Revista Cubana de Ciencias Informáticas
Vol. 17, No. 1, Mes Enero-Marzo, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-15

Tipo de artículo: Artículos originales
Temática: Matemática computacional
Recibido: 06/11/2022 | Aceptado: 18/12/2022 | Publicado: 28/01/2023

Ataque al PRESENT-80 con el Algoritmo Genético mediante aproximaciones sucesivas de componentes fijas

Attack to PRESENT-80 with the Genetic Algorithm using successive approximations of fixed components

Adrian Donatien Charón 0000-0002-0441-8282[1*]
Miguel A. Borges Trenard 0000-0003-3364-9677[2]
Mijail Borges Quintana 0000-0002-5656-0037[3]

[1*]Departamento de Matemática, Universidad de Oriente, Santiago de Cuba, Cuba. 90500. adriand@uo.edu.cu
[2]Doctorate in Mathematics Education, University Antonio Nariño, Bogotá, Colombia. borgestrenard2014@gmail.com
[3]Departamento de Matemática, Universidad de Oriente, Santiago de Cuba, Cuba. 90500. mijail@uo.edu.cu

[*]Autor para correspondencia: (adriand@uo.edu.cu)

RESUMEN

Con el surgimiento y desarrollo de la llamada internet de las cosas y la necesidad del uso criptográfico en dispositivos de relativamente pocos recursos, es cada vez más creciente el análisis y criptoanálisis de la familia de los cifrados en bloques ligeros. El cifrado PRESENT es un representante bien conocido de esa familia. En los últimos años se ha incrementado el uso que se le ha dado al Algoritmo Genético (AG) en el criptoanálisis a cifrados en bloque, no obstante, todavía es necesario seguir profundizando en su estudio. Por otra parte, el Algoritmo Genético es un método de optimización basado en búsquedas heurísticas en el espacio de las posibles soluciones, el cual intenta simular el comportamiento de la evolución de las especies. Buscar la clave en todo el espacio de las claves es un problema complejo y puede resultar imposible. En el presente

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 1, Mes Enero-Marzo, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-15

trabajo, se muestra un criptoanálisis al PRESENT-80, utilizando el Algoritmo Genético con una metodología de reducción sucesiva del conjunto de las posibles soluciones mediante un procedimiento de fijar componentes iterativamente.

Palabras clave: criptoanálisis, cifrados en bloques, algoritmo genético, cifrado PRESENT-80.

ABSTRACT

With the emergence and development of the so-called Internet of Things and the need for cryptographic use in devices with relatively few resources, the analysis and cryptanalysis of the family of light block ciphers is increasingly growing. The PRESENT block cipher is a well-known representative of that family. In recent years, the use that has been given to the Genetic Algorithm (GA) in the cryptanalysis of block ciphers has been increasing, however, it is still necessary to continue studying it in depth. On the other hand, the Genetic Algorithm is an optimization method based on heuristic searches in the space of possible solutions, which tries to simulate the behavior of the evolution of species. Searching for the key in the entire key space is a complex problem and can become impossible to solve. In the present work, a cryptanalysis of the PRESENT-80 is shown, using the Genetic Algorithm with a metodology of progresive reducing the set of possible solutions by means of a procedure of iterative fixing components.

Keywords: cryptanalyisis, block ciphers, genetic algorithm, PRESENT-80 block cipher.

Introducción

Existen varios algoritmos heurísticos que son utilizados como métodos de optimización y predicción en el contexto de la Criptografía; en (Grari et al., 2019), se utiliza el método heurístico (ACO) (Ant Colony Optimization), y fue probada una metodología con el cifrado en bloque S-AES (Simplified Advanced Encryption Standard), estudiando dos pares de textos claros y cifrados.

En (Ali and Jawad, 2020), se emplea una combinación del Algoritmo Genético (AG) y (ACO) para el criptoanálisis de cifrados en flujo. En (So, 2020), (Lee et al., 2020), fueron mostradas las posibilidades de combinación y diseño de estos algoritmos, analizadas mediante las herramientas de machine learning y deep lear-

ning. En (Kramer, 2017) los autores diseñan e implementan un algoritmo de cifrado simétrico cuya estructura interna se basa en el Algoritmo de Búsqueda Tabú.

