_0 y m_1, n_1 los parámetros asociados a los grafos de cada uno, el número de bits intercambiados es $(6m_i + n_i) \cdot \log n_i, i = 0, 1$.

La imparcialidad viene garantizada por la dificultad del problema del isomorfismo.

En cuanto a la privacidad del protocolo, dado que el algoritmo $CS-GI$ sólo sirve para intercambiar isomorfismos (secretos a priori) no existe posibilidad de descubrimiento de información sensible.

2.4. Lanzamiento de Monedas

El objetivo de esta primitiva bilateral, que fue una de las primeras en aparecer en la bibliografía sobre protocolos, es generar un bit aleatorio en común entre ambos usuarios $\mathcal{A}$ y $\mathcal{B}$, o lo que es equivalente, descubrir cuál de los dos participantes gana una apuesta sobre un valor binario generado al azar. A continuación se incluye una definición más detallada.

Definición 2.4.1. Lanzamiento de Monedas (CF, Coin Flipping)
Un protocolo de Lanzamiento de Monedas debe permitir la generación compartida y aleatoria de un bit entre dos participantes $\mathcal{A}$ y $\mathcal{B}$ garantizando a ambos la imposibilidad de que la moneda sea lanzada después de conocer el valor seleccionado por el contrario, y de conocer el valor del lanzamiento antes de realizar la propia elección.

Se puede diseñar un lanzamiento de monedas no arbitrado a partir de cualquier OT de manera que el usuario $\mathcal{B}$ gana si logra recibir el secreto transferido, y en caso

contrario pierde. Por tanto el protocolo propuesto a continuación coincide prácticamente con el primero planteado en el apartado referente a la transferencia inconsciente y sólo hay que especificar cuándo se considera que el resultado es favorable a cada oponente. º

También es posible definir un algoritmo de CF basándose en un esquema cualquiera de BC. En este caso $\mathcal{A}$ y $\mathcal{B}$ escogen al azar sendos bits aleatorios secretos a y b, se comprometen con ellos intercambiándose los testigos de dichos compromisos mediante BC. Tras las fases de apertura y verificación de los compromisos, ambos participantes toman como resultado del lanzamiento $a+b$. Por otra parte, el algoritmo que se describe a continuación [CH02b] resulta de la aplicación bilateral del esquema $BC - GI$.

En la versión de CF como apuesta, previo al comienzo del protocolo ambos usuarios habrán decidido qué bit de salida es el ganador. Si ambos oponentes actúan de manera honesta, es decir el protocolo se desarrolla según el modelo semi-honesto, los resultados de $a + b$, obtenidos por ambos coinciden.

Las aplicaciones principales de este protocolo se encuentran en generación de claves compartidas, intercambio de secretos y póquer mental.

2.4.1. Estado del Arte

La primera propuesta para este protocolo la aporta Blum [Blu82b], usando como base el problema de los restos cuadráticos. También desde el campo de la computación cuántica surge una alternativa al protocolo de Blum [BC90]. Micali y Rabin proponen en [MR90] un protocolo de CF que no hace uso de hipótesis referentes a la existencia de funciones unidireccionales y tal que la probabilidad de error asociada al mismo nula. En el reciente trabajo de Lindell [Lin01] se demuestra la importancia de este protocolo en el campo del diseño de protocolos generales. Los resultados allí obtenidos garantizan la computación segura de cualquier función bipartita usando para ello un

protocolo de CF paralelo.

2.4.2. Algoritmo CF-GI

En primer lugar tal y como ya se indicó, fácilmente es posible aplicar el algoritmo $OT - GI$ entendiendo que $\mathcal{B}$ gana el lanzamiento si logra recibir el isomorfismo secreto de $\mathcal{A}$ entre G_0 y G_1 y en cambio, $\mathcal{A}$ es el vencedor si dicho isomorfismo no es transferido.

En el algoritmo $CF - GI$ la propiedad de corrección queda garantizada dada la no existencia de isomorfismos entre los cuatro grafos de partida. Por otra parte se entiende que los grafos no son reutilizables por lo que la privacidad también se mantiene.

En cuanto a la tolerancia a fallos, las estafas de las que puede ser objeto el esquema propuesto coinciden con las mencionadas en el apartado 2.2.3 sobre el algoritmo $BC - GI$.

La generación en común de los cuatro grafos no isomorfos regulares conlleva $O(n^2)$ para cada uno de los grafos.

Además, al igual que sucedía con el protocolo de firma de contratos, también en este caso se aprecia una total simetría en las acciones a desarrollar por cada participante, siendo dichas acciones de orden $O(m)$.

En lo referente a la complejidad de las comunicaciones ambos participantes envían un grafo isomorfo a uno de los originales y el isomorfismo usado para construirlo, siendo por tanto el número de bits transferidos $2(2m + n) \cdot \log n$.

En todos los protocolos pertenecientes al capítulo 2 las acciones a realizar por el usuario $\mathcal{B}$ pueden acometerse con capacidad de cálculo polinomial. Otra característica común en estos protocolos es la coincidencia en el valor de la complejidad de las comunicaciones, siendo en todos los casos de orden $O((m + n) \cdot \log n)$.

Configuración: $\mathcal{A}$ y $\mathcal{B}$ seleccionan conjuntamente cuatro grafos no isomorfos (G_1, G_2, G_3 y G_4)

1. Compromiso:

- $\mathcal{A} \longrightarrow \mathcal{B}$: un grafo H_1 isomorfo a G_i, siendo
  - $i = 1$, si $a = 0$, o bien
  - $i = 2$, si $a = 1$.
- $\mathcal{B} \longrightarrow \mathcal{A}$ un grafo H_2 isomorfo a G_j, siendo
  - $j = 3$, si $b = 0$, o bien
  - $j = 4$, si $b = 1$.

2. Apertura:

- $\mathcal{A} \longrightarrow \mathcal{B}$ el isomorfismo entre H_1 y el correspondiente G_i.
- $\mathcal{B} \longrightarrow \mathcal{A}$ el isomorfismo entre H_2 y el correspondiente G_j.

3. Verificación:

- $\mathcal{B}$ comprueba el isomorfismo y obtiene a.
- $\mathcal{A}$ comprueba el isomorfismo y obtiene b.

4. Ambos calculan $a + b$

Figura 2.17: Algoritmo CF-GI

Capítulo 3

Demostraciones de Conocimiento

En este capítulo se estudia en profundidad una serie de protocolos bipartitos entre los que destacamos los Sistemas Interactivos de Demostración y las Demostraciones de Conocimiento Nulo. Entre estas últimas se hace un mayor hincapié en las Demostraciones de Conocimiento Nulo de Conocimiento debido a la aplicación que tienen en diferentes campos de la criptografía como la demostración de que un participante actúa correctamente en un determinado protocolo y el diseño de esquemas de identificación y control de accesos, principalmente. Esta clase de demostraciones conlleva la utilización de números aleatorios como retos y el uso de esquemas de compromisos de bits.

Las definiciones básicas imprescindibles en las descripciones de los protocolos se aportan en la primera sección. La segunda sección se destina al estado del arte de las demostraciones de conocimiento nulo y sus variantes, haciendo un recorrido por aquellos trabajos más relevantes.

Para la formalización de los protocolos de conocimiento nulo se utiliza el paradigma de la simulación, concepto que permite el estudio de la seguridad de estos protocolos. Es por esto que dicho paradigma, tratado en la sección 3.3, está irremediablemente unido a los protocolos estudiados en el presente capítulo.

Dos propiedades que han sido extensamente estudiadas particularmente en estos

algoritmos son la interacción entre los usuarios y la posibilidad de ejecutarlos en paralelo sin que su seguridad se vea afectada. Ambos temas se tratan en la sección 3.4.

En el resto del capítulo se describen y analizan las diferentes propuestas, la mayoría de las cuales pertenecen a la categoría de las demostraciones de conocimiento, utilizando en todos los casos problemas de la Teoría de Grafos.

También se proponen dos esquemas de identificación cuya principal diferencia consiste en que uno de ellos es determinista y utiliza contraseñas de un único uso, mientras que el segundo es un protocolo probabilista que usa las demostraciones de conocimiento descritas en el resto del capítulo.

Además se aporta un esquema general válido para todas aquellas demostraciones de conocimiento nulo que usan las técnicas de corte y elección y reto-respuesta.

3.1. Definiciones

Los protocolos tratados en este capítulo surgen a partir de la noción de Sistemas de Demostración Interactivos y sus variantes, por lo que se hace necesario incluir en esta sección la descripción y estudio de dichos conceptos.

Goldwasser, Micali y Rackoff introdujeron el concepto de Sistema de Demostración Interactivo como un protocolo interactivo con varias iteraciones. El marco general en el que se define este protocolo unilateral está determinado por la presencia de dos participantes, una Probadora $\mathcal{A}$, y un Verificador $\mathcal{B}$, cuyo objetivo consiste en que $\mathcal{A}$ convenza a $\mathcal{B}$ de la validez de una afirmación sin que exista estrategia alguna que le permita convencer a $\mathcal{B}$ de la validez de una afirmación falsa. Además se supone que existe una entrada común a ambos y una fuente generadora de bits seudoaleatoria privada para cada uno. El desarrollo de dichas demostraciones se lleva a cabo a través de un intercambio de mensajes, siendo el verificador el encargado de aceptar o rechazar la validez de la información cuestionada.

Por tanto, un sistema de demostración no es más que una formalización del proceso de hacer pública una demostración y establecer su validez. La idea que subyace en dichos esquemas consiste en conseguir procedimientos de verificación eficientes en un entorno distribuido y aleatorio, [Gol02b].

Para que estos sistemas puedan considerarse una extensión válida de la idea de demostración hay dos propiedades que deben verificarse con determinada probabilidad, siendo esta probabilidad un parámetro intrínseco del sistema.

- *Completitud*: $\mathcal{B}$ acepta como válidas todas aquellas afirmaciones que son realmente correctas con una probabilidad superior a una determinada constante.
- *Solidez*: $\mathcal{B}$ está protegido ante probadores deshonestos que pretenden convencerle de la veracidad de sentencias falsas, puesto que queda garantizado que casi nunca $\mathcal{B}$ aceptará afirmaciones no válidas (es decir, este suceso ocurre con una probabilidad ínfima).

En los modelos formales se representa a ambos participantes mediante máquinas de Turing que interaccionan compartiendo determinadas cintas, tal y como se establece a continuación.

Definición 3.1.1. Máquina de Turing Interactiva (ITM, Interactive Turing Machine)
Una Máquina de Turing Interactiva es una máquina de Turing determinista que posee las siguientes cintas: una de entrada privada (sólo de lectura), una de salida privada (sólo de escritura), una aleatoria (que contiene una sucesión infinita de bits aleatorios), una de entrada pública (sólo de lectura)y una de salida pública (sólo de escritura), de forma que éstas dos últimas se utilizan para las comunicaciones con otra máquina de las mismas características.

Definición 3.1.2. Par de Máquinas de Turing Interactivas

Se dice que dos máquinas de Turing interactivas forman un par $[\mathcal{A}, \mathcal{B}]$ cuando comparten la cinta de entrada y la cinta de escritura públicas y las cintas de salida, de forma que la cinta de entrada pública de $\mathcal{A}$ es la cinta de salida pública de $\mathcal{B}$ y viceversa.

La definición de sistema de demostración interactivo se plantea para resolver una situación que consiste en demostrar que determinada cadena x pertenece a un determinado lenguaje binario L, por lo que también se le denomina Demostración de Pertenencia a un Lenguaje, y en tal situación se dice que el par de máquinas $[\mathcal{A}, \mathcal{B}]$ reconocen el lenguaje L.

Definición 3.1.3. Sistema de Demostración Interactivo (IPS, Interactive Proof System)
Dado un lenguaje binario $L \subset \{0,1\}^*$, se define un *Sistema de Demostración Interactivo para* L como un par de máquinas de Turing interactivas $[\mathcal{A}, \mathcal{B}]$ tales que $\mathcal{A}$ tiene capacidad computacional no acotada y $\mathcal{B}$ la tiene limitada polinomialmente, y ambas cumplen las propiedades siguientes:

- *Completitud*: Para toda cadena perteneciente al lenguaje L, dicha cadena es aceptada por $\mathcal{B}$ con probabilidad superior a $1-\epsilon$, es decir, $\forall x \in L,\ Prob([\mathcal{A}, \mathcal{B}]$ acepte $x) \geq 1-\epsilon$.

- *Solidez*: Para toda cadena no perteneciente al lenguaje L y cualquier estrategia de una probadora deshonesta $\mathcal{A}^*$, $\mathcal{B}$ acepta la pertenencia de x al lenguaje con probabilidad a lo sumo ϵ. Usando una descripción formal, la propiedad de solidez se verifica si $\forall \mathcal{A}^* \wedge \forall x \notin L,\ Prob([\mathcal{A}^*, \mathcal{B}]\ acepte\ x) \leq \epsilon$.

Obsérvese que en la definición de IPS que utilizan algunos autores, como por ejemplo Goldreich [Gol02b] se exige que la propiedad de completitud se cumpla siempre para una probabilidad 1, pasando a ser así una propiedad determinista. En este trabajo, sin embargo, se usa la definición probabilista anterior menos restrictiva.

Coloquialmente hablando se puede afirmar que la funcionalidad de los IPS se centra en el papel del verificador, ya que la probabilidad de error asociada a la propiedad de solidez es susceptible de ser interpretada como el nivel de consentimiento que el verificador está dispuesto a mantener frente a cadenas no pertenecientes al lenguaje en cuestión, mientras que la probabilidad asociada a la propiedad de completitud refleja la predisposición del verificador a ser convencido de la veracidad de la pertenencia de la cadena al lenguaje.

De esta manera, en ambos casos se obtienen probabilidades de error similares a las correspondientes a los errores tipo I y tipo II de los contrastes de hipótesis. La probabilidad asociada a la solidez es análoga a la probabilidad de error de tipo II, mientras que la probabilidad del suceso ocmplementario al descrito en la propiedad de completitud se corresponde con la probabilidad de error de tipo I ya que implicaría rechazar una hipótesis verdadera.

Tal como se observará en apartados posteriores, la probabilidad de error que aparece en la definición de la solidez puede disminuirse sin más que iterar la interacción entre las máquinas. Además, la elección del parámetro ϵ es libre para los usuarios, aunque habitualmente se tome menor que 1/3. Dado que lo único que es necesario verificar es que dicha probabilidad sea menor que 1/2, una propuesta general válida es $(1/2)^{|x|}$.

Existen variantes de los IPS en las que también se acotan las capacidades computacionales de la probadora apareciendo la propiedad denominada *completitud computacional*. En la definición de esta propiedad, se entiende que $\mathcal{A}$ está acotado polinomialmente. Análogamente la propiedad de solidez puede modificarse garantizando su verificación en el caso de probadores deshonestos polinomialmente acotados, en cuyo caso se habla de *solidez computacional*.

Definición 3.1.4. Argumento

Un sistema de demostración interactivo que satisface la propiedad de solidez computacional se denomina sistema de demostración computacionalmente sólido o *argumento*.

La siguiente variante relacionada con los IPS son los denominados *juegos de Arturo-Merlín* definidos por Babai en [Bab85]. Simplemente constituyen sistemas de demostraciones interactivos con algunas restricciones. En dicha variante el papel de la probadora $\mathcal{A}$ lo realiza Merlín y el del verificador $\mathcal{B}$ Arturo, quienes poseen los mismos recursos computacionales que en el modelo propuesto inicialmente.

Definición 3.1.5. Juegos de Arturo-Merlín, (AMG, Arthur-Merlin Games)
Un Juego de Arturo-Merlín es un IPS en el que el verificador Arturo tiene su capacidad de comunicación rigurosamente limitada, los únicos mensajes que puede producir son secuencias de bits aleatorios, y sólo interviene en el protocolo para tomar una decisión que depende del contenido de la cinta de entrada y de la transcripción de la ejecución. Además, en este caso, el probador Merlín debe conocer todos los bits aleatorios generados por Arturo.

Uno de los protocolos más significativos en Criptografía no sólo por lo interesante de su definición sino también por sus diferentes aplicaciones (esquemas de identificación [FFS88], demostración de la corrección del comportamiento de un usuario en determinado protocolo [GMW86], resultados relacionados con la jerarquía establecida por la complejidad computacional [Sha92] e incluso en el campo de la aproximación de problemas de optimización combinatorios [FK98], etc.) que además ilustra el uso de BC en protocolos de propósito particular es el protocolo bipartito denominado Demostración de Conocimiento Nulo. El resto del presente capítulo está dedicado al análisis de esta clase de protocolos y a la presentación de nuevas propuestas.

Definición 3.1.6. Demostración de Conocimiento Nulo (ZKP, Zero Knowledge Proof)
Una Demostración de Conocimiento Nulo es un IPS que permite a la usuaria $\mathcal{A}$

convencer de manera probabilista a $\mathcal{B}$ de la validez de cierta información sin revelar nada más sobre la misma.

También se puede definir una variante que se ha dado en llamar Demostración de Conocimiento Nulo de Conocimiento

Definición 3.1.7. Demostración de Conocimiento Nulo de Conocimiento ($ZKPK$, Zero Knowledge Proof of Knowledge)
En en el caso de una Demostración de Conocimiento Nulo de Conocimiento $\mathcal{A}$ trata de convencer a $\mathcal{B}$ sobre su posesión de cierta información secreta sin revelar nada sobre la misma, estando las capacidades computacionales de ambos usuarios acotados polinomialmente.

En el entorno de los protocolos es ésta la versión más utilizada debido sobre todo a su gran utilidad en esquemas de identificación. Una característica que los hace especialmente indicados en este caso es la capacidad que tienen para garantizar que la interacción entre ambos participantes no facilita la suplantación de $\mathcal{A}$ por parte de $\mathcal{B}$ ante terceras partes.

Tal y como se comenta en la siguiente sección, la noción de ZKP fue introducida por Goldwasser, Micali y Rackoff en 1989 [GMR89], y desde entonces varios algoritmos basados en grafos (particularmente en problemas tales como los de circuito Hamiltoniano, 3-coloración, isomorfismo y no isomorfismo) así como en otras herramientas de Teoría de Números han sido propuestos. En la primera definición que aparece en el trabajo previamente citado se define la propiedad de conocimiento nulo para cualquier protocolo bipartito, independientemente de los IPS, restringiendo más tarde la definición para protocolos considerados sistemas de demostración interactivos.

Definición 3.1.8. Conocimiento Nulo (ZK, Zero Knowledge)
La propiedad de conocimiento nulo garantiza el hecho de que todo lo que puede ser computado por un adversario cualquiera a partir de la interacción con una probadora

previamente fijada, sobre una entrada común a ambos, también fija, es equivalente a lo que puede computar un algoritmo polinomial al que sólo se le proporciona la entrada común, conocido como *simulador*.

Una versión más restrictiva de conocimiento nulo definida a continuación son las demostraciones de conocimiento nulo reiniciables.

Definición 3.1.9. Demostración de Conocimiento Nulo Reiniciable ($RZKP$, Resettable Zero Knowledge Proof)
Las demostraciones de conocimiento nulo reiniciables son aquellas ZKP resistentes a ataques en los que la capacidad de la probadora para generar nuevas secuencias aleatorias está limitada.

Para describir la definición formal de dicho protocolo propuesta en [GMR89] es necesario realizar algunas definiciones previas que a continuación se detallan y que están relacionadas con variables aleatorias y su simulación.

Definición 3.1.10. Variables Aleatorias Estadísticamente Indistinguibles
Sea $L \subset \{0,1\}^*$ un lenguaje binario. Se dice que dos familias de variables aleatorias $U(x)$ y $W(x)$ son estadísticamente indistinguibles en L si para toda constante $c > 0$ y toda cadena $x \in L$ lo suficientemente larga, se verifica:

$$\sum_{\alpha \in \{0,1\}^*} |prob(U(x) = \alpha) - prob(W(x) = \alpha)| < |x|^{-c}.$$

Si $U = \{U(x)\}$ es una familia de variables aleatorias polinomialmente acotada, es decir para todas las v.a. $U(x) \in U$ existe una constante $d > 0$ tal que las únicas cadenas a las que les asigna probabilidad positiva son aquellas cuya longitud es exactamente $|x|^d$, y $C = \{C_x\}$ es una familia de circuitos booleanos polinomialmente acotada (es decir para todo circuito $C \in C_x$, existe una constante positiva e tal que

el número de puertas de C es como máximo $|x|^e$), entonces $P(U, C, x)$ denota la probabilidad de que la salida de C_x sea 1 para una entrada seleccionada aleatoriamente según $U(x)$.

Definición 3.1.11. Variables Aleatorias Computacionalmente Indistinguibles
Se dice que dos familias de v.a. polinomialmente acotadas, U y W, son computacionalmente indistinguibles en L si para toda familia de circuitos polinomialmente acotada, para toda constante $c > 0$ y toda cadena $x \in L$ lo suficientemente larga,

$$P(U, C, X) - P(W, C, x) < |x|^{-c}.$$

Definición 3.1.12. Variables Aleatorias Aproximables
Sea $L \subset \{0,1\}^*$ un lenguaje binario y $U = \{U(x)\}$ una familia de variables aleatorias. Se dice que U es *perfectamente aproximable* en L si existe una máquina de Turing probabilista polinomial M, tal que para toda cadena $x \in L$, $M(x)$, que denota a la variable aleatoria que simula la salida de la máquina de Turing M sobre la entrada x, es igual a $U(x)$. Análogamente, se dice que U es *estadísticamente (computacionalmente) aproximable* en L si existe una máquina de Turing probabilista, con tiempo esperado de ejecución polinomial, tal que las familias de v.a. $M(x)$ y $U(x)$ son estadísticamente (computacionalmente) indistinguibles en L.

Extendiendo la notación utilizada en [GMR89], el par $[\mathcal{A}, \mathcal{B}]$ denota un protocolo y $\mathcal{B}'$ a un verificador deshonesto. Se supone que dicho verificador está provisto de una entrada auxiliar, y y que está acotado polinomialmente. Además la v.a. denominada denominada *perspectiva* o *transcripción* de la interacción $View_{\mathcal{A},\mathcal{B}'}(x, y)$ representa la v.a. que modela la interacción y el intercambio de mensajes entre $\mathcal{A}$ y $\mathcal{B}'$. Dicha perspectiva además contiene los elementos aleatorios utilizados por $\mathcal{B}'$.

Definición 3.1.13. Protocolo de Conocimiento Nulo Perfecto/Estadístico/Computacional

Sean $L \subset \{0,1\}^*$ un lenguaje binario, $[\mathcal{A}, \mathcal{B}]$ un protocolo y $\mathcal{B}'$ un verificador deshonesto. Se dice que $[\mathcal{A}, \mathcal{B}]$ es un protocolo de conocimiento nulo perfecto/estadístico/-computacional para toda ITM probabilista polinomial $\mathcal{B}'$ si se verifica que la familia de v. a. $View_{\mathcal{A},\mathcal{B}'} = \{View_{\mathcal{A},\mathcal{B}'}(x,y)\}$ es perfectamente/estadísticamente/computacionalmente aproximable en $L' = \{(x,y)|\ x \in L, |y| = |x|^c$ para alguna constante fija $c > 0\}$.

La definición anterior trasladada a los sistemas de demostración interactivos queda de la siguiente forma:

Definición 3.1.14. Demostración de Conocimiento Nulo Perfecto/Estadístico/Computacional ($PZKP/SZKP/CZKP$, Perfect/Statistical/Computational ZKP)

Dado un lenguaje binario $L \subset \{0,1\}^*$, se dice que $[\mathcal{A}, \mathcal{B}]$ determina una Demostración de Conocimiento Nulo Perfecto para L (Estadístico/Computacional) si es un IPS para L y además es un protocolo de conocimiento nulo perfecto (estadístico/-computacional) para L.

La relación entre las tres modalidades de conocimiento nulo anteriores es obvia, toda $PZKP$ es una $SZKP$ y a su vez toda $SZKP$ es una $CZKP$. La mayoría de las ZKP propuestas en la bibliografía pertenecen a la categoría de $CZKP$, incluidas las que se proponen en la presente memoria. Esto es debido a que esta propiedad es suficiente para garantizar la imposibilidad práctica de extracción de información sensible por parte de cualquier verificador polinomial a través de la interacción. Además la mayoría de ZKP son probablemente seguras ya que los problemas utilizados pertenecen a la clase $NP - completa$, lo que conlleva la no existencia de $PZKP$ para ellos demostrada en [For87]. Dicho resultado implica que en el caso de que existiera dicha $PZKP$, la jerarquía de tiempo polinomial colapsaría.

Sin embargo, es de destacar que sí se han conseguido diseñar argumentos de conocimiento nulo perfecto para problemas de esta clase [BCC88].

Siguiendo la nomenclatura usada en capítulos previos se debe mencionar que la combinación de las dos primeras propiedades que debe satisfacer un protocolo para ser considerado una ZKP, completitud y solidez, es lo que garantiza la corrección del protocolo. En cuanto a la privacidad, se debe indicar que queda garantizada a través de la propiedad de conocimiento nulo, es decir queda sujeta a la construcción del simulador.

Una cuestión a destacar en cuanto a las propiedades previas, es su relación con los participantes, es decir a quién están asociadas. Por ejemplo la propiedad de completitud establece el comportamiento tanto de $\mathcal{A}$ como de $\mathcal{B}$ en el modelo semi-honesto, acotando en ese caso la probabilidad de aceptación de este último. La condición de solidez protege al verificador ante la posibilidad de que un probador deshonesto intente convencerle de la aceptación de afirmaciones erróneas. Por último, la propiedad de conocimiento nulo a quien protege es a la probadora $\mathcal{A}$, asegurándole que independientemente del verificador con el que interaccione, éste no obtendrá información adicional alguna que no pueda obtener sin su participación en el protocolo. De esta manera la verificación de las tres propiedades garantiza la tolerancia a fallos del protocolo en el modelo semi-honesto.

Posteriormente se observa, al igual que sucedió en la definición de sistema de demostración interactivo y en la de argumento, la necesidad de contemplar la incorporación de una entrada auxiliar asociada a cada participante. La justificación de la inclusión de la entrada auxiliar asociada al verificador viene dada por el hecho de que se debe garantizar la no transferencia de información incluso en el caso de utilizar una ZKP como parte integrante de otro protocolo más complejo, o bien si ha habido ejecuciones previas del mismo. En cuanto a la entrada auxiliar de la probadora $\mathcal{A}$ suele estar relacionada con la generación de la cadena x puesto que suele ser $\mathcal{A}$ quien

la genera, y por tanto, quien posee un conocimiento adicional sobre la misma.

Gracias a la inclusión de la entrada auxiliar en la definición, todo protocolo de conocimiento nulo continua siéndolo incluso después de realizar repetidas iteraciones, es decir, la propiedad de conocimiento nulo es cerrada frente a la composición secuencial.

Sin embargo, no se puede garantizar lo mismo cuando dicho protocolo se ejecuta en paralelo (ver Sección 3.4). Ante la necesidad de buscar alternativas a las ZKP que garanticen la privacidad incluso cuando se realizan ejecuciones diferentes en modelos paralelos surgen las nociones de *testigos indistinguibles* y *testigos ocultos*, [Fei90]. La noción de testigos indistinguibles está asociada a protocolos bipartitos en los que $\mathcal{A}$ usa uno o más testigos de un problema de decisión de la clase NP, y $\mathcal{B}$ es incapaz de decidir cuál es el testigo realmente usado por $\mathcal{A}$.

Las principales diferencias entre las ZKP y los protocolos de testigos indistinguibles que se definen a continuación son la posibilidad de ejecuciones en paralelo, la restricción a participantes polinomialmente acotados y sobre todo la no necesidad del simulador.

De la definición formal incluida a continuación se deduce que cualquier protocolo de conocimiento nulo se puede considerar de testigos indistinguibles. Se parte de una relación R comprobable en tiempo polinomial formada por los elementos $\{x, w\}$ donde la longitud de ambos está relacionada también polinomialmente, y para cualquier x se tiene que su conjunto de testigos $w(x)$ es el conjunto de elementos w tal que $(x, w) \in R$. Así, el lenguaje definido por $L = \{x/\exists w \ t.q.(x, w) \in R\}$ es un lenguaje de la clase NP.

Definición 3.1.15. Testigos Indistinguibles (WI, Witness Indistinguishability)
El sistema de demostración interactivo $[\mathcal{A}, \mathcal{B}]$ se dice que es de testigos perfectamente (estadísticamente/computacionalmente) indistinguibles para una relación R, si para todo verificador deshonesto $\mathcal{B}'$, para cualquier cadena x lo suficientemente larga, para

cualesquiera elementos del conjunto de testigos de x, $w_1, w_2 \in w(x)$, y para toda entrada auxiliar y de $\mathcal{B}'$, las distribuciones de las v.a. $View_{\mathcal{A},\mathcal{B}'}(x, w_1)$ y $View_{\mathcal{A},\mathcal{B}'}(x, w_2)$, generadas como las transcripciones de $\mathcal{B}'$, son perfectamente (estadísticamente/computacionalmente) indistinguibles.

