Implementación del Algoritmo de Descifrado para un Esquema Criptográfico de Dinámica Distribuida

Hugo Solís-Sánchez
Centro de Investigaciones en Tecnologías de la Información y Comunicación y Escuela de Física
Universidad de Costa Rica
11501-2060 San José, Costa Rica
hugo.solis@ucr.ac.cr

Elena Gabriela Barrantes
Centro de Investigaciones en Tecnologías de la Información y Comunicación y Escuela de Computación e Informática
Universidad de Costa Rica
11501-2060 San José, Costa Rica
gabriela.barrantes@ecci.ucr.ac.cr

***Abstract*— Este trabajo ha adaptado el Cifrado de Dinámica Distribuida (CDD) a un sistema caótico de baja dimensión para evaluar la debilidad y seguridad de dicha adaptación en un ejemplo realista. Esto debido a que varios autores apuntan a una deficiencia del CDD en el proceso de descifrado. En concreto, se utilizó un mapa logístico acoplado, el cual presenta múltiples atractores caóticos hecho que elimina la deficiencia del mapa logístico simple manifiesta para las aplicaciones criptográficas. Además se exploran optimizaciones al algoritmo de descifrado llegando a rendimientos apreciables para su futura escalabilidad.**

Keywords—descifrado, criptografía, llave pública, caos, dinámica distribuida, mapa logístico, mapa logístico acoplado

I. Introducción

Los sistemas caóticos tienen grandes aplicaciones potenciales en el cifrado de la información. Los sistemas criptográficos se clasifican en dos ramas: llave privada y llave pública [1]. Se han hecho múltiples esfuerzo desde las aplicaciones del caos en la parte de llave privada en comparación con la otra rama [2].

El mapa logístico es por excelencia el ejemplo de un sistema caótico. Originalmente formulado para representar un modelo demográfico simple con el fin de explicar el aumento de una población, el mapa logístico es un mapa unimodal y a consecuencia de esto su dinámica es limitada [3]. Puede expresarse usando la ecuación:

$$f(x) = \mu x(1 - x) \tag{1}$$

Donde el parámetro μ está en el intervalo $0 \leqslant \mu \leqslant 4$. El aspecto unimodal del mapa logístico hace que sea inadecuado para las aplicaciones criptográficas, aunque se han creado un número razonable de aplicaciones basados en éste [4].

Un mapa de retícula acoplado (MRA) es un sistema dinámico que modela el comportamiento de sistemas no lineales. Se utilizan principalmente para estudiar cualitativamente la dinámica caótica de los sistemas espacialmente extendidos. Esto incluye la dinámica del caos espaciotemporal donde el número de grados de libertad efectivos diverge a medida que aumenta el tamaño del sistema. Los MRA incorporan un sistema de ecuaciones (acopladas o desacopladas), un número finito de variables, un esquema de acoplamiento global o local y los términos de acoplamiento correspondientes [5].

El mapa logístico acoplado es uno de los más simples MRA, considerado en sus inicios como un modelo sencillo biológicamente realista que incorpora efectos espaciales, se basa en dos mapas logísticos acoplados por un acoplamiento lineal:

$$x_{n+1} = f(x_n) + \alpha(y_n - x_n) \tag{2}$$

$$y_{n+1} = f(y_n) + \alpha(y_n - x_n), \tag{3}$$

Donde $f(x)$ es el mapa logístico de la ecuación (1) y α es un parámetro de acoplamiento. En el mapa logístico sólo se observan dos rutas al caos (duplicación del período e intermitencia), la segunda dimensión del mapa acoplado logístico permite que la ruta cuasiperiódica ocurra [6]. Los atractores caóticos múltiples presentes en este sistema eliminan el defecto criptográfico del mapa logístico simple. Las figuras 1 y 3 muestran ejemplos de atractores caóticos para el mapa logístico acoplado (MLA).