En los últimos años es cada vez más creciente el uso que se le ha dado al Algoritmo Genético AG en el criptoanálisis a cifrados en bloque. El marco de los AGs es direccionado a problemas de criptoanálisis. En (Tortella et al., 2016) se investiga el uso del AG para generar claves, en particular, solo aquellas claves que proporcionan una señal cifrada con una taza de error de bit suficientemente alta, y lo contrario para la señal descodificada. Los resultados mostraron que el AG genera claves aceptables en el contexto de la Criptografía. Una breve panorámica sobre algunas ideas claves de criptoanálisis que usan el AG, y han sido desarrolladas en los últimos 15 años, puede encontrarse en (Baragada and Reddy, 2013). Por su parte, en (Khan et al., 2015) se realiza un interesante estudio sobre las bases de datos en las que se encuentran referencias sobre el AG en el criptoanálisis. De la lectura de (Baragada and Reddy, 2013) y (Khan et al., 2015) puede observarse que el criptoanálisis basado en el AG es un área de investigación aun emergente y a su vez promisoria.

Con el uso del AG y la necesidad de disminuir la complejidad de recorrer todo el espacio de las claves, surge la idea de dividir dicho espacio y modificar los algoritmos para que sean capaces de mantener la búsqueda en una región del mismo. En este sentido en el artículo (Brown et al., 2009) se puede apreciar una idea al respecto: los autores seleccionan un pequeño grupo de claves el cual dividen en varios subgrupos de 3, 4 y hasta 5 claves, con el objetivo de determinar cuál grupo es más débil con respecto a determinados criterios que ellos tienen en cuenta. Ideas más formales aparecen en (Borges-Trenard et al., 2019) donde se propone un procedimiento para particionar el espacio de las claves basándose, en cierto sentido, en el uso de congruencia aritmética y el algoritmo de la división con resto en los enteros y (Tito-Corrioso et al., 2019) particiona el espacio de las claves en clases de equivalencia, usando Teoría de Grupos, y así concentrar el ataque sobre estas clases, el objetivo de ambos es dividir el espacio en intervalos para luego orientar el ataque en uno de estos intervalos. En (Tito-Corrioso et al., 2021; Tito-Corrioso, 2021) se nombran éstas metodologías como BBM (Borges-Trenard et al., 2019) y TBB (Tito-Corrioso et al., 2019) respectivamente y se realiza el estudio de varios parámetros del AG en conjunto con la utilización de las mismas.

La idea de utilizar la congruencia aritmética en el criptoanálisis resulta interesante porque permite abordar problemas complejos con herramientas sencillas. Esta técnica también se utiliza en otros trabajos, en (Valluri, 2018) el autor demuestra que un determinado protocolo de intercambio de claves públicas se puede romper con un ataque man-in-the-middle, que se basa (en principio) en trabajar con congruencias aritméticas.

Como es conocido, el espacio de claves del cifrado en bloque PRESENT es relativamente grande, para evitar un ataque de fuerza bruta, por lo tanto, el AG tiene que enfrentarse con la dificultad de buscar en este espacio y, aunque la búsqueda de AG es heurística, el tamaño del espacio de claves debe tenerse en cuenta. En (Borges-

Trenard et al., 2019) se realiza una partición del espacio de claves, de modo que se pueda enfocar el ataque solo en algunos de los subconjuntos de la partición, por lo tanto, de esta manera es posible reducir el conjunto de claves por investigar.

Buscar la clave en todo el espacio de las claves es un problema complejo y puede llegar a ser imposible de resolver en la práctica, en este sentido, el objetivo de este trabajo es realizar un ataque al PRESENT-80 mediante el (AG), con una metodología de particionamiento del espacio claves de una forma más general que la que se utiliza en (Borges-Trenard et al., 2019) y (Tito-Corrioso et al., 2021).

Métodos o Metodología Computacional

El Algoritmo Genético

El Algoritmo Genético AG es un método de búsqueda heurística que ha demostrado su efectividad en múltiples campos ver (Lambora et al., 2019), (Katoch et al., 2021). En términos generales, el AG toma como punto de partida una población inicial, conjunto de individuos que representan posibles soluciones, y mediante sucesivas iteraciones (generaciones, en el argot de esta teoría) intenta obtener la solución del problema que se esté investigando. Elementos típicos asociados con el AG son, m la cantidad de elementos en la población inicial, g el número de generaciones que se recorrerá, F la función de aptitud que se utilizará (las funciones de aptitud miden cuánto se acercan los individuos de la población muestreada a la solución), el tipo de cruzamiento *cru* y mutaciones *mut* que se realizarán en cada iteración, s la cantidad de individuos que se cruzarán, así como las probabilidades para cruzamiento y mutación. Saber elegir convenientemente esos parámetros y otros por el estilo forma parte de la pericia que debe lograrse para una exitosa aplicación del Algoritmo Genético. Los interesados en profundizar en este algoritmo pudiesen ver (Bodenhofer, 2003) para una primera lectura y (Kramer, 2017) para ulterior profundización.

