TESIS DE LA UNIVERSIDAD DE ZARAGOZA

2020 143

Adrián Pérez Resa

# Encriptación sobre Capa Física para Ethernet Óptico de Alta Velocidad

Departamento
Ingeniería Electrónica y Comunicaciones

Director/es
Celma Pueyo, Santiago
Sánchez Azqueta, Carlos



ISSN 2254-7606

Prensas de la Universidad
Universidad Zaragoza

ISSN 2254-7606

Universidad Zaragoza
1542

Tesis Doctoral

# ENCRIPTACIÓN SOBRE CAPA FÍSICA PARA ETHERNET ÓPTICO DE ALTA VELOCIDAD


Autor

Adrián Pérez Resa

Director/es

Celma Pueyo, Santiago
Sánchez Azqueta, Carlos

**UNIVERSIDAD DE ZARAGOZA**

Ingeniería Electrónica y Comunicaciones


2020

Hoy en día, los estándares Ethernet ópticos de alta velocidad están ampliamente extendidos. Diferentes técnicas de encriptación pueden ser aplicadas para lograr la confidencialidad de los datos en los diferentes niveles de la comunicación. Cuando la encriptación se lleva a cabo en la capa física algunos de los beneficios que se pueden conseguir son la no pérdida de rendimiento en la transmisión de datos y el enmascaramiento de las comunicaciones.
Encriptación sobre Capa Física para Ethernet Optico de Alta Velocidad explora diferentes arquitecturas de encriptación por *streaming* en la capa PCS (Physical Coding Sublayer) de dos de los principales estándares Ethernet sobre fibra óptica: 1000Base-X y 10GBase-R.

Universidad Zaragoza
1542

GDE Grupo de Diseño Electrónico

ENCRIPTACIÓN SOBRE CAPA FISICA PARA ETHERNET OPTICO DE ALTA VELOCIDAD

ENCRIPTACIÓN SOBRE CAPA FISICA PARA ETHERNET OPTICO DE ALTA VELOCIDAD

# ENCRIPTACION SOBRE CAPA FISICA PARA ETHERNET OPTICO DE ALTA VELOCIDAD

Adrián Pérez Resa



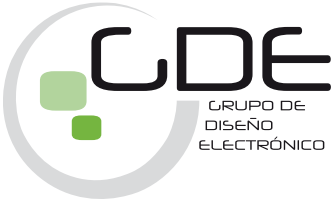
Universidad de Zaragoza
Grupo de Diseño Electrónico – i3A

# ENCRIPTACIÓN SOBRE CAPA FÍSICA PARA ETHERNET ÓPTICO DE ALTA VELOCIDAD

**Adrián Pérez Resa**

Grupo de Diseño Electrónico – I3A
Departamento de Ingeniería Electrónica y Comunicaciones
Facultad de Ciencias
Universidad de Zaragoza


Universidad Zaragoza



# ENCRIPTACIÓN SOBRE CAPA FÍSICA PARA ETHERNET ÓPTICO DE ALTA VELOCIDAD

Tesis presentada a la Universidad de Zaragoza
para optar al grado de Doctor en el Programa de Tecnologías de la Información y Comunicaciones en Redes Móviles


por

**Adrián Pérez Resa**

bajo la supervisión de los doctores

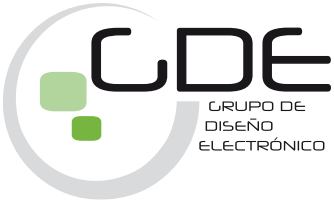
**Dr. Santiago Celma Pueyo**

**Dr. Carlos Sánchez Azqueta**

Grupo de Diseño Electrónico – I3A
Departamento de Ingeniería Electrónica y Comunicaciones
Facultad de Ciencias - Universidad de Zaragoza



Universidad Zaragoza




Zaragoza, febrero de 2020

# Informe del Director de Tesis

Dr. Santiago Celma Pueyo, catedrático del departamento de Ingeniería Electrónica y Comunicaciones de la Universidad de Zaragoza y responsable del Grupo de Diseño Electrónico de la facultad de ciencias.

Y

Dr. Carlos Sánchez Azqueta, profesor ayudante doctor del departamento de física aplicada de la Universidad de Zaragoza y miembro investigador del Grupo de Diseño Electrónico de la facultad de ciencias.

**CERTIFICAN**

Que la presente memoria de Tesis Doctoral titulada:

*Encriptación sobre Capa Física para Ethernet Óptico de Alta Velocidad*

Ha sido realizada por Don Adrián Pérez Resa bajo nuestra dirección en el Grupo de Diseño Electrónico del departamento de Ingeniería Electrónica y Comunicaciones dentro del programa de Doctorado de Tecnologías de la Información y Comunicaciones en Redes Móviles y **AUTORIZAN** su presentación como compendio de publicaciones.

Zaragoza, a 19 de febrero de 2020

Fdo: Dr. D. Santiago Celma Pueyo Fdo: Dr. D. Carlos Sánchez Azqueta

# Publicaciones

La presente tesis doctoral está realizada por un compendio de trabajos de investigación que han sido previamente publicados en revistas científicas. A continuación se listan las referencias bibliográficas que conforman la tesis.


[PER19a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems". *IEEE Transactions on Instrumentation and Measurement*, 68(12):4876–4886, Dec 2019.

[PER19b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Physical Layer Encryption for Industrial Ethernet in Gigabit Optical Links". *IEEE Transactions on Industrial Electronics*, 66(4):3287–3295, April 2019.

[PER19c] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaotic Encryption for 10-Gb Ethernet Optical Links". *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(2):859–868, Feb. 2019.

[PER19d] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-Synchronized Encryption for Physical Layer in 10Gbps Optical Links". *IEEE Transactions on Computers*, 68(6):899–911, June 2019.

[PER19e] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-Synchronized Encryption Using an FPE Block Cipher for Gigabit Ethernet". In *2019 15th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, pages 81–84, Lausanne, Switzerland, July 2019.

[PER20a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "A New Method for Format Preserving Encryption in High-Data Rate Communications". IEEE Access, 8:21003–21016, 2020.

[PER20b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-synchronized Encryption for Physical Layer in 1Gbps Ethernet Optical Links". *IEEE Access*, Pending Acceptance.

*A mi familia*

# Agradecimientos

A mis directores, Dr. Santiago Celma y Dr. Carlos Sánchez-Azqueta, por su apoyo y consejos a lo largo de este periodo y por haberme dado la oportunidad de llevar a cabo esta tesis doctoral, que representa el máximo grado académico al que se puede aspirar.

También, muy en especial, a Miguel y a Guillermo, mis queridos compañeros, gracias a los cuales este tiempo se ha convertido en algo mucho más llevadero y divertido. Les deseo todo lo mejor en sus futuros profesionales, que seguro les irá bien.

Al resto de mis compañeros, Antonio, Alejandro, Jorge y Guillermo2, que también han llenado la vida del despacho y a los cuales espero verlos pronto defendiendo su tesis. Por supuesto también al resto de las personas que forman parte del GDE, Pedro, Nicolás, Belén, Concha, Oscar y Pepe, por su compañía y por haber creado un ambiente agradable en el trabajo.

A mis padres, por la educación que he recibido de ellos y por su apoyo incondicional, y también a mi hermana, mi cuñado y mis sobrinos, por su cariño y aprecio durante todo este tiempo.

Por último, agradezco también a todas las instituciones que han financiado la investigación recogida en este trabajo: Ministerio de Economía y Competitividad del Gobierno de España, Fondo Europeo de Desarrollo Regional, Gobierno de Aragón y Horizon 2020 a través de los proyectos (TEC2014-52840-R, RTC-2015-3807-7, A38790, T26-17R y TEC2017-85867-R).

# Contenido

# Listado de Figuras

# Listado de Tablas

# Listado de Acrónimos

| | |
|---|---|
| **AES** | Advanced Encryption Standard |
| **ASIC** | Application-Specific Integrated Circuit |
| **CBC** | Cipher Block Chaining |
| **CDR** | Clock and Data Recovery |
| **CEN** | Carrier Ethernet Networks |
| **CFB** | Cipher Feedback |
| **CMAC** | Cipher-based Message Authentication Code |
| **CRC** | Cyclic Redundancy Check |
| **CTR** | Modo Contador |
| **DES** | Data Encryption Standard |
| **EDM** | End Decryption Message |
| **EEM** | End Encryption Message |
| **ENISA** | European Union Agency for Network and Information Security |
| **FPE** | Format Preserving Encryption |
| **FPGA** | Field Programmable Gate Array |
| **HDL** | Hardware Description Languaje |
| **ICM** | Integrity Check Message |
| **IDS** | Intrusion Detection Systems |
| **IFG** | Inter Frame Gap |
| **IND-CPA** | Indistinguishability under Chosen Plaintext Attack |
| **IV** | Initialization Vector |
| **KDF** | Key Derivation Function |
| **LFSR** | Linear FeedBack Shift Register |
| **MAC** | Medium Access Control / Message Code Authentication |

| | |
|---|---|
| **NIST** | National Institute of Standards and Technology |
| **OCDM** | Optical Code Division Multiplexing |
| **OCFB** | Optimized Cipher Feedback |
| **OFB** | Output Feedback |
| **OSI** | Open System Interconnection |
| **OTN** | Optical Transport Network |
| **OTP** | One-Time Pad |
| **PCS** | Physical Coding Sublayer |
| **PMA** | Physical Medium Attachment |
| **PMD** | Physical Medium Dependent |
| **PRF** | Pseudo Random Function |
| **PRNG** | Pseudo Random Number Generator |
| **PRP** | Pseudo Random Permutation |
| **PSCFB** | Pipelined Statistical Cipher Feedback |
| **QKD** | Quantum Key Distribution |
| **SCADA** | Supervisory Control And Data Acquisition |
| **SCFB** | Statistical Cipher Feedback |
| **SCM** | State Check Message |
| **SCOC** | Secure Communications using Optical Chaos |
| **SDM** | Start Decryption Message |
| **SEM** | Start Encryption Message |
| **SERDES** | Serializer/Deserializer |
| **SFP** | Small Form Factor Pluggable |
| **TCP/IP** | Transmission Control Protocol/Internet Protocol |

# 1

# Introducción y Conceptos Básicos



## 1.1 Introducción

Hoy en día, los enlaces ópticos con tasas de transmisión de hasta 100 Gbps y superiores son ya una realidad. Gracias a los avances logrados en las comunicaciones ópticas durante las últimas décadas es posible afrontar anchos de banda cada vez mayores, lo que satisface las demandas de las aplicaciones más exigentes [CIS16], como por ejemplo las basadas en *cloud computing* o *big data*. Por otro lado, la seguridad en la información sigue siendo un asunto de gran importancia en las comunicaciones ya que el volumen de amenazas en la red se ha incrementado durante los últimos años [CIS18]. Los fallos en la seguridad podrían llevar al mal funcionamiento de un servicio o la pérdida de confidencialidad en datos críticos de los clientes.

En un sistema de comunicaciones por capas, como por ejemplo en el modelo OSI (Open System Interconnection) o TCP/IP (Transmission Control

Protocol/Internet Protocol), se pueden llevar a cabo tanto ataques pasivos como activos en los diferentes niveles de la comunicación. Dependiendo de las capas de comunicación utilizadas, distintos mecanismos pueden ser adoptados para lograr la seguridad de la información. Por ejemplo, protocolos estandarizados tales como MACsec [IEE06] o IPsec [KEN05] son empleados normalmente en la capa 2 (capa de enlace de datos) y capa 3 (capa de red), respectivamente. En ambos casos la encriptación es llevada a cabo en cada trama o paquete de datos de forma individual.

Para el caso particular de las redes ópticas, el análisis de las amenazas en su capa 1 (capa física) también es considerado crítico para garantizar unas comunicaciones seguras [SKO16], [FUR14]. En este caso se pueden destacar tres tipos de ataques: ataques de inserción de señal, ataques por *splitting* y ataques a las infraestructuras físicas. Los ataques por *splitting* son normalmente empleados para espionaje pasivo o para producir degradación en la señal [SKO16], estos se pueden llevar a cabo fácilmente gracias a técnicas de derivación en la fibra. De hecho hoy en día ya existen métodos de bajo coste para interceptar la señal óptica gracias a dispositivos de acoplamiento óptico y conversores electroópticos sin la necesidad de interferir perceptiblemente en las comunicaciones [ZAF11]. Para evitar o detectar escuchas ilegales, tanto la encriptación como los IDS (Intrusion Detection Systems) son considerados como posibles soluciones.

Con el fin de tratar estas amenazas y proteger la confidencialidad de los datos en la capa física, varios mecanismos relacionados con tecnologías fotónicas han sido propuestos [FOK11], por ejemplo OCDM (Optical Code Division Multiplexing) [JI17], SCOC (Secure Communications using Optical Chaos) [HIZ10] o QKD (Quantum Key Distribution) [ELK13]. Otras técnicas, también relacionadas con protocolos de capa física, cifran la información a nivel de bit independientemente de la tecnología fotónica empleada, como la encriptación de los datos del *payload* en las tramas OTN (Optical Transport Network) [GUA16].

Algunas de las ventajas reivindicadas por estas técnicas de encriptación consisten en cifrar la información "*al vuelo*" introduciendo un *overhead* nulo en los datos y una latencia muy baja (en el rango de nanosegundos) en la

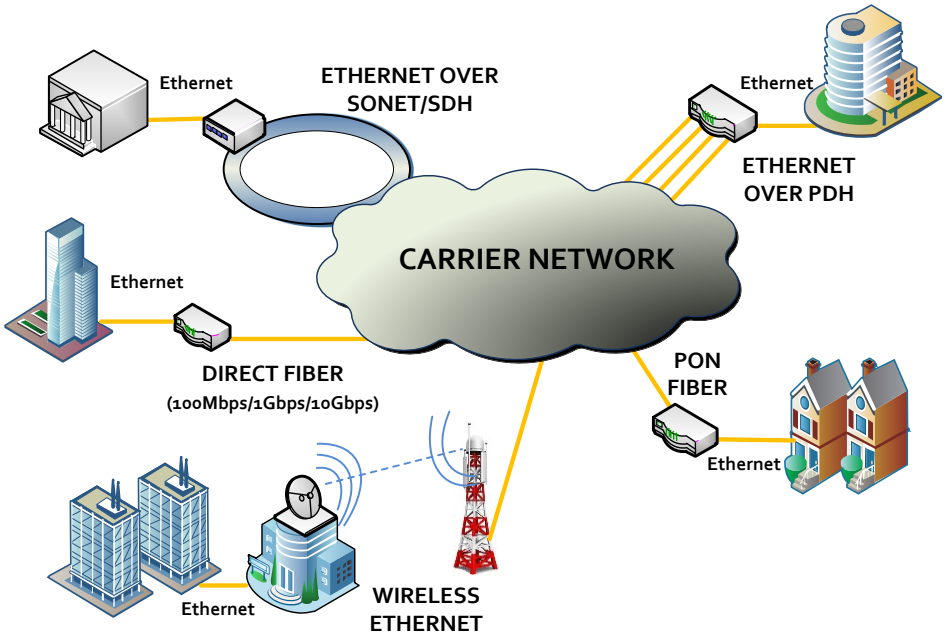



Fig.1-1. Ejemplo de diferentes tecnologías de acceso a una CEN (Carrier Ethernet Network).

información transmitida [GUA16]. De hecho, hoy en día ya están disponibles en el mercado equipos de comunicaciones OTN que realizan el cifrado a la velocidad de línea sin mermar el *throughput,* es decir consiguiendo un rendimiento de la transmisión del 100% [MIC16]. Esto contrasta con lo que hacen ciertos protocolos en otras capas de comunicación [KOL13], [XEN06]. Por ejemplo IPsec generalmente introduce latencias en el rango de milisegundos. Además, el *overhead* introducido por IPsec durante el cifrado limita el rendimiento de transmisión a valores entre el 20% y el 90% de la máxima tasa de datos posible sin encriptación [TRO05], [KOL13].

Aparte de lograr la confidencialidad, alguno de los métodos mencionados anteriormente también es capaz de conseguir privacidad contra intrusos pasivos [FOK11], entendiendo esta como la amenaza cuando dichos intrusos pueden detectar simplemente la presencia de comunicaciones aunque sean incapaces de descifrar el contenido de la información de las mismas. Esta habilidad puede ofrecer seguridad contra ataques basados en el análisis de los patrones del tráfico, que permitirían revelar información del comportamiento de una compañía o instalación.

Dentro de los estándares de comunicaciones ópticas, Ethernet es uno de los más empleados hoy día. Un claro ejemplo es el acceso a las redes de transporte

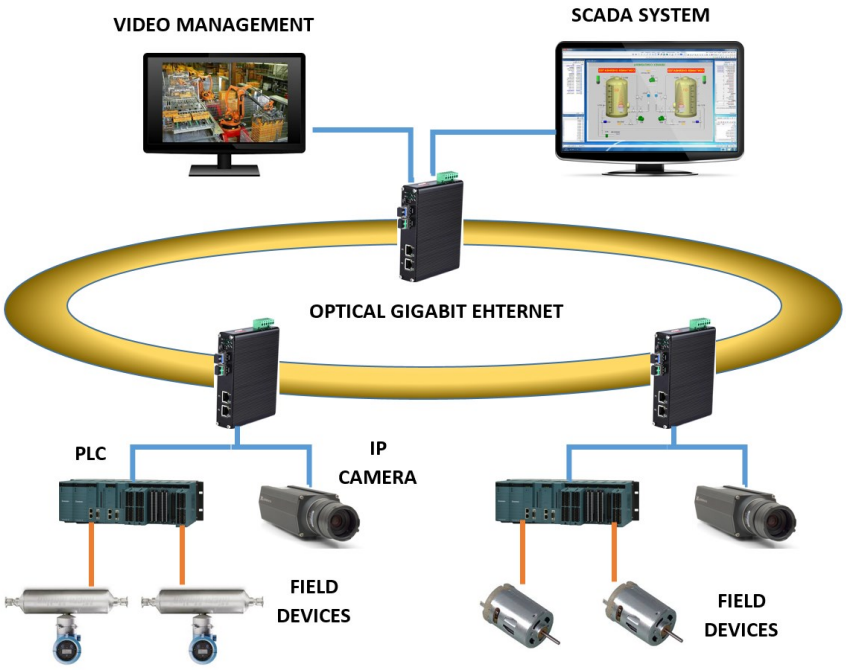



Fig.1-2. Ejemplo de una red SCADA compartiendo el tráfico de los dispositivos de campo y video vigilancia.

ópticas donde este estándar es utilizado normalmente cuando las tasas de acceso superan el gigabit por segundo. Tal y como se muestra en la Fig.1-1, algunas tecnologías de acceso en los tramos de última milla de las CEN (Carrier Ethernet Networks) son Ethernet sobre fibra (Fibra Directa con Ethernet, Ethernet sobre SONET/SDH, Ethernet sobre PON), Ethernet sobre PDH o Ethernet inalámbrico [MET09].

Otros ejemplos de aplicaciones donde Ethernet óptico se ha implantado ampliamente son las redes industriales. En las últimas décadas, las soluciones de comunicación a nivel de automatización en los ICS (Industrial Control Systems) han evolucionado desde los tradicionales buses de campo a la tecnología Ethernet [SAU10], [LEE06], [SAU11]. Por un lado, Ethernet ha incrementado su uso como solución de comunicación para redes SCADA (Supervisory Control And Data Acquisition). Por otro, en este tipo de redes no solo se transporta la información de equipos industriales, sino que también se mezcla con los datos de redes de seguridad con video vigilancia y de futuras aplicaciones de IoT, tal y como se muestra en la Fig.1-2. Ambas aplicaciones demandan un elevado ancho de banda, lo que hace que muchos proveedores

dispongan de equipos de telecomunicación adaptados a entornos industriales con velocidades de hasta 1 Gbps y superiores.

Dos de los estándares ópticos Ethernet más empleados hoy en día son los denominados 1000Base-X y 10GBase-R con tasas de transmisión de 1 Gbps y 10 Gbps, respectivamente.

## 1.2 Objetivos de la Tesis

En el caso de las comunicaciones sobre Ethernet óptico no existe ningún mecanismo que logre la mencionada privacidad al mismo tiempo que la confidencialidad, sin que además introduzca un *overhead* o latencias indeseadas.

El objetivo de esta tesis es el de proporcionar soluciones a dos de los estándares ópticos Ethernet más empleados, tales como 1000Base-X o 10GBase-R, logrando las características citadas anteriormente.

En general los principales aspectos que se pretenden desarrollar en esta tesis son los siguientes:

- Realizar propuestas viables de modificación de ambos estándares, 1000Base-X y 10GBase-R, de forma que se pueda llevar a cabo la encriptación en la capa física.
- Lograr la compatibilidad de las nuevas arquitecturas de encriptación con dichos estándares de forma que el hardware electrónico más dependiente del medio de transmisión, como los módulos ópticos SFP, los SERDES o los circuitos de recuperación de reloj y datos, no necesite modificaciones adicionales.
- Realizar un estudio de los posibles esquemas de encriptación por *streaming* que sean capaces de cifrar datos a velocidades superiores a 1 Gbps y adaptarlos a las arquitecturas propuestas.
- Estudiar posibles mecanismos para llevar a cabo la sincronización de los módulos de encriptación entre dos terminales remotos.

