Criptografía Ligera en Internet de las Cosas para la Industria

Eterovic, Jorge; Cipriano, Marcelo; García, Edith, Torres, Luis

Instituto de Investigación en Ciencia y Tecnología. Facultad de Ingeniería.
Vice-Rectorado de Investigación y Desarrollo.
Universidad del Salvador
Lavalle 1854. C1051 AAB. Buenos Aires, Argentina.
{jorge.eterovic; cipriano1.618; edithxgarcia}@gmail.com, torreslu@ar.ibm.com

Abstract. La Criptografía Ligera o Liviana (Lightweight Cryptography) es uno de los temas de actualidad de la Criptología. Una gran variedad de algoritmos "livianos" han sido diseñados para garantizar Confidencialidad, Autenticidad e Integridad de los datos en dispositivos de lo que se ha dado en llamar Internet de las Cosas (IoT por sus siglas en inglés). Algunos de ellos surgen del ámbito académico y se aplican en la Industria; otros son propietarios, desarrollados por las empresas para satisfacer sus requerimientos de seguridad. En este trabajo se presenta el estado del arte de algunos de tales algoritmos empleados en diferentes dispositivos IoT. Se describen brevemente sus características criptológicas generales y se muestran los diferentes ataques a los que fueron sometidos. Finalmente se enumeran algunas de las tendencias para el diseño e implementación de dichas primitivas.

Keywords: Criptografía Ligera, Lightweight Cryptography, Internet de las Cosas, IoT, Internet Industrial de las Cosas, IIoT, Criptosistemas Livianos.

1. Introducción

En la actualidad el concepto "*Internet de las Cosas"* o *IoT* [1] es una de las expresiones más usadas en diferentes ámbitos. Es un término muy amplio que abarca a diferentes dispositivos y tecnologías. Aunque ya se observa un presente con una plétora de estos dispositivos, en un futuro cercano habrá más y más dispositivos aún conectados entre sí. El concepto de *IoT* surgió en 1999 cuando el investigador *Kevin Ashton* se refirió a él en una conferencia; y su relación a las investigaciones en el campo de la *Identificación por Radio Frecuencia* o *RFID*[2] y en la Tecnología de las Redes de Sensores. En este último caso se conectan una gran cantidad de nodos muy simples con un *Centro de Control (CC)*.

Algunos dispositivos IoT funcionan a batería y/o generan su propia energía usando por ejemplo, paneles solares. Para garantizar la *Confidencialidad*, *Autenticidad* e *Integridad* de los datos e información que viaja a través de los canales de comunicación entre los sensores y su CC, se deben emplear algoritmos criptográficos.

[1] Internet of Things: interconexión de sensores y objetos cotidianos, mediante Internet.
[2] Radio Frequency IDentification.

Algunos de estos equipos tienen grandes procesadores por lo que es posible usar los mismos algoritmos criptográficos usados en las PCs y Servidores. Por ejemplo los teléfonos celulares, tablets y demás objetos similares. Otros, sin embargo, necesitan procesadores de muy baja potencia, de la cual sólo puede dedicarse una pequeña parte a la seguridad. Asimismo podrían estar limitados en el consumo de energía, capacidad de memoria o en su reducido tamaño, debido a las funcionalidades para los que fueron diseñados. Este tipo de equipos se caracterizan por sus capacidades restringidas o limitadas.

Los algoritmos tradicionales podrían incurrir en un alto consumo de recursos o provocar retardos en la transmisión de datos cuando se los hace trabajar en estos dispositivos. Por lo tanto, dadas las capacidades restringidas y que la seguridad es fundamental para su funcionamiento, los algoritmos criptográficos utilizados deben ser tan "livianos" como sea posible.

Dado que los chips *RFID* son usados para identificación, la información asociada a los chips debe ser cifrada para, entre otras cosas, prevenir el acceso no autorizado a la misma. Debido a que estos cuentan con un número muy pequeño de compuertas lógicas *(GE: Gate Equivalent)* y además tienen muy poca capacidad de consumo energético, se requieren algoritmos criptográficos con diseños específicos.

Es aquí, donde la Criptología Liviana se ha propuesto cubrir las necesidades de seguridad en IoT. Sus diseños varían ampliamente, pero en todos los casos procuran satisfacer los recursos limitados de las "cosas". Siendo muy amplia la propuesta, es posible en todos los casos establecer una serie de tendencias y criterios que persiguen los algoritmos a la hora de ser implementados.

2. Algoritmos livianos basados en Stream Ciphers

A continuación se muestran algunos algoritmos livianos de tipo *Stream Cipher* utilizados en el ámbito industrial. Se describen las características generales de los mismos y los ataques criptográficos a los que fueron sometidos.

A5/1.