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 1, Mes Enero-Marzo, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-15

Algorithm 1 Algoritmo Genético.

1: Generar una población inicial de m individuos de forma aleatoria:
2: **while** no se encuentre la solución o no se llegue a las g generaciones **do**
3: **Seleccionar** s progenitores:
4: **Cruzar** los s progenitores seleccionados y generar parejas de descendientes:
5: **Mutar** cada uno de los descendientes resultantes:
6: **Seleccionar** los individuos de la próxima generación:
7: **end while**
8: **return** El individuo con mayor aptitud.

Cifrado en bloque ligero PRESENT

El cifrado PRESENT es una red de substitución permutación con 31 rondas, soporta claves de 80 y 128 bits (de ahí las clasificaciones PRESENT-80 y PRESENT-128) (Bogdanov et al., 2007), llamado comunmente el paradigma de la FCBL (Familia de los Cifrados en Bloques Ligeros). La longitud de bloques es de 64 bits. Cada ronda inicia con una suma XOR del bloque de estado con la clave de la ronda correspondiente (estas claves de ronda son generadas por el algoritmo a partir de la clave que introduce el usuario), posteriormente se pasa en paralelo al bloque que se está procesando por una S-caja de 4 bits (16 realizaciones simultáneas de la S-caja sobre el bloque de 64 bits en proceso) y finalmente se procede a una transposición de las componentes del bloque de estado obtenido. Para más detalles, la fuente original sigue siendo la mejor referencia (Bogdanov et al., 2007). Atacar al PRESENT de manera exitosa es un desafío actual, ver (Pandey et al., 2019), (Gunathilake et al., 2021).

Resultados y discusión

El tipo de ataque realizado es a texto claro conocido, para profundizar ver (Borges-Trenard et al., 2019) .
Sea $E : \{0,1\}^m \times \{0,1\}^n \rightarrow \{0,1\}^n$ un cifrado en bloque, donde m es la longitud de clave y n la longitud de bloque. Sea T un texto claro, K es la clave y C el correspondiente texto cifrado, $C = E(K,T)$. Se dice que K' es una clave consistente E, T y C si $C = E(K',T)$. Denotemos con $Cons_E$ el conjunto de claves consistentes con E, T y C.

El problema que se intenta resolver en este trabajo es, **dado** E, T y C, **calcular** una clave en $Cons_E$.

Método de fijar componentes para el Algoritmo Genético

En el caso del criptoanálisis a cifrados en bloque la población está dada por el conjunto de claves, que consisten en cadenas binarias de longitud k_1, donde k_1 es la longitud de clave del cifrado. Sea k_2 la longitud de clave grupal (Borges-Trenard et al., 2019), que representa la longitud de los individuos de la población sobre los que realmente trabaja el AG. El método de fijar componentes consiste en fijar el valor de $k_1 - k_2$ componentes de las cadenas binarias de longitud k_1. Sean $I = \{i_1, \ldots, i_{k_1-k_2}\}$ dichas componentes (posiciones) y $\rho_1, \ldots, \rho_{k_1-k_2}$ los valores 0-1 correspondiente a cada una de las componentes.

La clave está en que todos los procedimientos del AG trabajan sobre los individuos de longitud k_2 y donde único se necesita trabajar con sus representaciones como cadenas de longitud k_1 es para el cálculo de la aptitud. En este caso, dado un individuo Z de longitud k_2, el correspondiente elemento en el espacio de longitud k_1 sería $Z' = ext(Z, I)$, que consiste en extender Z a Z' teniendo en cuenta que en las componentes del conjunto I los valores serían los ρ_i's, mientras que en las k_2 componentes restantes los valores de Z' serían los de Z en orden creciente de las componentes de izquierda a derecha.

Si la longitud de la clave es k_1 entonces el espacio clave K tiene cardinal 2^{k_1} y existe una correspondencia uno a uno entre K y los números enteros en el intervalo $[0, 2^{k_1} - 1]$. En este hecho se basan las metodologías BBM y TBB (Tito-Corrioso et al., 2021). La función de extensión en estos casos, de la clave grupal a la clave real, es la función *num*, $num(k_2, q, r) = q \cdot 2^{k_2} + r$. Donde la clave grupal Z está representada por el resto r (en BBM) o el cociente q (en TBB). Ambas metodologías tienen un enfoque dual, en el caso de BBM se fijan todas componentes correspondientes al cociente q, mientras que en el caso de TBB se fijan todas las componentes correspondientes al resto r. En BBM se trata de elementos que se encuentran, en su representación decimal, de forma consecutiva en un intervalo de longitud 2^{k_2}, mientras que en el caso de TBB es una cantidad de $2^{k_1-k_2}$ elementos que se encuentran distribuidos a lo largo del intervalo de números enteros $[0, 2^{k_1} - 1]$, los cuales dejan el resto r en la división por 2^{k_2}.

