Universidad Zaragoza
1542

Trabajo Fin de Grado

Funciones no clonables físicamente para IoT basadas en acelerómetros MEMS comerciales

Autor

Sergio Pardina Quirós[I]

Directores

Miguel García Bosque [II]

Guillermo Díez Señorans[III]

FACULTAD DE CIENCIAS, FÍSICA
Junio 2021

[I] 761224@unizar.es
[II] mgbosque@unizar.es
[III] gds@unizar.es

AGRADECIMIENTOS

A mis directores Miguel y Guillermo, por introducirme a este tema de investigación tan interesante y sobre todo por su tutela, constante apoyo y casi infinita paciencia.

Al departamento de electrónica por el buen trato y ambiente positivo que ha hecho que me haya sentido bienvenido desde el primer día.

Índice general

RESUMEN

En este Trabajo de Fin de Grado se realiza un estudio sobre la naturaleza de las PUFs (Physical Unclonable Function) y su posible aplicación en ecosistemas de comunicaciones como el IoT (Internet of Things) para mejorar la seguridad de los mismos.

Se busca estudiar la existencia de un comportamiento compatible con el desarrollo de PUFs para sensores frecuentemente encontrados en dispositivos al IoT. En concreto, para este trabajo el estudio se realizará para MEMS (sistemas microelectromecánicos del inglés: *microelectromechanical systems*), en concreto acelerómetros. Si se demuestra que efectivamente su comportamiento es compatible con el desarrollo de PUFs se intentará una PUF para este tipo de sensores y se estudiará su validez para ser implementada.

Objetivos El objetivo central de este trabajo es proponer nuevas PUFs que sean adecuadas para IoT. Concretamente en este caso acelerómetros MEMS.

Se definirán posibles PUF con la que estudiar las respuestas de los sistemas y si son válidas y se adecúan para su empleabilidad en IoT entonces se estudiará su tasa de error y otros aspectos que puedan influenciar la PUF de aquellas que sea factibles de implementar.

Capítulo 1

Introducción

Si bien el internet de las cosas o *Internet of things* es entendido por el público más general como las comunicaciones entre dispositivos inteligentes como por ejemplo: Smart-Tvs, dispositivos de telefonía móvil y demás dispositivos pertencientes a las denominadas "cosas inteligentes" la realidad abarca un campo mucho más mayor. Definimos el internet de las cosas como el nombre dado a la red formada por los diferentes elementos o dispositivos que tienen conexión a Internet y pueden comunicarse mediante la red. Tenemos entonces que un objeto cualquiera físico o software (*thing*) que es capaz de conectarse a Internet y comunicarse e intercambiar información con otro dispositivo a través del internet forma parte del IoT.

Es innegable que cada año el mundo está más interconectado. Este grado de interconectividad está aumentando a grandes velocidades debido a la proliferación de elementos capaces de conectarse a internet y formar parte del IoT.

Es entonces una cuestión de especial interés el estudio y avance de de la criptografía para así asegurar que los intercambios de información de los dispositivos que componen esta red sean lo más seguros posibles y evitar posibles brechas de seguridad que pudiesen causar efectos perjudiciales.

1.1. Criptografía

Tal y como se define en [1] la criptografía es el estudio de las técnicas matemáticas relacionadas con la seguridad de la información. El objetivo de la criptografía es asegurar que en un intercambio de información se logre confidencialidad (la información solo puede ser accedida por aquellos autorizados), integridad (evitar la posibilidad de manipular los datos por terceros) y autenticación (verificación de la autoría de un mensaje y la identidad del receptor).

Como su definición indica, la criptografía se ha considerado hasta la actualidad como un campo puramente matemático. Esta definición asume que en un intercambio

de información se consideran tanto emisor como receptor como cajas negras y el único enlace que puede ser atacado es el mismo mensaje. Es decir, el atacante solo se centra en el mensaje encriptado. De ahí que para asegurar que un mensaje es seguro se estudie si el método de encriptado es lo suficientemente robusto desde un punto de vista matemático.

Si bien esta concepción como caja negra de los participantes ha permitido el desarrollo de algoritmos de encriptado tremendamente refinados desde un punto de vista matemático, no es suficiente. Cualquier persona con suficientes conocimientos básicos de Internet sabe que la mayoría de ataques para hacerse con control de información de terceros se hacen centrándose sobre el usuario (*phising*). Esto ilustra que si el método de encriptado de un mensaje es lo suficientemente robusto el método de ataque simplemente cambiará el objetivo del ataque a los partícipes de la comunicación. Es decir, se ataca el elemento más fácil de explotar. Un sistema es entonces, tan fuerte como su elemento más débil. En nuestro caso al estar el mensaje completamente reforzado por el encriptado, el atacante pasará a atacar el mensaje mediante métodos que tienen como objetivos los emisores y receptores.

Algunos ejemplos de este tipo de ataque son el *fault-attack* [2] en el cual se ataca un dispositivo electrónico sometiéndolo a un esfuerzo externo con el objetivo de provocar errores que den lugar a fallos de seguridad del sistema o el *side-channel attack* [3] donde se observan parámetros del sistema que alberga las comunicaciones como el consumo de potencia o el tiempo de ejecución y a partir de un estudio exhaustivo de estos parámetros se intenta atacar.

Tenemos entonces que expandir el ámbito de la criptografía al conjunto de técnicas que tienen como objetivo frenar un amplio abanico de posibles ataque para garantizar una comunicación segura.

Esta situación ha provocado que en los últimos años las PUFs hayan cobrado gran importancia como posible solución. Fueron introducidas por primera vez por Pappu en un sistema óptico [4]. Como se explicará posteriormente, algunas de las funciones de una PUF son la autenticación, identificación y generación de claves aleatorias. Que son tres de las funciones buscadas por la criptografía para mantener un nivel aceptable de seguridad en comunicaciones.

1.2. IoT

El Internet de las Cosas o IoT es un término muy comúnmente empleado y que en los últimos años ha aparecido constantemente. Según la Unión Internacional de Telecomunicaciones [5] el IoT se define como la infraestructura global para la sociedad

de la información permitiendo así servicios avanzados interconectando de forma tanto física como virtual objetos basadándose en el avance de las ciencias de la tecnología y la comunicación.

Por esta definición hay una miríada de posibles campos que pueden beneficiarse del IoT y una miríada de objetos físicos o virtuales que forman parte de esta red. Al haber una tan grande variedad de estos objetos surge la necesidad de asegurar que las comunicaciones realizadas entre elementos interconectados en el IoT sean seguras. Esto no está exento de desafíos, pues por la gran diversidad de estos objetos surgen necesidades para poder acomodar el funcionamiento seguro de todos estos. Si bien nos encontraremos con dispositivos como ordenadores conectados a IoT también jugarán un papel vital dispositivos cuyo tamaño no supere el tamaño de un dedo (y en muchas ocasiones no sea ni visible por el ojo humano). Estos dispositivos necesitarán que su método de asegurar comunicaciones tenga el menor consumo de potencia y el menor tamaño posibles.

Así pues se intenta en este trabajo proponer PUFs compatibles con estas condiciones de los dispositivos. Por tanto intentaremos proponer PUFs que debido a su bajo consumo de potencia sean adecuados para el IoT. Para ello intentaremos proponer componentes ya pertenecientes al dispositivo. Lo que además acarreará un una reducción de costes. Un ejemplo de estos dispositivos son los acelerómetros.

Capítulo 2

PUFs

Uniendo la búsqueda por aplicaciones a la respuesta de sensores a la rama de la criptografía nos da lugar a la concepción de las PUF.

2.1. Definición

Definiremos una PUF como **PUF**:Physical Unclonable Function

Pasemos a realizar un breve análisis de cada una de las palabras que componen el acrónimo para darle más sentido:

- ***Physical***: Físico, al contrario que un algoritmo matemático, nuestra función tiene origen en un fenómeno físico mesurable y cuantizable.
- ***Unclonable***: Por no clonable hacemos alusión a lo propiedad de nuestra PUF que hace que sea imposible de replicar.
- ***Function***: Función. Según la definición matemática una función es una regla que asigna una serie de valores (input) a otros (output).

Podemos definir la PUF como un objeto físico que tiene una serie de posibles respuestas frente a posibles entradas (desafíos o *challenges*). Esta funcionalidad desafío-respuesta depende de las variaciones aleatorias e incontrolables que ocurren durante el proceso de fabricación. Debido a estas variaciones, distintas instancias de una PUF producirán respuestas diferentes para un mismo desafío. Esencialmente esto implica que las PUF es una forma de identificar un sistema electrónico a partir de un comportamiento único del mismo sistema. Estamos entonces adjudicando "huellas dactilares" para el sistema electrónico. Pero en vez de basarse en la forma de las yemas de los dedos, se basan en la forma de responder del sistema electrónico frente a una entrada o desafío.

2.2. Función de PUFs

En nuestro caso, la función de una PUF será identificar de forma unívoca un elemento único o nodo que forma parte del IoT. Idealmente se buscaría no solo la identificación si no también la autenticación. La identificación se suele conseguir a partir de un número o cadena de bits generado por la PUF. Es necesario que este número sea único al dispositivo a identificar, impredecible e imposible de replicar. Una vez generado este número por medio de la PUF, puede emplearse para desarrollar identificación, autenticación y generación de claves seguras.