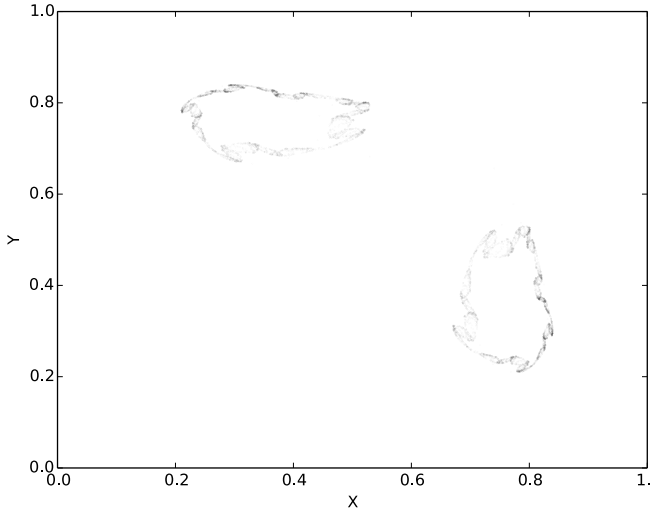

Fig 1. Atractor caótico para el mapa logístico acoplado para $\mu = 2.9$ $\alpha = 0.3314$

Este trabajo ha sido parcialmente financiado por MICITT y CONICIT FI-0204-2012 y por 834-B5-293 de la Universidad de Costa Rica por Vicerrectoría de Investigación.

Un grupo de la University of California San Diego introdujo un esquema teórico para el cifrado asimétrico que explota las propiedades de los sistemas dinámicos no lineales donde un sistema dinámico no lineal disipativo de alta dimensionalidad se distribuye entre un transmisor y un receptor, por esto se llama al método, cifrado de dinámica distribuida (CDD). La dinámica del transmisor es pública y la dinámica del receptor está oculta, no se comparte en el canal de comunicación. Un mensaje es codificado por modulación en los parámetros del transmisor, y esto resulta en un cambio del atractor general del sistema. Un receptor no autorizado no conoce la dinámica oculta en el receptor y no puede decodificar el mensaje [7].

Este trabajo tomará la propuesta teórica del CDD y la adaptada por primera vez a un sistema de baja dimensión (el MLA). Se estudiará el cifrado, el descifrado y el ataque común para este nuevo sistema criptográfico.

II. Cifrado

La idea básica del CDD es dividir un sistema dinámico de dimensión $D_T + D_R$ en dos partes con D_T variables de transmisor $t(n) = [t_1 (n); \ldots ; t_{D_T}(n)]$, y las D_R variables del receptor de $r(n) = [r_1 (n); \ldots ; r_{D_R}(n)]$. El receptor recibe la señal escalar $s_t (n)$ desde el transmisor, y el transmisor recibe la señal escalar $s_r (n)$ desde el receptor.

$$t(n+1) = F_T(t(n), s_r(n), m(n)) \quad (4)$$

$$r(n+1) = F_R(r(n), s_t(n), m(n)) \quad (5)$$

Donde $m(n)$ es el mensaje que queremos cifrar. Permitimos que $m(n)$ sólo tome valores 0 o 1, esto crea un mensaje binario.

El cifrado para nuestra implementación de baja dimensión viene de una relación entre las ecuaciones (2), (3) y (4), (5), donde ahora x (ecuación 2) será la dinámica dividida al transmisor y y (ecuación 3) será la parte del receptor, donde se sigue:

