

PHYSICAL UNCLONABLE FUNCTIONS (PUFS) Y TECNOLOGÍAS EMERGENTES PARA EL FUTURO DE LA SEGURIDAD

Sergio Vinagrero Gutiérrez | vinagres@univ-grenoble-alpes.fr | TIMA Laboratory

UN FALLO, MILLONES EN PÉRDIDAS

- ▶ PlayStation 3 (Sony, 2010)
- ▶ Sistema de firma digital para proteger el software
- ▶ Error criptográfico:
 - Uso de un número aleatorio constante (nonce)
 - Recuperación de la clave privada
- ▶ Consecuencia:
 - Homebrew (Bueno para los usuarios :D)
 - Copias no autorizadas (Malo para la empresa)
 - Pérdidas económicas masivas (Peor todavía)

```
int getRandomNumber()
{
    return 4; // chosen by fair dice roll.
              // guaranteed to be random.
}
```

EL MAYOR ENEMIGO DE NINTENDO

- ▶ **Nintendo Switch (2018)**
- ▶ Vulnerabilidad en el **BootROM** del SoC (NVIDIA Tegra X1)
- ▶ El BootROM:
 - Es código inmutable
 - Se almacena en **memoria de solo lectura**
- ▶ Ataque:
 - Cortocircuito de pines con un *clip* o *paperclip*
 - Entrada en modo de recuperación (RCM)
 - Ejecución de código no autorizado
- ▶ Consecuencia:
 - Homebrew y Piratería
 - **Vulnerabilidad imposible de parchear por software**



Figura: Imagen de <https://noirscape.github.io/RCM-Guide/>

**¿CÓMO SABE UN PROGRAMA QUE NO
HA SIDO MANIPULADO?**

RAÍZ DE CONFIANZA EN HARDWARE

La **raíz de confianza** (Root of Trust) es el elemento fundamental sobre el que se construye la seguridad de un sistema.

La raíz de confianza debe ser:

- Intrínseca al hardware
- Difícil de copiar o clonar
- Accesible de forma controlada

El problema es que un secreto almacenado puede ser leído, copiado o extraído.

¿Puede una memoria ser una raíz de confianza?

POR QUÉ LAS NVM NO SON UNA BUENA RAÍZ DE CONFIANZA

- ▶ Las NVM embebidas (Flash, eFuse, EEPROM) retienen información sin alimentación
- ▶ **Almacenan secretos de forma estática**
- ▶ Debido a su importancia, son el objetivo prioritario de atacantes:
 - Lectura directa
 - Ataques invasivos (decapado, FIB)
 - Canales laterales (corriente, EM, fotones)

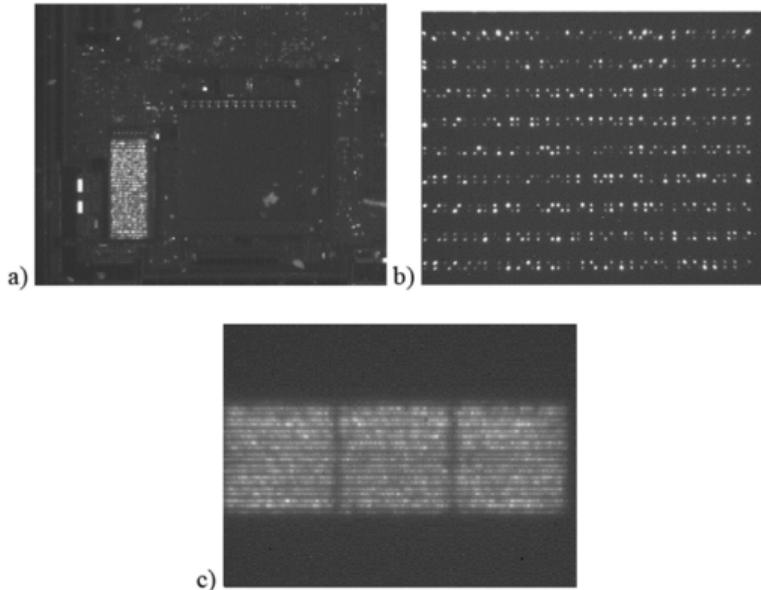


Figura: Ataque de emisión de fotones en Cortex-M0 SRAM en "Quiescent photonics side channel analysis: Low cost SRAM readout attack"

**¿Y SI LA CLAVE NO SE ALMACENA, NO
EXISTE COMO DATO Y SE GENERA
SOLO CUANDO SE NECESITA?**

MATERIAL

Las diapositivas, material, notebook y datos experimentales están disponibles en Github:

https://github.com/servinagrero/workshop_puf_meristors.git

¿QUIÉN SOY YO?

- ❖ Grado en Ingeniería Industrial Electrónica Y Automática en UC3M
- ❖ Master en Wireless Integrated Communication Systems (WICS) en Université Grenoble Alpes
- ❖ Doctorado en *Métodos para el diseño, modelado y la evaluación de la calidad de Physical Unclonable Functions (PUFs)*
- ❖ Postdoctorado en el proyecto NEUROPULS para el desarrollo de protocolos de autenticación resistentes al aprendizaje automático basados en PUFs



Figura: Campus de la UC3M en Leganés



Figura: Ciudad de Grenoble, Francia

¿CÓMO SE FABRICAN LOS CIRCUITOS?

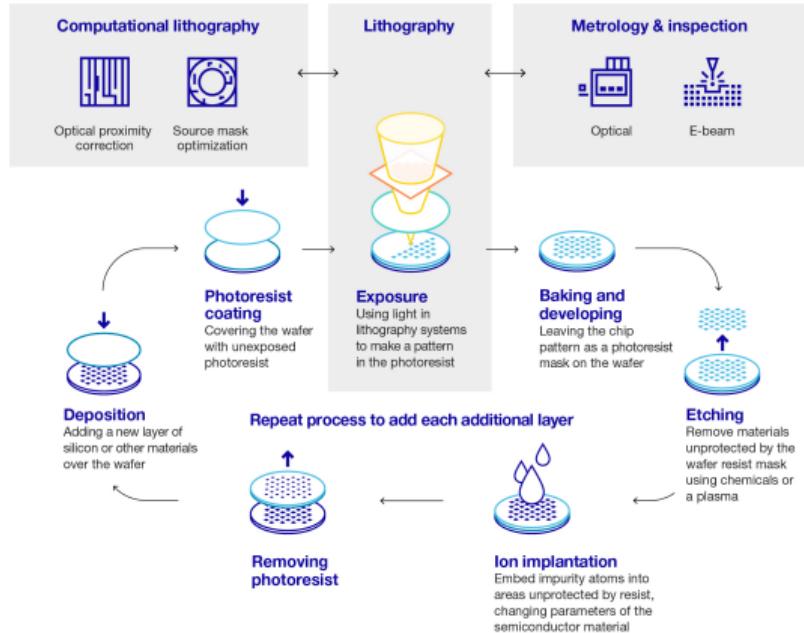


Figura: Proceso de Fotolitografía

LA VANGUARDIA DE LA FOTOLITOGRAFÍA

- ▶ Resolución de $\sim 2 \text{ nm}$
- ▶ **185–220** obleas por hora (wph)
- ▶ Coste aproximado de 400 M€