2.3. Ventajas de PUFs

Actualmente el papel que podrían tomar las PUFs en la criptografía se realiza con resultados satisfactorios por medio de algoritmos criptográficos. Si bien cumplen su función, tienen debilidades que pueden ser explotadas. Las PUFs carecen de algunos de estos defectos teniendo ventajas con respecto al actual método de encriptado e identificación.

Algunas de estas ventajas son la reducción de costes. Mediante una PUF el mismo dispositivo es el encargado de identificación. Por lo general en caso de usarse los métodos matemáticos actuales se necesita generar la clave identificativa mediante un proceso externo y almacenarla en una memoria no volátil (NVM). En muchas ocasiones esto se hace añadiendo una memoria al dispositivo que almacenará la identificación del mismo. La necesidad de generar este número, que se empleará como base para el proceso de la identificación del dispositivo en un entorno externo y la adición de la memoria que lo guardaría constituye costes que serían eliminados fácilmente por medio de una PUF. La PUF está ubicada en el mismo dispositivo que buscamos manipular y genera el output por sí misma (en muchas ocasiones de forma pasiva), por lo que eliminamos todos los costes que conlleva el proceso de la generación del número empleado.

Otra ventaja que caracteriza a las PUFs es un aumento de la potencial seguridad. Al generar la información delicada en el mismo dispositivo reducimos opciones de ataque al eliminar todo el proceso de almacenado de la información en el dispositivo desde el exterior y la generación de la misma información en un lugar diferente al mismo dispositivo a caracterizar.

2.4. Desventajas de PUFs

Si bien las ventajas que conllevan la identificación, autenticación y cifrado de elementos mediante una PUFs tienen suficiente peso como para garantizar un interés

en su estudio, también existen aspectos negativos.

Algunos de estos negativos, son una falta de estandarización de los protocolos PUF. Además no se ha realizado tanta investigación y trabajo en las PUFs como en los diferentes algoritmos matemáticos que sustituirán al tratarse de una vía de investigación relativamente novedosa. Esto conlleva una falta de conocimiento y *know-how* en la aplicabilidad y el grado de refinación de los sistemas ideados.

También al basarse en fenómenos físicos y en mediciones de los mismos sistemas se ven más afectados al error introducido por la medida del fenómeno o las variaciones del entorno en el que se encuentra. Esto suele provocar que, a pesar de que hipotéticamente la respuesta de una PUF para un mismo desafío sería la misma siempre, en realidad nos encontraremos con un rango de medidas válidas asociado a la PUF.

2.5. Aplicaciones de PUFs

Las ventajas mencionadas, junto a varias otras permiten vislumbrar ya posibles aplicaciones de las PUFs en contextos reales.

Una posible aplicación es la lucha contra la falsificación. Se pueden desarrollar PUFs para dispositivos (o incluso medicinas) que permitan distinguir el origen de los mismos y así saber si se trata de una muestra genuina o una copia. Como por ejemplo relojes de alta gama. Se profundiza en esta posible aplicación en trabajos como [6].

Otra aplicación de interés comercial es en logística. Se pueden emplear PUFs asociadas a un RFID (Radio Frequency Identification). La función sería sustituir los códigos de barras. Los códigos de barras son muy baratos y se emplean para asignar ID a un grupo de productos. En el caso de emplear nuestra PUF se podría identificar cada producto de forma individual. Mediante la identificación y autenticación se permitiría una comunicación del producto con una base de datos central. Tras identificar el producto, se podría solicitar a la base de datos información como su fecha de caducidad, precio...

Esta capacidad de adquirir información específica de cada producto a partir del mismo producto tiene un gran potencial frigoríficos inteligentes, que conectándose con la base de datos podrían obtener un listado de los ingredientes que están enfriando y poder desarrollar posibles recetas, información nutricional o realizar de forma automática la lista de la compra.

Hay más aplicaciones, como la verificación de la licencia de canales de pago en una televisión o un control más cuidadoso del exceso de productos fabricados que luego son vendidos ilegalmente (un problema especialmente común con microchips).

El libro [7] explica, en su segundo capítulo, con bastante detalle como funcionarían

muchas de estas aplicaciones y más.

2.6. Funciones de una PUF

Las funciones que lleva a cabo una PUF son esencialmente tres:

Identificación: Esto es la asignación de un ID único a una entidad. Este ID debe estar asignado a un único elemento para evitar que se realicen identificaciones no correspondientes. La ventaja de una PUF con respecto a los métodos actuales sería eliminar el proceso de generación del ID de forma externa y la necesidad de añadirlo mediante una memoria, así se reducen costes.

Autenticación: La autenticación es el proceso de verificado del ID de un dispositivo. Generalmente esta autenticación se hace estudiando la respuesta de una PUF a un desafío. Durante la fase de autenticación una entidad (nuestra PUF) se identifica enviando su ID a un agente verificador. El agente verificador busca en la base de datos el ID y responde enviando un desafío a la entidad. El agente verificador conoce de antemano cual debería ser la respuesta de ese ID para el desafío pues es información que tiene almacenada en una base de datos. Entonces evaluará la respuesta de la entidad y si se acerca lo suficiente (es decir, no supera un umbral definido de autenticación, definido por el agente) se verifica la identidad. Que sea necesario que el valor se acerque lo suficiente y no sea el mismo exacto se debe a que, por lo general las PUFs no son ideales y las respuestas que generan suelen tener cierto grado de imprecisión asociada.

Generación y almacenamiento de claves: Es otra de las posibles funciones de una PUF. Para poder asegurarnos que una clave generada es segura esta debe ser impredecible antes de generarse. La mejor forma de asegurar impredecibilidad es generarla de forma aleatoria. Los ordenadores y humanos son malos sistemas porque los números que son capaces de generar no son aleatorios. La mejor forma de generar números aleatorios de verdad es a partir de fenómenos físicos. Siendo que una PUF se basa en un fenómeno físico entonces debería ser también capaz de generar claves de forma verdaderamente aleatorias. Esto puede verse si por ejemplo, los datos obtenidos se distribuyen de forma binomial o superar tests de aleatoriedad. En función del tamaño de la cantidad de desafíos a los que responde una PUF pueden definirse las PUF fuertes y PUF débiles. En su tesis doctoral [8] Maes define la fuerza de una PUF tras proporcionársela al adversario. Es decir le damos acceso total al adversario a estudiar nuestra PUF. Entonces una PUF fuerte es aquella que nos permite plantear un desafío cuya respuesta será con casi total certidumbre no predicha por el adversario. Para que esto ocurra es necesario que la cantidad de desafíos posibles de la PUF sea enorme y sea inviable crear un modelo predictivo de la respuesta. Una PUF débil no cumple esta

condición y es evidentemente menos segura que una PUF fuerte.

2.7. Estudios anteriores realizados de PUFs

Anteriormente se han realizado estudios de las PUFs generadas por osciladores de anillos como por ejemplo el realizado en [9] . Se trabaja con una matriz de osciladores.

El estudio se basa en la frecuencia de oscilación de cada oscilador de anillo. Se aplica el mismo desafío a cada uno de los osciladores de nuestra matriz de osciladores y se elige una serie de n parejas a comparar. A partir de este resultado se construye una palabra binaria sobre la que trabajará nuestra PUF. Para cada pareja se comparan las frecuencias y se asigna un bit. Por ejemplo "1" cuando el primer oscilador tiene mayor frecuencia y "0" en caso contrario.

Se obtiene entonces que a partir de la serie de parejas osciladores en anillo se genera una palabra de un número determinado de bits a partir de la cual se estudiarán las propiedades de la PUF. Esta palabra de "1" y "0" seguiría una distribución binomial.

2.7.1. Definiciones de interés para caracterizar una PUF

Si bien hay muchas expresiones de interés para describir el comportamiento estadístico de una PUF, las de mayor interés son la definición de la inter-distancia y la intra-distancia. Las definiremos de acuerdo a la tesis doctoral de Maes [8].

Inter-Distancia: Se define como la variable aleatoria que describe la distancia entre dos respuestas de diferentes PUFs sometidas al mismo desafío:

$$D_P^{inter} = \mathbf{dist}\left[Y\left(x\right); Y'\left(x\right)\right] \tag{2.1}$$

Intra-Distancia: Se define como la variable aleatoria que describe la distancia entre dos respuestas de mismo PUFs sometidas al mismo desafío:

$$D_P^{intra} = \mathbf{dist}\left[Y\left(x\right); Y\left(x\right)\right] \tag{2.2}$$

Como veremos a continuación, estas distancias pueden verse representadas mediante bits (como en el trabajo de fin de grado de Daniel Gil Marco [9])

En todos estos trabajos se parte de una respuesta que consiste en una palabra de bits, así se definen los siguientes parámetros: Tenemos por ejemplo distancia Hamming para dos palabras de n bits. Sea b_i el bit i-ésimo de la primera palabra y x_i el bit i-ésimo de la segunda palabra:

$$HD = \sum_{i=0}^{n-1} b_i \oplus x_i \tag{2.3}$$

La **distancia Hamming** no es entonces más que la cantidad de bits diferentes dos palabras de n bits. Si las palabras se originan de estudiar la respuesta ante un mismo desafío por la misma PUF en dos ocasiones, la HD indica la intra-distancia. Si se obtiene a partir de las respuestas de dos PUFs diferentes frente a un mismo desafío se trata de inter-distancia.

