Un sistema de identiffcación basado en un problema indecidible

J. M. Sempere

Resumen

En el presente trabajo se propone un esquema de identificación basado en *demostraciones interactivas* [2, 4, 5] y en algunos resultados de la Teoría de la Indecidibilidad. A diferencia de otros esquemas planteados sobre problemas difíciles de resolver, la presente aproximación se basa en un problema irresoluble. Por lo tanto, la seguridad del esquema se establece no en términos de complejidad computacional sino en términos de cálculo automático.

Palabras Clave: *Base teórica, protocolos criptográficos, computabilidad.*

1 Introducción

Un *problema de decisión* es cualquier cuestión establecida en términos formales y cuyo resultado es la afirmación o negación de un enunciado o un predicado. De forma natural, podemos establecer que cualquier problema de decisión se corresponde con

un lenguaje sobre un alfabeto de codificación predefinido Σ cuyas cadenas denotan aquellas instancias del problema con una solución afirmativa. Basándonos en la teoría de la computabilidad podemos asociar a todo problema de decisión un modelo de cálculo, por ejemplo una máquina de Turing, que resuelva el citado problema. La clasificación más básica que podemos realizar en los problemas de decisión establece que los mismos se agrupan en dos clases: *indecidibles* (aquellos para los que no existe ningún algoritmo que los resuelva) y *decidibles* (aquellos que sí cuentan con un algoritmo de resolución). Desde el punto de vista de lenguajes formales, los problemas decidibles se corresponden con lenguajes *recursivos* y los problemas indecidibles con lenguajes *recursivamente enumerables no recursivos* o, simplemente, lenguajes no recursivamente enumerables.

Habitualmente, en los esquemas de identificación basados en la teoría de la complejidad (un área de estudio perteneciente al área más general de la teoría de la computabilidad) se ha trabajado con problemas recursivos con una complejidad de cálculo exponencial. Típicamente se ha trabajado con la clase de problemas $\mathcal{NP}$-completos y superiores [1, 3, 7]. La confianza depositada en los citados problemas se basa en el estado actual de la tecnología y en la asunción de algunas hipótesis no demostradas (por ejemplo, $\mathcal{P} \neq \mathcal{NP}$). En este trabajo se pretende establecer esquemas prácticos de identificación basándonos en problemas indecidibles, es decir irresolubles. Obsérvese que, a diferencia de los trabajos en $\mathcal{NP}$-completitud, aquí la única posibilidad de establecer un cambio sustancial en los problemas que se utilizan es producir un cambio profundo en el paradigma de lo que es o no computable. Tal vez el citado cambio se produzca a través de algunos modelos no convencionales de computación (i.e. computación con ADN o computación cuántica). La tecnología que permita traducir ese cambio de paradigma en una serie de herramientas útiles en la criptografía seguramente cambiará la forma de ver esta en su totalidad.

La estructura del presente trabajo es como sigue: En primer lugar proporcionaremos aquellos conceptos básicos acerca de la teoría de lenguajes formales y del Problema de Correspondencia de Post. A continuación, estableceremos dos métodos para la generación de instancias positivas (con solución) y negativas (sin solución) del citado problema. Proporcionaremos dos esquemas de demostración

para cada uno de los casos establecidos. Por último, plantearemos futuras líneas de investigación relacionadas con el presente trabajo.

2 Conceptos básicos

En primer lugar, estableceremos aquellos conceptos básicos sobre la Teoría de lenguajes formales y la computabilidad que nos serán de utilidad en la exposición del problema en el que nos basaremos para proporcionar un esquema de identificación. Las definiciones y conceptos básicos que vamos a mostrar pueden consultarse en cualquier libro sobre la materia como [6].

Un *alfabeto* Σ es cualquier conjunto finito y no vacío a cuyos elementos llamaremos *símbolos*. Una *cadena* o *palabra* sobre el alfabeto Σ será cualquier secuencia finita y ordenada de símbolos de Σ tomados con o sin repetición. Al conjunto infinito de todas las posibles cadenas de Σ le denominaremos *lenguaje universal* de Σ y lo denotaremos por Σ^*. Dadas dos cadenas $x, y \in \Sigma^*$, la operación de *concatenación* de x e y, la denotaremos por xy, dando como resultado la secuencia formada por la secuencia x seguida de la secuencia y. La cadena inversa de la cadena x es aquella obtenida a partir de x invirtiendo su secuencia de símbolos y la denotaremos por x^r. Un lenguaje sobre Σ es cualquier conjunto de cadenas definidas sobre Σ.