El diseño exacto del algoritmo no fue claro en sus comienzos, una primera aproximación de su funcionamiento interno fue publicada en 1994 [1]. El algoritmo genera un *Keystream* a partir de un *Vector de Inicialización (IV* por sus siglas en inglés*)* de *22 bits* y de una clave secreta de *64 bits,* utilizando tres *Registros de Desplazamiento con Realimentación Lineal* (*LFSR)*[3] diferentes cuyas longitudes suman *64 bits*. Ataques prácticos se han implementado usando *Time-Memory Trade-Offs* a través de *Rainbow Tables (RTs),* sacando ventaja del hecho que la *Función de Actualización* de su estado interno no es biyectiva [2]. Se requieren sólo 2^{24} pasos luego de la construcción de las *RTs*. Además *10 bits* de la clave fueron siempre seteados a *0* en muchas implementaciones. El protocolo *2G GSM* todavía usa este algoritmo.

[3] Linear Feedback Shift Registers.

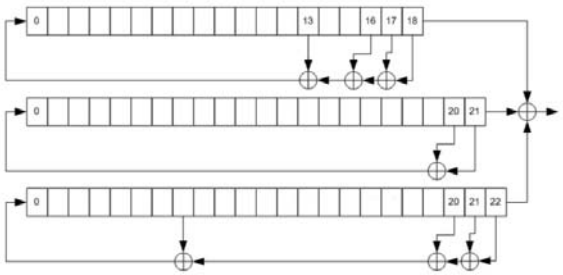

Figura 1: esquema del algoritmo A5/1

A5/2.
El algoritmo es de alguna manera similar al ***A5/1*** pero aún mucho más débil. Fue destinado para ser usado en algunos países con restricciones de exportación de Estados Unidos. Es vulnerable a ataques de "*texto cifrado solamente*" con una complejidad de 2^{16} pasos, usando la redundancia de los *códigos correctores-detectores de errores*. Los ataques posibles requieren un pre-cálculo de complejidad práctica [3, 4]. Por su debilidad, el consorcio *3GPP* recomienda fuertemente la no utilización de este algoritmo a partir de mediados de 2006.

Familia A5-GMR-x
Estos algoritmos criptográficos fueron adoptados por protocolos de telefonía satelital. Los algoritmos ***A5-GMR-1*** *y* ***A5-GMR-2*** fueron obtenidos por ingeniería inversa por *Driessen et al.* [5]. Aunque son diferentes entre sí, son fácilmente vulnerados.

A5-GMR-1 es una variante de *A5/2* con un estado interno que consiste en cuatro *LFSRs* cuyas longitudes suman *81bits.* Los registros son cloqueados irregularmente como en el *A5/2*. Puede ser atacado usando solamente "*texto cifrado conocido*", invirtiendo 2^{21} matrices triangulares de tamaño *532x532*, teniendo una complejidad aproximada a $2^{38.1}$ operaciones, previamente haciendo un significativo pre-cálculo.

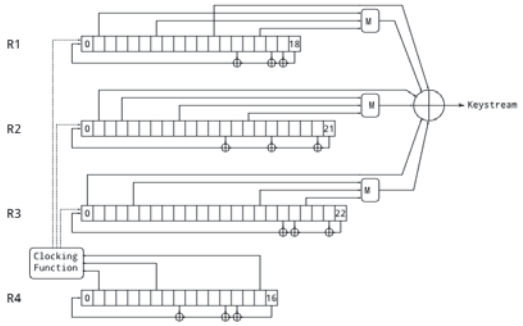

Figura 2: esquema del algoritmo A5-GMR-1

A5-GMR-2 es un stream cipher en bytes, con una estructura más sofisticada basada en 3 componentes llamadas *F*, *G* y *H* por *Driessen et al*. La estructura *H* utiliza dos *S-boxes* del algoritmo *DES:* las cajas *2* y *6.* Un ataque práctico con pocos datos y baja

complejidad en tiempo se puede encontrar en [6]. Se necesita suponer a lo sumo *32 bits* usando sólo un frame de *15 bytes* con un complejidad promedio de 2^{28}.

Atmel Ciphers.
SecureMemory (SM), CryptoMemory (CM) y *CryptoRF(CR)* son una familia de chips con amplias aplicaciones. Utilizan stream ciphers similares llamados *Atmel Ciphers*. Son algoritmos propietarios que fueron obtenidos por ingeniería inversa y atacados por *García et al*. [7]. Otros ataques mucho más eficientes fueron propuestos por *Biryukov et al*. [8] quebrando el cifrado de *SM* con una complejidad en tiempo de $2^{29.8}$ segundos, usando 1 frame y el cifrado de *CM* con una complejidad en tiempo 2^{50} segundos, usando 30 frames y alrededor de 530 Mb de memoria. El algoritmo consiste en 3 *NLFSRs*[4] con un tamaño total *117 bits.*

***Crypto-1*.**
Es un stream cipher usado por las tarjetas inteligentes de la línea *Mifare classic* de *NXP Semiconductors*, antes *Philips Semiconductors*. Fue recuperado por ingeniería inversa por *Nohl el al.* [9]. Está basado en un *LFSRs* de *48 bits* combinado con varias funciones Booleanas no lineales. Fue sucesivamente atacado por varios grupos de investigación [10, 11] con una complejidad en tiempo de 2^{32}. Estas tarjetas han sido usadas desde 1998 pero la fecha exacta de su diseño no es clara.

