Análisis del desordenamiento aleatorio de bloques de mensajes como medida de integridad y seguridad de bajo coste

Analysis of the random shuffling of message blocks as a low-cost integrity and security measure

Francisco Alcaraz-Velasco, José M. Palomares, Joaquín Olivares
Departamento Ingeniería Electrónica y de Computadores
Universidad de Córdoba
Córdoba, España
francisco.alcaraz.velasco@juntadeandalucia.es jmpalomare@uco.es olivares@uco.es

Resumen — **Recientemente, se ha propuesto un mecanismo que desordena aleatoriamente los datos que se envían y que permite asegurar la comunicación sin necesidad de encriptar toda la información. Esta propuesta es ideal para sistemas IoT de baja capacidad de cómputo. En este trabajo, analizamos la fortaleza de dicha propuesta desde un planteamiento de ataque por fuerza bruta para obtener el mensaje original sin conocimiento del desordenamiento aplicado. Se consigue demostrar que para un conjunto de 10x10 datos de 16 bits, el tiempo de cómputo y la memoria requerida son inasumibles con la tecnología actual y, por tanto, es segura.**

Palabras Clave - Desordenamiento; IoT; Bajo coste; Integridad; Privacidad; Seguridad; Fuerza bruta.

Abstract — **Recently, a mechanism that randomly shuffles the data sent and allows securing the communication without the need to encrypt all the information has been proposed. This proposal is ideal for IoT systems with low computational capacity. In this work, we analyze the strength of this proposal from a brute-force attack approach to obtain the original message without knowledge of the applied disordering. It is demonstrated that for a set of 10x10 16-bit data, the processing time and the required memory are unfeasible with current technology. Therefore, it is safe.**

Keywords – Shuffling; IoT; Low cost; Integrity; Privacy; Security; Brute force.

I. INTRODUCCIÓN

Se han desplegado múltiples sistemas distribuidos para captar información de entornos mediante sensores y almacenar dichos datos para un estudio a posteriori. Sin embargo, la gran mayoría de estos sistemas han evolucionado hacia la toma de decisiones cerca de los puntos de generación de dichos datos, es decir, donde se han obtenido los datos muestreados del entorno físico [1]. Pero no sólo se captan datos de las variables físicas, sino que se activan o desactivan actuadores, que, a su vez, modifican bien el propio entorno o bien a sistemas que dependen de éste [2].

En muchos casos, la utilidad de estos datos es temporalmente limitada. Sólo son útiles mientras que estos datos presentan frescura de la información. Es decir, mientras que estos datos representan fielmente el estado del sistema en ese momento. Cuanto más tiempo pasa, menos fiable es que ese dato represente la realidad del entorno en ese instante temporal. Y por tanto, no se deberían tomar decisiones con datos desactualizados. Se podría decir que los sistemas tienden a un mecanismo de funcionamiento de datos de “usar y tirar”.

A pesar de la evolución tecnológica asociada, que ha permitido incrementar la capacidad de procesamiento de los nodos desplegados en este tipo de sistemas, siguen teniendo baja potencia de cálculo comparativamente con los servidores. Los protocolos de transmisión inalámbricos, que son los que principalmente se utilizan en estos sistemas, también han evolucionado hacia la reducción de los requisitos energéticos, el incremento de las distancias de comunicación y el aumento del ancho de banda. Sin embargo, las tasas de transferencia son mucho menores que los protocolos que se utilizan en las grandes redes de comunicaciones.

Todas estas limitaciones vienen impuestas para incrementar el tiempo de vida útil de estos sistemas distribuidos y reducir sus costes económicos y energéticos. Todo ello hace que estos dispositivos se presenten como grandes desafíos para el despliegue de nuevos algoritmos, cada vez más demandantes computacionalmente.

Por otro lado, los ataques a las infraestructuras inalámbricas de sensado [3] están teniendo mayor repercusión e importancia. Ataques más activos en los que nodos atacantes alteran paquetes e intentan modificar el comportamiento del sistema

(*tampering*) o en los que los nodos maliciosos capturan paquetes antiguos y los vuelven a enviar, afectando a la frescura de los datos enviados (*replay*) hasta ataques menos agresivos en los que nodos maliciosos capturan datos para tener información del sistema a posteriori (*sniffering*).

Los protocolos de comunicaciones inalámbricas incorporan mecanismos de comprobación de la integridad utilizando códigos cíclicos redundantes (CRC) incluidos en la capa MAC en el campo FCS (Frame Check Sequence). Los protocolos calculan, de manera transparente, el CRC (usualmente, un CRC de 16 bits utilizando la propuesta CRC-CCITT) del payload del mensaje. Se compara con el valor incluido en el campo FCS de la cabecera y, en caso de diferencia, el mensaje se rechaza y se solicita, dependiendo del protocola, el reenvío del mensaje.