- Lograr que las soluciones propuestas lleven a cabo la encriptación introduciendo la menor latencia posible, al menos en un orden de magnitud igual o inferior al de soluciones en otros estándares de comunicaciones como OTN.
- Llevar a cabo un análisis de la seguridad de las soluciones propuestas, incluyendo el estudio de la capacidad de privacidad en las comunicaciones.
- Proponer un esquema de chequeo de integridad, autenticación y refresco de claves a nivel de capa física.
- Llevar a cabo la implementación y verificación física de las soluciones propuestas.

## 1.3 Metodología y Estructura de la Tesis

Durante el desarrollo de la tesis, se ha abordado el estudio de diferentes soluciones de encriptación para flujos de datos de alta velocidad, consistentes en cifradores por *streaming*, prestando atención tanto a las soluciones sincronizadas como autosincronizadas. Al mismo tiempo se ha llevado a cabo el estudio de mecanismos capaces de logar el cifrado preservando el formato de la codificación de los datos encriptados, de forma que se pueda lograr la compatibilidad necesaria con los estándares Ethernet mencionados. Una vez llevados a cabo estos estudios se han propuesto cambios en las arquitecturas de las capas PCS de ambos estándares, 1000Base-X y 10GBase-R. Concretamente se ha indicado tanto la forma de llevar a cabo la operación de cifrado entre el texto plano y el *keystream* como su localización en el camino de datos de transmisión y recepción. Finalmente se han diseñado y analizado diferentes soluciones para la generación de las secuencias de *keystream* atendiendo a la seguridad de las mismas, mediante el estudio de su ventaja de tipo IND-CPA.

Las diferentes soluciones en ambos estándares han sido diseñadas y simuladas en lenguajes HDL (Hardware Description Languaje) haciendo uso del simulador de lógica digital Modelsim-SE de Mentor. Algunas de ellas, como las basadas en

algoritmos FPE también han sido inicialmente simuladas mediante entorno Matlab de Mathworks.

La implementación hardware se ha realizado sobre plataformas FPGA del fabricante Xilinx, concretamente utilizando el dispositivo Virtex-7. La herramienta software Vivado, también de dicho fabricante, ha sido la empleada tanto en la síntesis como la implementación de las diferentes soluciones. El testeo del sistema se ha llevado a cabo sobre la placa de evaluación VC707 utilizando un enlace de fibra óptica multimodo y módulos ópticos SFP+ comerciales, capaces de transmitir a tasas de hasta 10 Gbps.

Esta memoria se divide en tres capítulos. El primer capítulo se dedica a introducir conceptos básicos de la capa física Ethernet y criptografía en general. En el segundo se presentan los trabajos desarrollados en esta tesis, donde se tratan diversas soluciones de encriptación aportadas para los estándares 1000Base-X y 10GBase-R, incluyendo su implementación y una discusión sobre la seguridad de las mismas. En el tercero se resumen las conclusiones y aportaciones de este trabajo. Posteriormente, se adjunta un apartado donde se incluyen las referencias bibliográficas citadas a lo largo de la memoria. Finalmente, se añaden tres apéndices que contienen una copia de las publicaciones que conforman la tesis, la información relativa a los factores de impacto y áreas temáticas de las revistas donde han sido publicadas y un listado de las publicaciones del autor de la tesis.

## 1.4 Conceptos básicos – Capas físicas Ethernet

### 1.4.1 Introducción capas Ethernet

El estándar Ethernet, al igual que otros sistemas de comunicaciones se encuentra dividido en diferentes capas tal y como se especifica en el modelo OSI. Particularmente Ethernet se corresponde con las dos primeras capas de este modelo, la capa física o capa 1 y la capa de enlace de datos o capa 2. A estas capas también se les suele denominar PHY en el caso de la capa 1 y MAC (Medium Access Control) en el de la capa 2, tal y como se muestra en la Fig.1-3.

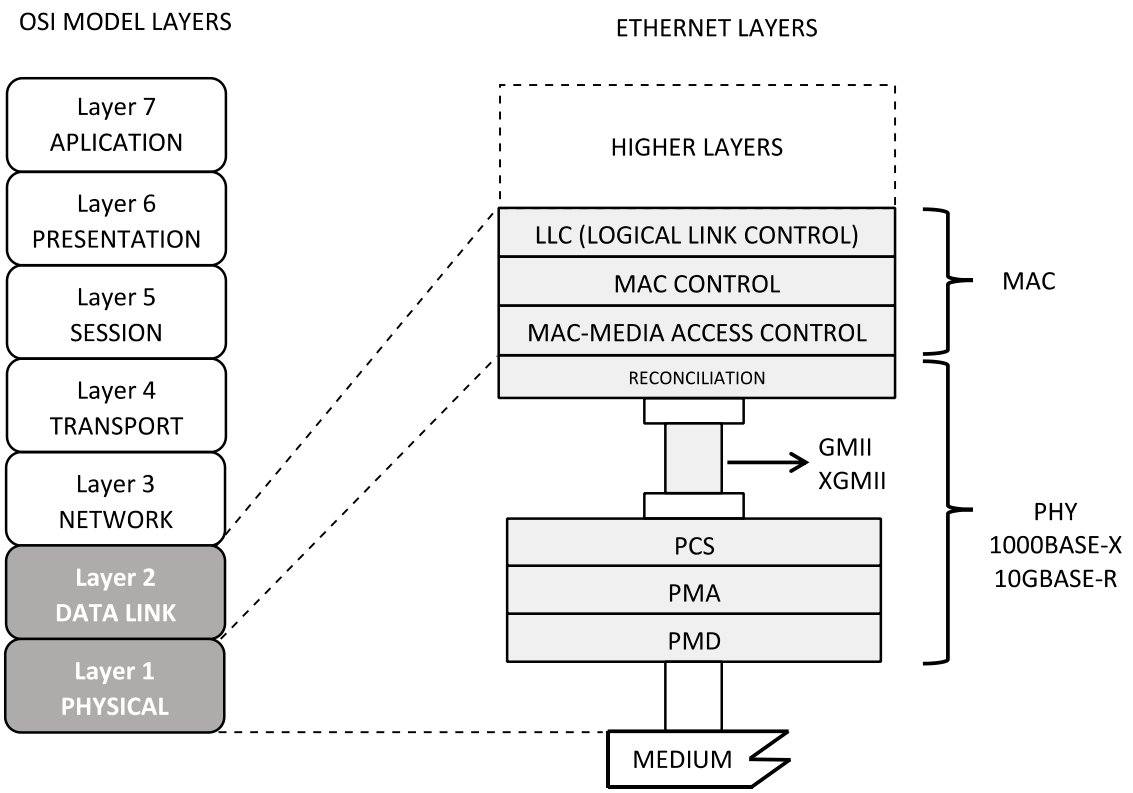



Fig.1-3. Capas de comunicación en Ethernet.

De las dos capas, la capa física es la encargada de llevar a cabo las funciones de más bajo nivel en la transmisión. Según el modelo OSI, esta define las especificaciones eléctricas, mecánicas y funcionales para activar, mantener y desactivar el enlace físico. En el caso de los estándares Ethernet se suele dividir en otras tres subcapas con diferentes funcionalidades: PMD (Physical Medium Dependent), PMA (Physical Medium Attachment) y PCS (Physical Coding Sublayer). El nivel PMD desempeña principalmente las funciones más dependientes del medio físico, por ejemplo en el caso de una transmisión óptica lo identificaríamos con componentes como el láser o fotodiodo de un transceptor óptico. El nivel PMA se encarga de la adaptación y conexión entre la capa de codificación PCS y la dependiente del medio PMD, e incluye componentes o bloques electrónicos tales como los SERDES (Serializer/Deserializer) o circuitos CDR (Clock and Data Recovery) para recuperación de reloj y datos. Finalmente, el nivel PCS es el encargado de llevar a cabo la codificación de los datos, el *scrambling*, la adaptación de tasas y relojes entre terminales y otro tipo de funcionalidades dependientes del protocolo de capa física, como la autonegociación y el establecimiento del enlace [IEE16].

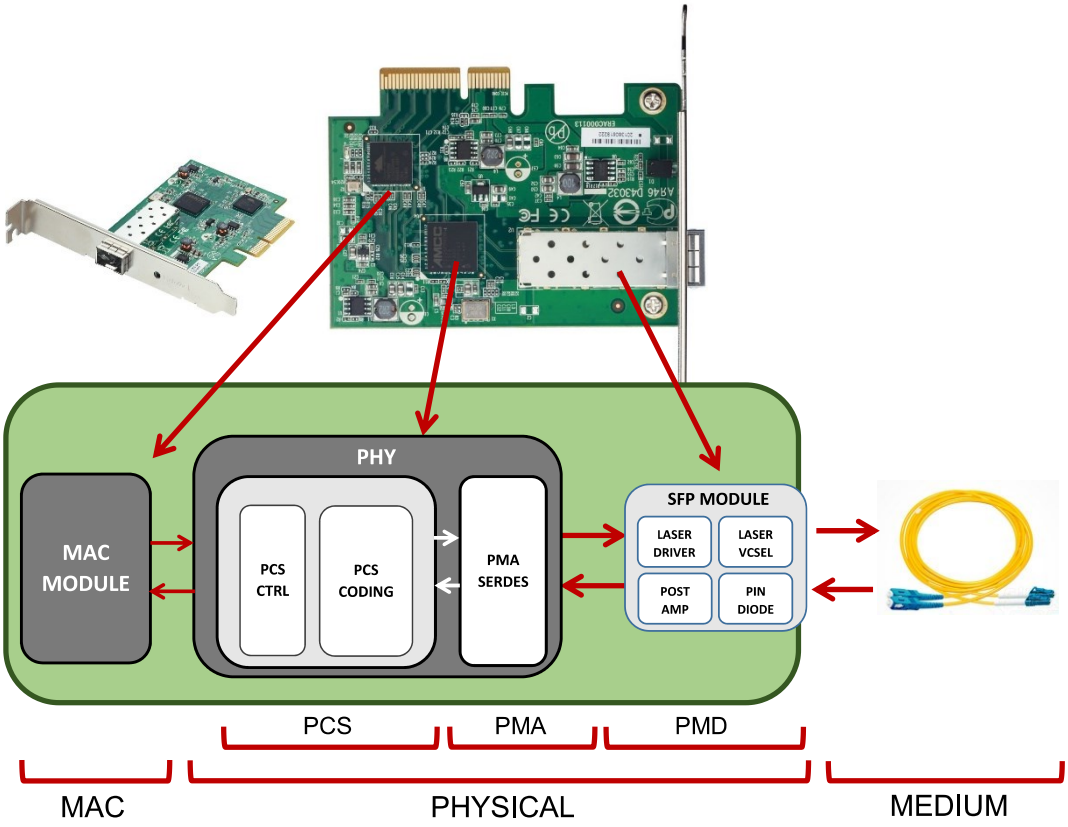



Fig.1-4. Tarjeta controladora Ethernet 10 Gbps modelo D-Link DXE-810S basada en los ASICs QT2025 de AMCC y TN4010 de Tehuti Networks.

A modo de ejemplo, en la Fig.1-4 se muestra una tarjeta comercial Ethernet en la que cada bloque/capa de comunicación se encuentra implementado por diferentes elementos electrónicos. En dicha imagen se pueden distinguir tres de los elementos principales de la tarjeta electrónica, un módulo óptico o módulo SFP (Small Form-Factor Pluggable) y dos ASICs (Application-Specific Integrated Circuit) de los fabricantes AMCC y Tehuti Networks.

En el primero de los ASICs, de Tehuti Networks, se implementan las funciones de la MAC que genera los paquetes de datos Ethernet. Estos paquetes son transferidos al segundo ASIC, del fabricante AMCC, donde el bloque PCS codifica los datos y los transfiere al nivel PMA. Dentro de este nivel, el SERDES lleva a cabo la serialización de los datos codificados dando lugar a un flujo de bits serie o *bitstream* de alta velocidad. En recepción la operación es la opuesta, el SERDES se encarga de recibir el *bitstream* serie y paralelizarlo para llevar a cabo su decodificación en el PCS.

En el extremo de la cadena se encuentra el módulo SFP óptico que implementa el nivel PMD y engloba el *hardware* necesario para llevar a cabo las funciones más dependientes del medio (transmisión y recepción por fibra óptica) que en este caso son principalmente el láser, fotodiodo y su electrónica asociada.

### 1.4.2 Codificaciones del nivel PCS

En una transmisión serie de alta velocidad los datos son serializados y transmitidos sin enviar físicamente la señal de reloj con la que están sincronizados. En el receptor es necesario extraer dicho reloj a partir de las transiciones del flujo de datos recibido, de forma que a su vez estos puedan ser muestreados en el instante óptimo. El circuito encargado de realizar esta función es denominado CDR. Para facilitar la labor del CDR, es necesario que el transmisor genere suficientes transiciones en los bits enviados de forma que la información frecuencial del reloj esté contenida en el propio *bitstream*.

Por esta razón una de las funciones de la capa PCS es la de codificar los datos forma que se genere un *bitstream* que no contenga secuencias demasiado largas de un mismo valor de bit, lo que disminuiría las transiciones entre unos y ceros. Esto dificultaría la labor del CDR en el receptor pudiendo hacer que este perdiese su sincronismo.

Por otro lado, el hecho de que la diferencia entre ceros y unos esté desbalanceada introduce un *offset* en el nivel de DC del enlace serie. Si se trata de un enlace con acoplo AC esto puede producir errores en el receptor. Debido a esto, otra de las funciones de la codificación es la de minimizar la diferencia entre la cantidad de unos y ceros transmitidos (también conocida como disparidad) y así evitar este desbalanceo.

Por último, la capa PCS también ha de introducir ciertas secuencias en los bits transmitidos que permitan realizar un alineamiento de los datos una vez estos han sido paralelizados en recepción. Esto es necesario para poder delimitar los bits de inicio y fin de cada símbolo o dato codificado. En el caso del estándar 1000Base-X la secuencia de alineamiento se denomina *comma*, mientras que en 10GBase-R se utiliza la cabecera de sincronismo de los bloques de datos codificados.

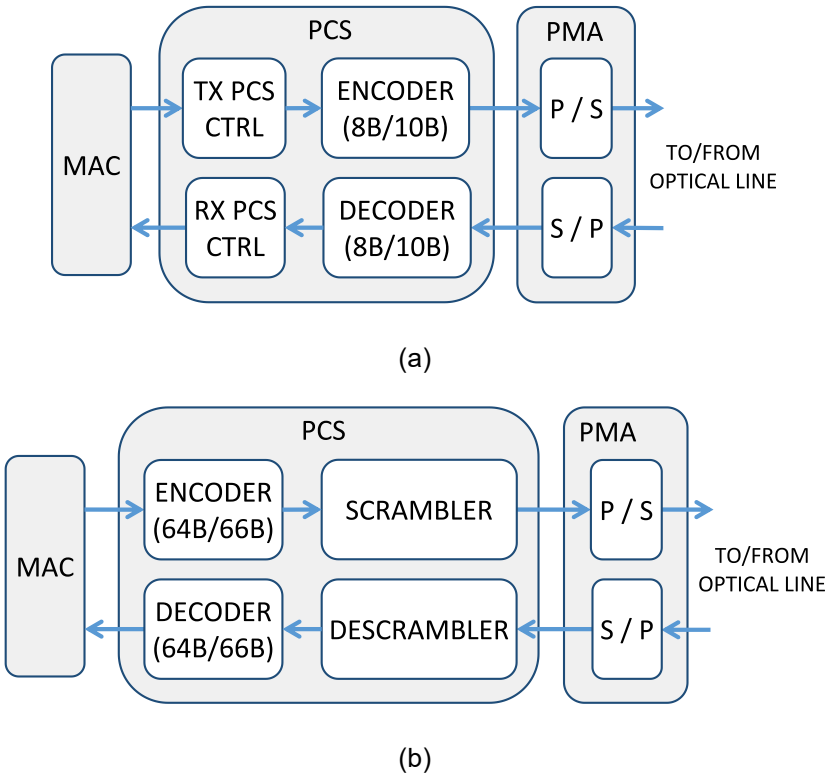



Fig.1-5. (a) Estructura PCS para el estándar 1000Base-X; (b) estructura PCS para el estándar 10GBase-R. Los bloques P/S y S/P representan al serializador y deserializador de la comunicación serie, respectivamente.

Las funciones mencionadas, limitación de la longitud de las ráfagas de bits, alta densidad en las transiciones de bit, balanceo en DC y alineamiento del *bitstream* son conseguidas gracias al empleo de codificaciones. Particularmente, en el caso del estándar 1000Base-X esto se consigue gracias a la codificación 8b/10b, mientras que en 10GBase-R se logra mediante la combinación de la codificación 64b/66b y posteriormente un proceso de *scrambling* [IEE16]. Un esquema conceptual de los esquemas PCS en ambos estándares se muestra en la Fig.1-5.

Como se verá más adelante, las soluciones de encriptación propuestas en la capa física se han llevado a cabo en el nivel PCS. Por tanto, es importante tener en cuenta que el hecho de tener que preservar las propiedades inferidas por la codificación a los datos influirá en la localización los bloques de encriptación y desencriptación. Esto ocurrirá tanto dentro de la cadena de transmisión como la de recepción del interfaz Ethernet.

## 1.5 Conceptos básicos – Criptografía

### 1.5.1 Criptografía – Introducción

Se define criptografía como el estudio y desarrollo de técnicas matemáticas relacionadas con la seguridad de la información para dotar de confidencialidad, privacidad, integridad y autenticidad a un conjunto de datos determinado.

Cada una de las funcionalidades mencionadas se explica a continuación:

- Confidencialidad: permitir el acceso a la información sólo a aquellos usuarios que estén autorizados a ello. Para ello pueden ser empleadas técnicas como la protección física de la información o el tratamiento de datos haciéndolos ininteligibles.
- Privacidad: evitar que un observador malintencionado pueda detectar la presencia de una comunicación de datos en el canal de transmisión.
- Integridad: evitar que la información sea alterada sin autorización, detectando la inserción, eliminación o sustitución de datos no deseados.
- Autenticación: permitir que los dos participantes en una comunicación se identifiquen y que dicha identidad sea verificada. También permite a los participantes acreditar que los mensajes transmitidos hayan sido generados por ellos mismos.

De las funcionalidades mencionadas, el sistema de encriptación de capa física propuesto en esta tesis se encarga principalmente de las dos primeras, es decir, no solo cifrará los datos a transmitir (confidencialidad) sino que también enmascarará la presencia de las tramas Ethernet y por tanto el patrón y la actividad del tráfico de datos (privacidad). Estas funcionalidades son implementadas en ambos estándares de comunicación, 1000Base-X y 10GBase-R, introducidos en el Apartado 1.4. En cuanto a las funcionalidades de integridad y autenticación, una solución es aportada, pero sólo para el estándar 1000Base-X.

Las herramientas criptográficas básicas que permiten abarcar estos objetivos son denominadas “primitivas criptográficas”. Los cifradores son las primitivas encargadas de lograr la confidencialidad y se clasifican en dos grandes grupos,

los de clave simétrica (privada) y los de clave asimétrica (pública). Para la encriptación de datos a alta velocidad se suelen emplear los primeros ya que requieren una carga computacional menor y su tasa de encriptación puede alcanzar mayores cotas. Su inconveniente es que tanto el transmisor como el receptor han de compartir la misma clave para las operaciones de encriptación y desencriptación, lo que requiere un intercambio previo de dicha información. Respecto a los sistemas de cifrado de clave asimétrica, estos suelen tener mayor carga computacional, alcanzando por tanto menor velocidad. Como ventaja no requieren que ambos terminales, transmisor y receptor, compartan la misma clave previamente. Por esta razón, los sistemas de encriptación de clave pública suelen ser empleados en el cifrado de pequeños volúmenes de información sensible en el que no se requiere un elevado ancho de banda.

Los sistemas de encriptación de clave simétrica se pueden definir de forma genérica de la siguiente manera [STI02]:

Un criptosistema es una quíntupla $(\boldsymbol{\mathcal{P}}, \boldsymbol{\mathcal{C}}, \boldsymbol{\mathcal{K}}, \boldsymbol{\mathcal{E}}, \boldsymbol{\mathcal{D}})$ donde se satisfacen las siguientes condiciones:

- $\boldsymbol{\mathcal{P}}$ es un set finito de posibles textos planos o *plaintext* (datos sin cifrar).
- $\boldsymbol{\mathcal{C}}$ es un set finito de posibles textos cifrados o *ciphertext* (datos cifrados).
- $\boldsymbol{\mathcal{K}}$ es el espacio de claves, un set finito de posibles claves.
- Para cada clave $k \in \boldsymbol{\mathcal{K}}$, hay una regla de encriptación $E_k \in \boldsymbol{\mathcal{E}}$ y una regla de desencriptación $D_k \in \boldsymbol{\mathcal{D}}$. Cada $E_k: \boldsymbol{\mathcal{P}} \to \boldsymbol{\mathcal{C}}$ y $D_k: \boldsymbol{\mathcal{C}} \to \boldsymbol{\mathcal{P}}$ son funciones tales que $D_k\big(E_k(P)\big) = P$ para todo elemento $P \in \boldsymbol{\mathcal{P}}$.

