José Luis Caro Bozzino

tfm  Master en Ciberseguridad

cRIPTOGRAFÍA LIGERA APLICADA A IOT

Contenido

[1. Resumen 3](#_Toc106356805)

[2. Incidentes destacados 3](#_Toc106356806)

[2.1 Botnet Mirai 3](#_Toc106356807)

[2.2 Ataque a coches Jeep 4](#_Toc106356808)

[2.3 Vulnerabilidades en los monitores de frecuencia cardiaca para bebés Owlet 6](#_Toc106356809)

[3. STATE OF THE ART 8](#_Toc106356810)

[3.1 Cifrado en bloque 10](#_Toc106356811)

[3.1.1 Present 10](#_Toc106356812)

[3.1.2 SIMON 11](#_Toc106356813)

[3.1.3 SPECK 12](#_Toc106356814)

[3.2 Funciones Hash 13](#_Toc106356815)

[3.2.1 PHOTON 14](#_Toc106356816)

[3.2.2 QUARK 15](#_Toc106356817)

[3.3 Cifrado en flujo 15](#_Toc106356818)

[3.3.1 Grain 15](#_Toc106356819)

[3.3.2 Trivium 16](#_Toc106356820)

[3.3.3 MICKEY 16](#_Toc106356821)

[3.4 Cifrados MAC 17](#_Toc106356822)

[3.4.1 Chaskey 18](#_Toc106356823)

[3.4.2 LightMAC 18](#_Toc106356824)

[4. Bibliografía 20](#_Toc106356825)

[4.1 Recursos online 20](#_Toc106356826)

[4.2 Imágenes 20](#_Toc106356827)

# Resumen

El presente Trabajo Final de Máster busca concienciar sobre los peligros del uso de dispositivos IoT que no han sido diseñados con la seguridad en mente, demostrando mediante eventos ocurridos en los últimos años las consecuencias de esto, así como enumerando algunos de los algoritmos criptográficos ligeros que podrían ser usados para evitar esto.

# 2. Incidentes destacados

Para ponernos en contexto sobre la importancia de aplicar medidas criptográficas a la hora de interactuar con dispositivos IoT, comentaremos una serie de incidentes de seguridad ocurridos en distintas partes del mundo, que ponen sobre la mesa las nefastas consecuencias que puede acarrear el ignorar un aspecto tan importante como es la seguridad en estos dispositivos de uso diario.

## Botnet Mirai

El caso de la botnet Mirai es uno de los más conocidos y al mismo tiempo uno de los más inquietantes.

Mirai se trata de un malware cuyo funcionamiento se basa en realizar amplios barridos de direcciones IP en busca de dispositivos (principalmente IoT) vulnerables a ataques de fuerza bruta, utilizando para ello un diccionario de contraseñas.

Una vez un dispositivo es infectado por Mirai, este no cambia su comportamiento, pero alberga el malware “latente”, esperando órdenes, lo que hace que en el momento en el que el atacante con acceso a esta botnet lo desee, pueda utilizar de forma simultánea millones de dispositivos para realizar ataques DDoS desde todo el mundo.

El código de este malware se ha publicado en foros de hacking en numerosas ocasiones, y se ha vuelto bastante sencillo de encontrar.

Esta pieza fue construida por Paras Jha Fanwood, Josiah White y Dalton Norman, quienes fueron sentenciados a servicio comunitario.

Curiosamente, el malware Mirai tiene una serie de tablas configuradas con máscaras de red a las que no infecta, entre las cuales se encuentran las pertenecientes al Servicio Postal de Estados Unidos, el Departamento de Defensa o IANA.

Uno de los ataques DDoS más destacados en los que se empleó esta botnet fue el llevado a cabo el 21 de Octubre de 2016, que tuvo como objetivo al proveedor de servicios DNS Dyn, y que causó que algunas webs tan importantes como Netflix, GitHub o Twitter fueran inaccesibles.[1]

Diagrama

Descripción generada automáticamente

1. Estructura de la Botnet Mirai

## Ataque a coches Jeep

En el año 2015, durante la feria sobre ciberseguridad y hacking Black Hat USA, se comprobó que ciertos vehículos Jeep podían ser hackeados para poder manejarlos de forma remota, lo que podría ocasionar que un atacante provocase un accidente al secuestra un vehículo durante un trayecto.

Los autores; Charlie Miller y Chris Valasek, explicaron y demostraron el ataque que habían elaborado, impactando al público con la sencillez de este.