En este trabajo se evalúa la fortaleza de la privacidad de la propuesta de envío de datos en la que se incorpora el mecanismo de códigos solapados [4], para garantizar integridad de los datos y a los que se le aplica una técnica de desordenamiento aleatorio, para garantizar la seguridad. Se quiere mostrar, que para un cierto volumen y sin información previa adicional sobre los datos, el mecanismo de desordenamiento proporciona un nivel suficiente de privacidad sin necesidad de hacer uso de la encriptación.

A continuación se describe la estructura del artículo. En la Sección II se realiza una breve revisión bibliográfica de las técnicas más habitualmente utilizadas en WSN para garantizar la protección de los datos. Se propondrá el mecanismo de envío de datos mediante códigos solapados con desordenamiento aleatorio en la Sección III. En la Sección IV se realiza la evaluación de la fortaleza de privacidad asociada a las técnicas de envíos de datos mediante códigos solapados con desordenamiento y se mostrará un ejemplo de funcionamiento en la Sección V. Para finalizar, en la Sección VI se proporcionan las conclusiones de este estudio.

II. Revisión Bibliográfica

A. Integridad

En IoT y otras infraestructuras con comunicaciones inalámbricas, la garantía de que los datos que se envían y reciben no han sido alterados es importante. Este mecanismo busca el principio de integridad de las comunicaciones. Existen múltiples métodos para reforzar la integridad, aunque se centran en incorporar información asociada a los datos, habitualmente llamada *firma digital*, de tal manera, que una alteración en los datos sea detectada por una inconsistencia en la firma digital incorporada. Esta firma digital se genera mediante sistemas matemáticos, basados en la suma de comprobación *checksum* bajo polinomios cíclicos generadores o CRC [5], *hash* [6] o incluso *watermarking* [7]. Siendo el mecanismo CRC el más habitual, puesto que la gran mayoría de sistemas microcontroladores incorporan elementos hardware para el cálculo y la comprobación del CRC.

B. Confidencialidad

La confidencialidad [8] de los datos (también llamada *privacidad de información orientada al contenido*) pretenden es asegurar que los datos que se transmiten por la red no puedan ser comprendido por entidades externas no autorizadas. Es decir, estos métodos no buscan impedir el acceso sino que no tengan sentido para atacantes. Lo más habitual ha sido garantizar la confidencialidad mediante mecanismos de encriptación, en la que se impide el acceso a los datos directos, que son alterados y sustituidos por otros diferentes.

Sin embargo, la encriptación es un proceso matemático complejo y requiere sistemas computacionales potentes, lo que eleva el tiempo de cómputo y la energía consumida. Por eso, la encriptación, con carácter general para todos los datos enviados, no es utilizada en dispositivos IoT de baja potencia [9].

III. Envío Solapado con Desordenamiento

En [4], los autores propusieron un mecanismo de envío de múltiples datos con control de la integridad en el que, además, se incorporaba un desordenamiento de estos datos. Según indicaban los autores este desordenamiento incrementaba la confidencialidad. En este trabajo analizaremos el impacto de dicha propuesta y evaluaremos el grado de privacidad que aporta. Comenzamos describiendo matemáticamente el planteamiento de dicha propuesta:

La entrada es un mensaje compuesto por un conjunto ordenado de r variables, divididas en c bloques del mismo tamaño, tal y como se muestra en la Ec. (1).

$$M = (m_{11}, \dots, m_{1c}, \dots, m_{r1}, \dots, m_{rc}) \quad (1)$$

La Fig. 1 muestra de manera gráfica la estructura lógica de este mensaje.

A1	A2	...	Am	CRC-A
B1	B2	...	Bm	CRC-B
⋮	⋮	...	⋮	⋮
N1	N2	...	Nm	CRC-N
CRC-1	CRC-2	...	CRC-m	CRC-mN

Figure 1: Bloque de datos de la matriz M.

La salida será un nuevo mensaje con los mismos datos de entrada con un orden alterado, compuesto por el mismo conjunto de r variables estructuradas en c bloques de igual tamaño, tal y como se indica en la Ec. (2).

$$S = (s_{11}, \dots, s_{1c}, \dots, s_{r1}, \dots, s_{rc}) \quad (2)$$

Los contenidos de las variables originales se desordenan siguiendo la permutación de desordenamiento Π, de manera que $s_{ij} = m_{\Pi^{-1}(i,j)}$. Por tanto, esta propuesta garantizaría la confidencialidad si teniendo únicamente el conjunto de datos S no se revela información sobre Π, y por tanto, no es posible obtener el mensaje original M.