En lo referente a la identificación de sistemas, encontramos que varía la validez de una PUF en función del umbral que consideramos correcto. Esto se ha explicado antes en la sección 2.6, se elige un umbral y si la distancia es inferior al umbral se acepta que provienen de la misma PUF.

FAR: False acceptance rate o índice de falsa aceptación. Esto nos relaciona la facilidad que tiene nuestra función no clonable de aceptar una respuesta no correspondiente a nuestra PUF como válida. Se define matemáticamente como:

$$FAR = \frac{FA}{TE} * 100 \tag{2.4}$$

Donde FA es el número de falsas aceptaciones y TE es el número total de evaluaciones realizadas. Su significado no es más que la probabilidad de que la inter-distancia sea menor o igual al valor umbral definido.

FRR: False Rejection Rate o índice de falso rechazo. Esto es la facilidad que tendrá nuestro algoritmo de rechazar y considerar como errónea una respuesta válida. Se define como:

$$FRR = \frac{FR}{TE} * 100 \tag{2.5}$$

Donde FR es el número de falsos rechazos y TE es el número total de evaluaciones realizadas. Se trata de la probabilidad de que la intra-distancia sea superior al valor umbral definido.

Como se verá en el desarrollo del trabajo FAR y FRR dependerán del umbral elegido. Además cuando aumente FRR disminuirá FAR y vice-versa. Para minimizar esta tasas de error, se busca elegir elegir el valor donde coinciden, pues minimizará ambas.

Capítulo 3

Desarrollo y resultados

3.1. Descripción de los experimentos

En nuestro estudio se busca poder introducir las PUFs y sus respectivas aplicaciones al *Internet of Things*, que recordemos la hemos definido anteriormente en 1.2. Por tanto proponemos que nuestro sistema estudiado se base en sensores relativamente comunes a lo largo de dispositivos muy usados. Así podrían reutilizarse elementos ya presentes en el dispositivo. Se ha decidido elegir acelerómetros. Pues son sensores que se encuentran integrados en algunos de los aparatos más usados de forma global. Los teléfonos móviles, automóviles, relojes... Nuestro estudio se ha realizado con acelerómetros **ADXL335**[10].

Se trata de un MEMS que mide las aceleraciones mediante el desplazamiento relativo de un condensador diferencial. Tiene 3 salidas, pues mide independientemente la aceleración en X, Y y Z.

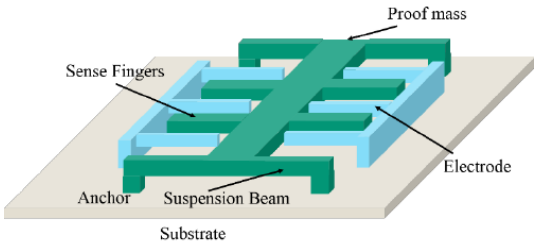


Figura 3.1: Esquema de condensador diferencial, obtenido de [11]

Sabido que la gravedad y el voltaje de salida son proporcionales trabajando con el módulo del voltaje de salida trabajamos indirectamente con la gravedad. Se ha calculado la norma de las señales medidas por cada uno de los ejes. De esta forma se evita realizar un estudio independiente para cada uno de los ejes y que pequeñas rotaciones de los acelerómetros introdujeran un error que pudiese confundirse con una señal correspondiente a fenómenos físicos externos.

Como fuente de alimentación se ha empleado una fuente de laboratorio que alcanza voltajes estables con muchas cifras significativas de precisión. La toma de datos se ha realizado midiendo los voltajes de salida de cada eje para un acelerómetro en reposo durante periodos de aproximadamente una hora.

3.2. Primera aproximación

La primera forma de atacar el problema propuesto en el trabajo podría ser realizar un estudio similar a los hechos a partir de osciladores de anillo como [12], [9] o [13] pero sustituyendo los osciladores de anillo por acelerómetros. Pero esto sería absurdo. Se requiere una matriz de muchos acelerómetros para poder desarrollar como output una palabra de suficientes bits que sea interesante de estudiar y permita estudiar el sistema de forma equivalente a [12], [9] o [13]. Si bien realizando el estudio con una matriz de acelerómetros podríamos aprovecharnos de las definiciones y resultados ya estudiados por [12], [9] o [13] esto no sería viable. No es factible en dispositivos que interactúan con el IoT ya que por lo general estos dispositivos (como móviles o relojes) tienen únicamente un acelerómetro y pretender que los fabricantes de estos dispositivos comiencen a introducir matrices de decenas o centenas de estos sensores no es práctico ni viable económicamente.

3.3. PUF propuesta: Individual

La forma de realizar nuestro estudio se basará inicialmente en la toma de medidas de un acelerómetro único incorporado al dispositivo que sea, sea un móvil, un vehículo...

Así pues, lo que se hará será tomar las medidas de varios acelerómetros en reposo para un mismo voltaje de entrada $V_{input} = 3{,}3000 \pm 0{,}0005V$. Realizaremos las tomas de medidas e intentaremos desarrollar una PUF analógica. En vez de estudiar una palabra de n bits, se buscará comprobar el comportamiento de PUF a partir de un output analógico. Tomaremos como output analógico de cada acelerómetro la medida que toma en reposo.

3.3.1. Datos y análisis de los mismos

Una vez hemos obtenido los datos de la señal en reposo (es decir, sometidos únicamente a g) de los diferentes acelerómetros (el estudio se ha realizado para 22 acelerómetros) se muestran los resultados en la Figura 3.2.

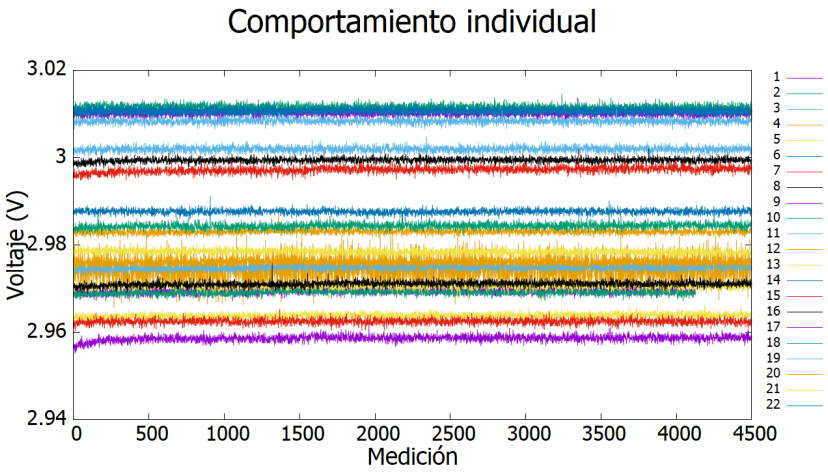


Figura 3.2: Voltaje respuesta medido en reposo para cada acelerómetro a lo largo de una hora.

Pasaremos a estudiar qué conclusiones pueden extraerse de estos experimentos para generar una PUF.

Diseño de una PUF

Como se ha mencionado antes, no realizaremos una PUF binaria. Es decir en nuestro caso no basaremos la PUF en una palabra binaria de X bits, si no que tomaremos como elemento central de nuestro estudio un valor numérico diferente.

Como se puede observar en la figura 3.2 en todos los casos las medidas se concentran en torno a un valor promedio, dispersándose a nivel individual de forma, a priori, aleatoria.

No podemos desarrollar una PUF refinada a partir de estos datos directamente. Es por ello por lo que nos decantamos por tomar el promedio de los datos para realizar el estudio a nuestra PUF. Para poder afirmar que las PUFs generadas tiene unicidad (i.e: no se repite una misma PUF para varios acelerómetros diferentes) lo primero que tendremos que analizar es que estos valores no se repiten. Tendremos que observar como efectivamente, cada señal corresponde a un único acelerómetro y no se repite la misma respuesta para dos acelerómetros diferentes

Nos encontramos con un resultado positivo (mostrado en Figura 3.3), pues nos confirma que la respuesta de cada acelerómetro es única. No coincide ningún valor para varios acelerómetros. En la gráfica esto no es apreciable con facilidad, pues el tamaño de los marcadores hace parecer que coinciden. Puede comprobarse el comportamiento haciendo zoom en las siguientes regiones (Figura: 3.4).

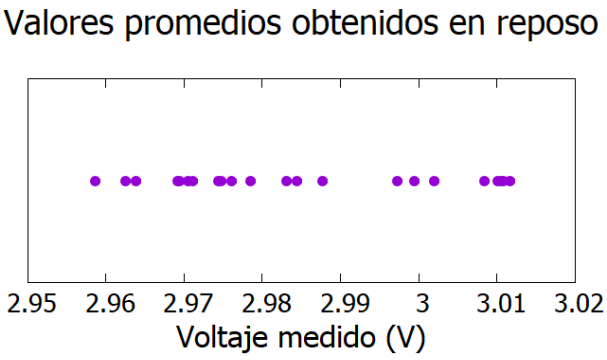


Figura 3.3: Voltaje respuesta promedio en reposo para cada acelerómetro.