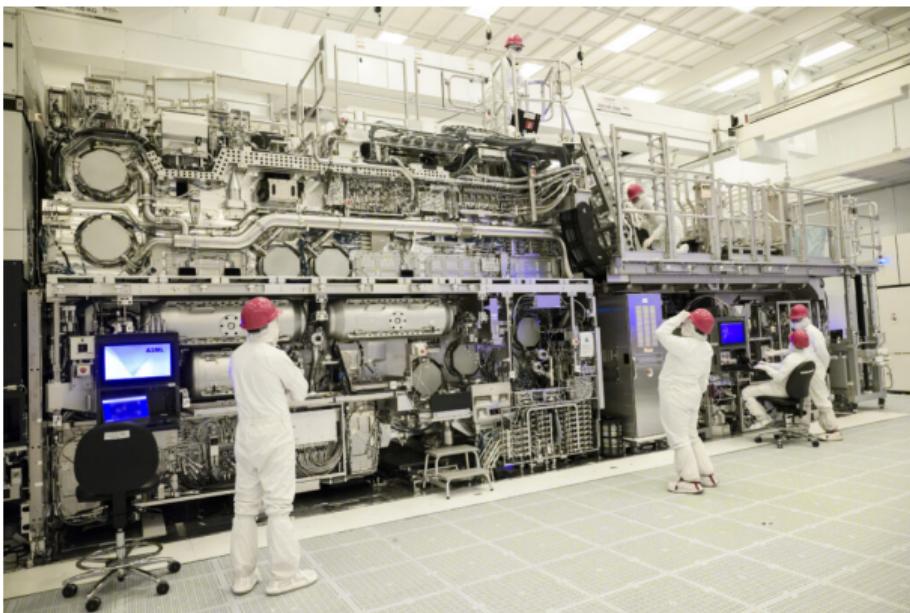


Figura: Máquina de ASML para "High NA EUV Photolithography"

LA FOTOLITOGRÁFIA NO ES PERFECTA

- Como todo en esta vida, la fotolitografía no es perfecta
- *Line edge roughness*, errores de alineación, variaciones en la dosis de dopado...
- La propia luz se deforma al pasar por los espejos y las lentes
 - Optical Proximity Correction (OPC) y Computational Lithography
- Estos efectos físicos producen **variaciones geométricas y eléctricas microscópicas** entre dispositivos incluso en el mismo chip.

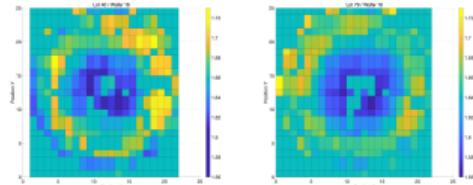


Figura: Variabilidad de frecuencia con respecto a la posición en la oblea. Datos de Infineon

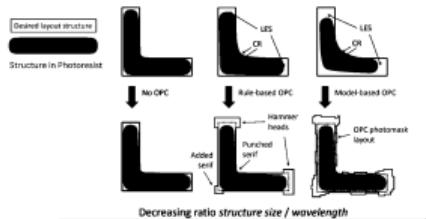


Figura: Proceso de Optical Proximity Correction. Imagen extraída de <https://siliconvlsi.com>

BIOMÉTRICA DEL SILICIO

- Todas estas pequeñas imperfecciones hacen que el funcionamiento de nuestro circuito no sea tal y como se desea
- Podemos explotar estas pequeñas **variaciones aleatorias** para generar **secretos que sean únicos, dependan del dispositivo, y no se puedan manipular**

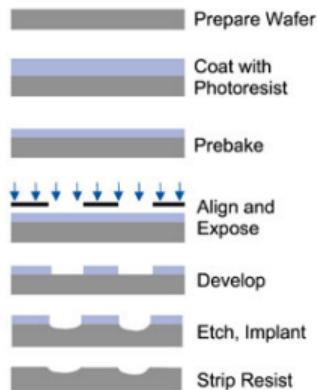


Figura: Distintas etapas en el proceso de fotolitografía

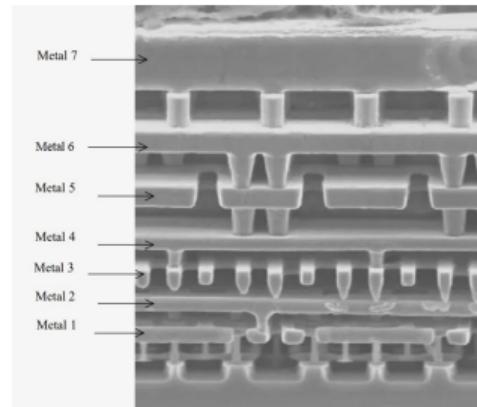


Figura: Diversos niveles de interconexión de los transistores

¿EL TAMAÑO IMPORTA?

¿Teniendo dos circuitos idénticos, fabricados con 2 nodos de tecnología distinta, cuál presentaría más variabilidad?

1. 65 nm
2. 28 nm
3. Tendrían la misma

LEYES DE MOORE Y PELGROM

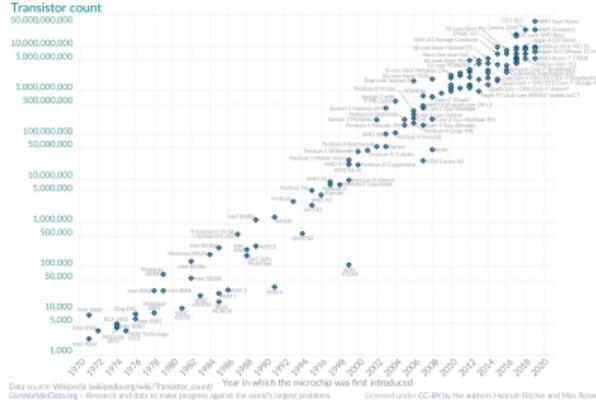
Ley de Moore

- El número de transistores por circuito integrado crece exponencialmente
- La reducción del nodo tecnológico permite, más transistores, mayor densidad y menor área por dispositivo

Ley de Pelgrom

- La variabilidad entre dispositivos aumenta al reducir el área
- La variabilidad está ligada a efectos estadísticos del proceso

Moore's Law: The number of transistors on microchips doubles every two years
Moore's law describes the empirical regularity that the number of transistors on integrated circuits doubles approximately every two years.
This advancement is important for other aspects of technological progress in computing – such as processing speed or the price of computers.