Inicialmente, antes de una transmisión el transmisor y receptor escogen una clave aleatoria $k \in \boldsymbol{\mathcal{K}}$, desconocida para observadores externos. Esta clave puede ser generada por el transmisor y ser enviada por un canal seguro al receptor, tal y como se muestra en la Fig.1-6. Posteriormente, el transmisor aplica la transformación $E_k$ a todos los textos planos que envíe, mientras que el receptor realiza la transformación inversa $D_k$ recuperando el mensaje original sin encriptar. Al no conocer la clave $k$, los posibles observadores pasivos en mitad del canal no podrán descifrar el mensaje.

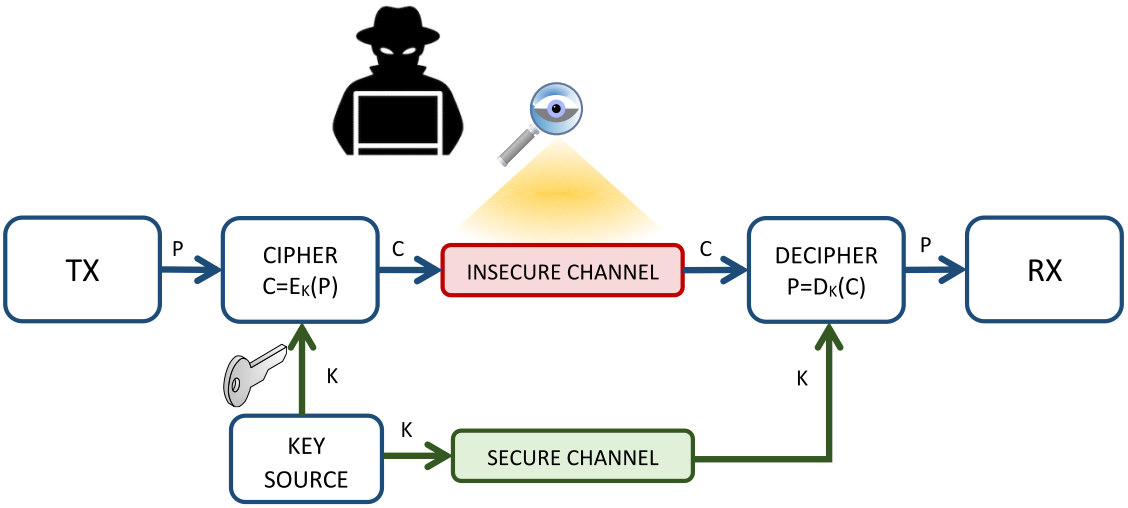



Fig.1-6. Diagrama de un sistema de encriptación por clave simétrica.

Dentro de los sistemas de clave simétrica encontramos dos subcategorías, los cifradores por *streaming* y los cifradores por bloque. Los primeros encriptan el texto plano cifrando individualmente los caracteres o palabras de bits con una transformación que varía a lo largo del tiempo, mientras que los cifradores por bloque encriptan bloques de caracteres de forma simultánea con una transformación que es fija. Los cifradores por *streaming* suelen ser más rápidos que los de bloque por lo que son más aptos para sistemas de comunicaciones donde se requiere una elevada tasa de transmisión.

### 1.5.2 Cifradores por *streaming* sincronizados

Los cifradores por *streaming* se fundamentan en un criptosistema llamado OTP (One-Time Pad) y patentado por Gilbert Vernam en 1917. Claude E. Shannon demostró matemáticamente que este criptosistema era invulnerable ya que cumplía una propiedad criptográfica que él mismo había denominado como '*secreto perfecto*' [SHA45]. Los cifradores por *streaming* surgieron como una forma práctica de implementar un sistema similar al OTP pero sacrificando esta condición de invulnerabilidad. El esquema teórico de un cifrador por *streaming* es definido en la Fig.1-7.

A diferencia de la Fig.1-6 donde el bloque KEY SOURCE genera directamente la clave $k$, que es la que opera en la encriptación, en un cifrado por *streaming*

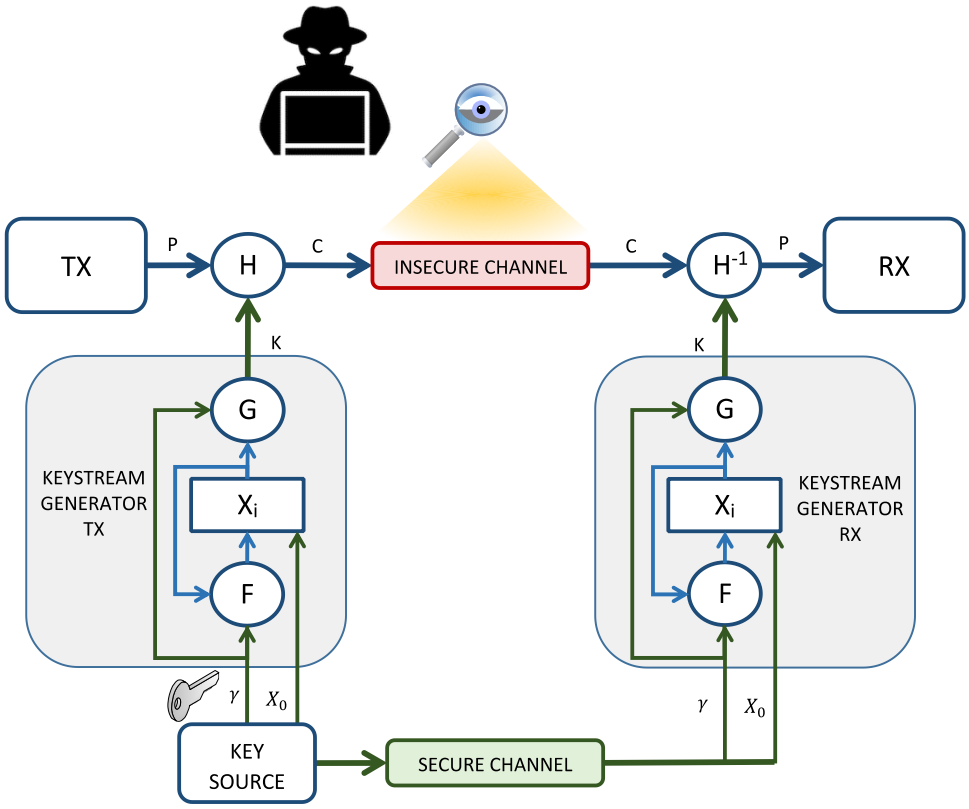



Fig.1-7. Esquema teórico de un cifrador por *streaming* sincronizado.

este bloque genera como clave una semilla o valor inicial $X_0$ y un parámetro de control $\gamma$ que configuran un módulo denominado KEYSTREAM GENERATOR. El generador de *keystream* es el encargado de generar $k$, que en este caso consiste en una secuencia o *stream* de valores obtenidos de forma pseudoaleatoria mediante un algoritmo matemático conocido. En la práctica la clave 'secreta' de este sistema ya no será $k$, como en la Fig.1-7, sino $X_0$ y $\gamma$.

Tanto en el receptor como en el transmisor existirá un generador de *keystream* y en ambos estará configurado de la misma forma. Cada generador actualiza el estado interno en función del estado actual $X_i$ y de $\gamma$ mediante una función matemática $F$. La salida del generador se obtiene a partir $X_i$ y $\gamma$ gracias a la función $G$. Posteriormente, el keystream generado opera con el texto plano a través de la función $H$ en el transmisor o la inversa de $H$ en el receptor.

Para sistemas donde el cifrado se realiza bit a bit, $H$ normalmente se suele implementar como una función XOR, tal y como funciona el mencionado OTP. Esta operación es equivalente a realizar la suma módulo-2 entre el bit a cifrar y

el bit correspondiente del *keystream*. En caso de no realizar un cifrado de valores comprendidos entre 0 y 1 (bit a bit), sino entre 0 y *N*-1, la operación $H$ se podría extender a una suma módulo-*N*.

El cifrador que acabamos de definir se trata de un cifrador por *streaming* síncrono, es decir necesita que los generadores de *keystream* en el receptor y transmisor estén sincronizados de forma que un mismo dato del texto plano opere con el mismo valor del *keystream* en transmisión y recepción. Esto implica que el transmisor y receptor han de establecer un mecanismo o protocolo de sincronización lo que incrementa la complejidad del sistema.

### 1.5.3 Cifradores por *streaming* autosincronizados

En los cifradores por *streaming* autosincronizados no es necesario sincronizar sus generadores de *keystream* por medio de protocolos u otros mecanismos, sino que de forma automática los algoritmos que los definen son capaces de lograr dicho alineamiento. Estos generadores de *keystream* suelen actualizar su estado interno en función del texto cifrado que se ha generado en instantes anteriores. El nuevo estado interno junto con parte de la clave opera en la función $G$ generando los nuevos valores del *keystream*. En recepción se lleva a cabo la misma operación, empleando el texto cifrado recibido para la actualización del estado interno. Un esquema típico de este tipo de cifradores se muestra en la Fig.1-8.

### 1.5.4 Cifradores por *streaming* basados en cifradores por bloque

Los tipos de cifradores por *streaming* explicados en los dos apartados anteriores se caracterizan por tener una estrategia de implementación ad-hoc, es decir que sin basarse en bloques preconstruidos los algoritmos de encriptación son diseñados específicamente para funcionar en este tipo de cifradores. Como ejemplo de los mismos se pueden destacar los pertenecientes a la propuesta del proyecto europeo eSTREAM [ROB08], tanto para versiones sincronizadas, por ejemplo Trivium o Grain, como para versiones autosincronizadas, SSS o Mosquito.

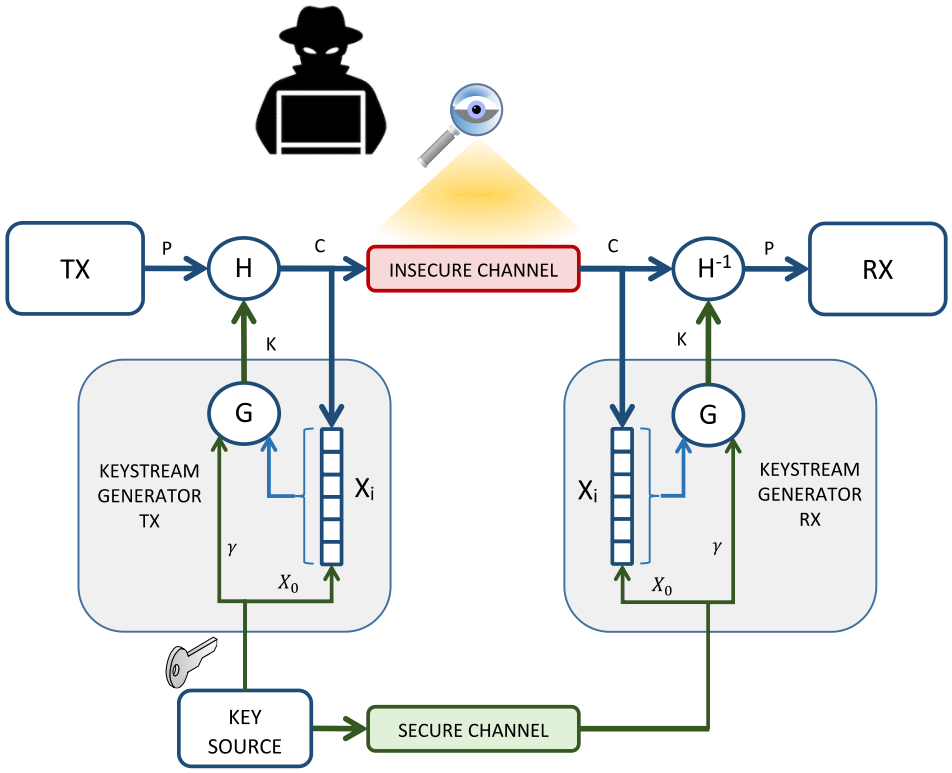



Fig.1-8. Esquema teórico de un cifrador por *streaming* autosincronizado.

Otra forma de llevar a cabo el diseño de cifradores por *streaming* es a partir de cifradores por bloque funcionando en determinados modos de operación. Es decir, en este caso no serían cifradores por *streaming* diseñados ad-hoc, sino que se basarían en bloques prediseñados de antemano, que serían cifradores por bloque. Un ejemplo típico es el cifrador AES (Advanced Encryption Standard), que funcionando en modo CTR (Contador) es capaz de generar el *keystream* necesario para la encriptación.

Los cifradores por *streaming* ad-hoc suelen ser aptos para aplicaciones de alta velocidad donde los recursos de computación empleados y su consumo en potencia son críticos. En cambio, una ventaja de los cifradores por bloque, aunque no sean tan aptos para dichas aplicaciones, es que gracias a su modularidad su análisis criptográfico suele estar más desarrollado y es más comprensible que en los de *streaming*, lo que resulta en una mayor facilidad de uso por parte de los mismos [KLE13]. De hecho, gracias a esto es posible estudiar el modo de operación de un cifrador por bloque sin necesidad de comprender al detalle su estructura interna.

| **algorithm** $E_k(P)$ | **algorithm** $D_k(\langle IV, C\rangle)$ |
|---|---|
| static $ctr \leftarrow 0$ | |
| Break $P$ into $l$ bit blocks<br>$P[1] \dots P[m]$ | Break $C$ into $l$ bit blocks<br>$C[1] \dots C[m]$ |
| $C[0] \leftarrow IV \leftarrow ctr$<br>**for** $i \leftarrow 1$ **to** $m$ **do**<br>$C[i] \leftarrow F_k(C[i-1] \oplus P[i])$<br>$C \leftarrow C[1] \dots C[m]$<br>$ctr \leftarrow ctr + 1$<br>**return** $\langle IV, C\rangle$ | $C[0] \leftarrow IV$<br>**for** $i \leftarrow 1$ **to** $m$ **do**<br>$P[i] \leftarrow F_k^{-1}(C[i] \oplus C[i-1])$<br>$P \leftarrow P[1] \dots P[m]$<br>**return** $P$ |

Fig.1-9. Algoritmos de encriptación y desencriptación para CBC.

Tal y como se ha mencionado en el Apartado 1.5.1, un cifrador por bloque encripta la información del texto plano en bloques de datos simultáneamente. Por tanto este tipo de cifradores se pueden modelar como una función $F_k$ tal que $F: \mathcal{K} \times \{0,1\}^l \rightarrow \{0,1\}^l$, donde $\mathcal{K}$ es el espacio de claves, $k$ es la clave del sistema y $l$ es el tamaño de bloque del cifrador en bits, es decir el cifrador encripta simultáneamente $l$ bits en bloque.

Según [BEL05a], un modo de operación de un cifrador por bloque se puede definir como un criptosistema de clave simétrica $S\mathcal{E}$ cuyas reglas de encriptación/desencriptación $E_k$ y $D_k$ son definidas en base a la función de encriptación $F_k$ de dicho cifrador. Estos modos se suelen emplear cuando la longitud del mensaje a transmitir es mayor que la longitud del bloque de datos de encriptación. En dicho caso, el mensaje se ha de dividir en subbloques con tamaño igual al tamaño de bloque del cifrador y emplearlo en el modo escogido para encriptar cada uno de los subbloques.

Por ejemplo dos de los modos más comúnmente empleados, CBC (Cipher Block Chaining) y CTR son definidos en [BEL05a] tal y como muestran los algoritmos de la Fig.1-9 y Fig.1-10, respectivamente.

Esquema CBC: Asumamos que $F: \mathcal{K} \times \{0,1\}^l \rightarrow \{0,1\}^l$ es un cifrador por bloque. Haciéndolo operar en modo CBC con un contador $ctr$ e IV (Initialization Vector), se obtiene un sistema de encriptación simétrico $S\mathcal{E}$ con estado. Inicialmente se escoge una clave aleatoria $k \in \mathcal{K}$ para la función $F$. La función de

| **algorithm** $E_k(P)$ | **algorithm** $D_k(\langle IV, C\rangle)$ |
|---|---|
| static $ctr \leftarrow 0$ | |
| Break $P$ into $l$ bit blocks<br>$P[0] \dots P[m-1]$ | Break $C$ into $l$ bit blocks<br>$C[0] \dots C[m-1]$ |
| $IV \leftarrow ctr$<br>**for** $i \leftarrow 0$ **to** $m-1$ **do**<br>$K[i] \leftarrow F_k(ctr)$<br>$C[i] \leftarrow K[i] \oplus P[i]$<br>$ctr \leftarrow ctr + 1$<br>$C \leftarrow C[0] \dots C[m-1]$<br>**return** $\langle IV, C\rangle$ | $ctr \leftarrow IV$<br>**for** $i \leftarrow 0$ **to** $m-1$ **do**<br>$K[i] \leftarrow F_k(ctr)$<br>$P[i] \leftarrow K[i] \oplus C[i]$<br>$ctr \leftarrow ctr + 1$<br>$P \leftarrow P[0] \dots P[m-1]$<br>**return** $P$ |

Fig.1-10. Algoritmos de encriptación y desencriptación para CTR.

encriptación $E_k$ mantiene el valor inicial del contador $ctr$ a cero. Al IV se le carga el valor actual del contador $ctr$. El contador es incrementado cada vez que un mensaje es encriptado y a su vez es una variable estática, lo que significa que su valor es preservado entre diferentes invocaciones del algoritmo, representando así el estado interno del sistema de encriptación. Los algoritmos de encriptación/desencriptación $E_k$ y $D_k$ se muestran en la Fig.1-9.

Esquema CTR: Asumamos que $F: \mathcal{K} \times \{0,1\}^l \rightarrow \{0,1\}^l$ es un cifrador por bloque. Haciéndolo operar en modo CTR con un contador $ctr$ e IV, se obtiene un sistema de encriptación simétrico $\mathcal{SE}$ con estado. Inicialmente se escoge una clave aleatoria $k \in \mathcal{K}$ para la función $F$. La función de encriptación $E_k$ mantiene el valor inicial del contador $ctr$ a cero. No se permite que el contador $ctr$ sea reiniciado en ningún momento y su valor es preservado entre diferentes invocaciones del algoritmo, representando así el estado interno de $\mathcal{SE}$. Los algoritmos de encriptación/desencriptación $E_k$ y $D_k$ se muestran en la Fig.1-10.

Dada la definición del modo CTR y tal y como se ha comentado al principio de este apartado, se podría visualizar este modo como una forma de implementación de un cifrador por *streaming* donde el texto plano estaría formado por los $m$ bloques a cifrar $P[0] \dots P[m-1]$ de tamaño $l$ bits. Para obtener los bloques texto cifrado $C[0] \dots C[m-1]$ se aplicaría la operación XOR entre el

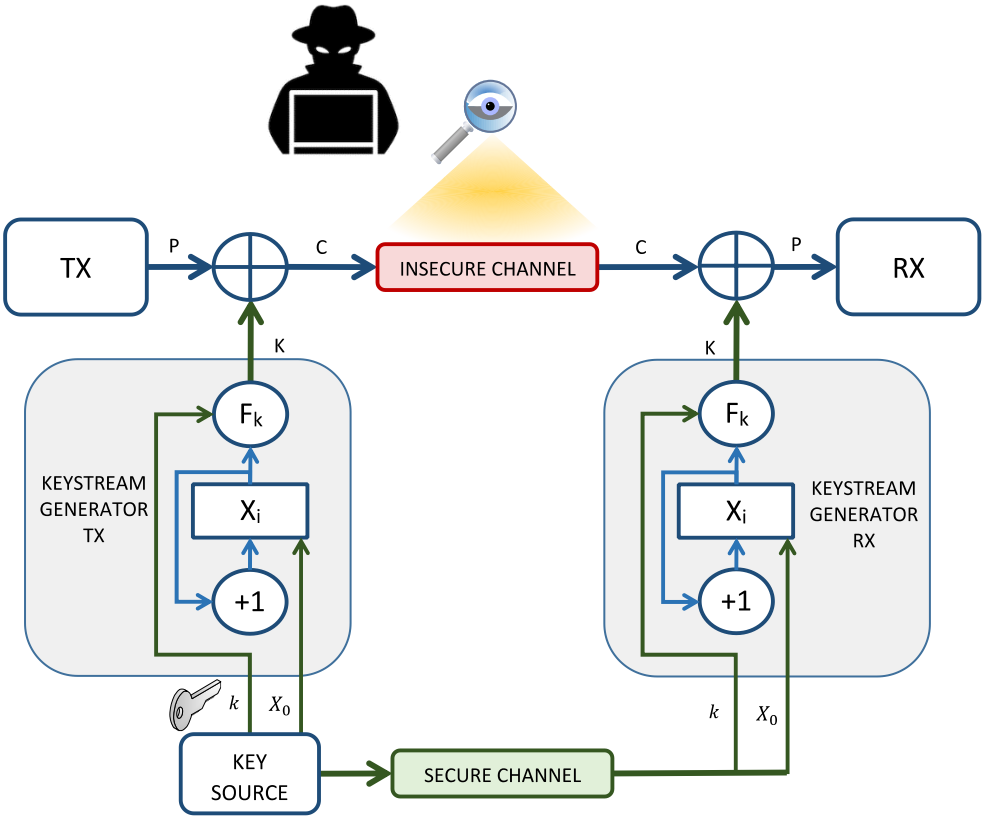



Fig.1-11. Esquema de cifrado por *streaming* basado en modo CTR.

texto plano y la secuencia de *keystream* $K[0] \dots K[m-1]$. Esta secuencia se obtendría como la salida del cifrador por bloque $F_k(\cdot)$ al encriptar los sucesivos valores del contador $ctr$. Una representación gráfica de este esquema se muestra en la Fig.1-11. En relación al esquema genérico de un cifrador por *streaming* de la Fig.1-7 vemos que se ha sustituido la función $G$ por la función del cifrador por bloque $F_k(\cdot)$, la función $F$ por un bloque de incremento '+1' y la función $H$ por la operación XOR. El estado $X_i$ se corresponde ahora con el contador $ctr$ del modo de operación CTR. Aunque en el algoritmo de la Fig.1-10 el valor de $ctr$ se inicializa a cero, en la práctica este se podrá inicializar a un valor inicial $X_0$ que deberán conocer ambos terminales, transmisor y receptor. Respecto a la clave $k$, esta se corresponderá con el parámetro $\gamma$, pero se empleará únicamente como clave de configuración del cifrador por bloque $F_k(\cdot)$.