Este ataque se basa en conectarse a la señal WiFi del vehículo, que en circunstancias normales de activaba mediante una suscripción contratada por el propietario del vehículo.

La contraseña de esta señal consistía en una clave generada a partir de la fecha de puesta en marcha del vehículo y su centralita, con precisión de segundos.

Este método de generación de claves hace que un atacante que conozca el año de fabricación del vehículo y acierte el mes, solo tendría que enfrentarse a unos 15 millones de combinaciones posibles; una cifra bastante pequeña a la hora de llevar un ataque de fuerza bruta con las herramientas adecuadas.

Con ese conocimiento, el siguiente paso fue tratar de acelerar el proceso para evitar tener que mantenerse cerca del vehículo durante el tiempo que tarda en realizarse el ataque de fuerza bruta.

Fue entonces cuando Charlie y Chris descubrieron que la contraseña se generaba antes de que la centralita configurase la fecha y hora reales, en su lugar usando la fecha del sistema y sumándole los segundos que tardaba en inicializarse la centralita.

Pantalla de computadora con imágen de hombre

Descripción generada automáticamente con confianza media

2. Ejemplo de generación de contraseña

Con esto lograron el acceso a la unidad central del vehículo y pudieron extender el ataque al reproductor multimedia del vehículo, lo que permitía alterar la radio, así como el reproductor de música o el volumen de este.

Una vez consiguieron interceptar el reproductor multimedia, pudieron generar un payload camuflado como actualización del firmware del coche, con el que poder infectar el Bus CAN (red interna que gestiona las conexiones de todas las piezas). Esto no fue difícil ya que las actualizaciones no tenían comprobaciones de seguridad ni requerían ninguna clase de firma.

Con dicha actualización pudieron tomar el mando del controlador V850, lo que les permitió controlar la dirección, el motor, el sistema de bloqueo de puertas e incluso el termostato del vehículo. [2]

Este caso resulta especialmente perturbador, ya que pone de manifiesto como, si bien los dispositivos IoT nos ofrecen cantidad de opciones y personalización que hace unas décadas hubieran resultado impensables, también hacen que si la empresa encargada del diseño y fabricación del dispositivo no toma las suficientes medidas de seguridad, puede convertir algo tan familiar y a lo que estamos acostumbrados como un coche, en una trampa mortal si es interceptado por un atacante con suficiente conocimiento.

Un grupo de personas en un auditorio

Descripción generada automáticamente con confianza media

3. Demostración del hack en la Black Hat USA 2015

## Vulnerabilidades en los monitores de frecuencia cardiaca para bebés Owlet

En 2016 saltó la voz de alarma debido a una serie de vulnerabilidades descubiertas en los monitores de frecuencia cardiaca de la marca Owlet, que se colocan en el calcetín del recién nacido y se sincronizan con un hub, y en caso de detectar alguna anomalía, envían una alerta al smartphone de los padres.

El investigador especializado en seguridad informática Jonathan Zdziarski fue quien dio la voz de alarma, al descubrir que el monitor encriptaba la información del dispositivo, que era enviada a los servidores de la empresa propietaria, que serían quienes contactarían con los padres en caso de ser necesario. El problema es que la red Wi-Fi generada para que el dispositivo se conectase al hub estaba totalmente abierta, permitiendo a cualquier persona dentro del rango poder interceptar la información.

Además de esto, si el atacante envía comando HTTP simples al hub, puede hacer que este se desconecte de la red Wi-Fi doméstica, volviéndolo inútil, así como conectarlo a otra red creada por el atacante, permitiendo que no sólo este pueda interceptar la información enviada por el dispositivo, sino que puede hacer que esta nunca llegue a los servidores de la empresa, inutilizando el dispositivo y dejando expuesto al bebé en caso de que sufriera alguna complicación, ya que la alerta nunca sería enviada.

Este ataque es tan sencillo de realizar como realizar un escaneo de dispositivos con el puerto 80 abierto dentro de nuestra red doméstica, y una vez localizado, conectarnos a él, ya que no implementa ninguna medida de autentificación ni de autorización, ni a la hora de conectarnos a él ni a nivel de interfaz, lo que permite que un atacante pueda fácilmente eliminar la información sobre la red inalámbrica a la que este se conecta, o modificarla para vincularla a la red deseada. [3]

Este caso demuestra una vez más lo que hemos visto en durante esta sección, que un dispositivo que debería hacernos la vida más fácil, si no es diseñado con la seguridad en mente por parte del fabricante, puede acarrear nuevos problemas, que en casos como este son más grandes e importantes que las facilidades y beneficios que aporta su existencia.