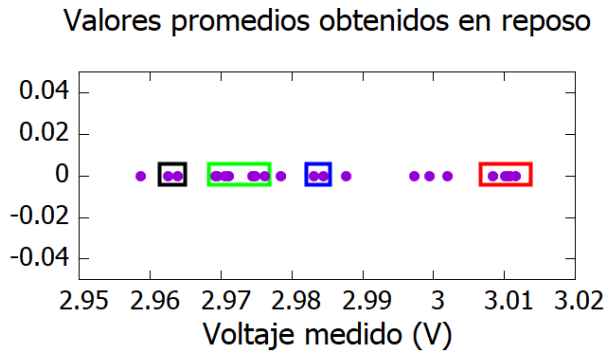


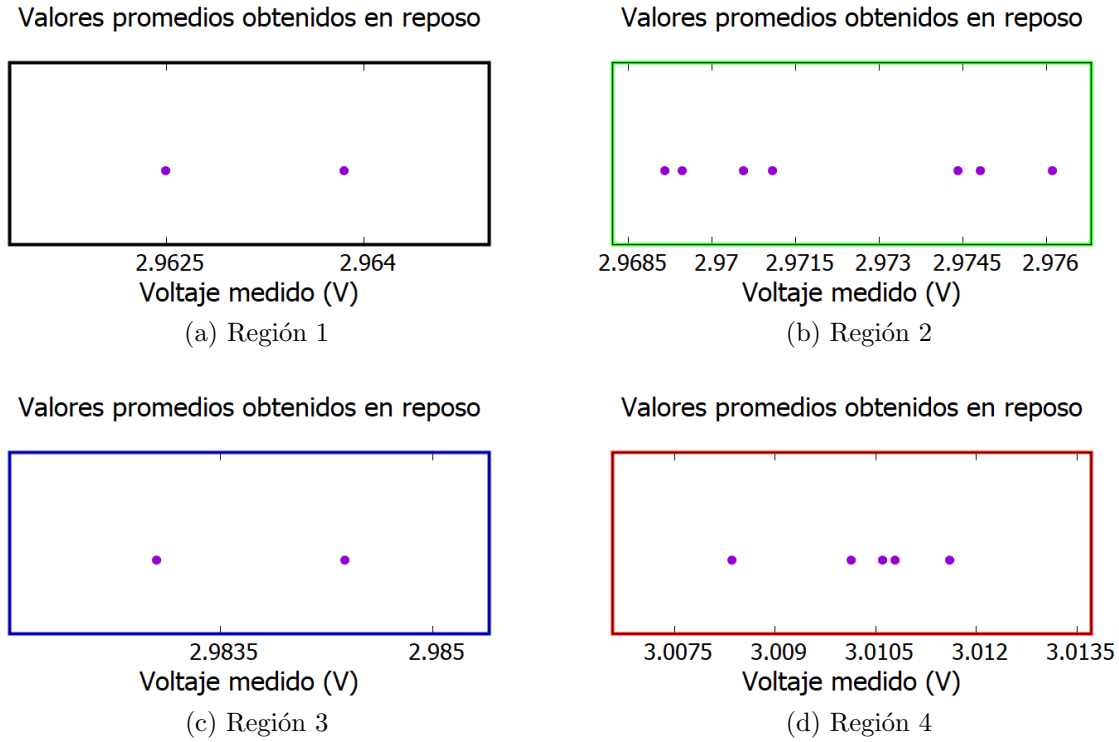
Figura 3.4: Regiones donde los valores promedio parecen coincidir.

Marcadas las regiones que más podrían hacer sospechar, mostramos cómo quedan los resultados tras hacer zoom en la Figura 3.5.

Demostramos así el comportamiento que buscábamos probar como objetivo. Cada acelerómetro devuelve un voltaje diferente (y por ende una gravedad distinta) a pesar de estar sometidos a las mismas condiciones (V_{in} y g) Los diferentes acelerómetros tienen comportamientos únicos y demuestran una unicidad que podría ser empleada para desarrollar PUFs. Puesto que cada acelerómetro mide un V_{out} (o de forma indirecta, una g) propio, un ID bastante bueno para cada acelerómetro sería este valor promedio único.

El hecho de que solo nos hayamos fijados en el valor del promedio se basa en la suposición de que es posible realizar una toma de datos durante un tiempo lo suficientemente largo como para que sea "infinito". Los datos medidos para cada acelerómetro se distribuyen con forma de distribución normal.

Por tanto si tomamos muestras durante $t \to \infty$ esto implica que $N \to \infty$ (siendo N el número de medidas realizadas). Al tener efectivamente un número infinito de medidas podríamos aprovecharnos del teorema del límite central y generarse una nueva distribución de números a partir de los promedios de n valores. Ya que nuestras medidas serían a efectos prácticos infinitas, los promedios podrían ser elegidos de n medidas



(a) Región 1

(b) Región 2

(c) Región 3

(d) Región 4

Figura 3.5: Zoom de las regiones marcadas para comprobar que efectivamente, los valores de los promedios no coinciden

siendo n lo suficientemente grande como para que la distribución binomial que surgiese sea lo suficientemente estrecha como para que la varianza sea despreciable y se anulase σ. Pasamos entonces a tener un comportamiento de PUF puntual. Marcado por el promedio.

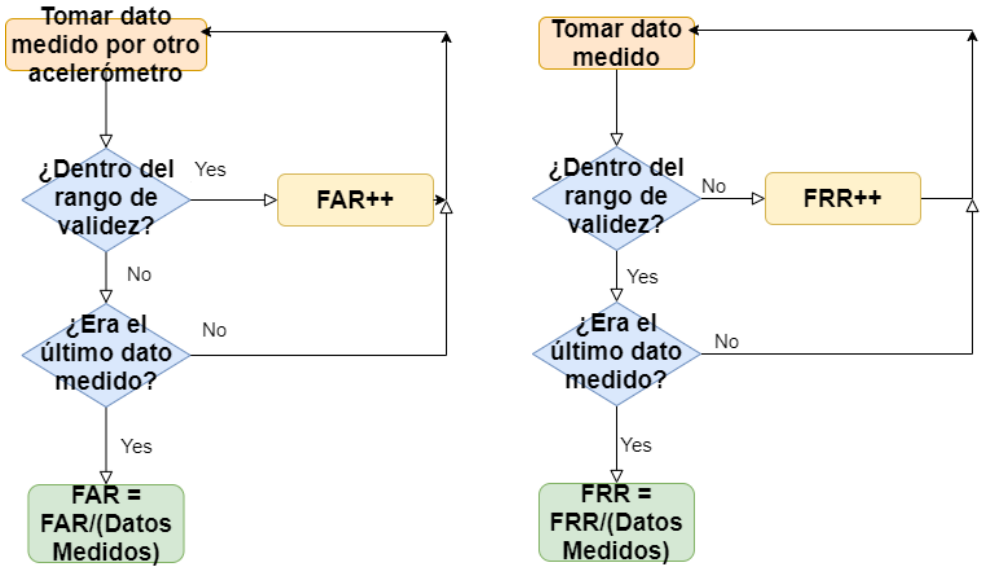
Aún así, desarrollar PUFs que requieran un tiempo de toma de medidas que tienda a infinito tiene un rango de aplicaciones válidas muy reducido, pues se pide que el elemento esté conectado a la red de forma constante y no varíe sus condiciones locales. Un posible elemento que se aprovecharía de esta idea serían por ejemplo los *Smart-fridge*.

Por otra parte, en la mayoría de casos no tenemos tiempo arbitrariamente largo para realizar la toma de medidas y poder emplear nuestra PUF para identificar/autenticar. No podemos aprovecharnos del teorema de límite central para disminuir la varianza de la función final. Tenemos que tener en cuenta entonces que los datos que irá midiendo nuestro acelerómetro se verán afectados por ruido aleatorio. Tendremos que tener en cuenta esta situación a la hora de caracterizar la PUF.

Una vez asignados los ID, la autenticación se basará en un rango marcado por el promedio y un umbral $\mu \pm \delta$. Entonces si una respuesta está a menos distancia que δ de μ (es decir, entra dentro del rango de identificación) se considerará que la señal

proviene de ese acelerómetro y se verificará el ID. Para cuantificar cómo de identificable es la PUF, emplearemos **FAR** y **FRR** que indicarán la probabilidad de que haya un error de identificación, lo que implica una verificación no válida.

FRR la calcularemos a partir de los datos medidos por un acelerómetro. Se definirá un rango de validez de respuestas tal que todos los valores medidos por el acelerómetro que estén fuera del rango $(\mu - \delta, \mu + \delta)$ (δ siendo el valor del umbral elegido) se contabilizarán como falso rechazo. Pues al estar fuera del rango se rechaza, pero ha sido medidos por el acelerómetro, por lo que deberían considerarse como correctos. De ahí falso rechazo.