***Content Scrambling System (Css)*.**
Este algoritmo era aplicado en el cifrado de DVDs para considerar los *Derechos Digitales (DRMs Digital Rights Managements)*. Consiste en dos *LFSRs* de longitudes de *17 y 25 bits* que generan *dos bytes* en paralelo. Luego eran sumados módulo 2^8 para obtener *1 byte* de keystream. A diferencia de la mayoría de los cifradores de flujo, este *keystream* no es simplemente xoreado con el texto claro. El texto claro primero se modifica pasando por una *S-box* biyectiva de *8 bits* cuyo resultado es sumado con el *keystream* para obtener el texto cifrado. Esta operación es a menudo llamada *Mangling Step*. Una descripción completa del algoritmo puede encontrarse en [12, 13]. El algoritmo recibió varios ataques importantes, puesto que la longitud de la clave secreta es de *40 bits*. El cifrado solamente es vulnerable a un ataque de fuerza bruta con una complejidad en tiempo de 2^{40}.

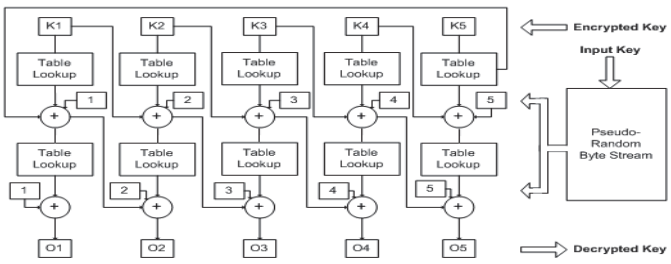

Figura 3: esquema del algoritmo Css.

[4] NonLinear-Feedback Shift Register: Registros de Desplazamiento con Realimentación No Lineal.

Common Scrambling Algorithm (Csa-SC).
El *CSA* es utilizado para proteger la emisión de televisión digital. Consta de dos cifrados en cascada [14]. El primer cifrado es un block cipher el cual se llama *Csa-BC,* el segundo es un stream cipher llamado *Csa-SC*. El stream cipher se basa en dos *FSRs (Feedback Shift Registers)* de 20 celdas de *4 bits* de contenido. La función de realimentación de los registros involucra, entre otras cosas, varias *S-boxes* de 5x2 bits. Se combinan usando suma módulo 2^4 para extraer *2 bits* del *keystream* del estado interno y de los dos registros, en cada clock. Este algoritmo tiene varias vulnerabilidades [14]. A menudo el *keystream* tiene ciclos muy cortos. Es posible recuperar la clave secreta solucionando alrededor de 2^{28} sistemas de 60 ecuaciones lineales con 40 incógnitas lo cual demanda una complejidad de a lo sumo $2^{45.7}$ operaciones.

Dsc.
El *Dect Standard Cipher (Dect "Digital Enhanced Cordless Telecommunications")* o *Dsc* es un algoritmo usado para cifrar las comunicaciones de los teléfonos inalámbricos. Los primeros ataques se centraron en su protocolo y en sus fallas en la implementación [15]. Fue paulatinamente descripto por ingeniería inversa y se encontraron ataques que requieren sólo alrededor de 2^{15} muestras de *keystream* y 2^{34} cifrados, lo cual lleva un par de horas para recuperar la clave secreta. En ese trabajo [16] los autores lo describen como "un stream cipher asincrónico con baja complejidad no lineal que utiliza una clave secreta de *64* bits y un IV de *35 bits*". Su estructura basada en *LFSRs* que cloquean irregularmente recuerda al algoritmo *A5/1*.

E0.
La privacidad del protocolo *Bluetooth* está basada actualmente en el algoritmo *AES*, que reemplazó al stream cipher llamado *E0*. Su estado interno de *128 bits* está dividido en *4 LFSRs* y su función filtro tiene un diseño propio. Una descripción completa de *E0* se puede encontrar en los trabajos en donde se muestran ataques contra el algoritmo [17, 18, 19]. *Lu et al.* Encontraron un ataque que permite recuperar la clave secreta usando los primeros *24 bits* de $2^{23.8}$ frames, con una complejidad de 2^{38} operaciones.