### 1.5.5 Concepto de seguridad IND-CPA

Hoy en día se puede entender como noción de seguridad de un esquema de encriptación la imposibilidad de relacionar diferentes textos cifrados resultado de la encriptación de mensajes con igual longitud. De esta manera se intenta evitar que se produzcan fugas de información. Según [BEL05a] esto implica que la encriptación debe ser probabilística o depender del estado actual del sistema de cifrado. Si no es así cualquier mensaje que fuese reenviado sería cifrado de la misma forma dos veces, lo que permitiría al adversario su detección aun sin conocer la clave del sistema ni el contenido del mensaje.

Esta noción de seguridad va en contra de la noción popular e histórica de la encriptación, donde un mapeo fijo de textos planos es realizado sobre textos cifrados. Sin embargo la visión actual no es esta, ya que un texto plano debería tener muchos mapeos posibles en textos cifrados, por ejemplo dependiendo del estado de encriptación del algoritmo en cada momento.

Teniendo en cuenta lo explicado anteriormente, se define como criterio de seguridad la indistinguibilidad bajo un ataque de texto plano conocido o IND-CPA (Indistinguishability under Chosen Plaintext Attack). Para ello, se considera que el adversario (sin saber la clave) es capaz de encriptar dos mensajes de la misma longitud conocidos por él mismo gracias al acceso a un oráculo [BEL05a]. Supongamos un adversario A, el modelo para analizar la seguridad IND-CPA consistiría en un juego realizado entre dicho adversario y un oráculo capaz de implementar el esquema de encriptación que queremos estudiar. Dicho esquema de encriptación $S\mathcal{E}$ estaría configurado por una clave $k$ y un bit de experimento $b$. Durante el juego, el adversario A escoge una secuencia de $q$ pares de mensajes de igual longitud $(M_1^0, M_1^1), \dots, (M_q^0, M_q^1)$ y los envía al oráculo. Por cada par de mensajes $(M_i^0, M_i^1)$ el adversario recibe del oráculo el texto cifrado $C_i$ correspondiente al mensaje $M_i^b$. Finalmente este ha de adivinar qué set de mensajes $(M_1^0, \dots, M_q^0)$ o $(M_1^1, \dots, M_q^1)$ fue encriptado durante el juego, lo que implica que el adversario ha de adivinar el bit $b$ que configura el experimento. Suponiendo que el esquema de encriptación $S\mathcal{E}$ es CTR, el diagrama del juego entre adversario y oráculo se muestra en la Fig.1-12.

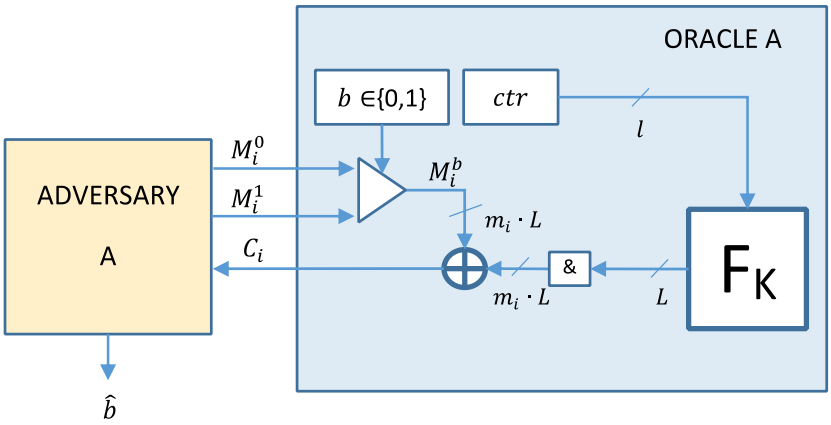



Fig.1-12. Esquema de juego entre adversario A y un oráculo que implementa el modo de encriptación CTR.

En el esquema mostrado se ha sustituido el cifrador por bloque por una función más generalista $F_k$ cuyo tamaño de bloque es diferente en la entrada y la salida, con $l$ y $L$ bits respectivamente. El bloque $ctr$ representa al contador mencionado en la Fig.1-10, en este caso con longitud $l$ bits. Cada mensaje $M_i^b$ tiene una longitud de $m_i$ bloques de $L$ bits, así que para encriptar el $M_i^b$ escogido, el oráculo ha de ejecutar $m_i$ veces la encriptación del contador, concatenar los $m_i$ bloques encriptados en el bloque "&" y finalmente realizar la XOR con el mensaje seleccionado según $b$. Al final del experimento el adversario devuelve la estimación $\hat{b}$ del bit $b$.

Como medida de la capacidad de un adversario para romper el esquema de encriptación $S\mathcal{E}$, en [BEL05b] se define la ventaja IND-CPA del adversario respecto del esquema de encriptación $S\mathcal{E}$ con la siguiente expresión:

$$ADV_{S\mathcal{E}(F)}^{IND-CPA}(A) = 2 \cdot \Pr\left(\hat{b} = b\right) - 1 \qquad (1)$$

donde $ADV_{S\mathcal{E}(F)}^{IND-CPA}(A)$ es la ventaja del adversario A sobre el esquema de encriptación y $\Pr\left(\hat{b} = b\right)$ es la probabilidad de que el adversario adivine el valor correcto de $b$. Esta ventaja puede ser entendida como el exceso de esta probabilidad sobre 1/2. Cuando la probabilidad de adivinar $b$ es casi 1/2 y la ventaja del adversario es despreciable, se puede considerar que el esquema de encriptación $S\mathcal{E}$ analizado es seguro.

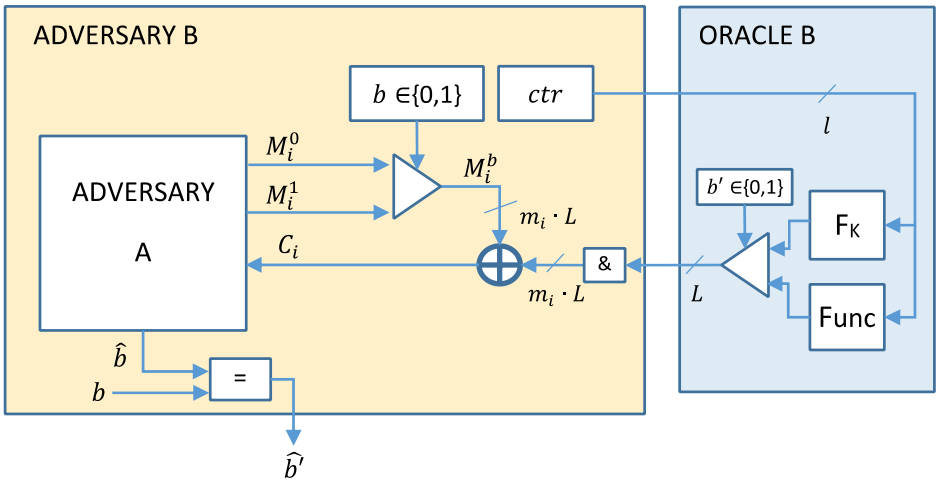



Fig.1-13. Esquema de juego entre adversario B y un oráculo que implementa la encriptación con una PRF.

Asumiendo que la función $F_k$ es una PRF (Pseudo Random Function), se puede demostrar en [BEL05c] que un adversario B atacando la seguridad de $F_k$ puede ser construido gracias al adversario A tal y como se muestra en la Fig.1-13.

En este juego el adversario B trata de adivinar si el oráculo está implementando la encriptación mediante una función PRF $F_k$ o una función aleatoria $Func$ . Esto significa que el adversario trata de obtener una estimación del bit del nuevo experimento, en este caso $b'$. Durante el experimento el adversario lanza $q'$ consultas de valores de contador que son encriptadas en el oráculo. Ya que B está construido a partir de A, $q'$ será el número total de bloques correspondiente con la longitud total de los $q$ mensajes $M_i^b$ encriptados desde el punto de vista de A. Al final del juego, el adversario B devuelve una estimación del bit de experimento $\hat{b}'$, que será la comparación entre $b$ y $\hat{b}$.

La ventaja de B sobre la función $F_k$ medirá la capacidad del adversario de distinguir dicha función pseudoaleatoria de una función completamente aleatoria. Dicha ventaja se puede relacionar con la ventaja IND-CPA de A sobre el esquema $S\mathcal{E}$ con la siguiente expresión:

$$ADV_{S\mathcal{E}(F)}^{IND-CPA}(A) = 2 \cdot ADV_F^{PRF}(B) + ADV_{S\mathcal{E}(Func)}^{IND-CPA}(A) \qquad (2)$$

donde $ADV_{SE(F)}^{IND-CPA}(A)$ es la ventaja de A al atacar $S\mathcal{E}$ cuando su función interna es la PRF $F_k$, $ADV_{SE(Func)}^{IND-CPA}(A)$ es la ventaja of A sobre $S\mathcal{E}$ cuando la función

interna es una función aleatoria $Func(l, L)$ y $ADV_F^{PRF}(B)$ es la ventaja de tipo PRF de B tal y como se define en [BEL05a] de acuerdo al experimento mostrado en la Fig.1-13.

Como generalización, hemos supuesto desde un principio que la función $F_k$ que opera en el modo de encriptación es una PRF, sin embargo los cifradores por bloque se corresponden mejor con una PRP (Pseudo Random Permutation) donde las longitudes de los bloques de entrada y salida son iguales. Las ventajas del oráculo B frente a una PRF o una PRP se pueden relacionar según el lema de conmutación PRF-PRP [BEL97]. Teniendo en cuenta esto y desarrollando la ecuación de $ADV_{S\mathcal{E}(F)}^{IND-CPA}(A)$ en cada uno de los modos de encriptación es posible llegar a una expresión que depende de la cantidad de información encriptada y de ciertos parámetros del cifrador por bloque funcionando en dicho modo.

Por ejemplo, para los modos CTR y CBC mencionados previamente las expresiones finales de su ventaja IND-CPA quedarían de la siguiente manera:

$$ADV_{CTRC(F)}^{IND-CPA}(A) \leq 2 \cdot ADV_F^{PRP}(B) + \frac{\mu^2}{L^2 2^L} \tag{3}$$

$$ADV_{CBC(F)}^{IND-CPA}(A) \leq 2 \cdot ADV_F^{PRP}(B) + \frac{2\mu^2}{L^2 2^L} \tag{4}$$

donde $\mu$ es el número total de bits encriptados en las $q$ consultas llevadas a cabo durante el juego del adversario A frente al oráculo, $L$ es el tamaño de bloque de entrada y salida del cifrador por bloque, $ADV_F^{PRP}(B)$ la ventaja de tipo PRP del adversario B frente al cifrador y $ADV_{CTRC(F)}^{IND-CPA}(A)$ y $ADV_{CBC(F)}^{IND-CPA}(A)$ son las ventajas IND-CPA de A frente a ambos esquemas de encriptación CTR y CBC, respectivamente.

Suponiendo que el cifrador por bloque es seguro y que podemos considerarlo una buena PRP con un valor de $ADV_F^{PRP}(B)$ despreciable, se puede observar que la ventaja del adversario aumenta en el juego de ambos esquemas de encriptación conforme mayor es la longitud de los datos cifrados $\mu$, es decir, la seguridad del sistema de encriptación quedaría comprometida cuanta mayor cantidad de información cifremos con la misma clave. Además se puede concluir que aproximando $ADV_F^{PRP}(B)$ a cero, ya que es despreciable, la ventaja de A

sobre el esquema CBC es el doble que sobre CTR, lo que implica que el modo CTR es más seguro que el CBC. De hecho, según [ROG11], el modo CTR puede ser considerado el mejor y más moderno método de solo encriptación.

# 2

# Encriptación sobre Capa Física



## 2.1 Localización del cifrador y estructura para 1000Base-X

Tal y como se ha comentado en el Apartado 1.4.2, la localización de los módulos de encriptación en el camino de datos se ha de hacer de tal forma que se preserven las propiedades de la codificación del estándar correspondiente. En el caso de 1000Base-X el codificador 8b/10b es el que consigue la limitación de ráfagas, la alta densidad de transiciones de bit, y el balanceo en la disparidad. Por ello el mecanismo de encriptación para soluciones de Gigabit Ethernet como en [PER19a], [PER19b] y [PER20a] se localiza entre el controlador PCS y el codificador, tal y como se muestra en la Fig.2-1.

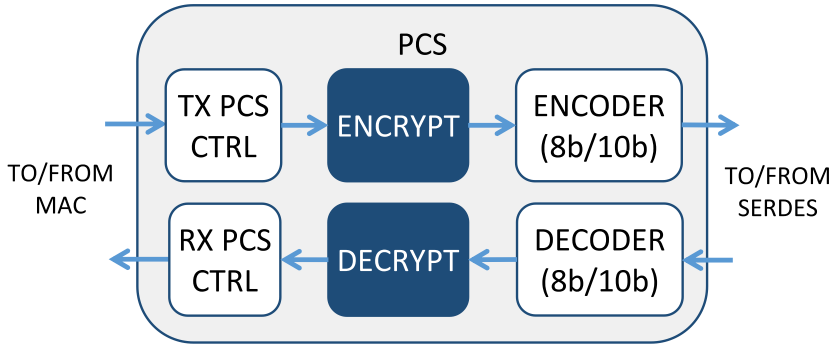



Fig.2-1. Emplazamiento de la función de encriptación en la capa PCS para 1000Base-X.

Al preservar las propiedades que transfiere el codificador 8b/10b a los datos, se puede conseguir que el mecanismo de encriptación mantenga la compatibilidad con el resto de elementos *hardware* más dependientes del medio, en las subcapas PMA y PMD. Por ejemplo, módulos ópticos comerciales como los SFP o circuitos electrónicos como los CDR o SERDES en el estándar 1000Base-X serán compatibles con el esquema de encriptación propuesto.

En general, los codificadores de línea por bloque, como el 8b/10b, agrupan la información en bloques de $m$ bits que son mapeados en grupos de $n$ bits, donde $n > m$. Gracias a la redundancia introducida en los datos, el codificador por bloque logra las propiedades mencionadas anteriormente. Para llevar a cabo la encriptación antes que la codificación cada símbolo de $m$ bits ha de ser cifrado de forma que dé lugar a otro símbolo de $m$ bits, también válido en el estándar.

Hemos de tener en cuenta que el esquema de cifrado tiene la estructura de un cifrador por *streaming*, donde se aplica una operación matemática a cada uno de los datos del texto plano con la salida del generador de *keystream*. Si $S$ es el número posible de símbolos, la operación matemática a aplicar deberá ser una suma módulo-S en vez de la clásica XOR (suma módulo-2), ya que los datos en este caso no son binarios sino que se encuentran representados en base $S$. Además, los símbolos de $m$ bits han de ser mapeados a un valor perteneciente al intervalo $[0, S-1]$ antes de aplicar la suma, y el proceso inverso ha de ser llevado a cabo sobre el resultado de esta para obtener finalmente los símbolos cifrados de $m$ bits. Por otro lado, el generador de *keystream* deberá ser capaz de generar una secuencia uniformemente distribuida de valores en base $S$. Según esto, la estructura necesaria para el esquema de cifrado se muestra en

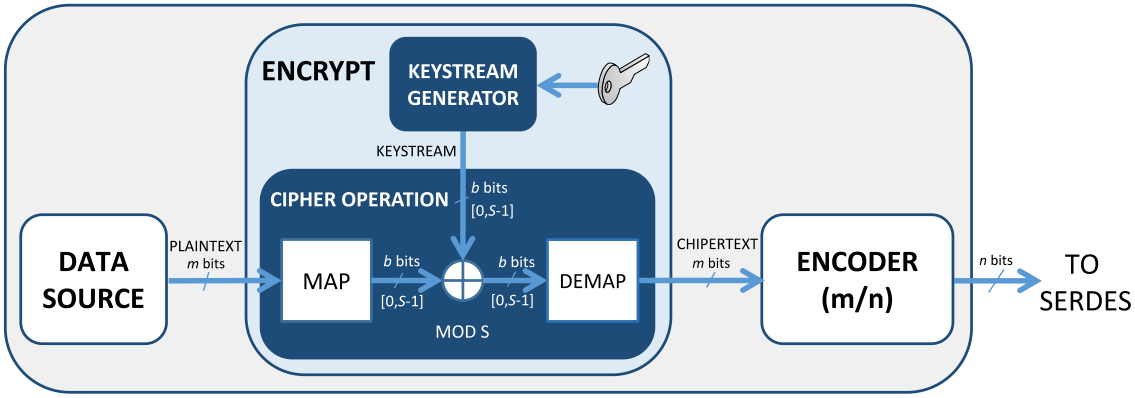



Fig.2-2. Localización y estructura genérica de un cifrador en una capa física con codificador de línea por bloque $m/n$.

la Fig.2-2. La estructura de desencriptación será la misma que la mostrada en dicha figura pero utilizando una resta módulo-S en vez de la suma. En la Fig.2-2, los valores mapeados en el intervalo $[0, S-1]$ son representados con $b$ bits siendo $b = \lceil \log_2 S \rceil$. Una vez encriptado el texto plano, este será demapeado a símbolos de $m$ bits y finalmente codificado a valores de anchura $n$ bits.

En el estándar 1000Base-X los símbolos de datos y de control suman 267 posibles símbolos, lo que implica que $S$=267 y $b$=9. Teniendo en cuanta esto, el esquema de encriptación genérico para 1000Base-X se muestra en la Fig.2-3. En este estándar, cada símbolo 8b/10b sin codificar se representa por un valor de 8 bits acompañado de un bit de control llamado flag K indicando el tipo de símbolo que le corresponde, que puede ser control o datos. Los símbolos son mapeados en un valor en base 267, antes de su encriptación y después son demapeados de nuevo y codificados en un valor de 10 bits, que será el que se transmita a la línea serie.

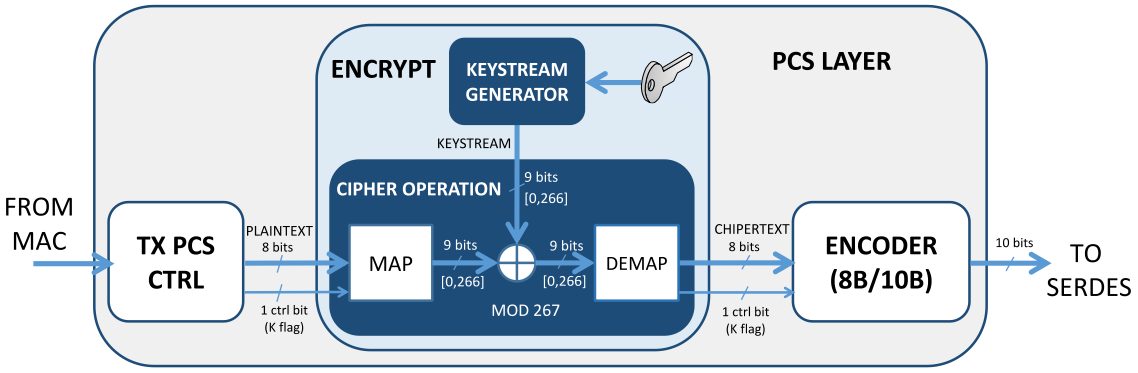



Fig.2-3. Esquema de la operación de encriptación en el estándar 1000Base-X.

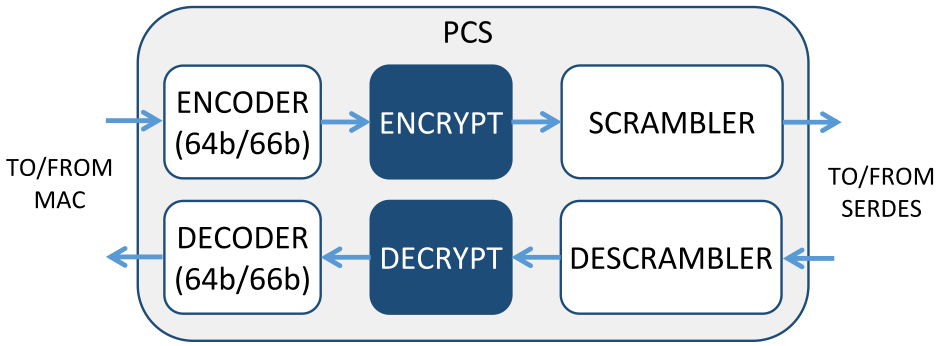



Fig.2-4. Emplazamiento de la función de encriptación en la capa PCS para 10GBase-R.