Otra definición alternativa propuesta por Feige [Fei90] es la conocida como de testigos ocultos. Este nuevo concepto surge debido a que la aplicación de las ZKP como esquemas de identificación garantiza que ninguna información es revelada, pero es posible diseñar alternativas más flexibles y tales que basta con garantizar que no se transfiere información directamente relacionada con el proceso de identificación (contraseñas, nombres de usuarios, etc). De esta idea surgen los conceptos de *información transferible* y *testigos ocultos*, [FFS88].

Definición 3.1.16. Testigos Ocultos
Un protocolo se considera de testigos ocultos si el verificador no puede obtener un testigo para la entrada común a partir de su interacción con $\mathcal{A}$.

También en este caso se puede deducir que todo protocolo de conocimiento nulo es de testigos ocultos.

Otros protocolos relacionados con las ZKP son las *demostraciones de mínimo descubrimiento* sugeridas en [BCC88] y [Bra88]. De hecho son protocolos similares a las ZKP pero también en este caso se relaja la condición de transferencia de información.

Definición 3.1.17. Demostración de Mínimo Descubrimiento (MDP, Minimum Disclosure Proof)
Una Demostración de Mínimo Descubrimiento garantiza que el verificador no obtiene pista ninguna sobre la información cuya posesión desea demostrar $\mathcal{A}$. Además se supone que los participantes tienen sus capacidades computacionales acotadas polinomialmente, por lo que la no transferencia de información se garantiza sólo mientras

el protocolo está teniendo lugar.

También en el caso de las demostraciones de conocimiento nulo de conocimiento, igual que en el caso de los IPS, se puede dar una definición formal basada en el reconocimiento de lenguajes. En los esquemas ZKP la probadora $\mathcal{A}$ se contenta con demostrar que una determinada cadena pertenece a un lenguaje, o lo que es lo mismo dada una instancia de un problema de decisión demuestra que dicha instancia es resoluble. En el caso de la $ZKPK$ el objetivo de la probadora es convencer a su contrario de que *conoce* una solución de dicha instancia. Es decir, en este caso se supone que ambos participantes saben previamente que la instancia en cuestión tiene solución.

Modelar formalmente el hecho de que $\mathcal{A}$ conoce determinada información es lo que hace que la definición de estos esquemas (y también de las Demostraciones de Conocimiento Nulo) sea tan compleja. Decir en ZKP que la probadora conoce cierta información indica la existencia de un algoritmo polinomial conocido como extractor (análogo al simulador en la propiedad de conocimiento nulo), que es capaz de utilizar a la probadora como una MTO para obtener con una determinada probabilidad la información conocida por ella.

La manera de proceder en las $ZKPK$ conlleva normalmente una etapa previa de construcción de la instancia del problema a utilizar. Esta etapa de pre-cálculo o configuración la desarrolla normalmente $\mathcal{A}$, garantizándose así el conocimiento por su parte de una solución para una instancia del problema. Este hecho junto con el uso habitual de problemas de la clase NP hace que las $ZKPK$ sean apropiadas para su uso como esquemas de identificación.

La mayoría de los algoritmos que se proponen en este capítulo constituyen realmente $ZKPK$, aunque nos referiremos a ellos como ZKP con el único objetivo de simplificar la nomenclatura.

Otro sistema de demostración interesante son las demostraciones probabilistas

comprobables.

Definición 3.1.18. Demostración Probabilista Comprobable (PCP, Probabilistic Checkable Proof)
Dado un lenguaje, dicho sistema se identifica con una máquina de Turing oráculo probabilista polinomial (el verificador), denotado por M satisfaciendo las propiedades de completitud y solidez, enunciadas ahora de la siguiente manera:

- *Completitud*: Para toda $x \in L$ existe una OTM ξ_x tal que: $Prob[\mathcal{B}^{\xi_x(x)}] = 1$, donde $\mathcal{B}^{\xi_x}$ denota la salida de la máquina $\mathcal{B}$ para la entrada x accediendo al oráculo ξ_x.

- *Solidez*: Para cada $x \notin L$ y cualquier OTM ξ: $Prob[\mathcal{B}^{\xi}(x) = 1] \leq 1/2$ donde la probabilidad se toma sobre la cinta aleatoria de $\mathcal{B}$.

3.2. Estado del Arte

En esta sección se hace un breve recorrido sólo por aquellos trabajos más significativos relacionados con las demostraciones de conocimiento nulo y sus variantes. Esto es así porque el número de publicaciones en la materia es tan extenso que se hace impracticable una revisión bibliográfica más exhaustiva.

Tanto el concepto de sistema de demostración interactivo (IPS) como el de demostración de conocimiento nulo aparecen en una versión preliminar ([GMR85] presentada en el STOC 1985) del trabajo [GMR89]. En ambos trabajos Goldwasser, Micali y Rackoff dan las nociones básicas proporcionando además varias $PZKP$ para diferentes lenguajes, por lo que se pueden considerar dos de los trabajos fundamentales en este área. También dichos trabajos son de las primeras referencias en las que se usan problemas de la teoría de grafos como son el problema del isomorfismo de grafos y el del no isomorfismo para diseñar ZKP. La primera y más inmediata aplicación

de las ideas allí expuestas fue el establecimiento de resultados relacionados con las clases de complejidad ($IP = PSPACE$).

Con el trabajo de Goldreich, Micali y Wigderson [GMW86] la importancia de las ZKP se intensifica debido a que los autores no sólo demuestran la existencia de $CZKP$ para todo lenguaje de la clase NP, proporcionando una $CZKP$ para el problema $NP-completo$ de la 3-coloración de grafos con la única hipótesis subyacente de que existen funciones seguras de cifrado, sino que además apuntan como primera aplicación general su utilización en el diseño de protocolos, proporcionando resultados que son susceptibles de ser usados como metodología general en el área del diseño de protocolos criptográficos. Posteriormente, en [BOGG$^+$89], bajo la misma hipótesis, se demuestra que cualquier lenguaje que admita un sistema de demostración interactivo también posee una $CZKP$. Otro resultado que aparece en este último trabajo se refiere a la existencia de $PZKP$, concluyendo que al sustituir la hipótesis anterior por la de la existencia de unos esquemas particulares de compromisos de bits se garantiza que cualquier lenguaje que admita un sistema de demostración interactivo también posee una $PZKP$.

Otro trabajo de obligada referencia es el de [Blu86] debido a que en él se usan también problemas de la Teoría de Grafos, en concreto el del circuito Hamiltoniano y el de la 3-coloración, proponiendo para ambos casos una nueva propuesta de ZKP. Blum argumenta que su propuesta basada en el circuito Hamiltoniano es más eficiente que la de la 3-coloración propuesta en [GMW86] debido a que la suya requiere un menor número de interacciones entre $\mathcal{A}$ y $\mathcal{B}$ para alcanzar el mismo nivel de seguridad. Además en su propuesta para la 3-coloración el número de retos usados en cada iteración es 2 a diferencia del protocolo que aparece en [GMW86], coincidiendo en este último caso el número de retos con el numero de aristas que posee el grafo de entrada.

Goldwasser y Sipser demuestran en [GS86] que las dos definiciones de IPS y AMG

son equivalentes en cuanto a reconocer un lenguaje, es decir, si un lenguaje tiene un sistema de demostración interactivo en el que el número de mensajes intercambiados está acotado polinomialmente en función de la longitud de la entrada, dicho lenguaje también posee un juego de Arturo-Merlín cuando el número de mensajes intercambiados posee una cota muy cercana a la anterior.

La primera aplicación práctica directa de los ZKP planteada fue la de la utilización de la variante $ZKPK$ en los esquemas de identificación surgiendo varias propuestas en este sentido [FS86], [FFS88], [GQ88], [Bet88], [Sch89a], etc. En el primero de los trabajos, [FS86], de Fiat y Shamir se propone un esquema de identificación y de firma digital que utiliza como base de su seguridad la intratabilidad del problema de la extracción de raíces modulares. Su importancia radica en ser uno de los primeros trabajos en el que se apunta la implementación en tarjetas inteligentes, y en que el protocolo propuesto como esquema de identificación es más eficiente que sus homólogos basados en el esquema de cifrado RSA. Curiosamente uno de los mayores revuelos de la comunidad criptográfica fue causado por la solicitud por parte de los autores de patentar el protocolo propuesto en el trabajo [FFS88], (incluso el New York Times se hizo eco de la noticia [Lan88]). La respuesta del Departamento de Defensa del Gobierno de los Estados Unidos llegó cuando ya el trabajo había sido aceptado para su publicación e incluso había sido presentado en diferentes congresos. Dicha respuesta fue la consideración de que la revelación de dicho trabajo sería perjudicial para la seguridad nacional, instando a los autores a ponerse en contacto con todas aquellos ciudadanos americanos que conocían el trabajo para comunicarles que en caso de revelar el protocolo incurrirían en un delito sujeto al pago de multas e incluso a penas de cárcel. La intervención de la comunidad criptográfica y de la propia NSA (National Security Agency) consiguió la rescisión de la orden de secreto.

El trabajo de Guillou y Quisquater [GQ88] propone una solución al problema de la identificación relacionada con la presentada en [FS86] y también orientada a su

implementación en tarjetas inteligentes, consiguiendo una reducción en el número de iteraciones a realizar y manteniendo en niveles aceptables el parámetro definido por la probabilidad de que un usuario ilegítimo se identifique correctamente, a costa de utilizar operaciones con tiempo de computación mayor.

Una cuestión que varía de unas definiciones a otras son las restricciones sobre las capacidades computacionales de los participantes. Así por ejemplo, en la definición incluida en [FFS88], ambos participantes se restringen a TM polinomiales, mientras que en [BG92] las capacidades de la probadora quedan sin restricción.

Durante estos años también se publican múltiples trabajos relacionados con las variantes de las ZKP tales como [BCC88] y [BC89]. De entre ellos merece especial atención el trabajo [BFM88] en el que se propone el concepto de demostración de conocimiento nulo no interactiva.

Desde 1989, han aparecido varios esquemas más eficientes con la peculiaridad de que su seguridad está basada en problemas combinatorios $NP - completos$. Por ejemplo Shamir publica en [Sha89] el primer esquema de identificación basado en un problema $NP - duro$, el problema del Kernel Permutado.

La definición de Sistema de Demostración Interactivo aparece por primera vez en [GMR89], como respuesta a la necesidad de formalizar los procedimientos de verificación en el ámbito de los protocolos criptográficos.

El concepto de demostración de conocimiento nulo de conocimiento surge en [GMR89] aunque el establecimiento de una definición formal no fue inmediato, [FFS88], [FS90], [TW87], [BG92], debido a lo complejo de su formalización. En las primeras definiciones de $ZKPK$ se incluyen las propiedades de completitud y solidez, siendo en la definición de esta última donde se describe la posible intervención del extractor, [FFS88].

Por otra parte, las referencias [BCC88] y [MM90] proporcionan respectivamente

una útil discusión sobre demostraciones de mínimo descubrimiento basadas en compromiso de bits, y un extenso recorrido por la materia. Además, el artículo [QGAB89] da una introducción no matemática al tema, mientras que el libro [Gol99b] aporta una extensa revisión de las demostraciones probabilistas.

Posteriormente, en 1990 [LS90] propone una nueva demostración de conocimiento nulo basada en el problema de los circuitos hamiltonianos, de nuevo generalizando dichos esquemas. También aparecen extensiones del modelo básico de sistemas de demostración en referencias tales como [Gol99a] y [Bea91] en las que se trasladan a situaciones donde existe más de un probador interactuando con un único verificador. Este nuevo modelo permite la obtención de $PZKP$ para cualquier lenguaje de NP sin tener que asumir hipótesis alguna.

Uno de los trabajos relacionados con los sistemas de demostración interactivos que merece ser destacado es el publicado en 1992 por Shamir [Sha92] en el que demuestra que la clase de lenguajes que poseen sistemas de demostración interactivos (clase IP) coincide con $PSPACE$. La importancia de este resultado en el ámbito de la complejidad computacional se debe a que la introducción de estos sistemas permite el reconocimiento de lenguajes de una clase más amplia que NP.

Dos aproximaciones de ZKP sobre problemas $NP-duros$ son las propuestas por Stern, basadas respectivamente en un problema de codificación, el de la Decodificación Síndrome [Ste93], y en el problema combinatorio de las Ecuaciones Lineales con Restricciones [Ste94]. Más tarde, Pointcheval [Poi95] propone otro esquema de identificación basado en el problema de la clase $NP-duro$ denominado problema de los Perceptrones Permutados perteneciente a la Teoría de Máquinas de Aprendizaje. En todos estos esquemas de identificación, la dificultad de los problemas basados garantizan el cumplimiento de las tres propiedades descritas en la definición de protocolo de conocimiento nulo [Pou97]. Sin embargo, un problema común a todos los esquemas anteriores es el gran número de bits de información a intercambiar entre la

probadora y el verificador, lo que reduce considerablemente la eficiencia de los protocolos. Este hecho se debe al número de iteraciones requeridas en el algoritmo, lo que está directamente relacionado con la gran probabilidad de fraude en cada iteración.

En la década de los noventa se hace un gran esfuerzo en la dirección de determinar si una ZKP mantiene la propiedad de conocimiento nulo ante ejecuciones en paralelo. Resultados obtenidos por diferentes autores, [Fei90] y [GK96a], establecen incluso la revelación de información en tales circunstancias cuando ambos participantes están polinomialmente acotados.

Algunos de los trabajos más recientes [CGGM00] versan sobre lo que se ha dado en llamar conocimiento nulo concurrente (ver Sección 3.4), y reiniciable, así como sobre la posibilidad de acotar el numero de iteraciones e intercambios de mensajes (3.13).

Una de las referencias más significativas del área en los últimos años es sin duda el trabajo de Barak [Bar01]. En dicho trabajo se modifica el modelo utilizado para el simulador que garantiza la propiedad de conocimiento nulo huyendo de los simuladores clásicos (ver Sección 3.3) para así poder generar argumentos de conocimiento nulo que preservan dicha propiedad ante ejecuciones concurrentes.

Algunos autores como [BCC88] y [Gol99a] consideran en la definición de los argumentos la posible utilización de entradas auxiliares de manera que para toda cadena del lenguaje siempre exista alguna entrada auxiliar que permita a $\mathcal{A}$ convencer a $\mathcal{B}$ de que acepte la cadena con determinada probabilidad. Recientemente se han planteado nuevas alternativas relacionadas con los argumentos y los sistemas computacionalmente sólidos [BG02]. Por último, también han demostrado la existencia de protocolos WI con un número constante de intercambio de mensajes para todo elemento de la clase NP.

3.3. Paradigma de la Simulación

Informalmente se definen las ZKP como demostraciones que no aportan información alguna salvo la garantía de validez de la información cuestionada. De esta manera, demostrar la propiedad de conocimiento nulo conlleva demostrar que todo lo que puede ser eficientemente computado por el verificador usando la interacción con la probadora puede ser computado por él mismo de manera aislada. Para poder hacerlo se necesita considerar la perspectiva o transcripción de la interacción entre $\mathcal{A}$ y $\mathcal{B}$.

Por tanto, para demostrar que un protocolo es $PZKP(SZKP/CZKP)$ se debe especificar el comportamiento del simulador, y posteriormente demostrar que para cualquier verificador acotado polinomialmente $\mathcal{B}^*$ (incluso los que actúan de manera deshonesta) la distribución que se genera sobre el conjunto de mensajes intercambiados durante la interacción con $\mathcal{A}$ para una entrada dada es totalmente (estadísticamente/computacionalmente) indistinguible de la distribución generada por el simulador a partir de la misma cadena de entrada. Todo ello sin que el simulador tenga interacción alguna con $\mathcal{A}$.

El primer problema surge del hecho de que según la definición parece que para cada verificador (acotado polinomialmente) debe existir un simulador también polinomial, pero existe la posibilidad de definir un único modelo de simulador válido para cualquier verificador. Así se comienza a hablar del concepto de *simulador universal*. La mejor manera de intentar construir este simulador es obteniéndolo a través de algún procedimiento eficiente que aproveche las especificaciones del programa de cada verificador.

Una alternativa de solución a este problema se basa en la idea de máquina de Turing oráculo. Esta manera de definir el simulador se basa en hacer que éste utilice al verificador $\mathcal{B}^*$ como una caja negra reiniciable [BC89] para la simulación de la

interacción, teniendo control sobre él. En otras palabras, el simulador escoge tanto la entrada aleatoria de $\mathcal{B}^*$, como los supuestos mensajes enviados por $\mathcal{A}$, facilitando toda esta información al verificador $\mathcal{B}^*$. Este tipo de simulador recibe el nombre de simulador de caja negra, y fue introducido por [Ore87].

En versiones posteriores de esta definición [Gol01a] se destaca el hecho de que ésta realmente no depende en absoluto del verificador $\mathcal{B}^*$, sino de la estrategia desarrollada por la probadora $\mathcal{A}$ puesto que se define un único simulador para cualquier verificador acotado polinomialmente, siendo las simulaciones generadas válidas para cualquier cadena de entrada.

Esta metodología ha sido extendida al campo de los protocolos en general [Can00] y a los esquemas de cifrado [Gol98]. La principal ventaja que posee el trabajar con esta aproximación del simulador estriba en que de esta manera se puede estudiar la seguridad del protocolo independientemente de las capacidades del verificador.

Sin embargo, recientemente [Bar01] Barak ha demostrado que la creencia de que no poder demostrar ciertas propiedades utilizando simuladores de caja negra se debía a las propias limitaciones de los protocolos de conocimiento nulo, era infundada. En dicho trabajo Barak presenta una alternativa a la de los simuladores de caja negra que le permite probar la existencia de argumentos de conocimiento nulo para todos los problemas de la clase NP. Además hay que destacar que la hipótesis que se utiliza en este importante resultado, es la existencia de funciones resumen resistentes a colisiones [Pre00]. Dicha suposición le permite la verificación simultánea de una serie de propiedades cuya concurrencia se había demostrado imposible en la simulación de caja negra. Estas propiedades son:

- Tener un número constante de intercambio de mensajes
- Mantener la propiedad de conocimiento nulo frente a un número determinado de ejecuciones concurrentes

- Ser un juego de Arturo-Merlín
- El simulador diseñado tiene asociado un tiempo estrictamente polinomial en lugar de un tiempo esperado polinomial

Todas estas propiedades se consiguen a través de la composición de dos protocolos básicos. El primero de ellos es un juego de Arturo-Merlín y el segundo una demostración de testigos indistinguibles.

3.4. Interacción y Paralelización

En esta sección se tratan dos de los aspectos más analizados en la bibliografía sobre ZKP, relativos al grado necesario de interacción y a la posibilidad o no de paralelización de las propuestas. Los principales trabajos que tratan el tema de la interacción aparecen poco después de la introducción de la ZKP. Concretamente los trabajos [BFM88], [DSMP87], [DSMP89], [BG89], [LS90] y [BSMP91] son los primeros y se pueden considerar los más significativos. El modelo usado para formalizar esta idea consiste en considerar al verificador como una DTM polinomial.

La noción de interacción se estudia por primera vez en [BFM88] ante la necesidad de reducir el costo de las comunicaciones entre los participantes de protocolos en los que se necesite incluir una ZKP. Para eliminar la interacción se propone la utilización de una cadena aleatoria compartida por ambos usuarios (que es el ingrediente principal) convirtiendo la comunicación en unidireccional de forma que sólo la probadora envía información al verificador. Una posibilidad para generar esa cadena aleatoria compartida es la incorporación al esquema de una función hash criptográfica, siendo su única entrada todos los compromisos generados por la probadora. En la sección 3.7 se incluye la descripción de una nueva propuesta no interactiva que hace uso de esta idea basada en funciones resumen.

En las referencias sobre el tema se sugiere la necesidad de aumentar el número de

iteraciones en este tipo de protocolos para alcanzar un nivel de seguridad aceptable desde el punto de vista del verificador [Sch93]. En este artículo se proporciona una demostración de conocimiento nulo no interactiva ($NIZKP$, Non Interactive ZKP) para el problema de la 3-coloración, demostrando así que cualquier lenguaje de la clase NP posee una demostración de este tipo. Este resultado se obtiene para una determinada longitud de la cadena aleatoria compartida. Los trabajos posteriores mencionados presentan mejoras en la dirección de la reducción de la longitud de la cadena aleatoria, y también las hipótesis se hacen menos exigentes.

Particularmente interesantes son las denominadas demostraciones de conocimiento nulo no interactivas *públicamente verificables*, que se caracterizan por ser verificables para cualquier participante en lugar de estar dirigidas a un único verificador [BG89]. Las principales aplicaciones de este nuevo modelo están relacionadas con las firmas digitales y la autenticación de mensajes [BG89]. Quizás el resultado más general es el incluido en [LS90] donde se demuestra la existencia de una $NIZK$ públicamente verificable para cualquier lenguaje en NP partiendo de la hipótesis de que existen las así denominadas permutaciones unidireccionales. También se plantea allí una demostración de conocimiento nulo no interactiva para el problema del circuito hamiltoniano apuntando como posible la extensión de este tipo de demostraciones a otros problemas planteados sobre grafos. Dando un paso más adelante, en [FLS90] se establece la construcción de transformaciones que utilizando la noción de testigos indistinguibles facilitan la generación de $NIZK$ válidas frente a un número indeterminado de verificadores de capacidades computacionales acotadas polinomialmente que comparten una misma secuencia aleatoria.

El diseño de la mayoría de los protocolos de conocimiento nulo definidos para demostrar el conocimiento de una solución de un problema se ajusta a la técnica de reto-respuesta, estableciéndose el esquema general en tres pasos. En el primero de ellos la probadora transforma el problema original utilizando para ello el azar y

una transformación unidireccional. A continuación el verificador solicita un testigo que confirme que la transformación ha sido correcta o bien una solución del nuevo problema. La probadora responde con la información que le fue solicitada. Este sencillo esquema hace que en principio la paralelización parezca posible y sencilla. En una primera aproximación, para llevar a cabo la paralelización propuesta basta con que la probadora envíe todas las posibles transformaciones aleatorias del problema original al verificador al comienzo de la ejecución del protocolo, y que $\mathcal{B}$ realice a su vez todas sus solicitudes paralelamente, acabando el protocolo con la transferencia también en paralelo de la información solicitada por $\mathcal{B}$.

En este caso se asume un modelo síncrono para las comunicaciones en todas las ejecuciones ya que el envío de mensajes debe estar sincronizado.

Uno de los principales motivos por los que la paralelización de los protocolos de conocimiento nulo ha sido un tema tan estudiado en la bibliografía es para intentar reducir el número de iteraciones que realizan los participantes durante el intercambio de información. Además, suele ser común diseñar protocolos incluyendo a su vez otros protocolos, por lo que en estos casos es imprescindible garantizar la conservación de la propiedad de conocimiento nulo.

El problema principal de la paralelización radica en la posibilidad de que en una ejecución el usuario seleccione un reto y en otra seleccione su complementario, de manera que al unir las dos porciones de información obtenidas, el secreto pueda verse en peligro.

En general, la ejecución en paralelo de un protocolo de conocimiento nulo no conserva dicha propiedad [GK96b]. Feige proporciona en [Fei90] una descripción de dicho problema incluyendo una ZKP con tres participantes, que deja de verificar la propiedad de conocimiento nulo cuando se ejecuta en paralelo dos veces. En dicha propuesta el mismo participante interviene a la vez en las dos ejecuciones de forma que en una actúa como probador y en la otra como verificador, aprovechando su

situación para hacer de intermediario entre los otros dos participantes consiguiendo así la información cuestionada, y rompiendo la propiedad de conocimiento nulo.

Otra noción relacionada con la paralelización de los protocolos es la de composición concurrente. Esta noción generaliza la composición de ejecuciones paralelas puesto que se refiere a la situación en que varias instancias de un mismo protocolo se ejecutan en un modelo de comunicación totalmente asíncrono. El primer trabajo en el que se estudia la concurrencia de las ZKP es [DNS98]. En este trabajo se considera un modelo más restrictivo que el totalmente asíncrono, demostrando la existencia de ZKP y argumentos de conocimiento nulo resistentes a ejecuciones concurrentes. En el modelo general se han obtenido algunas ZKP para problemas NP resistentes a ejecuciones concurrentes, pero continúa sin respuesta la pregunta de si todo lenguaje de NP posee ZKP resistentes a ejecuciones concurrentes manteniendo constante el número de intercambios de mensajes.

3.5. Esquema de Identificación Determinista

En esta sección se describen brevemente algunas de las soluciones conocidas para el problema de la identificación determinista, para a continuación proponer un esquema propio [CH01]. El objetivo de los esquemas de identificación es permitir a un participante (el verificador) asegurarse de la identidad de otro (la probadora). Uno de los principales propósitos de estos esquemas es facilitar el control de accesos a sistemas o recursos. Una extensa discusión sobre esta materia se puede encontrar en [dWQ93].

En [FS86] se incluye una clasificación de los esquemas de identificación existentes en función del nivel de seguridad que resulta conveniente a la hora de decidir qué tipo de sistema se implanta en función de las necesidades de seguridad existentes. Así, cuando la usuaria $\mathcal{A}$ prueba su identidad ante el sistema denotado por $\mathcal{B}$ se habla de sistemas de autenticación donde ningún otro individuo puede suplantar a la usuaria

$\mathcal{A}$ frente a $\mathcal{B}$, mientras que se habla de esquemas de identificación cuando se exige además que $\mathcal{B}$ no pueda hacerse pasar por $\mathcal{A}$ ante terceras partes.

La técnica más común en identificación se basa en sistemas de autenticación que usan contraseñas secretas e invariables que son compartidas entre cada usuario y el sistema. En estos esquemas la probadora $\mathcal{A}$ muestra su contraseña al verificador $\mathcal{B}$, el cual comprueba la correspondencia entre dicha información y la contraseña que tiene almacenada para dicha usuaria. Este débil esquema se corresponde con una demostración de máximo descubrimiento y su principal inconveniente es la posible intercepción de la contraseña secreta y su posible consiguiente uso.

Dos soluciones bien conocidas para este problema son las siguientes. El método más débil se basa en listas de contraseñas de un único uso compartidas por sistema y usuario. Estos esquemas son seguros frente a adversarios pasivos que se limitan a espiar para luego intentar suplantar identidades. En la mayoría de casos, dichas listas de contraseñas secretas compartidas son usadas de forma secuencial, bien directamente o bien mediante la aplicación de funciones unidireccionales. En ambos casos usuaria y sistema tienen que compartir alguna información secreta lo que conlleva los problemas de la distribución de secretos y de la indistinguibilidad entre ambos participantes.

Por otra parte, la solución más robusta consiste en utilizar la idea de los mencionados protocolos de reto-respuesta, [CH01], [HCB02a]. De este modo $\mathcal{A}$ convence a $\mathcal{B}$ de que posee cierta información sin revelarla. Esta idea se corresponde con la de las demostraciones de mínimo descubrimiento, que se han implementado haciendo uso de cifrados de clave secreta y de clave pública, y demostraciones de conocimiento nulo, [FFS88], [GQ88], [Bet88], [Sch89a]. Este tipo de protocolos poseen características que los hacen especialmente válidos para su aplicación en identificación:

- No es necesario distribuir ningún secreto entre los participantes.

- El verificador, en este caso el sistema, no necesita almacenar ninguna información secreta, con lo que se puede reducir el nivel de seguridad del mismo.
- Permite al sistema verificador demostrar que un usuario ha accedido al mismo, a través de la interacción.
- En el caso de las ZKP, la identificación es probabilista.

El esquema presentado a continuación es una propuesta determinista orientada a la identificación de usuarios. Se caracteriza por ser una sencilla combinación de los conceptos de contraseña de un único uso y clave pública. Posteriormente, en la sección 3.12 se encuentra un esquema de identificación fuerte que combina los dos conceptos anteriormente mencionados con una demostración de conocimiento nulo. La importancia de este tipo de esquemas hace que su seguridad sea objeto de estudios continuos [BP02].

Tanto en el esquema determinista como en el probabilista presentados en la presente memoria, los usuarios que participan poseen cierta información pública que debe ser mantenida en un directorio público certificado. En esta base de datos, en el caso de la identificación determinista que se define a continuación cada usuario publica un problema P y un grafo G, o bien la imagen mediante una función resumen del mismo, información que es válida sólo para una identificación, por lo que debe ser actualizada una vez haya sido utilizada. De esta manera, cada vez que una probadora $\mathcal{A}$ establece contacto con un verificador $\mathcal{B}$ para identificarse, éste debe obtener la información asociada a $\mathcal{A}$ de la base de datos pública, la cual cambia tan pronto como el proceso de identificación termine.