Data source: Wikipedia (https://en.wikipedia.org/wiki/Transistor_count)
Year in which the microchip was first introduced
OurWorldInData.org - Research and data to make progress against the world's largest problems
Licensed under CC-BY by the authors Harald Richter and Max Roser.

$$\sigma \propto \frac{1}{\sqrt{W \cdot L}}$$

TABLE OF CONTENTS

Physical Unclonable Functions (PUFs)

Métricas y Análisis

Workshop Práctico: Extracción y Análisis de Datos Reales

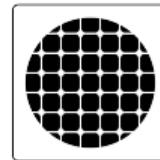
Workshop Práctico: Modelado y Aprendizaje Automático

Tecnologías Emergentes: PUFs de Memristores

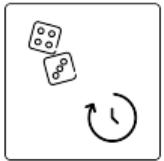
PHYSICAL UNCLONABLE FUNCTIONS (PUFS)



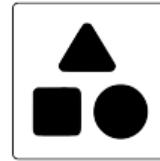
Mecanismo para generar IDs únicos para cada dispositivo
Fácil de utilizar, intrínseco al dispositivo y no se puede clonar al fabricarlo



PUFs explotan las variaciones físicas intrínsecas
Variaciones sistemáticas, en la oblea, entre obleas, entre dispositivos...



IDs deberían ser estables, aleatorios y únicos para cada dispositivo



Pares de Challenge-Response (CRP)
Se pueden clasificar por en el número de CRPs, fuente de entropía, método de uso...

TAXONOMÍA DE PUFS

Weak PUFs

- Sus CRPs escalan linealmente con su tamaño
- Número limitado de CRPs (128, 256, 512 bits)
- Usadas para clavas o identificadores de dispositivos

Strong PUFs

- Sus CRPs escalan exponencialmente con su tamaño
- Número inmenso de CRPs ($> 2^{80}$)
- Se utilizan para autenticación recíproca o sistemas de verificación complejos

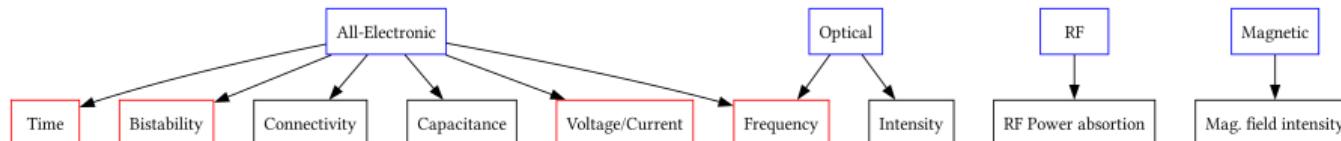


Figura: Distintas fuentes de entropía que se pueden utilizar

- Como diseñadores tenemos que tener una visión holística de todos las dificultades y ataques
- **La seguridad no es absoluta, es relativa**
- En esta presentación pondremos el foco en las fuentes de entropía, el diseño y su calidad

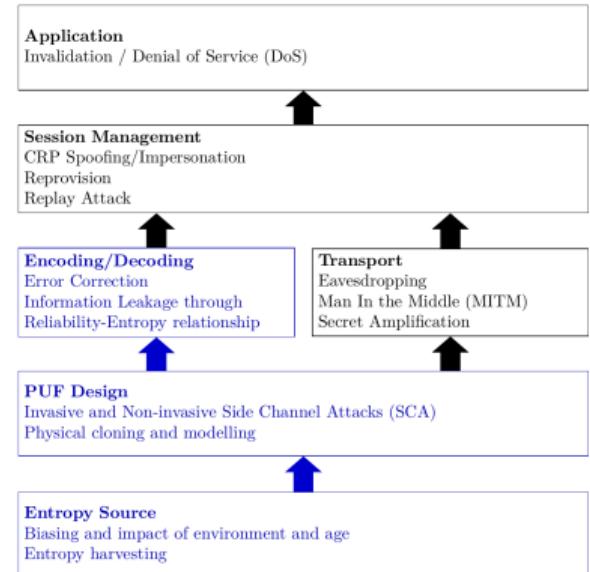


Figura: Vista general de las dificultades que a la hora de diseñar y usar PUFs

SRAM PUF

- Una celda SRAM se compone de 2 inversores cruzados
- Debido a la variabilidad, cada inversor acaba siendo ligeramente mayor o menor
- Al encenderse, cada celda tiende a un estado aleatorio (0 o 1)
- Este patrón inicial actúa como una firma fija del chip.
- **La arquitectura de las celdas y de la memoria SRAM tiene un gran impacto en la calidad del patrón**

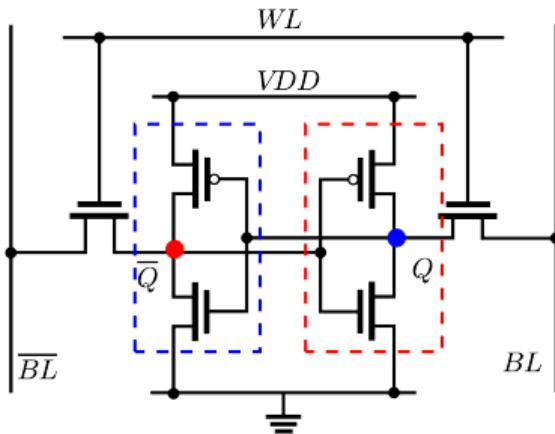


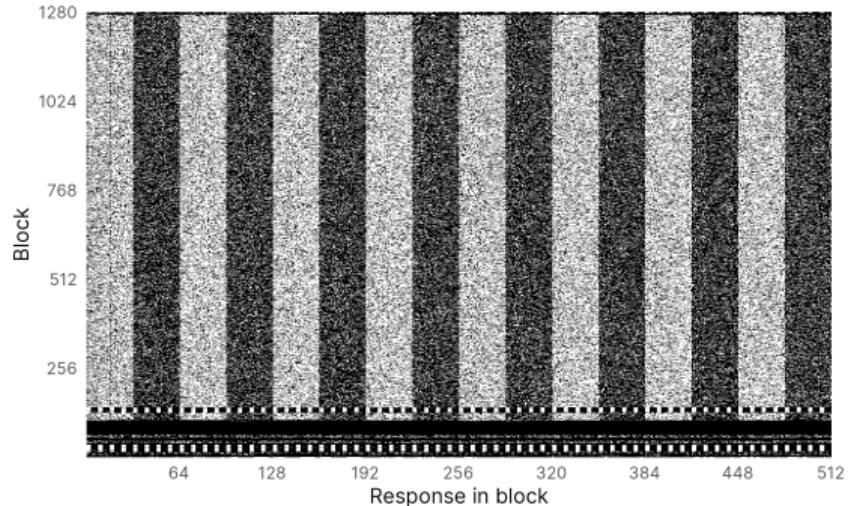
Figura: Diagrama de una típica celda SRAM 6T

PATRONES EN SRAM PUF

Que queremos obtener



Que se suele obtener...