Tanto BBM como TBB constituyen casos particulares del método general de fijar componentes que se expone en este trabajo.

Revista Cubana de Ciencias Informáticas
Vol. 17, No. 1, Mes Enero-Marzo, 2023
ISSN: 2227-1899 | RNPS: 2301
http://rcci.uci.cu
Pág. 1-15

Criptoanálisis a PRESENT-80 mediante el Algoritmo Genético.

En el PRESENT el espacio de la clave es demasiado grande para una búsqueda exhaustiva, también podría ser incluso para una forma de búsqueda heurística.

Se considera para los experimentos que se ha podido hacer una primera partición con $k_2 = 44$, utilizando BBM, donde en ese mismo intervalo se encuentra la clave. Esto reduce el espacio de búsqueda del AG a una longitud de 2^{36}. Entonces a partir de este punto se evaluará comenzar a fijar sucesivamente cierto número de componentes con mayor frecuencia en las soluciones obtenidas en un número de corridas realizadas en cada paso. La idea consiste en ir disminuyendo el espacio de las claves posibles. En trabajos anteriores (Borges-Trenard et al., 2019; Tito-Corrioso et al., 2021) se obtuvo buenos resultados aplicando el AG en subconjuntos del espacio de claves de tamaño 2^{15} y 2^{16}, lo que fundamentalmente con cifrados de longitud de bloque hasta 2^{48} como el $AES(3)$ de la familia $AES(t)$ (Monier-Columbié, 2018; Tito-Corrioso, 2021).

La población del AG en este caso está formada por individuos que consisten en bloques de 80 bits (K). El tamaño escogido para la población en cada generación del algoritmo es $m = 800$. Se realizará un máximo de $g = 50$ iteraciones (generaciones) por experimento, a cada resultado del AG se le llamará corrida del AG. Como función de aptitud se toma la cantidad de ceros en la diferencia entre el texto cifrado C y el valor $E(T,X)$, donde $E(T,X)$ es el texto cifrado de T con la clave X que se esté muestreando, todo ello dividido por la longitud de clave (80 en este caso). De ese modo, la función de aptitud da una medida de cuán cercano, según la distancia de Hamming (cantidad de componentes iguales), puede estar el cifrado de T mediante X del cifrado del mismo T mediante la clave buscada K. En cada generación se cruzan 400 individuos de la población de 800. Como función de cruzamiento se utiliza el cruzamiento por 2 puntos, mientras que la mutación mueve hasta 2 posibles componentes del individuo mutante. Las probabilidades de cruzamiento y mutación son respectivamente 0,9 y 0,1.

La instrumentación del AG se realizó en el lenguage de programación del sistema de Álgebra Computacional denominado MAPLE (https://www.maplesoft.com), los programas en MAPLE de aplicación específica del AG para el ataque a cifrados en bloques así como la programación del cifrado PRESENT fueron elaborados por los autores de este trabajo. Se trabajó sobre un CPU Intel(R), Core^{TM}, i5-M 460, a 2.53GHz, con RAM de 8GB y sistema operativo de 64 bits.

Se llevaron a cabo un total de 60 experimentos en una primera etapa, organizados en 6 iteraciones, en cada

iteración se realizaron 10 corridas del AG. En la primera iteración con una longitud de clave $K = 80$, y la clave grupal $k_2 = 36$ (se están fijando inicialmente 44 componentes), con la cantidad de individuos m y las generaciones g, el AG en cada corrida genera una cantidad a lo sumo de 40000 individuos (que representan posibles claves), el espacio de las claves es de $2^{36} = 68719476736$, esto representa 0,0000582076 por ciento de todas las posibles claves, en ninguna de los diez corridas que se hicieron con esa clave grupal se encontró la clave, se hizo un análisis de frecuencia con los resultados de los individuos más aptos obtenidos y se fijaron las tres componentes con mayor frecuencia para la próxima iteración, en esta iteración la aptitud promedio fue de 0,71.