Como veremos más adelante, los esquemas de encriptación propuestos en esta tesis para el estándar 1000Base-X tratarán de dar diferentes soluciones para la implementación del generador de *keystream*, siendo la operación de cifrado igual en todos ellos.

## 2.2 Localización del cifrador y estructura para 10GBase-R

En el caso del estándar 10GBase-R, como ya se ha mencionado en el Apartado 1.4.2, se emplea una codificación densa por bloque denominada 64b/66b. En este tipo de esquemas la densidad en las transiciones de bit y su balanceo en DC son conseguidas de forma estadística gracias al *scrambler* después del codificador, mientras que la longitud de ráfaga es limitada gracias a la cabecera de sincronismo que se añade a cada bloque de 64 bits durante su codificación.

Una vez que los bloques de 64 bits son codificados dan lugar a bloques de 66 bits, de los cuales dos son la cabecera mencionada mientras que el resto conforman el *payload* de cada bloque. De estos bloques de 66 bits sólo el *payload* es procesado por el *scrambler*, dejando intacta la cabecera.

Para mantener las propiedades del *scrambler* será necesario por tanto llevar a cabo la encriptación de los datos justo antes del mismo [PER19c], [PER19d], tal y como se muestra en la Fig.2-4.

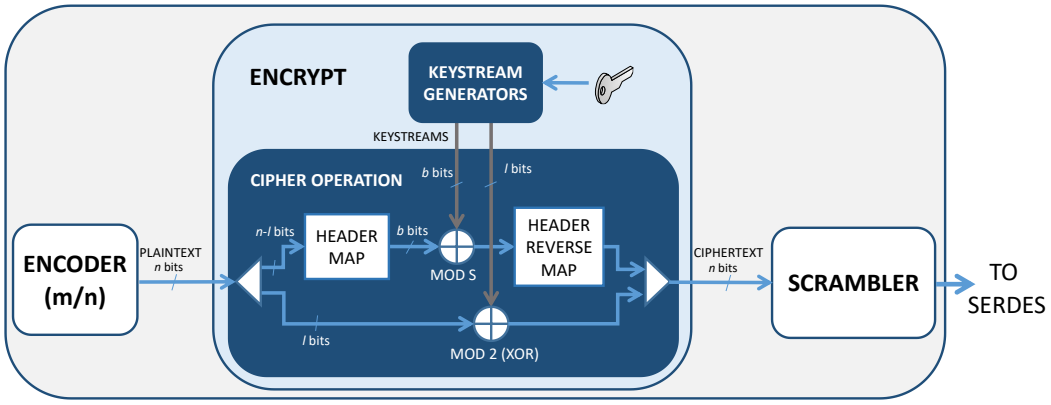



Fig.2-5. Localización y estructura genérica de un cifrador en una capa física con una codificación densa y *scrambler*.

Al igual que en con el estándar 1000Base-X el mecanismo de encriptación está basado en un cifrado por *streaming*, por lo que el cifrado se ha de llevar a cabo operando por separado cada bloque codificado con la secuencia de *keystream*.

En una codificación densa como esta se ha de mantener intacta la transición de bits introducida por la cabecera a la par que es encriptada. Es necesario, por tanto, obtener un valor válido de cabecera después de su encriptación. Para ello, después de la codificación, esta ha de ser mapeada en el intervalo $[0, S-1]$ siendo $S$ el número de posibles valores que puede tomar según el estándar. Su encriptación se llevará a cabo mediante una suma módulo-S entre dichos valores mapeados y una secuencia de *keystream* que también tomará valores en $[0, S-1]$. Finalmente, a la cabecera encriptada se le aplicará el mapeo inverso realizado antes del cifrado obteniendo un valor válido del estándar que mantenga las transiciones de bit.

Para el resto de la información del bloque, el *payload*, será necesario llevar a cabo la operación XOR entre dichos bits y otra secuencia de *keystream* que será binaria. Directamente el valor del *payload* será concatenado a la cabecera encriptada obteniendo el resultado final del bloque encriptado.

Según lo explicado anteriormente la estructura general de la operación de cifrado se muestra en la Fig.2-5, donde cada bloque codificado está representado por $n$ bits de los cuales $l$ representan al *payload*. La cabecera de sincronismo, mapeada en el intervalo $[0, S-1]$, vendrá representada por $b$ bits con $b = [\log_2 S]$.

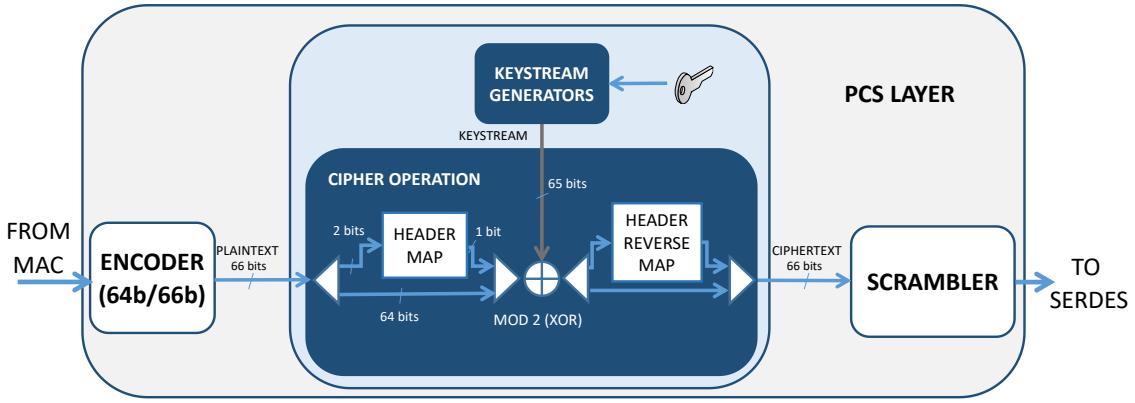



Fig.2-6. Esquema de la operación de encriptación en el estándar 10GBase-R. La cabecera de de sincronismo es mapeada en un bit y concatenada al *payload*. Ambos son encriptados mediante la XOR con un *keystream* de 65 bits de ancho.

En el caso de la codificación 64b/66b los posibles valores de la cabecera son sólo dos, lo que implica que la cabecera sea mapeada en un solo bit y se pueda emplear la operación XOR en su encriptación. Por ello, particularizando el esquema de la figura Fig.2-5 al del estándar 10GBase-R se obtiene la estructura mostrada en la Fig.2-6, donde sólo es necesario un *keystream* binario de anchura 65 bits para realizar el cifrado de los bloques.

## 2.3 Encriptación por *streaming* sincronizada para 1000Base-X

Tal y como se ha explicado en el Apartado 1.5.2 los cifradores por *streaming* síncronos necesitan que los generadores de *keystream* en el receptor y transmisor estén sincronizados para que los símbolos Ethernet originales se puedan desencriptar correctamente. Para ello es necesario un mecanismo de sincronización entre ambos terminales.

Un protocolo básico de sincronización se ha aplicado en las soluciones propuestas para 1000Base-X y este ha sido detallado en [PER19a]. En este mecanismo se ha propuesto la introducción de un nuevo set ordenado Ethernet que se ha denominado /X/ y cuya estructura se muestra en la Tabla 2-1 junto al resto de sets ordenados ya existentes en el estándar.

| Code | Ordered Set | Number of Code-Groups | Encoding |
|---|---|---|---|
| /C/ | CONFIGURATION | | Alternating /C1/ and /C2/ |
| /C1/ | Configuration 1 | 4 | /K28.5/D21.5/Config_Reg |
| /C2/ | Configuration 2 | 4 | /K28.5/D2.2/Config_Reg |
| /I/ | IDLE | | Correcting /I1/, Preserving /I2/ |
| /I1/ | Idle 1 | 2 | /K28.5/D5.6/ |
| /I2/ | Idle 2 | 2 | /K28.5/D16.2/ |
| | ENCAPSULATION | | |
| /R/ | Carrier_extend | 1 | /K23.7/ |
| /S/ | Start_of_Packet | 1 | /K27.7/ |
| /T/ | End_of_Packet | 1 | /K29.7/ |
| /V/ | Error_Propagation | 1 | /K30.7/ |
| | **ENCRYPTION** | | **Enable/Disable** |
| **/X/** | **Cipher_on_off** | **4** | **/K28.1/D21.5/D21.2/D21.2/** |

Tabla 2-1. Estructura de los sets ordenados Ethernet en 1000Base-X incluyendo la propuesta del nuevo set ordenado /X/.

El mecanismo de sincronización consiste en la introducción del set ordenado /X/ antes del módulo que ejecuta la operación de encriptación. Para su inserción en el flujo de símbolos 8b/10b dos sets consecutivos de tipo *idle* son reemplazados por los símbolos que componen /X/. Cuando /X/ es detectado justo antes del bloque de encriptación se habilita tanto la operación de cifrado como el generador de *keystream*, comenzando así a cifrar los símbolos después de que /X/ haya sido transmitido.

De igual forma, en el receptor, el set /X/ al ser detectado habilita la operación de descifrado y el generador de *keystream*, lo que permite la desencriptación de los símbolos 8b/10b recibidos posteriormente a /X/. Finalmente /X/ es reemplazado en el receptor por dos sets consecutivos de tipo *idle* antes de que el flujo de símbolos llegue al controlador PCS.

Para llevar a cabo esta manipulación del flujo de octetos 8b/10b es necesario introducir un bloque de control tal y como se muestra en la Fig.2-7, donde el bloque MANAGEMENT es el encargado de insertar y extraer los nuevos sets /X/.

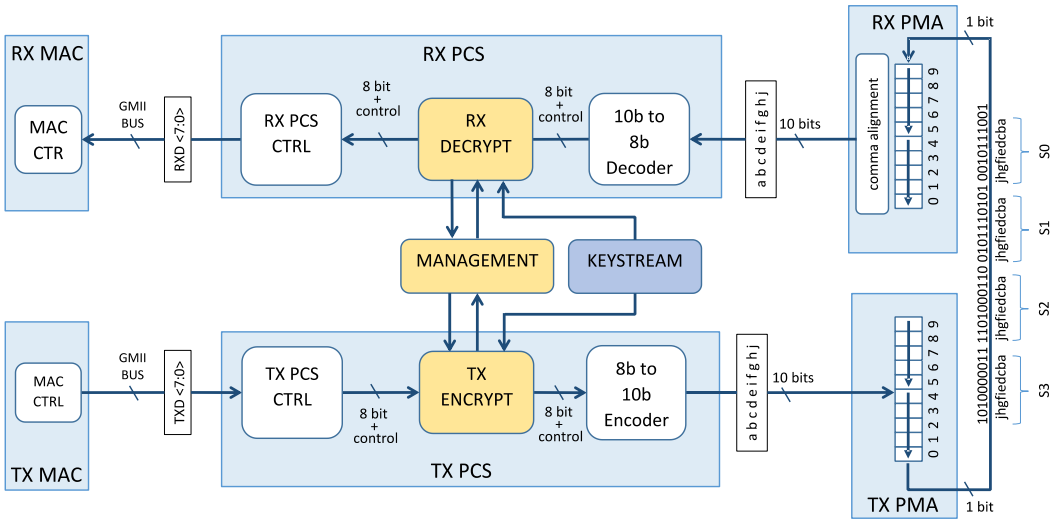
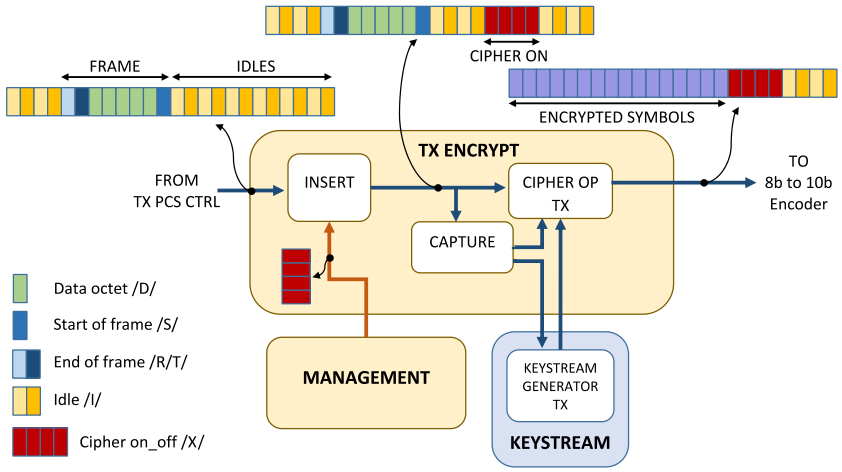
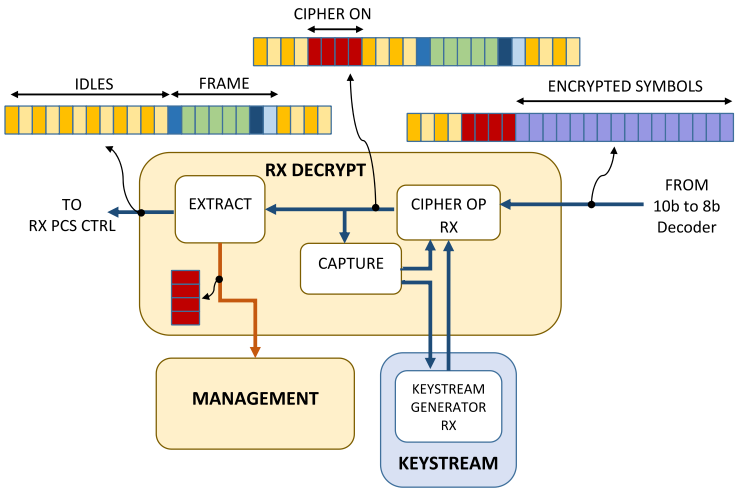



Fig.2-7. Emplazamiento del módulo de encriptación en la capa física 1000Base-X junto a módulo de control.




(a)




(b)

Fig.2-8. Mecanismo inicial de sincronización, basado en el set ordenado /X/, (a) transmisión, (b) recepción.

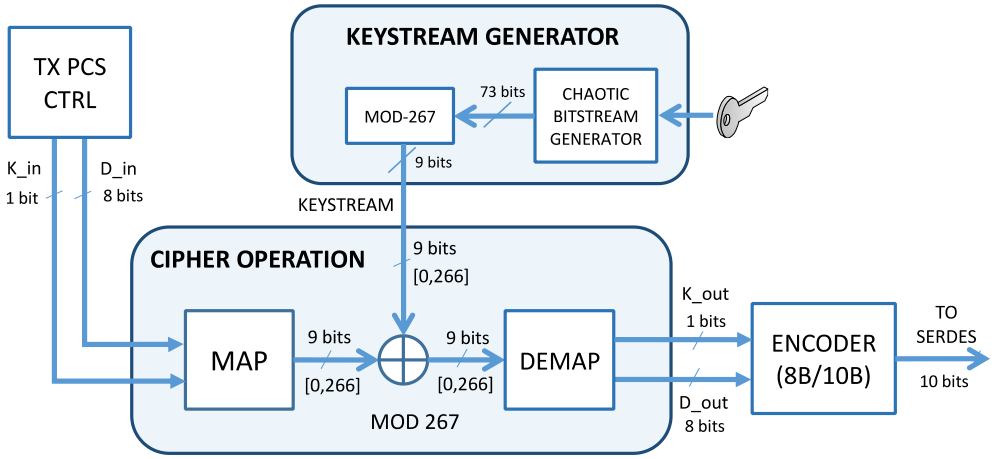



Fig.2-9. Estructura del generador de *keystream* ad-hoc junto con la operación de encriptación. en el estándar 1000Base-X.

Este mecanismo de inserción y extracción de /X/ se muestra de forma simplificada en la Fig.2-8. Una explicación más detallada del funcionamiento del sistema se encuentra disponible en [**PER19a**].

A continuación, en los siguientes apartados se presentan las diferentes soluciones aportadas para la implementación del generador de *keystream*, donde se han empleado tanto soluciones ad-hoc como basadas en el empleo de cifradores por bloque.

### 2.3.1 Encriptación por *streaming* ad-hoc para 1G

Como ya se ha mencionado en el Apartado 1.5, los cifradores por *streaming* se pueden implementar ad-hoc con estructuras específicamente diseñadas para ese propósito o mediante bloques preconstruidos, como los cifradores por bloque funcionando en ciertos modos de operación.

En la implementación desarrollada en [PER19a] se optó por la primera opción. Concretamente se implementó un cifrador por *streaming* basado en un algoritmo caótico consistente en una modificación del mapa caótico denominado *skew-tent map* [GAR17a], [GAR17b]. La estructura propuesta del *keystream* consistió en un generador pseudoaleatorio basado en el mencionado algoritmo a cuya salida se le aplicó la operación módulo-267, tal y como se muestra en la Fig.2-9. En esta estructura, al igual que en la Fig.2-3, del controlador PCS salen los símbolos 8b/10b formados por su valor de 8 bits D_in más el *flag* de control K, K_in.

Ambos, {D_in, K_in}, formarán la entrada del módulo de cifrado. Después de su encriptación, los nuevos símbolos encriptados {D_out, K_out} son codificados para obtener los 10 bits que se envían al SERDES.

En [PER19a] la operación módulo-267 es utilizada para obtener un *keystream* compuesto por palabras de 9 bits con valores comprendidos entre 0 y 266, aptos para cifrar los símbolos 8b/10b ya mapeados. Un inconveniente surge al emplear este método de generación de *keystream,* ya que si la salida del generador pseudoaleatorio estuviera compuesta por palabras de 9 bits sus valores estarían comprendidos entre 0 y $2^9$-1. Al realizar la operación módulo-267, aunque obtuviéramos valores entre 0 y 266, se introduciría un sesgo en la distribución de la secuencia de números resultante, es decir, el *keystream* no estaría uniformemente distribuido.

Según la recomendación del NIST (National Institute of Standards and Technology) [NIS15], para generar secuencias de números pseudoaleatorios comprendidas en un intervalo determinado y obtenidas a partir de una operación módulo sobre una secuencia pseudoaleatoria, es necesario que el tamaño en bits de los valores sobre los que se aplica la operación módulo sea suficientemente grande, de forma que se evite el sesgo mencionado anteriormente. Concretamente, en [NIS15] se establece que dicho tamaño ha de superar al menos en 64 bits a la anchura los datos a la salida de la operación módulo. Esto da como resultado que el generador caótico pseudoaleatorio tenga que producir palabras de al menos 73 bits, tal y como se muestra en Fig.2-9.

La estructura del generador pseudoaleatorio ha sido diseñada mediante la implementación de un banco de 9 generadores caóticos cuyas salidas son concatenadas para lograr la anchura de 73 bits deseada. Cada uno de los generadores está basado en el algoritmo *skew-tent map* según la modificación propuesta en [GAR17b] y su estructura se muestra en la Fig.2-10. Las operaciones del algoritmo *skew-tent map* son llevadas a cabo en el bloque STM_CELL y los 8 bits más bajos de su salida $x_i$ son operados con la salida de un LFSR (Linear Feedback Shift Register). La palabra resultante $\tilde{x_i}$ es utilizada como nueva entrada de la celda STM_CELL y finalmente sus 8 bits más bajos, $\tilde{x_i}[7:0]$ son tomados como salida del generador caótico.

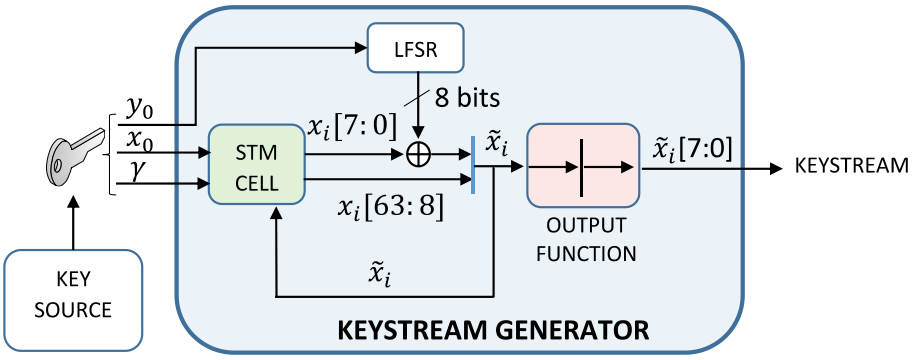



Fig.2-10. Generador caótico básico basado en el algoritmo *skew-tent map*.

La estructura final de los generadores caóticos y su agrupación en el banco de generadores para dar lugar al *keystream* se muestra en la Fig.2-11.

Para considerar la secuencia de *keystream* como válida, esta debe ser indistinguible de una secuencia verdaderamente aleatoria. Por un lado es necesario comprobar que la secuencia del generador pseudoaleatorio pasa ciertos test de aleatoriedad y por otro que el sesgo introducido por la función módulo-267 es despreciable.

