Ataques por canal lateral contra AES mediante correlación de consumo de potencia

Miguel Ángel González de la Torre
I. Tecnologías Físicas y de la Información
ITEFI-CSIC, Madrid
ma.gonzalez@csic.es

Ventura Sarasa Laborda
I. Tecnologías Físicas y de la Información
ITEFI-CSIC, Madrid
ventura.sarasa@csic.es

Luis Hernández-Álvarez
I. Tecnologías Físicas y de la Información
ITEFI-CSIC, Madrid
luis.hernandez@csic.es

Iván Morales Sandoval
Constructor University Bremen
Bremen, Alemania
imorales@constructor.university

Luis Hernández Encinas
I. Tecnologías Físicas y de la Información
ITEFI-CSIC, Madrid
luis.h.encinas@csic.es

***Resumen*—Los ataques por canal lateral son un tipo de ataque que pretende vulnerar la seguridad de los algoritmos criptográficos implementados en dispositivos físicos. El proceso consiste en obtener información de determinadas medidas colaterales (tiempo, consumo de potencia, radiación electromagnética, etc.) que afectan al comportamiento del dispositivo atacado y deducir de ellas información valiosa, principalmente, las claves secretas almacenadas en los mismos. En este trabajo se investiga la seguridad de las implementaciones sin contramedidas del algoritmo estándar de cifrado simétrico AES para sus tres longitudes de clave: 128, 192 y 256 bits, utilizando la plataforma ChipWhisperer. Se han logrado ataques exitosos que permiten obtener las claves secretas de AES para las tres versiones consideradas ejecutando el algoritmo de cifrado. También se ha conseguido obtener la clave de AES-128 considerando el proceso de descifrado, conjeturándose que este ataque es extensible a las otras dos versiones de AES.**

***Index Terms*—AES, ChipWhisperer, Correlación, CPA, Traza.**

Tipo de contribución: *Investigación original*

I. Introducción

Desde hace unos años se han venido desarrollando ataques contra los dispositivos que implementan algoritmos criptográficos, fundamentando tales ataques en las características específicas de dichas implementaciones más que en el problema matemático que sustenta la seguridad del algoritmo en cuestión (ver [1], [2]). Este tipo de ataque se suele denominar ataque por canal lateral o ataque por inducción de fallos.

En tales ataques se considera como premisa básica que su éxito depende, en gran medida, del dispositivo utilizado y de la implementación concreta realizada. Se da por hecho, además, que el atacante tiene acceso completo al dispositivo físico, conoce los algoritmos implementados, puede ejecutar dichos algoritmos cuantas veces necesite y es capaz de medir determinados parámetros generados por el propio dispositivo. De hecho, lo único que el atacante no conoce es la clave almacenada en el dispositivo, que es, precisamente, su objetivo.

Los ataques por canal lateral se suelen clasificar atendiendo al tipo de parámetro a medir en función del canal empleado, que intenta vulnerar el dispositivo. De este modo, un dispositivo se puede atacar mediante análisis temporal, análisis de potencia, análisis de emanaciones electromagnéticas, etc. [3, Cap. 11]. En particular, los ataques por análisis de correlación del consumo de potencia o CPA (*Correlation Power Analysis*) miden la potencia consumida por un dispositivo electrónico en cada ciclo de reloj a medida que va realizando los diferentes cómputos que el algoritmo criptográfico requiere [4]. Esta medida permite obtener información sobre los datos que está procesando el dispositivo.

En el contexto de operaciones que involucran datos sensibles, como las realizadas en procesos criptográficos, se destaca la influencia significativa que incluso pequeñas variaciones en dichos datos pueden ejercer sobre el consumo de potencia por parte del dispositivo. Por ejemplo, al cifrar un mensaje, ciertas operaciones en el dispositivo pueden requerir más o menos potencia dependiendo de los valores específicos de los datos procesados en ese momento. Estas variaciones en el consumo pueden ser sutiles, pero son detectables. Luego, mediante el análisis de estas mediciones de potencia junto con el conocimiento de las operaciones criptográficas que se están llevando a cabo, se buscan correlaciones entre los patrones en la potencia consumida y los valores de los datos secretos, como son las claves del algoritmo criptográfico. Esta correlación se lleva a cabo mediante los pesos de Hamming (*Hamming Weight* o HW) obtenidos a partir de los datos sensibles.

De forma más concreta, en este trabajo se ha llevado a cabo un ataque a diferentes implementaciones de AES (*Advanced Encryption Standard*), sin contramedidas, mediante un ataque CPA [5]. El ataque se ha realizado mediante la ejecución del estándar criptográfico AES en la plataforma Chipwhisperer, para lo que se ha llevado a cabo un proceso de medición de trazas de consumo de potencia durante la ejecución del algoritmo.

En la literatura se pueden encontrar otros ataques CPA [6] similares al que se presenta aquí. En nuestro caso, nos planteamos un doble objetivo: el primero es probar el alcance del ataque, algo que no queda detallado en las versiones iniciales que hemos estudiado. Además, hemos adaptado el ataque a las versiones de AES-192 y AES 256 y posteriormente, lo hemos ejecutado considerando las trazas del descifrado en vez de las del cifrado (que es el caso más comúnmente publicado). El segundo objetivo es iniciar una línea de investigación que aborde el estudio de las posibles vulnerabilidades de

las implementaciones de diferentes algoritmos, siendo AES el primero que hemos considerado. La razón es que es uno de los estándares más empleados en la actualidad. A partir de los resultados y la experiencia obtenidos nos proponemos expandir este trabajo a versiones de AES con contramedidas y a algoritmos postcuánticos, como pueden ser CRYSTALS-Kyber o ASCON.

Este ataque nos ha permitido recuperar la totalidad de la clave de las versiones AES-128, AES-192 y AES-256 empleadas en los procesos de cifrado. Siguiendo un enfoque similar y partiendo de [6], hemos conseguido recuperar, de nuevo, la clave completa de la variante AES-128 pero en el proceso de descifrado, conociendo únicamente el texto cifrado.

El resto de este trabajo se distribuye de la siguiente manera. En la sección II se describe, brevemente, el algoritmo AES. La sección III explica el tipo de ataque realizado, es decir, el ataque CPA ejecutado para en las operaciones de cifrado para cada una de las tres versiones de AES y de descifrado para el caso del AES-128, y las especificaciones pertinentes de Chipwhisperer. La sección IV muestra los resultados obtenidos, incluyendo los patrones del consumo de potencia medidos. Finalmente, la sección V enuncia las conclusiones de esta investigación, así como posibles trabajos futuros.