Se debe destacar que dentro de las componentes de mayor frecuencia hay una mayoría que coinciden con los valores de la clave en esas componentes, no obstante, no en todas ocurre así. En este caso, para evaluar la convergencia del proceso, las componentes que se fijan para pasar a la siguiente iteración se toman de forma tal que coincidan con la clave (ver Tabla 1), de este modo se asegura que en la nueva partición se encuentra la clave. En la práctica esto consistiría en un proceso de prueba y error, donde el tomar relativamente pocas componentes en cada paso y entre las de mayor frecuencia, garantizan mayor garantía de éxito.

Para pasar a la segunda iteración, se fijan 3 nuevas compomentes (47 componentes fijas), por lo que la clave grupal es $k_2 = 33$, el espacio de las claves se redujo a $2^{33} = 8589934592$, considerando un máximo de 40000 individuos que genera cada corrida del AG, esto representa el $0,00004656612$ por ciento de todas las posibles claves. En ninguno de las diez corridas que se hicieron se encontró la clave, se hizo un análisis de frecuencia con los resultados de los 10 individuos más aptos que se obtuvieron y se fijaron las 5 componentes con mayor frecuencia para la próxima iteración, la aptitud promedio fue 0,73.

En la tercera iteración, se tiene que la clave grupal es $k_2 = 28$, el espacio clave se redujo a $2^{28} = 268435456$, por lo que cada corrida del AG recorre el 0,0149011611 por ciento de todas las posibles claves. En ninguno de los diez experimentos que se hicieron se encontró la clave, se hizo un análisis de frecuencia con los resultados de los 10 individuos más aptos que se obtuvieron y se fijaron las 3 componentes con mayor frecuencia para la próxima iteración, la aptitud promedio fue de 0,71.

En la cuarta iteración, la clave grupal es $k_2 = 25$, el espacio clave se redujo a $2^{25} = 33554432$ esto representa el 0,1192092896 por ciento de todas las posibles claves, aquí vemos como el espacio clave se va reduciendo y esto nos puede ayudar considerablemente a encontrar la clave. En ninguno de los diez experimentos que se hicieron se encontró la clave, se hizo un análisis de frecuencia con los resultados de los 10 individuos más

aptos que se obtuvieron y se fijaron las 5 componentes con mayor frecuencia para la próxima iteración, la aptitud promedio fue 0,71.

Ni en la quinta (2^{20} longitud del espacio de claves), ni en la sexta iteración (2^{15} longitud del espacio de claves) se encontró la clave y la aptitud estuvo muy baja, esto provocó que se revisara la ejecución de la instrumentación del AG, debido a que en la sexta iteración el espacio clave es de $2^{15} = 32718$ y el AG debe generar en cada corrida 40000 individuos (que representan claves posibles), esto significa que en el algún momento debía de encontrarse la clave con un porciento de éxito elevado del total de corridas. Al revisar los resultados parciales del programa, se observó que la longitud de clave y la constitución del PRESENT hacen que sea frecuente que caiga en óptimos locales y que la cantidad de individuos diferentes en la población tienda a disminuir y por ende que no resulte cierto asumir que en cada corrida del AG se generen aproximadamente 40000 individuos. Esto conllevó a realizar varios cambios en el programa.

Transformaciones realizadas al programa del AG

Lo primero que se realizó estuvo motivado por la detección de que la población en cada generación tendía a reducirse rápidamente la cantidad de individuos diferentes, en buena parte motivado por la repetición muchas veces del individuo más apto encontrado, entonces mediante el proceso de selección y posterior cruzamiento estos iban multiplicando este efecto y haciendo que la población pudiera llegar a ser incluso de un solo individuo.

Se han tenido en cuenta recomendaciones sobre el manejo de la población y los óptimos locales dadas en (LOZANO and HERRERA, 2010; Eyal, 2020). En (Tito-Corrioso, 2021) se analiza el comportamiento de varios parámetros del AG con el uso de las metodologías BBM y TBB, donde se pueden estimar por ejemplo, valores adecuados para la longitud de la clave grupal y su relación con el tiempo disponible o deseado, pero todo ello requiere de mantener en cada generación del AG una población constituida por una cantidad de individuos diferentes similar al tamaño de la población.

Algunos principios que se han establecido:

1. Tratar de que en cada generación la población se mantenga con un tamaño cercano a *m* atendiendo a los individuos que puedan ser diferentes y a los procesos que intervienen, Cru, Mut, y Select (nueva

población).

2. El método de los torneos se dejó solo a torneo entre 3 ($t = 2$). Ya que esto estaba también influyendo en que se redujera el tamaño de la población. Este método se utiliza para la selección de los progenitores para el cruzamiento y para la selección de la población de la próxima generación a partir de los individuos obtenidos en los diferentes procesos del AG.