(a) Algoritmo empleado para definir FAR

(b) Algoritmo empleado para definir FRR

Figura 3.6: Diagramas de flujo del algoritmo definido para calcular las curvas FAR y FRR

FAR la definiremos a partir de los datos obtenidos para todos los acelerómetros salvo el acelerómetro que buscamos estudiar. De nuevo consideraremos un rango de aceptación $(\mu - \delta, \mu + \delta)$ siendo δ el mismo valor que para FRR. Ahora evaluaremos todos los datos medidos por los demás acelerómetros. Si un valor medido por un acelerómetro diferente al que estamos estudiando entra dentro del valor de aceptación, lo consideraremos como falsamente aceptado. Esto es porque está dentro del rango, por lo que de evaluarse sin contexto, se consideraría como un valor esperado por el acelerómetro a estudiar, pero en realidad es la señal proveniente de un acelerómetro diferente.

Como FAR y FRR describen errores y comportamientos no ideales, buscamos minimizar ambos y hallar un valor δ donde ambos se minimicen. Idealmente se buscaría que las funciones de FAR y FRR en función de δ se cortasen en en 0, que es lo que

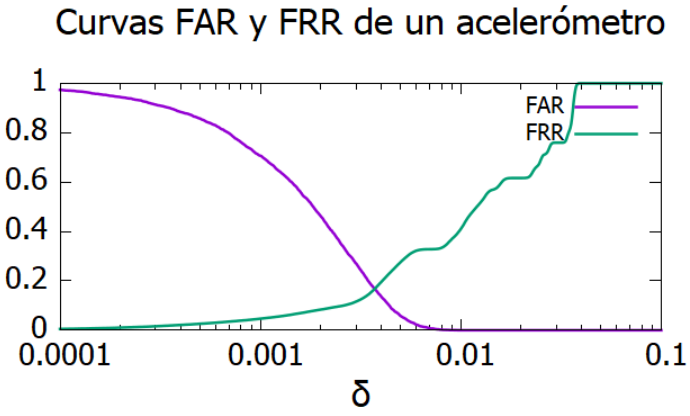


Figura 3.7: Curvas FAR y FRR

ocurriría en una PUF ideal. Esto implicaría que no se acepta ni se rechaza ningún valor que no sea correcto. Como se ve esto no ocurre para el acelerómetro estudiado en la Figura 3.7.

Observamos que el punto donde se cortan las gráficas es para un $\delta = 0{,}00372$ y ocurre para $y = 0{,}1675$. Se obtiene así valor de δ en el que se minimizan FAR y FRR tendremos que la posibilidad de leer un valor la asignación del mismo sea errónea (i.e., es válido y se asigna como no válido o bien es no válido y se asigna como válido) es de un 16,75 %

Puede observarse que efectivamente hay acelerómetros en los que sí ocurre que el corte se realiza en $y \approx 0$. Como por ejemplo en Figura 3.8. Mostramos entonces el

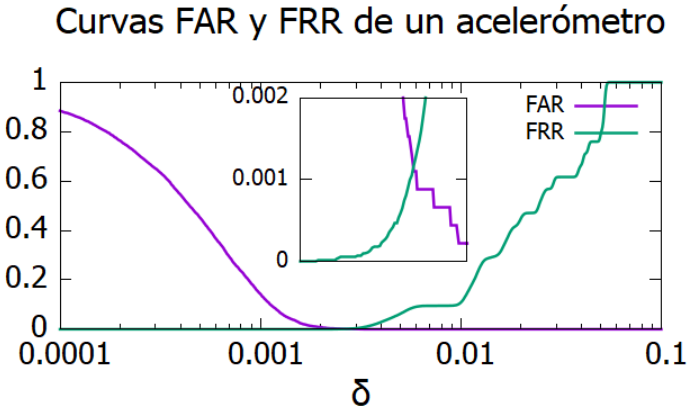


Figura 3.8: Curvas FAR y FRR

valor promedio del punto donde se cruzan FAR y FRR para todos los acelerómetros así como su varianza, para poder hacernos una idea de si δ se mantiene constante y la probabilidad es la misma en todos los acelerómetros:

$$\delta = 0{,}0015 \pm 0{,}0006\text{V} \qquad y = 0{,}04 \pm 0{,}04$$

Se observa a partir de estos resultados que sí hay un rango bien definido para δ en el cual se cruzan la mayoría de las curvas. Por el contrario el y en el que se cortan en muy disperso. Esto último nos indica que encontraremos acelerómetros en los que FAR y FRR intersecarán en valores muy bajos y acelerómetros en los que intersecarán en valores más elevados.

3.3.2. Análisis de los resultados

Tras realizar el estudio hemos comprobado uno de los objetivos principales del trabajo. En el caso de MEMS que como acelerómetros se observa que puede desarrollarse una PUF compatible con IoT. Es decir podrían desarrollarse las funciones de una PUF partir de la señal medida por cada acelerómetro en reposo. Cada acelerómetro mide para mismas condiciones un valor diferente y único.

3.4. PUF propuesta: Parejas

Los resultados obtenidos en la sección 3.3 nos han mostrado que efectivamente se puede desarrollar un comportamiento de PUF a partir de MEMS.

Si bien los resultados parecen ser prometedores, el estudio se ha realizado para los acelerómetros en reposo. Esta condición es muy limitante, especialmente para acelerómetros pertenecientes al IoT. Muchos de estos acelerómetros forman parte de sistemas más complejos, como dispositivos de telefonía móvil o vehículos. En la mayoría de ocasiones, los acelerómetros se encontrará en una situación en la cual se vean sometidos a aceleraciones diferentes a g.

Se ve entonces una limitación importante. Imaginemos una situación donde se pone de manifiesto esto. Supongamos que tenemos un acelerómetro situado en un móvil. Si el individuo que posee el móvil se encuentra en un movimiento con $a \neq 0$ el acelerómetro registrará está aceleración que será diferente al rango de valores que esperaríamos medir para la δ del acelerómetro. Estas situaciones son extremadamente comunes. Subir las escaleras, conducir o incluso andar son actividades que se realizan a diario y en las que el individuo se ve sometido a fuerzas (y por ende aceleraciones) diferentes a la medida en reposo.

Tenemos entonces que por como se explica en la sección 3.2 tenemos que intentar reducir el número de acelerómetros al mínimo y a la vez evitar medir únicamente un sensor como se acaba de explicar.

Atendiendo entonces a las condiciones expuestas, el paso más natural a realizar será basar nuestra PUF en la diferencia medida por dos acelerómetros en un mismo dispositivo. En esta situación supondremos que el dispositivo tiene dos acelerómetros

diferentes. Si bien no es lo ideal ya que muchos dispositivos del IoT tienen tan solo un sensor de este tipo, es logísticamente mucho más fácil de conseguir que introducir cientos de acelerómetros en el sistema como sería necesario en caso de querer realizar un estudio basado similar a los realizados matrices de osciladores de anillo. Se elimina el problema de medir aceleraciones diferentes a g. Al someterlos a una aceleración diferente a la medida en reposo esta será registrada por ambos acelerómetros y al restar la señal de ambos estamos eliminando esa aceleración.

Entonces el sistema pasa de ser de 22 acelerómetros a 11 parejas de ellos.

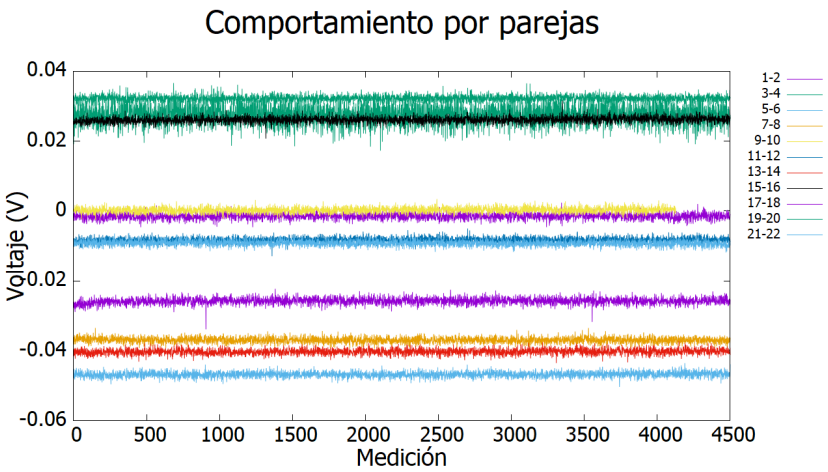


Figura 3.9: Voltaje respuesta medido en reposo para cada pareja de acelerómetros a lo largo del tiempo.

Veamos de nuevo que los promedios son únicos para cada pareja de acelerómetros (Figura 3.10). Como se observa, se distinguen 11 señales diferentes. Es decir, cada

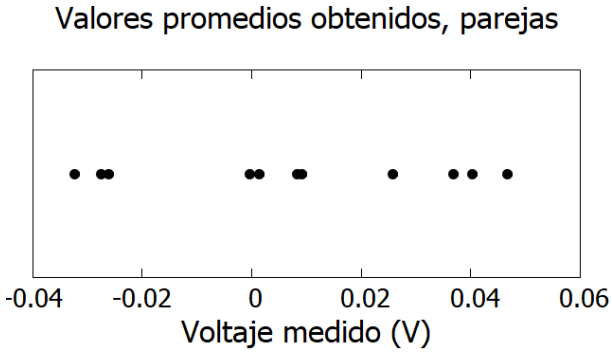


Figura 3.10: Voltaje respuesta promedio en reposo para cada par de acelerómetros.

pareja de acelerómetros tienen un promedio de valores único.