II. Advanced Encryption Standard (AES)

AES es el algoritmo de cifrado por bloques simétrico más utilizado para proteger la confidencialidad de la información en sistemas de comunicación y almacenamiento [7]. AES opera en bloques de datos de 128 bits, utiliza claves de diferente longitud: 128, 192 o 256 bits, y un número variable de rondas, que depende de la longitud de la clave. Así, para las longitudes de 128, 192 y 256 bits, el número de rondas es, respectivamente, 10, 12 y 14.

Su fortaleza radica en su diseño matemático y su capacidad para resistir diversos tipos de ataques criptográficos conocidos. El AES ha demostrado ser esencial en la protección de datos sensibles en una amplia gama de aplicaciones, desde la seguridad informática hasta las transacciones en línea, contribuyendo significativamente a la seguridad de la información en la era digital.

II-A. Estructura de AES

El esquema de cifrado de AES puede verse en la Figura 1, donde la entrada es el texto claro o mensaje, m, junto con la clave de cifrado o clave maestra, k, y su salida es el texto cifrado o criptograma, c. Debe tenerse en cuenta que el proceso de descifrado de AES sigue el mismo proceso que el de cifrado, pero de modo inverso. Internamente, los procesos de cifrado y descifrado de AES consisten en aplicar una serie de funciones invertibles.

El resultado obtenido al aplicar las funciones internas del algoritmo se denomina "estado" y se denota por s. Todos los estados constan de 128 bits y se suelen representar como matrices de tamaño 4×4, en las que cada entrada es un byte.

En todas las versiones de AES se aplica una transformación inicial, seguida de $N_r - 1$ vueltas regulares y se concluye con una una vuelta final. Las vueltas inicial y final son diferentes de las regulares: en la inicial, se hace un XOR entre el bloque a cifrar (de 128 bits) y 128 bits de la clave. Así, para AES-128

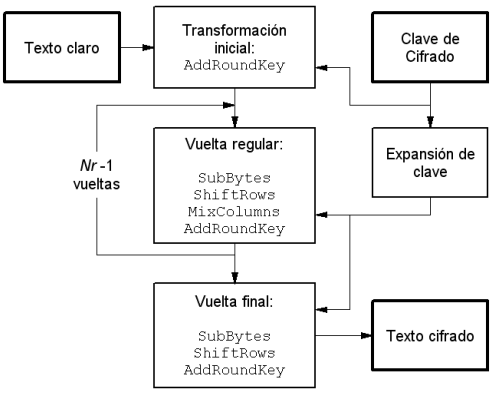

Figura 1. Esquema del algoritmo de cifrado de AES

se emplea la clave maestra completa, mientas que en AES-192 y AES-256, solo se emplean los 128 primeros bits de la clave. La transformación del estado que se produce en las vueltas regulares, cada una de las cuales es la combinación de las siguientes cuatro transformaciones:

1. SubBytes(s) o SB(s): esta función aplica, de forma independiente, una matriz de sustitución, llamada Sbox, a cada byte del estado S.
2. ShiftRows(s) o SR(s): esta transformación permuta cíclicamente los valores de las filas del estado s. La primera fila del estado permanece fija, la segunda fila desplaza cada entrada una posición a la izquierda, en la tercera se desplazan dos posiciones y en la cuarta tres posiciones.
3. MixColumns(s) o MC(s): aplica una multiplicación matricial por la izquierda del estado s con una matriz característica predeterminada. Su objetivo es el de maximizar la difusión.
4. AddRoundKey(s, k_r) o ARK(s, k_r): lleva a cabo la operación XOR entre los bytes del estado s y cada una de las subclaves k_r.

Todas la funciones internas de AES tienen sus correspondientes inversas, que permiten intercambiar los procesos de cifrado y descifrado.

II-B. Expansion de clave en AES

La clave maestra de AES, sea de longitud 128, 192 o 256 bits, está sujeta a lo que se denomina algoritmo de expansión de la clave. Este algoritmo es necesario porque en cada una de las iteraciones se utiliza una clave de longitud menor que la clave maestra, pero que se deriva de dicha clave. Cada una de las claves que se emplea en cada una de las vueltas se denomina subclave.

El proceso de la expansión de la clave de AES es un punto básico en su estructura y su funcionamiento tiene gran importancia en el diseño de los ataques por canal lateral propuestos contra este criptosistema. Inicialmente, la clave maestra, sea k, se considera como una entrada al algoritmo (tanto en el proceso de cifrado como en el descifrado), siendo su longitud variable, dependiendo de la versión de AES que se considere: 16, 24 o 32 bytes. A partir de esta clave maestra, se generan N_r subclaves, denotadas por k_r, una para cada ronda, y, adicionalmente, los primeros 16 bytes de la clave maestra se

toman como la primera expansión (teniendo entonces $N_r + 1$ expansiones) para ser utilizada en la primera de las vueltas. Cada una de las subclaves es una entrada de la función $\mathsf{ARK}(s, k_r)$ en cada una de las vueltas del algoritmo.

Con el fin de observar más detenidamente la influencia de los diferentes algoritmos de expansión de la clave para cada uno de las versiones de AES y su repercusión en el ataque realizado, vamos a analizar con algo de detalle cómo es dicho algoritmo para las tres versiones de AES [3, §3.5].

El proceso de expansión de clave en AES se lleva a cabo mediante la construcción de una secuencia, W, compuesta por $4(N_r + 1)$ palabras, W_i, de 32 bits cada una, con $0 \leq i \leq 4(N_r + 1) - 1$. Esto significa que, con la clave, se generarán 44 palabras de 32 bits con la clave de 128 bits, 52 palabras de 32 bits en el caso de AES-192 y 60 palabras de 32 bits en el caso de una clave de 256 bits (véase [3, §3.5.6]. En todas las situaciones, las $N_k \in \{44, 52, 60\}$ palabras iniciales de la clave constituyen las primeras N_k palabras de la clave expandida: $W_0, \ldots, W_{N_k-1}$.