Por simplificar la nomenclatura a lo largo del trabajo, se define M, el mensaje de entrada del bloque b con n-1 variables divididas en m-1 trozos como la matriz definida en la Ec. (3).

$$M_{nm}^{b} = \begin{pmatrix} m_{(1,1)}^{b} & \cdots & m_{(1,m)}^{b} \\ \vdots & \ddots & \vdots \\ m_{(n,1)}^{b} & \cdots & m_{(n,m)}^{b} \end{pmatrix} \quad (3)$$

$$\mathrm{m}_{(\mathrm{i,m})}^{\mathrm{b}} = \mathrm{CRC16}_{\mathrm{h_i}}^{\mathrm{b-1}}, \forall \mathrm{i} = 1 \ldots \mathrm{n} - 1 \quad (4)$$

$$\mathrm{m}_{(\mathrm{n,j})}^{\mathrm{b}} = \mathrm{CRC16}_{\mathrm{v_j}}^{\mathrm{b-1}}, \forall \mathrm{j} = 1 \ldots \mathrm{m} - 1 \quad (5)$$

$$\mathrm{m}_{(\mathrm{n,m})}^{\mathrm{b}} = \mathrm{CRC16}_{\mathrm{hv}}^{\mathrm{b-1}} \quad (6)$$

En esta matriz, cada fila es una variable diferente (bien, un único valor muestreado de gran tamaño que se ha dividido entre diferentes columnas consecutivas con valores de menor tamaño, o bien, varios valores de una serie lógica o temporal consecutiva muestreada del entorno que se ha posicionado en columnas consecutivas). Para reducir la correlación interna, los CRC16 que se calculen se envían en el siguiente mensaje. Así, la última columna de cada fila incluirá el resultado del cálculo del CRC16 de dicha fila, pero del bloque anterior *b-1*, indicado en la Ec. (4). De manera similar, la última fila de la matriz incluye los cálculos de los valores CRC16 de los valores asociados en la misma columna, pero del bloque anterior *b-1*, notado según se indica en la Ec. (5). Finalmente, la Ec. (6) proporciona el resultado CRC16 de los CRC horizontales y verticales indicados anteriormente. Este mecanismo desordenando valores dentro de un bloque y con cálculos entrecruzados entre bloques, reduce la correlación directa entre los valores y dificulta la detección de celdas específicas.

La matriz de salida S se describe en la Ec. (7), de manera análoga, como una matriz de $(n \times m)$ valores, que incluyen de $(n-1 \times m-1)$ datos del bloque *b* y los CRC16 de las filas y columnas del bloque *b-1*. La matriz S incluirá todos esos valores desordenados.

$$\mathrm{S}_{\mathrm{n,m}}^{\mathrm{b}} = \begin{pmatrix} \mathrm{s}_{(1,1)}^{\mathrm{b}} & \cdots & \mathrm{s}_{(1,\mathrm{m})}^{\mathrm{b}} \\ \vdots & \ddots & \vdots \\ \mathrm{s}_{(\mathrm{n},1)}^{\mathrm{b}} & \cdots & \mathrm{s}_{(\mathrm{n,m})}^{\mathrm{b}} \end{pmatrix} \quad (7)$$

En la que el cálculo de cada valor $s_{(i,j)}^{b}$ se obtiene aplicando la Ec. (8), donde Π_{ij} representa la permutación *(i,j)* de la matriz de permutación para dicha posición en una matriz de $(n \times m)$ valores. Estas matrices de permutación están compuestas por un único 1 en una única posición *(i,j)*, con *i=1… n* y *j=1 … m*. Además, la suma de todo el conjunto de matrices de permutación es igual a la matriz todo unos, tal y como se indica en la Ec. (9).

$$S_{(n,m)}^{b} = \sum_{i=1}^{n}\sum_{j=1}^{m}\left(m_{(i,j)}^{b} \cdot \Pi_{ij}\right) \quad (8)$$

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\Pi_{ij} = \begin{pmatrix} 1 & \cdots & 1 \\ \vdots & \ddots & \vdots \\ 1 & \cdots & 1 \end{pmatrix} \quad (9)$$

De esta manera, el intercambio de posiciones es único, no está repetido y debe existir un procedimiento que proporcione de manera exacta cada matriz Π_{ij} tanto en el emisor (que deberá realizar el desordenamiento) como en el receptor (que deberá reordenar para obtener el mensaje original).