Por otro lado, a lo largo de ambos protocolos, cada contraseña de identificación secreta se corresponde con una solución de un problema de grafos difícil. Debe prestarse especial atención a la elección adecuada de soluciones, grafos, funciones y representaciones de problemas usados debido a los requisitos de memoria. Cada solución se

simboliza por el valor decimal de su representación binaria donde el valor uno indica la presencia de los correspondientes elementos de la solución. Así, la complejidad de las comunicaciones no es un gran problema porque sólo implica el envío de un vector binario cuya longitud depende de la clase de problemas usados.

Además de los grafos, una relación de posibles problemas puede ser también publicada en el mismo directorio permitiendo el uso de un código corto correspondiente a cada problema. La selección de la representación del grafo depende del problema utilizado y del proceso de verificación. Así, la matriz de adyacencia es adecuada si el conocimiento de la presencia de ciertas aristas es necesaria, lo que se corresponde con los casos aquí estudiados. Una alternativa interesante consiste en representar las filas y columnas de dicha matriz por vectores binarios. Otra posibilidad es usar listas de adyacencia, pero esta no es la mejor opción para la mayoría de los problemas sugeridos aquí.

A continuación en la figura 3.1 se propone el esquema de identificación descrito [CH01]. El esquema mostrado no es probabilista, de manera que después de aplicarlo, no existe duda alguna sobre la identidad de la usuaria $\mathcal{A}$ gracias a la dificultad del problema y la correcta elección de los parámetros. Otra ventaja de esta propuesta es que no hay necesidad de compartir ninguna información secreta entre ambos participantes, en contraste con otros esquemas bien conocidos basados en contraseñas de un único uso.

En este entorno se podría considerar que el grafo es el login del usuario y la solución del problema es la contraseña.

Otra utilidad de este esquema es su uso como función unidireccional, ya que dado un grafo y un problema difícil es prácticamente imposible encontrar la solución.

Obsérvese que con este protocolo, una vez usada una solución S, ésta pierde su condición secreta, así que es absolutamente necesario actualizarla después de cada proceso de identificación. Así, durante la fase de cálculo off-line, $\mathcal{A}$ debe desarrollar

Configuración: $\mathcal{A}$ realiza las siguientes acciones:

- selecciona el problema P
- genera su contraseña secreta de un único uso (S) a través de la codificación de una solución adecuada
- construye un grafo G correspondiente al par (P, S)
- establece su identificación pública, (P, G)

Identificación : $\mathcal{A} \longrightarrow \mathcal{B}$ la solución S.

Verificación: $\mathcal{B}$ comprueba que S es una solución válida de acuerdo con la clave pública (P, G).

Figura 3.1: Esquema de identificación determinista

un paso de construcción que es completamente independiente de anteriores ejecuciones del esquema de identificación puesto que la solución cambia en cada iteración. Para hacer más eficiente este paso de construcción, en una sección posterior se propone un esquema probabilista donde la usuaria $\mathcal{A}$ puede aprovechar las ventajas de ejecuciones previas para construir nuevos grafos. Otra posibilidad consiste en utilizar grafos isomorfos en sucesivas ejecuciones.

Por último destacar que una ventaja del esquema descrito en este apartado es la posibilidad de generalizarlo para usar como base cualquier problema difícil de cualquier disciplina diferente a la Teoría de Grafos.

3.6. Esquema General

En la mayoría de las ZKP conocidas existen ciertas características comunes que se traducen en el esquema general que a continuación se describe [HC00b]. En primer lugar, son procesos iterativos por lo que en estos casos es importante acordar

previamente entre ambos participantes el número de iteraciones a desarrollar. Dicho parámetro denotado por m en el esquema presentado debe garantizar los intereses contrapuestos de ambos. Además se utilizan las técnicas de Corte y Elección y de Reto-Respuesta, apareciendo así un segundo parámetro que refleja el número de retos equiprobables e impredecibles disponibles durante la ejecución del protocolo correspondiente, que habitualmente toma el valor 2.

Dada la combinación de ambas técnicas mencionadas, la información secreta se divide en partes descritas por preguntas y respuestas que demuestran la posesión de la información, y sirven como garantía frente a posibles fraudes cometidos por $\mathcal{A}$. De acuerdo con la notación previa usada para los parámetros número de iteraciones y número de retos, el nivel de confianza del usuario $\mathcal{B}$ queda determinado por la probabilidad de que $\mathcal{A}$ consiga realizar una estafa, siendo dicha probabilidad de valor 2^{-m}. Usualmente el secreto que interviene en una ZKP es una solución S a un problema difícil y se admite la hipótesis de que $\mathcal{B}$ posee habilidades computacionales acotadas polinomialmente.

Utilizando todas estas ideas se puede establecer el esquema general $ZKP - GS$ para este tipo de protocolos [HC00a] dividido en cinco etapas y mostrado en la figura 3.2.

En este esquema la corrección del protocolo se estudia usando las propiedades de completitud y solidez ya mencionadas anteriormente. La completitud garantiza la correcta ejecución del protocolo cuando ambos participantes actúan correctamente mientras que la solidez en las ZKP protege a $\mathcal{B}$ en el caso de que $\mathcal{A}$ no conozca la información secreta. Por otra parte, la privacidad es alcanzada a través de la propiedad de conocimiento nulo que asegura que $\mathcal{B}$ no obtiene información alguna sobre el secreto a través de su participación en el protocolo.

A continuación se encuentran las descripciones y análisis de las ZKP propuestas como alternativas a las existentes en la bibliografía utilizando en todas ellas problemas

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ una partición de una instancia del problema de entrada $\{P_0, P_1\}$.

Iteraciones: Se iteran los siguientes pasos desde $i = 1$ *hasta* m :

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ un testigo asociado a una solución de una instancia aleatoria a_i, obtenido a través de una función unidireccional h, $h(a_i)$.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ un bit aleatorio b_i.

Respuesta: $\mathcal{A} \longrightarrow \mathcal{B}$ la solución del problema $P_j,\ j \in \{0, 1\}$, definida por ambas elecciones aleatorias a_i y b_i, y la solución secreta de $\mathcal{A}$, S, $Sol(a_i, b_i, S)$.

Verificación:
B verifica las propiedades de la solución recibida.

Figura 3.2: Esquema ZKP-GS

de la Teoría de Grafos. Las descripciones se adaptan al esquema presentado en este apartado, siendo la mayoría de ellas interactivas y pertenecientes a la categoría de las $CZKP$.

3.7. Algoritmos ZKP-GP y NIZKP-GP

En este apartado se usa el problema del isomorfismo para demostrar la posesión de una solución de un problema difícil cualquiera de la teoría de grafos en un grafo G [CH02a]. Nótese que ésta es la principal diferencia con el esquema propuesto por [GMR85], puesto que allí se propone una demostración de conocimiento nulo para demostrar la posesión de un isomorfismo entre dos grafos. De nuevo se requiere el uso de grafos suficientemente grandes y la etapa de compromiso implica la generación seudoaleatoria de los grafos que intervienen.

El algoritmo $ZKP-GP$ presentado en la Figura 3.3 se puede describir de acuerdo con la notación usada en el esquema $ZKP - GS$ de la siguiente forma:

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ el grafo G y el problema P.

Iteraciones: Se iteran los siguientes pasos desde $i = 1$ *hasta* m :

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ un grafo G_i, isomorfo a G.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ un bit aleatorio b_i.

Respuesta: $\mathcal{A} \longrightarrow B$:

- si $b_i = 0$, el isomorfismo $G \sim G_i$.
- si $b_i = 1$, $Sol_P(G_i)$.

Verificación: $\mathcal{B}$ comprueba la información recibida.

Figura 3.3: Algoritmo ZKP-GP

- La partición $\{P_0, P_1\}$ viene dada por $\{G, P\}$.
- El testigo $h(a_i)$ es el grafo G_i, y la elección secreta de $\mathcal{A}$, a_i es el isomorfismo $G \sim G_i$.
- La solución $Sol(a_i, b_i, S)$ se corresponde con el isomorfismo $G \sim G_i$ cuando $b_i = 0$, y con la solución $Sol_P(G_i)$ cuando $b_i = 1$.

Teorema 3.7.1. *El algoritmo $ZKP - GP$ es una Demostración de Conocimiento Nulo.*

Demostración. El cumplimiento de la propiedad de completitud es fácil de verificar. Si $\mathcal{A}$ y $\mathcal{B}$ son honestos, $\mathcal{B}$ recibe el correcto isomorfismo $G \sim G_i$ o una solución correcta de P en G_i. En cuanto a la propiedad de solidez, en el caso de que $\mathcal{A}$ no conozca la solución secreta, ella podría intentar estafar a $\mathcal{B}$ en cada iteración de dos maneras complementarias:

1. $\mathcal{A}$ puede usar un grafo G_i que no es isomorfo a G, pero este fraude tiene éxito sólo si el valor del bit seleccionado por $\mathcal{B}$ es 1.

2. $\mathcal{A}$ puede construir una falsa solución del problema en un grafo isomorfo G_i, pero en este caso, $\mathcal{B}$ detectaría el fraude cuando el valor del bit es 1.

De esta manera, el nivel de confianza al final del protocolo es del orden $O(2^{-m})$. También la propiedad de conocimiento nulo se verifica debido al hecho de que el poder computacional de $\mathcal{B}$ está polinomialmente acotado, por lo que no puede deducir ninguna información sobre $Sol_P(G)$ a través de exclusivamente una de las dos posibles respuestas, $G \sim G_i$ o $Sol_P(G_i)$. □

Una característica de este protocolo es la posibilidad de desarrollar los últimos tres pasos del algoritmo utilizando una $OT1C-2$ [CH02b] donde los dos secretos a transferir en cada iteración son la solución $Sol_P(G_i)$ y el isomorfismo $G_i \sim G$. Sin embargo esto supone costo adicional para el algoritmo ya que se cumple la propiedad de inconsciencia de la probadora $\mathcal{A}$, que no es requerida en la definición de ZKP.

A partir de la idea general del algoritmo anterior es fácil inferir la posibilidad de adaptarlo para usar una función hash en la construcción de los compromisos. En tal caso la probadora se compromete con la solución secreta elegida mediante una función hash pública y se realizan m iteraciones del esquema propuesto. Si el verificador solicita el primer reto, entonces se le entrega el isomorfismo entre el grafo de partida y el generado. Si el reto seleccionado es el segundo, entonces el verificador debe comprobar la viabilidad de la solución, y que la imagen mediante la función resumen coincide con la que la probadora le envió previamente. Para que el esquema planteado funcione correctamente la función hash utilizada debe producir la misma imagen sólo para soluciones isomorfas en grafos isomorfos. De esta manera, se estaría trabajando con la versión de recuento de soluciones, problema que es más difícil [Gol99a] que la correspondiente versión de búsqueda.

También el algoritmo $ZKP-GP$ puede adaptarse de forma sencilla a una propuesta no interactiva, usando una función hash tal y como se describe en la Sección

3.4. En la Figura 3.4 se muestra cómo mediante una función resumen se puede obtener el algoritmo no interactivo $NIZKP-GP$ que puede usarse indiscriminadamente para que $\mathcal{A}$ se identifique ante cualquier usuario, [HC01].

Configuración: $\mathcal{A}$ publica el grafo G, el problema P, y m grafos $G_i, i = 1, 2, ..., m$ isomorfos a G, y mantiene en secreto la solución de P en cada uno de estos grafos, $Sol_P(G_i)$.

Compromiso: $\mathcal{A}$ obtiene las imágenes $h(Sol_P(G_i))$ mediante una función hash pública h sobre cada una de las soluciones $Sol_P(G_i), i = 1, 2, ..., m$.

Reto: Se asume como retos los bits b_i correspondientes al menos significativo de cada una de las imágenes $h(Sol_P(G_i))$.

Respuesta: $\mathcal{A}$ publica:

- el isomorfismo entre $G \sim G_i$, si $b_i = 0$
- en otro caso, $Sol_P(G_i)$.

Verificación: Cualquier verificador que disponga de acceso a la información publicada por $\mathcal{A}$ comprueba además de la validez de los isomorfismos y soluciones públicas, las imágenes de dichos elementos obtenidas por la función h.

Figura 3.4: Algoritmo NIZKP-GP

Obsérvese que en el paso de configuración, la probadora $\mathcal{A}$ debe generar los m grafos isomorfos a G mediante simples permutaciones, y usar dichos isomorfismos para obtener a partir de la solución secreta S, las soluciones del problema P en cada uno de estos grafos.

En cuanto a las propiedades de la función hash a utilizar hay que mencionar que se requieren funciones hash resistente a las colisiones [Pre00] para que al repetir la ejecución del protocolo no se transfiera información perjudicial, y a la vez $\mathcal{A}$ no sea capaz de generar los compromisos adecuados a sus intereses en caso de querer realizar una estafa.

Puesto que el problema del isomorfismo es un problema abierto en la jerarquía de la complejidad computacional actual a continuación se proponen esquemas alternativos en los que la seguridad se basa en gran parte en el problema del isomorfismo. Concretamente en los próximos esquemas de ZKP descritos, la probadora $\mathcal{A}$ posee varios secretos (soluciones alternativas a un mismo problema en un mismo grafo) y pretende demostrar dicho conocimiento a $\mathcal{B}$ sin entregarle ninguna otra información adicional. Para ello en cada iteración la probadora genera un grafo isomorfo al de partida y transforma, usando dicho isomorfismo, uno de los secretos originales para obtener una solución en el grafo isomorfo. Después, en el turno del verificador, éste selecciona su reto de forma que escoge entre que $\mathcal{A}$ le muestre el isomorfismo o bien le muestre el compromiso determinado por la solución. Obsérvese que así $\mathcal{B}$ no recibe en ningún momento una solución completa. Además, en estos casos es especialmente importante que los compromisos no sean reutilizados, por lo que $\mathcal{A}$ debe seleccionar secretos diferentes en cada iteración, lo que hace que el número de iteraciones venga definido directamente por el número de soluciones diferentes conocidas por la probadora.

Para ilustrar el esquema interactivo descrito se proporcionan en las dos siguientes secciones la adaptación concreta a dos problemas clásicos de la Teoría de Grafos: el del circuito Hamiltoniano y el del conjunto independiente.

3.8. Algoritmo ZKP-HC

En el algoritmo de ZKP descrito a continuación sobre la base del problema del circuito hamiltoniano, se parte del hecho de que la probadora posee un número m de circuitos hamiltonianos de un grafo público G, y pretende demostrar dicho conocimiento a $\mathcal{B}$ con conocimiento nulo. En la realidad práctica, se supone que $\mathcal{A}$ genera el grafo a partir de los circuitos hamiltonianos secretos previamente seleccionados [HC97], [HC99a] de forma que todas las aristas estén presentes en algún circuito

hamiltoniano.

El objetivo de $\mathcal{A}$ es convencer a $\mathcal{B}$ de la posesión de esas soluciones secretas sin entregárselas ni proporcionarle ninguna información sobre ellas. Para ello realiza m iteraciones (desde $i = 1, 2, ..., m$) de los pasos mostrados en el algoritmo $ZKP - HC$ descrito en la Figura 3.5.

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ el grafo G.

Iteraciones: Se iteran los siguientes pasos desde $i = 1$ *hasta* m :

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ un grafo G_i isomorfo al de partida G y las n aristas del correspondiente circuito hamiltoniano isomorfo C_i comprometidas.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ un bit aleatorio b_i, y un vértice x de G_i si $b_i = 1$.

Respuesta: $\mathcal{A} \longrightarrow B$:

- si $b_i = 0$, el isomorfismo $G \sim G_i$.
- si $b_i = 1$, los dos vértices adyacentes a x que determinan las aristas incidentes en x pertenecientes a C_i.

Verificación: $\mathcal{B}$ comprueba:

- el isomorfismo, si $b_i = 0$.
- la adyacencia respecto de x de los dos vértices recibidos en el paso anterior y la corrección de los compromisos asociados a los vértices correspondientes, si $b_i = 1$.

Figura 3.5: Algoritmo ZKP-HC

Este esquema garantiza que en el caso de que $\mathcal{B}$ no supiera resolver el problema del isomorfismo obtendría sólo dos aristas de una solución en el grafo G_i, información insuficiente para reconstruir la solución completa de $\mathcal{A}$. Así, si el número de vértices es suficientemente grande y el grafo G es denso, realmente es difícil que $\mathcal{B}$ pueda reconstruir la solución, dada la complejidad del problema del circuito hamiltoniano.

3.9. Algoritmo ZKP-IS

En esta propuesta, al igual que en la anterior se pretende demostrar la posesión de múltiples secretos, siendo éstos en este caso diferentes conjuntos independientes de un grafo G. El protocolo comienza con la generación por parte de $\mathcal{A}$ de m conjuntos independientes de tamaño k y a partir de ellos genera el grafo de partida que debe ser regular y disperso. En la figura 3.6 se describen los pasos a seguir en el algoritmo $ZKP - IS$ propuesto, una vez que $\mathcal{A}$ ha publicado el grafo de partida G. Para este caso se fija el número de iteraciones m igual al número de conjuntos independientes conocidos por $\mathcal{A}$, [HC99b].

Configuración: $\mathcal{A} \longrightarrow \mathcal{B}$ el grafo G.

Iteraciones: Se iteran los siguientes pasos desde $i = 1$ *hasta* m :

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ un grafo G_i isomorfo al de partida (G) y los k vértices del correspondiente conjunto independiente isomorfo I_i comprometidos.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ un bit aleatorio b_i y selecciona los compromisos correspondientes a dos vértices x e y de I_i, si $b_i = 1$.

Respuesta: $\mathcal{A} \longrightarrow B$:

- si $b_i = 0$, el isomorfismo $G \sim G_i$.
- si $b_i = 1$, el contenido de los compromisos seleccionados por B, (x,y).

Verificación: $\mathcal{B}$ comprueba:

- el isomorfismo, si $b_i = 0$.
- la adyacencia entre x e y en el grafo G_i, si $b_i = 1$.

Figura 3.6: Algoritmo ZKP-IS

Si los parámetros son los adecuados, esto es, el número de vértices es grande, el grafo es disperso y la cardinalidad de los conjuntos independientes es suficiente,

la dificultad del problema del conjunto independiente garantiza la imposibilidad de reconstrucción de las soluciones por parte de $\mathcal{B}$.

Este esquema es fácilmente adaptable al problema del subgrafo maximal completo (clique). Basta con generar un grafo denso y modificar la comprobación a realizar por el verificador, pues en este caso los vértices que éste seleccione deben ser adyacentes en lugar de no adyacentes.

3.10. Algoritmo ZKP-BG

A continuación, descrito en la Figura 3.7, se plantea un esquema en el cual el papel de la probadora es más activo y no se limita únicamente a facilitar la información seleccionada por el verificador, ya que además ahora selecciona el reto a responder entre dos opciones propuestas por el verificador.

Este protocolo se puede interpretar como un juego bipartito entre los dos participantes $\mathcal{A}$ y $\mathcal{B}$.

Se debe tener en cuenta que al terminar el protocolo $\mathcal{A}$ debe haber usado cada grafo una única vez puesto que si $\mathcal{B}$ sabe resolver el problema del isomorfismo y los grafos son reutilizados, entonces puede reconstruir la solución del problema planteado en el grafo de partida.

El número de iteraciones coincide en principio con el número de soluciones conocidas por $\mathcal{A}$. Si $\mathcal{B}$ no considera suficiente este número de iteraciones, se pueden renovar los compromisos usando otros grafos isomorfos al de partida y comenzando de nuevo el protocolo. Sin embargo en el entorno de los protocolos criptográficos y de la complejidad de las comunicaciones en que se busca que el intercambio de mensajes sea mínimo, dicha reiniciación implica un incremento innecesario.

Configuración: $\mathcal{A}$ genera y publica un grafo G en el que conoce l soluciones de un determinado problema.

Compromiso: $\mathcal{A}$ genera y publica l grafos isomorfos al de partida $G_i, i = 1, 2, \ldots, l$. $\mathcal{A}$ compromete los elementos de cada una de las soluciones en cada uno de los grafos G_i.

Reto de $\mathcal{A}$: $\mathcal{B}$ selecciona aleatoriamente dos grafos del conjunto anterior (G_{i1}, G_{i2}) que no hayan sido utilizados como reto de $\mathcal{B}$ y un bit aleatorio b.

Reto de $\mathcal{B}$: $\mathcal{A}$ elige aleatoriamente uno de los dos grafos que $\mathcal{B}$ seleccionó $G_j, j \in \{i_1, i_2\}$.

Respuesta: $\mathcal{A} \longrightarrow B$:

- si $b = 0$, el isomorfismo entre el grafo de partida $G \sim G_j$.
- si $b = 1$, la fracción necesaria del compromiso construido por $\mathcal{A}$ en el grafo G_j para poder comprobar las propiedades de la solución.

Verificación: $\mathcal{B}$ verifica la información entregada.

Figura 3.7: Algoritmo ZKP-BG

3.11. Algoritmo ZKP-ISDL

A continuación se define una Demostración de Conocimiento Nulo de Conocimiento cuya seguridad descansa en la $NP - completitud$ del problema del Conjunto Independiente, [CH03b]. Este algoritmo está construido a partir de un compromiso de bits basado en la dificultad del problema del Logaritmo Discreto, lo cual garantiza el cumplimiento de las tres propiedades de completitud, solidez y conocimiento nulo computacional. Además, la propuesta es lo suficientemente versátil como para ser adaptada para ser usada con propósitos de identificación y/o en entornos no interactivos.

Tal y como se ha observado en las secciones anteriores y en la bibliografía existente,

la Teoría de Grafos se ha utilizado como amplia fuente de problemas difíciles tales como el isomorfismo, el no isomorfismo, la coloración, los circuitos hamiltonianos y los conjuntos independientes, [Blu86], [GMW86], [HC98], [SCGP99], [Gol99b],[CH01], para diseñar ZKP . Sin embargo es destacable que todas ellas están relacionadas de alguna manera con el problema del Isomorfismo de Grafos. Dado que dicho problema es un problema de complejidad computacional todavía por determinar y que además puede resultar fácil para muchos grafos aleatorios [For96b], resulta recomendable la propuesta de un nuevo protocolo alternativo que sea absolutamente independiente del problema del isomorfismo. Esta nueva propuesta utiliza como base los problemas del Conjunto Independiente y del Logaritmo Discreto, dejando al margen el problema del Isomorfismo de Grafos [CH03b].

El algoritmo utiliza el hecho de que siempre se puede generar alguna coloración válida en la que todos los vértices de cualquier conjunto independiente de cardinalidad k estén coloreados con el mismo color.

En la propuesta descrita a continuación la entrada de $\mathcal{A}$ está formada por un grafo $G = (V, E)$ y un entero k, y su objetivo es convencer a $\mathcal{B}$ de que conoce un conjunto independiente de tamaño k. En la etapa de configuración $\mathcal{A}$ genera un grafo aleatorio G con n vértices y un conjunto independiente I de tamaño k incluido en él a través del método descrito en [BC94], terminando esta etapa con la publicación de su entrada (G, k). Durante el desarrollo del algoritmo, $\mathcal{A}$ genera una c-coloración del grafo G en la que los k vértices de I poseen el mismo color. Entonces construye sus compromisos formados por c vectores binarios n-dimensionales $a_i = (a_i^j), a_i^j \in 0, 1, i = 1, 2, ..., c, j = 1, 2, ..., n$, donde las posiciones correspondientes al vértice j coloreado con el color i contienen un uno y el resto contienen un cero.

La cardinalidad de cada subconjunto de vértices definido por la c-coloración viene dada por el peso de Hamming del vector a_i, $W_H(a_i)$, siendo este un valor que juega un papel importante en el algoritmo. Tal y como se describe en la Figura 3.8 el algoritmo

$ZKP-ISDL$ contiene una fase de inicialización donde $\mathcal{A}$ y $\mathcal{B}$ acuerdan los parámetros m, c, los primos $p_i (i = 1, 2, ..., c)$ los generadores de $\mathbb{Z}^*_{p_i}$, g_i $(i = 1, 2, ..., c)$ y los enteros aleatorios de $\mathbb{Z}^*_{p_i}$, r_i $(i = 1, 2, ..., c)$. A continuación el algoritmo $ZKP-ISDL$ consiste en un número m de iteraciones previamente acordado entre las partes de una serie de pasos descritos en la figura 3.8.

Obsérvese que en los pasos de compromiso y verificación se puede utilizar el bien conocido algoritmo de exponenciación rápida para proporcionar mayor eficacia al proceso.

Teorema 3.11.1. *El algoritmo $ZKP-ISDL$ es una Demostración de Conocimiento Nulo.*

Demostración. La comprobación de que el algoritmo es completo y sólido es sencilla. Si $\mathcal{A}$ conoce un conjunto independiente en G de tamaño k, y ambos participantes toman parte en el protocolo de manera correcta, entonces el verificador acepta siempre. Esto es así puesto que si $b_l = 0$, entonces $\mathcal{B}$ comprueba que los dos vértices adyacentes seleccionados por él (v y w) están coloreados con dos colores diferentes. En caso contrario, si el bit seleccionado en la iteración l es uno, $\mathcal{B}$ comprueba la corrección de los vectores testigo y que al menos existe un conjunto independiente de cardinalidad k.

En lo que respecta a probar la solidez del esquema se debe recordar la definición de dicha propiedad: cuando la probadora $\mathcal{A}$ no conoce ningún conjunto independiente de tamaño k en G y el verificador $\mathcal{B}$ actúa de manera honesta, no importa cómo actúe $\mathcal{A}$, $\mathcal{B}$ debe rechazar la demostración con una alta probabilidad.

En este caso, $\mathcal{A}$ puede usar una c-coloración de G inválida usando un subconjunto de vértices de tamaño k cualquiera, y entonces si $b_l = 0$ $\mathcal{B}$ detectaría el fraude con una probabilidad de al menos $1/|E|$ para cada iteración.

Otra posibilidad para $\mathcal{A}$ consiste en calcular correctamente los vectores a_i, vectores que reflejan la coloración, sin usar ningún subconjunto de vértices con peso de Hamming k, por lo que $\mathcal{B}$ detectaría el fraude si el bit de reto seleccionado toma valor uno. Finalmente, $\mathcal{A}$ puede usar vectores testigo falsos empleando una coloración inválida con algún subconjunto de vértices de cardinalidad k. Esta estafa es detectada por $\mathcal{B}$ cuando el bit que envió es igual a uno. Todo esto hace que después de un número suficiente de iteraciones, la probabilidad de que la estafa cometida por $\mathcal{A}$ no sea detectada por su contrario sea insignificante.

Para demostrar la propiedad de conocimiento nulo computacional es necesario demostrar que la probadora $\mathcal{A}$ no transfiere conocimiento alguno independientemente del verificador con el que interacciona, incluyendo los que no se ajustan al comportamiento descrito en el protocolo.

A partir del testigo que recibe, $\mathcal{B}$ podría obtener la c-coloración comprometida y por tanto el conjunto independiente, sólo en el caso de que fuera capaz de resolver el problema del logaritmo discreto.

Por tanto, de acuerdo con el paradigma de la simulación, es posible construir un simulador probabilista polinomial que genera una distribución de probabilidad que es polinomialmente indistinguible de la distribución deducida durante la interacción entre $\mathcal{A}$ y $\mathcal{B}$.

Concretando, el simulador intenta en primer lugar adivinar el reto que $\mathcal{B}$ plantearía por lo que selecciona un bit b aleatoriamente. Entonces, si el valor supuesto es cero, el simulador genera unos vectores que contienen una coloración factible aleatoria y como consecuencia el proceso de verificación tendrá éxito si b coincide con el reto planteado. Si el bit generado es uno, el simulador construye unos vectores a_i que contienen una coloración falsa de forma tal que permiten que el proceso de verificación tenga éxito.

El simulador selecciona aleatoriamente uno de los dos posibles retos y si coincide

con el reto seleccionado por $\mathcal{B}$ (lo que sucede con una probabilidad de exactamente 1/2) entonces su salida es indistinguible de la salida de $\mathcal{A}$. En caso contrario el simulador se reinicia. De este modo, en tiempo esperado polinomial el simulador descrito puede reemplazar a $\mathcal{A}$, y por tanto el conocimiento nulo computacional queda demostrado. □

3.12. Esquema de Identificación Probabilista

En esta sección se presenta un esquema general interactivo de reto-respuesta basado en problemas de grafos difíciles que se obtiene al adaptar el algoritmo $ZKP-GP$ descrito en la figura 3.3 al propósito de la identificación. Posteriormente se describen algunas implementaciones específicas de este esquema para varios problemas $NP-completos$ donde las posibles soluciones vienen determinadas por un subconjunto de vértices [CH01]. Concretamente, se describen los procesos de construcción de instancias y el procedimiento de verificación para los problemas del recubrimiento de vértices, del conjunto independiente y del clique.