La obtención de cada una de las palabras, W_i, $0 \leq i \leq 4(N_r + 1) - 1$ puede establecerse de la siguiente manera. En primer lugar, las $4(N_r + 1) - 1$ palabras W_i se alinean en filas de modo que el número de columnas de cada fila será $Nk = 4, 6, 8$, según que se trate del AES-128, AES-192 o AES-256, respectivamente. Cada nueva palabra, W_j, se construye como sigue

$$\left.\begin{array}{l} W_j = W_{j-4} \oplus W_{j-1}, \text{ si } j > Nk, j \equiv i \mod Nk, i \neq 0 \\ W_j = W_{j-4} \oplus \mathcal{T}(W_{j-1}), \text{ si } j \geq Nk, j \equiv 0 \mod Nk \end{array}\right\}$$

donde la transformación $\mathcal{T}(W_{j-1})$ consiste en la aplicación sucesiva de sucesiva de las siguientes tres transformaciones: RW (RotWord), SW (SubWord) y Rc_n (Rcon$[n]$).

La transformación RW consiste en rotar cíclicamente a izquierda, byte a byte, cada palabra de entrada. Por su parte, la transformación SW consiste en aplicar la función SB a cada uno de los 4 bytes de cada palabra de entrada. Finalmente, Rc_n es una constante de 32 bits dependiente del número de vuelta n.

La Figura 2 muestra, a modo de ejemplo, el algoritmo de expansión de la clave para AES-128.

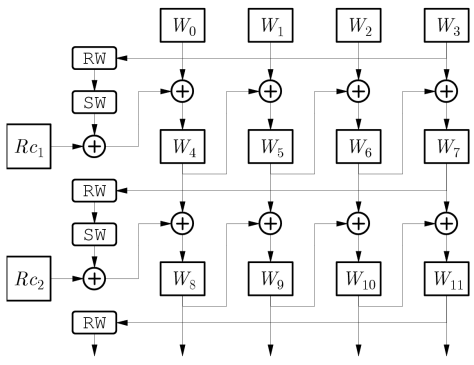

Figura 2. Algoritmo de expansión de la clave para AES-128

Téngase en cuenta que en el proceso de descifrado, la forma de expandir la clave maestra para obtener las subclaves debe seguir el proceso inverso, de modo que la primera subclave de cifrado será la última de descifrado y viceversa.

III. Ataque de tipo CPA a la implementación

Los ataques de tipo CPA son, como ya hemos mencionado, un tipo específico de ataque por canal lateral que aprovecha las fugas de información relacionadas con el consumo de energía eléctrica del dispositivo durante la ejecución de un algoritmo criptográfico [3, Cap. 11].

Este tipo de ataque aprovecha la tecnología usada en los microprocesadores actuales, conocida como CMOS o metal-óxido-semiconductor complementario (*Complementary Metal-Oxide-Semiconductor*), que posee dos características distintivas: alta inmunidad al ruido y bajo consumo de potencia estática. De hecho, solo se requiere una cantidad significativa de potencia dinámica cuando los transistores conmutan entre los estados de encendido y apagado. En particular, existe una relación directa entre el consumo de potencia dinámico y el número de bits que cambian [11], por lo que un atacante puede adivinar un valor secreto utilizado en determinada operación observando la curva de consumo de potencia o traza.

Uno de los modelos de fuga más empleados es el del peso de Hamming (HW) del bit que el atacante pretende adivinar [6]. En este modelo se considera que las transiciones $0 \rightarrow 0$ y $1 \rightarrow 0$ no producen un exceso de consumo de potencia; mientras que las transiciones $0 \rightarrow 1$ y $1 \rightarrow 1$ necesitan un exceso de consumo de potencia.

Así pues, los ataques CPA explotan la existencia de una correlación entre los datos procesados y la información que se obtiene midiendo la potencia instantánea consumida por el dispositivo [5]. Dado que esta correlación es realmente muy pequeñas, se suelen realizar muchas mediciones de los proceso de cifrado o descifrado y obtener así muchas trazas. Luego se hace uso de métodos estadísticos que permitan obtener datos cuando se llevan a cabo comparaciones entre las trazas obtenidas con las salidas obtenidas cuando se emplea determinado modelo en el dispositivo que se ataca. Esto, es, el objetivo de los ataques CPA consiste en encontrar relaciones entre las trazas obtenidas al medir el consumo de potencia y al aplicar un modelo de fuga a los datos que se pueden medir a partir de cierta información inicial. Dependiendo del algoritmo y del tipo de dato capturado, el modelo de fuga puede variar. Encontrar esta relación suele permitir reducir enormemente el coste de ataques por fuerza bruta tomando como objetivo partes muy concretas de la información sensible.

Los ataques CPA suelen desarrollarse en tres fases:

1. El atacante mide la potencia consumida en un dispositivo físico durante la ejecución del algoritmo objetivo. Dependiendo del caso de estudio, el atacante también puede tener acceso a información no pública. Finalmente, las medidas de consumo de potencia, o trazas, son almacenadas en un vector de consumo.
   A modo de ejemplo, en la Figura 3 puede verse la captura del consumo de potencia de una parte de la ejecución de un AES-128. En particular, esta traza muestra las tres primeras rondas del AES con una definición de unos 17000 puntos. Se puede apreciar que el mismo patrón se repite tres veces (se han añadido dos líneas verticales para separarlos). Cada una de las rondas mostradas presenta el consumo de los primeros

cómputos hasta llegar a cuatro picos que corresponden a los consumos de la función MC.

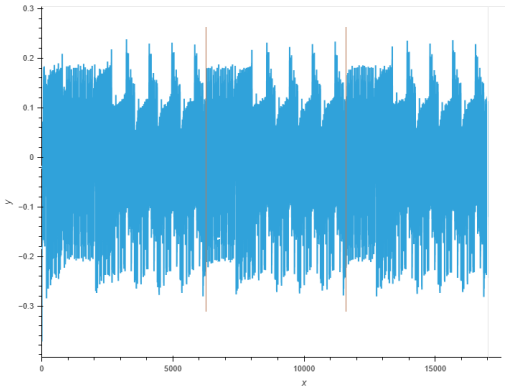

Figura 3. Traza del consumo de potencia de tres rondas de un AES-128

2. El atacante predice la potencia consumida por el dispositivo en un determinado instante del procedimiento del algoritmo. En este trabajo, el modelo de fuga considerado para las distintas versiones de AES se basa en la relación de la potencia consumida y los pesos de Hamming de las salidas de la Sbox. Estos cálculos se llevan a cabo a nivel de byte, para los posibles valores de la clave hipotética y un número N de textos claros de entrada. Los resultados son almacenados en matrices de predicción.
3. Por último, el atacante compara las diversas predicciones con las medidas reales de la potencia consumida. En este caso, se ha utilizado el coeficiente de correlación de Pearson [12], que mide la correlación lineal entre los dos conjuntos de datos, X e Y, de longitud N, y con medias $\bar{X}$ y $\bar{Y}$ respectivamente, como sigue:

$$r = \frac{cov(X,Y)}{\sigma_X \sigma_Y},$$

donde la covarianza y la desviación típica tienen las siguientes expresiones:

$$cov(X,Y) = \sum_{n=1}^{N} \left[(Y_n - \bar{Y})(X_n - \bar{X}) \right],$$

$$\sigma_X = \sqrt{\sum_{n=1}^{N} (X_n - \bar{X})^2}.$$

De este modo, es posible calcular la correlación entre el vector de consumo y las columnas de la matriz de predicción correspondiente para todos los valores que son candidatos a ser la clave empleada. En caso de que el ataque sea satisfactorio, se espera que un único valor, el correspondiente a la clave candidato correcta, produzca el coeficiente de correlación más alto.

III-A. Chipwhisperer

ChipWhisperer es una plataforma de código abierto diseñada para la evaluación y mejora de la seguridad de los sistemas embebidos, centrándose en la evaluación de algoritmos criptográficos. Desarrollada por NewAE Technology Inc., ChipWhisperer se utiliza para analizar vulnerabilidades de seguridad, especialmente aquellas relacionadas con ataques de canal lateral. Las principales características y funcionalidades de ChipWhisperer incluyen la captura de trazas, el análisis de correlación, la inyección de fallos, las pruebas de seguridad en hardware y el desarrollo de contramedidas.

Existen diferentes placas para la realización de los ataques y análisis de seguridad. Para la realización de este trabajo se ha utilizado Chipwhisperer-*lite* y un hardware añadido que se muestra en el Figura 4. En particular, Chipwhisperer-*lite* se ha empleado como capturador de trazas (dispositivo negro situado a la izquierda de la Figura 4), junto con la *UFO board* (placa roja de la Figura 4) a la que se conecta una placa objetivo que es donde se ejecuta el algoritmo estudiado (placa azul de la Figura 4). En este trabajo se ha empleado como placa objetivo un microcontrolador STM32F3.

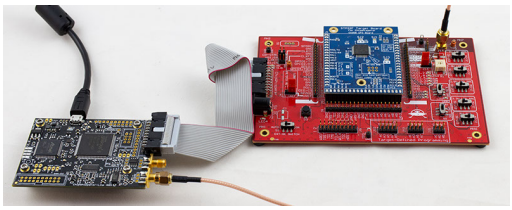

Figura 4. Hardware usado para la ejecución de los ataques contra AES

La versión ChipWhisperer-*lite* de la placa única está disponible con un objetivo Microchip Technology/Atmel XMEGA o un objetivo STMicroelectronics STM32F3 Arm®. En particular, este dispositivo es el encargado de la adquisición de las trazas de consumo de potencia y medida de tiempo durante la ejecución de algoritmos criptográficos en los microcontroladores y otros dispositivos embebidos. Por otra parte, la STM32F3 es una familia de microcontroladores desarrollada por STMicroelectronics. Estos microcontroladores están basados en la arquitectura ARM Cortex-M. Esta placa se destaca por su rendimiento y versatilidad, siendo especialmente adecuada para sistemas embebidos en tiempo real y aplicaciones que involucran procesamiento de señales.

Aprovechando la plataforma de código abierto de ChipWhisperer, hemos utilizado jupyter para programar el ChipWhisperer-*lite*. En cuanto a la implementación de AES objetivo, hemos hecho uso del protocolo SimpleSerial, que es una interfaz de comunicación entre un dispositivo objetivo (como puede ser un microcontrolador) y un host (ordenador). La comunicación SimpleSerial implica enviar y recibir comandos entre el dispositivo objetivo y el host. Dichos comandos pueden utilizarse para configurar el dispositivo objetivo y realizar operaciones específicas en él. La documentación oficial de ChipWhisperer ofrece una guía detallada sobre cómo implementar y aprovechar al máximo este protocolo en experimentos de seguridad con dispositivos ChipWhisperer. Se puede encontrar más información al respecto en https://chipwhisperer.readthedocs.io/en/latest/simpleserial.html.

IV. Descripción del ataque

En esta sección describiremos los ataques que hemos llevado a cabo para la determinación de la clave secreta de cada uno de los procesos de cifrado de AES que hemos

analizado. Tomaremos como referencia el AES-128, que luego extenderemos a las otras dos versiones de AES. Hemos de señalar que los ataques llevados a cabo han sido al texto claro para las tres versiones de AES y al texto cifrado solo para el AES-128.

Inicialmente se genera un clave objetivo, que es la que hipotéticamente va a utilizar el AES-128 o que se supone almacenada en el dispositivo:

$k =$ `63 ba 46 9e 8c 5f 64 fb 4b af 17 00 80 6d e2 f5.`

A continuación se considera un conjunto de mensajes o textos en claro, $\mathcal{M}$, y su correspondiente conjunto de textos cifrados, $\mathcal{C}$. El número de mensajes que se van a utilizar determina el número de trazas que se van a capturar, dado que será una traza por mensaje. En nuestra experiencia la elección de ciertos mensajes o la definición de la clave no son factores que supongan un detrimento en el rendimiento del ataque.

Existen diversas estrategias que analizan el número de trazas mínimo necesario para atacar una implementación [6], que dependen de la implementación que se toma como objetivo. En nuestro caso experimental, hemos optado por garantizar el éxito del ataque calculando entre 50 y 75 trazas para cada clave (o subclave). El dispositivo de captura, Chipwhisperer-*lite*, permite definir el número de puntos capturados en cada traza. En el caso del ataque CPA al texto claro hemos considerado que 5000 puntos por traza es un número suficiente como para recuperar la clave.

En el caso del ataque al texto cifrado al tratar de obtener todas las subclaves utilizadas es necesario capturar la traza completa del algoritmo. La implementación de AES que se ha utilizado en este trabajo requiere de un tiempo mayor al tiempo de captura que permite el Chipwhisperer-*lite*, por lo que hemos aplicado un retraso para obtener varias partes de la traza completa y luego unir estas partes para determinar la traza completa del algoritmo.