Espresso.
Es un algoritmo diseñado por *Elena Dubrova y Martin Hell* [20] propuesto para proteger las comunicaciones *5G (Wireless Communication Systems).* Consiste en dos partes fundamentales: un *NLFSR* de *256 bits* en la configuración *Galois* y una *Función de Salida* no lineal de 20 variables. Utiliza una clave secreta de *128 bits* y un IV de *96 bits*. De acuerdo a los autores, es sumamente liviano: tiene *1497 GE*. Procesa *2.22 Gbits/seg* y una latencia de *232 ns.* La estructura del algoritmo *Grain* y *Espresso* son muy similares a partir de lo cual *Wang y Lin* [21] aplicaron un S*lide Attack* usando el análisis del *Grain*, desarrollaron un ataque de IV elegido que les permite recuperar la clave secreta de *128 bits* con solo dos pares de *IVs* relacionados, no más que 2^{42} *IVs* elegidos y con una complejidad computacional de 2^{64}. Así el stream cipher *Espresso* no es seguro para *128 bits* de clave.

Hitag2 y Megamos.
Estos stream ciphers son usados en los *Sistemas de Inmovilización de Automóviles (car immobilizers)* implementados por diferentes empresas que evitan que el motor del auto se encienda a menos que el "*transponder*" se encuentre cerca del vehículo. Inicialmente los algoritmos se mantenían secretos. Luego *Hitag2* fue publicado por *Wiener* y detallado en [22] y *Megamos* obtenido por ingeniería inversa por *Verdult* et al. [23]. Ambos algoritmos cuentan con un estado interno pequeño, de *48* y *57 bits* respectivamente. Los tamaños de clave, otras debilidades en los cifradores y en el protocolo que los utiliza, permiten ataques prácticos contra los dispositivos cuya seguridad depende de estos algoritmos. Por ejemplo, es posible atacar una clave de *Hitag2* usando 1 minuto de comunicación entre la clave y el automóvil con alrededor de 2^{35} mensajes encriptados. La clave secreta de *Megamos* puede ser recuperada con una complejidad en tiempo de 2^{48}, aunque ataques aún más eficientes son considerados cuando se tiene en cuenta el método de manejo de las claves usadas en el dispositivo.

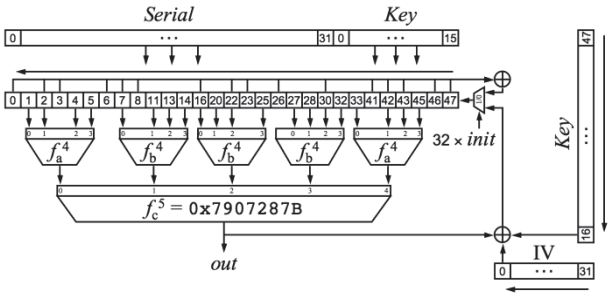

Figura 4: Diagrama del algoritmo HITAG2

iClass.
Es una familia de tarjetas inteligentes comercializadas en 2002. El stream cipher utilizado fue obtenido por ingeniería inversa y atacado por *García et al*. [24]. Tiene un estado interno de *40 bits*, registrando 2^{22} intentos de autenticación, la clave puede ser recuperada con 2^{40} cifrados.

Kindle Cipher (PC1).
Este algoritmo fue primero publicado en *Usenet* por *Alexander Pukall* en 1997. Este algoritmo no fue técnicamente diseñado en la industria ni por académicos. La empresa *Amazon* lo utilizó al menos hasta 2012 para los productos con derechos digitales *DRM,* protegiendo así sus e-books usando el formato de archivo *MOBI*. Utiliza una clave secreta de *128 bits* y un estado interno de *24 bits* que se actualiza usando diferentes operaciones incluyendo multiplicación modular. En cada clock el algoritmo genera *1byte*, resultando así un *keybytestream*. Ha sido quebrado por *Biryukov et al.* [25] usando por ejemplo 2^{20} textos claros conocidos y una complejidad en tiempo de 2^{31}. Ataques mucho más prácticos de *texto cifrado conocido* se pueden aplicar en ciertos contextos.

Oryx.
Mientras *A5/1* aseguraba las comunicaciones *GSM* en Europa, el algoritmo *Oryx* fue elegido por *Telecom Industry Association Standard (TIA)* para proteger las comunicaciones telefónicas en USA. Una descripción del algoritmo se puede encontrar en [26] donde ataques prácticos son presentados. El algoritmo utiliza una clave secreta de *96 bits*, un estado interno también de *96 bits* repartidos en *3 LFSRs de 32 bits* cada uno y una *S-box* de *8 bits* variable. Es posible atacarlo con una complejidad en tiempo de 2^{16} usando *25 bytes* de texto claro conocido.

3. Algoritmos livianos basados en Block Ciphers

A continuación se muestran algunos Block Ciphers livianos utilizados en el ámbito industrial. Se describen sus características y los ataques criptográficos a los que fueron sometidos.