En la Fig. 2 se muestra una imagen representando la matriz desordenada. Las celdas con * indican que es el CRC de dicha fila o columna del mensaje anterior.

N2	B1	…	N1	Am
CRC-m*	Nm	…	CRC-mN*	CRC-B*
⋮	⋮	…	⋮	⋮
CRC-A*	A1	…	CRC-1*	Bm
A2	CRC-2*	…	B2	CRC-N*

Figure 2: Ejemplo de bloque de datos desordenado con celdas de CRC de filas y columnas del mensaje anterior (marcadas con *).

IV. Análisis de la Confidencialidad

En esta sección analizaremos la propuesta anterior bajo la hipótesis de que el desordenamiento interno y el entremezclado entre bloques proporciona un nivel de confidencialidad suficiente.

Se acepta que un atacante pudiese haber obtenido todos los datos enviados, S^j, con $j = 1 \ldots b$, con anterioridad al bloque actual, *b*. Partimos de que se desconocen las matrices de permutación. Por tanto, solo conociendo los datos enviados $\mathrm{S^i}$ con $\mathrm{i} = 1 \ldots \mathrm{b} + 1$, es posible conocer el mensaje original M^b. Es decir, demostraremos que para conocer un mensaje es necesario haber obtenido previamente todos los anteriores y el mensaje posterior al que se desea extraer.

El mensaje de salida desordenado S^b está compuesto por datos que vienen de dos mensajes originales consecutivos M^{b-1} y M^b. Estos mensajes se envían sin encriptación, es decir, que el contenido original está presente en esos mensajes pero con un desordenamiento que es desconocido para terceros (que no sean el emisor y el receptor). Al estar los valores originales presentes en los mensajes, sería posible obtener el mensaje original probando diferentes combinaciones de las celdas de la matriz colocadas en el orden correcto.

Este proceso de prueba-error consume una elevada cantidad de tiempo y de memoria, por lo que, para mensajes de un determinado tamaño no es viable y a efectos prácticos, se puede considerar que la confidencialidad está garantizada hasta que esos datos son consumidos por el legítimo receptor.

El mensaje original se podría obtener por dos vías:

- Conociendo el valor único que permite al emisor y al receptor generar el mismo conjunto de matrices de permutación.
- Obtener la combinación correcta de los datos en filas y columnas, y la combinación correcta de CRC en el resto de celdas que no contienen datos.

En el primer caso, si se conoce el valor generador de las matrices de permutación, el atacante podría conocer el

desordenamiento tal y como lo hace el receptor legítimo. Pero, el envío de dicho valor entre el emisor y el receptor se realiza mediante un mecanismo seguro por encriptación [10]. Siguiendo este procedimiento, el atacante debe resolver un problema con una complejidad computacional O(|G|), con cardinalidad del grupo generador #(|G|) = *n-1* (siendo *n* un número primo de gran tamaño) que puede ser ajustado a un tamaño tan elevado como sea necesario para hacer inabarcable por ataque de fuerza bruta usando la potencia computacional actual. Es decir, la primera vía de obtención del valor único generador de las matrices de permutación está protegida contra ataques de fuerza bruta.

A. Fuerza bruta de matrices completas

La segunda vía, por tanto, es la única que se puede tomar para obtener los datos originales y romper la confidencialidad de la propuesta. El atacante deberá ir probando todas las posibles variaciones de datos en cada celda que compone la matriz de $(n \times m)$, sabiendo que habrá $(n-1 \times m-1)$ celdas de la matriz que contendrán los datos y $(n+m+1)$ celdas que contendrán valores CRC16 de las filas y columnas originales del bloque anterior. Sin información sobre los datos que se están enviando, no es posible identificar a priori qué celdas contienen datos y cuales CRC16, por tanto, el atacante debería almacenar todas las posibles combinaciones de alteraciones de orden de los datos, a la espera de detectar en el siguiente bloque que se envíe datos CRC16 que coincidan con los esperados en aquella combinación almacenada que sea coincidente con el mensaje original. Por tanto, necesitará obtener el mensaje del bloque b que contiene la matriz desordenada $S^b_{(n,m)}$. A partir de ese mensaje deberá generar todo el conjunto completo de matrices de $(n-1 \times m-1)$ datos, dejando libre la última fila y columna, que deberá calcularse con los CRC16 a partir de los datos de las filas y columnas. El número de diferentes matrices de datos posibles viene dado por el número de variaciones de $(n-1 \cdot m-1)$ elementos tomados de los $(n \cdot m)$ datos totales, tal y como se desarrolla en la Ec. (10).