Interfaz de usuario gráfica, Aplicación

Descripción generada automáticamente

4. Revisión actual del Owlet Sock

# STATE OF THE ART

En un mundo cada vez más dominado por un IoT que, si bien podríamos considerar en fase relativamente temprana, ya ha avanzado hasta cuotas que hace unos años eran inimaginables (se calcula que en 2022 existen alrededor de 18.000.000.000 dispositivos IoT), nos encontramos con ciertas convenciones sobre criptografía ligera aplicada a estos dispositivos de usos cotidianos.

Estos dispositivos tienen la necesidad de funcionar con procesadores reducidos y de baja potencia con el fin de primar su utilidad cotidiana por encima de su complejidad, lo que deja poco espacio para desarrollar medidas de seguridad en la mayoría de los casos. Cabe destacar el caso de los dispositivos basados en etiquetas RFID como podrían ser las pulseras de acceso a centros deportivos; estos dispositivos contienen información que debe protegerse, a pesar de ser tan sencillos en su construcción que no poseen ni batería propia, sino que se alimentan de la propia energía del lector cuando las acercamos.

Hay que tener en cuenta que los sistemas criptográficos modernos, en muchos casos requieren de una potencia computacional demasiado elevada para poder implementarse en estos dispositivos, que, por otra parte, necesitan poder cifrar información sensible y personal con el fin de evitar filtraciones de datos, suplantación de usuarios…

Esta tarea es compleja cuando tratamos de aplicarla a dispositivos basados en microprocesadores que en muchos casos no superan los 16 bytes de memoria RAM para un procesador de 4, 8 o 16 bits.

De esta forma, un algoritmo AES o RSA sería prácticamente imposible de implementar en uno de estos dispositivos, lo que nos sitúa constantemente en la tesitura de tener que encontrar un equilibrio entre ligereza y seguridad. A más seguro sea el algoritmo, peor será el funcionamiento del dispositivo o más potencia necesitará, lo que repercute directamente de forma negativa en el producto, que, por otro lado, mientras menos potente sea su cifrado, más peligroso se vuelve su uso y peor imagen dará de él.

Esta necesidad de seguridad, que a primera vista puede pensarse comúnmente que solo afecta a dispositivos como relojes digitales, hay que sumarle que cada día más dispositivos como bombas de insulina, marcapasos, wearables de todo tipo, sistemas de peaje, lectores de tarjetas de crédito contactless…

Es por ello que la necesidad de proteger la información manejada por dichos dispositivos, así como controlar el acceso a esta sin disminuir la eficiencia ni aumentar costes se ha convertido en uno de los mayores desafíos actuales de la criptografía.

Para empezar a profundizar podemos enumerar los algoritmos de cifrado ligero más utilizados actualmente para dispositivos enfocados al IoT, que serán analizados en el siguiente capítulo.

* Cifrado en bloque
  + PRESENT
  + SPECK
  + SIMON
* Funciones Hash
  + PHOTON
  + QUARK
* Cifrado en flujo
* Grain
* Trivium
* MICKEY
* MAC (Código de autenticación de mensajes)
* Chaskey
* LightMAC

## 3.1 Cifrado en bloque

Estos cifrados ligeros se han diseñado con el objetivo de poder crear un sustituto de AES que pueda funcionar de forma eficiente en dispositivos con poca capacidad de procesamiento, pero siempre con el objetivo de poder mantener un nivel de seguridad lo más cercano posible a pesar de trabajar con bloques más pequeños.

En esta sección trataremos algunos algoritmos ligeros de cifrado en bloque que he considerado interesantes a la hora de una posible implementación en dispositivos inteligentes de baja potencia.

### 3.1.1 Present

El algoritmo de cifrado Present fue desarrollado en 2007 por Orange Labs junto a la Universidad de Bochum y la Universidad Técnica de Dinamarca y está enfocado a etiquetas RFID.

La seguridad de estas etiquetas en el IoT es muy importante, ya que son estas las que permiten la identificación inequívoca de un equipo, y el no preocuparse por protegerlas puede dar lugar a ataques de suplantación.