3. Al recibir la población P de una generación, para crear la población de la siguiente generación:

   a) Se seleccionan de P el conjunto de los padres (progenitores).

   b) Se obtienen los Q_1 descendientes por Cru.

   c) Se mutan los individuos de P y Q_1 y se obtiene Q_2.

   d) Con P, Q_1 y Q_2 se obtiene la lista solo con los diferentes y se completa la población con elementos aleatorios hasta que la cantidad sea $2m$. Pueden repetirse en ese caso los de la generación aleatoria. Esto de completar $2m$ se hace para que la selección de los m individuos de la nueva población tenga más posibilidades de transformarse con respecto a la generación anterior, mientras más cercana a m fuese la cantidad de individuos, mayor posibilidad de estancamiento puede haber ya que sería obtener m elementos de casi m elementos, o sea de los mismos o incluso menos. La selección de los individuos se realiza por el método de torneos entre 3.

   e) Luego se ordena la población de acuerdo a la aptitud de los individuos y se hace el control correspondiente para evitar el estancamiento.

Hay dos fenómenos típicos en el Algoritmo Genético, uno de ellos es el carácter oscilante de la función de aptitud, la misma puede incrementar que disminuir a lo largo de las generaciones. El otro fenómeno es el estancamiento alrededor de un valor de la función de aptitud, en la literatura esto se asocia con que alrededor de la región explorada se alcanzó un óptimo local. En el presente trabajo se eludió estos fenómeno, utilizando técnicas elitistas, es decir, en cada generación se logra que una parte de los mejores permanezca para la próxima iteración y se combine con otros seleccionados al azar, pero por el Método de los Torneos (que también conlleva a cierto elitismo). Por otra parte, si un individuo de mayor aptitud prevalece de una generación a la siguiente, este se salva a una lista de los individuos más aptos obtenidos y se sustituyen todas las ocurrencias de ese individuo en la población por otros generados de forma aleatoria. Si en algún momento se obtiene un individuo de aptitud 1, el AG termina y devuelve los datos correspondientes a ese individuo.

Con las transformaciones hechas al programa se ejecutó una segunda etapa con los mismos parámetros para el AG, repitiendo las iteraciones 5 ($k_2 = 20$) y 6 ($k_2 = 15$).

En la iteración 5, recordando que el AG genera en cada corrida unos 40000 individuos, y el tamaño del espacio de claves es igual a $2^{20} = 1048576$, lo que representa el 3,814697266 por ciento de todas las posibles claves. Con estos parámetros de las 10 corridas que se hicieron en 2 se encontró la clave y la aptitud promedio aumentó a 0,779.

En la iteración 6, que ya se conoce que cada corrida del AG puede generar 40000 individuos, que en este caso es una cantidad que supera el tamaño del espacio de las claves ($2^{15} = 32768$), se encontró la clave en las 10 corridas, pero además en la mayoría de las corridas no se llegó a realizar 25 generaciones de un total de 50.

A continuación se muestra en dos tablas el resumen del análisis de frecuencia de las componentes en las 6 iteraciones de la primera etapa y las dos repetidas en la segunda etapa. En las tablas, CG significa clave grupal y AP aptitud promedio. En la tercera columna de las tablas se ponen solo aquellas componentes que coinciden con el valor de la clave en esas componentes y se especifica el orden que ocupan según la frecuencia.

Tabla 1 - Resumen de la primera etapa antes de las tranformaciones realizadas al (AG).

Iteración	CG	Componentes con mayor frecuencia que coinciden con la clave	AP
1	$k_2 = 44$	1ra 43 (0.9), 2da 48 (0.9), 3ra 77 (0.9)	0.71
2	$k_2 = 47$	1ra 45 (0.9), 2da 62 (0.8), 3ra 64 (0.7), 5ta 67 (0.7), 6ta 69 (0.7)	0.73
3	$k_2 = 52$	3ra 46 (0.7), 4ta 59 (0.7), 5ta 63 (0.7)	0.68
4	$k_2 = 55$	3ra 44 (0.7), 7ma 57 (0.7), 8va 66 (0.7) , 9na 68 (0.7), 10ma 70 (0.7)	0.71
5	$k_2 = 60$	1ra 58 (0.8), 3ra 42 (0.7), 4ta 51 (0.7) , 5ta 53 (0.7), 7ma 61 (0.7)	0.75
6	$k_2 = 65$	2da 43 (0.9), 4ta 75 (0.8), 5ta 52 (0.7) , 6ta 56 (0.7), 7ma 71 (0.7)	0.71

Tabla 2 - Resumen de la segunda etapa después de las tranformaciones realizadas al (AG).