Al igual que sucede en el esquema de identificación determinista descrito en la sección 3.5, la información pública de cada usuario debe estar disponible en un directorio público certificado, siendo esta información un problema P y un grafo G. Además la construcción de la información secreta se realiza exactamente de la misma manera.

Se debe destacar que en el desarrollo del protocolo se maneja una información secreta fija durante todo el protocolo consistente en un entero positivo S perteneciente a un rango predefinido y otra información pública de un único uso determinada por un grafo G y un problema P. El esquema de identificación probabilista mostrado en la Figura 3.9 se divide en dos partes. La primera es una etapa de configuración off-line donde la probadora elige su número de identificación secreto S, y su información pública de un único uso: un problema P y un grafo G. La segunda parte corresponde

al procedimiento de conocimiento nulo formado por m iteraciones donde $\mathcal{A}$ se compromete con cierta información, $\mathcal{B}$ elige un reto, $\mathcal{A}$ responde y $\mathcal{B}$ verifica la respuesta y el compromiso, de forma que finalmente $\mathcal{B}$ acepta el número de identificación secreto de $\mathcal{A}$ como válido.

Obsérvese que la respuesta al reto $b_i = 0$ puede ser reemplazada por el descubrimiento total de la solución T cuando el conocimiento de todas sus características, tales como por ejemplo su cardinalidad, no ayuda a deducir ninguna información sobre la solución S.

Este esquema es probabilista, es decir, existe cierta probabilidad de que un probador fraudulento pase la verificación. Por tanto, se debe seleccionar un número de iteraciones m suficientemente grande como para garantizar un mínimo de seguridad, aunque por otro lado hay que tener en cuenta que este hecho incrementa el tráfico de mensajes. Por tanto, el número de iteraciones es un parámetro que debe ser acordado entre ambas partes $\mathcal{A}$ y $\mathcal{B}$ previamente.

Antes de proceder a la demostración formal de la validez del esquema descrito, es importante observar que se requiere la suposición de la siguiente hipótesis:

> "El conocimiento de pares diferentes de problemas difíciles y grafos correspondientes para los que existe una solución común no ayuda a resolver ninguno de los problemas de grafos."

En otras palabras,

> "Añadir restricciones a un problema difícil no lo hace más fácil"

En cualquier caso, en la práctica resulta recomendable cambiar el número secreto de identificación S correspondiente a la solución con la suficiente frecuencia como para prevenir posibles ataques heurísticos basados en el conocimiento acumulado de demasiados parámetros.

Teorema 3.12.1. *El Esquema de Identificación Probabilista descrito en la Figura 3.9 es una Demostración de Conocimiento Nulo.*

Demostración. Para comprobar la propiedad de completitud debe observarse que si $\mathcal{A}$ conoce una solución válida S de P en G, entonces ella es capaz de obtener una solución válida de P en cualquier copia isomorfa G generada por ella misma, por lo que no tiene ningún problema en mostrar alguna característica de esa solución o bien el isomorfismo entre ambos grafos.

La verificación de la solidez es como sigue. Si un probador falso $\mathcal{A}'$ envía a $\mathcal{B}$ un grafo H que no es isomorfo a G, entonces este probador no tiene manera alguna de mostrar más tarde un isomorfismo entre G y H ya que no existe tal isomorfismo. Así, en este caso, la probabilidad de convencer al verificador de que G y H son isomorfos es nula. Supongamos que $\mathcal{A}'$ envía una solución falsa T' de P en una copia isomorfa de G. Entonces, si $\mathcal{B}$ decide conocer el isomorfismo, $\mathcal{A}'$ sería capaz de mostrarlo, pero en otro caso, si selecciona comprobar una propiedad de T', como T' no es una solución válida, $\mathcal{B}$ rechaza la identificación con una probabilidad positiva p (la cual, en los casos particulares presentados a continuación, es mayor o igual que $1/|\mathrm{E}|$). Así, la probabilidad de que $\mathcal{B}$ detecte el fraude es mayor que $1/2$ tras un número suficiente de iteraciones.

La manera habitual de probar que el protocolo es de conocimiento nulo computacional es construyendo un simulador adecuado. El simulador propuesto selecciona en primer lugar un bit aleatorio y una permutación del conjunto de vértices de G. Si $b = 1$, entonces construye una copia isomorfa H de G y una solución falsa aleatoriamente generada. Si $b = 0$, entonces el simulador selecciona aleatoriamente una solución T' y genera, usando el procedimiento de construcción descrito en el primer capítulo, un grafo adecuado correspondiente a la identificación pública de $\mathcal{A}$, (P, G), manteniendo las características esenciales de G para las que la comprobación exacta es polinomial. Algunas de estas propiedades son por ejemplo los grados de los vértices,

el número de aristas, la conectividad o si el grafo es bipartido o no. Si la elección de $\mathcal{B}$ no coincide con el bit b generado por el simulador, entonces el simulador reinicia el proceso. Así, en tiempo esperado polinomial, el simulador es capaz de generar una salida que es computacionalmente indistinguible de la auténtica. □

Nótese que con este esquema la solución S no pierde su condición secreta, así que no es necesario cambiarla después de cada ejecución y no hay necesidad de compartir información secreta alguna. También obsérvese que el proceso de construcción puede implementarse de manera que para una solución fijada y para cada problema diferente, la generación de cada grafo aproveche generaciones previas, o incluso se implemente en paralelo.

Para los protocolos basados en un problema cuya solución es un subconjunto de vértices, el rango predefinido para seleccionar el entero secreto S es $[0, 2^n]$ porque existen 2^n soluciones posibles. En este caso, las posiciones de los bits que contengan la unidad en la representación binaria de S indican los vértices que pertenecen a la solución correspondiente.

El esquema general propuesto es también fácilmente aplicable [HCB02b] a aquellos problemas cuya solución consiste en determinar una partición del conjunto de vértices, como por ejemplo el conocido problema de la 3-coloración. En este caso, la codificación de la solución se realiza a través de un n-upla ternaria, o bien una t-upla binaria, donde $t = log_2 3^n$.

A continuación se describe en mayor detalle los procedimientos de construcción y verificación de implementaciones particulares del esquema descrito para los problemas de rescubrimiento de vértices, conjunto independiente y subgrafo maximal completo.

Recubrimiento de Vértices

El proceso de construcción de G en este caso puede llevarse a cabo fácilmente con cualquiera de los tres métodos descritos en el primer capítulo:

CM Añadiendo sólo aquellas aristas con al menos uno de los extremos en la solución.

DM Eliminando primero aquellas aristas sin ningún extremo en la solución y luego otras aristas seleccionadas aleatoriamente.

IM Generando primero un grafo aleatorio, y eliminando aquellas aristas tales que no tengan ningún vértice dentro del conjunto solución.

El procedimiento de comprobación de $\mathcal{B}$ en el caso $b_i = 0$ consiste en seleccionar aleatoriamente una arista (u, v) de H y asegurarse de que al menos uno de los extremos u ó v pertenecen a la solución. También debe verificar que el número de vértices en la solución T es menor o igual que k.

Nótese que la transformación de una solución de este problema en una solución del problema del conjunto independiente en el mismo grafo se puede realizar en tiempo polinomial. Por ello es necesario prevenir el posible uso fraudulento evitando la utilización del protocolo con ambos problemas en el mismo grafo.

Conjunto Independiente

La construcción del grafo G en este caso puede hacerse de las siguientes maneras:

CM Añadiendo aleatoriamente aristas que no unan ningún par de vértices del subconjunto solución.

DM Eliminando todas las aristas que unan vértices de la solución y algunas de las demás seleccionadas aleatoriamente.

IM Generando un grafo aleatorio y eliminando todas aquellas aristas que unan vértices del subconjunto solución.

$\mathcal{B}$ debe verificar que ninguna arista de H seleccionada aleatoriamente une dos vértices del subconjunto solución T, y que el número de vértices en este subconjunto es mayor o igual que k.

El protocolo no debe usarse con este problema y grafos bipartidos, grafos arista o cualquiera de las clases de grafos específicas para las que ha sido demostrado que es resoluble en tiempo polinomial [GJ79]. Así, en este caso la generación de los grafos debe ser realizada cuidadosamente. Igualmente debe evitarse el uso de grafos en los que la diferencia de los grados de los vértices pertenecientes a la solución y el resto sea elevada.

Clique

En este último caso el proceso de construcción de G se hace mediante alguno de los siguientes métodos:

CM Se añaden todas las aristas que unen cada par de vértices de la solución y algunas otras aristas seleccionadas aleatoriamente.

DM Se eliminan aristas aleatoriamente seleccionadas que no unan dos vértices del subconjunto solución.

IM Se genera aleatoriamente un grafo y se añaden las aristas necesarias hasta conseguir que todo par de vértices del conjunto solución sean adyacentes.

La definición de este problema es complementaria a la del problema del conjunto independiente. Por tanto, en el proceso de verificación correspondiente, $\mathcal{B}$ debe verificar que para cualquier arista de H seleccionada aleatoriamente, existen los extremos asociados dentro de la solución T, y que el número de vértices en la solución es igual o mayor que k.

Para algunos casos concretos de grafos tales como grafos planares o arista, este problema es resoluble en tiempo polinomial, por lo que estos casos deben eludirse para poder utilizar el protocolo con un mínimo de seguridad. Asimismo es aconsejable que el tamaño del grafo sea del orden de 2^k, siendo k el número de vértices que componen la solución [JP98].

3.13. Complejidad de los Algoritmos

Dos medidas estándares a la hora de comparar demostraciones de conocimiento nulo son la complejidad de las comunicaciones y la complejidad computacional.

Para medir la complejidad de las comunicaciones se usa el número de mensajes intercambiados necesario para conseguir una determinada probabilidad de éxito en la ZKP, de ahí la importancia del estudio de las demostraciones de conocimiento nulo con un número constante de iteraciones. Normalmente, para alcanzar este objetivo se manejan compromisos de bits o bien se relaja la propiedad de solidez, tal y como se mencionó en la primera sección de este capítulo.

También el número de bits intercambiados, ya mencionado en la descripción de cada uno de los protocolos propuestos es imprescindible a la hora de realizar comparaciones entre algoritmos. En cuanto a la complejidad computacional se estudia básicamente el número de pasos elementales.

Para dar el orden de complejidad del esquema de identificación determinista propuesto se necesita conocer cuáles son las características de las soluciones del problema base. De todas maneras la complejidad asociada a los procedimientos de $\mathcal{A}$ es la determinada por la construcción del grafo con la solución insertada puesto que la codificación de la solución como un número entero se desarrolla de manera eficiente. Si tomamos como referencia el problema del conjunto independiente de tamaño k, esta construcción se puede realizar en $O((n-k)^2)$.

En cuanto a las capacidades computacionales asociadas a $\mathcal{B}$, se deduce que son claramente polinomiales debido a que se limita a verificar la solución entregada por $\mathcal{A}$.

La clave pública de $\mathcal{A}$ (P, G) se publica en un directorio certificado para el que $\mathcal{B}$ posee derechos de lectura, por lo que esto no afecta a la complejidad de las comunicaciones. De hecho, la única información transferida es la contraseña, que es una

solución secreta asociada al par (P, G) por lo que depende de las características del problema. En el caso de usar un problema cuya solución consiste en dar un subconjunto de vértices, dicha transferencia es de $k \cdot \log n$ bits, siendo k la cardinalidad de la solución.

Es importante tener en cuenta que los protocolos de conocimiento nulo se describen generalmente como procesos iterativos, lo que afecta directamente al análisis de la complejidad, tal y como se puede observar a continuación.

Como ya sucedió con otras propuestas, el análisis explícito de los algoritmos $ZKP - GP$ y $NIZKP - GP$ requiere conocer cuáles son las características del problema seleccionado, por lo que a continuación sólo se presentan algunos datos orientativos.

Si se opta por usar grafos regulares que garanticen la dificultad del problema del isomorfismo, la generación del grafo de entrada asociada a $\mathcal{A}$ se desarrolla en orden $O(|E|)$. Posteriormente, y en cada una de las m iteraciones, $\mathcal{A}$ debe generar un grafo isomorfo al original, siendo por tanto la complejidad de los procesos asociados a $\mathcal{A}$ de orden $O(m \cdot |E|)$.

Para el caso concreto del problema del conjunto independiente se ha desarrollado la implementación (ver apéndice A) sin utilizar grafos regulares como entrada. En este caso, se garantiza la seguridad del esquema a través de la construcción de grafos en los que la selección de los grados de los vertices implica la dificultad del problema base. De esta manera aunque $\mathcal{B}$ sea capaz de deducir el isomorfismo, no puede obtener ninguna información sobre la solución secreta. Esta construcción alternativa se desarrolla en orden $O((n-k)^2)$, denotando k de nuevo el tamaño de la solución secreta.

En el caso de las acciones a realizar por $\mathcal{B}$, también dependen obviamente del problema base seleccionado. En cualquier caso están acotadas polinomialmente debido al uso de problemas $NP - completos$. Para el caso del conjunto independiente, la complejidad asociada a las mismas es de orden $O(m \cdot |E|)$.

Para la versión no interactiva de este protocolo, el análisis resulta inmediato puesto que sólo habría que añadir:

- al probador el costo del uso de la función hash para la definición de los retos, y
- al verificador la comprobación de la correspondencia de dicha definición y las respuestas publicadas.

La transferencia de información queda establecida por la expresión $2(m+1)|E|\log n + m + mn\log n$, siendo por tanto el orden de las comunicaciones $O(m|E|\log n)$.

El análisis de complejidad para el protocolo $ZHP-HC$ comienza con la generación del grafo con las m soluciones insertadas de manera que se el resultado sea un grafo regular que garantice la dificultad del problema del isomorfismo se desarrolla en orden $O(|E|)$.

Posteriormente, en cada una de las m iteraciones, $\mathcal{A}$ debe construir un grafo isomorfo (de orden $O(|E|)$), y definir un compromiso sobre las aristas de la solución asociada a la iteración actual. Debido a cuestiones de eficiencia se aconseja usar una función hash para la construcción de los compromisos. Por tanto, la complejidad asociada a $\mathcal{A}$ es $O(m\cdot|E|)$.

La complejidad de los procedimientos a realizar por $\mathcal{B}$ queda acotada también por la misma cantidad, aunque hay que destacar que si selecciona el reto $b_i = 1$, la complejidad de la iteración correspondiente se vería drásticamente reducida ya que la verificación de la adyacencia se desarrolla en tiempo constante y la verificación de los compromisos depende de la función hash seleccionada, siendo dicha comprobación en cualquier caso eficiente.

Suponiendo el uso de la función hash SHA-1, el número de bits intercambiados se puede acotar superiormente por la expresión , $2(m+1)|E|\cdot\log n + 160mn + m + mn\cdot\log n$.

El análisis del protocolo $ZKP - IS$ es totalmente análogo al análisis para el algoritmo $ZKP - HC$ por lo que no se describe en detalle. La principal diferencia es que esta vez el reto de B cuando selecciona el bit 1 queda definido por dos enteros, cuestión que puede incrementar de forma no significativa el ancho de banda necesario.

El análisis detallado del protocolo $ZKP-BG$ depende directamente del problema y compromiso de bits seleccionado. Sin embargo, para problemas concretos se pueden desarrollar extensiones basadas en los dos protocolos anteriores $ZKP-HC$ y $ZKP-IS$.

En el caso del algoritmo $ZKP - ISDL$ es conveniente destacar que ambos participantes necesitan sólo capacidades computacionales acotadas polinomialmente. Concretamente en este caso la complejidad asociada a $\mathcal{A}$ es de orden $O(n^3)$, y la asociada a $\mathcal{B}$ es de orden $O(c \log^3 l)$, siendo $l = \max\limits_{i=1,2,\ldots,c} \{p_i\}$.

La complejidad de las comunicaciones viene dada por la expresión $(n+c+2)\log l + 2|E|\log n$, siendo por tanto de orden $O(m \cdot c \cdot \log n)$.

En el caso del algoritmo $ZKP - ISDL$ es conveniente destacar que ambos participantes necesitan sólo capacidades computacionales acotadas polinomialmente. Concretamente en este caso la complejidad asociada a $\mathcal{A}$ es de orden $O(n^3)$, y la asociada a $\mathcal{B}$ es de orden $O(c \log^3 l)$, siendo $l = \max\limits_{i=1,2,\ldots,c} \{p_i\}$.

La complejidad de las comunicaciones viene dada por la expresión $(n+c+2)\log l + 2|E|\log n$, siendo por tanto de orden $O(m \cdot c \cdot \log n)$.

La complejidad de los procedimientos a realizar por ambos usuarios en el esquema de identificación probabilista descrito en la memoria dependen de cuál es el problema seleccionado, y en el caso de $\mathcal{A}$, de cuál es el esquema de cifrado escogido. Suponiendo que la generación del grafo inicial se desarrolla en orden $O(n^2)$, la complejidad de cada iteración es para $\mathcal{A}$ de orden $O(|E|)$. Por tanto, la complejidad asociada a $\mathcal{A}$ es de orden $O(m \cdot |E|)$.

El usuario $\mathcal{B}$ tiene dos posibles procedimientos asociados en cada iteración:

- comprobar el isomorfismo (de orden $O(|E|)$), o
- comprobar una solución de un problema $NP-completo$ (de tiempo polinomial).

Además en cualquiera de los dos casos debe comprobar que los compromisos han sido correctamente definidos. Analizando todas estas cuestiones se concluye que la complejidad de los procedimientos asociados a $\mathcal{B}$ son de orden $O(m \cdot |E|)$.

Configuración: $\mathcal{A}$ realiza las siguientes operaciones:

- Genera y publica un grafo aleatorio G de n vértices conteniendo un conjunto independiente, I de cardinalidad k.
- Construye los vectores $a_i = (a_i^1, a_i^2, ..., a_i^n), i = 1, 2, ..., c$.
- Selecciona aleatoriamente $y_j \in \mathbb{Z}_{p-1}^*,\ j = 1, 2, \ldots, n$, siendo $p = \min\limits_{i=1,2,...,c} \{p_i\}$.

Inicialización: $\mathcal{A}$ y $\mathcal{B}$ acuerdan los siguientes parámetros:

- El número de iteraciones m y el número de colores c.
- Los primos p_i.
- Los generadores g_i.
- Los enteros aleatorios r_i, $(i = 1, 2, ..., c)$.

Iteraciones: Para $l = 1$ hasta m:

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ los vectores $v_i = ((r_i^{a_i^j} \cdot g_i^{y_j}) \mod p_i)$, $(i = 1, 2, \ldots, c,\ j = 1, 2, \ldots, n)$.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ un bit b_l seleccionado aleatoriamente y si $b_l = 0$ dos vértices adyacentes en G, (v, w) seleccionados también aleatoriamente.

Respuesta: $\mathcal{A} \longrightarrow B$:

- si $b_l = 0$, los enteros y_v y y_w
- si $b_l = 1$, $y = \sum\limits_{j=1}^{n} y_j$ y $W_H(a_i) = \sum\limits_{j=1}^{n} a_i^j,\ i = 1, 2, \ldots, c$.

Verificación: $\mathcal{B}$ comprueba si los valores proporcionados por $\mathcal{A}$ en los pasos previos son correctos, es decir, verifica

- si $b_l = 0$, que $\exists s, t \in \{1, 2, \ldots, c\} \,|s \neq t$, $a_s^v = a_t^w = 1$.
- si $b_l = 1$, que
  - $\sum\limits_{i=1}^{c} W_H(a_i) = n$,
  - $\exists i \in \{1, 2, ..., c\} \,|W_H(a_i) = k,\ y$
  - $\forall i \in \{1, 2, ..., c\}$: $(\prod\limits_{j=1}^{n} r_i^{a_i^j} \cdot g_i^{y_j}) \mod p_i = (r_i^{W_H(a_i)} \cdot g_i^{y}) \mod p_i$.

Figura 3.8: Algoritmo ZKP-ISDL

Configuración: $\mathcal{A}$ selecciona un problema P y una solución apropiada cuya codificación entera es S, y construye un grafo adecuado G en el que S es una solución de P. $\mathcal{A}$ publica el par (P, G).

Iteraciones: Los siguientes cuatro pasos se ejecutan m veces:

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ el grafo H isomorfo a G, junto con el cifrado del isomorfismo $f : G \sim H$ y de la solución $T = f(S)$ del problema P en el grafo H.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ un bit aleatorio b_i, entendiéndose que con él solicita a $\mathcal{A}$ que descifre:

1. una característica de la solución, si $b_i = 0$.
2. el isomorfismo f, si $b_i = 1$.

Respuesta: $\mathcal{A} \longrightarrow \mathcal{B}$ la información solicitada.

Verificación: $\mathcal{B}$ verifica el correspondiente cifrado y comprueba que:

1. T cumple las propiedades de una solución válida del problema P en el grafo H, si $b_i = 0$.
2. f transforma el grafo G en H, si $b_i = 1$.

Figura 3.9: Esquema de identificación probabilista

Capítulo 4

Protocolos Basados en Otras Herramientas Combinatorias

En los capítulos anteriores se ha mostrado la valía de la Teoría de Grafos como herramienta práctica en el diseño de protocolos criptográficos. En éste se extiende el marco, utilizando otras herramientas combinatorias con la misma finalidad.

Lo que se persigue cuando se usa un problema determinado como base de un criptosistema es que al fijar una de sus instancias, encontrar una solución de la misma sea impracticable, mientras que generar pares formados por $(instancia, solución)$ pueda realizarse de manera eficiente. Otra propiedad que debe verificar el problema seleccionado consiste en que el procedimiento de verificación para cualquier solución sea lo más sencillo posible. Las tres razones previas justifican que el uso de problemas pertenecientes a las clases NP y $NP - completa$ estén tan extendidas en el área del diseño de criptosistemas.

Sin embargo, con el desarrollo de la Teoría de la Complejidad Computacional, y concretamente gracias a los avances hechos en el análisis del Caso Medio, se ha demostrado que algunos problemas $NP - completos$ pueden ser eficientemente resueltos cuando se genera la instancia correspondiente según ciertas distribuciones [Kar76]. Una de las soluciones más inmediatas es elegir como problemas base aquéllos

cuya dificultad está garantizada por el análisis del caso medio. El problema seleccionado para los protocolos descritos a continuación es el *Problema Distribucional de la Representación de Matrices* [Wan97], que posee esta característica.

Seguidamente se describe la estructura de este capítulo. La siguiente sección se ha reservado para introducir algunos conceptos directamente relacionados con la definición formal del problema usado como base, a la vez que se incluyen los resultados computacionales relacionados con su clasificación según el análisis del caso medio y se comenta el contexto en el que surgen las propuestas presentadas. Posteriormente se encuentran los dos apartados principales en que se ha dividido el capítulo, dedicados cada uno a un esquema representativo de los protocolos bipartitos y multipartitos respectivamente. En la primera de estas secciones se describe la primera propuesta, correspondiente a una demostración de conocimiento nulo definida para ser usada como esquema de identificación, mientras que en el ámbito de los protocolos multipartitos se incluye un esquema de reparto de secretos siendo su principal aplicación la distribución de claves. Se cierra cada una de las dos secciones con sendos análisis de la complejidad de cada una de las propuestas.

4.1. Problema Distribucional de la Representación de Matrices

Tal y como ya se mencionó en el primer Capítulo, la clase distribucional análoga a la clase NP en la jerarquía asociada al análisis del caso medio es la clase $DistNP$. Dicha clase está formada por pares que contienen un problema de decisión de la clase NP, y una distribución de probabilidad computable en tiempo polinomial, $DistNP =< NP, P - computable >$.

La principal dificultad a solventar cuando se usan problemas de esta clase en aplicaciones prácticas es lo artificioso de sus formulaciones, situación que no se manifiesta en el caso del problema distribucional de la representación de matrices. Es

ésta la principal razón por la que se ha optado por dicho problema como base de los criptosistemas propuestos.

Los elementos componentes de todas las matrices que intervienen en el problema son enteros y generados según la distribución uniforme.

A continuación se describe el problema mencionado en su versión más general.

Definición 4.1.1. Problema Distribucional de la Representación de Matrices (DMR, Distributional Matrix Representability)
Dada una matriz Z y una colección de k matrices distintas y con la misma dimensión $M = \{M_1, M_2, ..., M_k\}$, debe decidirse si Z puede ser expresada como producto de matrices pertenecientes al conjunto dado.

En este trabajo además se usa la versión acotada cuya definición aparece en [VR92] y que a continuación se incluye.

Definición 4.1.2. Problema DMR según [VR92]
Las instancias de este problema están formadas por una matriz Z, un conjunto de k matrices distintas $M = \{M_1, M_2, ..., M_k\}$ y un entero positivo n. Todas las matrices que intervienen en este problema son de dimensión 20×20. La pregunta a responder es si es posible expresar Z como un producto perteneciente a M^n, conjunto formado por todos los productos de n matrices de M, siendo $n \leq k$. La distribución considerada para generar k y n se entiende que es la distribución uniforme.

Los esquemas propuestos a continuación hacen uso de la versión de búsqueda del problema distribucional anterior cuya dificultad es equivalente a la del problema de decisión de partida, cuestión que se deduce del resultado general establecido en [BDCGL92], (ver sección 1.2.3) de acuerdo con el cual, las versiones de búsqueda y de decisión de los problemas distribucionales son equivalentes desde el punto de vista del análisis del caso medio.

4.2. Demostración de Conocimiento Nulo

Las ya mencionadas anteriormente primeras demostraciones de conocimiento nulo usadas como esquema de identificación (Fiat-Shamir [FS86] y sus variantes [GQ88], [OO89],[OS90], y Schnorr [Sch89b]) están basadas en dos problemas de la Teoría de Números bien conocidos como bases de los criptosistemas más habituales, que son el de la factorización de enteros grandes y el del logaritmo discreto. De ahí que una de las mayores desventajas de dichos protocolos se debe a la no demostración de la dificultad de los problemas base, así como al importante costo computacional requerido por las operaciones aritméticas necesarias.

Puesto que la definición de conocimiento nulo no requiere exactamente funciones trampa (que son tan importantes en clave pública), y en lugar de ellas, se utilizan funciones unidireccionales (que representan un requisito menos restrictivo), el camino hacia otras técnicas que no usan problemas de la Teoría de Números está abierto tal y como ha quedado de manifiesto en el capítulo anterior [CHB03b].

El algoritmo incluido a continuación surge al plantear la posibilidad de diseñar esquemas de identificación de conocimiento nulo basados en problemas difíciles en media. En él se incluye un nuevo protocolo bipartito de identificación cuya seguridad recae sobre un problema clasificado como $DistNP-Completo$ bajo el análisis del caso medio. Dicho problema es el problema distribucional de representación de matrices ya descrito en el apartado anterior.

4.2.1. Algoritmo ZKP-DMR

El propósito del esquema propuesto a continuación [CH03a] es permitir la identificación de usuarios en un sistema controlado por una autoridad central de confianza. Dicho esquema utiliza k matrices $M_i, i = 1, 2, ..., k$ fijas y públicas de dimensión $r \times r$ invertibles cuyos elementos son enteros. Dichas matrices son comunes a todos los

usuarios y originalmente son generadas aleatoriamente por la autoridad.

Se pueden considerar dos variantes del esquema correspondientes a las dos versiones del problema distribucional de representación de matrices, de forma que cada participante $\mathcal{A}$ selecciona el número secreto n de matrices para calcular el producto correspondiente a su identificación, o bien se considera la versión acotada, y el número fijo n es aleatoriamente seleccionado por la autoridad y publicado como parte del sistema de identificación.

Como paso previo a la identificación, cada usuaria $\mathcal{A}$ selecciona aleatoriamente n matrices $\{M_{i_1}, M_{i_2},...,M_{i_n}\}$ del conjunto público, calcula su producto $\prod_{j=1}^{n} M_{i_j} = Z_A$, y lo comunica a la autoridad. Esta matriz debe hacerse pública en un directorio público relacionándola con la identidad de la usuaria $\mathcal{A}$, o bien certificarse por medio de la firma digital de la autoridad. Así, cuando una usuaria $\mathcal{A}$ quiere identificarse ante otro usuario $\mathcal{B}$, el primer paso consiste en remitirle su matriz de identificación pública para comenzar una sesión de identificación. Una vez verificada dicha identificación pública, bien a través del directorio público o por medio de la firma de la autoridad, el protocolo interactivo de identificación puede tener lugar.