BeepBeep.
El algoritmo *BeepBeep (Embedded Real-Time Encryption)* es un block cipher *autokey*. Trabaja con arreglos de *32 bits* y un IV que tiene diferentes opciones de acuerdo a la aplicación en la que se esté implementando. Fue diseñado por *Kevin Driscoll* de *Honeywell Laboratories* [27] para cubrir los requerimientos de seguridad que incluyen las comunicaciones wireless, manejo remoto de sistemas de control para plantas químicas y de energía, para manejo de redes distribuidas, acceso y control remoto. Salvo la presentación de *BeepBeep* y los requerimientos de criptografía en tiempo real *FSE (Fast Software Encryption)* en 2002, no se ha encontrado información acerca de análisis y ataques de este algoritmo.

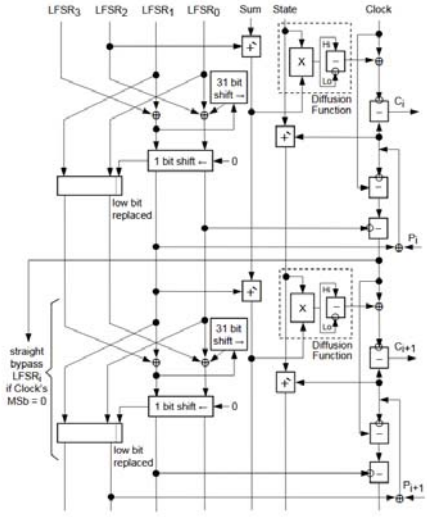

Figura 5: Diagrama de cifrado del algoritmo BeepBeep.

CMEA.
Este algoritmo fue usado por *TIA* para asegurar la transmisión de números de teléfono a través de las líneas telefónicas. Una buena descripción de este algoritmo se puede encontrar en [28], en el cual accidentalmente se describe un ataque contra él. Cifra un bloque de longitud arbitraria en bytes (usualmente en la práctica de *2* a *6 bytes*) usando una clave de *64 bits*. Es vulnerable a ataques de "*texto claro conocido"*, necesitando entre 40 y 80 bloques de datos, considerando 2^{24} y 2^{32} cifrados. Sus *S-boxes* parecen contener una estructura oculta según [29].

Cryptomeria.
En forma abreviada se le llama *"C2"*. Comparte la misma estructura que el *DES (Data Encryption Standard)*: cifra bloques de *64 bits* usando una clave secreta de *56 bits* y utiliza una *función Feistel* de *32 bits*. Opera mezclando una sub-clave de *32 bits* a través de una suma modular, luego usa una caja *S-box* de *8 bits* seguida por una permutación lineal de *32 bits*. La *S-box* es secreta, de manera tal que un ataque de recuperación de la caja ha sido propuesto en [30]. El mismo trabajo presenta un ataque de recupero de clave de complejidad en tiempo de 2^{48}. Este algoritmo fue destinado a ser usado en reproductores de DVD, en cuyo caso podría entenderse como un sucesor del algoritmo *Css,* y también para algunas tarjetas SD. En total, se utilizan 10 rondas lo que implica que sólo 10 llamadas a la caja son necesarias para cifrar un bloque de *64 bits,* comparado con las 160 llamadas a la caja que son necesarias para cifrar un bloque de *128 bits* usando *AES-128*.

Common Scrambling Algorithm (Csa-BC).
El algoritmo *Csa* usa un stream cipher (ya mencionado anteriormente) y un block cipher llamado *Csa-BC*. Cifra bloques de *64 bits* usando una clave de la misma longitud. Su estructura recuerda a una *Red Feistel* de *56 rondas*, usando ocho sub-bloques de *8 bits* cada uno. Las funciones están basadas en una única *S-box* aleatoria de *8 bits* (llamada *B*), componiendo una variante de ella con una permutación simple de bits llamada *sigma* (σ), obteniéndose $\sigma \circ B$. Una especificación completa del algoritmo se puede encontrar en [14]. Al momento, no se tiene conocimiento de algún tipo de ataque.

Dst40.
Este algoritmo fue recuperado por ingeniería inversa a través de información parcial descubierta en una patente y de su implementación física en un dispositivo [31]. Fue usado en dispositivos de *RFID* vendidos por *Texas Instrument*, para inmovilizadores de automóviles y pago electrónico. El cifrador trabaja con un bloque de *40 bits* con una clave de igual longitud, también usando 200 rondas de una *Red Feistel* asimétrica. La función *Feistel* aplica *38 bits* del estado interno y *40 bits* de una subclave y produce *2 bits* de salida componiendo varias funciones booleanas. Debido al tamaño de su clave de *40 bits* es posible efectuar un ataque por fuerza bruta.