ARQUITECTURA SRAM PUF

- La calidad de una sola celda no determina totalmente la calidad del SRAM PUF
- Depende en gran medida de la arquitectura, diseño global, enrutado...
- Crear y probar variaciones de grandes diseños puede ser muy complejo
- Mínima ayuda de DRC y LVS dado que se suelen crear en menor tamaño que el resto del circuito (más celdas por unidad de área)
- OpenRAM: An open-source static random access memory (SRAM) compiler.
<https://openram.org/>

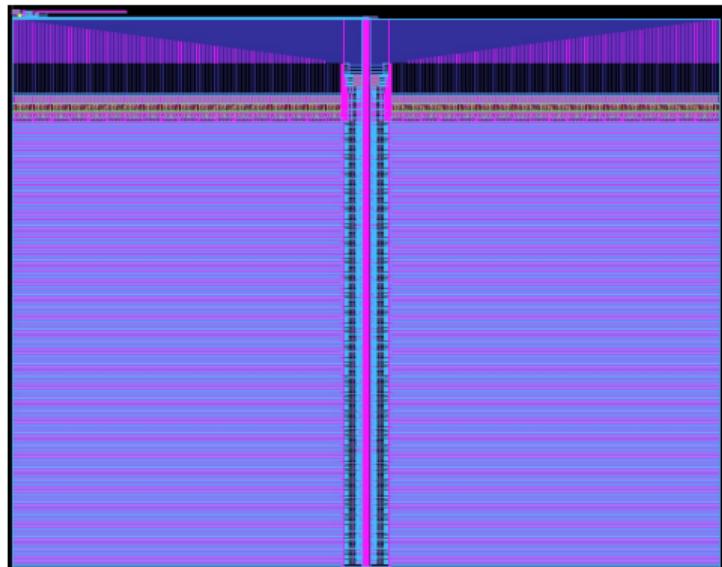
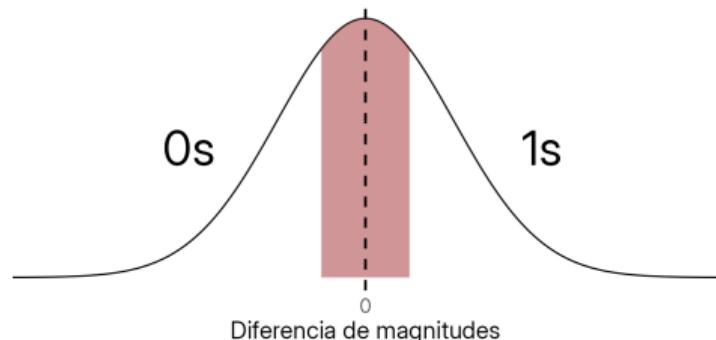


Figura: Imagen de un diseño de SRAM de 16kb de OpenRAM

PUFS "DIFERENCIALES"

- Comparación entre varios componentes **diseñados idénticamente**
- Debido a la variabilidad, cada componente, se comporta de manera ligeramente diferente
- Las magnitudes suelen ser analógicas y hay que digitalizarlas
- Las respuestas deben de ser únicas por dispositivo y repetibles**, o la clave puede cambiar cada vez que se interroga al PUF



$$\text{Respuesta} = \underbrace{\text{sign}}_{\text{Digitalización}} \underbrace{(\text{Medición 1} - \text{Medición 2})}_{\text{Diferencia}}$$

RING OSCILLATOR (RO) PUF

- Consiste en comparar la diferencia de frecuencia de "anillos oscilantes"
- En ASIC conviene separar físicamente los ROs para aumentar variabilidad
- En FPGA queremos pares de RO que estén cerca (si están muy lejos hay muchísima variabilidad)
- En FPGA es crucial fijar LUTs y enrutado para evitar optimizaciones del sintetizador.
- El enrutado puede introducir sesgos sistemáticos: hay que balancear diseño para que la diferencia entre rutas sea mínima salvo por variación aleatoria.

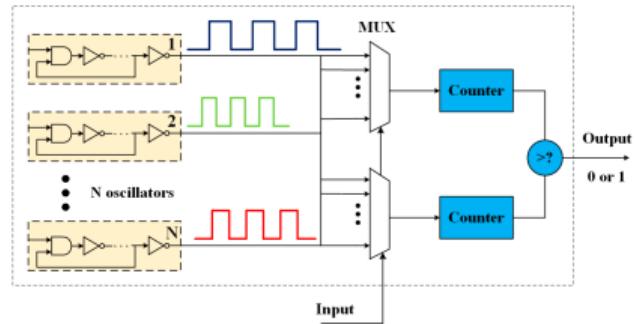


Figura: Diseño clásico de un Ring Oscillator PUF

EJEMPLO DE IMPLEMENTACIÓN EN VHDL

- Hay multiples maneras de crear un RO PUF
- El proceso es ligeramente distinto para ASIC y para FPGA
- Hay que asegurarse que la fuente de entropía proviene solamente de las diferencias de frecuencias y que no introducimos ningún sesgo

```
library IEEE;
use IEEE.STD_LOGIC_1164.ALL;

entity ro_puf is
    Port ( clk_i : in STD_LOGIC;
           rst_i : in STD_LOGIC;
           clk_o : buffer STD_LOGIC);
end ro_puf;
```

```
architecture Behavioral of ro_puf is
    signal chain :std_logic_vector(30 downto 0);
    attribute syn_keep: boolean;
    attribute syn_keep of chain: signal is true;
begin
    gen_chain:
    for i in 1 to 30 generate
        chain(i) <= not chain(i-1);
    end generate;
    chain(0) <= not chain(30) or not rst_i;

    t_FF: process(rst_i, chain(0))
    begin
        if rst_i = '0' then
            clk_o <= '0';
        elsif rising_edge(chain(0)) then
            clk_o <= not clk_o;
        end if;
    end process t_FF;
end Behavioral;
```

- En el caso de FPGAs debemos fijar las LUTs y el enrutado
- Método más común es usar directrices como DONT_TOUCH o fijar las LUTS a mano (Mediante coordenadas)
- El retardo de un inversor es "0" así que hay que simular después de Place & Route
(Post-Implementation)