Hemos llevado a cabo el ataque incrementando el número de muestras hasta 20000, lo que facilita el cálculo de la correlación en las partes en las que una ronda de AES se ve interrumpida por la limitación temporal comentada de Chipwhisperer-*lite*. Para obtener una traza del cifrado completo hemos capturado inicialmente las tres primeras vueltas y la mitad de la cuarta haciendo uso de la configuración estándar del capturador. Posteriormente hemos incluido un retraso en la captura de las 20000 muestras y con eso se obtienen las siguientes 3 rondas (y el final de la ronda que quedó incompleta previamente). En la parte final del descifrado, hemos incluido un retraso de 38000 muestras, obteniendo una traza con el consumo durante las dos últimas rondas estándar y la ronda final (recordamos que esta ronda lleva a cabo una función menos, lo que supone un menor proceso y así se ve reflejado en la traza). No hemos mantenido la misma cantidad que la vez anterior (produciendo una superposición en las medidas iniciales con las finales calculadas previamente), debido que si consideramos un retraso de 40000 muestras, la ejecución de SB en la octava vuelta queda cortada entre las trazas calculadas con un retraso de 20000 y 40000.

Al ejecutar el cálculo de la correlación, se obtienen diferentes valores para las correlaciones entre los bytes hipotéticos y los asociados a las trazas calculadas con antelación. En la Figura 5 se muestran las líneas (filas impares) que contienen los cinco mayores valores de la correlación obtenida para los diferentes bytes considerados, que se muestran debajo de estas líneas (en las filas pares). Se puede notar cómo el mayor valor de la correlación corresponde al primero de los bytes, que es precisamente el que se considera como parte de la clave maestra que se está buscando.

Por su parte, la Tabla I presenta los valores máximos de las correlaciones para cada uno de los bytes que, como se aprecia, son precisamente los bytes de la clave maestra original.

El ejemplo que se muestra corresponde al ataque CPA al texto claro, por lo que en el caso de que el ataque fuese al texto cifrado, habría que aplicar la función MC a la salida obtenida. En todo caso, veamos con más detalle cada uno de estos dos ataques.

La versión software de AES empleada en este estudio se ha obtenido de la web https://github.com/kokke/tiny-AES-c. Las principales razones para haber elegido esta en lugar de cualquiera de las otras muchas posibles se debe al uso previo de una versión adaptada de tiny-AES como parte del software disponible de ChipWhisperer, la posibilidad de trabajar con distintos parámetros de AES, a elección del usuario, y que la implementación no incluye contramedidas contra ataques por canal lateral.

IV-A. Ataques al texto claro

En el desarrollo de los diferentes ataques se van a recoger N trazas, una para uno de los mensajes utilizados, cada una de 5000 muestras. En general, el número de trazas necesario para que el ataque tenga éxito depende de la cantidad y calidad de las contramedidas que hayan sido consideradas de las implementaciones de AES que se consideren como objetivo.

El ataque CPA con acceso al texto claro se ha realizado para las tres versiones de AES, es decir, AES-128, AES-192 y AES-256. Analizaremos cada uno de estos casos en profundidad.

Es claro que el ataque más sencillo es el de AES-128, dado que es el que tiene una clave más pequeña. Para este caso, la clave se obtiene analizando la correlación entre el HW de la salida de la Sbox en la primera ronda y el consumo de potencia.

La operación inicial, $\mathsf{ARK}(m, k)$ se aplica al mensaje y a la clave original. Después de esta primera transformación, se lleva a cabo la operación SB, que se aplica a cada byte del estado. La expansión de la clave de AES funciona de manera que los primeros 16 bytes coinciden con los de la clave original, por lo que este análisis es suficiente.

Para establecer el modelo de fuga, se hace uso de los mensajes enviados al Chipwhisperer, para cada uno de los cuales se ha obtenido una traza del proceso de cifrado. El esquema del modelo de fuga se presenta en la Figura 6.

Por su parte, el proceso que se sigue es el siguiente:

- Como el análisis se hace para cada byte de la clave, se consideran los 256 candidatos posibles.
- A cada uno de los candidatos $\hat{k} \in \{0, 255\}$ se le aplica la función

$$\mathsf{SB}\left(\mathsf{ARK}\left(m_i^j, \hat{k}\right)\right), \tag{1}$$

donde m_i^j denota el j-ésimo byte del i-ésimo mensaje considerado, siendo $i = 1, \ldots, N$.

```
[0.9491990992316507, 0.5471390909222287, 0.5246838053610452, 0.5212479227723763, 0.5195567717466734]
['0x63', '0x8f', '0xd2', '0xa5', '0x7b']
[0.9251179226626082, 0.5625065284027929, 0.5502400689880526, 0.5372507102194368, 0.5331852945268827]
['0xba', '0xb1', '0xd2', '0xe', '0x9d']
[0.9185911593714917, 0.5923073012052694, 0.5884967883360923, 0.5656429688230191, 0.5240271072496835]
['0x46', '0x3e', '0xa', '0xe6', '0xad']
[0.9445978644602289, 0.6039028046127242, 0.5374221823693941, 0.5276425976420723, 0.5154902649384864]
['0x9e', '0xb1', '0xa3', '0x5', '0x55']
[0.9164817004043974, 0.5712154234656487, 0.5322241031574186, 0.5262292490933294, 0.5199201837031919]
['0x8c', '0x7f', '0xbb', '0xcd', '0x5e']
[0.9262740704653208, 0.5787228439861454, 0.5647962127008792, 0.5570420273368322, 0.5306666361862167]
['0x5f', '0x60', '0xbe', '0x57', '0x4c']
[0.9442099248262605, 0.5507352464423496, 0.5445476193608697, 0.5429895580513819, 0.5236134940403094]
['0x64', '0xb7', '0x54', '0x8', '0xf']
[0.9363726334070287, 0.5633206311847105, 0.5303185122869634, 0.5299177950548415, 0.5245866422028106]
['0xfb', '0x7f', '0x96', '0xd9', '0xcf']
[0.9135909590306666, 0.5248258753906456, 0.5234526169799818, 0.5184811187553051, 0.5093046596331867]
['0x4b', '0x37', '0x5e', '0x96', '0xa8']
[0.957094978242206, 0.5549824376108405, 0.535455358689448, 0.5012024155645869, 0.5005268639415531]
['0xaf', '0xb2', '0x6b', '0x13', '0x2d']
[0.8793496792285799, 0.5606341290207191, 0.5579495526024669, 0.5252015376278936, 0.5152344907517454]
['0x17', '0x97', '0x32', '0xd8', '0xb7']
[0.8934378677454463, 0.5732033508446834, 0.5334335215765476, 0.5300091129613276, 0.5265447037924181]
['0x0', '0x99', '0x37', '0x2b', '0x96']
[0.9277515779862964, 0.5446423614099105, 0.5299626197406594, 0.5112748682651395, 0.5107798913073338]
['0x80', '0xce', '0xd', '0x4c', '0x12']
[0.878293746321322, 0.5423715435014004, 0.5283979540987721, 0.5264482877771219, 0.5257979510779265]
['0x6d', '0x64', '0x17', '0x80', '0xbd']
[0.9433975592712099, 0.5594197769731447, 0.529482619956554, 0.5279913238895293, 0.5272042057823886]
['0xe2', '0x68', '0xe5', '0xf3', '0xfe']
[0.9102468335027226, 0.5448251425904077, 0.544568910952931, 0.5149594656370305, 0.5134594695168887]
['0xf5', '0xd1', '0xd3', '0x1f', '0x83']
```