El problema de la correspondencia de Post

El Problema de la Correspondencia de Post (PCP) [8] se puede definir como sigue:

"*Sea* Σ *un alfabeto y las dos listas* $X = \{x_1, x_2, \ldots, x_n\}$ *e* $Y = \{y_1, y_2, \ldots, y_n\}$, *donde* $\forall i\ 1 \leq i \leq n\ x_i, y_i \in \Sigma^*$. *Se pide establecer si existe o no una secuencia finita de índices* $i_1, i_2, \ldots i_l$ *que cumpla las dos siguientes condiciones:*

1. para todo valor j, $1 \leq j \leq l$, $i_j \in \{1, 2, \ldots, n\}$

2. $x_{i_1}x_{i_2}\cdots x_{i_l} = y_{i_1}y_{i_2}\cdots y_{i_l}$"

El anterior problema fue demostrado como indecidible tal y como se enuncia en el siguiente teorema.

Teorema 1 *(Post, [8]) El Problema de la Correspondencia de Post es indecidible.*

El resultado enunciado en el anterior teorema establece que no es posible encontrar ningún algoritmo que, a partir de las listas X e Y, decida si existe o no existe un emparejamiento de secuencias de palabras con resultado idéntico. Por lo tanto, el problema PCP no admite ningún esquema de resolución computacional aunque éste disponga de recursos ilimitados.

Variaciones del PCP decidibles e indecidibles

A partir del problema PCP, como suele ser habitual en el campo de la complejidad y la computabilidad, se establecieron variaciones del problema, algunas de las cuales resultaron ser también indecidibles. A continuación comentaremos dos de ellas

- *El problema modificado PCP*

  El problema se establece en los mismos términos que el PCP salvo que la segunda condición se define como $x_1x_{i_1}x_{i_2}\cdots x_{i_l} = y_1y_{i_1}y_{i_2}\cdots y_{i_l}$. Es decir, la variación con respecto al PCP es que los emparejamientos deben comenzar con las primeras palabras de las listas. El problema establecido en estos términos permanece indecidible.

- *Listas con palabras de igual longitud*

  En este caso se añade la restricción de que en las listas X e Y se cumpla para todo valor i, tal que $1 \leq i \leq n$, $|x_i| = |y_i|$. Es decir, las palabras tienen igual longitud tomadas dos a dos. En este caso, el problema no sólo es decidible sino que es lineal

con el valor n, ya que sólo existe solución afirmativa si hay al menos un par de cadenas iguales.

3 Generación de instancias del PCP

Una vez enunciado el problema PCP, nos proponemos dar dos esquemas de generación de instancias para el mismo. Una instancia del PCP, consistirá en un par de listas X e Y *apareadas*.

Dado que el protocolo de identificación que estableceremos más tarde se basa en la demostración de la existencia o no de una solución para el PCP, deberemos generar tanto instancias positivas (esto es con solución afirmativa) como instancias negativas (esto es con solución negativa). A continuación, proporcionamos los esquemas de generación de instancias.

Instancias positivas

- **Paso 1**: Selección de un alfabeto

  El alfabeto Σ debe contener al menos dos símbolos ya que, en caso contrario, la resolución de una instancia pasa a ser decidible. Obsérvese que, con un sólo símbolo, un grupo de pares que sumen igual longitud ya es una solución de la instancia.

  En nuestro caso, bastará con tomar un alfabeto $\Sigma = \{0, 1\}$.

- **Paso 2**: Selección de una cadena generatriz

  A partir del alfabeto anterior tomamos una cadena *aleatoria* definida sobre Σ. Conviene que la cadena sea lo suficientemente larga para que no se produzcan posteriormente instancias triviales. Tomemos, por ejemplo, una longitud de 1024 caracteres cada uno de ellos seleccionados aleatoriamente. La cadena resultante la denotamos por $z = z_1 z_2 \ldots z_{1024}$

- **Paso 3**: Factorización de la cadena generatriz

  A partir de la cadena z obtenida anteriormente seleccionamos un conjunto de valores enteros entre 1 y 1024. La cardinalidad del citado conjunto, que denotaremos por I_x, la podemos establecer en 255. Cada uno de esos valores los tomamos sin repetición y de forma aleatoria. El conjunto I_x lo podemos enumerar de forma ordenada como $\{x^1, x^2, \ldots, x^{255}\}$ y, a partir del anterior conjunto, obtenemos la lista $\widehat{X} = \{x_1, x_2, \ldots, x_{256}\}$ donde definimos $x_1 = z_1 \ldots z_{x^1}$ y $x_j = z_{x^{j-1}} \ldots z_{x^j}$ y $x_{256} = z_{x^{255}} \ldots z_{1024}$.