Si realizamos de nuevo el estudio de FRR y FAR obtenemos para un pareja de acelerómetros la gráfica de la Figura 3.11. También encontramos parejas en las que las condiciones ideales de corte FRR-FAR intersecan en $y \approx 0$

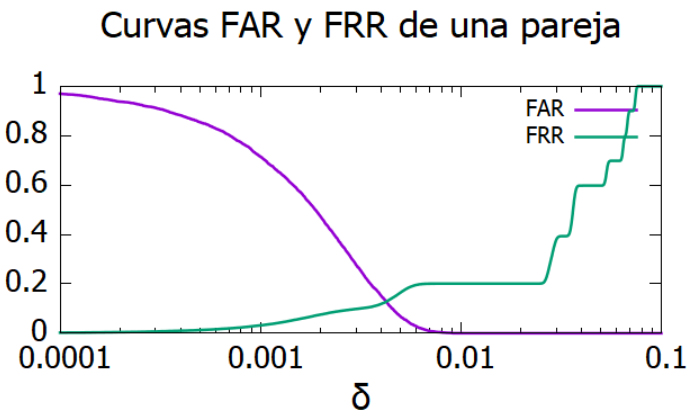


Figura 3.11: FAR y FRR para pareja de acelerómetros.

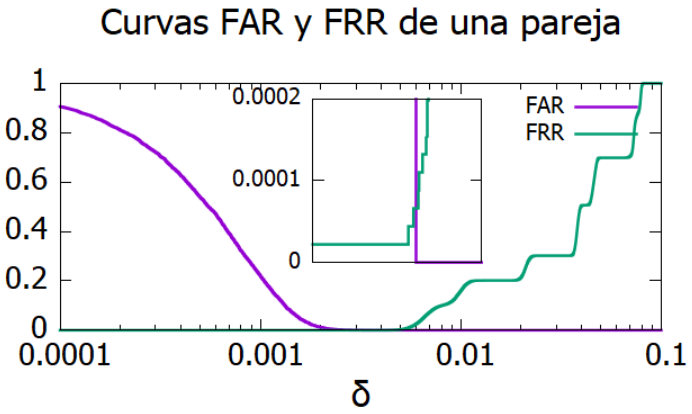


Figura 3.12: FAR y FRR para otra pareja de acelerómetros.

Un ejemplo de curva donde FRR y FAR intersecan en prácticamente 0 se aprecia en Figura 3.12. Se puede ver que el comportamiento es el mismo. Estudiemos el δ donde intersecan y su probabilidad.

$$\delta = 0{,}003 \pm 0{,}002\text{V} \qquad y = 0{,}04 \pm 0{,}04$$

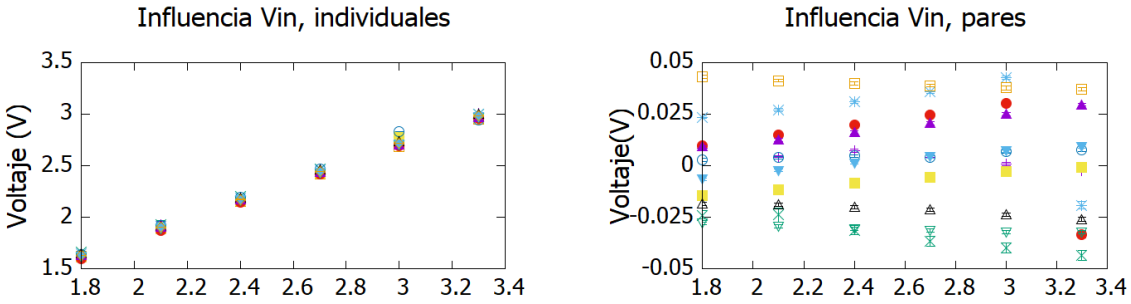
Como vemos, la probabilidad sigue tratándose de un valor relativamente bajo pero muy disperso. La δ promedio ha pasado a tener una mayor desviación. A pesar de las ventajas del estudio por parejas (eliminamos ruidos no inherentes al acelerómetro) esto implica una menor precisión a la hora de definir nuestro δ.

Observamos entonces como pueden desarrollarse la resta de la aceleración medida por dos acelerómetros como una PUF.

3.5. Influencia de otros factores

Hasta ahora todo el estudio se ha realizado con una fuente de alimentación que potencia los acelerómetros a $3{,}3V$ de forma constante.

Veamos como influye el voltaje aplicado al acelerómetro. Se someterán los acelerómetros (y las parejas) a diferentes voltajes de entrada y se estudiará el voltaje obtenido de salida (que recordemos se relaciona con g):



(a) Estudio de acelerómetros individuales

(b) Estudio de pares de acelerómetros

Figura 3.13: Representación voltaje medido de salida en función de V_{in} para ambos casos

Se observa que se trata de comportamiento lineal creciente $V_{out}(V_{in})$ para acelerómetros individuales. Modelando la dependencia como $y = mx + n$, tanto n como m son similares para cada acelerómetro. La expresión lineal promedio para los individuales es:

$$y_{input} = (0{,}905 \pm 0{,}005) * V_{input} + (0{,}012 \pm 0{,}011)$$

. En el caso de estudiar parejas sus respuestas no tienen un comportamiento igual en todos casos. Los datos no tiene una disposición predilecta, encontramos que las estimaciones lineales arrojarán rectas de diversas m, teniendo estas hasta valores negativos. Se ve que $m \in (-0{,}0157\ ,\ 0{,}0137)$.

Comparando m en ambos casos, la dependencia de V_{in} disminuye al estudiar el sistema por parejas. Al aumentar V_{in}, ambos acelerómetros de la pareja se ven afectados por ese aumento de voltaje (de forma similar pero no idéntica). Al restar las señales medidas por ambos acelerómetros se anula gran parte del aumento de voltaje causado por el aumento V_{in}. Esto supone una ventaja frente a medir las aceleraciones de forma individual.

Hay más factores que influyen a la señal medida. En el laboratorio se comprobó que la temperatura hace variar la respuesta medida. Esta afirmación es cualitativa, pues carecemos de datos como para realizar un análisis estadístico detallado. Si que

afirmaremos que se aprecia un correlación positiva $T - Vout$. Es decir, un aumento de temperatura provocaba un aumento en V_{out}.

Capítulo 4

Conclusiones y posibles líneas de investigación futuras

Como se puede observar, se ha propuesto una posible PUF compatible con IoT basada en acelerómetros.

Si bien se ha demostrado que cada acelerómetro/pareja de acelerómetros corresponde con una medida en reposo única para ese sistema, hay elementos externos cuya influencia podría ser de interés investigar. Además se puede observar que el sistema individual tiene una mayor precisión a la hora de definir δ pero se ve muy influenciado por aceleraciones externas o V_{in}. Pasa al contrario con las parejas. Encontramos entonces que cada método tiene su ventaja y desventaja.

Algunas cuestiones de interés para investigar serían la influencia del voltaje de la batería que alimenta el acelerómetro. Sería por ejemplo ilustrativo investigar cómo variarían las curvas FRR y FAR cuando alimentamos con fuentes de alimentación más rudimentarias. Una fuente de alimentación menos estable que la nuestra podría sufrir picos o caídas de voltaje puntuales, que dependiendo del tiempo (y por ende la cantidad de medidas realizadas) de estudio del acelerómetro podría provocar cambios notables en la forma e intersección de las curvas.

Otra cuestión que no se ha atendido en este trabajo pero despierta gran interés es la influencia de la temperatura en el estudio. Se vio en el laboratorio que un aumento de temperatura implica un aumento del voltaje medido. Esto abre la posibilidad de que la señal de un acelerómetro a una temperatura sea confundido por la señal que daría un acelerómetro diferente a otra temperatura. Esto podría mitigares realizando el estudio por parejas (igual que se ha hecho V_{in}), pero solo sería válido si la influencia de la temperatura sobre el voltaje es la misma para todos los acelerómetros, que no puede asegurarse a priori.

Bibliografía

[1] Alfred J Menezes, Paul C Van Oorschot, and Scott A Vanstone. *Handbook of applied cryptography*. CRC press, 2018.

[2] Olivier Benot. *Encyclopedia of Cryptography and Security*, chapter Fault Attack, pages 452–453. Springer US, Boston, MA, 2011.

[3] Kris Tiri. Side-channel attack pitfalls. In *2007 44th ACM/IEEE Design Automation Conference*, pages 15–20, 2007.

[4] Ravikanth Pappu, Ben Recht, Jason Taylor, and Neil Gershenfeld. Physical one-way functions. *Science*, 297(5589):2026–2030, 2002.

[5] Felix Wortmann and Kristina Flüchter. Internet of things. *Business & Information Systems Engineering*, 57(3):221–224, 2015.