Figura 5. Correlaciones asociadas a los diferentes bytes hipotéticos de la clave buscada

Mejor hipótesis	63	ba	46	9e	8c	5f	64	fb	4b	af	17	00	80	6d	e2	f5
Max. Correlación	0,94	0,92	0,91	0,94	0,91	0,92	0,94	0,93	0,91	0,95	0,87	0,89	0,92	0,87	0,94	0,91

Tabla I
VALORES DE LAS CORRELACIONES OBTENIDAS EN EL ATAQUE AL TEXTO CLARO CON AES-128

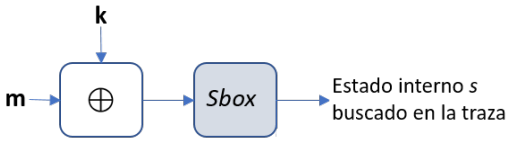

Figura 6. Esquema del modelo de fuga

- Por último se calculan los HW de las salidas obtenidas, resultando los datos que se van a utilizar en la correlación con las trazas capturadas previamente. Cada HW está asignado a un mensaje m_i, por lo que este se compara con la traza del cifrado de dicho mensaje.

Tomando los valores máximos de las correlaciones obtenidas, se espera obtener los bytes correctos de la clave objetivo.

IV-B. Extensión del ataque a AES-192 y AES-256

Como se ha mencionado previamente, la expansión de la clave de AES genera N_r+1 subclaves (incluyendo la utilizada en la transformación inicial), cada una de ellas de 16 bytes. En el caso de AES-128 la primera expansión consiste es la propia clave maestra porque consta de precisamente 16 bytes. Esta situación cambia en AES-192 y AES-256, dado que en ambos casos, las claves tienen longitud mayor: 24 y 32 bytes, respectivamente. Los primeros 16 bytes de las claves maestras para ambos casos siguen siendo las correspondientes a la primera expansión de la clave; mientras que los bytes restantes pasan a formar parte de la segunda expansión de la clave.

Para realizar el ataque mencionado contra la versión de AES-192 hemos procedido según el siguiente razonamiento: se obtienen los primeros 16 bytes de la clave (primera expansión de la clave) de la misma forma que con AES-128. Posteriormente, se utilizan los bytes obtenidos para calcular el estado interno que precede a la siguiente aplicación de la función ARK, esto es,

$$s = \mathsf{MC}\left(\mathsf{SR}\left(\mathsf{SB}\left(\mathsf{ARK}\left(m, k_0\right)\right)\right)\right).$$

Entonces se ejecuta de nuevo el cálculo de la correlación, con el mismo modelo de fuga, pero considerando los estados calculados en vez de los mensajes originales para la función

$$\mathsf{SB}\left(\mathsf{ARK}\left(m_i^j, \hat{k}\right)\right).$$

Se obtienen de ese modo 16 bytes, que constituyen la segunda expansión de la clave, 8 de los cuales completan la clave de cifrado introducida al algoritmo. Si se ataca de la misma forma a una implementación de AES-256 se tienen los 16 bytes restantes de la clave de cifrado, completándola también.

IV-C. Ataque al texto cifrado

Si bien el ataque CPA contra el texto claro es muy eficaz si no se protege la implementación de AES que se toma como objetivo, dota de una considerable ventaja al atacante, ya que

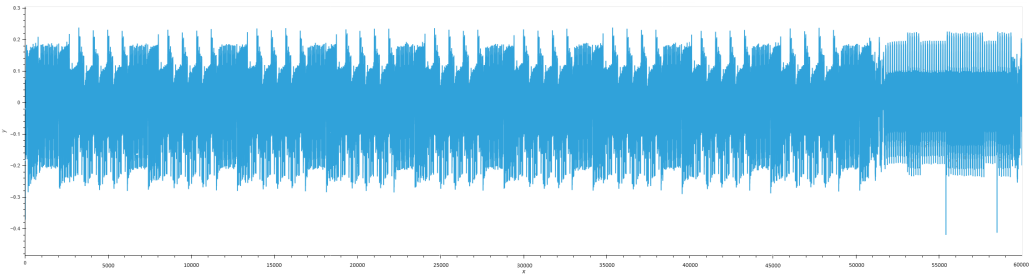

Figura 7. Traza capturada en la ejecución del descifrado de AES-128

este tiene acceso a los mensajes que se cifran y a la fuga de consumo de potencia del cifrado correspondiente. Es por eso que nos hemos planteado un nuevo escenario en el que el atacante no tiene acceso a los textos claros sino solo a los textos cifrados, pero manteniendo el acceso a las trazas [6].

En la Figura 7 se muestra una traza capturada en el proceso de descifrado de AES-128 cuando solo se tiene acceso a los textos cifrados.

Este nuevo ataque consta de varias similitudes con el desarrollado anteriormente, tanto en su planteamiento como en su ejecución. En efecto, el modelo de fuga sigue siendo el de HW de la salida, pero ahora se debe tener en cuenta la función Sbox correspondiente, esto es, Sbox^{-1}, dado que se está llevando a cabo el proceso de descifrado. Así pues, se calcula la correlación entre estos valores y las trazas del consumo de potencia durante el cifrado.