Iteración	CG	Componentes con mayor frecuencia que coinciden con la clave	AP
1	$k_2 = 60$	1ra 60 (0.91), 2da 71 (0.91), 3ra 74 (0.91) 4ta 41 (0.83) 5ta 42 (0.83)	0.79
2	$k_2 = 65$	En todas las corridas se encontró la clave	1

De la observación de las dos tablas se puede resumir que para la segunda etapa, luego de resolver el problema con el estancamiento, se logra que, para $k_2 = 60$, la aptitud promedio aumenta, la frecuencia de las componentes que coinciden con la llave aumenta, al punto de que las 5 primeras de mayor frecuencia en las corridas coinciden con la clave.

Conclusiones

Con este trabajo se logra mostrar que el cifrado PRESENT-80 puede ser vulnerable a un ataque mediante el Algoritmo Genético con aproximaciones sucesivas de componentes fijas, queda aún mucho por hacer en esta dirección, PRESENT-80 es realmente un cifrado difícil de romper, no obstante, el resultado que aquí se presenta es esperanzador.

Fijar como conocidas algunas componentes de la clave pudiese contribuir a una mayor efectividad del AG y es una forma de proceder relativamente natural en el escenario del criptoanálisis.

Este trabajo contribuye a la obtención de mayor información cuando se disminuye la longitud de la clave en movimiento o clave grupal (incremento del número de componentes fijas), lo cual permite realizar búsquedas posteriores reduciendo nuevamente el espacio de búsqueda haciendo uso de la información obtenida. Este ataque podría ser especialmente adecuado para otros cifrados en bloque y también para combinarlo con otros métodos de ataque.

Las medidas adoptadas para evitar el estancamiento de la población que genera el AG y mantener una cantidad de individuos diferentes similar al tamaño de la población, hicieron posible mejorar sustancialmente los resultados de los experimentos realizados con el cifrado PRESENT-80, y pueden constituir procedimientos adecuados a ser utilizados en otros escenarios con la utilización del AG.

Referencias

Faez Hassan Ali and Riyam Noori Jawad. Using evolving algorithms to cryptanalysis nonlinear cryptosystems. *Baghdad Science Journal*, 17(2 Special Issue NICST), 2020.

SambasivaRao Baragada and P Satyanarayana Reddy. A survey of cryptanalytic works based on genetic

algorithms. *International Journal of Emerging Trends & Technology in Computer Science (IJETTCS)*, 2 (5):18–22, 2013.

Ulrich Bodenhofer. Genetic algorithms: theory and applications. *Lecture notes, Fuzzy Logic Laboratorium Linz-Hagenberg, Winter*, v. 2004, 2003.

Andrey Bogdanov, Lars R Knudsen, Gregor Leander, Christof Paar, Axel Poschmann, Matthew JB Robshaw, Yannick Seurin, and Charlotte Vikkelsoe. Present: An ultra-lightweight block cipher. In *International workshop on cryptographic hardware and embedded systems*, pages 450–466. Springer, 2007.

Miguel Á Borges-Trenard, Mijail Borges-Quintana, and Lázaro Monier-Columbié. An application of genetic algorithm to cryptanalysis of block ciphers by partitioning the key space. *Journal of Discrete Mathematical Sciences and Cryptography*, 25(2):325–334, 2019. doi: 10.1080/09720529.2019.1649028.

Joseph Alexander Brown, Sheridan Houghten, and Beatrice Ombuki-Berman. Genetic algorithm cryptanalysis of a substitution permutation network. In *2009 IEEE Symposium on Computational Intelligence in Cyber Security*, pages 115–121. IEEE, 2009.

W Eyal. Hands-on genetic algorithms with python. Packt Publishing Ltd., Birmingham, UK, 2020.

Hicham Grari, Ahmed Azouaoui, and Khalid Zine-Dine. A cryptanalytic attack of simplified-aes using ant colony optimization. *International Journal of Electrical & Computer Engineering (2088-8708)*, 9(5), 2019.

Nilupulee A Gunathilake, Ahmed Al-Dubai, William J Buchanan, and Owen Lo. Electromagnetic analysis of an ultra-lightweight cipher: Present. *arXiv preprint arXiv:2106.15225*, 2021.