Para lo primero se llevó a cabo la batería de test de aleatoriedad recomendados por el NIST [NIS10] a diferentes secuencias generadas por el generador básico mostrado en la Fig.2-10. Esta batería de test comprende diversos tests estadísticos que permiten determinar si una secuencia es o no aleatoria. Estos

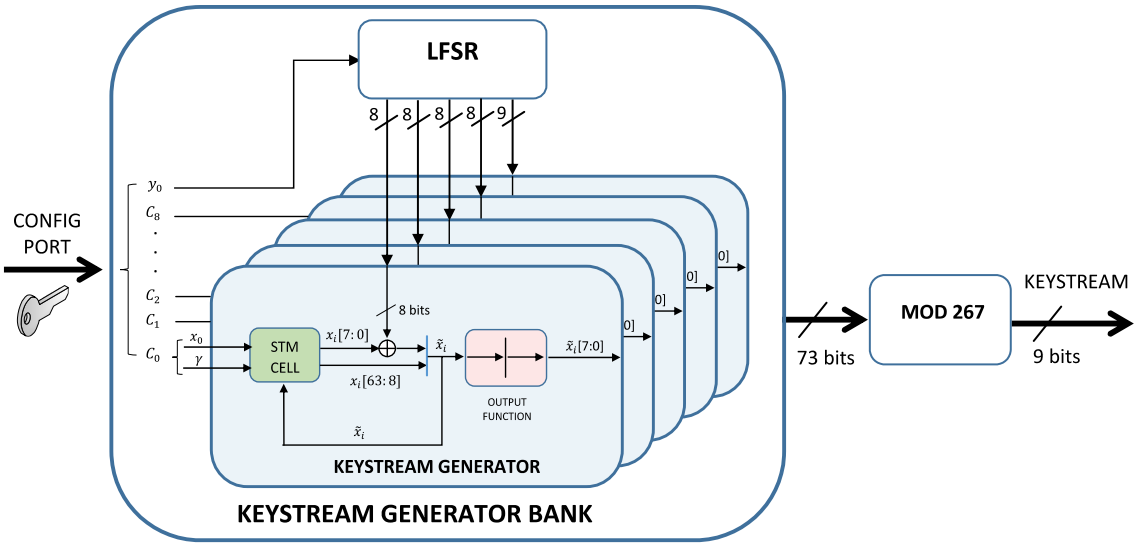



Fig.2-11. Banco de generadores caóticos junto a operación módulo. La salida de cada módulo caótico está formada por 8 bits, $\tilde{x}_i[7:0]$, excepto en el último de los módulos del que se toman los 9 bits $\tilde{x}_i[8:0]$.

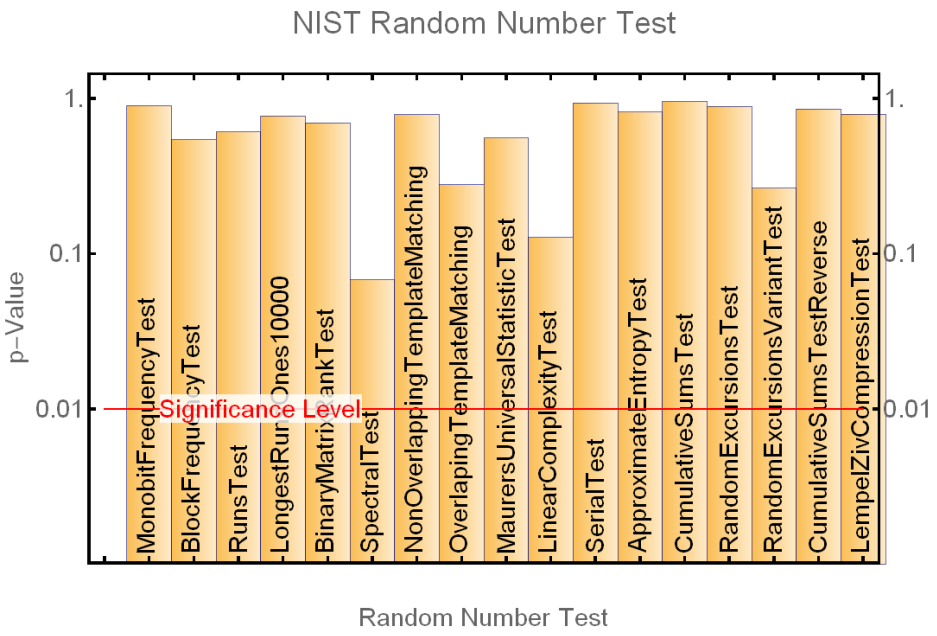



Fig.2-12. Resultados de los test NIST para un *bitstream* generado por el generador caótico básico.

se definieron con un nivel de significancia $\alpha = 0.01$ y como ejemplo, en la Fig.2-12 se muestra el resultado satisfactorio de los mismos al aplicarse a una secuencia en particular.

Para lo segundo se obtuvieron los histogramas de las secuencias de salida de la operación modulo-267, tanto al cumplir la recomendación del NIST usando datos de entrada con anchura de 73 bits, como empleando datos de tan solo 9 bits, es decir sin cumplir dicha recomendación. Se observó claramente cómo existía un sesgo en la distribución de los valores de secuencia del segundo caso mientras que era inexistente en el primero. Una manera de analizar este sesgo sería aplicando algún test de aleatoriedad a los datos de salida de la operación módulo, sin embargo las baterías de test más utilizadas, tales como la del NIST [NIS10], u otras como Diehard [MAR97] o TestU01 [LEC07] están pensados para secuencias de números enteros dentro de un rango que sea potencia de dos. En nuestro caso la secuencia no se encuentra en un rango así, ya que está comprendida entre 0 y 266. Para analizar su aleatoriedad se aplicaron varios de los tests propuestos por Knuth [KNU97]. Concretamente la secuencia de salida fue sometida al test de frecuencia, test serie para parejas y tripletes de símbolos y test de póker, con lo que se pudo concluir que efectivamente se correspondía con una secuencia uniformemente distribuida entre 0 y 266.

Una de las ventajas de los cifradores por bloque respecto a los cifradores por *streaming* es que el análisis criptográfico suele estar más desarrollado y es más compresible. En este caso en particular, tanto en [GAR17b] como en [**PER19a**] no existe un criptoanálisis exhaustivo del algoritmo caótico que garantice su seguridad. Por ello resulta de interés proponer otras soluciones basadas en cifradores por bloque con un criptoanálisis conocido. Como se verá en los siguientes apartados, estas soluciones tendrán que ser capaces de proponer modos de funcionamiento seguros para dichos cifradores que al mismo tiempo preserven el formato de los símbolos 8b/10b.

### 2.3.2 Encriptación por *streaming* con modo CTR-FPE para 1G

Actualmente no existe ningún estándar o recomendación de cifrador por *streaming* que preserve el formato de los datos del texto plano. Sin embargo sí que existen modos de operación recomendados por el NIST [NIS16] para cifradores por bloque. Estos modos son denominados modos FPE (Format Preserving Encryption). Aunque la recomendación [NIS16] describe modos de operación, en la práctica las estructuras de encriptación resultantes al emplear dichos modos se comportan propiamente como cifradores por bloque, es decir, estos modos no realizan una encriptación por *streaming*, como CTR o CBC, sino que cifran bloques de datos de forma independiente, con la particularidad de que pueden configurarse para trabajar con diferentes anchuras de bloque y con datos representados en diferentes bases numéricas, no sólo binarias como en los cifradores por bloque tradicionales.

De hecho, en [NIS16], los modos descritos se basan fundamentalmente en el empleo de una red tipo Feistel, y tal y como se demostró en [LUB88], dicho tipo de redes pueden ser consideradas buenas PRPs y al mismo tiempo ser empleadas en la construcción de cifradores por bloque, como por ejemplo DES (Data Encryption Standard).

Una posible alternativa para el diseño de un generador de *keystream* que preserve el formato para un texto plano comprendido entre 0 y 266 es la utilización de uno de los modos FPE descritos en [NIS16] a modo de cifrador por bloque, funcionando a su vez en un modo similar a CTR. El algoritmo de este

modo sería como el mostrado en la Fig.1-10 pero empleando una base distinta a 2. Siendo $F_k$ la función del "cifrador por bloque" FPE tal que $F: \mathcal{K} \times \{0, 1, \dots, S\text{-}1\}^l \rightarrow \{0, 1, \dots, S\text{-}1\}^l$ donde $l$ es el tamaño de bloque, $\mathcal{K}$ el espacio de claves y $S$ la base en la que trabaja el cifrador y en la que se expresan los símbolos del texto plano, el nuevo algoritmo en modo CTR es descrito en la Fig.2-13.

La diferencia principal con el modo CTR original estriba en que el contador ya no es binario sino que se encuentra en base $S$, al igual que la operación de encriptación, ya no es una XOR (suma módulo-2), sino una suma módulo-$S$.

De acuerdo con [PER19b], es posible emplear este mecanismo de cifrado obteniendo al menos la misma seguridad que tendría un cifrador binario estándar tipo AES funcionando en modo CTR tradicional. Tal y como se explica en [PER19b] la ventaja de cualquier adversario A sobre el esquema CTR propuesto, al que denominamos CTR-FPE, viene determinada por la siguiente ecuación:

$$ADV_{CTR-FPE}^{IND-CPA}(A) \leq 2 \cdot ADV_F^{PRP}(B) + \frac{\mu^2}{T^2 \cdot l^2 \cdot S^l} \quad (5)$$

donde $\mu$ es el número total de bits encriptados durante el juego del adversario, $l$ es el tamaño de bloque de entrada y salida del cifrador por bloque FPE, $S$ es la base numérica del texto plano y texto cifrado y $T$ el número de bits representado por cada símbolo. $ADV_F^{PRP}(B)$ es la ventaja de tipo PRP frente al cifrador y $ADV_{CTR-FPE}^{IND-CPA}(A)$ es la ventaja IND-CPA de A frente al esquema propuesto CTR-FPE.

Teniendo en cuenta la expresión de la ventaja de un adversario A sobre el esquema CTR en la Fig.1-10 y comparándola con la que tendría sobre el esquema descrito en la Fig.2-13, en [PER19b] se concluye que el esquema en la Fig.2-13 tendrá la misma seguridad o mayor siempre y cuando se cumpla la siguiente condición:

$$T^2 \cdot l_{CTR-FPE}^2 \cdot S^L \cdot 2^{l_{CTR-FPE}} \geq l_{CTR}^2 \cdot 2^{l_{CTR}} \quad (6)$$

**algorithm** $E_k(M)$
static $ctr \leftarrow 0$

Break $M$ into $l$-symbol blocks
$M[0] \dots M[m-1]$

$IV \leftarrow ctr$
**for** $i \leftarrow 0$ **to** $m-1$ **do**
  $C[i] \leftarrow (F_k(ctr) + M[i]) mod\ S$
  $ctr \leftarrow ctr + 1$
$C \leftarrow C[0] \dots C[m-1]$
**return** $\langle IV, C \rangle$

**algorithm** $D_k(\langle IV, C \rangle)$

Break $C$ into $l$-symbol blocks
$C[0] \dots C[m-1]$

$ctr \leftarrow IV$
**for** $i \leftarrow 0$ **to** $m-1$ **do**
  $M[i] \leftarrow (F_k(ctr) - C[i]) mod\ S$
  $ctr \leftarrow ctr + 1$
$M \leftarrow M[0] \dots M[m-1]$
**return** $M$

Fig.2-13. Algoritmos de encriptación y desencriptación para CTR en base $S$.

donde $l_{CTR-FPE}$ y $l_{CTR}$ son las longitudes de bloque de los cifradores en modo CTR-FPE y CTR, respectivamente.

De acuerdo con esto se concluye que para lograr la misma seguridad que un cifrador por bloque de 128 bits, $l_{CTR} = 128$, como AES funcionando en modo CTR, será necesario que el cifrador por bloque FPE en modo CTR-FPE tenga al menos una longitud de bloque igual o superior a 16 símbolos, $l_{CTR-FPE} \geq 16$, ya que en nuestro caso $S = 267$ y $T = log_2 S \cong 8$ bits/símbolo.

En [PER19b] el cifrador por bloque FPE fue implementado utilizando una estructura como la recomendada en el modo FF3 del NIST [NIS16]. De entre las dos estructuras recomendadas, FF1 y FF3, FF3 es la que menos rondas de procesado requiere en su red Feistel, y por tanto menores recursos hardware son necesarios. En este modo, la longitud del bloque de la estructura FPE está limitada por lo siguiente:

$$\begin{gathered} S \in [2 \dots 2^{16}] \\ S^{minlen} \geq 100 \\ 2 \leq minlen \leq maxlen \leq 2 \lfloor log_S(2^{96}) \rfloor \end{gathered} \tag{7}$$

Donde $minlen$ y $maxlen$ son los límites mínimo y máximo para el tamaño de bloque y $\lfloor \cdot \rfloor$ el operador de truncado. Como la base de nuestro cifrador es $S$ = 267, el valor de $l_{CTR-FPE}$ tendrá que estar entre 2 y 22. Además, como según

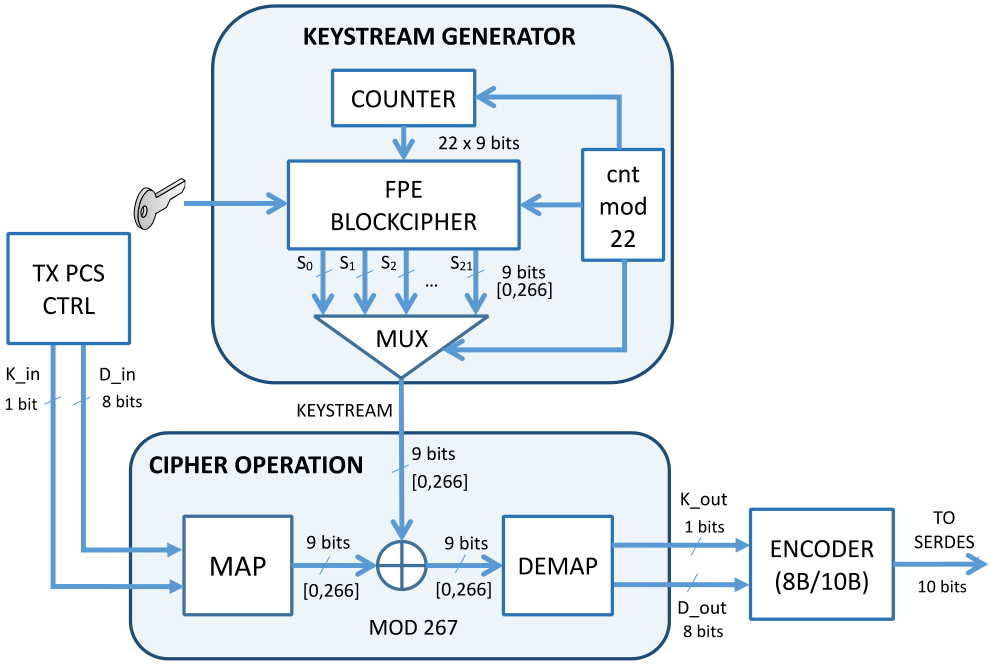



Fig.2-14. Estructura completa del sistema de encriptación por *streaming* sincronizado empleando un cifrador por bloque FPE.

(6), $l_{CTR-FPE} \geq 16$, en la implementación de [PER19b] se escogió $l_{CTR-FPE} = 22$, ya que este valor implicaba una implementación *hardware* más eficiente en el caso de emplear una arquitectura de tipo *pipeline* en la estructura FF3.

Dado este parámetro y sustituyendo el bloque de generación de *keystream* de la estructura mostrada en Fig.2-3 por el generador descrito en la Fig.2-13 el cifrador por *streaming* de esta solución fue implementado tal y como se muestra en la Fig.2-14. En dicha estructura el módulo FPE_BLOCKCIPHER representa al cifrador por bloque FPE cuya salida da lugar al *keystream* necesario para la encriptación. La anchura de bloque de este cifrador es de 22 símbolos y su entrada proviene del módulo contador COUNTER. A su vez, el módulo cnt_mod_22 es un contador módulo 22. Por cada vuelta de este contador se incrementa el valor de COUNTER y se obtiene un nuevo valor a la salida de FPE_BLOCKCIPHER.

Tal y como se ha comentado anteriormente, la estructura del cifrador por bloque FPE consiste en una red de tipo Feistel y según [NIS16] esta ha de ser implementada internamente con un cifrador por bloque AES. Según esto, la implementación de la estructura FF3 se llevó a cabo tal y como se muestra en la

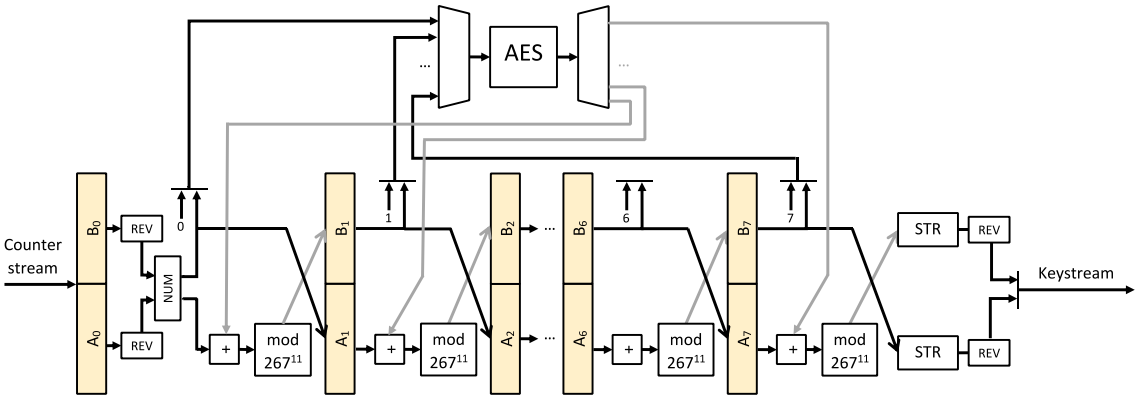



Fig.2-15. Estructura interna de la implementación del cifrador por bloque FPE.

Fig.2-15, donde los datos de entrada son los valores del módulo COUNTER de la Fig.2-14 y el resultado a la salida los valores del propio *keystream*.

### 2.3.3 Encriptación por *streaming* con modo CTR-MOD para 1G

Con el objetivo de reducir la complejidad y los recursos *hardware* de la solución descrita en el Apartado 2.3.2, en esta solución, detallada en [PER20a], el generador de *keystream* fue construido a partir de un cifrador por bloque binario funcionando en modo CTR a cuya salida se le aplicó la operación módulo-267.

Esta estructura es similar a la del Apartado 2.3.1 pero con la posibilidad de emplear un cifrador por bloque cuya seguridad esté reconocida y su criptoanálisis sea más claro que con un cifrado por *streaming* diseñado ad-hoc. Aún con todo, a pesar de poder aplicar la recomendación del NIST [NIS15] como en [PER19a], el hecho de utilizar una operación módulo, y por tanto de introducir un sesgo, debería ser analizada desde el punto de vista criptográfico.

Si $E_k$ es la función de encriptación que describe al cifrador por bloque de longitud de bloque $l$, la estructura mencionada ha sido denominada modo de operación CTR-MOD y se puede representar como en Fig.2-16.

En esta estructura se toman $L$ bits de la salida del cifrador $E_k$ y se operan con la operación módulo-S obteniendo símbolos en base $S$ que representan $T$ bits de información. El hecho de truncar la salida de $E_k$ en $L$ bits equivale a realizar la operación módulo-$2^L$. En la Fig.2-16 se ha denominado $F_k$ a la función formada

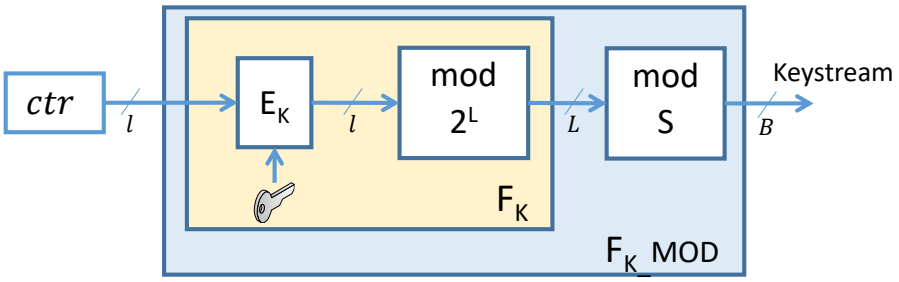



Fig.2-16. Estructura del generador de *keystream* basado en el cifrador por bloque $E_k$ funcionando en el modo CTR-MOD. Cada símbolo del *keystream* $\in [0, S\text{-}1]$ estará codificado en $B$ bits y representará a $T$ bits de información.

por $E_k$ más la operación módulo-$2^L$, y a $F_{k\_}MOD$ a la función formada por $F_k$ y la operación módulo-S. La función resultante $F_{k\_}MOD$ tiene por entrada números en base binaria de longitud $l$ bits provenientes de un contador $ctr$, mientras que su salida será el *keystream* formado por valores enteros en base $S$.