Keeloq.
También llamado "*code-hopping encoder*". Diseñado en 1985, se le concede la patente en el año 1996. Es empleado por muchas compañías de automóviles como *Chrysler, Daewoo, Fiat, GM, Honda, Toyota, Volvo, Volkswagen,* en los sistemas de

apertura remota de puertas. Utiliza una clave secreta de *64 bits*, y bloques de *32 bits*. Es un cifrado iterativo, que consiste en 528 rondas. En sus comienzos el algoritmo era secreto. Pero en 2006 se filtraron sus especificaciones. Con esta información varios grupos de investigación presentaron ataques contra el algoritmo [32]. Por ejemplo, la clave secreta puede recuperarse usando 2^{16} textos claros conocidos y $2^{44.5}$ cifrados. Ataques mucho más eficientes por *canal lateral* se han propuesto para las implementaciones comerciales del cifrado. Una observación interesante es que en algunas marcas de automóviles cifra el texto claro 0 e incrementa la clave sumando 1 al entero representado por los bits de la clave secreta, de esta forma no utilizaría nunca la misma clave. Esto es por supuesto, bastante raro y se podría modelar *Keeloq* por un algoritmo dual en donde se tiene un texto claro de *64 bits* y una clave de sólo *32 bits*, con lo cual se rompería el algoritmo fácilmente por fuerza bruta.

4. Estrategias de Diseño de Algoritmos Livianos

La *Liviandad* puede ser vista como un conjunto específico de propiedades para tener en cuenta en el diseño de los algoritmos criptográficos livianos. Considerando los avances y los actuales resultados académicos, estos aspectos son adaptados de manera diferente en cada uno de ellos. Si bien hay nuevos aportes, prevalecen técnicas, metodologías y operaciones que son utilizadas en los criptosistemas convencionales.
A la hora de seguir una estrategia de diseño liviano, surgen entre otras cosas, una serie de características criptológicas a considerar como ser: Estructura General del Algoritmo, Esquema de Generación de Sub-claves y las Características de las Operaciones Lineales y No Lineales.
Los algoritmos diseñados a partir de estas propiedades, deben ser seguros, simples, rápidos y de alta performance.

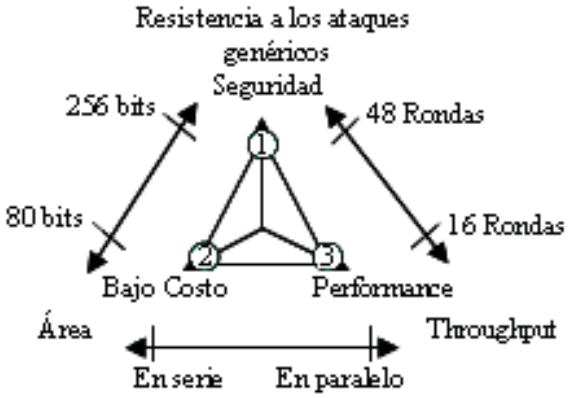

Figura 6: Inspirada en *A. Poschmann,* Lightweight Cryptography: Cryptographic engineering a pervasive world.

4.1 Estructura General del Algoritmo

De acuerdo a la forma de procesar la información se puede optar por un Cifrado en Bloques (Block Cipher), un Cifrado en Cadena (Stream Cipher) o un mecanismo "*híbrido"* entre los dos principios.
En el caso de los Block Ciphers una de las construcciones más utilizadas es la de SP-Networks (Redes de Sustitución-Permutación) y dentro de ellas las redes tipo Feistel.
Para los *Stream Ciphers*, la construcción es del tipo Vernam: el *texto claro* es sumado *XOR* con la secuencia de salida binaria pseudoaleatoria del algoritmo.

4.2 Esquema de Generación de Subclaves (Key Schedule)

El *KS* es el área de diferencia más notable entre los algoritmos livianos y los convencionales. Los algoritmos con *Generación de Subclaves* convencionales implementados sobre PCs tienden a ser más complejos por lo que requieren más tiempo de procesamiento y memoria, es por ello que el diseño de *KS livianos* no debe ser desatendido. Una construcción muy habitual de *KS* livianos meramente "selecciona" diferentes bits de una clave maestra secreta en cada ronda del algoritmo. El propósito de esta selección es que se necesite un cálculo muy sencillo para producir una subclave para cada ronda.

4.2 Operaciones Lineales y No Lineales

La no linealidad es una propiedad indispensable para cualquier primitiva criptográfica. Esta puede ser proporcionada por *Tablas de Sustitución (S-Boxes)* o a través del uso de operaciones aritméticas no lineales. Estas deben estar bien diseñadas para resultar ser resistentes a los ataques conocidos. A la vez deben ser *livianas*, esto significa que deben utilizar la menor cantidad de compuertas *(GE)* en su implementación. Para el caso de operaciones aritméticas no lineales, se considera la suma con acarreo módulo 2^{16}, 2^{32} o 2^{64} y también la multiplicación modular.
En la *Tecnología ARX (Addition, Rotation, XOR)* sólo estas tres simples operaciones son usadas en el diseño del algoritmo. Mostrándose como el nuevo paradigma a la hora de construir *rondas livianas*. Se puede agregar a este concepto de no linealidad el uso de *NLFSRs) y LFSRs clockeados* irregularmente.