En el fichero de constraints .xdc podemos usar las siguientes directivas
Para fijar los LUTS

```
set_property BEL <LUT name> [get_cells {<Inverter_name>}]
set_property LOC <SLICE Coords> [get_cells {<Inverter_name>}]
```

Para fijar los PINs

```
set_property LOCK_PINS {<In:Out>} [get_cells {<Inverter_name>}]
set_property LOCK_PINS {I0:A6 I1:A5 I2:A4 I3:A3 I4:A2 I5:A1} [get_cells
    ↪ {PUF/ring0[*].ro0/LUT6_NAND0}]
```

Para habilitar bucles combinacionales

```
set_property ALLOW_COMBINATORIAL_LOOPS true [get_nets {MyPUF/ring0[0].ro0/o}]
```

HAY QUE CORREGIR LOS ERRORES

■ ECCs

- Hamming Codes, Reed-Solomon, BCH, LDPC, Polar Codes
- Si la tasa de errores es alta, hay que usar *debiasing schemes* (con los problemas que conlleva)

■ Fuzzy Extractors

- Son muy eficaces
- Suelen ser más complejos y también requieren *helper data*

Sin embargo, los PUFs son funciones físicas muy complejas, y se requieren más investigación

```
import numpy as np

puf_bits = np.array([1, 0, 1, 1])

def hamming_encode(bits):
    d1, d2, d3, d4 = bits
    p1 = d1 ^ d2 ^ d4
    p2 = d1 ^ d3 ^ d4
    p3 = d2 ^ d3 ^ d4
    return np.array([p1, p2, d1, p3, d2, d3, d4])

def hamming_decode(code):
    p1, p2, d1, p3, d2, d3, d4 = code
    s1 = p1 ^ d1 ^ d2 ^ d4
    s2 = p2 ^ d1 ^ d3 ^ d4
    s3 = p3 ^ d2 ^ d3 ^ d4
    syndrome = s1 + (s2 << 1) + (s3 << 2)

    corrected = code.copy()
    if syndrome != 0:
        corrected[syndrome - 1] ^= 1

    return corrected[[2,4,5,6]]
```

TABLE OF CONTENTS

Physical Unclonable Functions (PUFs)

Métricas y Análisis

Workshop Práctico: Extracción y Análisis de Datos Reales

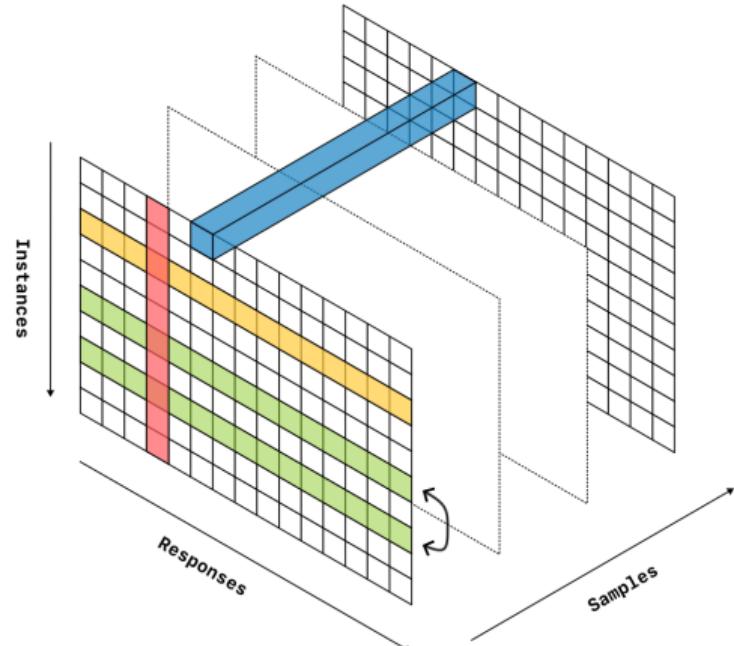
Workshop Práctico: Modelado y Aprendizaje Automático

Tecnologías Emergentes: PUFs de Memristores

COMO MEDIMOS LA CALIDAD DE UN PUF

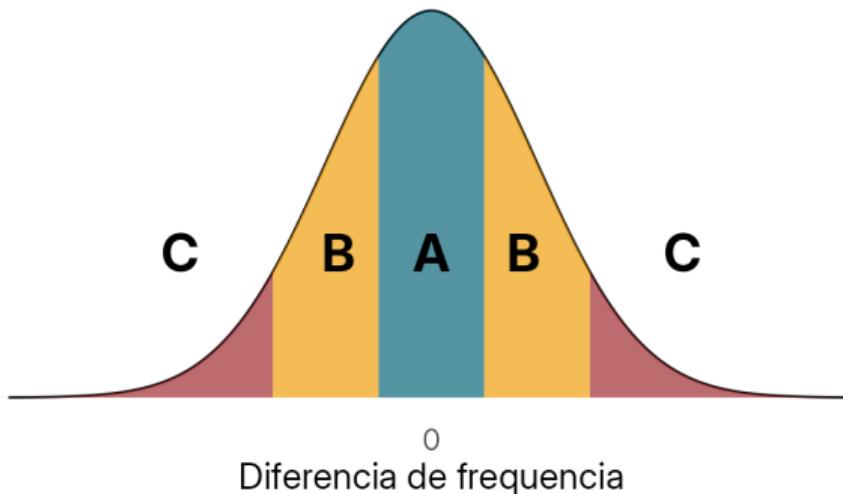
Se han propuesto muchas métricas, tests y maneras de medir la calidad de PUFs
Las más utilizadas a día de hoy son las propuestas por Maiti [1]

- ❑ **Uniformity:** Mide la distribución de valores en cada dispositivo
- ❑ **Bit-aliasing:** Mide la distribución de valores en cada respuesta
- ❑ **Uniqueness:** Mide como de diferentes los dispositivos son entre si
- ❑ **Reliability:** Mide la fiabilidad de cada respuesta



HAY UNA RELACIÓN ENTRE ENTROPIA Y FIABILIDAD

1. Del centro
2. De la mitad
3. De los extremos
4. Da igual



RELACIÓN ENTRE ENTROPÍA Y FIABILIDAD

La calidad de un PUF está determinada por la relación entre **entropía** y **fiabilidad**.

✚ Alta entropía

- Respuestas más impredecibles
- Mayor sensibilidad a la variabilidad
- Más susceptibles al ruido y a cambios ambientales

✚ Alta fiabilidad

- Respuestas más estables en el tiempo
- Menor tasa de error intra-dispositivo
- Normalmente menor entropía utilizable

Diseñar un PUF implica equilibrar estabilidad, entropía, consumo y área.