En [PER20a] se demuestra que la ventaja de un adversario A sobre dicho esquema se puede expresar como en (8).

$$ADV_{CTR-MOD}^{IND-CPA}(\mathrm{A}) \leq 2 \cdot ADV_{E}^{PRP}(B) + \frac{\mu^2}{T^2 \cdot 2^l} + \frac{\mu}{T \cdot 2^{I-1}} \tag{8}$$

donde $ADV_{CTR-MOD}^{IND-CPA}(\mathrm{A})$ es la ventaja del adversario A sobre dicho modo, $ADV_{E}^{PRP}(B)$ es la ventaja de tipo PRP sobre $E_k$, $\mu$ es el número de bits encriptado, $T$ son los bits de información por símbolo, $l$ es el tamaño de bloque e $I$ es la diferencia entre los bits de entrada a la operación modulo-S y $T$.

Para garantizar que el cifrador $E_k$ operando en el modo CTR-MOD logra al menos la misma seguridad que un cifrador tradicional, en [PER20a] se compara la expresión de $ADV_{CTR-MOD}^{IND-CPA}(\mathrm{A})$ con la expression de la ventaja de A sobre el modo CTR clásico funcionando con un cifrador con longitud de bloque $l_{CTR}$. De esta forma se extrae la conclusión de que la longitud del bloque de $E_k$ ha de cumplir la condición mostrada en (9).

$$l_{CTR-MOD} \geq T + 1 + \log_2\left(\frac{{l_{CTR}}^2 \cdot 2^{l_{CTR}}}{T} \cdot \left(1 + \frac{1}{T \cdot 2^{T+1}}\right)\right) \tag{9}$$

Si asumimos que se desea conseguir una seguridad de tipo IND-CPA igual o mayor que un cifrador de 128 bits y teniendo en cuenta que $T = log_2 S$ con $S$=267, entonces se deberá cumplir que $l_{CTR-MOD} \geq 149$ bits.

De acuerdo con esto, al reemplazar el bloque de generación de *keystream* de la Fig.2-3 con la estructura mostrada en Fig.2-16 se obtiene el esquema completo para esta solución, mostrado en la Fig.2-17.

En esta implementación se empleó el cifrador por bloque Rijndael. Como se ha de cumplir la condición $l_{CTR-MOD} \geq 149$ bits, de sus tres posibles anchuras de bloque, 128, 192 y 256 se optó por una configuración con 192 bits, que es el valor inmediatamente superior a 149.

Como veremos en el Apartado 2.6, con este esquema conseguimos mejores resultados en cuanto a recursos *hardware* que con las soluciones mostradas en los Apartados 2.3.1 y 2.3.2, al mismo tiempo que garantizamos la seguridad del sistema al basarnos en cifradores por bloque estándar cuyo criptoanálisis es claro y conocido.

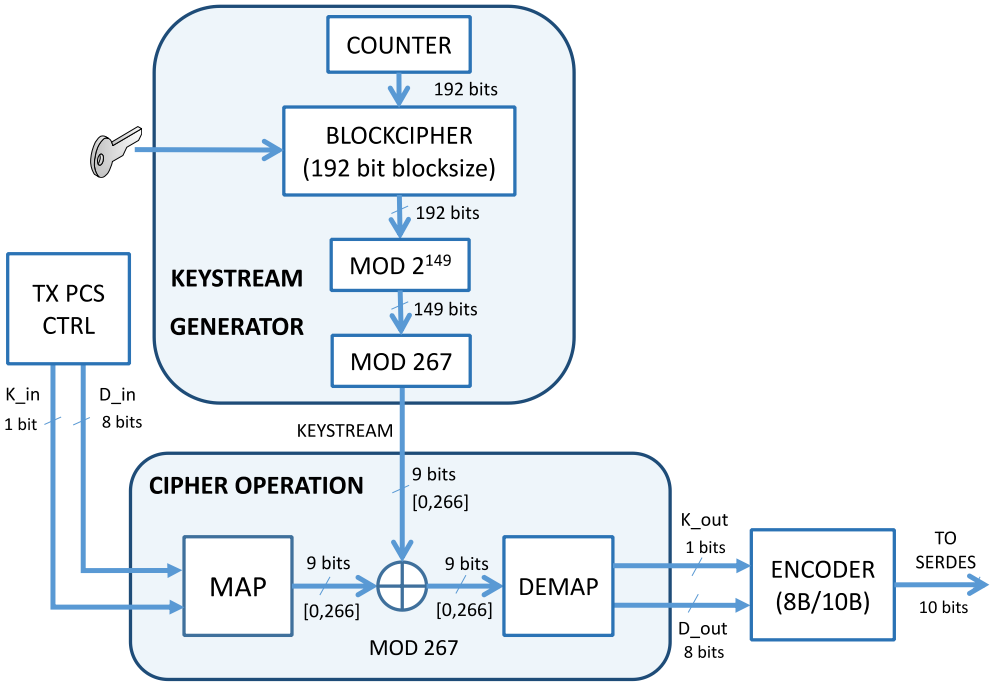



Fig.2-17. Estructura completa del sistema de encriptación por *streaming* empleando un cifrador por bloque funcionando en modo CTR-MOD.

## 2.4 Encriptación por *streaming* sincronizada para 10GBase-R

A igual que para el estándar 1000Base-X, en una solución de encriptación por *streaming* sincronizado para 10GBase-R es necesario un mecanismo para lograr el alineamiento de las secuencias de *keystream* tanto en el receptor como en el transmisor.

Un mecanismo básico de sincronización se aplicó en la solución para 10GBase-R y este fue detallado en [PER19c]. Al igual que en [PER19a], se definió una secuencia de sincronización, pero esta vez acorde a la codificación de este estándar, es decir basada en bloques 64b/66b. De entre los diferentes tipos de bloques de 66 bits fue escogido el tipo 0x55 tal y como se muestra en la Tabla 2-2.

| **Input Data** | **Sync** | **Block Payload** | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| **Bit Position:** / **Data Block Format:** | 0 1 | 2 | | | | | | | 65 |
| $D_0\ D_1\ D_2\ D_3/D_4\ D_5\ D_6\ D_7$ | 01 | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
| **Control Block Formats:** | | **Block Type** | | | | | | | |
| $C_0\ C_1\ C_2\ C_3/C_4\ C_5\ C_6\ C_7$ | 10 | 0x1e | $C_0$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $C_0\ C_1\ C_2\ C_3/O_4\ D_5\ D_6\ D_7$ | 10 | 0x2d | $C_0$ | $C_1$ | $C_2$ | $C_3$ | $O_4$ | $D_5$ | $D_6$ | $D_7$ |
| $C_0\ C_1\ C_2\ C_3/S_4\ D_5\ D_6\ D_7$ | 10 | 0x33 | $C_0$ | $C_1$ | $C_2$ | $C_3$ | | $D_5$ | $D_6$ | $D_7$ |
| $O_0\ D_1\ D_2\ D_3/S_4\ D_5\ D_6\ D_7$ | 10 | 0x66 | $D_1$ | $D_2$ | $D_3$ | $O_0$ | | $D_5$ | $D_6$ | $D_7$ |
| $O_0\ D_1\ D_2\ D_3/O_4\ D_5\ D_6\ D_7$ | 10 | 0x55 | $D_1$ | $D_2$ | $D_3$ | $O_0$ | $O_4$ | $D_5$ | $D_6$ | $D_7$ |
| $S_0\ D_1\ D_2\ D_3/D_4\ D_5\ D_6\ D_7$ | 10 | 0x78 | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
| $O_0\ D_1\ D_2\ D_3/C_4\ C_5\ C_6\ C_7$ | 10 | 0x4b | $D_1$ | $D_2$ | $D_3$ | $O_0$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $T_0\ C_1\ C_2\ C_3/C_4\ C_5\ C_6\ C_7$ | 10 | 0x87 | | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $D_0\ T_1\ C_2\ C_3/C_4\ C_5\ C_6\ C_7$ | 10 | 0x99 | $D_0$ | | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $D_0\ D_1\ T_2\ C_3/C_4\ C_5\ C_6\ C_7$ | 10 | 0xaa | $D_0$ | $D_1$ | | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $D_0\ D_1\ D_2\ T_3/C_4\ C_5\ C_6\ C_7$ | 10 | 0xb4 | $D_0$ | $D_1$ | $D_2$ | | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $D_0\ D_1\ D_2\ D_3/T_4\ C_5\ C_6\ C_7$ | 10 | 0xcc | $D_0$ | $D_1$ | $D_2$ | $D_3$ | | $C_5$ | $C_6$ | $C_7$ |
| $D_0\ D_1\ D_2\ D_3/D_4\ T_5\ C_6\ C_7$ | 10 | 0xd2 | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | | $C_6$ | $C_7$ |
| $D_0\ D_1\ D_2\ D_3/D_4\ D_5\ T_6\ C_7$ | 10 | 0xe1 | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | | $C_7$ |
| $D_0\ D_1\ D_2\ D_3/D_4\ D_5\ D_6\ T_7$ | 10 | 0xff | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | |

Tabla 2-2. Formato de los diferentes tipos de bloques 64b/66b disponibles en el estándar 10GBase-R.

| LANE 0 | LANE 1 | LANE 2 | LANE 3 | DESCRIPTION |
|---|---|---|---|---|
| Sequence | 0x00 | 0x00 | 0x00 | Reserved |
| Sequence | 0x00 | 0x00 | 0x01 | Local Fault |
| Sequence | 0x00 | 0x00 | 0x02 | Remote Fault |
| Sequence | 0x00 | 0x00 | 0x03 | Link Interruption |
| **Sequence** | **0x00** | **0x00** | **0x04** | **Cipher ON** |
| **Sequence** | **0x00** | **0x00** | **0x05** | **Cipher OFF** |
| Sequence | ≥0x00 | ≥0x00 | ≥0x06 | Reserved |

Tabla 2-3. Sets ordenados definidos en el estándar 10GBase-R y los nuevos sets propuestos.

El bloque 64b/66b de tipo 0x55 está compuesto por una cabecera de sincronismo '10' y dos sets ordenados, $\{O_0, D_1, D_2, D_3\}$ y $\{O_4, D_5, D_6, D_7\}$. Cada set ordenado es iniciado por un carácter de control $O$ y contiene tres caracteres de datos $D$ que definen el tipo de secuencia del set. Estas secuencias son empleadas en el estándar para notificar ciertas señalizaciones entre los dos terminales de una comunicación y tienen un valor comprendido entre 0x000000 y 0x000003 tal y como se muestra en la Tabla 2-3. En esta misma tabla se han definido dos nuevas secuencias de set ordenado con el propósito de iniciar y finalizar una sesión de encriptación. Su funcionamiento es análogo al del set ordenado /X/ definido en el Apartado 2.3. Por ejemplo, para iniciar la encriptación, el set ordenado 'Cipher ON' ha de ser introducido en el flujo de bloques de 66 bits reemplazando un bloque completo de *idles* antes de que los datos sean encriptados. Una vez que 'Cipher ON' es transmitido se inicia la encriptación. De la misma forma cuando en el receptor dicho set es detectado se activa su generador de *keystream* y se inicia su operación de desencriptación. Finalmente, el bloque que contiene 'Cipher ON' es reemplazado por un bloque completo de *idles* antes de entrar en el decodificador 64b/66b.

Al igual que en [**PER19a**], para llevar a cabo esta manipulación del flujo de bloques 64b/66b es necesario introducir un bloque de control, bloque MANAGEMENT, encargado de introducir y extraer dichos sets ordenados, tal y como se muestra en la Fig.2-18.

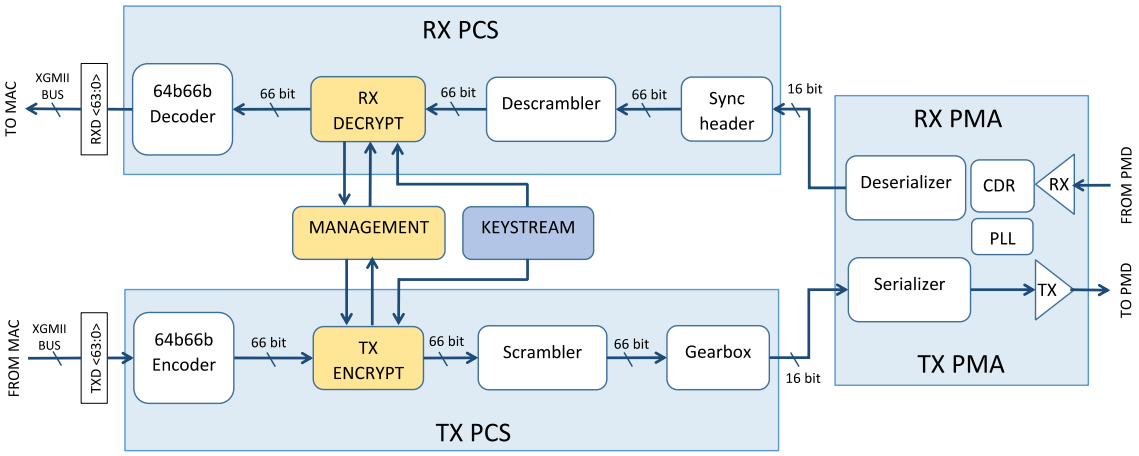



Fig.2-18. Emplazamiento del módulo de encriptación en la capa física 10GBase-R junto al módulo de control.

En el siguiente apartado se presenta la solución aportada para la implementación del generador de *keystream*, donde se han empleado una solución ad-hoc sincronizada.

### 2.4.1 Encriptación por *streaming* ad-hoc para 10G

En la implementación desarrollada en [PER19c] se opta por la utilización de un cifrador por *streaming* basado en el mismo algoritmo caótico que en [PER19a]. Para obtener los 65 bits de anchura del *keystream* mostrado en la Fig.2-6, la estructura del generador consiste por un lado en un banco de cuatro generadores caóticos con una estructura similar a los empleados en [PER19a]. Estos dan lugar a los 64 bits necesarios para la encriptación del *payload* de los bloques 64b/66b. Por otro lado, un generador caótico aislado del que se toma un solo bit se emplea para encriptar la cabecera de sincronismo. El esquema de encriptación basado en dichos generadores pseudoaleatorios se muestra en la Fig.2-19.

Las estructuras del generador caótico básico y del banco de generadores caóticos se muestran en las Fig.2-20 y Fig.2-21, respectivamente. Estos son explicados con más detalle en [PER19c]. Al igual que en [PER19a], Las operaciones del algoritmo *skew-tent map* son llevadas a cabo en la celda STM (STM_CELL). Los 8 bits más bajos de salida $x_i$ de este algoritmo son operados

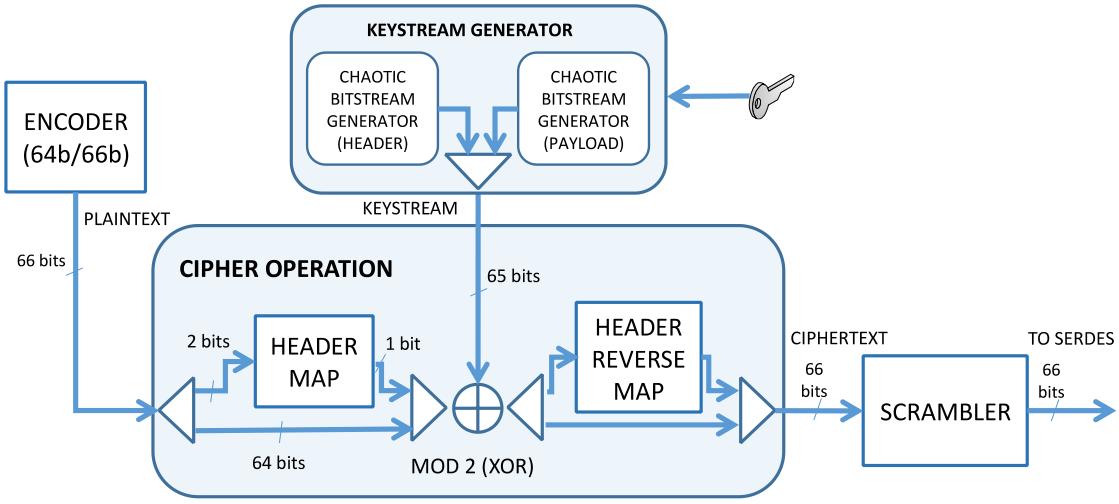



Fig.2-19. Estructura general de cifrado ad-hoc para 10GBase-R.

con la salida de un LFSR y la palabra resultante $\tilde{x}_i$ será realimentada como nueva entrada de la celda STM. En este caso son los 16 bits más bajos, $\tilde{x}_i[15:0]$, los que se toman como salida del generador caótico.

Finalmente, al igual que en el Apartado 2.3.1 es necesario comprobar que la secuencia de *keystream* sea indistinguible de una secuencia verdaderamente aleatoria. Por ello también se llevaron a cabo las pruebas de aleatoriedad recomendadas por el NIST [NIS10], obteniendo también resultados satisfactorios similares a los obtenidos en [**PER19a**].

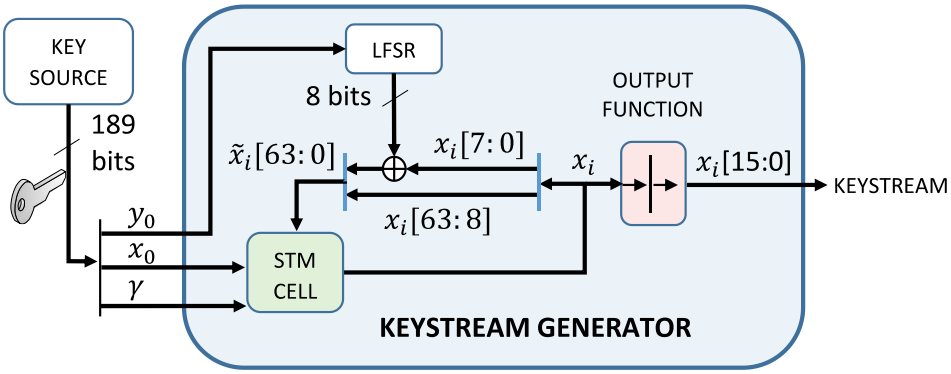



Fig.2-20. Generador caótico básico. En el caso de utilizar el generador caótico básico para encriptar la cabecera se toma como salida sólo un bit: $\tilde{x}_i[0]$.

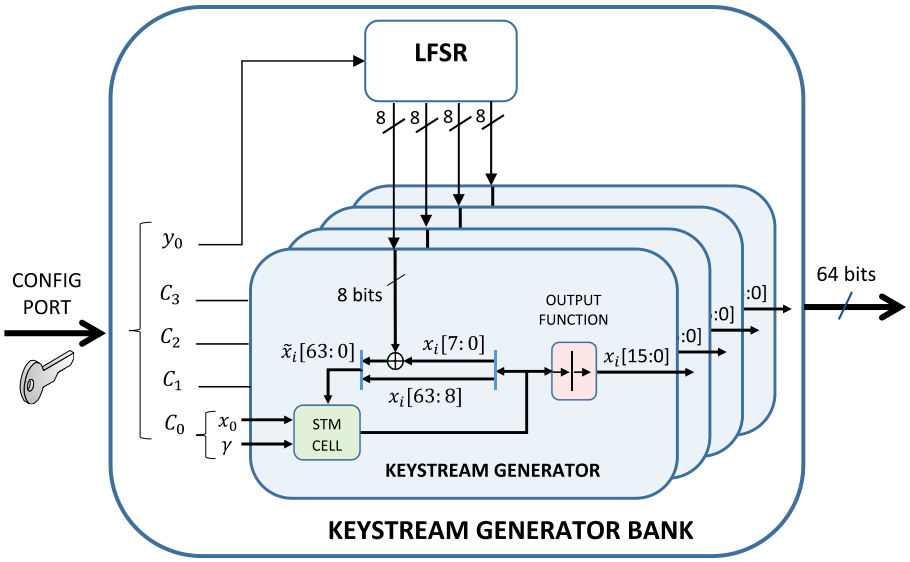



Fig.2-21. Banco de generadores caóticos. La salida de cada módulo caótico está formada por 16 bits, $\tilde{x}_i[15:0]$.

## 2.5 Encriptación por *streaming* autosincronizada

Los cifradores por *streaming* autosincronizados generalmente generan el *keystream* en función de la clave y el *bitstream* cifrado anteriormente. A pesar de sus propiedades de sincronización automática, este tipo de cifradores son menos conocidos y su análisis de seguridad es más complejo que los cifradores por *streaming* sincronizados tradicionales. De hecho, hay pocas propuestas de estos algoritmos. Por ejemplo, solo dos de los cifradores por *streaming* propuestos en el proyecto eSTREAM, SSS y Mosquito, fueron autosincronizados [ROB08]. Además ambos fueron rechazados debido a sus vulnerabilidades [DAE05], [JOU06]. Como se ha comentado en apartados anteriores, los cifradores por *streaming* también se pueden basar en los diferentes modos de operación de los cifradores por bloque, como CTR, CBC, OFB (Output Feedback) o CFB (Cipher Feedback) [NIS01]. Para sincronización automática, CFB es el único modo recomendado por el NIST.