Como ya se adelantó en el Capítulo 3, los esquemas de identificación de conocimiento nulo recaen en la importante noción de compromiso de bits. Para ello se usa un proceso de dos etapas con una etapa de compromiso y otra de apertura. En la etapa de compromiso, la usuaria $\mathcal{A}$ envía a $\mathcal{B}$ el testigo del secreto comprometido. Posteriormente, durante la fase de respuesta, la usuaria $\mathcal{A}$ revela cierta información que permite a $\mathcal{B}$ la verificación del testigo entregado.

Como es normal en los diseños de conocimiento nulo el algoritmo propuesto se compone de varias iteraciones independientes de una subrutina básica. El primer paso de dicha subrutina consiste en la generación de un vector de enteros aleatorios v con el mismo tamaño que las matrices que intervienen en el protocolo. El vector generado y su traspuesto v^T se usan para generar $2k$ vectores testigo de tamaño r,

$\{vM_i, (M_i v^T)^T\}_{i=1,...,k}$. Por tanto, la propuesta de identificación interactiva incluye m iteraciones de la subrutina mostrada en la figura 4.1.

Configuración: $\mathcal{A}$ genera aleatoriamente un vector secreto de tamaño r, v, y un entero $x \in [2, 2^n]$.

Compromiso: $\mathcal{A} \longrightarrow \mathcal{B}$ en orden aleatorio los $2k$ vectores de tamaño r, $\{vM_i, (M_i v^T)^T\}_{i=1,...,k}$ y el entero $vZ_A{}^x v^T$.

Reto: $\mathcal{B} \longrightarrow \mathcal{A}$ un bit aleatorio e_i.

Respuesta: $\mathcal{A} \longrightarrow \mathcal{B}$:

- si $e_i = 0$, el vector v y el entero x.
- si $b_i = 1$, los vectores vM_{i_1} y $M_{i_n} v^T$, y la matriz de dimensión $r \times r$, $Z_{A_1} = M_{i_1}{}^{-1} Z_A{}^x M_{i_n}{}^{-1}$.

Verificación: Dependiendo del reto seleccionado, $\mathcal{B}$ comprueba que:

a. si $e_i = 0$, la información es correcta.

b. si $e_i = 1$, se obtiene $vZ_A{}^x v^T$ del producto entre el vector fila vM_{i_1}, la matriz Z_{A_1} y el vector columna $M_{i_n} v^T$. También, $\mathcal{B}$ usa recursivamente los siguientes pasos con $j = 2, ..., t$ para verificar que $\mathcal{A}$ conoce los factores de la matriz Z_A:

b1. $\mathcal{A} \longrightarrow \mathcal{B}$ el testigo $vZ_{A_{j-1}}{}^x v^T$

b2. $A \longrightarrow \mathcal{B}$ los dos vectores vM_{i_j}, $M_{i_{n-j+1}} v^T$, y la matriz de dimensión $Z_{A_j} = M_{i_j}{}^{-1} Z_{A_{j-1}}{}^x M_{i_{n-j+1}}{}^{-1}$de dimensión $r \times r$.

b3. $\mathcal{B}$ comprueba que del producto entre el vector fila vM_{i_j}, la matriz Z_{A_j} y el vector columna $M_{i_{n-j+1}} v^T$ se obtiene el entero testigo $vZ_{A_{j-1}}{}^x v^T$.

Figura 4.1: Algoritmo ZKP-DMR

En la primera variante original del problema, dado que $\mathcal{B}$ no conoce el número n, el número de iteraciones recursivas t debe acordarse previamente por los participantes teniendo en cuenta sus intereses diferentes. En la segunda variante ese número debe

ser $t = n/2$ para así poder verificar el conocimiento de los n factores de la matriz original.

Como en cualquier esquema de identificación de conocimiento nulo, en éste también existe cierta probabilidad de que un probador fraudulento pase la fase de verificación. En este caso, esta probabilidad depende fuertemente del número de iteraciones m puesto que la probabilidad de error está acotada por 2^{-m} (y también del número de iteraciones recursivas t en la primera variante).

La elección aleatoria de vectores v permite detectar los fraudes de un probador que no genere adecuadamente los testigos comprometidos. El algoritmo de Monte Carlo descrito por Freivalds [Fre79] (también usado en el siguiente algoritmo propuesto) puede usarse para la verificación del producto de dos matrices consiguiendo de esta manera un proceso de detección de estafa. Además, todos los productos de matrices requeridos en el protocolo se pueden desarrollar usando el algoritmo propuesto en [CW87], aprovechando así su eficiencia.

La seguridad de este esquema está garantizada por la complejidad del problema distribucional de la representación de matrices puesto que para que todas las verificaciones sean correctas, $\mathcal{A}$ debe conocer las n matrices $M_{i_j}, j = 1, 2, ..., n$ cuyo producto es la matriz de identificación pública. Además, puesto que el problema base del esquema es $NP - completo$ según el análisis del caso medio, una elección adecuada de los parámetros k y n, y la generación realmente aleatoria de las matrices, garantizan que resolver el problema está más allá de los límites de la tecnología actual independientemente de la generación aleatoria de las instancias.

En lo referente a la elección de los parámetros k y n, se debe comentar que para evitar ataques por búsqueda exhaustiva a lo largo de todas las $\binom{k}{n}$ posibles combinaciones de n matrices del conjunto público, el entero k debe tener un valor aproximadamente igual el doble de n, y ambos valores deben ser lo suficientemente grandes, lo que implica que $k \geq 130$ teniendo en cuenta los recursos computacionales

disponibles actualmente.

Por otra parte, en la primera variante el tamaño de la matriz r es uno de los parámetros de seguridad, pero en este caso resulta difícil recomendar un tamaño mínimo para él porque se ha demostrado que el problema asociado es indecidible incluso para valores pequeños (Markov así lo demostró en [Mar58] para $r = 4$). De todas maneras, se debe asumir que habitualmente valores altos proporcionan un mejor nivel de seguridad.

Teorema 4.2.1. *El algoritmo $ZKP - DMR$ es una Demostración de Conocimiento Nulo.*

Demostración. Respecto a la propiedad de completitud, se muestra a continuación que $\mathcal{B}$ siempre acepta la demostración de $\mathcal{A}$. Si $\mathcal{B}$ selecciona el reto e=0, entonces $\mathcal{A}$ le entrega el vector v y el entero x, con lo que $\mathcal{B}$ puede generar los productos $\{vM_i, (M_i v^T)^T\}_{i=1,...,k}$ y el entero $vZ_A{}^x v^T$ y puede comparar lo obtenido con los testigos entregados por $\mathcal{A}$ en la fase de compromiso. En el caso de que $\mathcal{B}$ seleccione el reto $e = 1$, recibe el entero $vZ_{A_{j-1}}{}^x v^T, j = 1, ..., t$, los vectores vM_{i_j} y $M_{i_{n-j+1}} v^T$, y la matriz $Z_{A_j} = M_{i_j}{}^{-1} Z_{A_{j-1}}{}^x M_{i_{n-j+1}}{}^{-1}$ de dimensión $r \times r$, de manera que cuando $\mathcal{B}$ multiplica el vector fila vM_{i_j}, la matriz Z_{A_j} y el vector columna $M_{i_{n-j+1}} v^T$, obtiene el entero $vZ_{A_{j-1}}{}^x v^T$ que le fue enviado por $\mathcal{A}$ en la etapa de compromiso.

Para examinar la seguridad con respecto al punto de vista del verificador (es decir la propiedad de solidez) se debe contemplar que un probador deshonesto puede intentar adivinar el reto planteado por el verificador con antelación, seleccionando una respuesta adecuada. Si la pregunta realizada por el verificador e, entonces dicho probador deshonesto pasa con éxito la verificación. Sin embargo, la probabilidad de este suceso es $1/2$ para una iteración, pasando a ser 2^{-m} para el protocolo completo. Para poder responder todas las preguntas posibles del verificador, un impostor debe ser capaz de generar testigos que satisfagan la etapa de compromiso y verificación

a la vez. Sin embargo, encontrar esta combinación es tan difícil como resolver la instancia del problema Distribucional de la Representación de Matrices construida por el probador legítimo.

La propiedad de conocimiento nulo se demuestra a continuación usando el concepto de simulador del tipo caja negra descrito en el capítulo anterior. La construcción del simulador se realiza comenzando por el final del protocolo, tal y como a continuación se describe. En primer lugar, el adversario supone el reto del verificador y el simulador construye la información requerida para pasar el proceso de verificación de acuerdo con esta suposición. Esto significa que en el caso de que el reto simulado sea $e = 0$, se construyen los testigos tal y como se especifica en el algoritmo. Por otro lado, si $e = 1$, el simulador debe construir testigos falsos a partir de matrices artificialmente construidas que verifiquen que el producto de las mismas coincida con Z_A, o bien a partir de matrices del conjunto original tales que su producto no coincida con la citada matriz Z_A. En ambos casos el simulador puede usar su conocimiento sobre todas las matrices que se corresponden con los factores del producto para que la verificación se desarrolle con éxito. Obsérvese que el hecho de que los retos sean seleccionados por $\mathcal{B}$ de manera aleatoria e independiente en cada iteración del protocolo es de particular importancia ya que en otro caso $\mathcal{A}$ podría aprovecharlo para una posible estafa. □

4.2.2. Complejidad del Algoritmo

El esquema propuesto hace uso de capacidades de computación y almacenamiento limitadas para ambos participantes, además de una complejidad de las comunicaciones razonable.

A primera vista puede parecer que el esquema definido en esta sección requiere una gran cantidad de memoria y tiempo de computación, pero esto no es exacto si se acota superiormente el valor que los elementos de las matrices pueden tomar,

ya que las operaciones a desarrollar son muy simples y se pueden implementar en hardware de manera eficiente. Concretamente, la carga computacional de $\mathcal{A}$ consiste en el cálculo del producto de matrices, producto que puede calcularse de manera eficiente a través del algoritmo propuesto en [CW87]. De todas formas, puesto que en algunos casos es posible que se requiera que los cálculos de $\mathcal{A}$ se desarrollen en un dispositivo portátil con capacidad computacional limitada, la elección del parámetro r debe ser estudiada si se opta por la primera variante ya que la obtención de los productos anteriores puede resultar costosa para valores grandes de r. Sin embargo, se debe tener en cuenta que el cálculo de los productos sucesivos para el reto $e = 1$ se puede realizar aprovechando los resultados parciales previos obtenidos.

En cuanto al espacio de memoria requerido, en cada iteración recursiva $\mathcal{A}$ necesita únicamente almacenar resultados intermedios, de manera que los requerimientos totales al respecto son limitados.

Con la intención de acotar la complejidad de las comunicaciones, se puede proponer una versión no interactiva basada en el compromiso de los retos. En este caso, todas las elecciones aleatorias de ambos participantes se pueden realizar al comienzo para posteriormente enviarlas a través de un protocolo de Intercambio de Secretos (ver sección 2.3), o simplemente usar una función hash para construir los compromisos. En ambos casos, la única interacción necesaria durante la identificación es la respuesta de $\mathcal{A}$ a cada reto planteado previamente por $\mathcal{B}$.

Por último, se debe destacar que una ventaja práctica del esquema descrito es la posibilidad de añadir fácilmente nuevos usuarios al esquema de identificación.

4.3. Reparto de Secretos

Al igual que sucede en otros protocolos interactivos el principal problema con el que se tropieza el diseñador de un esquema de Reparto de Secretos consiste en encontrar una manera segura y eficiente de determinar y repartir las fracciones de

información secreta a los usuarios correspondientes. Esto se consigue utilizando de manera adecuada la denominada técnica de corte y elección.

Definición 4.3.1. Reparto de Secretos (SS, Secret Sharing)
Un esquema de Reparto de Secretos consiste en dividir un secreto en fracciones de manera que una vez seleccionada una de las fracciones ésta no vierta al entorno información alguna sobre el secreto.

Los protocolos multipartitos de SS resuelven situaciones prácticas usuales en las que es necesaria la distribución de un secreto particular S entre un conjunto de participantes P. Este contexto puede ser ilustrado a través del problema de protección de claves secretas. Por tanto, el principal objetivo de los esquemas de Reparto de Secretos consiste en garantizar que sólo un subconjunto de participantes previamente designados son capaces de reconstruir el secreto combinando colectivamente su información (o sombras) sobre S. La especificación de todos los subconjuntos de participantes autorizados a reconstruir el secreto se denomina estructura de acceso del esquema SS. Esta estructura se dice monótona si cualquier subconjunto que contenga a su vez un subconjunto capaz de reconstruir el secreto puede usarse también para recuperar el secreto. Por otra parte, los esquemas de reparto de secretos que no revelan información alguna sobre el secreto compartido a los individuos no autorizados se denominan perfectos.

4.3.1. Estado del Arte

Los primeros SS fueron propuestos en 1979 de manera independiente por Shamir y Blakley [Sha79], [Bla79]. Posteriormente se demostró que ambas propuestas pueden ser consideradas como casos particulares de un esquema más general debido a que se basan en los mismos principios de álgebra lineal, [Kot85]. El número de estructuras matemáticas diferentes usadas tanto para modelar como para plantear nuevas

SS alternativos es considerable. Algunas de estas estructuras son polinomios [Sha79], configuraciones geométricas [Bla79], diseños en bloque, códigos de Reed-Solomon, espacios vectoriales, matroides, grafos bipartitos completos, arrays ortogonales y cuadrados latinos, [Sti87]. Una metodología para diseñar esquemas de SS con estructuras de acceso monótonas arbitrarias se describe en [ISN87] y [BL89]. No obstante, estos resultados no son aplicables para el esquema propuesto aquí porque la estructura de acceso no es monótona. En [Sim89] se encuentra un extenso recorrido por el campo de los protocolos de reparto de secretos centrándose sobre todo en esquemas planteados sobre problemas y herramientas geométricas.

Aunque también se han estudiado los esquemas de reparto de secretos en los que no participa ninguna TTP [IS90], un modelo común en SS consiste en dividir el protocolo en dos fases. En la fase de inicialización, una TTP llamada negociador, distribuye las sombras del secreto a los participantes autorizados a través de un canal seguro. En la fase de reconstrucción, los participantes autorizados pertenecientes a un subconjunto de la estructura de acceso combinan sus sombras para reconstruir el secreto. Un esquema basado en matrices, se describe en [KH83], pero allí el secreto es una solución de un sistema de ecuaciones.

4.3.2. Algoritmo SS-DMR

El esquema descrito a continuación [HC03b] hace uso de la generación de una matriz secreta como producto de otras con la misma dimensión. Las sombras se seleccionan entre un conjunto de matrices previamente determinado.

Puesto que se trata de un protocolo multipartito en esta ocasión se tiene un conjunto de participantes denotado por $P = \{P_1, P_2, \ldots, P_n\}$.

En la inicialización del SS propuesto a continuación el negociador publica todas las sombras y la única información secreta que se revela a cada participante por separado es un puntero a una sombra concreta junto con las identidades del resto de

participantes en la misma estructura de acceso. Por tanto, el SS aquí propuesto es perfecto y su seguridad es incondicional.

Es posible establecer dos variantes del protocolo dependiendo de si el arbitraje de una TTP es necesario o no. Si se decide su intervención esta entidad estará a cargo no sólo de la generación del secreto sino que además reconstruirá el secreto, manteniéndolo protegido del resto de participantes. En cualquier caso el usuario encargado de generar el secreto es denotado por $\mathcal{T}$.

Para facilitar la descripción general del protocolo éste se ha dividido en cuatro etapas: configuración, distribución, verificación y, finalmente recuperación. Una primera e informal descripción del algoritmo se incluye a continuación.

El principal parámetro de este tipo de protocolos es el número de participantes indispensables para recuperar el secreto, conocido como cardinalidad del conjunto privilegiado de la estructura de acceso. En el esquema actual dicho número se corresponde con n y debe ser lo suficientemente grande como para evitar que un ataque por búsqueda exhaustiva tenga éxito. En este protocolo el espacio de búsqueda para este tipo de ataques está formado por todos los productos posibles formados con n matrices del conjunto M. Por tanto, la cardinalidad del espacio de búsqueda es $\begin{pmatrix} k+n-1 \\ n \end{pmatrix}$. Otra característica de este protocolo es la publicación del conjunto M en un directorio indexado con permisos exclusivamente de lectura donde cada participante autorizado tiene acceso a su sombra correspondiente.

La etapa de distribución requiere la existencia de un canal de comunicaciones seguro, o bien el uso de algún esquema de cifrado. El problema principal en esta etapa es el ancho de banda necesario para transferir las sombras. Sin embargo, dicha dificultad se evita en este esquema enviando a cada participante autorizado el índice que fue asignado a la matriz correspondiente en la estructura de acceso.

En cuanto a la etapa de verificación se debe destacar que permite detectar la presencia de participantes deshonestos entre los poseedores de sombras y garantiza la

corrección del secreto reconstruido. Para conseguir este procedimiento de verificación se usa el algoritmo de Monte Carlo descrito por Freivalds [Fre79] para la verificación del producto de dos matrices. La probabilidad de error en este algoritmo está acotada por 2^{-t}, donde t es el número de iteraciones a desarrollar. Además, todas los productos de matrices necesarios se obtienen usando el algoritmo propuesto en [CW87] debido a su eficiencia.

El primer paso de la etapa de verificación es la generación de un vector binario U de la misma dimensión que las matrices que intervienen en el protocolo. Dicha generación puede ser realizada de dos maneras, la puede realizar la TTP que posee el secreto o bien puede ser obtenida a través de un generador público. El vector U generado se multiplica por la matriz secreta Z obteniéndose un nuevo vector U' que contiene una combinación lineal aleatoria de las filas de la matriz secreta. Tanto este vector U' como el vector aleatorio U se ubican en el directorio público previamente mencionado.

En la generación del vector binario U se deben descartar aquellos vectores con peso de Hamming 1, ya que de no hacerlo el vector U' coincidiría con una columna de la matriz secreta Z. Por tanto, la cardinalidad del conjunto de vectores binarios posibles es $2^{20} - 20$.

Una vez tenido esto en cuenta, se selecciona una permutación aleatoria del conjunto de participantes $\{P_{(1)}, P_{(2)}, \ldots, P_{(n)}\}$, que establece el orden en que se desarrolla el proceso de verificación. El participante designado en primer lugar ($P_{(1)}$) calcula de forma privada el producto de su sombra por el vector aleatorio binario obteniendo $M_{(1)}U$, y envía el resultado al siguiente participante ($P_{(2)}$). De esta manera, $P_{(2)}$ calcula el producto determinado por su sombra y el vector proporcionado por $P_{(1)}$, $(M_{(2)}M_{(1)}U)$, y así sucesivamente. Únicamente si todos los participantes son honestos, $P_{(n)}$ obtiene U', y se lo comunica a los otros participantes.

De acuerdo con el proceso anterior, si algunos de los participantes falsea su sombra

es detectado con probabilidad estrictamente superior a 1/2. Si se desea un mejor nivel de seguridad, entonces el proceso de verificación completo puede repetirse un número suficiente de veces.

Por último en la etapa de reconstrucción cada participante debe acceder al directorio en el que el conjunto M está disponible para obtener su sombra, después de lo cual y dependiendo de la existencia de una TTP, hay dos acciones posibles:

- Si se considera la intervención de la TTP, cada participante le envía su sombra, y entonces la TTP se encarga de reconstruir y comunicar el secreto.

- Si se omite la TTP, entonces el primer participante según la permutación revela el producto $M_{(1)}U$ al siguiente usuario, quien a su vez multiplica su sombra por la información transferida y así sucesivamente (teniendo en cuenta que los productos intermedios deben enviarse usando siempre medios seguros). Esta forma de proceder permite el reparto equitativo del costo computacional entre todos los participantes.

En la figura 4.2 se incluye una descripción formal del protocolo.

Una ventaja del algoritmo $SS - DMR$ es que el secreto no es revelado a los participantes en ningún momento, por lo que permite la reutilización tanto del secreto como de las sombras.

Como se mencionó anteriormente, la seguridad incondicional de los esquemas perfectos es un concepto extensivamente utilizado en SS. Un SS se considera incondicionalmente seguro frente a participantes deshonestos si la probabilidad de realizar una estafa con éxito no depende de las habilidades y/o capacidades computacionales de los estafadores. En este sentido, y gracias al procedimiento de verificación descrito, el SS incluido en este apartado se considera incondicionalmente seguro.

4.3.3. Complejidad del Algoritmo

Al ser un protocolo multipartito, en el análisis de complejidad se distingue entre el negociador $\mathcal{T}$ y el resto de participantes P_i.

De esta manera, la generación de las k matrices originales, la generación del vector U, el producto de n cualesquiera de dichas matrices, y el producto del vector U por la matriz secreta son las acciones a realizar por $\mathcal{T}$. Debido a que la dimensión de las matrices y de los vectores está prefijada, los parámetros con los que se debe trabajar son k y el número de participantes n. En función de ambos parámetros los cálculos a desarrollar por $\mathcal{T}$ implican un tiempo lineal.

El número de operaciones elementales asociadas a cada participante P_i, y el costo de las mismas es constante puesto que si se realizan los pasos de verificación y recuperación del secreto sin la intervención de una TTP, lo único que debe hacer cada P_i es calcular el producto de una matriz por un vector.

En cuanto a la complejidad de las comunicaciones, ésta depende del rango en el que se escogen los enteros que forman las matrices, y de la participación de una TTP, siendo menor en el caso de que no participe puesto que el protocolo se ahorra el envío de la sombra de cada participante a la TTP.

Configuración:

$\mathcal{T}$ genera los siguiente elementos:

- los enteros n, k
- $M = \{M_1, M_2, \ldots, M_k\}, M_i \in \mathbb{Z}^{20} \times \mathbb{Z}^{20}$.
- $A = \prod_{j=1}^{n} M_j$
- $\mathcal{T}$ publica M.

Distribución: $\mathcal{T} \longrightarrow P_i$ el índice correspondiente a $M_i, i = 1, 2, \ldots, n$.

Verificación:

- $\mathcal{T}$ genera aleatoriamente $U \in \mathbb{Z}_2^{20}$.
- $\mathcal{T}$ calcula $U' = U \cdot A$.
- $\mathcal{T}$ publica U' y U.
- Se permuta aleatoriamente el conjunto P, obteniéndose $\{P_{(1)}, P_{(2)}, \ldots, P_{(n)}\}$
- $P_{(1)} \longrightarrow P_{(2)} : M_{(1)} \cdot U$.
- $P_{(j)} \longrightarrow P_{(j+1)} : M_{(j)} \cdot [M_{(j-1)} \cdots M_{(2)} \cdot M_{(1)} \cdot U], j = 2, 3, \ldots, n-1$.
- $P_{(n)}$ comunica el resultado de $M_{(n)}[M_{(n-1)} \cdots M_{(2)} \cdot M_{(1)} \cdot U]$.

Recuperación: Dependiendo de si interviene una TTP o no:

- Si participa una TTP:
  - $P_{(i)} \longrightarrow TTP : M_i i = 1, 2, \ldots, n$.
  - TTP calcula $\prod_{i=1}^{n} M_i$.
- Si no participa ninguna TTP:
  - $P_{(1)} \longrightarrow P_{(2)} : M_{(1)}$.
  - $P_{(j)} \longrightarrow P_{(j+1)} : M_{(j)}[M_{(j-1)} \cdots M_{(1)}], j = 2, \ldots, n-1$.
  - $P_{(n)}$ comunica el resultado de $M_{(n)}[M_{(n-1)} \cdots M_{(1)}]$.

Figura 4.2: Algoritmo SS-DMR

Capítulo 5

Conclusiones

A lo largo del presente trabajo se ha profundizado en el estudio del diseño de protocolos criptográficos. El punto de partida de la investigación realizada se puede describir de la siguiente manera.

La mayoría de herramientas criptográficas existentes (incluidos los protocolos) basan su seguridad en herramientas propias de la Teoría de Números, siendo además frecuente el uso exclusivo de problemas tales como el de la factorización, la residuosidad cuadrática o el logaritmo discreto. Dada la inconveniencia que supone tal dependencia, uno de los objetivos primordiales de este trabajo es precisamente proponer la Teoría de Grafos como nueva fuente de inspiración y prontuario de problemas base, dada su gran utilidad y versatilidad. De esta forma, se amplía de forma importante el conjunto de herramientas disponibles para el diseño de protocolos criptográficos.

Por otra parte, el criterio habitual con que se mide la seguridad práctica de los esquemas criptográficos modernos se basa en la Teoría de la Complejidad Computacional, y más concretamente en el análisis del caso peor. Dicho análisis dificulta la garantía de seguridad criptográfica ya que se ha demostrado que problemas difíciles para el caso peor son habitualmente fácilmente resolubles cuando las instancias se generan aleatoriamente. Por tanto, en la presente memoria se propone como alternativa la utilización de la complejidad del caso medio para garantizar de forma eficaz

la seguridad práctica de los algoritmos.

Por último, resulta destacable que la abundante bibliografía existente en el área de los protocolos criptográficos se dedique principalmente a la propuesta independiente de algoritmos para cada tipo de protocolo, sin una visión o metodología general de diseño, ni investigaciones de las interrelaciones entre distintos protocolos. Por ello, una de las tareas abordadas ha sido el diseño de esquemas generales para cada protocolo, así como el análisis de la relaciones entre distintos protocolos.

A continuación se enumeran las principales aportaciones obtenidas en este trabajo:

Se han definido esquemas generales de diseño para los dos principales protocolos primitivos de transferencia inconsciente y compromiso de bits.

Se han descrito dos nuevos esquemas algorítmicos aplicables a cualquier problema difícil de grafos para los protocolos de transferencia inconsciente y compromiso de bits.

Se han introducido algoritmos basados en el problema del isomorfismo de grafos tanto para ambas primitivas, como para los protocolos de firma de contratos y de lanzamiento de monedas.

Se han analizado las variantes más conocidas de transferencia inconsciente, la OT uno de dos, la OT uno seleccionado de dos, y la venta de secretos, proponiéndose en los tres casos algoritmos basados en grafos.

Se ha presentado un esquema de identificación determinista basado en contraseñas de un sólo uso, criptografía de clave pública y grafos.

Se ha propuesto un nuevo esquema general de diseño de demostraciones de conocimiento nulo.

Se han definido sendos esquemas generales de demostraciones de conocimiento nulo interactiva y no interactiva, basadas en cualquier problema difícil de grafos.

Se han descrito dos algoritmos de demostraciones de conocimiento nulo basadas en los problemas del circuito hamiltoniano y el conjunto independiente.

Se ha presentado una demostración de conocimiento nulo como un juego bipartito.

Se han combinado herramientas de la Teoría de Grafos (problema del conjunto independiente) y de la Teoría de Números (problema del logaritmo discreto) para proponer una demostración de conocimiento nulo.

Se ha introducido un esquema general de identificación probabilista basado en problemas difíciles de grafos.

Se ha propuesto una demostración de conocimiento nulo basada en un problema clasificado como difícil según el análisis del caso medio, el llamado Problema Distribucional de la Representación de Matrices.

Se ha descrito un algoritmo mutipartito de reparto de secretos basado en el Problema Distribucional de la Representación de Matrices.

Para cada uno de los algoritmos propuestos se ha realizado un análisis teórico de su especificación formal, así como un estudio práctico de los parámetros de complejidad asociados.

Se han implementado programas de generación aleatoria de grafos y de demostraciones de conocimiento nulo generales basadas en problemas cuyas soluciones son un subconjunto de vértices o un subconjunto de aristas.

Entre las siguientes cuestiones que han quedado abiertas en la presente memoria, algunas están siendo objeto actual de investigaciones, mientras que otras lo serán en el futuro.

Implementación eficiente de todas las propuestas realizadas y análisis computacional de sus ejecuciones.

Estudio comparativo entre cada uno de los algoritmos propuestos con los análogos existentes en la bibliografía.

Profundización en el análisis de las interrelaciones y dependencias entre protocolos bipartitos y multipartitos.

Análisis de la aplicabilidad práctica de las transferencias inconscientes y compromisos de bits como herramientas primitivas de diseño de protocolos multipartitos concretos.

Relajación las hipótesis introducidas en la definición de varios de los algoritmos propuestos.

Extensión del rango de aplicación de los algoritmos bipartitos diseñados a partir de problemas difíciles de grafos, al conjunto de los protocolos multipartitos.

Estudio de construcciones concretas de instancias de los problemas usados, que sean adecuados para el diseño de los esquema propuestos.

Desarrollo de otros protocolos bipartitos, como la transferencia inconsciente, el compromiso de bits, la firma de contratos y el lanzamiento de monedas, y de otros protocolos multipartitos, como el reparto jerarquizado de secretos o la distribución anónima de mensajes, con base en el Problema Distribucional de la Representación de Matrices.

Propuesta de nuevos protocolos basados en otros problemas combinatorios NP-completos según el análisis del caso medio, como el Problema Distribucional de la Coloración de Aristas.