  De forma similar, definimos un conjunto I_y y la lista $\widehat{Y}$. Es importante que se cumpla que $I_x \cap I_y = \emptyset$.

  Finalmente, obtenemos las listas X e Y a partir de una permutación aleatoria de las listas $\widehat{X}$ e $\widehat{Y}$. Obviamente, la instancia formada por X e Y tiene solución afirmativa y un posible emparejamiento es el formado por los índices $1, 2, \ldots 256$ de las listas $\widehat{X}$ e $\widehat{Y}$ ya que $x_1 x_2 \ldots x_{256} = y_1 y_2 \ldots y_{256}$.

- **Paso 4**: Adición de palabras nuevas para enmascarar la solución

  A las listas X e Y se le añaden un conjunto de palabras que formen una instancia negativa. De esta forma, resultará más difícil identificar las palabras que han formado la cadena generatriz original.

Instancias negativas

A continuación establecemos un esquema de generación de instancias negativas.

- **Paso 1**: Selección de un alfabeto

  Al igual que en el caso de instancias positivas el alfabeto Σ debe contener al menos dos símbolos. Bastará con tomar un alfabeto $\Sigma = \{0, 1\}$.

- **Paso 2**: Selección de una cadena generatriz

A partir del alfabeto anterior tomamos una cadena *aleatoria* definida sobre Σ. Conviene que la cadena sea lo suficientemente larga como para que no se produzcan posteriormente instancias triviales. Tomemos, por ejemplo, una longitud de 1024 caracteres cada uno de ellos seleccionados aleatoriamente. La cadena resultante la denotamos por $z = z_1 z_2 \ldots z_{1024}$

- **Paso 3**: Codificación de la cadena generatriz

  A partir de la cadena z obtenida en el paso 2, obtenemos una nueva cadena mediante codificación de cada uno de sus símbolos utilizando códigos no repetidos. Si la cadena z tiene una longitud n entonces utilizaremos códigos binarios de longitud $\lceil \log_2 n \rceil = k$. Así, al símbolo z_j le corresponde el código binario de j con longitud k. De esta forma, obtenemos una nueva cadena que denotaremos por z_{bin}. Por ejemplo, si la cadena $z = 01011$ entonces $z_{bin} = 0000010100111100101110$.

  Podemos tomar una longitud de cadena de 1024 y códigos de longitud $\log_2 1024 = 10$.

- **Paso 4**: Obtención de las listas X e Y

  Para construir la lista X trabajaremos con la cadena z_{bin}. La cadena z_{bin} la factorizaremos en 256 subcadenas de igual forma que hicimos en el paso 3 de instancias positivas. Para construir la lista Y trabajaremos con la cadena z^I_{bin} y la factorizaremos de igual forma que la cadena z_{bin} pero sin repetir índices. Por último, permutaremos las listas X e Y. Obsérvese que, las cadenas z_{bin} y z^I_{bin} cumplen, entre otras, las siguientes propiedades:

1. $z_{bin} \neq z^I_{bin}$
2. Cualquier prefijo de z_{bin}, de longitud mayor que k, es distinto de cualquier prefijo de z^I_{bin}
3. Cualquier segmento de z_{bin} de longitud mayor que k, es distinto de cualquier otro segmento de z^I_{bin} que ocupe la misma posición

Por lo tanto, si obtenemos las listas X e Y de forma que no coincida ningún par y que las combinaciones $x_i x_j$ excedan la longitud k, podremos demostrar que la instancia obtenida es negativa.

4 Esquema de demostración del PCP

A partir de las listas X e Y podemos establecer protocolos de identificación basados en demostraciones interactivas. Fundamentalmente el protocolo se basa en la demostración de la solución para las listas generadas. Así, si $\mathcal{A}$ quiere autentificar su identidad a $\mathcal{B}$ debe probar que sus listas tienen o no solución.

Evidentemente, el esquema de demostración es distinto en función de que las instancias sean negativas o positivas. Analizaremos cada caso por separado.

Protocolo para instancias positivas

$\mathcal{A}$ publica sus listas de forma que sean accesibles para cualquier parte $\mathcal{B}$. Para que $\mathcal{A}$ demuestre a $\mathcal{B}$ que es el poseedor de las citadas listas, se sigue la siguiente demostración

- **Paso 1**: $\mathcal{A}$ envía a $\mathcal{B}$ la cadena $(x_1 x_2 \ldots x_{256})^l$. El valor l se elige arbitrariamente. $\mathcal{A}$ envía también los índices de las cadenas $x_1, x_2, \ldots, x_{256}$ cifrados.
- **Paso 2**: $\mathcal{B}$ envía a $\mathcal{A}$ un par de posiciones (i, j) de la cadena que ha recibido en el paso 1.
- **Paso 3**: $\mathcal{A}$ envía las claves de los índices que permiten comprobar que la subcadena (i, j) está apareada.