$$V_{(n\cdot m),(n-1\cdot m-1)} = \frac{(n \cdot m)!}{(n+m-1)!} \qquad (10)$$

En cuanto capture el mensaje S^b, el atacante podrá comenzar a preparar todas las variaciones posibles ordenándolas por filas y columnas, y calculando el CRC16 de cada fila y columna. Sin embargo, deberá esperar a capturar el mensaje S^{b+1} para comenzar a comparar por fuerza bruta los CRC calculados con todas las celdas de dicho nuevo mensaje, esperando encontrar dicho valor en alguna celda. En caso de encontrarlo, podría asumir que esa combinación de valores en fila o columna de la que se calculó el CRC era correcta y habría encontrado un posible vector-fila o vector-columna que aparecía en el mensaje original de inicio M^b. En caso de no encontrarlo, se podría descartar esa combinación. A modo de ejemplo, partiendo de matrices con datos de 2 bytes por celda, se presenta la Tabla I. Se incluye el tamaño en bytes del almacenamiento en memoria de todas las variaciones de matrices necesarias. También se incluye una estimación del tiempo de cómputo que llevaría el cálculo de los CRC16 necesarios, asumiendo una implementación software optimizada de dicho cálculo [11]. En la Fig. 3 se puede observar el resultado comparativo de la cantidad de memoria que ocupan las matrices en memoria y el tiempo de cómputo que llevaría el cálculo de todos los CRC16 de dichas matrices en memoria. Los ejes se encuentran en escala logarítmica. El área azul tiene como eje el lado izquierdo, en Bytes. Las barras rojas tienen como eje el lado derecho, en segundos.

TABLE I. FUERZA BRUTA DE MATRICES COMPLETAS

	Tamaño (Bytes)	***Nº Matrices***	***Memoria (Bytes)***	***Tº Cómputo (segundos)***
2 x 2	8	4	32	6,93E-10
3 x 3	18	3.024	54.432	5,23E-07
5 x 5	50	4,27E+19	2,14E+19	7,41E+09
7 x 7	98	9,77E+52	9,57E+54	1,69E+43
8 x 8	128	9,7E+76	1,24E+79	1,68E+67
10 x 10	200	7,67E+140	1,53E+143	1,33E+131

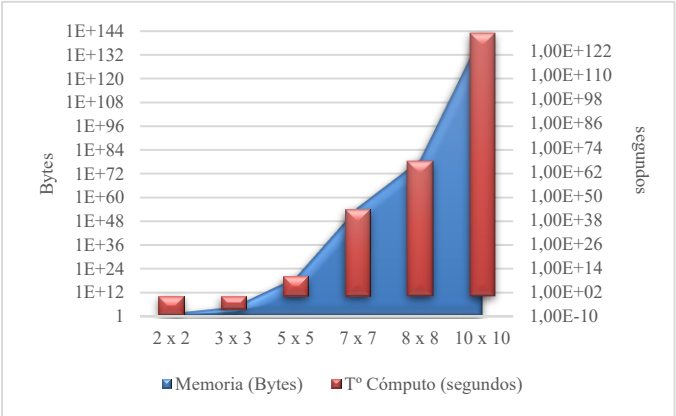

Figure 3: Tiempo de cómputo y ocupación de memoria según tamaño de matriz de datos para la aproximación por fuerza bruta de matrices completas

Conforme van aumentando el tamaño de las matrices de datos, el número de variaciones de matrices se incrementa, resultando en un mayor consumo de memoria para almacenar dichas matrices y en un consecuente, aumento del tiempo de cómputo para calcular todos los CRC asociados. Es evidente que en la actualidad no hay computadores con tal cantidad de memoria ni con tal capacidad de cómputo, por lo que, usando esta estrategia, se podría considerar completamente seguro esta propuesta con la tecnología actual.

B. Fuerza bruta en dos pasos por vectores-fila y matrices

El mecanismo de fuerza bruta de matrices completas es altamente ineficiente, puesto que existen múltiples filas y columnas repetidas a las que se le recalculan múltiples veces el CRC. Por tanto, se plantea una mejora en el mecanismo de fuerza bruta seleccionando todos los diferentes vectores-fila posibles y calculando los CRC de cada uno de ellos. En una segunda fase, se seleccionan aquellos vectores-fila del mensaje S^b cuyos CRC se detectan en el mensaje S^{b+1}. La última fase de este procedimiento será reorganizar el orden de estos vectores-fila para obtener los valores CRC correctos por columnas.