$$x_{n+1} = f(x_n) + \alpha(y_n - x_n) + A*m \quad (6)$$

$$y_{n+1} = f(y_n) + \alpha(y_n - x_n) \quad (7)$$

el parámetro A es una modulación del mensaje, en nuestra implementación A toma valores aleatorios entre 0,001 y 0,01 para proporcionar una seguridad adicional al sistema. La figura 2 muestra un mensaje de 8 bits encriptado (una letra, en este caso la W), para una fácil identificación diferenciamos los puntos que corresponden al bit 0 a los que corresponden al bit 1. La seguridad de esta implementación reside en la superposición y cercanía de esos puntos. Sólo si tiene la simulación anterior se puede descifrar el mensaje. La Figura 4 muestra un atractor diferente con un mensaje más largo de 32 bits (cuatro letras, en este caso Wort). En este esquema, un atractor caótico diferente representa un par diferente de claves criptográficas. La ecuación 2 con sus parámetros correspondientes es la llave pública y la ecuación 3 es la llave privada.

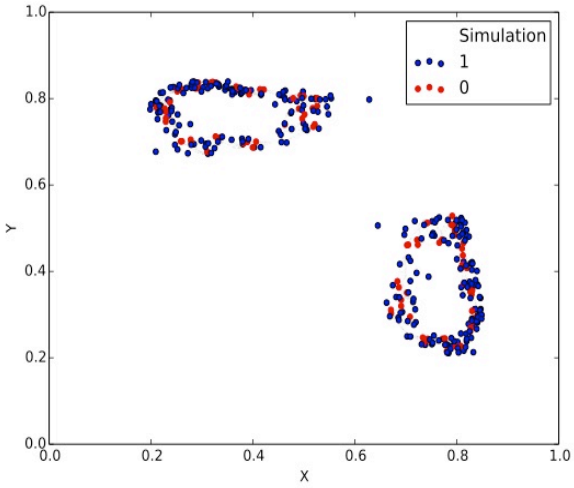

Fig 2. Mensaje cifrado sobre la dinámica totalmente conocida de la Fig 1.

III. Descifrado

Un receptor autorizado conoce todas las cantidades cifrado propuesto en la sección anterior (públicas y privadas) y puede establecer fuera de línea (antes de realizar la comunicación) los atractores permitidos, u otros aspectos dinámicos del sistema total, para todos los valores permitidos de $m(n)$. Para realizar el descifrado es necesario conocer previamente todos los puntos de la simulación. El proceso de descifrado corresponde al cálculo de la distancia euclídea de cada punto recibido a los puntos de la simulación. Si ésta distancia es menor que un parámetro de tolerancia, que está relacionado con el parámetro A de la ecuación (6) es un bit 0 y si es mayor es un bit 1.

Dicho procedimiento aunque fácil de describir es computacionalmente costoso como se muestra en la sección V de este trabajo y es donde reside la debilidad expuesta por varios autores como en [8]. En este estudio exploramos tres

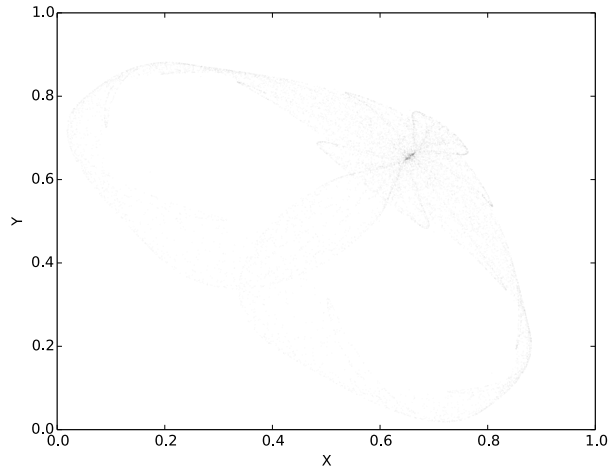

Fig 3. Otro atractor caótico para el mapa logístico acoplado para $\mu = 2.9$ $\alpha = 0.414$

formas distintas de abordar y mejorar la problemática, dicha variantes descritas a continuación.

A. Numpy

Haciendo uso de las posibilidades de Numpy de hacer cálculos de algebra lineal [9]. Se puede computar directamente la distancia euclídia sobre el arreglo de datos que contiene a los generados por la simulación.