[6] Skoric B. Tuyls P. Gujardo, J. Anti-counterfeiting, key distribution, and key storage in an ambient world via physical unclonable functions. *Inf Syst Front*, pages 19–41, 2009.

[7] Christoph Böhm and Maximilian Hofer. *Physical Unclonable Functions in Theory and Practice*. Springer, Graz, Austria, 2012.

[8] MAES Roel. Physically unclonable functions: Constructions, properties and applications. *Katholieke Universiteit Leuven, Belgium*, 2012.

[9] Daniel Gil Marco, Carlos Sánchez Azqueta, and Guillermo Díez Señorans. Generador hardware de claves basado en funciones físicamente no clonables. *Unizar*, 2020.

[10] Linear Technology. *Small, Low Power, 3-Axis ±3g Accelerometer*, 2010. Rev. B.

[11] Esquema condensador. $hhttps : //www.researchgate.net/figure/Capacitive - accelerometer - with - lateral - sensing_fig2_264833557$. Accedido: 2021-06-25.

[12] Abhranil Maiti and Patrick Schaumont. Improved ring oscillator puf: An fpga-friendly secure primitive. *Journal of cryptology*, 24(2):375–397, 2011.

[13] G Edward Suh and Srinivas Devadas. Physical unclonable functions for device authentication and secret key generation. In *2007 44th ACM/IEEE Design Automation Conference*, pages 9–14. IEEE, 2007.

Índice de figuras

Acrónimos

ID Identificador (Identifier).

IoT Internet de las cosas (Internet of Things).

MEMS Sistema microelectromecánico (Microelectromechanical systems).

NVM Memoria no volátil (Non volatile memory).

PUF Funciñon física no clonable (Physical Unclonable Function).

RFID Identificación por radiofrecuencia (Radiofrequency identification).

Anexos

Anexos A

Anexo

A.1. Código para la toma de datos a lo largo del tiempo

```
import pyvisa
import time
import matplotlib.pyplot as plt
import numpy as np
import time
import datetime


time_run = time.time()
print(time_run)


rm = pyvisa.ResourceManager()
print(rm.list_resources())

print("0")

#lista instrumentos
oscilloscope_1 = rm.open_resource('USB0::0x0957::0x179B:
:MY51250652::INSTR')
oscilloscope_2 = rm.open_resource('USB0::0x0957::0x179B:
:MY51135621::INSTR')
oscilloscope_3 = rm.open_resource('USB0::0x0957::0x179B:
:MY51250766::INSTR')
#print(oscilloscope_1.query('*IDN?'))
#print(oscilloscope_2.query('*IDN?'))
#print(oscilloscope_3.query('*IDN?'))
```

```
#Measures 2 acelerometres same time, same coordinate


file1 = open("3-4FuenteB.txt", "w")
#file2 = open("3h2", "w")


print(time.asctime(time.localtime(time.time())))
file1.write("#11" + str(time.asctime(time.localtime(time.time()))))
print("Intruduce detalles (tiempo de medicion, selftest,
modod adquisicion, escalas):")
details = input()
file1.write("#" + details)
file1.write("#x1    y1    z1    x2    y2    z2 \n")
total_meas = 0;
print("start")
t_start = time.time()
while ((time.time()-time_run)<=3600.0):#toma medidas 1.0 horas
    meas_x_1 = float(oscilloscope_1.query('meas:vav? chan1'))
    meas_y_1 = float(oscilloscope_1.query('meas:vav? chan2'))
    meas_z_1 = float(oscilloscope_2.query('meas:vav? chan1'))

    meas_x_2 = float(oscilloscope_2.query('meas:vav? chan2'))
    meas_y_2 = float(oscilloscope_3.query('meas:vav? chan1'))
    meas_z_2 = float(oscilloscope_3.query('meas:vav? chan2'))

    total_meas += 1;
    #print(total_meas)
    file1.write(str(total_meas) + '\t' +str(meas_x_1) + '\t'
    + str(meas_y_1) + '\t' + str(meas_z_1) + '\t' + str(meas_x_2)
    + '\t'+ str(meas_y_2) + '\t' + str(meas_z_2) + '\n')
    if(total_meas % 100 == 0):
        print(int(time.time()-time_run)/60)
    #print(total_meas)


file1.close()

print(total_meas)
print(time.time()- time_run)
print("Medidas segundo:" + str(float(total_meas/(time.time()- time_run))))
```

A.2. Código toma datos en función de V_{in}

```
import pyvisa
import time
import matplotlib.pyplot as plt
```

```
import numpy as np
import time
import datetime

#tiempo_toma_medidas = 20#minutos
#1 medida por segundo

time_run = time.time()
print(time_run)


rm = pyvisa.ResourceManager()
print(rm.list_resources())

print("0")

#lista instrumentos
oscilloscope_1 = rm.open_resource('USB0::0x0957::0x179B:
:MY51250652::INSTR')
oscilloscope_2 = rm.open_resource('USB0::0x0957::0x179B:
:MY51135621::INSTR')
oscilloscope_3 = rm.open_resource('USB0::0x0957::0x179B:
:MY51250766::INSTR')
print(oscilloscope_1.query('*IDN?'))
print(oscilloscope_2.query('*IDN?'))
print(oscilloscope_3.query('*IDN?'))




#Measures 2 acelerometres same time, same coordinate

#cambiar a cada ejecucion, describir tambien X
file1 = open("Input_21-22.txt", "w")
#file2 = open("3h2", "w")


print(time.asctime(time.localtime(time.time())))
file1.write("#11" + str(time.asctime(time.localtime(time.time()))))
print("Intruduce detalles (tiempo de medicion, selftest
, modod adquisicion, escalas):")
details = input()
file1.write("#" + details)
file1.write("#x1   y1    z1    x2    y2    z2 \n")
total_meas = 0;
print("start")
t_start = time.time()
meas_tot = 0
```

```
Vin = 1.8
print(type(Vin))
Vin = float(Vin)
file1.write(str(Vin))
max_meas = 100
while(Vin <= 3.4):
    while (meas_tot < max_meas):#toma medidas 1.0 horas
        meas_x_1 = float(oscilloscope_1.query('meas:vav? chan1'))
        meas_y_1 = float(oscilloscope_1.query('meas:vav? chan2'))
        meas_z_1 = float(oscilloscope_2.query('meas:vav? chan1'))

        meas_x_2 = float(oscilloscope_2.query('meas:vav? chan2'))
        meas_y_2 = float(oscilloscope_3.query('meas:vav? chan1'))
        meas_z_2 = float(oscilloscope_3.query('meas:vav? chan2'))

        meas_tot += 1;
        #print(total_meas)
        file1.write(str(total_meas) + '\t' +str(meas_x_1) + '\t' +
        str(meas_y_1) + '\t' + str(meas_z_1) + '\t' + str(meas_x_2)
        + '\t'+ str(meas_y_2) + '\t' + str(meas_z_2) + '\n')
        #if(total_meas % 100 == 0):
            #print(int(time.time()-time_run)/60)
        #print(total_meas)
        if (meas_tot%10 == 0):
            print(meas_tot)
        if(meas_tot == max_meas):
            print("Realizadas medidas cambia voltaje de",
            Vin, "a", Vin+0.3)
            input()

            file1.write(str(Vin+0.3))
    Vin += 0.3
    meas_tot=0
    print(Vin)
    print(meas_tot)




file1.close()

print(total_meas)
print(time.time()- time_run)
print("Medidas segundo:" +
str(float(total_meas/(time.time()- time_run))))
```