Por otra parte, debe tenerse en cuenta que como se está realizando el proceso inverso al de cifrado, la expansión de la clave debe ser la correspondiente para este proceso, esto es, al iniciar el descifrado se introduce en la función ARK la última subclave, no la primera. Si se considera el caso de AES-128 como ejemplo (los ataques para las otras dos versiones de AES funcionarán de forma análoga), el descifrado consta de 10 rondas y la clave de cifrado, que es la primera expansión de la clave, se opera con un XOR con el estado interno después de la ronda 10.

Aplicando el análisis de correlación a la salida de la Sbox^{-1} que se ejecuta en la primera vuelta del descifrado, se puede obtener k_{10}, la última subclave. Si denotamos por x_{10-i} la salida de la i-ésima aplicación de la función SB^{-1}, que son los valores sobre los que se calcula la correlación, y si k_{10-i} es la i-ésima subclave, con $i = 1, \ldots, N_r$. De modo similar a los ataques a AES-192 y AES-256 se calcula entonces el estado interno:

$$x_{10} = \mathsf{SB}^{-1}\left(\mathsf{SR}^{-1}\left(\mathsf{ARK}\left(m, k_{10}\right)\right)\right).$$

La Figura 8 muestra el esquema de la primera ronda del descifrado de AES, señalando el lugar donde se define el valor de x_{10}.

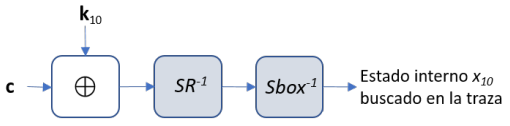

Figura 8. Esquema de la primera ronda del AES, donde se define x_{10}

Al generar los bytes de x_{i-1} para $i = 10, \ldots, 0$, en el cálculo de la correlación, se puede ignorar la función SR^{-1} dado que la operación se lleva a cabo a nivel de bytes. Los bytes del estado x_i se consideran de forma ordenada y por lo tanto la salida que se obtiene mantiene dicho orden (este paso puede requerir revisión en la implementación inicial del ataque). Una vez se ha conseguido la clave k_{i+1}, sí que es necesario tener en cuenta la función SR^{-1} para alcanzar el estado x_{i+1}.

Por su parte, la Figura 9 presenta una traza de parte del proceso de descifrado de AES-128, en la que se ha destacado la función de sustitución de bytes SB^{-1}, la inversa de SB. El zoom que se ha hecho en esta figura muestra, de forma ampliada, dónde se encuentran, en la traza representada, las 16 aplicaciones de la sustitución a todos los bytes del estado interno que se introduce como entrada de la función SB^{-1}. Cuando se lleva a cabo el análisis de correlación, se busca encontrar los valores de los bytes de la clave para los cuales la salida de la siguiente Sbox (o Sbox^{-1}) maximicen la correlación con los puntos de la traza, en particular, estos máximos son los mostrados en esta figura.

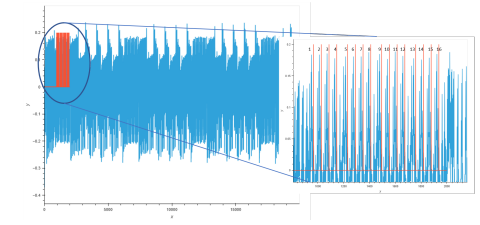

Figura 9. Localización de la Sbox^{-1} en una traza del descifrado y zoom

En este punto, si se considerara la misma estrategia que en AES-192 o AES-256 para calcular x_9, nos aparecería un problema añadido, dado que al tener que aplicar la función MC^{-1} no se pueden considerar los candidatos a bytes de la clave de forma independiente, ya que esta función involucra a 4 bytes del estado en el valor de cada byte de salida. Dicho de otro modo, se tiene que

$$x_9 = \mathsf{SB}^{-1}\left(\mathsf{SR}^{-1}\left(\mathsf{MC}^{-1}\left(\mathsf{ARK}\left(x_{10}, k_9\right)\right)\right)\right).$$

Así, se pasaría de tener que analizar 2^8 posibles candidatos a 2^{32}, lo que es un coste demasiado alto para realizar el ataque. Este problema se evita, como se describe en [8] y se demuestra en [9], considerando que la función MC^{-1} es, por definición, una función lineal, por lo tanto se tiene que

$$x_9 = \mathsf{SB}^{-1}\left(\mathsf{SR}^{-1}\left(\mathsf{MC}^{-1}\left(x_{10} \oplus k_9\right)\right)\right)$$
$$= \mathsf{SB}^{-1}\left(\mathsf{SR}^{-1}\left(\mathsf{MC}^{-1}\left(x_{10}\right) \oplus \mathsf{MC}^{-1}\left(k_9\right)\right)\right) \quad (2)$$

La expresión (2) coincide con el modelo de fuga previamente planteado, como se ve en la expresión (1). La subclave k_9 no se obtiene aplicando la correlación, sino que se obtiene $\mathsf{MC}^{-1}(k_9)$, siendo la verdadera k_9 fácilmente recuperable aplicando al función MC una vez adicional.

Repitiendo este proceso es posible obtener todas las subclaves. En particular, la subclave k_0, que para AES-128 es la clave maestra del algoritmo, no es recuperable con un análisis de la correlación. Ello se debe a que no se aplica la función SB^{-1} después del correspondiente $\mathsf{ARK}(\cdot, k_0)$. Para este último paso se puede aplicar la función inversa de la expansión de la clave con la información ya recuperada.

En nuestro estudio solo hemos aplicado este ataque a implementaciones de AES-128, pero la estrategia explicada es fácilmente aplicable tanto al AES-192 como al AES-256, ya que solo requeriría aumentar el número de iteraciones.

En todo caso, conviene señalar que para llevar a cabo con éxito este ataque, hemos considerado las diferencias en la programación entre C y Python. En efecto, dado que el algoritmo ejecutado en ChipWhisperer está escrito en lenguaje C, mientras que las órdenes ejecutadas para realizar el ataque están codificadas en Python, para lograr altos valores de correlación es fundamental que los datos utilizados para buscar los patrones de consumo de potencia (HW) estén en el mismo formato que los datos ejecutados en el ChipWhisperer. Por consiguiente, se han debido realizar los ajustes pertinentes de formato y transposiciones de los datos de entrada a las funciones del código ejecutado en Python.