Apéndice A

Implementaciones

En este apartado se incluyen las implementaciones desarrolladas en C

```
#ifndef GENBIN

#define GENBIN

#include <stdio.h>

/* genbin.h: Se establece el maximo numero
   de vertices del grafo aleatorio */

#define MAXVERT  1000

#define MAXVERTdiv8 125

#define BOOL     char

int vert, aristas; BOOL Bitmap[MAXVERT][MAXVERTdiv8];

char masks[ 8 ] =
{ 0x01, 0x02, 0x04, 0x08, 0x10, 0x20, 0x40, 0x80 };

#define MAXDESCRIP 10000 static char descrip[MAXDESCRIP];

#define MAXFICH 255 static char name[MAXFICH];

char *grafo; int grafocreado; int almacen[MAXVERT];
```

```
/* almacen: contiene el conjunto independiente secreto */

#endif

/*=========================================================*/

/* ConfGen: Se establece el tipo de generador y
su configuracion inicial */


#include <stdlib.h>

#include <time.h>



#ifdef LINUX

/* Establece el estado inicial del generador */
static long estado[32] = {
 3,
 0x9a319039, 0x32d9c024, 0x9b663182, 0x5da1f342,
 0x7449e56b, 0xbeb1dbb0, 0xab5c5918, 0x946554fd,
 0x8c2e680f, 0xeb3d799f, 0xb11ee0b7, 0x2d436b86,
 0xda672e2a, 0x1588ca88, 0xe369735d, 0x904f35f7,
 0xd7158fd6, 0x6fa6f051, 0x616e6b96, 0xac94efdc,
 0xde3b81e0, 0xdf0a6fb5, 0xf103bc02, 0x48f340fb,
 0x36413f93, 0xc622c298, 0xf5a42ab8, 0x8a88d77b,
 0xf5ad9d0e, 0x8999220b, 0x27fb47b9
}; #endif

/* Periodo de las secuencias generadas:
    16*[(2^31)-1] RedHat 7.2 */
void conf_generador(void);
void est_semilla( int semilla_ini);
int generador(void);
void GenGrafIsIndep(int indep,
                    char *file,
                    int Iorig[MAXVERT],
```

```
                    int IndepIso[MAXVERT],
                    int isomorfismo[MAXVERT],
                        char *solIso);
    void inicializo(BOOL G[MAXVERT][MAXVERT]);
    void write_graph_ascii(char *file);
    int parametros(int *vert, int *aristas);
void write_graph_bin(char *file);

void GuardarCjto (int in, int Cjto[MAXVERT],
                  char *fich , BOOL tipo);

/*=========================================================*/

void inicializo(BOOL g[MAXVERT][MAXVERT]) {

    int i, j;
    for (i = 0; i < vert; i++)
    {
        for (j = 0; j < vert; j++)
            g[i][j]= FALSE;

    }
}

/*=========================================================*/

void conf_generador(void) {
  unsigned semilla;
  int n;

#ifdef LINUX
  semilla = 1;
  n = 128;
  initstate(semilla, (char *) estado, n);
  setstate(estado);
#else

#endif } /* conf_generador */

/*=========================================================*/
```

```
void est_semilla( semilla_ini)
int semilla_ini;
{
#ifdef LINUX
  setstate(estado);
  srandom(semilla_ini);
#else
  srand(semilla_ini);
#endif } /* est_semilla */

/*=========================================================*/

int generador() { #ifdef LINUX
  return((int)(random()));
#else
  return(rand());
#endif } /* Final de la configuracion del generador */

/*=========================================================*/

/* Genera el grafo y el conjunto independiente isomorfos */

void GenGrafIsIndep(int indep,
                    char *binario,
                    int Iorig[MAXVERT],
                    int IndepIso[MAXVERT],
                    int isomorfismo[MAXVERT],
                    char *solIso)
{
    int temp, r, i, j, b,t;
    int c, oc;
    char * pp = descrip;
    FILE * fp, *f;
    char nombre[MAXFICH];
    int v,a;
    /* inicializacion */

    for(i = 0; i < vert ; i++)
        isomorfismo[i] = i+1;
```

```
/*est_semilla( );*/
printf("\nA genera el isomorfismo");
for (i = vert - 1; i >= 0; i--)
{
    r = 1+(int) (((float) i)*generador()/(RAND_MAX+1.0));
    temp = isomorfismo[i];
    isomorfismo[i] = isomorfismo[r];
    isomorfismo[r] = temp;

}

printf("\nA comienza a construir el grafo "
       "isomorfo al original");
printf("\nFichero binario que contiene el grafo "
       "original %s", binario);
write_graph_ascii( binario );

if ( (fp=fopen(binario,"r"))==NULL )
{
    printf("\nError en la apertura del fichero "
           "%s",binario);
  exit(10);
}
strcat(nombre, name);
strcat(nombre, ".Giso");
if( (f = fopen(nombre,"w")) == NULL)
{
    printf("\nError en la apertura del fichero "
           "%s",nombre);
    exit(10);
}


for(oc = '\0'
    ;(c = fgetc(fp)) != EOF && (oc != '\n' || c != 'e')
    ; oc = *pp++ = c);
    fprintf(f,"%s",descrip);

ungetc(c, fp);
*pp = '\0';
```

```
parametros(&v, &a);
printf("\nEl fichero en el que A guarda el grafo "
       "isomorfo es %s\n",nombre);

while ((c = fgetc(fp)) != EOF){
    switch (c)
      {
        case 'e':
          if (!fscanf(fp, " %d %d", &i, &j))
          {
                printf("Fichero de entrada  %s no se ajusta "
              "al formato\n", binario);
                exit(10);
          }
          fprintf(f, "e %d %d\n", isomorfismo[i-1],
                  isomorfismo[j-1]) ;


         case '\n':


         default:
                break;
           }/* switch */

    }/* while */
    fclose(fp);
    fclose(f);

strcat(nombre ,".b");
write_graph_bin(nombre);
printf("\nEl nombre del fichero con el grafo "
        "ISOMORFO generado por A "
        "en formato binario es %s",nombre);

printf("\nComienza la óconstruccin del conjunto "
        "indep en el grafo isomorfo");
for(i = 0; i < indep; i++){
     a = Iorig[i];
     b = isomorfismo[a];
```

```
            IndepIso[i] = b;
        }
        strcat(solIso, "SolIsomorfa.dat");
        GuardarCjto(indep,IndepIso,solIso, FALSE);
        printf("\n Se termino la construccion del grafo isomorfo"
               " y del conjunto indep. isomorfo por parte de A");
}

/*=========================================================*/

/*
   transformar: Convierte la representacion binaria de la
   instancia a ASCII

 */

#include "genbin.h"

#include <string.h>

#include <stdlib.h>

BOOL get_edge(int i, int j);
void write_graph_ascii(char *file);
void read_graph_bin(char*file);
int get_params(void);
void set_edge(register int i, register int j,char x);
void write_graph_bin(char *file);
void read_graph_ascii(char *file);

/*=========================================================*/

BOOL get_edge(i, j)
     int i;
     int j;
{
    int byte, bit;
    char mask;

    bit  = 7-(j & 0x00000007);
```

```
    byte = j >> 3;

    mask = masks[bit];
    return( (Bitmap[i][byte] & mask)==mask );
}

/*=================================================*/

void write_graph_ascii(file)
     char *file;
{
    int i,j;
    FILE *fp;

    if ( (fp=fopen(file,"w"))==NULL )
      {
       printf("␣Error␣en␣la␣apertura␣del␣fichero\n");
       exit(10);
       }

    fprintf(fp, descrip);

    for ( i = 0; i< vert; i++ )
      {
          for ( j=0; j<=i; j++ )
            if ( get_edge(i,j) ) fprintf(fp,"e␣%d␣%d\n",i+1,j+1 );
      }

    fclose(fp);
} /*=============================================================*/

void read_graph_bin(file)
     char *file;
{

    int i, length = 0;
    FILE *fp;

    if ( (fp=fopen(file,"r"))==NULL )
    {
```

```
        printf("\nError en la apertura de fichero %s",file);
        exit(10);
    }

    if (!fscanf(fp, "%d\n", &length))
      {
        printf("\nEl fichero %s no se ajusta al formato",file);
        exit(10);
      }

    if(length >= MAXDESCRIP)
      {
        printf("El fichero %s no se ajusta al formato.\n",file);
        exit(10);
      }

    fread(descrip, 1, length, fp);
    descrip[length] = '\0';

    if (!get_params())
    {
    printf("\nEl fichero %s no se ajusta al formato",file);
      exit(10);
    }

    for ( i = 0
     ; i < vert &&
     fread(Bitmap[i], 1, (int)((i + 8)/8), fp)
     ; i++ );

    fclose(fp);
}

/*=======================================================*/

int get_params(void) {
    char c, *tmp;
    char * pp = descrip;
    int stop = 0;
    tmp = (char *)calloc(100, sizeof(char));
```

```
    vert = aristas = 0;

    while (!stop && (c = *pp++) != '\0'){
        switch (c)
          {
            case 'c':
              while ((c = *pp++) != '\n' && c != '\0');
              break;

            case 'p':
              sscanf(pp, "%s %d %d\n", tmp, &vert,
                     &aristas);
              stop = 1;
              break;

            default:
              break;
          }
    }

    free(tmp);

    if (vert == 0 || aristas == 0)
      return 0;  /* error */
    else
      return 1;

}

/*=========================================================*/

void set_edge(i, j, x)
     register int i;
     register int j;
     char x;
{
    register int byte, bit, mask;

    bit  = 7 - (j & 0x00000007);
```

```
    byte = j >> 3;

    mask = masks[bit];
    if ( x == 1 ) Bitmap[i][byte] |= mask;
    else  Bitmap[i][byte] &= ~mask;
}

/*=========================================================*/

void write_graph_bin(file)
     char *file;
{

    int i;
    FILE *fp;

    if ( (fp=fopen(file,"w"))==NULL )
    {
        printf("\nError: No se encuentra el fichero"
               "%s", file);
        exit(10);
    }

    fprintf(fp, "%d\n", strlen(descrip));
    fprintf(fp, descrip);

    for ( i = 0
        ; i < vert &&
        fwrite(Bitmap[i], 1,(int)((i + 8)/8), fp)
        ; i++ );

    fclose(fp);
}

void read_graph_ascii(file)
     char *file;
{

    int c, oc;
    char * pp = descrip;
```

```
int i,j;
FILE *fp;

if ( (fp=fopen(file,"r"))==NULL )
{
    printf("\nError: No se encuentra el fichero %s", file);
    exit(10);
}

for(oc = '\0';(c = fgetc(fp)) != EOF &&
    (oc != '\n' || c != 'e')
    ; oc = *pp++ = c);

ungetc(c, fp);
*pp = '\0';
get_params();

while ((c = fgetc(fp)) != EOF){
    switch (c)
      {
        case 'e':
          if (!fscanf(fp, "%d %d", &i, &j))
          {
                printf("\nError: el fichero %s no "
                       "se ajusta a formato",file);
                exit(10);
          }

          if (i > j)
            set_edge(i-1, j-1, 1);
          else
            set_edge(j-1, i-1, 1);
          break;

        case '\n':

        default:
          break;
      }
}
```

```
    fclose(fp);
}

/*=========================================================*/

/*
   AlgIS.h: Almacena en la matriz de adyacencia el grafo
   aleatorio almacenado en el fichero ASCII
 */

#define TRUE 1

#define FALSE 0

#include <stdio.h>

#include <stdlib.h>

#include <math.h>

#include <string.h>

#include "genbin.h"

#include "ConfGen.h"


int creargrafo(int vert,int n_indep,float prob12,
               float prob22);

void GuardarCjto (int in, int Cjto[MAXVERT],
                  char *fich , BOOL tipo);

void GuardarIns (BOOL GIso[MAXVERT][MAXVERT],
                 char *fichGIso);

BOOL **crearMatriz(void);
```

```
void destruirMatriz(BOOL **m);

int *crearVector(void);

void destruirVector(int *m);

void InicializarGrafo(BOOL g[MAXVERT][MAXVERT]);

int parametros(int *vert, int *aristas);

void leerInst (BOOL G[MAXVERT][MAXVERT], int *vert,
               int *aristas, char *nombre);

void mostrarIns( BOOL Gr[MAXVERT][MAXVERT] );

void MostrarCjto(int ind, int InIso[MAXVERT]);

double func1(double x,double p,int n,int s,int i);

int ocultar(int vert,int n_indep,float f_prob, int def);

int borrargrafo(void);


/*=========================================================*/

void GuardarIns (BOOL GIso[MAXVERT][MAXVERT], char *fichGIso)

/*  Procedimiento que guarda un grafo en un fichero ASCII */
{
    int i;
    int j;

   FILE *fi;

   fi = fopen(fichGIso,"w");
    if (fi == NULL)
     {
        printf("Error al crear fichero %s: \n",fichGIso);
        exit(-1);
```

```
    } /* if */

 /*Mostrar la descripcion del fichero generado */

     fprintf(fi,"p aristas %d %d\n",vert,aristas);
    for ( i = 0; i< vert; i++ )
      {
          for ( j=0; j<i; j++ )
            if ( GIso[i][j] ) fprintf(fi,"e %d %d\n",i+1,j+1 );

      }


    fclose(fi);
}

/*=========================================================*/

void GuardarCjto (int in, int Cjto[MAXVERT],char *fich,
                  BOOL tipo)

/* Procedimiento que vuelca un conjunto de vertices de tamano
    in en un fichero ASCII
*/ {
    int i;

    FILE *sol;



     if (tipo == FALSE)
     {

      strcat(fich,".sol");
     }
     else
     {
        printf("\nfichero isomorfismo %s", fich);
     }
    sol = fopen(fich,"w");
```

```
    if (sol == NULL) {
        fprintf(stderr,"Error al crear fichero "
                "%s: \n", fich);
        exit(-1);
    } /* if */

 /*Mostrar la descripcion del fichero generado */

    for (i = 0; i < in; i++){
        fprintf(sol, "%d ",Cjto[i]);
     }
    fclose(sol);
}

/*=========================================================*/

BOOL **crearMatriz(void) {
    BOOL **m;
    int i;
    printf("\nprimera llamada");

    m = (BOOL **) malloc (vert * sizeof(BOOL *));
    if(m == NULL) printf("\nNo hay memoria para m");
    for(i = 0; i < vert; i++)
    {
        m[i] = (BOOL *) malloc (vert * sizeof(BOOL));
        if(m == NULL)
        {
            printf("\nNo hay memoria suficiente");
            exit(-3);
        }

    }
    return (m);
}

/*=========================================================*/

void destruirMatriz(BOOL **m) {
    int i;
```

```
    for(i = 0; i < vert; i++)
    {
        free(m);

    }

}

/*=========================================================*/

 int *crearVector(void) {
    int *array;

    array = (int *) malloc(vert*sizeof(int));
    if(array == NULL)
    {
        printf("\nNo hay memoria suficiente");
        exit(-3);
    }
    return (array);
}

void destruirVector(int *m) {

    free(m);

}

/*=========================================================*/
void InicializarGrafo(BOOL g[MAXVERT][MAXVERT])
{
    int i, j;
    for (i = 0; i < vert; i++)
    {
        for (j = 0; j < vert; j++)
            g[i][j]= FALSE;

    }
}
```

```
/*=============================================================*/

int parametros(int *vert, int *aristas)

/* Lee el numero de vertices y de aristas.
 formato de linea: p ??? vertices aristas
*/

{
    char c, *tmp;
    char * pp = descrip;
    int stop = 0;
    tmp = (char *)calloc(100, sizeof(char));

    *vert = 0 ; *aristas = 0;

    while (!stop && (c = *pp++) != '\0'){
        switch (c)
          {
            case 'c':
              while ((c = *pp++) != '\n' && c != '\0');
              break;

            case 'p':
              sscanf(pp, "%s %d %d\n", tmp, vert, aristas);
              stop = 1;
              break;

            default:
              break;
          }
    }

    free(tmp);

    if (*vert == 0 || *aristas == 0)
      return 0;  /* error */
    else
      return 1;
```

```
}

/*=========================================================*/

void leerInst (BOOL g[MAXVERT][MAXVERT], int *v, int *a,
                    char *nombre)
{

    int i = 0, j = 0, k = 0;

    int c, oc;
    char * pp = descrip;
    FILE * fp;

    if ( (fp=fopen(nombre,"r"))==NULL )
      { printf("Error en la apertura del fichero\n");
        exit(10);
       }

    for(oc = '\0'
        ;(c = fgetc(fp)) != EOF && (oc != '\n' || c != 'e')
        ; oc = *pp++ = c);

    ungetc(c, fp);
    *pp = '\0';
    parametros(v, a);

    while ((c = fgetc(fp)) != EOF){
        switch (c)
          {
            case 'e':
              if (!fscanf(fp, "%L%d", &i, &j))
                { printf("Fichero de entrada no se ajusta "
                  "al formato\n");
                    exit(10);
                }

                        if ( j < *v )
                        {
```

```
                        g[i][j] = TRUE;
                        g[j][i] = TRUE;
                        j++;
                        k++;
                    }       /* if */
                    else
                    {
                        i++;
                        j = 0;
                        g[i][j] = TRUE;
                        g[j][i] = TRUE;
                        k++;
                    }       /* else */

                case '\n':

                default:
                    break;
            }/* switch */

        }/* while */
        fclose(fp);
}

/*=================================================================*/

void mostrarIns( BOOL Gr[MAXVERT][MAXVERT] ) {

    int i, j;


    for (i = 0; i < vert; i++)
    {
        for (j = 0; j < vert; j++)
            printf("%d\t", Gr[i][j]);
        printf("\n");
    }
} /*=================================================================*/

void MostrarCjto(int ind, int InIso[MAXVERT]) {
```

```
    int i;

        for (i = 0; i < ind; i++)
            printf("%d ",InIso[i]);

}

/*=========================================================*/
int creargrafo(vert,n_indep,prob12,prob22)
int vert;

/* numero de vertices del conjunto independiente */

int n_indep;

/* probabilidad para la generacion de grafos
   segun el modelo G_{vert,prob12} */

float prob12;

/* parametro de seguridad para la ocultacion
   del conjunto independiente */

float prob22; {
  int cw, cy,cz,cx;
  int iprob12,iprob22;
  int n_fil;
  int aristas = 0;
  int t[MAXVERT];
  float eta,total;
  int delta;
  BOOL vertype[MAXVERT];
  int asize;
  char file[MAXFICH];

  asize = ((vert>>3)+1) * vert;
  grafo = (char *)malloc(asize);
  if (grafo == NULL) {
    fprintf(stderr,"\n Reserva de memoria fallida \n");
    exit(-1);
```

```
  }
  bzero(grafo,asize);



  for(cx=0;cx<vert;cx++) {
    vertype[cx] = FALSE;
  }

/* seleccion de los vertices del conjunto independiente */

  delta = 0.0;
  eta = (2.0*delta)/((float)(n_indep-1));
  total = 0.0;
  cy = 0;
  while (cy<n_indep) {
    cx = generador() & MAXVERT;
    if ((cx<vert) && (vertype[cx] == FALSE)) {
      almacen[cy] = cx;

      total += prob12*((1.0-delta)+(eta*cy));

      t[cy] =
      (int)((total*1048576.0/((float)(n_indep*prob12)))+0.5);

      cy++;
      vertype[cx] = TRUE;
    }
  }
  almacen[cy] = -1;
  t[cy-1] = 1048576;/* 2^ 20 */

  /* volcado del conjunto independiente en un fichero */

  strcat(file,name);
  GuardarCjto (n_indep, almacen,file, FALSE);
  strcat(file,".sol");

/* Inclusion de aristas en el grafo */
```

```
  n_fil = (vert>>3)+1;
  iprob12 = (int)(prob12*1048576.0+0.5);
  iprob22 = (int)(prob22*1048576.0+0.5);

  total = 0.0;

/* inclusion de aristas entre vertices de V-I */

  for(cx=0;cx<vert;cx++) {
   if (vertype[cx] == FALSE) {
    for(cy=0;cy<cx;cy++) {

      cz = generador() & 1048575;

      if (vertype[cy] == FALSE) {
       if(cz<iprob22) {
        grafo[(cx)*n_fil+(cy>>3)] |= (1<<(cy&7));
        grafo[(cy)*n_fil+(cx>>3)] |= (1<<(cx&7));
        aristas++;
         }
      }
    }

/* Inclusion de aristas entre vertices de V y de V-I */

    total += (n_indep*prob12);
    while (total >= 1.0) {
      cz = generador() & 1048575;
      cw = 0;
      while (t[cw] < cz) {cw++;}
      cy = almacen[cw];
      if ((grafo[(cx)*n_fil+(cy>>3)] & (1<<(cy&7))) == 0)
      {
        grafo[(cx)*n_fil+(cy>>3)] |= (1<<(cy&7));
        grafo[(cy)*n_fil+(cx>>3)] |= (1<<(cx&7));
        total -= 1.0;
          aristas++;
      } /* if */
    } /* while */
```

```
    }
   }

   grafocreado = TRUE;
   return(aristas);

} /* creargrafo */

/*=============================================================*/

/* transformacion de las probabilidades
     segun Brockintong y Culberson */

double func1(x,p,n,s,i) double x;

double p;

int n;

int s;

int i; {
   double f1,f2;

   f1 = pow((double)(1.0-x),(double)i);
   f2 = x*f1*((double)(n-s)) - p*( ((double)(s-i-1)) +
          f1*((double)(n-s)) );
   return(f2);

} /* func1 */


/*=============================================================*/

int ocultar(vert,n_indep,f_prob,def)
/* Funcion que oculta el conjunto independiente
   de algunas heuristicas (Extraida del articulo
   de Brockintong y Culberson )
*/
```

```
int vert; int n_indep; float f_prob; int def;

{
  int fl,idef;
  double min,mid,max,prtemp;
  double fmin,fmid,fmax;
  double d_prob;
  double d_epsilon /*, eps*/;
  float prob12,prob22;

  d_epsilon = 1.0e-12;
  d_prob = (double)f_prob;
  idef = def - 1;

  min = d_prob;
  max = (1.0 + ((double)idef)*((double)d_prob) );
  max /= (1.0 + ((double)idef));
  if (idef == 0) { max = 0.99; }
  fmin = func1(min,d_prob,vert,n_indep,idef);
  fmax = func1(max,d_prob,vert,n_indep,idef);

  if (abs(fmax) < d_epsilon) {

    prob12 = (float) max;

  } else if (fmax < 0.0) {

    printf("error:probabilidad transformada negativa \n");
    exit(-1);

  } else {

    mid = (min+max)/2.0;
    fmid = func1(mid,d_prob,vert,n_indep,idef);
    while (fabs(fmid) >= d_epsilon) {
      if (fmid < 0) {
        min = mid;
      } else {
        max = mid;
      }
```

```
      mid = (min+max)/2;
      fmid = func1(mid,d_prob,vert,n_indep,idef);
    } /* while */
    prob12 = (float) mid;

  }

  prtemp = (float)(pow((double)(1.0-prob12),(double)idef)) *
           ((float)(vert-n_indep));
  prob22 = prob12*(prtemp-((double)(n_indep-idef))) /
           ( ((float)prtemp) - 1.0 );

  fl = creargrafo(vert,n_indep,prob12,prob22);
  return(fl);

} /* ocultar */

/*=========================================================*/

/* Libera la memoria reservada para almacenar el grafo */

int borrargrafo() {
  if(grafocreado != TRUE) {
    return(-1);
  } else {
    free(grafo);
    grafocreado = FALSE;
    return(0);
  }
}

/*=========================================================*/

/* Verificacion.h contiene todas las funciones
   que necesita B para comprobar la respuesta
   al reto 0
*/

#include "transformar.h"
```

```
BOOL comparacion(char *Gisomorfo,char *nombre);

void MostrarCjto(int ind, int InIso[MAXVERT]);
BOOL VerifReto1(char *binario, char *Gisomorfo,
                char *isomorfismo);

void leerIso(char *nomfichIso, int Iso[MAXVERT]);

BOOL Comparar(BOOL GIsoConstruido[MAXVERT][MAXVERT],
              BOOL GComp[MAXVERT][MAXVERT]);

/*=========================================================*/

BOOL comparacion(char *Gisomorfo,char *nombre)
{
    int cG, ocG, cN, ocN;
    char *ppG = descrip, *ppN = descrip;
    FILE *fp, *f;
    int iG = 0, jG = 0, iN = 0, jN = 0;
    int vG,aG, vN,aN;
    BOOL result = FALSE;

    if ( (fp=fopen(Gisomorfo,"r"))==NULL )
    {
        printf("\nError en la apertura del "
               "fichero %s",Gisomorfo);
      exit(10);
   }

    if( (f = fopen(nombre,"r")) == NULL)
    {
        printf("\nError en la apertura del "
               "fichero %s",nombre);
      exit(10);
   }
    for(ocG = '\0'
      ;(cG = fgetc(fp)) != EOF && (ocG != '\n' || cG != 'e')
        ; ocG = *ppG++ = cG);
    ungetc(cG, fp);
    *ppG = '\0';
```

```
parametros(&vG, &aG);
for(ocN = '\0'
  ;(cN = fgetc(f)) != EOF && (ocN != '\n' || cN != 'e')
    ; ocN = *ppN++ = cN);
ungetc(cN, f);
*ppG = '\0';
parametros(&vN, &aN);

if(vG != vN || aG != aN)
     return(result);
else{
    while (((cG = fgetc(fp)) != EOF)&&
           ((cN = fgetc(f)) != EOF)){
        switch (cG)
           {
            case 'e':
               if (!fscanf(fp, "␣%d␣%d", &iG, &jG))
               {
                    printf("Fichero␣de␣entrada␣%s␣no␣se"
                    "ajusta␣al␣formato\n", Gisomorfo);
                    exit(10);
               }
               else if (cN == 'e'){
                   if(!fscanf(f, "␣%d␣%d", &iN, &jN))
                   {
                       printf("Fichero␣de␣entrada␣%s␣no␣se"
                       "ajusta␣al␣formato\n", nombre);
                       exit(10);
                   }
                   else
                        {
                           if(iN != iG || jN != jG)
                               return(result);

                        }
               }
               else return(result);

            case '\n':
                   if (cN != '\n') return result;
```

```
                    default:
                        break;
                }/* switch */

            }/* while */
        if ((cG == EOF) && (cN == EOF)){
            fclose(fp);
            fclose(f);
            result = TRUE;
        }
    }
  return(result);
}

/*=========================================================*/

BOOL Comparar(BOOL GIsoConstruido[MAXVERT][MAXVERT],
                BOOL GComp[MAXVERT][MAXVERT])
{
    int i = 0,j = 0;
    BOOL Comp = FALSE;


    while ((i < vert || j < vert) &&
           (GIsoConstruido[i][j] == GComp[i][j]))
    {
        if (j == vert)
        {
            j = 0;
            i ++;
        }
            else j++;

    }
    if(i == vert && j == vert)
        Comp = TRUE;

    return Comp;
```

```
}

/*=========================================================*/

void leerIso(char *nomfichIso,int Iso[MAXVERT])
{
    FILE *archivo;
    int i = 0;
    bzero( Iso, vert );

    if ( (archivo = fopen(nomfichIso,"r"))==NULL )
    {
        printf("Fichero del isomorfismo %s no entregado\n",
                nomfichIso);
        exit(10);
    }
    else
    {
        i = 0;
        while (i < vert)
        {
            fscanf(archivo, "%d", &Iso[i]);

            i++;
        }
    }
    fclose(archivo);
}

/*=========================================================*/

BOOL VerifReto1(char *binario, char *Gisomorfo,
                char *isomorfismo)

/* comprueba si al aplicar el isomorfismo en el grafo
   original se obtiene el grafo comprometido */
{
    BOOL verif = FALSE;

    int *Iso;
```

```
  int i = 0, j = 0;

  int c, oc;
  char * pp = descrip;
  FILE * fp, *f;
  char nombre[MAXFICH];
  int v,a;

  printf("\nEl fichero binario con el grafo original"
         " entregado por A es %s", binario);
  printf("\nComienza transformacion ascci del fichero "
          "del grafo original %s", binario);

  write_graph_ascii( binario );

  Iso = crearVector();
  leerIso(isomorfismo, Iso);

  printf("\ncSe realiza la apertura del fichero ASCII "
         "que contiene el grafo original %s", binario);

  if ( (fp=fopen(binario,"r"))==NULL )
  {
      printf("\nError en la apertura del fichero "
             "%s",binario);
    exit(10);
}
  strcat(nombre, name);
  strcat(nombre, ".Giso");
  printf("\nEl nombre del fichero en el que B genera "
         "el grafo isomorfo es %s", nombre);
  if( (f = fopen(nombre,"w")) == NULL)
  {
      printf("\nError en la apertura del fichero "
             "%s",nombre);
    exit(10);
}
```

```
    for(oc = '\0'
        ;(c = fgetc(fp)) != EOF && (oc != '\n' || c != 'e')
        ; oc = *pp++ = c);
        fprintf(f,"%s",descrip);

    ungetc(c, fp);
    *pp = '\0';
    parametros(&v, &a);

    while ((c = fgetc(fp)) != EOF){
        switch (c)
          {
            case 'e':
              if (!fscanf(fp, " %d %d", &i, &j))
              {
                    printf("Fichero de entrada %s no se "
                          "ajusta al formato\n", binario);
                    exit(10);
              }

              fprintf(f, "e %d %d\n", Iso[i-1], Iso[j-1]) ;
            case '\n':
              default:
                    break;
                }/* switch */

    }/* while */
    fclose(fp);
    fclose(f);
    printf("\n El fichero que contiene el grafo isomorfo"
          " creado por A es %s", Gisomorfo);
    write_graph_bin( Gisomorfo );
    printf("\n El fichero que contiene el grafo isomorfo"
           " creado por B  %s", nombre);

    destruirVector(Iso);
    verif = comparacion(Gisomorfo,nombre);
    return(verif);

}
```

```
/*============================================================*/

/* Generador aleatorio de un grafo en el hay un conjunto
   independiente insertado desarrollado en ANSI C
   ( Generador de una Instancia dificil del problema
   del conjunto Independiente )
*/

/* Para usar el generador aleatorio de sistemas
   UNIX/LINUX compilar:
   gcc GInsIndep.c -DLINUX -lm
   Para usarlo en otros sistemas: cc GInsIndep.c.
*/

/* El presente codigo inserta un conjunto independiente
   segun el trabajo de Brockintong y Culberson, Camouflaging
   Independent Sets in Quasi-random Graphs
*/

/* El generador seudoaleatorio usado tiene dos posibilidades:
   - el usado por el incluido en los sitemas LINUX/UNIX
   (generador no lineal aditivo con órealimentacin en
   RedHat 7.2)
   - el generador **** (implementado en la libreria *****)
*/

/* Para la construccion del conjunto independiente se
   genera un clique en un grafo y posteriormente se
   calcula el grafo complemento
*/

#include <stdio.h>

#include <stdlib.h>


#include <string.h>
```

```
#include "AlgIS.h"

#include "Verificacion.h"


#ifndef DEF_BZERO
#define bzero( ptr, n ) memset( ptr, 0, n )
#endif

#define TRUE 1 #define FALSE 0



void error_entrada(char *cadena);
BOOL VerifReto1(char *binario, char *Gisomorfo,
                char *isomorfismo);

void conf_generador(void);
void conf_generador(void);
void est_semilla( int semilla_ini);
int generador(void);
void GenGrafIsIndep(int indep,
                    char *binario,
                    int Iorig[MAXVERT],
                    int IndepIso[MAXVERT],
                    int isomorfismo[MAXVERT],
                    char *SolIso);
void InicializarGrafo(BOOL G[MAXVERT][MAXVERT]);

/*=================================================*/

void error_entrada(cadena)

/* Control del formato de los parametros de generacion de
    entrada usados para la generacion de la instancia
*/
char *cadena;
```

```
{
    fprintf(stderr,"\nArgumentos: "
            "%s .... Parametros Incorrectos\n",cadena);
    fprintf(stderr,"Formato de la "
            "entrada:\n");
    fprintf(stderr,"\t -g???? "
            "(Numero de Vertices del Grafo\n)");
    fprintf(stderr,"\t -i???? "
            "(Numero de Vertices del  Conjunto "
            "Independiente\n");
    fprintf(stderr,"\t -p?.?????? "
            "(Probabilidad de las Aristas)\n");
    fprintf(stderr,"\t -n? "
            "(Nivel de Ocultacion)\n");
    fprintf(stderr,"\t -s???? "
            "(Semilla)\n\n");
    fprintf(stderr,"Valores por Defecto: "
            "-g20 -c9 -p0.500 -d0 -s0\n");
    exit(-1);
} /* error_entrada */

/*=========================================================*/

int main(argc,argv) int argc; char *argv[]; {

  int cx;          /* Controla la lectura de parametros
                      del main */
  int gsize;      /* Numero de vertices del grafo */
  int n_ind;      /*  Numero de vertices del conjunto i
                      independiente */
  float p;        /* Parametro del modelo de grafos aleatorios
                     usado G_{n,p} */
  int nivel;      /* Nivel de ocultacion del conjunto
                     independiente */
  int semilla;

  long longitud;

  FILE *fp;
  /* fichero binario que contiene el grafo original */
```

```
  char fichero[MAXFICH];

  /* fichero binario que contiene el grafo isomorfo */
  char fichGIso[MAXFICH];
  /* fichero ASCII que contiene el isomorfismo */
  /* y la ósolucin en el grafo isomorfo */
  char fichIso[MAXFICH], SolIso[MAXFICH];

  char mm[MAXFICH];
  char desc_fichero[2000];
  int Tampt;
  char *gptr;

    int i, j, k;
    BOOL G[MAXVERT][MAXVERT], Giso[MAXVERT][MAXVERT];
     int v, a;
    int Iso[MAXVERT];

    int IndepIso[MAXVERT];

/* Numero de iteraciones a desarrollar en el protocolo */
    int m;

/* Nivel de confianza de la demsotracion,
    solicitado por el  usuario B */
    double confianza;

    int reto;


/* Variable que controla si se ha cometido una estafa */
    BOOL VERIFICACION = TRUE;

 /* Configuracion del generador de numeros
    seudoaleatorios */



    strcat(name, "IndSet");
```

```
    conf_generador();

/* Parametros por defecto para la entrada */
  gsize = 800;
  n_ind = 35;
  p = 0.5;
  nivel = 0;
  semilla = 0;

/* Lectura de parametros del main */
  cx = 1;
  while(cx < argc) {

    if (argv[cx][0] != '-') { error_entrada(argv[cx]); }

    switch(argv[cx][1]) {
/* Numero de vertices del grafo */
      case 'g':
      case 'G':
       sscanf(argv[cx],"%*c%*c%d",&gsize);
       break;

/* Numero de vertices del conjunto independiente */
      case 'i':
      case 'I':
       sscanf(argv[cx],"%*c%*c%d",&n_ind);
       break;

/* Probabilidad en el modelo G_{n,p} */
      case 'p':
      case 'P':
       sscanf(argv[cx],"%*c%*c%f",&p);
       break;

/* Nivel de ocultacion: comprendido entre 0 y 4  */
      case 'n':
      case 'N':
       sscanf(argv[cx],"%*c%*c%d",&nivel);
       break;
```

```
    case 's':
    case 'S':
      sscanf(argv[cx],"%*c%*c%d",&semilla);
      break;

    default:
      error_entrada(argv[cx]);
      break;

  } /* switch */
  cx++;
}

if (gsize > MAXVERT) {
  fprintf(stderr,"\nGrafo demasiado grande "
          "(maximo: %d) \n",MAXVERT);
  exit(-1);
}

if (n_ind > gsize) {
  fprintf(stderr,"\nConjunto Independiente"
          "mayor que el grafo.\n");
  exit(-1);
}

if ((nivel > 4) || (nivel<0)) {
  fprintf(stderr,"\nEl nivel de ocultacion debe"
          "estar entre 0 y 4\n");
  exit(-1);
}

p = 1.0 - p;

est_semilla(semilla);

switch(nivel) {
  case 0:
        aristas = creargrafo(gsize,n_ind,p,p);
    break;
  case 1:
```

```
    case 2:
    case 3:
    case 4:
       aristas = ocultar(gsize,n_ind,p,nivel);
      break;
  }

  p = 1.0 - p;

  strcat(fichero,name);
  strcat(fichero,".b");
  fp = fopen(fichero,"w");
  if (fp == NULL) {
    fprintf(stderr,"\nError al crear fichero %s:\n",
            fichero);
    exit(-1);
  } /* if */
/*Mostrar la descripcion del fichero generado */


/* Calculo del tamano del fichero de salida  en
   funcion del numero de vertices del grafo
   generado
*/

  if (gsize < 1000) {
    Tampt = (int)(aristas/10485.76 + 0.5);
  } else if (gsize < 1400) {
    Tampt = (int)(aristas/9532.51 + 0.5);
  } else {
    Tampt = (int)(aristas/8738.11 + 0.5);
  } /* if */

  printf("Estimacion del tamano del fichero"
         " que contiene el grafo: "
         "%5.1fMB\n\n",(Tampt/10.0));
  sprintf(mm,"\np aristas %d %d\n",gsize,aristas);


  strcat(desc_fichero,mm);
```

```
  longitud = strlen(desc_fichero);
  fprintf(fp,"%d\n", longitud);
  fprintf(fp,desc_fichero);

/* Volcado de las aristas en el fichero */

  for(cx=0;cx<gsize;cx++) {
    gptr = grafo + cx*((gsize>>3)+1);
    fwrite(gptr, 1, (int)((cx + 8)/8), fp);
  }

  fclose(fp);
printf("\nTerminada la óreacin del grafo");

/* A transforma el fichero binario en ASCII para
   poder almacenar el grafo original y construir
   el grafo isomorfo */


  read_graph_bin( fichero );


  strcat(fichero, name);
  strcat(fichero,".dat" );
  printf("\nSe transforma en ASCII el "
           "fichero %s",fichero);
  write_graph_ascii( fichero );
  printf("\ónTransformacin terminada "
         "%s",fichero);

  InicializarGrafo(G);

  leerInst (G, &vert, &aristas,fichero);

  write_graph_bin(fichero);
  printf("\nConversion a binario "
         "del fichero %s",fichero);
  /*strcat(fichero, name);*/
  strcat(fichero, ".b");
```

```
/*Comienzo de la interaccion */

 printf("\nEl usuario B debe introducir el nivel de"
        "confianza deseado en la demostracion: 0.????:  ");
  scanf("%lg", &confianza);
  confianza = - log(1.0 - confianza)/log(2.);
 m = (int) ceil(confianza);
  printf("\nEl numero de iteraciones necesarias para "
         " alcanzar el nivel de confianza solicitado es:"
         "%d ", m);

   i = 1;
   VERIFICACION = TRUE;
    printf("\nVertices: %d Aristas: %d", vert, aristas);
    while ( VERIFICACION  && i < m )
    {

       GenGrafIsIndep(n_ind, fichero, almacen,
                    IndepIso, Iso, SolIso);

       strcat(fichIso,name);
       strcat(fichIso,".iso");

       GuardarCjto (vert, Iso,fichIso,TRUE);
       printf("\nEl nombre del fichero en el que A"
              " guarda el isomorfismo es %s",fichIso);

     printf("\nEl usuario B debe seleccionar un reto: ");
     printf("\n\t\t Reto = 0: Se le entregara el "
            "isomorfimo ");
     printf("\n\t\t Reto = 1: Debe seleccionar "
           "un vertice ");
       reto = (int) (((float) 1)*generador() /
                    (RAND_MAX+1.0));
       printf("\nRETO: %d ", reto);
     if (reto == FALSE)
       {
       /* Respuesta al primer reto */
```

```
                strcat(fichGIso,name);
                 strcat(fichGIso,".Giso");
                 VERIFICACION = VerifReto1(fichero, fichGIso,
                                fichIso);
                 if(VERIFICACION == FALSE){
                 printf("\n Estafa detectada!!! "
                       "La identificacion se interrumpe");
                     exit(-1);
                 }
                 else i++;
            }
          else{
              write_graph_ascii(fichGIso);
              leerInst (Giso, &v, &a,fichGIso);
              leerIso(SolIso, IndepIso);
              for (j = 0; j < vert; j++){
                      for(k = j + 1; k < vert; k++){
                          if (Giso[j][k] == TRUE){
                                VERIFICACION = FALSE;
                                printf("\n Estafa detectada!!!"
                                " La identificacion se"
                                " interrumpe");
                                exit(-1);

                            }
                          }

              }
              if (VERIFICACION == TRUE) i++;
          }
    }/* fin de las iteraciones */

        if (VERIFICACION == TRUE) printf("Protocolo"
            " desarrollado correctamente");
        return 0;

} /* main */
```