De esta manera, este procedimiento tiene los siguientes pasos:

1. Obtener todas las posibles filas de *m-1* valores a partir de los $(n \times m)$ datos de la matriz desordenada S^b. La Ec. (11) determina el número de diferentes vectores-fila que se pueden generar a partir de los $(n \times m)$ datos. Todos estos vectores-fila generados se almacenan en memoria.
2. Se calcula el CRC de cada vector-fila y se almacena en memoria.
3. Al recibir el siguiente bloque desordenado S^{b+1}, se buscan los valores CRC de cada vector-fila en alguna celda del nuevo bloque capturado. Se deberá encontrar exactamente *n-1* vectores-fila del bloque S^b que cumplan que su CRC se ha encontrado en el bloque S^{b+1}.
4. Reordenar los vectores-fila seleccionados del paso anterior según todas las posibles variaciones de orden, indicados en la Ec. (12). Calcular los CRC de todas las variaciones y buscar dichos valores en el bloque S^{b+1}.

$$V_{(n\cdot m),(m-1)} = \frac{(n\cdot m)!}{(n\cdot m - m + 1)!} \quad (11)$$

$$V_{(n-1,n-1)} = (n-1)! \quad (12)$$

En la Tabla II, se presentan los resultados de este procedimiento en memoria ocupada tanto por los vectores-fila del paso 1 como las reordenaciones del paso 4, y el tiempo de cómputo global de todos los pasos. No se incorpora el tiempo que transcurre hasta la recepción del mensaje S^{b+1}, ya que es algo que depende de la aplicación. Hasta que no se capture el mensaje S^{b+1} no es posible continuar con el paso 3 anterior.

La Fig. 4 representa de manera gráfica la memoria total consumida y el tiempo de cómputo global de esta propuesta. Los ejes se encuentran en escala logarítmica. El área azul tiene como eje el lado izquierdo, en Bytes. Las barras rojas tienen como eje el lado derecho, en segundos.

TABLE II. FUERZA BRUTA EN 2 PASOS

	Nº Vectores (1er paso)	*NºMatrices (2º paso)*	*Memoria (Bytes)*	*Tº Cómputo (segundos)*
2 x 2	4	1	16	3,25E-10
3 x 3	72	2	180	9,88E-09
5 x 5	3,04E+05	24	6,08E+05	7,9E-05
7 x 7	1,01E+10	720	2,01E+10	3,93
8 x 8	3,13E+12	5040	6,26E+12	1,4E+03
10 x 10	6,9E+17	362880	1,38E+18	4,04E+08

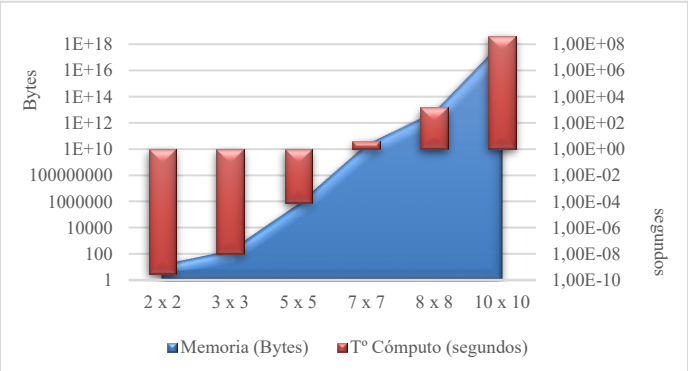

Figure 4: Tiempo de cómputo y ocupación de memoria según tamaño de matriz de datos para la aproximación por fuerza bruta de dos pasos por vectores-fila y matrices

En este caso, la ocupación de memoria total se contiene enormemente con respecto al método de fuerza bruta por matrices completas. El tiempo de cómputo es aún más acotado que en el caso anterior. Si bien hay que recordar que a este tiempo de cómputo hay que incluir el tiempo que transcurre hasta que se capture el siguiente mensaje S^{b+1}, ya que el atacante no puede determinar qué vectores-fila son candidatos hasta que no pueda comprobar el CRC en el siguiente mensaje. Así pues, para un tamaño de datos no excesivamente grande como 8x8, en el que el emisor envía 128 bytes de datos, el atacante debería destinar más de 45,5 TB de memoria para almacenar todos los posibles vectores-fila y matrices candidatas por cada mensaje capturado. Y, asumiendo que tuviese tal cantidad de memoria, necesitaría más de 23 minutos para el cómputo de los CRC. En el caso de la memoria, en la actualidad, no hay sistemas con tal cantidad de memoria salvo en servidores o en la nube, por lo que su aplicabilidad queda muy restringida a unos pocos ordenadores concretos. Pero en el caso de la potencia de cómputo, si el tiempo de envío de tramas de datos es inferior a esos 23 minutos, el atacante podría obtener el mensaje original después de que hubiera sido utilizado, restándole utilidad y frescura a dichos datos. En cualquier caso, si se envían matrices de 10x10, con 200 bytes de datos en total, el tiempo de cómputo de los CRC se elevaría a más de 12 años, lo que le resta el interés completamente a un atacante que desease obtener datos mínimamente actualizados.