4.3 Operaciones Lineales

Para poder garantizar una buena *difusión* de los bits de clave y de los bits del texto claro a lo largo del proceso de cifrado, transformaciones lineales deben aparecer combinadas con las operaciones no lineales. Esta propiedad puede alcanzarse con distintos procedimientos: con transformaciones lineales, con *LFSRs (Registros de Desplazamiento con Realimentación Lineal y* con matrices *MDS (Maximum Distance Separable)* conocidas en los códigos detectores-correctores de errores, tales como se puede observar en muchos de los algoritmos presentados.

5. Conclusiones y Futuros Trabajos

La Criptografía Ligera es el nuevo paradigma de la Criptología al poder dar respuesta a las necesidades crecientes de seguridad en dispositivos restringidos. En ellos no es posible aplicar los algoritmos más robustos de la Criptografía convencional ya que estos pueden llegar a consumir demasiados recursos.
En este trabajo se han presentado distintos algoritmos livianos que han sido diseñados para cubrir las necesidades específicas de la Industria. Asimismo se destacan sus características criptológicas, la resistencia a diferentes ataques genéricos y los ataques en los que algunos han sido vulnerados.
Finalmente se enumeran algunas tendencias en el diseño de estas primitivas, al observarse las mismas atravesando los principios y filosofías que les dieron origen.
Futuros trabajos de investigación podrían llevar adelante pesquisas respecto a las características en particular de las estrategias aquí presentadas, como así también indagar acerca de nuevas filosofías de diseño que pudieren aparecer en los próximos años.

6. Agradecimientos

El equipo de investigación agradece a las autoridades de la Facultad de Ingeniería de la Universidad del Salvador y al Vicerrectorado de Investigación y Desarrollo (VRID) a través de la Dirección de Investigación y del Instituto de Investigación en Ciencia y Tecnología, (en el cual se enmarca este proyecto Código VRID 1935 – Código Académico100091) por el apoyo recibido para poder llevar adelante esta investigación.

7. Referencias.

[1] Ross Anderson. A5 (Was: HACKING DIGITAL PHONES). uk.telecom (Usenet), https://groups.google.com/forum/?msg/uk.telecom/TkdCaytoeU4/Mroy719hdroJ#!msg/uk.telecom/TkdCaytoeU4/ Mroy719hdroJ, June 1994.
[2] Golic, J. Cryptanalysis of alleged A5 stream cipher. In Walter Fumy, editor, Advances in Cryptology – EUROCRYPT'97, volume 1233 of Lecture Notes in Computer Science, pages 239–255. Springer, Heidelberg, May 1997.
[3] Barkan, E. Biham, E. Keller, E. Instant ciphertext-only cryptanalysis of GSM encrypted communication. In Dan Boneh, editor, Advances in Cryptology – CRYPTO 2003, volume 2729 of Lecture Notes in Computer Science, pages 600–616. Springer, Heidelberg, August 2003.
[4] Barkan, E. Biham, E. Keller, E. Instant ciphertext-only cryptanalysis of GSM encrypted communication. Journal of Cryptology, 21(3):392– 429, July 2008.
[5] B. Driessen, R. Hund, C. Willems, C. Paar, and T. Holz. Don't trust satellite phones: A security analysis of two satphone standards. In 2012 IEEE Symposium on Security and Privacy, pages 128–142, May 2012.
[6] Li, R. Li, H. Li, C. Sun, B. A low data complexity attack on the GMR-2 cipher used in the satellite phones. Pages 485–501.
[7] Garcia, F. van Rossum, P. Verdult, R. Schreur, R. Dismantling SecureMemory, CryptoMemory and CryptoRF. In Proceedings of the 17th ACM Conference on Computer