Bibliografía

[AAB+89] M. Abadi, E. Allender, A. Broder, et al. *On generating solved instances of computational problems*. *Advances in Cryptology - Crypto '88*, ed. S. Goldwasser, págs. 297–310. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 403.

[ABSS96] G. Ateniese, C. Blundo, A. D. Santis, et al. *Visual cryptography for general access structures*. Information and Computation, 129(2), 86–106 (1996).

[AN95] R. Anderson, R.Ñeedham. *Robustness principles for public key protocols*. *Advances in Cryptology - Crypto '95*, ed. D. Coppersmith, págs. 236–247. Springer-Verlag, Berlin (1995). Lecture Notes in Computer Science Volume 963.

[AN96] M. Abadi, R.Ñeedham. *Prudent engineering practice for cryptographic protocols*. IEEE Transactions on Software Engineering, 22(1), 6–15 (1996). URL `citeseer.nj.nec.com/abadi96prudent.html`.

[ASW97] N. Asokan, M. Schunter, M. Waidner. *Optimistic protocols for fair exchange*. *ACM Conference on Computer and Communications Security*, págs. 7–17 (1997). URL `citeseer.nj.nec.com/article/asokan96optimistic.html`.

[Bab85] L. Babai. *Trading group theory for randomness. 17th ACM Symposium on the Theory of Computing*, págs. 421–429 (1985).

[Bac89] E. Bach. *Intractable problems in number theory (invited talk). Advances in Cryptology - Crypto '88*, ed. S. Goldwasser, págs. 77–93. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 403.

[BAN90] M. Burrows, M. Abadi, R.Ñeedham. *A logic of authentication*. ACM Transactions on Computer Systems, 1(8), 18–36 (1990).

[Bar01] B. Barak. *How to go beyond the black-box simulation barrier. 42nd Annual Symposium on Foundations of Computer Science, FOCS 2001*, págs. 106–115 (2001).

[BC89] G. Brassard, C. Crépeau. *Sorting out zero-knowledge. Advances in Cryptology - EuroCrypt '89*, eds. J.-J. Quisquater, J. Vandewalle, págs. 181–191. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 434.

[BC90] G. Brassard, C. Crépeau. *Quantum bit commitment and coin tossing protocols. Advances in Cryptology - Crypto '90*, eds. A. J. Menezes, S. A. Vanstone, págs. 49–61. Springer-Verlag, Berlin (1990). Lecture Notes in Computer Science Volume 537.

[BC94] M. Brockington, J. Culberson. *Camouflaging independent sets in quasi-random graphs. Cliques, Coloring and Satisfability*, eds. D. Johnson, M. Trick, tomo XXVI, págs. 75–88. American Mathematical Society (1994).

[BCC88] G. Brassard, D. Chaum, C. Crépeau. *Minimum disclosure proofs of knowledge*. Journal of Computer and System Sciences, 37(2), 156–189 (1988).

[BCR86] G. Brassard, C. Crépeau, J. M. Robert. *All-or-nothing disclosure of secrets*. *Advances in Cryptology - Crypto '86*, ed. A. M. Odlyzko, págs. 234–238. Springer-Verlag, Berlin (1986). Lecture Notes in Computer Science Volume 263.

[BDCGL92] S. Ben-David, B. Chor, O. Goldreich, et al. *On the Theory of Average Case Complexity*. Journal of Computer and System Sciences, 44(2), 193–219 (1992).

[BDG88] J. L. Balcázar, J. Diaz, J. Gabarró. *Structural Complexity I*. Springer (1988).

[Bea91] D. Beaver. *Secure multiparty protocols and zero-knowledge proof systems tolerating a faulty minority*. Journal of Cryptology, 4(2), 75–122 (1991).

[Bea95] D. Beaver. *Precomputing oblivious transfer*. *Advances in Cryptology - Crypto '95*, ed. D. Coppersmith, págs. 97–109. Springer-Verlag, Berlin (1995). Lecture Notes in Computer Science Volume 963.

[Bel98] M. Bellare. *Practice-Oriented Provable-Security*, cap. 1, págs. 1–15. Tomo LNCS 1561, LNCS Tutorial de Damgard [Dam99] (1998).

[Bet88] T. Beth. *Efficient zero-knowledged identification scheme for smart cards*. *Advances in Cryptology - EuroCrypt '88*, ed. C. G. Günther, págs. 77–86. Springer-Verlag, Berlin (1988). Lecture Notes in Computer Science Volume 330.

[BFM88] M. Blum, P. Feldman, S. Micali. *Non-interactive zero-knowledge and its applications*. *Proceedings of the Twentieth Annual ACM Symposium on Theory of Computing*, págs. 103 – 112 (1988).

[BFS91] A. Broeder, A. Frieze, E. Shamir. *On hidden hamiltonian cycles*. *Proceedings of the 23rd Annual ACM Symposium on Theory of Computing*, págs. 182–189 (1991).

[BG89] M. Bellare, S. Goldwasser. *New paradigms for digital signatures and message authentication based on non-interactive zero knowledge proofs*. *Advances in Cryptology - Crypto '89*, ed. G. Brassard, págs. 194–211. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 435.

[BG92] M. Bellare, O. Goldreich. *On defining proofs of knowledge*. *Advances in Cryptology - Crypto '92*, ed. E. F. Brickell, págs. 390–420. Springer-Verlag, Berlin (1992). Lecture Notes in Computer Science Volume 740.

[BG02] B. Barak, O. Goldreich. *Universal arguments and their applications*. *17th IEEE Annual Conference on Computational Complexity*, págs. 194–203 (2002).

[BGH+91] R. Bird, I. Gopal, A. Herzberg, et al. *Systematic design of two-party authentication protocols*. *Advances in Cryptology - Crypto '91*, ed. J. Feigenbaum, págs. 44–61. Springer-Verlag, Berlin (1991). Lecture Notes in Computer Science Volume 576.

[BKP86] R. Berger, S. Kannan, R. Peralta. *A framework for the study of cryptographic protocols*. *Advances in Cryptology - Crypto '85*, ed. H. C. Williams, págs. 87–103. Springer-Verlag, Berlin (1986). Lecture Notes in Computer Science Volume 218.

[BL89] J. Benaloh, J. Leichter. *Generalized secret sharing and monotone functions*. *Advances in Cryptology - Crypto '88*, ed. S. Goldwasser, págs. 27–36. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 403.

[Bla79] G. R. Blakley. *Safeguarding cryptographic keys*. *Proceedings of the National Conputer Conference*, tomo 48, págs. 242–268 (1979).

[Blu82a] M. Blum. *Coin flipping by telephone: A protocol for solving impossible problems*. *Advances in Cryptography: Proceedings of Crypto'81*, ed. A. Gersho, págs. 11–15. University of California, Santa Barbara, Santa Barbara, California, USA (1982).

[Blu82b] M. Blum. *Coin Flipping by Telephone: a Protocol for Solving Impossible Problems*. IEEE Computer Conference, págs. 133–137 (1982).

[Blu86] M. Blum. *How to prove a theorem so that no one else can claim it*. *International Congress of Mathematicians, Berkeley*, págs. 1444–1451 (1986).

[BM89] M. Bellare, S. Micali. *Non-interactive oblivious transfer and applications*. *Advances in Cryptology - Crypto '89*, ed. G. Brassard, págs. 547–559. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 435.

[BM94] D. Bleichenbacher, U. M. Maurer. *Directed acyclic graphs, one-way functions and digital signatures*. *Advances in Cryptology - Crypto '94*, ed. Y. Desmedt, págs. 75–82. Springer-Verlag, Berlin (1994). Lecture Notes in Computer Science Volume 839.

[BMSW00] C. Blundo, B. Masucci, D. Stinson, et al. *Constructions and bounds for unconditionally secure commitment schemes*. Cryptology ePrint Archive, Report 2000/043 (2000). `http://eprint.iacr.org/`.

[BO85] R. V. Book, F. Otto. *The verifiability of two-party protocols.*. *Advances in Cryptology - EuroCrypt '85*, ed. F. Pichler, págs. 254–260. Springer-Verlag, Berlin (1985). Lecture Notes in Computer Science Volume 219.

[BOGG+89] M. Ben-Or, O. Goldreich, S. Goldwasser, et al. *Everything provable is provable in zero-knowledge*. *Advances in Cryptology - Crypto '88*, ed. S. Goldwasser, págs. 37–56. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 403.

[BP02] M. Bellare, A. Palacio. *GQ and schnorr identification schemes: Proofs of security against impersonation under active and concurrent attacks*. *Advances in Cryptology: Proceedings of Crypto '02*. Springer-Verlag (2002). Lecture Notes in Computer Science Volume.

[BPT84] R. Berger, R. Peralta, T. Tedrick. *A provably secure oblivious transfer protocol*. *Advances in Cryptology: Proceedings of EuroCrypt '84*, eds. T. Beth, N. Cot, , et al., págs. 379–386. Springer-Verlag, Berlin (1984). Lecture Notes in Computer Science Volume 209.

[Bra88] G. Brassard. *Modern Cryptology (A Tutorial)*. Lecture Notes in Computer Science (325). Springer-Verlag (1988).

[BSMP91] M. Blum, A. D. Santis, S. Micali, et al. *Noninteractive zero-knowledge*. SIAM J. Comput., 20(6), 1084–1118 (1991). URL `citeseer.nj.nec.com/blum91noninteractive.html`.

[But99] L. Buttyán. *Formal methods in the design of cryptographic protocols*. Inf. Téc. SSC/1999/038, EPFL SSC, Lausanne, Switzerland (1999). URL `http://citeseer.nj.nec.com/buttyan99formal.html`.

[BVV84] M. Blum, U. V. Vazirani, V. V. Vazirani. *Reducibility among protocols*. *Advances in Cryptology: Proceedings of Crypto '83*, ed. D. Chaum, págs. 137–146. Plenum Publishing, New York, USA (1984).

[Can00] R. Canetti. *Security and composition of multiparty cryptographic protocols*. Journal of Cryptology: the journal of the International

Association for Cryptologic Research, 13(1), 143–202 (2000). URL citeseer.nj.nec.com/canetti98security.html.

[CdBvH+89] D. Chaum, B. den Boer, E. van Heyst, et al. *Efficient offline electronic checks (extended abstract). Advances in Cryptology - EuroCrypt '89*, eds. J.-J. Quisquater, J. Vandewalle, págs. 294–301. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 434.

[CF01] R. Canetti, M. Fischlin. *Universally composable commitments. Advances in Cryptology -Proceedings Crypto 2001*, ed. J.Kilian, págs. 19–40. Springer-Verlag (2001). Lecture Notes in Computer Science Vol. 2139.

[CGGM00] R. Canetti, O. Goldreich, S. Goldwasser, et al. *Resettable zero-knowledge. Proceedings of 32nd Annual ACM Symposium on Theory of Computing*, 42, págs. 235–244 (2000). URL citeseer.nj.nec.com/canetti99resettable.html.

[CGT95] C. Crépeau, J. v. d. Graaf, A. Tapp. *Committed oblivious transfer and private multi-party computation. Advances in Cryptology - Crypto '95*, ed. D. Coppersmith, págs. 110–123. Springer-Verlag, Berlin (1995). Lecture Notes in Computer Science Volume 963.

[CH01] P. Caballero, C. Hernández. *Strong Solutions to the Identification Problem. Computing and Combinatorics*, ed. J. Wang, págs. 257–261. Springer-Verlag, Berlin (2001). Lecture Notes in Computer Science Volume 2108.

[CH02a] P. Caballero, C. Hernández. *General interactive zero-knowledge schemes*. International Mathematical Journal, 2(1), 31–38 (2002).

[CH02b] P. Caballero, C. Hernández. *How to Construct Cryptographic Protocols from Graph Theory. Proceedings of 6th Multi-Conference on Systemics,*

Cybernetics and Informatics - CISIC 2002, ed. F. Y. N. Callaos, págs. 456–461. Florida, U.S.A (2002). Volume I.

[CH03a] P. Caballero, C. Hernández. *A zero-knowledge identification scheme based on an average-case NP-complete problem. "Mathematical Methods, Models and Architectures for Computer Networks Security"*. Springer-Verlag, Berlin (2003). Lecture Notes in Computer Science, Por aparecer.

[CH03b] P. Caballero, C. Hernández. *Zero-knowledge interactive proof of knowledge for the independent set problem. Algorithms and Computation*. Springer-Verlag (2003). Lecture Notes in Computer Science, En revisión.

[Cha87] D. Chaum. *Blinding for unanticipated signatures. Advances in Cryptology - EuroCrypt '87*, eds. D. Chaum, W. L. Price, págs. 227–236. Springer-Verlag, Berlin (1987). Lecture Notes in Computer Science Volume 304.

[CHB02a] P. Caballero, C. Hernández, C. Bruno. *Cryptographic applications. Computational Mathematics, Modelling and Algorithms*, ed. J. Misra, tomo Mathematical Sciences Series, págs. 214–229. Narosa Publishing House (2002).

[CHB02b] P. Caballero, C. Hernández, C. Bruno. *Diseño de Protocolos Criptográficos Bipartitos. Actas Del 1er. Congreso Internacional de la Sociedad de la Información Y Del Conocimiento - CISIC 2002*, págs. 369–376. McGrawHill, Gran Canaria, España (2002).

[CHB03a] P. Caballero, C. Hernández, C. Bruno. *Digital affairs: Two-party cryptoprotocols. Techno-Legal Aspects of Information Society and New Economy: An Overview*, Information Society. FORMATEX (2003).

[CHB03b] P. Caballero, C. Hernández, C. Bruno. *The teaching of cryptologic mathematics*. *Proceedings of the International Conference on Computer Systems and Technologies - CompSysTech'2003*. Sofía, Bulgaria (2003). Por aparecer.

[CK93] C. Crépeau, J. Kilian. *Discreet solitary games*. *Advances in Cryptology - Crypto '93*, ed. D. R. Stinson, págs. 319–330. Springer-Verlag, Berlin (1993). Lecture Notes in Computer Science Volume 773.

[Cop86] D. Coppersmith. *Cheating at mental poker*. *Advances in Cryptology - Crypto '85*, ed. H. C. Williams, págs. 104–107. Springer-Verlag, Berlin (1986). Lecture Notes in Computer Science Volume 218.

[Cra99] R. Cramer. *Introduction to Secure Computation*, cap. 2, págs. 16–62. Tomo LNCS 1561, LNCS Tutorial de Damgard [Dam99] (1999).

[Cré86a] C. Crépeau. *A secure poker protocol that minimizes the effect of player coalitions*. *Advances in Cryptology - Crypto '85*, ed. H. C. Williams, págs. 73–86. Springer-Verlag, Berlin (1986). Lecture Notes in Computer Science Volume 218.

[Cré86b] C. Crépeau. *A zero-knowledge poker protocol that achieves confidentiality of the players' strategy or how to achieve an electronic poker face*. *Advances in Cryptology - Crypto '86*, ed. A. M. Odlyzko, págs. 239–250. Springer-Verlag, Berlin (1986). Lecture Notes in Computer Science Volume 263.

[Cré87] C. Crépeau. *Equivalence between two flavours of oblivious transfers (cryptography)*. *Advances in Cryptology - Crypto '87*, ed. C. Pomerance, págs. 350–354. Springer-Verlag, Berlin (1987). Lecture Notes in Computer Science Volume 293.

[Cré90] C. Crépeau. *Correct and Private Reductions Among Oblivious Transfers*. Tesis Doctoral, Massachusetts Institute of Technology (1990).

[Cré93] C. Crépeau. *Cryptographic primitives and quantum theory*. *PhysComp '92, Proceedings of the second Physics of Computation Workshop*, págs. 200–204 (1993).

[Cré97] C. Crépeau. *Efficient cryptographic protocols based on noisy channels*. *Advances in Cryptology - EuroCrypt '97*, ed. W. Fumy, págs. 306–317. Springer-Verlag, Berlin (1997). Lecture Notes in Computer Science Volume 1233.

[CW87] D. Coppersmith, S. Winograd. *Matrix multiplication via arithmetic progresions*. *Proc. Nineteenth Annual ACM Symposium on Theory of Computing*, págs. 1–6. New York (1987).

[Dam89] I. B. Damgård. *On the existence of bit commitment schemes and zero-knowledge proofs*. *Advances in Cryptology - Crypto '89*, ed. G. Brassard, págs. 17–29. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 435.

[Dam99] I. Damgard, ed. *Lectures on Data Security*. Springer (1999).

[dB90] B. den Boer. *Oblivious transfer protecting secrecy*. *Advances in Cryptology - EuroCrypt '90*, ed. I. B. Damgård, págs. 31–45. Springer-Verlag, Berlin (1990). Lecture Notes in Computer Science Volume 473.

[DC95] G. Di Crescenzo. *Recycling random bits in composed perfect zero-knowledge*. *Advances in Cryptology - EuroCrypt '95*, eds. L. C. Guillou, J.-J. Quisquater, págs. 367–381. Springer-Verlag, Berlin (1995). Lecture Notes in Computer Science Volume 921.