Los pasos 2 y 3 se repiten un número suficiente de veces de forma que quede garantizada la demostración bajo un criterio probabilista.

Protocolo para instancias negativas

$\mathcal{A}$ publica sus listas de forma que sean accesibles para cualquier parte $\mathcal{B}$. Para que $\mathcal{A}$ demuestre a $\mathcal{B}$ que es el poseedor de las citadas listas, se sigue la siguiente demostración

- **Paso 1**: $\mathcal{A}$ envía a $\mathcal{B}$ la cadena z_{bin}.
- **Paso 2**: $\mathcal{B}$ envía a $\mathcal{A}$ un índice j de forma que designe uno de los elementos de X.
- **Paso 3**: $\mathcal{A}$ envía a $\mathcal{B}$ la posición de z_{bin} a partir de la que se obtiene x_j. De igual forma se comprueba la cadena y_j a partir de la cadena z^I_{bin}

Los pasos 2 y 3 se repiten un número de veces suficiente para que quede garantizada la demostración bajo un criterio probabilista.

5 Conclusiones y futuras líneas de investigación

En el presente trabajo se ha planteado la obtención de esquemas de identificación a partir de problemas indecidibles. Una ventaja inicial de este planteamiento es la inexistencia de un algoritmo de resolución genérico para el problema con el que se trabaja. Así, podemos establecer criterios de seguridad que superen a los de seguridad computacional basados en $\mathcal{NP}$-completitud.

Este trabajo pretende ser un punto de partida para una aproximación más amplia a los sistemas de identificación con *conocimiento nulo*. No obstante, como punto de partida para el posterior desarrollo de esa aproximación, deberemos abordar en futuros trabajos los siguientes aspectos:

- Los criterios de éxito en las demostraciones que se llevan a cabo en los protocolos expuestos deben ser analizadas bajo un punto de vista probabilista.

- Las implementaciones de los protocolos planteados nos darán un instrumento básico para su posterior aplicación a sistemas de identificación reales.
- Debemos formalizar el concepto de *demostración indecidible* utilizada en el presente trabajo.

Agradecimientos

El autor agradece a Pino Caballero y Candelaria Hernández las estimulantes discusiones mantenidas acerca de sistemas de demostración que han inspirado este trabajo.

Bibliografía

[1] P. Caballero y C. Hernández, Strong Solutions to the Identification Problem. En *Proceedings of 7th Annual International Conference* COCOON 2001. (J. Wang, ed.) LNCS 2108, pp 257-261. Springer. 2001.

[2] A. Fiat y A. Shamir, How to prove yourself: Practical solutions to identification and signature problems. En *Advances in Criptology*, Proceedings of CRYPTO'86. (A.M. Odlyzko, ed.) LNCS , pp 171-185. Ed. Springer. 1986.

[3] O. Goldreich, S. Micali y A. Wigderson, How to prove all NP statements in zero-knowledge and a methodology of cryptographic protocol design. En *Advances in Criptology*, Proceedings of CRYPTO'86. (A.M. Odlyzko, ed.) LNCS , pp 171-185. Ed. Springer. 1986.

[4] O. Goldreich, *Modern Cryptography, Probabilistic Proofs and Pseudorandomness*. Ed. Springer. 1999.

[5] O. Goldreich, *Foundations of Cryptography. Basic Tools*. Ed. Cambridge University Press. 2001.

[6] J. Hopcroft y J. Ullman, *Introduction to Automata Theory, Languages and Computation*. Addison-Wesley Publishing Company. 1979.

[7] C. Hernández y P. Caballero, Conocimiento nulo sobre problemas NP-completos. En *Criptología y Seguridad de la Información*, Actas de la VI Reunión Española sobre Criptología y Seguridad de la Información. (P. Caballero, C. Hernández, eds.) pp 65-72. Ed. Ra-Ma. 2000.

[8] E. Post. Recursive Unsolvability of a Problem of Thue. En *The Undecidable*. (M. Davis, ed.) Raven Press Books. 1965.

José M. Sempere
Departamento de Sistemas Informáticos y Computación
Universidad Politécnica de Valencia
Camino de Vera s/n
46022 Valencia.
E-mail: jsempere@dsic.upv.es