B. GPU

La unidad de procesamiento gráfico (GPU, por sus siglas en ingles) se encuentra optimizada para hacer los cálculos de algebra lineal y ha sido de mucha ayuda para aplicaciones donde existen extensivos cálculos numéricos. Con ayuda del paquete Kivy para Python es posible sacar ventaja del GPU a través de OPENGL [10]. Con el uso de vectores se pueden computar la distancia del mismo del caso anterior con ayuda del GPU tiene un desempeño considerable para cálculos matemáticos.

C. Nuestra Propuesta

En este caso es posible repensar el algoritmo de descifrado, si los datos para x mientras se crean en la simulación se van incluyendo de forma ordenada en el arreglo de datos con su respectivo y, cuando se debe calcular la distancia euclídea se puede hacer para una ventana de cercanía de unos cientos datos sin necesidad de buscar en todo el arreglo. El tamaño de la ventana es determinada por el valor del parámetro A.

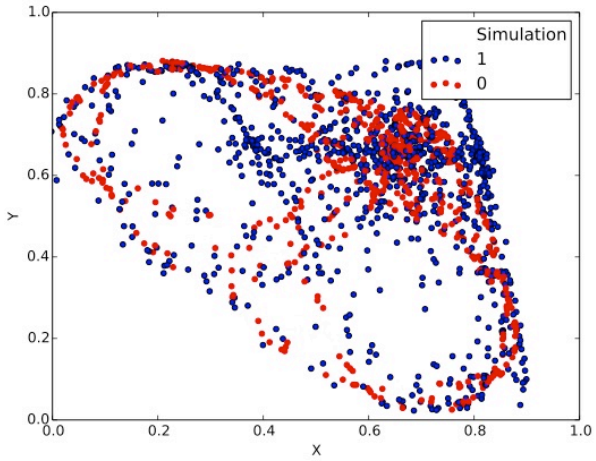

Fig 4. Mensaje cifrado sobre la dinámica totalmente conocida de la Fig 3.

IV. Criptoanálisis

Un receptor no autorizado puede intentar varios métodos para atacar CDD y descodificar el mensaje secreto $m(n)$. Uno de estos métodos consiste en reconstruir las posiciones de los atractores que corresponden a la transmisión de 0 y 1 almacenando y agrupando muestras de muchos bits transmitidos. Conocer las posiciones de los atractores permitiría al receptor no autorizado descodificar el mensaje utilizando el mismo método que el receptor autorizado [11].

La dinámica caótica puede contener ruido en el canal y o un circuito externo que harían que la dinámica sea estocástica. Un receptor no autorizado puede intentar generar un Modelo de Markov Oculto de la dinámica pública del transmisor para cada posible valor del mensaje m, y obtener una estimación de Máxima Verosimilitud (ML) del mensaje m. El mensaje decodificado será dado entonces por

$$m' = max_{m=(0,1)} p(s_t(1), \ldots, s_t(D_T+1)|m) \quad (8)$$

Para generar el modelo de Markov oculto, el receptor no autorizado necesitará cuantificar el estado del transmisor en un espacio de inserción reconstruido a partir de retardos en el tiempo y estimar las probabilidades de transición de estado así como las probabilidades de observación del modelo [8].

El ataque propuesto fue implementado para nuestro modelo de baja dimensionalidad. La figura 5 muestra cómo el caso de un mensaje cifrado usando un mapa no caótico donde la exactitud del entrenamiento es de 97% esto muestra la efectividad del ataque. Cuando el número de bits es menor que el necesario para entrenar el modelo según [7] la precisión se reduce a menos del 40% haciendo que ni aún el ataque probabilístico como se muestra en [8] pueda ayudar a este ataque en particular. La figura 6 muestra el caso en el que el número de bits es igual a necesario para realizar el ataque, es visible cómo la curva de decisión puede resolver para 0 o 1 bits.