[DF99] J. Domingo-Ferrer. *Anonymous fingerprinting based on committed oblivious transfer*. *Public Key Cryptography*, tomo 1560, págs. 43–52. Springer-Verlag, Berlín (1999). URL `citeseer.nj.nec.com/domingo-ferrer99anonymous.html`. Lecture Notes in Computer Science, Volume1560.

[DGB87] Y. Desmedt, C. Goutier, S. Bengio. *Special uses and abuses of the Fiat Shamir passport protocol*. *Advances in Cryptology - Crypto '87*, ed. C. Pomerance, págs. 21–39. Springer-Verlag, Berlin (1987). Lecture Notes in Computer Science Volume 293.

[DN02] I. Damgård, J.Ñielsen. *Perfect hiding and perfect binding universally composable commitment schemes with constant expansion factor*. *Advances in Cryptology - Proceedings of Crypto 2002*, ed. M. Yung, págs. 581–596. Springer-Verlag (2002). Lecture Notes in Computer Science Vol. 2442.

[DNS98] C. Dwork, M.Ñaor, A. Sahai. *Concurrent zero-knowledge*. *30th ACM Symposium on the Theory of Computing*, págs. 409–418 (1998).

[DPP93] I. B. Damgård, T. P. Pedersen, B. Pfitzmann. *On the existence of statistically hiding bit commitment schemes and fail-stop signatures*. *Advances in Cryptology - Crypto '93*, ed. D. R. Stinson, págs. 250–265. Springer-Verlag, Berlin (1993). Lecture Notes in Computer Science Volume 773.

[DS81] D. Denning, G. Sacco. *Timestamps in Key Distribution Protocols*. Communications of the ACM, 8(24), 198–208 (1981).

[DSMP87] A. De Santis, S. Micali, G. Persiano. *Noninteractive zero-knowledge proof systems*. *Advances in Cryptology - Crypto '87*, ed. C. Pomerance,

págs. 52–72. Springer-Verlag, Berlin (1987). Lecture Notes in Computer Science Volume 293.

[DSMP89] A. De Santis, S. Micali, G. Persiano. *Non-interactive zero-knowledge with preprocessing*. *Advances in Cryptology - Crypto '88*, ed. S. Goldwasser, págs. 269–283. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 403.

[DST00] Y. Dodis, S.Halevi, T.Rabin. *A cryptographic solution to a game theory problem*. *Advances in Cryptology: Proceedings of CRYPTO'2000*, ed. M. Bellare, págs. 112–130. Springer (2000). Lecture Notes in Computer Science Volume 1880.

[dWQ93] D. de Waleffe, J. Quisquater. *Better Login Protocols for Computer Networks*, págs. 50–71. LNCS 741. Springer-Verlag (1993).

[EGL82] S. Even, O. Goldreich, A. Lempel. *A randomized protocol for signing contracts (extended abstract)*. *Advances in Cryptology: Proceedings of Crypto '82*, eds. D. Chaum, R. L. Rivest, A. T. Sherman, págs. 205–210. Plenum Publishing, New York, USA (1982).

[Erd59] P. Erdös. *Graph theory and probability*. Canad. J. Math, 11, 34–38 (1959).

[Eve82] S. Even. *A protocol for signing contracts*. *Advances in Cryptography: Proceedings of Crypto'81*, ed. A. Gersho, págs. 148–153. University of California, Santa Barbara, Santa Barbara, California, USA (1982).

[EY80a] S. Even, Y. Yacobi. *Relations among public key signature systems*. Inf. Téc. 175, Computer Science Department, Technion, Haifa, Israel (1980).

[EY80b] S. Even., Y. Yacoby. *Relations among Public Key Signature Systems*. Inf. Téc. 175, Computer Science Department,Technion, Haifa, Israel (1980).

[Fei90] U. Feige. *Alternative Models for Zero Knowledge Interactive Proofs*. Tesis Doctoral, The Weizmann Institute of Science, Rehovot, Israel (1990).

[FFS88] U. Feige, A. Fiat, A. Shamir. *Zero-knowledge proofs of identity*. Journal of Cryptology, 1, 77–95 (1988).

[FK94] M. Fellows, N. Koblitz. *Combinatorially based cryptography for children (and adults). Proc. 24th Southeastern Intern. Conf. Combinatorics, Graph Theory and Computing*, págs. 9–41 (1994). URL `citeseer.nj.nec.com/95924.html`.

[FK98] U. Feige, J. Kilian. *Zero knowledge and the chromatic number*. Journal of Computer and System Sciences, 57(2), 187—199 (1998).

[FLS90] U. Feige, D. Lapidot, A. Shamir. *Multiple non-interactive zero knowledge proofs based on a single random string. 31st Annual Symposium on Foundations of Computer Science*, tomo I, págs. 22–24 (1990).

[FM85] S. Fortune, M. Merritt. *Poker protocols. Advances in Cryptology: Proceedings of Crypto '84*, eds. G. R. Blakley, D. Chaum, págs. 454–466. Springer-Verlag, Berlin (1985). Lecture Notes in Computer Science Volume 196.

[FM97] A. M. Frieze, C. McDiarmid. *Algorithmic theory of random graphs*. Random Structures and Algorithms, 10(1-2), 5–42 (1997). URL `citeseer.nj.nec.com/frieze97algorithmic.html`.

[FMR96] M. J. Fisher, S. Micali, C. Rackoff. *A secure protocol for the oblivious transfer*. Journal of Cryptolofy, 9, 191–195 (1996).

[For87] L. Fortnow. *The complexity of perfect zero-knowledge*. *Proceedings of the Nineteenth Annual ACM Symposium on Theory of Computing, STOC 87*, págs. 204–209 (1987). URL `citeseer.nj.nec.com/fortnow87complexity.html`.

[For96a] S. Fortin. *The Graph Isomorphism Problem*. Inf. Téc. TR 96-20, University of Alberta, Department of Computer Science (1996). URL `ftp://ftp.cs.ualberta.ca/pub/TechReports/1996/TR96-20/TR96-20.ps.gz`.

[For96b] S. Fortin. *The Graph Isomorphism Problem*. Inf. Téc. TR 96-20, University of Alberta, Department of Computer Science (1996). URL `ftp://ftp.cs.ualberta.ca/pub/TechReports/1996/TR96-20/TR96-20.ps.gz`.

[Fra93] M. Franklin. *Complexity and Security of Distributed Protocols*. Tesis Doctoral, Columbia University (1993). URL `http://citeseer.nj.nec.com/franklin93complexity.html`.

[Fre79] R. Freivalds. *Fast probabilistic algorithms*. *Proc.*, ed. J. Becvár, tomo 74 de *Lecture Notes in Computer Science*, págs. 57–69. Springer, Olomouc, Czechoslovakia (1979).

[FS86] A. Fiat, A. Shamir. *How to prove yourself: practical solutions to identification and signature problems*. *Advances in Cryptology - Crypto '86*, ed. A. M. Odlyzko, págs. 186–194. Springer-Verlag, Berlin (1986). Lecture Notes in Computer Science Volume 263.

[FS90] U. Feige, A. Shamir. *Witness indistinguishable and witness hiding protocols*. *Proceedings of the 22nd STOC*, págs. 416–426 (1990).

[FW93] M. Fischer, R. Wright. *Advances in Computational Complexity Theory*, tomo 13 de *DIMACS Series in Discrete Mathematics and Theoretical*

Computer Science, cap. An Application of Game-Theoretic Techniques to Cryptography, págs. 99–118. American Mathematical Society (1993).

[FW00] M. Franklin, R. Wrigh. *Secure communication in minimal connectivity models*. Journal of Cryptology, 13, 9–30 (2000).

[GB01] S. Goldwasser, M. Bellare. *Lecture Notes on Cryptography* (2001). URL `http://www-cse.ucsd.edu/users/mihir/papers/gb.html`. University of California, Computer Science and Engineering.

[GJ79] M. R. Garey, D. Johnson. *Computers ab¡nd Intractavility: A Guide to the Theory of NP-Completeneess*. W.H. Freeman and Company (1979).

[GK96a] O. Goldreich, H. Krawczyk. *On the composition of Zero-Knowledge Proof systems*. SIAM Journal on Computing, 25(1), 169–192 (1996). URL `citeseer.nj.nec.com/goldreich90composition.html`.

[GK96b] O. Goldreich, H. Krawczyk. *On the composition of zero-knowledge proof systems*. SIAM Journal on Computing, 25(1), 169–192 (1996).

[GM84] S. Goldwasser, S. Micali. *Probabilistic Encryption*. Journal of Computer and System Sciences, 28, 270–299 (1984).

[GMR85] S. Goldwasser, S. Micali, C. Rackoff. *The knowledge complexity of interactive proof-systems*. *Proceedings of the Seventeenth Annual ACM Symposium on Theory of Computing (STOC 85)*, págs. 291– –304 (1985).

[GMR89] S. Goldwasser, S. Micali, C. Rackoff. *The Knowledge Complexity of Interactive Proof Systems*. SIAM Journal on Computing, 18, 186–208 (1989).

[GMW86] O. Goldreich, S. Micali, A. Wigderson. *How to prove all NP-statements in zero-knowledge, and a methodology of cryptographic protocol design*.

Advances in Cryptology - Crypto '86, ed. A. M. Odlyzko, págs. 171–185. Springer-Verlag, Berlin (1986). Lecture Notes in Computer Science Volume 263.

[GMW87] O. Goldreich, S. Micali, A. Wigderson. *How to Solve any Protocol Problem. Proceedings of the 19th STOC*, págs. 218–229 (1987).

[Gol97] O. Goldreich. *Combinatorial property testing (a survey)*. Electronic Colloquium on Computational Complexity (ECCC), 4(56) (1997).

[Gol98] O. Goldreich. *Secure multi-pary computation* (1998). Working draft, June 1998.

[Gol99a] O. Goldreich. *Introduction to complexity theory* (1999). Lecture Notes available at url: http://www.wisdom.weizmann.ac.il/ oded/cc.html.

[Gol99b] O. Goldreich. *Modern Cryptography, Probabilistic Proofs and Pseudo-Randomness*. Algoritms and Combinatorics. Springer (1999).

[Gol01a] O. Goldreich. *Foundations of Cryptography* (2001). Draft, Weizzman Institute, Vol. 2.

[Gol01b] O. Goldreich. *Secure Multiparty Computation* (2001). URL `http://www.wisdom.weizmann.ac.il/õded/pp.html`. Working Draft, Weizzman Institute.

[Gol02a] O. Goldreich. *Property Testing in Massive Graphs*, cap. 5, págs. 123–147. Kluwer Academic Publishers, j. abello and p. m. pardalos and m. g. c. resende ed$^{\underline{\text{ón}}}$. (2002).

[Gol02b] O. Goldreich. *Zero-knowledge twenty years after its invention*. Cryptology ePrint Archive, Report 2002/186 (2002). `http://eprint.iacr.org/`.

[GQ88] L. C. Guillou, J. J. Quisquater. *A practical zero-knowledge protocol fitted to security microprocessor minimizing both transmission and memory. Advances in Cryptology - EuroCrypt '88*, ed. C. G. Günther, págs. 123–128. Springer-Verlag, Berlin (1988). Lecture Notes in Computer Science Volume 330.

[GS86] S. Goldwasser, M. Sipser. *Private coins versus public coins in interactive proof systems. Proceedings of the 18thAnnual ACM Symposium on the Theory of Computing (STOC)*, págs. 59–68 (1986).

[GT01] O. Goldreich, L. Trevisan. *Three theorems regarding testing graph properties. 42nd Annual Symposium on Foundations of Computer Science (FOCS2001)*, págs. 460–469. IEEE Computer Society, Las Vegas, Nevada, USA (2001).

[HC97] C. Hernández, P. Caballero. *Demostraciones de conocimiento nulo interactivas usando grafos. Actas de Las III Jornadas de Informática*, págs. 187–196. Cádiz, España (1997).

[HC98] C. Hernández, P. Caballero. *Demostraciones de conocimiento nulo basadas en la teoría de grafos. Actas de la V Reunión Española Sobre Criptología Y Seguridad de la Información - V RECSI*, págs. 101–112. Málaga, España (1998).

[HC99a] C. Hernández, P. Caballero. *Interactive zero-knowledge proofs based in graphs. Proceedings of 1999 SIAM Annual Meeting*. Georgia, USA (1999).

[HC99b] C. Hernández, P. Caballero. *Interactive zero-knowledge proofs: Identification tool based in graph theory problems. 1999 IEEE Symposium on Security and Privacy*. California, USA (1999).

[HC00a] C. Hernández, P. Caballero. *Conocimiento nulo sobre problemas NP-completos. Actas de la VI Reunión Española Sobre Criptología Y Seguridad de la Información - VI RECSI*, eds. P. Caballero, C. Hernández, págs. 65–72. Ra-Ma, Tenerife, España (2000).

[HC00b] C. Hérnandez, P. Caballero. *Design of zero-knowledge protocols using graphs. Recent Trends in Mathematical Sciences*, eds. J. C. Misra, S. Sinha, págs. 349–358. Narosa Publishing House, Kharagpur, India (2000).

[HC01] C. Hernández, P. Caballero. *Proofs for protecting knowledge*. International Journal of Applied Mathematics, 7(2), 169–182 (2001).

[HC02] C. Hernández, P. Caballero. *La teoría de grafos como alternativa para el diseño de criptoprotocolos. Actas de la VII Reunión Española Sobre Criptología Y Seguridad de la Información - VII RECSI*, tomo I, págs. 109–122. Oviedo, España (2002).

[HC03a] C. Hernández, P. Caballero. *A new role of graph theory: The design of probably secure cryptoprotocols*. Information System Management (2003). En revisión.

[HC03b] C. Hernández, P. Caballero. *A secret sharing scheme using matrices. Computational Science and its Applications*. Springer-Verlag (2003). Lecture Notes in Computer Science, Por aparecer.

[HCB02a] C. Hernández, P. Caballero, C. Bruno. *Aplicación criptográfica de problemas unidireccionales de la teoría de grafos* (2002). Póster.

[HCB02b] C. Hernández, P. Caballero, C. Bruno. *Propuestas de solución al problema de la identificación. Actas Del 1er. Congreso Iberoamericano de Seguridad Informática*. Morelia, México (2002). En CD-ROM.

[HM96] S. Halevi, S. Micali. *Practical and provably-secure commitment schemes from collision-free hashing*. *Advances in Cryptology - Crypto '96*, ed. N. Koblitz, págs. 201–215. Springer-Verlag, Berlin (1996). Lecture Notes in Computer Science Volume 1109.

[IL90] R. Impagliazzo, L. A. Levin. *No better ways to generate hard NP instances than picking uniformly at random*. *Proceedings of the 31st IEEE Symposium on Foundations of Computer Science (FOCS)*, págs. 812–821 (1990).

[IS90] I. Ingemarsson, G. J. Simmons. *A protocol to set up shared secret schemes without the assistance of mutually trusted party*. *Advances in Cryptology - EuroCrypt '90*, ed. I. B. Damgård, págs. 266–282. Springer-Verlag, Berlin (1990). Lecture Notes in Computer Science Volume 473.

[ISN87] M. Ito, A. Saito, T.Ñishizek. *Secret sharing scheme realizing general access structure*. *Proceedings IEEE Globecom '87*, págs. 99–102 (1987).

[JP98] A. Juels, M. Peinado. *Hiding cliques for cryptographic security*. *SODA: ACM-SIAM Symposium on Discrete Algorithms (A Conference on Theoretical and Experimental Analysis of Discrete Algorithms)*, págs. 678–684 (1998). URL `citeseer.nj.nec.com/article/peinado98hiding.html`.

[Kar76] R. Karp. *The Probabilistic Analysis of Some Combinatorial Search Algorithms*. Academic Press, NY (1976). Referencia cruzada de artículo Levin Venkatesan.

[KH83] G. Karnin, Hellman. *On secret sharing systems*. IEEE Transactions on Information Theory, IT-29, 35–41 (1983).

[Kil88] J. Killian. *Founding Cryptography on Oblivious Transfer*. *Proceedings of 20th ACM Symposium on Theory of Computing*, págs. 20–31. Chicago (1988).

[KKMO94] J. Killian, E. Kushilevitz, S. Micali, et al. *Reducibility and Completeness in Private Computations*. SIAM Journal on Computing, 29(4), 1189–1208 (1994).

[KKOT90] K. Kurosawa, Y. Katayama, W. Ogata, et al. *General public key residue cryptosystems and mental poker protocols*. *Advances in Cryptology - EuroCrypt '90*, ed. I. B. Damgård, págs. 374–388. Springer-Verlag, Berlin (1990). Lecture Notes in Computer Science Volume 473.

[Kot85] S. C. Kothari. *Generalized linear threshold scheme*. *Advances in Cryptology: Proceedings of Crypto '84*, eds. G. R. Blakley, D. Chaum, págs. 231–241. Springer-Verlag, Berlin (1985). Lecture Notes in Computer Science Volume 196.

[KR02] S. Kremer, J.-F. Raskin. *Game analysis of abuse-free contract signing*. Inf. Téc. 474, ULB (2002).

[Kuč92] L. Kučera. *A Generalized Encryption Scheme Based on Random Graphs*. *Graph-Teorethic Concepts in Computer Science, 17th International Workshop, WG'91*, eds. G. Schimdt, B. R, págs. 180–186. Springer-Verlag (1992). LNCS 570.

[Lan88] S. Landau. *Zero-knowledge and the deparment of defense*. Notices of the american Mathematical Society, 35, 5–12 (1988).

[Lee90] J. V. Leeuwen. *Algorithms and Complexity*, tomo A de *Handbook of Theoretical Computer Science*. Elsevier Science Publishers (1990).

[Lev86] L. Levin. *Average Case Complete Problems*. SIAM Journal on Computing, págs. 285–286 (1986).

[Lin01] Y. Lindell. *Parallel coin-tossing and constant-round secure two-party computation*. Lecture Notes in Computer Science, 2139, 171–?? (2001). URL `citeseer.nj.nec.com/lindell01parallel.html`.

[LS90] D. Lapidot, A. Shamir. *Publicly verifiable non-interactive zero-knowledge proofs*. *Advances in Cryptology - Crypto '90*, eds. A. J. Menezes, S. A. Vanstone, págs. 353–365. Springer-Verlag, Berlin (1990). Lecture Notes in Computer Science Volume 537.

[Mar58] A. Markov. *On the problem of representability of matrices*. Z. Math. Logik Grundlagen Math (in Russian), 4, 157–168 (1958).

[Mer83] M. J. Merritt. *Cryptographic Protocols*. Tesis Doctoral, Georgia Institute of Technology (1983).

[MLM82] R. d. Millo, N. Lynch, M. Merritt. *The design and analysis of cryptographic protocols*. *Advances in Cryptography*, ed. A. Gersho, págs. 71–72. University of California, Santa Barbara, Santa Barbara, California, USA (1982).

[MM90] A. Mitropoulos, H. Meijer. *Zero knowledge proofs - a survey*. Inf. Téc. 90-IR-05, Queen's University at Kingston. Ontario. Canada (1990).

[Moo87] J. H. Moore. *Strong practical protocols*. *Advances in Cryptology - Crypto '87*, ed. C. Pomerance, págs. 167–174. Springer-Verlag, Berlin (1987). LNCS 293.

[MOV96] A. Menezes, P. Oorschot, S. Vanstone. *Handbook of Applied Cryptography*. CRC Press (1996).

[MR90] S. Micali, T. Rabin. *Collective coin tossing without assumptions nor broadcasting*. *Advances in Cryptology - Crypto '90*, eds. A. J. Menezes, S. A. Vanstone, págs. 253–267. Springer-Verlag, Berlin (1990). Lecture Notes in Computer Science Volume 537.

[MZV02] Y. Mu, J. Zhang, V. Varadharajan. *M out of n oblivious transfer*. *Information Security and Privacy 7th Australasian Conference, ACISP 2002*, ed. J. S. L. Batten, págs. 395–405. Springer, Melbourne, Australia (2002). Lecture Notes in Computer Science, Volume 2384.

[Nao89] M.Ñaor. *Bit commitment using pseudo-randomness (extended abstract)*. *Advances in Cryptology - Crypto '89*, ed. G. Brassard, págs. 128–137. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 435.

[NNPV02] V.Ñikov, S.Ñikova, B. Prenee, et al. *On unconditionally secure distributed oblivious transfer*. *INDOCRYPT 2002*, eds. A. Menezes, P.Sarkar, tomo LNCS 2551, págs. 395–409. Springer, Hyderabad, India (2002).

[NP01] M.Ñaor, B. Pinkas. *Efficient oblivious transfer protocols*. *Proceedings of the Twelfth Annual Symposium on Discrete Algorithms ACM/SIAM SODA 2001*, págs. 448–457. SIAM (2001).

[NS78] R. M. Needham, M. Schroeder. *Using Encryption for Authentication in Large Networks of Computers*. Communications of the ACM, págs. 993–999 (1978).

[OO89] K. Ohta, T. Okamoto. *A modification of the Fiat-Shamir scheme*. *Advances in Cryptology - Crypto '88*, ed. S. Goldwasser, págs. 232–243. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 403.

[OOF92] K. Ohta, T. Okamoto, A. Fujioka. *Secure bit commitment function against divertibility*. *Advances in Cryptology - EuroCrypt '92*, ed. R. A. Rueppel, págs. 324–340. Springer-Verlag, Berlin (1992). Lecture Notes in Computer Science Volume 658.

[Ore87] Y. Oren. *On the cunning power of cheating verifiers: Some observations about ZeroKnowledge proofs*. *Proceedings of the Twenty Eighth Annual Symposium on the Foundations of Computer Science, IEEE*, págs. 462–471 (1987).

[OS90] H. Ong, C. P. Schnorr. *Fast signature generation with a Fiat Shamir - like scheme*. *Advances in Cryptology - EuroCrypt '90*, ed. I. B. Damgård, págs. 432–440. Springer-Verlag, Berlin (1990). Lecture Notes in Computer Science Volume 473.

[OVY92] O. Ostrovsky, R. Venkatesan, M. Yung. *Secure commitment against a powerful adversary*. *Proceedings of STACS92*, págs. 439–448. Springer (1992). Lecture Notes in Computer Science Vol. 577.

[Pal96] E. Palmer. *Graphs and random graphs*. Surveys in Graph Theory, Congr. Numer, 116, 93–113 (1996).

[Pfl96] C. P. Pfleeger. *Security in Computing*. Prentice Hall (1996).

[Poi95] D. Pointcheval. *A new identification scheme based on the perceptrons problem*. *Advances in Cryptology - EuroCrypt '95*, eds. L. C. Guillou, J.-J. Quisquater, págs. 319–328. Springer-Verlag, Berlin (1995). Lecture Notes in Computer Science Volume 921.

[Pou97] G. Poupard. *A realistic security analysis of identification schemes based on combinatorial problems*. European Transactions on Telecommunications, 8, No. 5, 471–480. (1997).

[PR59] P.Erdös, A. Rényi. *On random graphs*. Publicationes Mathematicae, 6, 290–297 (1959).

[Pre00] B. Preneel. *El estado de las funciones hash. Criptología Y Seguridad de la Información*, eds. P. Caballero, C. Hernández, págs. 3–37. Ra-Ma (2000).

[PSW00] B. Pfitzmann, M. Schunter, M. Waidner. *Provably secure certified mail*. Inf. Téc. RZ 3207 (#93253), IBM Research Division (2000). URL `citeseer.nj.nec.com/pfitzmann00provably.html`.

[QGAB89] J. J. Quisquater, L. C. Guillou, M. Annick, et al. *How to explain zero-knowledge protocols to your children. Advances in Cryptology - Crypto '89*, ed. G. Brassard, págs. 628–631. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 435.

[Rab78] M. O. Rabin. *Digital Signatures*. Foundations of Secure Communication, págs. 155–168 (1978).

[Rab81] M. O. Rabin. *How to Exchange Secrets by Oblivious Transfer*. Inf. Téc. TR-81, Harvard Aitken Computation Laboratory (1981).

[Ron01] D. Ron. *Handbook of Randomization*, tomo II, cap. Property Testing (A Tutorial), págs. 597–649. Kluwer Academic Publishers (2001).

[SCGP99] A. D. Santis, G. D. Crescenzo, O. Goldreich, et al. *The graph clustering problem has a perfect zero-knowledge proof*. Information Processing Letters, 69, 201–206 (1999).

[Sch89a] C. P. Schnorr. *Efficient identification and signatures for smart cards. Advances in Cryptology - EuroCrypt '89*, eds. J.-J. Quisquater, J. Vandewalle, págs. 688–689. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 434.

[Sch89b] C. P. Schnorr. *Efficient identification and signatures for smart cards*. *Advances in Cryptology - Crypto '89*, ed. G. Brassard, págs. 239–252. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 435.

[Sch93] B. Schneier. *Applied Cryptography*. John Wiley and Sons (1993).

[Sha48] C. E. Shannon. *A mathematical theory of communication*. The Bell System Technical Journal, 27, 379–423 (1948).

[Sha79] A. Shamir. *How to share a secret?*. Communications of the ACM, 22, 612–613 (1979).

[Sha89] A. Shamir. *An efficient identification scheme based on permuted kernels (extended abstract)*. *Advances in Cryptology - Crypto '89*, ed. G. Brassard, págs. 606–609. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 435.

[Sha92] A. Shamir. *IP = PSPACE*. Journal of the ACM, 39(4), 869–877 (1992).

[Sim89] G. J. Simmons. *How to (really) share a secret*. *Advances in Cryptology - Crypto '88*, ed. S. Goldwasser, págs. 390–449. Springer-Verlag, Berlin (1989). Lecture Notes in Computer Science Volume 403.

[SRA78] A. Shamir, R. Rivest, L. Adleman. *Mental poker*. Inf. téc., MIT (1978).

[Ste93] J. Stern. *A new identification scheme based on syndrome decoding*. *Advances in Cryptology - Crypto '93*, ed. D. R. Stinson, págs. 13–21. Springer-Verlag, Berlin (1993). Lecture Notes in Computer Science Volume 773.

[Ste94] J. Stern. *Designing identification schemes with keys of short size*. *Advances in Cryptology - Crypto '94*, ed. Y. Desmedt, págs. 164–173.

Springer-Verlag, Berlin (1994). Lecture Notes in Computer Science Volume 839.

[Sti87] D. R. Stinson. *A construction for authentication/secrecy codes from certain combinatorial designs*. *Advances in Cryptology - Crypto '87*, ed. C. Pomerance, págs. 355–368. Springer-Verlag, Berlin (1987). Lecture Notes in Computer Science Volume 293.

[Ted85] T. Tedrick. *Fair exchange of secrets*. *Advances in Cryptology: Proceedings of Crypto '84*, eds. G. R. Blakley, D. Chaum, págs. 434–438. Springer-Verlag, Berlin (1985). Lecture Notes in Computer Science Volume 196.

[Tur00] A. M. Turing. *Can a machine think?*. *The World of Mathematics, Vol. 4*, ed. J.Ñewman, tomo 4, cap. XIX, 2, págs. 2099–2123. Dover (2000).

[TW87] M. Tompa, H. Woll. *Random self-reducibility and zero knowledge interactive proofs of possession of information*. *Proceedings of the 28th Annual IEEE Symposium on the Foundations of Computer Science (FOCS*, págs. 472–482 (1987).

[VL88] R. Venkatesan, L. Levin. *Random Instances of a Graph Colouring Problem are Hard*. *ACM Symposium on Theory of Computing*, págs. 217–222 (1988).

[VR92] R. Venkatesan, S. Rajagopalan. *Average case intractability of diphantine and matrix problem*. *Proc. Of the 24th Annual Symposium on Theory of Computing*, págs. 632–642. ACM Press (1992).

[Wal01] T. Walsh. *Search on High Degree Graphs*. *Proceedings of 17th International Joint Conference on Artificial Inteligence* (2001). URL `http://www.dcs.st-and.ac.uk/ apes/papers/wijcai2001.pdf`.

[Wan97] J. Wang. *Average-Case Intractable NP Problems*. *Advances in Languages, Algorithms and Complexity*, eds. D. Du, K. Ko, págs. 313–378. Kluwer Academic Publishers (1997).

[Wig01] A. Wigderson. *The digital envelope* (2001). Conferencia.

[Zal99] E.Ñ. Zalta, ed. *The Stanford Encyclopedia of Philosophy*. The Metaphysics Research Lab, Center for the Study of Language and Information, Stanford University (1999). URL `http://plato.stanford.edu/entries/logic-games/`.

[Zem91] G. Zemor. *Hash functions and graphs with large girths*. *Advances in Cryptology - EuroCrypt '91*, ed. D. W. Davies, págs. 508–511. Springer-Verlag, Berlin (1991). Lecture Notes in Computer Science Volume 547.