ATAQUES

- Hay todo un abanico de ataques invasivos y no invasivos para recuperar claves de Weak PUFs
- Focused Ion Beam (FIB), Envejecimiento acelerado (NBTI)
- No profundizaremos en los posibles ataques
- Pero hay que tener en cuenta que las Vulnerabilidades de un circuito no siempre vienen de donde se esperan

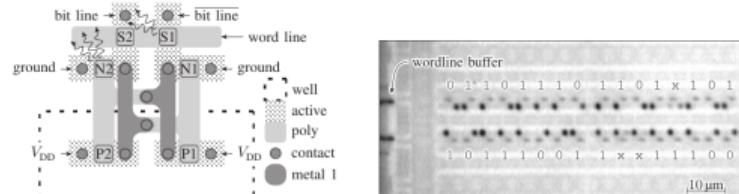


Figura: Ataque en SRAM-PUF mediante FIB[2]

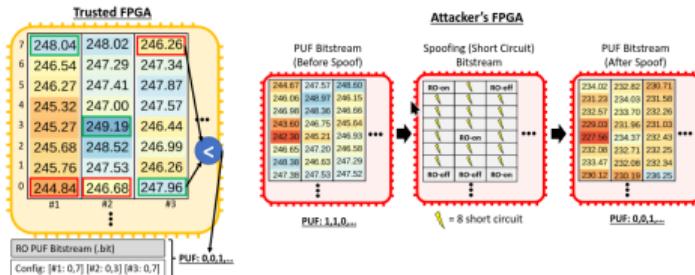


Figura: Ataque mediante NBTI en RO-PUF en FPGA[3]

TABLE OF CONTENTS

Physical Unclonable Functions (PUFs)

Métricas y Análisis

Workshop Práctico: Extracción y Análisis de Datos Reales

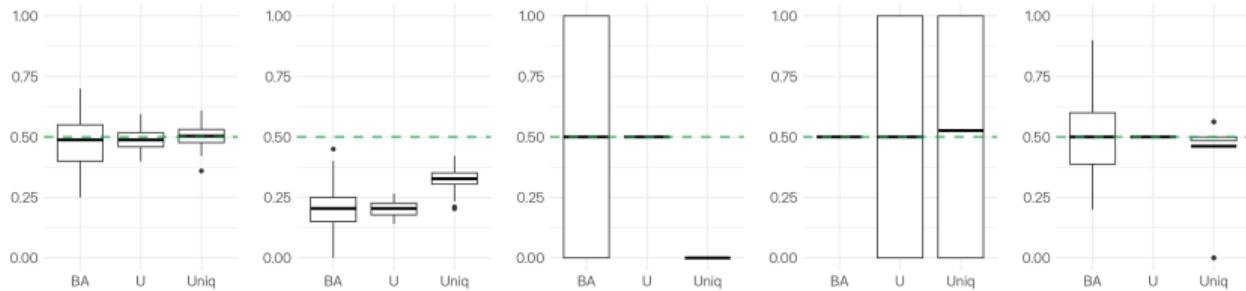
Workshop Práctico: Modelado y Aprendizaje Automático

Tecnologías Emergentes: PUFs de Memristores

WORKSHOP PRÁCTICO

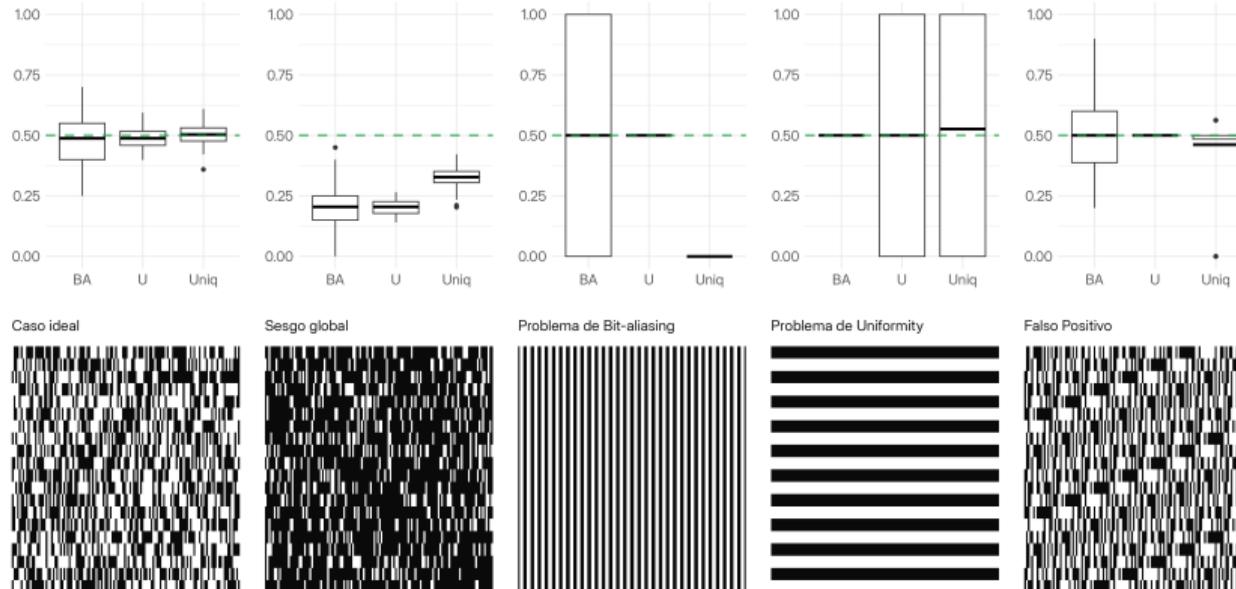
¿SON LAS MÉTRICAS SUFICIENTES?

Cuál de las siguientes distribuciones no es buena?



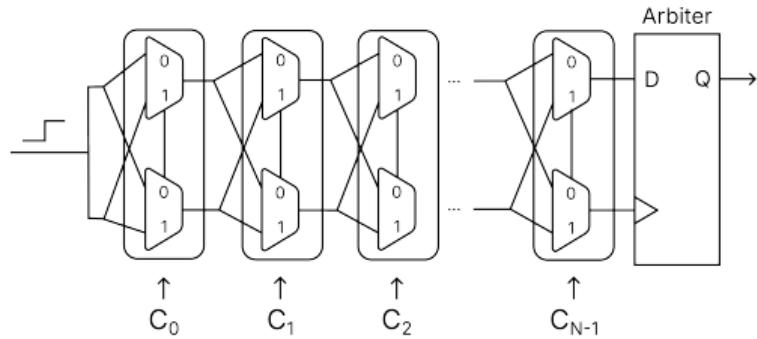
LAS MÉTRICAS NO LO SON TODO

Las métricas por si solas no nos dan toda la información.