V. EJEMPLO DE USO

Para facilitar la comprensión de los mecanismos de fuerza bruta que se han presentado en la Sección IV, en la Fig. 5 se presenta un ejemplo de un mensaje M^b con datos de 2x2.

A1	A2	CRC-A
B1	B2	CRC-B
CRC-1	CRC-2	CRC-mN

Figure 5: Mensaje original de 2x2 datos (A1, A2; B1, B2) y los CRC correspondientes a dichas filas y columnas. Los CRC se enviarán en el siguiente mensaje.

En la Fig. 6 se muestra un mensaje S^b. En la parte izquierda cómo estaría desordenado internamente cada celda, representando cada fila con un color y con * las celdas que contienen los CRC del mensaje anterior. Sin embargo, el atacante desconoce estas identificaciones internas y por lo tanto, el atacante observa un paquete en el que no puede hacer ningún tipo de presunción inicial y lo cataloga como se representa en la parte derecha.

B2	CRC-1*	CRC-2*
A2	CRC-Mn*	B1
CRC-A*	A1	CRC-B*

A	B	C
D	E	F
G	H	I

Figure 6: Izquierda: Ejemplo mensaje de 2x2 datos y CRC del mensaje anterior. Derecha: Mensaje sin identificación visto por el atacante.

El mecanismo de fuerza bruta por matrices completas almacenaría en memoria todas las combinaciones de 2x2 con

los datos A, B, C, D, E, F, G, H, I puesto que el atacante no sabe qué celdas contienen datos y cuáles CRC. En la Fig. 7 se muestran tres matrices de 2x2 de las 3024 matrices distintas, según cálculos de la Ec. (10), que almacenaría en memoria.

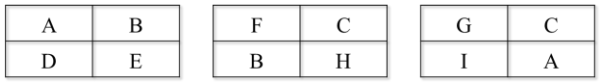

Figure 7: Ejemplo de tres matrices de datos de 2x2 almacenadas en memoria.

De cada una de las 3024 matrices calcularía por filas y columnas el CRC, lo almacenaría y compararía dichos valores con las celdas del siguiente mensaje, esperando encontrar alguna matriz en la que todos los CRC calculados se encuentren en el nuevo mensaje.

El segundo mecanismo de fuerza bruta en dos pasos por vectores-fila y matrices partiría de la Fig. 6 y generaría 72 diferentes vectores-fila de 2 valores, según la Ec. (11), de entre los datos A, B, C, D, E, F, G, H, I y calcularía el CRC de cada vector-fila. Posteriormente, tras capturar el siguiente mensaje, compararía los CRC computados con los datos en las celdas. Tras identificar los vectores-fila candidatos, se construirían en memoria las 2 matrices, según Ec. (12), que contendrían las posibles ordenaciones de las filas en la matriz de 2x2 final. La Fig. 8 muestra 6 vectores de los 72 posibles.

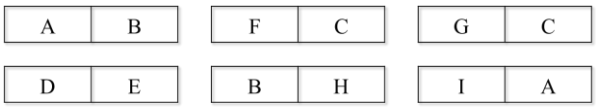

Figure 8: Ejemplo de 6 vectores-fila de 2 datos almacenados en memoria.

VI. CONCLUSIONES

En este trabajo se ha analizado el grado de confidencialidad que es capaz de proporcionar una propuesta que no hace uso generalizado de la encriptación, sino del desordenamiento de los datos. Dicha propuesta es particularmente interesante en sistemas de baja capacidad de cómputo con flujos de envíos de datos constantes con periodos no muy elevados de sensado y envío, en sistemas de uso de datos inmediatos y que, por tanto, requieren frescura en los datos.