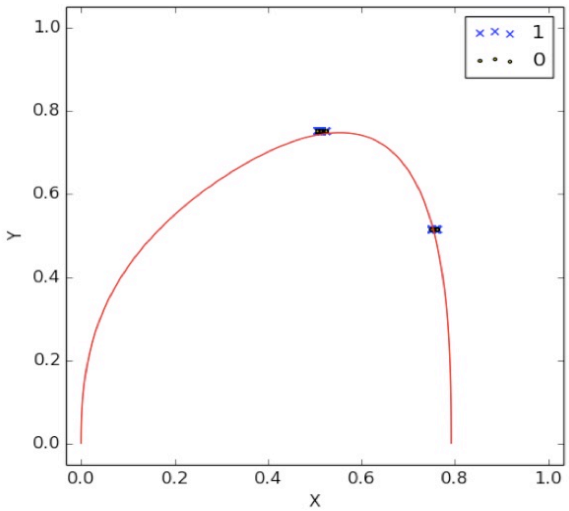

Fig 5. Superficie de decisión obtenida a partir de ataque cuando el atractor no es caótico.

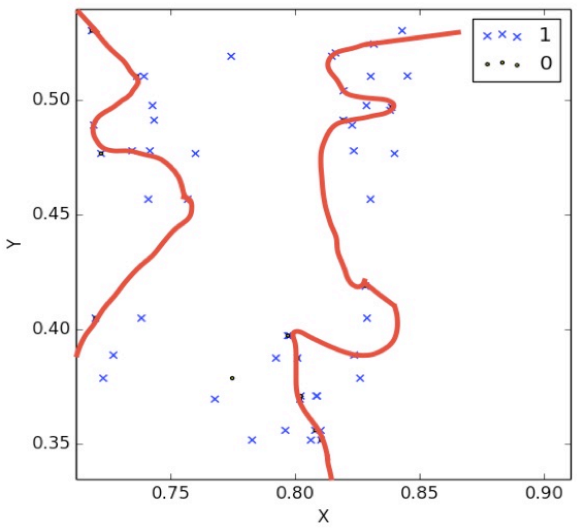

Fig 6. Superficie de decisión obtenida a partir de ataque cuando el atractor es el mismo de la Fig 1.

V. Experimento computacional

La implementación propuesta en este trabajo se ha hecho en Python con la ayuda del paquete Numpy para la gestión de datos y HMMLearn para la cadena de Markov utilizada en el ataque. Los cálculos se realizaron en un sistema Linux con un Core i7 2.6GHz PC con memoria RAM de 16 GB y gráficos por un Radeon Pro 450 con 2GB de memoria GDDR5. Las cifras de las Figuras 1 a 3 se calcularon con un millón de puntos para el atractor caótico con un tiempo medio de 33 s para el cálculo. La Figura 6 tiene el mayor tiempo de cálculo, en este caso, el entrenamiento de la cadena de Markov tomó un tiempo promedio de 72 horas debido a que se necesitan 5 millones de bits para el entrenamiento.

TABLA I. Detalles del tiempo de Cifrado/Decifrado

N bits	Cifrado tiempo (s)	Descifrado tiempo (s)	Numpy tiempo (s)	GPU tiempo (s)	Propuesta tiempo (s)
8	0.045	141.1081	70.5541	2.0990	3.0212
16	0.0997	293.1638	146.5819	4.1330	6.0222
32	0.2017	600.4974	300.2487	8.1670	9.0232
64	0.3199	1318.4119	659.2060	16.2010	16.0242
128	0.5973	2567.0675	1283.533	32.2350	32.0252
256	1.7431	5297.5988	2648.799	60.2690	60.0262
512	2.8889	10028.1301	5014.065	88.3030	89.0272
1024	4.0347	14758.6614	7379.330	116.337	117.361
2048	5.1805	19489.1927	9744.596	144.371	145.862
4096	6.3263	24219.7241	12109.86	172.405	174.363