ARBITER PUF

- ✓ diseño muy similar al RO PUF
- ✓ Consiste en una *carrera* entre dos señales, y el *arbitro* decide quien llega primero
- ✓ El camino de cada señal es seleccionado por los multiplexores
- ✓ Se puede ver como una función $f : \{0, 1\}^n \rightarrow \{0, 1\}$



EL ATAQUE DE LOS CLONES

Consideremos un ejemplo sencillo:

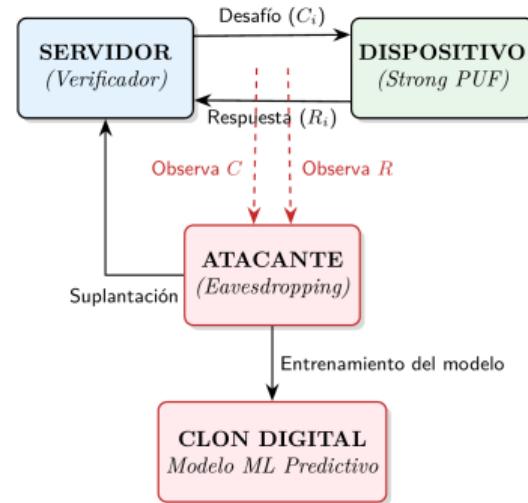
- ✖ Alice quiere verificar si Bob es realmente quien dice ser.
- ✖ Para ello, le hace una pregunta cuya respuesta sólo Bob debería conocer.

Un atacante puede:

- ✖ Escuchar el Challenge, y la Response
- ✖ Aprender la relación entre ambos

En el contexto de strong PUFs, este problema se formaliza como un problema de **aprendizaje (PAC learning)**

Existe un compromiso entre la **fiabilidad** del PUF y Su **resistencia al modelado**



POSIBLES CONTRAMEDIDAS

Introducir no linealidad

- **k-XOR Arbiter PUF:** XOR de las salidas de k Arbiter PUFs
- Incrementa la complejidad del modelo
- Dificulta los ataques de aprendizaje

Realimentación (feed-forward)

- La respuesta influye en etapas posteriores
- Se rompe la linealidad del modelo clásico

Compromiso de diseño

- Mayor seguridad \Rightarrow mayor área y consumo
- Menos CRPs útiles por bit generado
- El diseño debe equilibrar complejidad y robustez

Nuevas arquitecturas

- La no linealidad es parte intrínseca del diseño

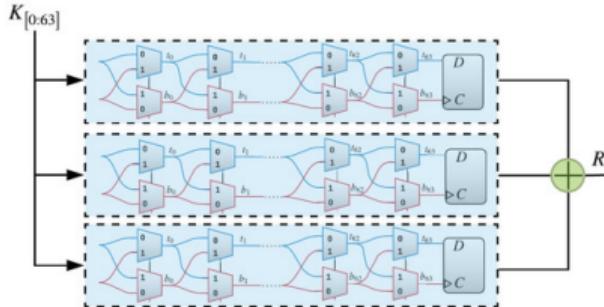


Figura: Diseño de un Arbiter PUF 3-XOR

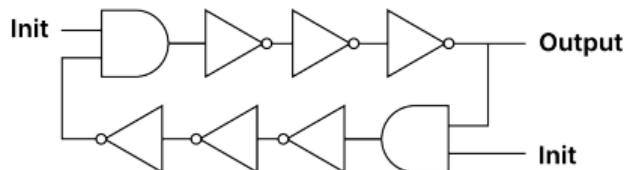


Figura: Diseño del bucle del TERO-PUF

TABLE OF CONTENTS

Physical Unclonable Functions (PUFs)

Métricas y Análisis

Workshop Práctico: Extracción y Análisis de Datos Reales

Workshop Práctico: Modelado y Aprendizaje Automático

Tecnologías Emergentes: PUFs de Memristores

WORKSHOP PRÁCTICO

TABLE OF CONTENTS

Physical Unclonable Functions (PUFs)

Métricas y Análisis

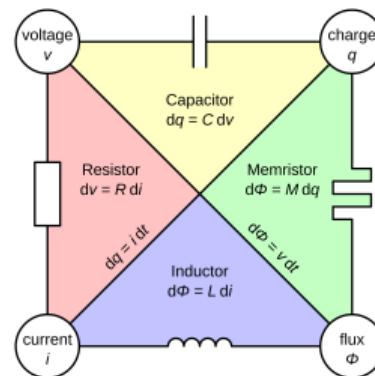
Workshop Práctico: Extracción y Análisis de Datos Reales

Workshop Práctico: Modelado y Aprendizaje Automático

Tecnologías Emergentes: PUFs de Memristores

MÁS ALLÁ DEL CMOS: MEMRISTORES

- El memristor fue originalmente "predicho" por Chua en 1971[4].
- El concepto se generalizó y hoy en día se habla de **sistemas memristivos**
- Debido a su **naturaleza estocástica** y a su **histéresis**, son una fuente de entropía con gran potencial
 - **Device-to-Device (D2D):** Memristores en distintos dispositivos se comportaran distinto ante los mismos estímulos
 - **Cycle-to-Cycle (C2C):** El mismo memristor se comportara de forma distinta en el tiempo incluso con los mismos estímulos



¿CÓMO FUNCIONA UN MEMRISTOR?

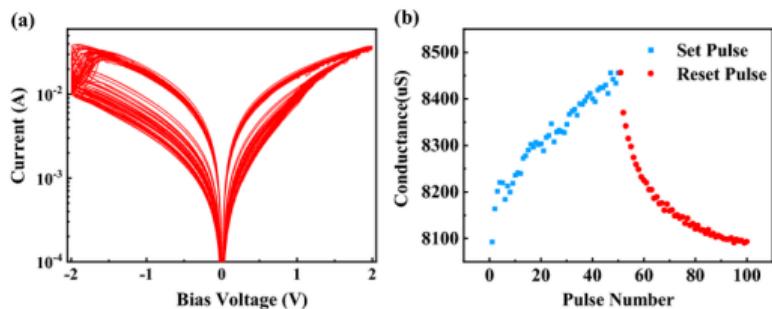


Figura: Imagen extraída de "Li, Jie, et al. "Reduction 93.7% time and power consumption using a memristor-based imprecise gradient update algorithm." Artificial Intelligence Review 55.1 (2022): 657-677."