A.3. Código lectura de medidas y cálculo curvas FRR y FAR

```
import numpy as np
import matplotlib.pyplot as plt

def promedio(array):#devuelve el promedio de un array
    den = 0
    nom = 0
    for i in array:
        den += 1
        nom += i
    avg = float(nom/den)
    return avg
def inside_range(center, bounds, value):
    upper_bound = center + bounds
    lower_bound = center - bounds
    if ((lower_bound < value)and(value < upper_bound)):
        return True
    else:
        return False
def stdev(array):#devuelve la stddev de un array
    den = -1 #N-1
    nom = 0
    avg = promedio(array)
    for i in array:
        nom += (i-avg)**2
        den += 1

    std = nom/den
    std = (std)**0.5
    return std

def logspace_numbers():
    #to 3 --> 0.477121255
    #to 5 --> 0.698970004
    #to 30--> 1.477121255
    #to 50--> 1.698970004
    return np.logspace(-4,0, num = 25_000)

def FRR(list_data, avg_val, stdev):#returns array
#of FRR(range of acceptance)
    ###iterate over stdev*x, where x belongs (0.01,3)
    x = logspace_numbers()
    frr_list = []
    for element in x:
        bound = element
```

```
        frr = 0
        for data in list_data:
            if (not(inside_range(avg_val, bound, data))):
                frr += 1
        frr /= len(list_data)
        frr_list.append(frr)
    return frr_list

def FAR(matrix_data, acel_number,avg_val, stdev):
#returns array of FAR(range of acceptance)
    ###iterate over stdev*x, where x belongs (0.01,3)
    x = logspace_numbers()
    far_list = []
    for element in x:
        bound = element
        far = 0#number of false acceptances
        total_eval = 0#control how many comparison have been done
        for acel in range(len(matrix_data)):#every acelerometer
            if (acel != acel_number):#not itself
                for data in matrix_data[acel]:
                #every measurement of acelerometer
                    total_eval += 1
                    if ((inside_range(avg_val, bound, data))):
                        far += 1
        far /= total_eval
        far_list.append(far)
    return far_list

def study_pairs(matrix):
    new_matrix = [[],[],[],[],[],[],[],[],[],[]]
    pair = 0
    for i in range(0,len(matrix),2):
        new_matrix[0].append(matrix[i+1][0]-matrix[i][0])
        for j in range(len(matrix[i])):
            new_matrix[pair].append(matrix[i+1][j]-matrix[i][j])
        pair += 1
    print("Pair matrix has columns:")
    print(len(new_matrix))
    print("rows")
    print(len(new_matrix[0]))
    return new_matrix

##raw data to matrix####
ficheros_num = 10
aceleroms_num = 22
gs_acelerometers = [[] for i in range(aceleroms_num+1)]
for i in range(1,aceleroms_num+1,2):
    filename = str(i) + "-" + str(i+1) + "FuenteB.txt"
```

```
    print(filename)

    with open(filename) as f:
        content = f.readlines()
    lin = 0
    for line in content:
        #print(line.split())
        if lin > 0:
            line = line.split()
            line = [float(j) for j in line]
            g = (line[1]**2 + line[2]**2 + line[3]**2)**0.5
            #print(gs_acelerometers)
            gs_acelerometers[i].append(g)
            #print(gs_acelerometers)
            g = (line[4]**2 + line[5]**2 + line[6]**2)**0.5
            gs_acelerometers[i+1].append(g)
            #print(g)
        lin += 1
        #input()
    #input()

gs_acelerometers.pop(0)#remove empty starting row
for i in range(len(gs_acelerometers)):
    print(i,len(gs_acelerometers[i]))
#####end raw data to matrix

print("Trabajar con parejas (y/n):")
parejas = input()
if (parejas == 'y'):
    gs_acelerometers = study_pairs(gs_acelerometers)
######get array with avgs and stdevs
avgs = []
stdevs = []
for i in range(len(gs_acelerometers)):
    avg = promedio(gs_acelerometers[i])
    avgs.append(avg)

    std = stdev(gs_acelerometers[i])
    stdevs.append(std)

for i in range(len(avgs)):
    print(0,i,avgs[i], stdevs[i])
input()
#######end array with avg and stdevs
for i in range(len(gs_acelerometers)):
    list_data = gs_acelerometers[i]
    avg_val = avgs[i]
    stdev = stdevs[i]
```

```
    frr = FRR(list_data,avg_val,stdev)
    far = FAR(gs_acelerometers, i,avg_val, stdev)
    print(i)
    if (i < 11):
        x = logspace_numbers()
        y = frr
        y_2 = far
        plt.figure(i)
        plt.plot(x,y, 'bo')
        plt.plot(x, y_2, 'go')
        plt.xscale('log')
        plt.xlabel('sigma*x')
        plt.title(i)

    file_name = "Signed" + str(i) + ".txt"
    if(parejas):
        file_name = "Pair" + file_name
    fo = open(file_name, "w")
    for i in range(len(x)):
        dump_to_file = str(x[i]) + "\t" + str(frr[i])
                    + "\t" + str(far[i]) + "\n"
        fo.write(dump_to_file)
    fo.close()
plt.show()
```

A.4. Código análisis respuesta frente al voltaje aplicado de entrada

```
    import numpy as np
import matplotlib.pyplot as plt

def promedio(array):#devuelve el promedio de un array
    den = 0
    nom = 0
    for i in array:
        den += 1
        nom += i
    avg = float(nom/den)
    return avg

def promedio_limits(array, lim_inf, lim_sup):
    den = 0
    nom = 0
    for i in range(lim_inf, lim_sup):
        den += 1
        nom += array[i]
```

```
    avg = float(nom/den)
    return avg


def inside_range(center, bounds, value):
    upper_bound = center + bounds
    lower_bound = center - bounds
    if ((lower_bound < value)and(value < upper_bound)):
        return True
    else:
        return False


def stdev(array):#devuelve la stddev de un array
    den = -1 #N-1
    nom = 0
    avg = promedio(array)
    for i in array:
        nom += (i-avg)**2
        den += 1

    std = nom/den
    std = (std)**0.5
    return std

def stdev_limits(array, lim_inf, lim_sup):
#devuelve la stddev de un array
    den = -1 #N-1
    nom = 0
    avg = promedio_limits(array, lim_inf, lim_sup)
    for i in range(lim_inf, lim_sup):
        nom += (array[i]-avg)**2
        den += 1

    std = nom/den
    std = (std)**0.5
    return std



def logspace_numbers():
    #to 3 --> 0.477121255
    #to 5 --> 0.698970004
    #to 30--> 1.477121255
    #to 50--> 1.698970004
    return np.logspace(-4,0, num = 25_000)
```

```
def study_pairs(matrix):
    new_matrix = [[],[],[],[],[],[],[],[],[],[],[]]
    pair = 0
    for i in range(0,len(matrix),2):
        new_matrix[0].append(matrix[i+1][0]-matrix[i][0])
        for j in range(len(matrix[i])):
            new_matrix[pair].append((matrix[i+1][j]-matrix[i][j]))
        pair += 1
    print("Pair matrix has columns:")
    print(len(new_matrix))
    print("rows")
    print(len(new_matrix[0]))
    return new_matrix



##raw data to matrix####
ficheros_num = 10
aceleroms_num = 22
gs_accelerometers = [[] for i in range(aceleroms_num+1)]
#matridx with raw data
for i in range(1,aceleroms_num+1,2):
    filename = "Input_" + str(i) + "-" + str(i+1) + ".txt"
    print(filename)

    with open(filename) as f:
        content = f.readlines()
    lin = 0
    for line in content:
        #print(line.split())
        if lin > 0:
            line = line.split()

            line = [float(j) for j in line]
            #print(line)
            #input()
            g = (line[1]**2 + line[2]**2 + line[3]**2)**0.5
            #print(gs_accelerometers)
            gs_accelerometers[i].append(g)
            #print(gs_accelerometers)
            g = (line[4]**2 + line[5]**2 + line[6]**2)**0.5
            gs_accelerometers[i+1].append(g)
            #print(g)
        lin += 1
        #input()
    #input()

    #check no dataerror//Removes e+38
```

```
for i in range(len(gs_acelerometers)):
    loop = len(gs_acelerometers[i])
    for j in range(loop):
        if (gs_acelerometers[i][j] > 10):
            gs_acelerometers[i][j] = gs_acelerometers[i][j-1]

gs_acelerometers.pop(0)#remove empty starting row




#print(gs_acelerometers[0])

print("Trabajar con parejas (y/n):")
parejas = input()
if (parejas == 'y'):
    gs_acelerometers = study_pairs(gs_acelerometers)



#array for Vin
Vin_means = [[] for i in range(len(gs_acelerometers))]
Vin_stdevs = [[] for i in range(len(gs_acelerometers))]
for i in range(len(gs_acelerometers)):
    #print(len(gs_acelerometers[i]))
    #print(gs_acelerometers[i])
    for j in range(6):#600 measurements
        mean = promedio_limits(gs_acelerometers[i], 100*j, 100*(j+1))
        standarddev= stdev_limits(gs_acelerometers[i], 100*j, 100*(j+1))

        Vin_means[i].append(mean)
        Vin_stdevs[i].append(standarddev)
print("Len Vin_means[0]:", len(Vin_means[0]))
####this returns two arrays with values g(Vin) for each acelerometers

#####end raw data to matrix




#print(len(Vin_means))
#for i in range(len(Vin_means)):
#    print(i,Vin_means[i][0])
```

```
######get array with avgs and stdevs this is avg across acelerometers

avgs = []#avg across
stdevs = []

for i in range(len(Vin_means[i])):
    lst = []
    print(i)
    #print(len(Vin_means))
    for j in range(len(Vin_means)):
        lst.append(Vin_means[j][i])
        #print(lst)

    avg = promedio(lst)
    avgs.append(avg)
    stdevs.append(stdev(lst))



for i in range(len(avgs)):
    print(i,avgs[i], stdevs[i])

#######end array with avg and stdevs
file_name = "Input.txt"
file_name_avg = "AVG_Inputs.txt"
if (parejas == 'y'):
    file_name = "Pair_" + file_name
    file_name_avg = "Pair_" + file_name_avg

fo = open(file_name_avg, "w")
ff = open(file_name, "w")
for i in range(len(avgs)):
    fo.write(str(1.8+0.3*i) + "\t" + str(avgs[i]) + "\t"
    + str(stdevs[i]) + "\n")

for i in range(len(Vin_means[0])):
    ff.write(str(i*0.3+1.8) + "\t")
    for j in range(len(Vin_means)):
        ff.write(str(Vin_means[j][i]) + "\t"
        + str(Vin_stdevs[j][i]) + "\t")
    ff.write("\n")
ff.close()
```