La tabla 1 muestra la realización de los algoritmos de cifrado y descifrado donde como se espera el tiempo de ambos incrementos sobre más bits necesitan ser procesados. Es notable cómo el tiempo de cifrado es relativamente pequeño comparado con el tiempo de descifrado. La columna denominada Numpy presenta el tiempo de descifrado mejorando el manejo de los datos el computo de las operaciones, dicho aspecto aunque optimizan el tiempo de ejecución no lo hacen considerablemente, en cambio se los cálculos son hecho en el GPU como se muestran en la columna siguiente si se presenta una mejoría que hace al algoritmo de descifrado práctico para se escalamiento al uso de aplicaciones criptográficas, aunque el uso de GPU puede en si mismo presentar debilidades a la seguridad informática [12], además de que no todas las plataforma cuentan con uno.

Nuestra modificación al algoritmo mostrada en la última columna representa una clara ventaja pues su desempeño es similar al uso del GPU, si la necesidad de paquetes ni requerimientos adicionales para su implementación.

VI. Conclusiones

En este trabajo, se realizó una implementación completa del CDD, donde se describió los procesos de cifrado y descifrado, mostrando cómo la implementación de CDD es posible con un sistema dinámico de baja dimensión. Esto es relevante para este campo de investigación porque proporciona un ejemplo funcional para CDD con el mismo tipo de seguridad proporcionada por los sistemas de alta dimensionalidad. Además, este tipo de implementación abre la posibilidad de investigar mejores formas de fortalecer la eficiencia del CDD.

En la sección anterior se mostró como la mejora propuesta por este trabajo al algoritmo teórico de CDD debate las críticas de la eficiencia detrás del descifrado abriendo paso para uso en situaciones reales.

Referencias

[1] A. Menezes, P. van Oorschot, and S. Vanstone, Handbook of Applied Cryptography. CRC: Boca Raton, 1997.

[2] X. Wang, X. Gong, M. Zhan, and C. H. Lai, “Public-key encryption based on generalized synchronization of coupled map lattices” Chaos: An Interdisciplinary Journal of Nonlinear Science 15, 023109, 2005.

[3] R. L. Devaney, An introduction to chaotic dynamical systems, volume 13046. Addison-Wesley Reading,1989.

[4] D. Arroyo, J. M. Amigo, Garcia, S. Li, and G. Alvarez, “On the inadequacy of unimodal maps for cryptographic applications”, RECSI, Spain, 2010.

[5] K. Kaneko, Theory and applications of coupled map lattices, volume 159. Wiley: New York, 1993.

[6] A. L. Lloyd, “The coupled logistic map” Journal of Theoretical Biology 173, 217, 1995.

[7] R. Tenny, L. S. Tsimring, L. Larson, and H. D. Abarbanel, “Using Distributed Nonlinear Dynamics for Public Key Encryption” Physical Review Letters 90, 047903, 2003.

[8] D. Xiao, X. Liao, and S. Deng, “A novel key agreement protocol based on chaotic maps” Information Sciences 177, 1136, 2007.

[9] I. Idris, NumPy Cookbook. Packt Publishing Ltd: Birmingham, 2012.

[10] H. Solis, Kivy Cookbook. Packt Publishing Ltd: Birmingham, 2015.

[11] R. Tenny, L. Tsimring, H. Abarbanel, and L. Larson, “Security of chaos-based communication and encryption”, in Digital Communications Using Chaos and Nonlinear Dynamics, edited by L. Larson, L. Tsimring, and J.-M. Liu, pages 191–229, (Springer, New York, 2006).

[12] S. Neves and F. Araujo, “On the performance of GPU public-key cryptography”. InApplication-Specific Systems, Architectures and Processors (ASAP), 2011 IEEE International Conference on 2011 Sep 11 (pp. 133-140). IEEE.