Present es casi tres veces más ligero que AES y está pensado para aparatos con un consumo de energía muy bajo que necesiten una gran eficiencia. Ha sido incluido en el nuevo standard internacional de métodos criptográficos ligeros.

Se trata de un algoritmo compuesto por una red SP (Sustitución-Permutación) de 31 rondas. El bloque es de longitud 64 bits, y admite dos tamaños distintos de clave; 80 y 128 bits. [4]

Diagrama

Descripción generada automáticamente

5. Esquema de alto nivel de Present

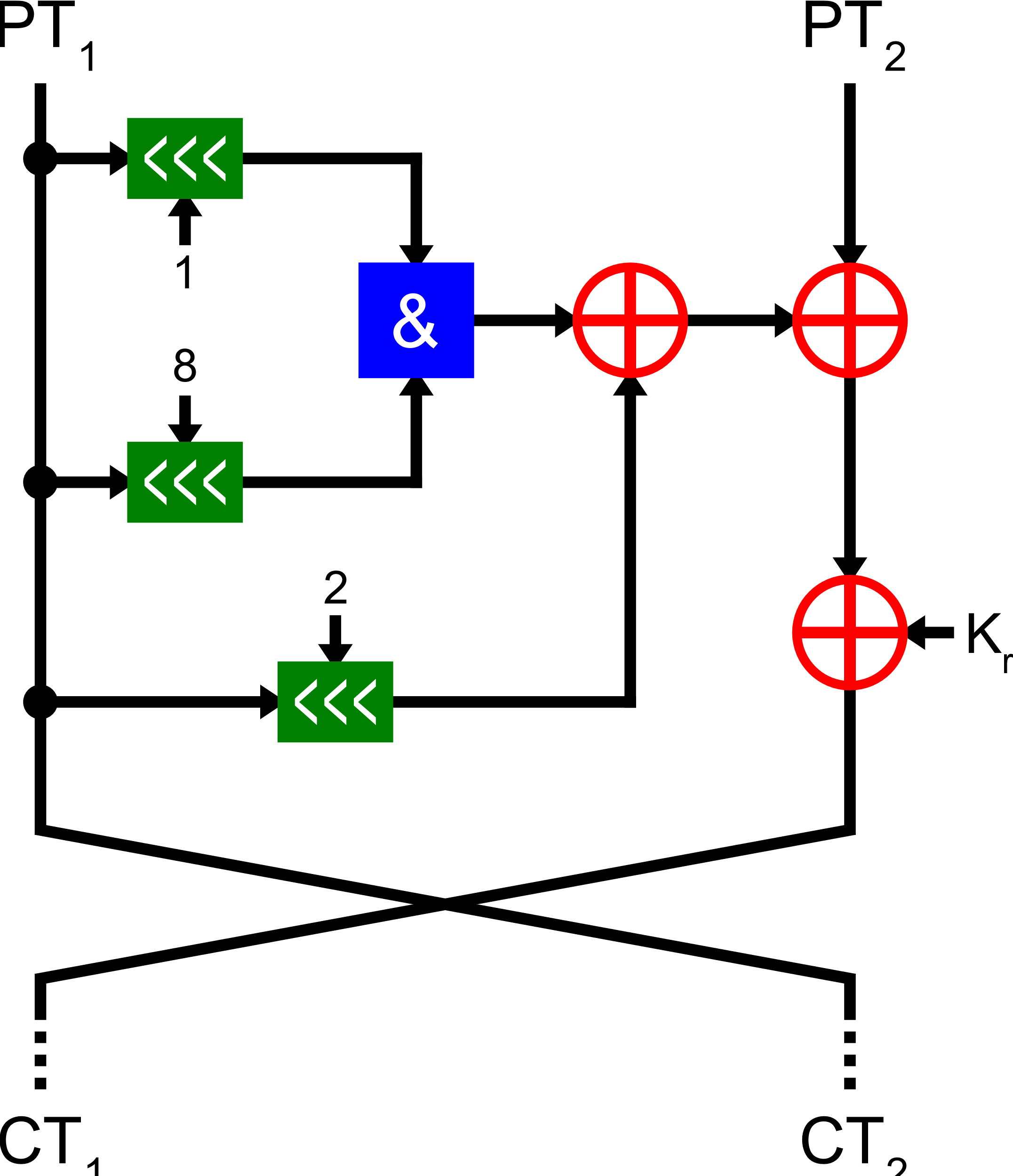
### 3.1.2 SIMON

La familia de algoritmos de cifrado SIMON fue desarrollada y lanzada por la NSA en el año 2013. SIMON está optimizado para su uso en hardware, mientras que SPECK, del que hablaremos a continuación, está enfocado al software.