and Communications Security, CCS '10, pages 250–259, New York, NY, USA, 2010. ACM.
[8] Biryukov, A Kizhvatov, I. Zhang, B. Cryptanalysis of the Atmel cipher in SecureMemory, CryptoMemory and CryptoRF. pages 91–109.
[9] Nohl, K. Evans, D. Starbug, S. Plötz, H. Reverseengineering a cryptographic RFID tag. In USENIX security symposium, volume 28, 2008.
[10] Courtois, N. Nohl, K. O'Neil, S. Algebraic attacks on the crypto-1 stream cipher in mifare classic and oyster cards. Cryptology ePrint Archive, Report 2008/166, 2008. http://eprint.iacr.org/2008/166.
[11] Golic, J. Cryptanalytic attacks on MIFARE classic protocol. In Ed Dawson, editor, Topics in Cryptology – CT-RSA 2013, volume 7779 of Lecture Notes in Computer Science, pages 239–258. Springer, Heidelberg, February / March 2013.
[12] Becker, M. Desoky, A. A study of the DVD content scrambling system (CSS) algorithm. In Proceedings of the Fourth IEEE International Symposium on Signal Processing and Information Technology, pages 353–356. 2004.
[13] Pedersen, L. Munk, K. Andersen, L. Cryptography – the rise and fall of DVD encryption. Available online at http://citeseerx.ist.psu.edu/viewdoc/download;jsessionid=3672D97255 B2446765DA47DA97960CDF?doi=10.1.1.118. 6103&rep=rep1&type=pdf, 2007.
[14] Weinmann R., Wirt, K. Analysis of the DVB Common Scrambling Algorithm. In David Chadwick and Bart Preneel, editors, Communications and Multimedia Security: 8th IFIP TC-6 TC-11 Conference on Communications and Multimedia Security, Springer US. 2004.
[15] Lucks, S. Schuler, A. Tews, E. Weinmann, R. Wenzel. M. Attacks on the DECT authentication mechanisms. In Marc Fischlin, editor, volume 5473 of Lecture Notes in Computer Science, pages 48–65. Springer, Heidelberg, April 2009.
[16] Nohl, K. Tews, E. Weinmann, R. Cryptanalysis of the DECT standard cipher. In Hong and Iwata [HI10], pages 1–18.
[17] Fluhrer, S. Lucks, S. Analysis of the E0 encryption system. In Vaudenay and Youssef [VY01], pages 38–48.
[18] Lu, Y. Vaudenay, S. Faster correlation attack on Bluetooth keystream generator E0. In Matthew Franklin, editor, Advances in Cryptology – CRYPTO 2004, volume 3152 of Lecture Notes in Computer Science, pages 407–425. Springer, Heidelberg, August 2004.
[19] Lu, Y. Meier, W. Vaudenay, S. The conditional correlation attack: A practical attack on bluetooth encryption. In Victor Shoup, editor, Advances in Cryptology – CRYPTO 2005, volume 3621 of Lecture Notes in Computer Science, pages 97–117. Springer, Heidelberg, August 2005.
[20] Dubrova, E. Hell, M. Espresso: A stream cipher for 5g wireless communication systems. Cryptography and Communications, 9(2):273–289, 2017.
[21] Wang, M. Lin, D. Related Key chosen IV Attack on Stream Cipher Espresso Variant. IEEE International Conference on Computational Science and Engineering (CSE) 2017.
[22] Verdult, R. Garcia, F. Balasch, J. Gone in 360 seconds: Hijacking with hitag2. In Proceedings of the 21st USENIX Conference on Security Symposium, Security'12, pages 37–37, Berkeley, CA, USA, 2012. USENIX Association.
[23] Verdult, R. Garcia, F. Ege, B. Dismantling Megamos crypto: Wirelessly lockpicking a vehicle immobilizer. In Supplement to the 22nd USENIX Security Symposium (USENIX Security 13), pages 703–718. USENIX Association, August 2013.
[24] Garcia, F. de Koning Gans, G. Verdult, R. Wirelessly lockpicking a smart card reader. International Journal of Information Security, 13(5):403–420, 2014.
[25] Biryukov, A. Leurent, G. Roy, A. Cryptanalysis of the "kindle" cipher. In Knudsen and Wu [KW13], pages 86–103.
[26] David Wagner, Leone Simpson, Ed Dawson, John Kelsey, William Millan, and Bruce Schneier. Cryptanalysis of ORYX. In Stafford E. Tavares and Henk Meijer, editors, SAC

1998: 5th Annual International Workshop on Selected Areas in Cryptography, volume 1556 of Lecture Notes in Computer Science, pages 296–305. Springer, Heidelberg, August 1999.

[27] Driscoll. BeepBeep Embedded Real-Time Encryption. International Workshop on Fast Software Encryption. In FSE2002:Fast Software Encryption. pp.164-178.

[28] Wagner, D. Schneier, B. Kelsey, J. Cryptanalysis of the cellular encryption algorithm. In Burton S. Kaliski Jr., editor, Advancesin Cryptology – CRYPTO'97, volume 1294 of Lecture Notes in Computer Science, pages 526–537. Springer, Heidelberg, August 1997.

[29] Perrin, L. More reverse-engineered S-boxes. Presentation at the rump session of ESC'2017. Slides available at https://www.cryptolux.org/mediawiki-esc2017/images /2/2e/Rump.pdf, 2017.

[30] Borghoff, J. Knudsen, L. Leander, G. Matusiewicz, K. Cryptanalysis of C2. In Halevi [Hal09], pages 250–266.

[31] Bono, S. Green, M. Stubblefield, A. Juels, A. Rubin, A. Szydlo, M. Security analysis of a cryptographically-enabled RFID device. In Proceedings of the 14th Conference on USENIX Security Symposium - Volume 14, SSYM'05, pages 1–1, USA, 2005.

[32] Indesteege, S. Keller, N. Dunkelman, O. Biham, E. Preneel, B. A practical attack on KeeLoq. In Nigel P. Smart, editor, Advances in Cryptology – EUROCRYPT 2008, volume 4965 of Lecture Notes in Computer Science, pages 1–18. Springer, Heidelberg, April 2008.