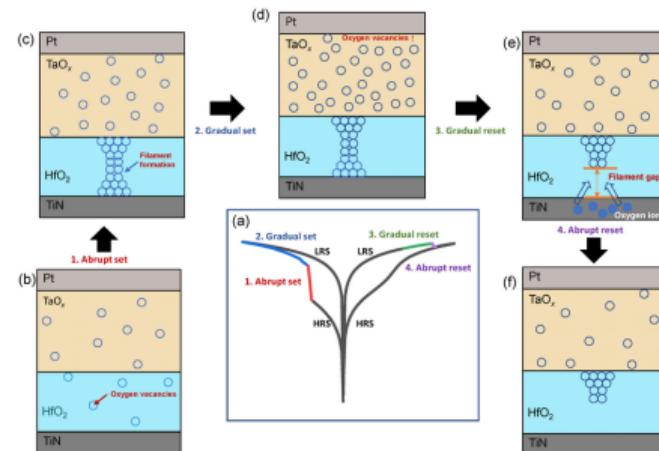


Figura: Imagen extraída de TaO_x/HfO₂ memristor using conducting defects and filaments. Pseudo-Interface Switching of a Two-Terminal TaO_x/HfO₂ Synaptic Device for Neuromorphic Applications

EJEMPLOS DE ARQUITECTURAS

Aplicar pulsos a un memristor hasta que cambie de estado y contar. El valor es único por memristor y dispositivo.

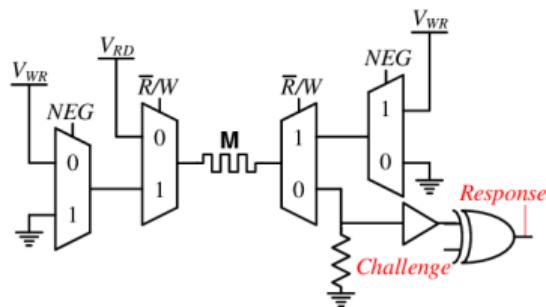


Figura: Diseño de 1-bit memristive PUF de "Foundations of memristor based PUF architectures"

Programar varios memristores con un mismo número de pulsos y comparar sus resistencias finales

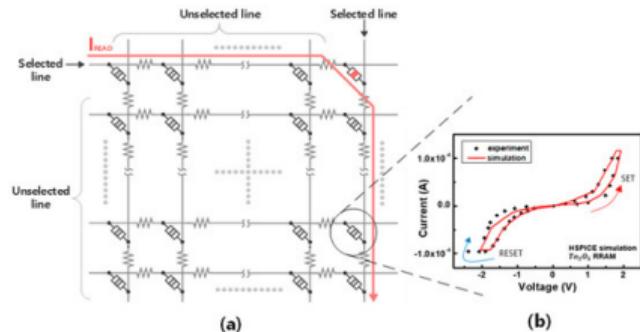


Figura: Crossbar Array de "Multibit-Generating Pulsewidth-Based Memristive-PUF Structure and Circuit Implementation"

- Ambos diseños sufren C2C y corrientes parásitas que introducen sesgos adicionales
- Además los memristores tienden a tener menos variabilidad entre dispositivos en tecnologías actuales

Los sistemas memristivos no son las únicas herramientas:

- Spin Transistor Torque (STT)
- FeRAM
- Fotónica

Computación neuromórfica

- Arquitecturas inspiradas en el cerebro
- Computación distribuida y analógica
- Tecnologías emergentes son aptas debido a su similaridad

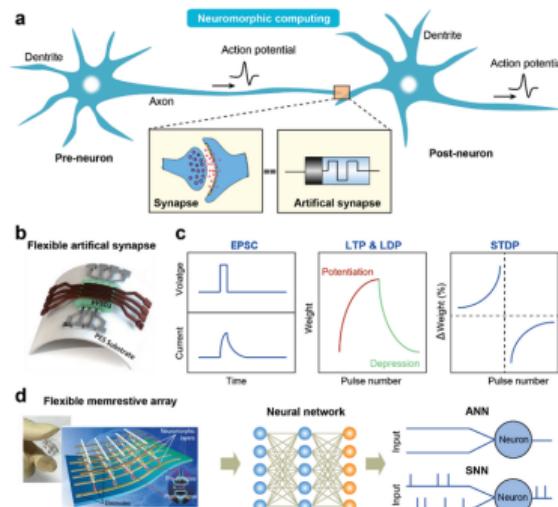
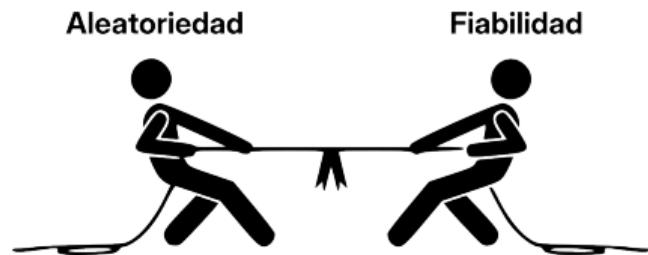


Figura: Memristores para la computación neuromórfica en "Artificial Skin Perception"

TIRA Y AFLOJA

- Las mismas propiedades que hacen que los sistemas memristivos sean una buena fuente de entropia, hacen que su uso en seguridad no sea fiable.
- Son analogicas, por lo que siempre habra pequenas variaciones
- Son sensibles a corrientes parasitas y a ruido
- La variacion C2C hace que sea muy dificil crear un PUF fiable



RESUMEN Y CONCLUSIONES

- ✖ La seguridad basada en NVM presenta vulnerabilidades reales.
- ✖ La raíz de confianza (Root of Trust) en hardware es un elemento crítico de cualquier sistema seguro.
- ✖ La variabilidad de fabricación, tradicionalmente un problema, puede explotarse como recurso.
- ✖ Las PUFs permiten generar secretos sin almacenarlos explícitamente en memoria.
- ✖ Existen distintos tipos de PUFs (weak y strong), cada uno con ventajas y limitaciones.
- ✖ El diseño de PUFs implica compromisos entre entropía, fiabilidad, área y resistencia a ataques.
- ✖ Tecnologías emergentes, como sistemas memristivos, abren nuevas posibilidades y desafíos.

Mensaje final: la seguridad moderna requiere soluciones físicas adaptadas al proceso y a la tecnología.

Q & A

REFERENCES I

1. Maiti, A., Casarona, J., McHale, L. y Schaumont, P. *A large scale characterization of RO-PUF.* en *2010 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)* (2010), 94-99.
2. Helfmeier, C., Boit, C., Nedospasov, D. y Seifert, J.-P. *Cloning physically unclonable functions.* en *2013 IEEE International Symposium on Hardware-Oriented Security and Trust (HOST)* (2013), 1-6.
3. Cook, H., Thompson, J., Tripp, Z., Hutchings, B. y Goeders, J. *Cloning the Unclonable: Physically Cloning an FPGA Ring-Oscillator PUF.* en *2022 International Conference on Field-Programmable Technology (ICFPT)* (2022), 1-10.
4. Chua, L. Memristor-the missing circuit element. *IEEE Transactions on circuit theory* **18**, 507-519 (1971).