Sourabh Katoch, Sumit Singh Chauhan, and Vijay Kumar. A review on genetic algorithm: past, present, and future. *Multimedia Tools and Applications*, 80(5):8091–8126, 2021.

Asif Hameed Khan, Auqib Hamid Lone, and Firdoos Ahmad Badroo. The applicability of genetic algorithm in cryptanalysis: A survey. *International Journal of Computer Applications*, 130(9):42–46, 2015.

Oliver Kramer. Genetic algorithms. In *Genetic algorithm essentials*, pages 11–19. Springer, 2017.

Annu Lambora, Kunal Gupta, and Kriti Chopra. Genetic algorithm-a literature review. In *2019 international conference on machine learning, big data, cloud and parallel computing (COMITCon)*, pages 380–384. IEEE, 2019.

Ting Rong Lee, Je Sen Teh, Jasy Liew Suet Yan, Norziana Jamil, and Wei-Zhu Yeoh. A machine learning approach to predicting block cipher security. In *Universiti Putra Malaysia*, pages 122–132, 2020.

M LOZANO and F HERRERA. Sf1. computación evolutiva y algoritmos bioinspirados. 2010. URL http://sci2s.ugr.es.

L Monier-Columbié. Criptoanálisis algebraico a cifrados en bloques ligeros. Tesis de Maestría en Ciencias Matemáticas. Universidad de La Habana, Cuba, 2018.

Jai Gopal Pandey, Tarun Goel, and Abhijit Karmakar. Hardware architectures for present block cipher and their fpga implementations. *IET Circuits, Devices & Systems*, 13(7):958–969, 2019.

Jaewoo So. Deep learning-based cryptanalysis of lightweight block ciphers. *Security and Communication Networks*, 2020, 2020.

O. Tito-Corrioso. Sobre el criptoanálisis a cifrados en bloque mediante el algoritmo genético. Tesis de Maestría en Ciencias Matemáticas. Universidad de La Habana, Cuba, 2021.

O Tito-Corrioso, MA Borges-Trenard, and M Borges-Quintana. Ataques a cifrados en bloques mediante búsquedas en grupos cocientes de las claves. *Rev. Cienc. Mat*, 33:71–74, 2019.

Osmani Tito-Corrioso, Miguel Angel Borges-Trenard, Mijail Borges-Quintana, Omar Rojas, and Guillermo Sosa-Gómez. Study of parameters in the genetic algorithm for the attack on block ciphers. *Symmetry*, 13 (5):806, 2021.

Felipe Luiz Tortella, Carlos Miguel Tobar Toledo, and Marcelo Luís Francisco Abbade. Aplicação de algoritmo genético para determinação de chaves usadas em criptografia óptica mediante fatiamento espectral. 2016.

Maheswara Rao Valluri. Cryptanalysis of xinyu et al.'s ntru-lattice based key exchange protocol. *Journal of Information and Optimization Sciences*, 39(2):475–479, 2018.

Conflicto de interés

Los autores autorizan la distribución y uso de su artículo.

Contribuciones de los autores

1. Conceptualización: Mijail Borges Quintana, Miguel Angel Borges Trenard,
2. Curación de datos: Miguel Angel Borges Trenard, Adrian Donatien Charón, Mijail Borges Quintana
3. Análisis formal: Miguel Angel Borges Trenard, Mijail Borges Quintana
4. Adquisición de fondos:
5. Investigación: Miguel Angel Borges Trenard, Adrian Donatien Charón, Mijail Borges Quintana
6. Metodología: Mijail Borges Quintana, Miguel Angel Borges Trenard
7. Administración del proyecto: Mijail Borges Quintana
8. Recursos:
9. Software: Miguel Angel Borges Trenard, Adrian Donatien Charón, Mijail Borges Quintana
10. Supervisión: Mijail Borges Quintana, Miguel Angel Borges Trenard
11. Validación: Miguel Angel Borges Trenard, Mijail Borges Quintana, Adrian Donatien Charón
12. Visualización: Adrian Donatien Charón, Miguel Angel Borges Trenard, Mijail Borges Quintana
13. Redacción - borrador original: Adrian Donatien Charón, Mijail Borges Quintana
14. Redacción - revisión y edición: Mijail Borges Quintana, Miguel Angel Borges Trenard, Adrian Donatien Charón

Financiación

La investigación que da origen a los resultados presentados en la presente publicación recibió fondos de la Oficina de Gestión de Fondos y Proyectos Internacionales bajo el código PN223LH006-015; así como de la Red CYTED “NUEVAS HERRAMIENTAS CRIPTOGRÁFICAS PARA LA E-COMUNIDAD”.