Se han propuesto dos alternativas, sin conocimiento a priori, para la ruptura de la confidencialidad de los datos. Estas dos propuestas son por fuerza bruta. La primera propuesta analiza todas las posibles matrices de datos que se pueden generar con los valores recibidos. Esta propuesta tiene unos requisitos muy elevados tanto de memoria requerida como de tiempo de cómputo necesitado para su resolución. Como ejemplo, para un envío de 8x8 datos, 128 bytes, que es un payload que cabe en un único mensaje en la mayoría de protocolos de comunicación usados en sistemas WSN e IoT, los requisitos de tiempo de cómputo son superiores a los miles de millones de años.

Sin embargo, este método propuesto es altamente ineficiente, y se propone un segundo método que baja drásticamente los requisitos tanto de memoria como de tiempo de cómputo. Este segundo método, también por fuerza bruta, realiza la estimación del mensaje original con un proceso de 2 fases: la primera para establecer los vectores-fila candidatos y en la segunda etapa, la organización de los vectores-fila en una matriz. En este caso, los resultados indican que para un envío de 8x8 datos, que son 128 bytes y que cabe en el payload de los mensajes de los principales protocolos de comunicación de IoT y WSN, los requisitos de memoria se elevan a los 45,5 TBytes y el tiempo de cómputo para calcular los CRC sería de casi 24 minutos. Si incrementamos a 200 bytes de datos (una matriz de 10x10 valores), el tiempo de cómputo se eleva a más de 12 años y la cantidad de memoria requerida se incrementaría hasta los 14.000 PBytes. En cualquier caso, ambos parámetros exceden con creces las posibilidades actuales de los sistemas informáticos.

Por tanto, a no ser que haya mecanismos que permitan identificar los datos, agruparlos, descartar algunas celdas y de esa manera, reducir los requisitos, por fuerza bruta, se puede concluir que el procedimiento propuesto en [4] presenta un nivel de confidencialidad suficiente para el tipo de aplicaciones habituales en los sistemas distribuidos de sensado y actuación para mensajes de tamaño de 8x8 datos.

AGRADECIMIENTOS

Este trabajo ha sido parcialmente financiado por MINECO España bajo el Proyecto SmartFog (RTI2018-098371-B-I00).

REFERENCIAS BIBLIOGRÁFICAS

[1] F. León-García, J. M. Palomares y J. Olivares, «D2R-TED: Data-Domain reduction model for threshold-based event detection in sensor networks,» *Sensors,* vol. 18, nº 11, p. 3806, 2018.

[2] F. León-García, F. J. Rodríguez-Lozano, J. Olivares y J. M. Palomares, «Data communication optimization for the evaluation of multivariate conditions in distributed scenarios,» *IEEE Access,* nº 2019, pp. 133473-123489, 2019.

[3] A. Diaz y P. Sanchez, «Simulation of attacks for security in wireless sensor networks,» *Sensors,* vol. 16, nº 11, pp. 1-27, 2016.

[4] F. Alcaraz, J. M. Palomares y J. Olivares, «Light-weight method of shuffling overlapped data-blocks for da-ta integrity and security in WSNs,» *Computer Networks,* vol. 199, p. 108470, 2021.

[5] W. An, M. Médard y K. Duffy, «CRC Codes as Error Correction Codes,» de *ICC 2021 - IEEE International Conference on Communications 2021*, Montreal, Canada, 2021.

[6] J. Bok, Y. Lee, J. Park y J. Yoo, «An Energy-efficient scheme in Wireless Sensor Networks,» *Journal of Sensors,* vol. 1321079, pp. 1-11, 2016.

[7] G. Zhang, L. Kou, L. Zhang, C. Liu, Q. Da y J. Sun, «A New Digital Watermarking Method for Data Integrity Protection in the Perception Layer of IoT,» *Security and Communication Networks,* vol. 3126010, pp. 1-12, 2017.

[8] L. J, R. Rios, F. Bao y G. Wang, «Evolving privacy: From sensors to the internet of things,» *Future Generation Computer Systems,* vol. 75, pp. 46-57, 2017.

[9] J. Lee, K. Kapitanova y S. Son, «The price of security in wireless sensor networks,» *Computer Networks,* vol. 54, nº 17, pp. 2967-2978, 2010.

[10] F. Bouakkaz, M. Omar, S. Laib, L. Guermouz, A. Tari y A. Bouabdallah, «Lightweight Sharing Scheme for Data Integrity Protection in WSNs,» *Wireless Personal Communications,* vol. 89, pp. 211-226, 2016.

[11] V. Gopal, E. Ozturk, J. Guilford y W. Feghali, «Choosing a crc polynomial and associated method for Fast CRC Computation on Intel Processors,» Intel Corporation, 2012.