Estos cifrados comenzaron su desarrollo en 2011 con una serie de necesidades para dispositivos IoT en mente, y la NSA presionó mucho para intentar incluirlos en el standard internacional.

Esto se ralentizó ya que países como Alemania, Japón o Israel se opusieron, alegando que la NSA estaba tratando de estandarizarlos a sabiendas de sus debilidades, cosa que la NSA sigue negando hoy en día. A pesar de esto, en 2018 fueron aceptados como standard para RFID.

Este cifrado se basa en una red de Feistel con una palabra de *n* bits, por lo que su longitud de bloque es de *2n*. Su clave *m* tiene longitud múltiplo de 2, 3 o 4 por *n.* A la hora de referirnos a un cifrado de SIMON también se suele hacer como Simon*n/nm*. Así, por ejemplo, un cifrado Simon64/128 tendría una palabra de 32 bits y una clave de 128 bits. [5]



6. Ronda de cifrado SIMON

### 3.1.3 SPECK

Como hemos mencionado antes, el cifrado SPECK es la contraparte de SIMON, con un enfoque en el software.

En este caso, el algoritmo consta de un bloque formado siempre por dos palabras, que pueden variar su longitud en bits entre 16, 24, 32, 48 o 64. Por otra parte, su clave puede ser de 2, 3 o 4 palabras. Cada ronda consiste en dos rotaciones, sumándole la palabra derecha a la izquierda, realizando una operación XOR entre la clave y la palabra izquierda y otro XOR de la palabra izquierda con la palabra derecha. [6]

El número de rondas depende de los parámetros seleccionados:

|  |  |  |
| --- | --- | --- |
| Tamaño del bloque en bits | Tamaño de la clave en bits | Rondas |
| 2 × 16 = 32 | 4 × 16 = 64 | 22 |
| 2 × 24 = 48 | 3 × 24 = 72 | 22 |
| 4 × 24 = 96 | 23 |
| 2 × 32 = 64 | 3 × 32 = 96 | 26 |
| 4 × 32 = 128 | 27 |
| 2 × 48 = 96 | 2 × 48 = 96 | 28 |
| 3 × 48 = 144 | 29 |
| 2 × 64 = 128 | 2 × 64 = 128 | 32 |
| 3 × 64 = 192 | 33 |
| 4 × 64= 256 | 34 |

Diagrama, Esquemático

Descripción generada automáticamente

7. Tres rondas de Speck con 2 palabras por clave

## 3.2 Funciones Hash

Las funciones hash son algoritmos criptográficos que reciben un dato de entrada en una salida en forma de texto que, dependiendo de la función, puede tener tamaño variable o siempre fijo independientemente del tamaño de los datos recibidos.

Son comúnmente utilizadas en criptografía para almacenar contraseñas en bases de datos sin necesidad de guardarlas en limpio, y a continuación vamos a tratar dos de las funciones hash ligeras más comunes en IoT.

### 3.2.1 PHOTON

Esta familia de funciones hash ligera fue diseñada por Jian Guo, Thoman Peyrin y Axel Poschmann como respuesta a la necesidad de una función hash ligera aplicable a etiquetas RFID.

El diseño de esta familia de funciones, al menos en el momento de su publicación, la convirtió en la función hash más compacta conocida hasta entonces.

Este algoritmo tiene una estructura inspirada en las construcciones llamadas “de esponja” o “sponge functions” en inglés (funciones criptográficas con un estado interno que pueden tomar como entrada un stream de bits de cualquier tamaño y producir una salida de un tamaño deseado), siendo su principal diferencia la capacidad de utilizar un bitrate distinto para la salida que el empleado en la entrada. [7]

Diagrama, Esquemático

Descripción generada automáticamente

8. Función esponja

Por otra parte, sus permutaciones están muy inspiradas en la estructura de cifrado AES, donde el estado interno puede ser representado por una matriz cuadrada de tamaño *d.d*, donde cada permutación interna se compone de 12 rondas.