V. Conclusiones

Nuestro objetivo final, del que este trabajo es una primera parte con resultados interesantes y prometedores, es disponer de herramientas para evaluar y mejorar la seguridad de las implementaciones de algoritmos criptográficos. Debido a su importancia y uso generalizado, hemos escogido a AES como primer objeto de análisis y hemos comenzado nuestro estudio reproduciendo distintas versiones de un ataque CPA.

Basándonos en el ataque inicial efectivo realizado contra AES-128, hemos considerado su extensión para las versiones de AES con claves de 192 y 256 bits.

Por otro lado, se ha llevado a cabo experimentalmente la versión del ataque CPA contra el texto cifrado, que dota al atacante de bastante menos información y recursos.

Como trabajo futuro, pretendemos completar el trabajo realizado llevando a cabo ataques de descifrado para AES-192 y AES-256, así como ampliar esta línea de investigación a otros algoritmos, como ASCON. También pretendemos realizar ataques similares, y de ser posible con el mismo hardware y software, a versiones de AES dotadas de diferentes contramedidas.

Finalmente, deseamos ampliar una línea de investigación que hemos iniciado realizando estudios sobre la seguridad de determinados algoritmos postcuánticos. Hasta la fecha hemos analizado diferentes tipos de ataques por canal lateral contra los algoritmos postcuánticos CRYSTALS-Kyber y CRYSTALS-Dilithium cuando se basan en técnicas de Inteligencia Artificial [10]. Pretendemos continuar con un análisis similar al realizado en este trabajo con las mismas herramientas software y hardware contra los algoritmos postcuánticos mencionados.

Agradecimientos

Este trabajo ha sido parcialmente financiado por la Agencia Estatal de Investigación (AEI) del Ministerio de Ciencia e Innovación (MCIN) y por la Unión Europea NextGerationUE/PRTR en los proyectos P2QProMeTe (PID2020-112586RBI00/AEI/10.13039/501100011033, cofinanciado por el Fondo Europeo de Desarrollo Regional: FSE, FEDER y FEDER, UE); y QURSA (TED2021-130369BC33, MCIN/AEI/10.13039/501100011033); en parte por el Proyecto ORACLE (PCI2020-120691-2, MCIN/AEI/10.13039/501100011033) y en parte por el programa de investigación e innovación Horizonte 2020 de la UE, proyecto SPIRS (Grant Agreement N° 952622). M.A.G.T. y L.H-A. desean agradecer su apoyo a los proyectos del CSIC EFiDiP y SAIACAP, respectivamente.

Referencias

[1] P.C. Kocher, “Timing attacks on implementation of Diffie-Hellman, RSA, DSS, and Other Systems,” in *Advances in Cryptology - CRYPTO’96, Lecture Notes Comput. Sci.*, vol. 1109, 1999, pp. 104–113, https://doi.org/10.1007/3-540-68697-5_9

[2] D. Boneh, R.A. DeMillo, R.J. Lipton, “On the importance of checking cryptographic protocols for faults”, in *Theory and Applications of Cryptographic Techniques - EUROCRYPT’97, Lecture Notes Comput. Sci.*, vol. 1233, 1997, pp. 37–51, https://doi.org/10.1007/3-540-69053-0_4

[3] A. Fúster Sabater, L. Hernández Encinas, A. Martín Muñoz, F. Montoya Vitini, J. Muñoz Masqué, “Criptografía, protección de datos y aplicaciones. Guía para estudiantes y profesionales,” RA-MA, Madrid, 2012, 364 pp., https://www.ra-ma.es/libro/criptografiaproteccion-de-datos-y-aplicaciones-una-guia-para-estudiantes-y-profesionales_48492/

[4] P.C. Kocher, J. Jaffe, B. Jun, “Differential power analysis,” in *Advances in Cryptology - CRYPTO’99, Lecture Notes Comput. Sci.*, vol. 1666, 1999, pp. 388–397, https://doi.org/10.1007/3-540-48405-1_25

[5] E. Brier, C. Clavier, F. Olivier, “Correlation power analysis with a leakage model,” in *Cryptographic Hardware and Embedded Systems - CHES 2004, Lecture Notes Comput. Sci.*, vol. 3156, 2004, pp. 135–152, https://doi.org/10.1007/978-3-540-28632-5_2

[6] C. O’Flynn, Z. Chen, “Side Channel Power Analysis of an AES-256 Bootloader,” in *2015 IEEE 28th Canadian Conference on Electrical and Computer Engineering (CCECE)*, 2015, pp. 750–755, https://doi.org/10.1109/CCECE.2015.7129369

[7] NIST, “Advanced Encryption Standard,” *National Institute of Standard and Technology, Federal Information Processing Standard Publication, FIPS 197*, 2001, http://csrc.nist.gov/publications/fips/fips197/fips-197.pdf

[8] M. Neve, K. Tiri, “On the complexity of side-channel attacks on AES256 – Methodology and quantitative results on cache attacks,” Cryptology ePrint Archive, Report 2007/318, 2007, https://eprint.iacr.org/2007/318.pdf

[9] A. Moradi, M. Kasper, C. Paar, “Black-box side-channel attacks highlight the importance of countermeasures” in *Topics in Cryptology - CT-RSA 2012*, Lecture Notes Comput. Sci. 7178, 2012, pp. 1–18, https://doi.org/10.1007/978-3-642-27954-6_1

[10] L. Hernández-Álvarez, M.A. González de la Torre, E. Iglesias Hernández, L. Hernández Encinas, “How to attack a galaxy: from Star Wars to Star Trek,” in *2023 Congress in Computer Science, Computer Engineering, & Applied Computing (CSCE)*, 2023, 2347–2354, https://doi.org/10.1109/CSCE60160.2023.00381

[11] V. Tiwari, S. Malik and A. Wolfe, “Power analysis of embedded software: a first step towards software power minimization,” in *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 2, no. 4, pp. 437-445, Dec. 1994, https://doi.org/10.1109/92.335012

[12] P. Bottinelli, Bos. Joppe, “Computational aspects of correlation power analysis,” in *Journal of Cryptographic Engineerings*, vol. 7, https://doi.org/10.1007/s13389-016-0122-9