Tabla

Descripción generada automáticamente con confianza media

9. Esquema de PHOTON

### 3.2.2 QUARK

La familia de funciones Quark fue desarrollada por Jean-Philippe Aumasson, Luca Henzen, Willi Meier y María Naya-Plasencia, y nació con la mente puesta en las etiquetas RFID.

Esta familia se basa en una construcción en esponja y un único nivel de seguridad con el fin de reducir las necesidades de memoria.

Está inspirada por otros protocolos ligeros como Grain o KATAN y se compone de tres instancias: u-Quark, d-Quark, and t-Quark.

Esta familia de funciones hash ha sido muy comparada con la familia anteriormente mencionada, PHOTON. En estas comparaciones se comprobó que, si bien ambas familias de algoritmos son muy similares, Quark está menos optimizada para un uso a nivel de software, aunque esto no tiene demasiada importancia ya que su implementación está pensada para ser realizada a nivel de hardware. En conclusión, ambos tienen un buen equilibrio entre rendimiento y seguridad.

Cabe también destacar que, en el momento de dicha comparación, ninguno de los dos algoritmos ha sido roto aún. [8]

## 3.3 Cifrado en flujo

Los algoritmos de cifrado en flujo son cifrados de clave simétrica en los que la entrada pasa por un flujo de claves o *keystream* generada a partir de la clave de cifrado, en las que se opera dígito por dígito para obtener una salida codificada.

Estos algoritmos suelen tener el problema de depender de un generador de números pseudoaleatorios (PRNG) que cumpla unos mínimos de calidad en lo que a ser criptográficamente seguros se refiere.

### 3.3.1 Grain

Este algoritmo de cifrado fue subido a eSTREAM en 2004, y está diseñado para funcionar en entornos de hardware restrictivos, lo que lo vuelve una buena opción para su aplicación en IoT.

Grain tiene un estado interno de 160 bits que consiste en 80 bits de LSFR y 80 bits de NLSFR.

A día de hoy, se le han encontrado númerosas vulnerabilidades, las cuales han sido en su mayoría corregidas en Grain 128a, que es la versión recomendada y que aporta seguridad en 128 bits y mecanismos de autenticación. [9]

### 3.3.2 Trivium

Algoritmo de cifrado en flujo síncrono diseñado por Christophe De Cannière y Bart Preneel.

Fue subido a eSTREAM y a pesar de no estar patentando, ha sido incluido en el standard internacional ISO/IEC 29192-3.

Posee un estado interno de 288 bits que consiste en 3 registros de desplazamiento de longitud variable. [10]

Diagrama, Dibujo de ingeniería

Descripción generada automáticamente

. Esquema de Trivium

### 3.3.3 MICKEY

El cifrado **Mutual Irregular Clocking KEYstream generator** ó Generador de keystream de irregularidad de reloj mutua en castellano, es un algoritmo de cifrado en flujo diseñado con el hardware en mente y enfocado a dispositivos con pocos recursos.

Es uno de los tres cifrados de hardware aceptados por eSTREAM junto a los mencionados anteriormente; Grain y Trivium.

Este algoritmo convierte una clave de 80 bits y un vector de inicialización con un tamaño variable, entre 0 y 80 bits, en un generador de claves con tamaño de hasta 240 bits. [11]

## 3.4 Cifrados MAC

Los cifrados MAC son aquellos que se encargan de cifrar los códigos de autenticación de mensaje o MAC por sus siglas en inglés.

Estos valores se crean a partir de una función hash con clave secreta K, que debe ser criptográficamente segura para poder resistir ante ataques.

Esta clave secreta solo es conocida por el emisor y el receptor, por lo que, con ella, el receptor puede recalcular el hash del mensaje y comprobar que cumple el principio de no repudio y que la información contenida en este no ha sido alterada.

Dentro de este tipo de funciones, actualmente podemos distinguir tres categorías para clasificarlas:

1. **CBC-MAC**: Estas funciones cifran el mensaje mediante algún algoritmo de cifrado en bloque en modo CBC (Cipher block chaining). De este modo se crea una cadena de bloques en los que el cifrado de cada uno depende del resultado de cifrar el bloque anterior, haciendo que cualquier cambio en un bit cambie completamente el resultado del cifrado, impidiendo que el atacante pueda modificar el mensaje sin ser detectado.
2. **HMAC**: Siglas de “Código de autenticación de mensaje basado en hash” son aquellas funciones MAC en las que se utiliza una función hash y una clave secreta para obtener como resultado un hash que permita demostrar el origen y contenido del mensaje. Se pueden construir en base a cualquier función hash, haciendo que su seguridad dependa de la función escogida, el tamaño de salida escogido para este hash y la clave privada utilizada.
3. **UMAC**: Por último, tenemos los cifrados MAC basados en un hash universal. Estas funciones utilizan una función hash escogida de entre varias mediante un proceso de selección desconocido para el atacante. Acto seguido, el código vuelve a encriptarse para tratar de dificultar la identificación del cifrado utilizado. De esta forma, solo el receptor es capaz de averiguar que función se ha utilizado para realizar el hash y usarlo para comprobar la integridad y el origen del mensaje.

Dentro de este tipo de cifrado, vamos a tratar algunos que en los últimos años han sido bastante relacionados con el mundo de la seguridad aplicada al Internet of Things.

### 3.4.1 Chaskey

Chaskey es una función MAC basada en permutaciones diseñada por Nicky Mouha, Bart Mennink, Anthony Van Herrewege, Dai Watanabe, Bart Preneel e Ingrid Verbauwhede.

Esta función se compone de una clave de 128 bits K y, usando una permutación de 128 bits π, convierte un mensaje m en bloques de 128 bits.

Estas permutaciones están basadas en ARX (Addition-Rotation-XOR).

Chaskey está diseñado para microcontroladores de 32 bits, es implementable entre distintas plataformas y es resistente a ataques de temporización.

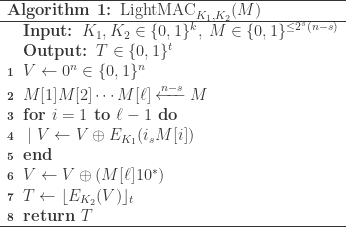
Además de todo esto, Chaskey no está patentado, lo que permite su uso de forma gratuita. [12]

### 3.4.2 LightMAC

LightMAC es una función MAC diseñada por Atul Luykx, Bart Preneel, Elmar Tischhauser y Kan Yasuda, publicada en 2016.

Dentro de las funciones MAC la podríamos englobar dentro de las CBCMAC, ya que se basa en bloques de cifrado.

Esta función permite usar PRESENT o SPECK como cifradores, por lo que su uso en dispositivos IoT está demostrado como viable y eficiente. [13]



11. Función LightMAC

# Bibliografía

## 4.1 Recursos online

[1] <https://www.cloudflare.com/es-es/learning/ddos/glossary/mirai-botnet/>

[2] <https://www.kaspersky.es/blog/blackhat-jeep-cherokee-hack-explained/6552/>

[3] <https://www.theregister.com/2016/10/13/possibly_worst_iot_security_failure_yet/>

[4] <http://lightweightcrypto.org/present/present_ches2007.pdf>

[5] <https://en.wikipedia.org/wiki/Simon_(cipher)>

[6] <https://www.cryptopp.com/wiki/SPECK>

[7] <https://sites.google.com/site/photonhashfunction/design>

[8] <https://link.springer.com/article/10.1007/s00145-012-9125-6>

[9] <https://www.ecrypt.eu.org/stream/e2-grain.html>

[10] <https://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf>

[11] <https://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf>

[12] <https://mouha.be/chaskey/>

[13] <https://tinycrypt.wordpress.com/2017/03/22/asmcodes-lightmac/>

## 4.2 Imágenes

1. <https://news.softpedia.com/news/there-are-almost-half-of-million-iot-devices-infected-with-the-mirai-iot-malware-509432.shtml>
2. <https://www.kaspersky.es/blog/blackhat-jeep-cherokee-hack-explained/6552/>
3. <https://www.kaspersky.es/blog/blackhat-jeep-cherokee-hack-explained/6552/>
4. <https://www.amazon.es/Owlet-Smart-Vigilabeb%C3%A9s-card%C3%ADaco-ox%C3%ADgeno/dp/B08CY71MZM>
5. <http://lightweightcrypto.org/present/present_ches2007.pdf>
6. <https://en.wikipedia.org/wiki/Simon_%28cipher%29>
7. <https://en.wikipedia.org/wiki/Speck_%28cipher%29>
8. <https://sites.google.com/site/photonhashfunction/design>
9. <https://sites.google.com/site/photonhashfunction/design>
10. <https://en.wikipedia.org/wiki/Trivium_%28cipher%29>
11. <https://tinycrypt.wordpress.com/2017/03/22/asmcodes-lightmac/>