Al igual que el resto de modos, CFB utiliza internamente un cifrador por bloque. Supongamos que $l$ es el tamaño de bloque de dicho cifrador. Para lograr la sincronización después de una pérdida de un número arbitrario de bits en el canal, este modo solo puede retroalimentar un bit de texto cifrado por cada operación del cifrador por bloque. Por esta razón, la tasa de encriptación

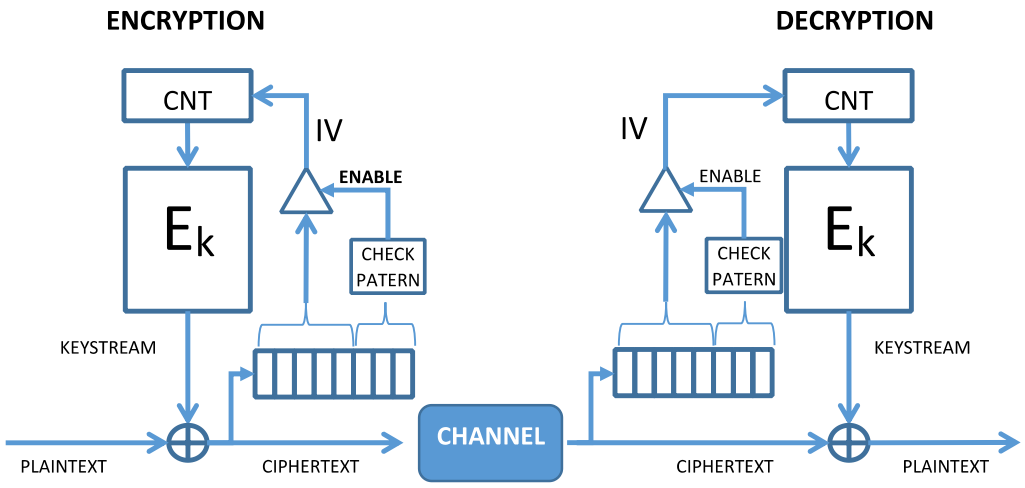



Fig.2-22. Estructura genérica del modo PSCFB en transmisión y recepción.

resultante al operar en modo CFB será de $1/l$ del máximo que podría alcanzar, ya que tan sólo un bit de cada $l$ es aprovechado como *keystream* a la salida del cifrador por bloque.

Para resolver esta limitación en el rendimiento de la encriptación, se propusieron modos de operación como SCFB (Statistical Cipher Feedback) [JUN99] y OCFB (Optimized Cipher Feedback) [ALK02]. Particularmente, SCFB fue analizado en [HEY03], [HEY01], y en [YAN04] se comparó con OCFB, concluyendo que en dicho modo se podrían lograr mejores propiedades para la seguridad en flujos de datos de alta velocidad. A pesar de esta ventaja, se recomienda limitar el uso de SCFB hasta un rendimiento del 50% del máximo alcanzable por el cifrador por bloque empleado en dicho modo. Esta restricción es necesaria para garantizar que no se pierdan bits del texto plano debido al desbordamiento en los *buffers* de entrada/salida de la estructura SCFB. Para superar esta limitación, se propuso el modo PSCFB (Pipelined Statistical Cipher Feedback) [HEY11]. En este modo, gracias a su estructura con *pipeline,* se puede obtener un rendimiento cercano al 100%.

En esta tesis, tanto para la solución de encriptación autosincronizada del estándar 1000Base-X como del 10GBase-R se ha propuesto la utilización y modificación del modo PSCFB ya que su rendimiento en la encriptación es el mayor de los modos autosincronizados propuestos hasta la fecha. La estructura genérica de dicho modo de encriptación se muestra en la Fig.2-22.

El modo PSCFB se puede entender como una combinación de dos modos, CTR y CFB. En la Fig.2-22 el cifrador por bloque, representado por la PRP $E_k$, funciona normalmente en modo CTR mientras el texto cifrado es analizado por el módulo CHECK_PATTERN, que busca de un patrón predefinido de sincronización. En el momento que este patrón es detectado, el cifrador cambia su modo de funcionamiento a CFB, lo que implica que el contador CNT ya no es incrementado como en el modo CTR, sino reinicializado por la secuencia de datos posterior al patrón de sincronismo mencionado. A dicho valor de actualización del contador se le denomina IV (Initialization Vector). Como el patrón de sincronismo es detectado tanto en transmisión como en recepción, los contadores de ambos terminales se sincronizarán periódicamente, ambos con el mismo valor, y por tanto el mismo *keystream* será generado tanto en la encriptación como en la desencriptación.

### 2.5.1 Encriptación por *streaming* con modo PSCFB para 10G

Esta solución fue desarrollada en [PER19d], donde un cifrador por bloque fue utilizado en modo PSCFB para encriptar los bloques 64b/66b. La estructura de cifrado general es como la mostrada en la Fig.2-6, pero sustituyendo el módulo de generación del *keystream* por un cifrador por bloque en modo PSCFB.

Para conocer las propiedades que ha de tener el cifrador por bloque funcionando en dicho modo se analizó el esquema de encriptación con el objeto de obtener la expresión de la ventaja IND-CPA de cualquier posible adversario sobre dicho modo. Modelando el cifrador por bloque como una PRP $E_k$, se pudo obtener en [PER19d] la siguiente ventaja:

$$ADV_{PSCFB}^{IND-CPA}(A) \leq 2 \cdot ADV_{E}^{PRP}(B) + \frac{\mu^2}{l^2 \cdot 2^l} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right) \qquad (10)$$

donde $ADV_{PSCFB}^{IND-CPA}(A)$ es la ventaja del adversario A sobre dicho esquema de encriptación, $ADV_{E}^{PRP}(B)$ es la ventaja de tipo PRP sobre el cifrador por bloque funcionando en dicho modo, $\mu$ es el número de bits encriptado, $l$ es el tamaño de bloque y $P$ el número de etapas internas de la arquitectura *pipeline* del cifrador por bloque $E_k$.

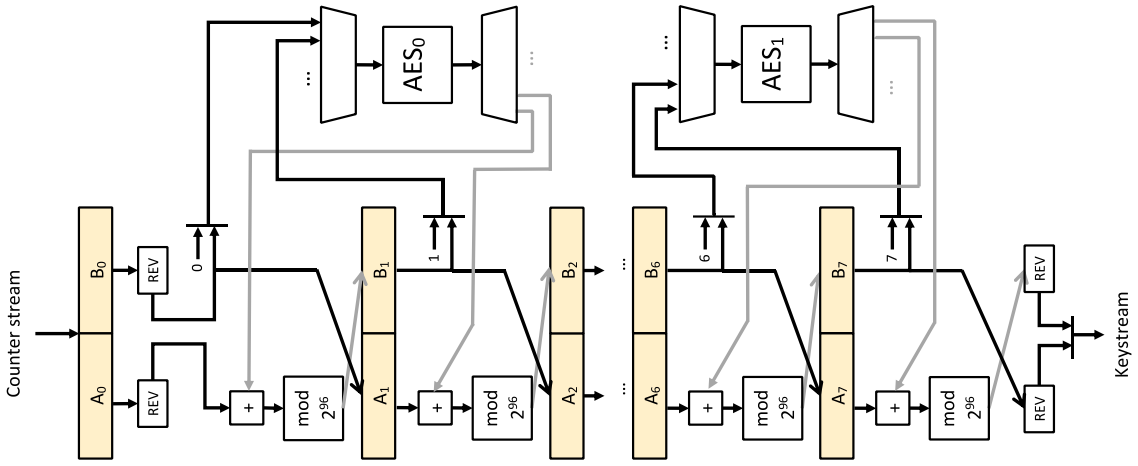



Fig.2-23. Estructura de la implementación del cifrador por bloque FPE FF3 en base binaria para su uso en un esquema PSCFB.

Para garantizar que el cifrador operando en el modo PSCFB logra al menos la misma seguridad que un cifrador tradicional sincronizado, en [PER19d] se comparó la expresión de $ADV_{PSCFB}^{IND-CPA}(A)$ con la expresión de la ventaja de A sobre el modo CTR clásico funcionando con un cifrador con longitud de bloque $l_{CTR}$. De esta forma se extrajo la conclusión de que la longitud por bloque de $E_k$, $l_{PSCFB}$, ha de cumplir la condición mostrada en (11).

$$l_{PSCFB}^2 \cdot 2^{l_{PSCFB}} \geq l_{CTR}^2 \cdot 2^{l_{CTR}} \left(1 + \frac{1}{P} + \frac{1}{P^2}\right) \tag{11}$$

Teniendo en cuenta que en la ecuación anterior el término de la derecha se hace máximo con $P = 1$ y asumiendo que se desea conseguir una mayor seguridad de tipo IND-CPA que un cifrador tradicional, como AES, con $l_{CTR} = 128$ bits y funcionando en modo CTR, entonces se tendrá que cumplir que $l_{PSCFB} \geq 130$ bits.

En [PER19d], para lograr la implementación de un cifrador con tamaño de bloque superior a 128 bits se hizo uso de un cifrador tipo FPE FF3 de base 2 y tamaño de bloque 192 bits, lo que sobrepasa suficientemente el valor recomendado para $l_{PSCFB}$. La estructura propuesta para este cifrador por bloque se muestra en la Fig.2-23. Finalmente, haciendo uso de este cifrador FPE y adaptando el sistema PSCFB de la Fig.2-22 al esquema genérico de encriptación en 10GBase-R de la

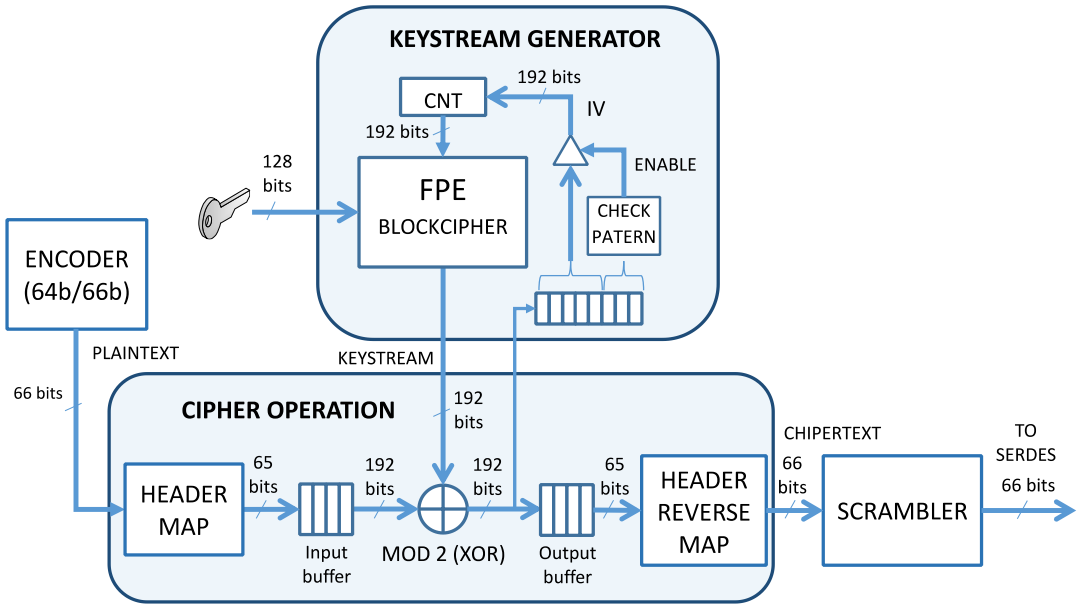



Fig.2-24. Estructura completa del sistema de encriptación por *streaming* empleando un cifrador por bloque funcionando en modo PSCFB para 10GBase-R.

Fig.2-6 se desarrolló el sistema mostrado en Fig.2-24, donde la estructura PSCFB se utiliza internamente para implementar el generador de *keystream*. Respecto a soluciones no autosincronizadas, la diferencia principal en la operación de cifrado es la utilización de un *buffer* de entrada y otro de salida para adaptar los relojes de la estructura PSCFB y el resto del camino de datos. Para lograr una encriptación a máxima tasa de transmisión es necesario que el reloj del modo PSCFB funcione a una frecuencia algo mayor que la empleada en el resto del sistema.

### 2.5.2 Encriptación por *streaming* con modo PSCFB-FPE para 1G

Para lograr la encriptación autosincronizada en 1000Base-X se propuso en [PER19e] una adaptación del modo PSCFB a la solución no autosincronizada implementada en [PER19b]. Suponiendo que trabajamos con un texto plano compuesto por símbolos en base $S$, se podría generalizar la estructura PSCFB para utilizar un cifrador por bloque FPE tal y como se muestra en la Fig.2-25. En este caso el cifrador por bloque está en base $S$ y tiene longitud de bloque de $L$ símbolos. El módulo P/S serializa los $L$ símbolos para después llevar a cabo la operación módulo-S con el texto plano. El texto cifrado es analizado, al igual que

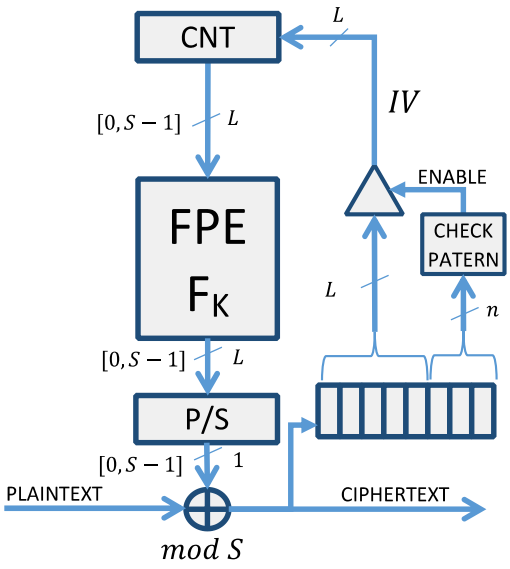



Fig.2-25. Generalización del modo PSCFB para su uso con una base no binaria.

en la Fig.2-22, para buscar el patrón de sincronismo de $n$ símbolos de longitud que indicará cuando se ha de reiniciar el contador CNT con el nuevo IV, ambos en base $S$.

De igual manera que en [PER19d] se puede demostrar que la ventaja IND-CPA de cualquier adversario sobre este esquema de encriptación viene definida por la siguiente expresión:

$$ADV_{PSCFB-FPE}^{IND-CPA}(A) \leq 2 \cdot ADV_{E}^{PRP}(B) + \frac{\mu^2}{l^2 \cdot S^l \cdot T^2} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right) \qquad (12)$$

donde $ADV_{PSCFB-FPE}^{IND-CPA}(A)$ es la ventaja del adversario A sobre dicho esquema de encriptación, $ADV_{E}^{PRP}(B)$ es la ventaja de tipo PRP sobre el cifrador por bloque funcionando en dicho modo, $\mu$ es el número de bits encriptado, $l$ es el tamaño de bloque en símbolos de base $S$, $T$ es el número de bits por símbolo, $T = \log_2 S$, y $P$ el número de etapas internas de la arquitectura pipeline del cifrador por bloque FPE $F_k$.

Para garantizar que el cifrador operando en el modo PSCFB logra al menos la misma seguridad que un cifrador tradicional, en [PER19e] se comparó la expresión de $ADV_{PSCFB-FPE}^{IND-CPA}(A)$ con la expresión de la ventaja de A sobre el modo CTR clásico funcionando con un cifrador con longitud de bloque $l_{CTR}$. De esta

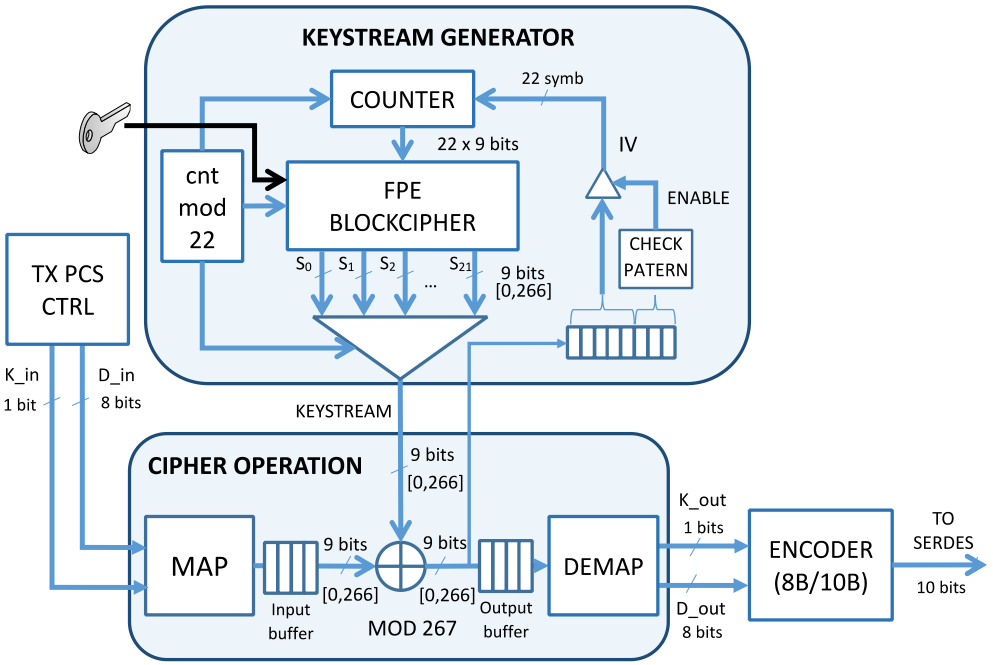



Fig.2-26. Estructura completa del sistema de encriptación por *streaming* empleando un cifrador FPE funcionando en modo PSCFB para 1000Base-X.

forma se extrae la conclusión de que la longitud por bloque de $F_k$, $l_{PSCFB}$, ha de cumplir la condición en (13).

$$l_{PSCFB}^2 \cdot S^{l_{PSCFB}} \geq \frac{l_{CTR}^2 \cdot 2^{l_{CTR}}}{T^2}\left(1+\frac{1}{P}+\frac{1}{P^2}\right) \tag{13}$$

Teniendo en cuenta que el término de la derecha en la ecuación anterior se hace máximo con $P = 1$, que se desea conseguir una mayor seguridad de tipo IND-CPA que un cifrador con $l_{CTR} = 128$ bits funcionando en modo CTR y que en 1000Base-X $S$ = 267, entonces se deberá cumplir que $l_{PSCFB} \geq 17$ símbolos en base 267.

Finalmente, haciendo uso del mismo cifrador FPE que en [PER19b], con una longitud de bloque de 22 símbolos, y adaptando el modo PSCFB de la Fig.2-25 en el esquema de encriptación de 1000Base-X de la Fig.2-2, se implementó la estructura mostrada en la Fig.2-26. Como se puede observar, una de las principales diferencias entre esta estructura y la de la Fig.2-14 estriba en que en la operación de cifrado es necesario la utilización de los *buffers*, ya que al tratarse del modo PSCFB es necesario llevar a cabo la adaptación de tasas entre el generador de *keystream* y el resto del sistema.

### 2.5.3 Encriptación por *streaming* con modo PSCFB-MOD para 1G

Con el objetivo de lograr la encriptación autosincronizada para 1000Base-X intentando reducir los recursos *hardware* empleados en [PER19e] se propuso adaptar el modo PSCFB a la solución sin autosincronización implementada en [PER20a].

En el apartado anterior hemos generalizado el modo PSCFB de forma que el cifrador por bloque subyacente no tenga por qué estar en base binaria, sino una base cualquiera $S$. En dicho caso se ha considerado que el cifrador por bloque se puede emular como una PRP con una determinada anchura de bloque $l_{PSCFB}$. Al adaptar el modo PSCFB a la solución [PER20a] se podría considerar el análisis del bloque del cifrador junto a la operación módulo como una PRF de diferentes anchuras de datos en su entrada y salida. En [PER20b] se llevó a cabo esta generalización, realizando el análisis de seguridad de esta nueva estructura PSCFB.

En la Fig.2-27, se muestran los dos esquemas implementados en esta tesis para autosincronización en 1000Base-X. A la izquierda se muestra la estructura estudiada en [PER19e], cuyo cifrador por bloque es un cifrador FPE en base $S$ = 267 y anchura $L$ símbolos con $L$ = 22. A la derecha se muestra la adaptación a PSCFB de un cifrador por bloque binario junto a una operación módulo $S$. En [PER20b] ambos bloques representan una PRF denominada $F_k\_MOD$, al igual que en la figura Fig.2-16. El modo resultante ha sido denominado PSCFB-MOD.

Al igual que en anteriores soluciones, para la estructura de la derecha en la Fig.2-27 se llevó a cabo el análisis de la ventaja IND-CPA. La expresión de dicha ventaja se muestra en (14).

$$ADV^{IND-CPA}_{PSCFB-MOD}(A) \leq 2 \cdot ADV^{PRP}_{E}(B) + \frac{\mu}{T \cdot 2^{l-1}} + \frac{\mu^2}{T^2} \cdot \left( \frac{1}{2^l} + \frac{P_{max_{r_i}}}{P - 1 + L_{IV-SO}} \right) \quad (14)$$

con

$$P_{max_{r_i}} \leq 1/2^l + 1/S^{L_{IV-so}} \quad (15)$$

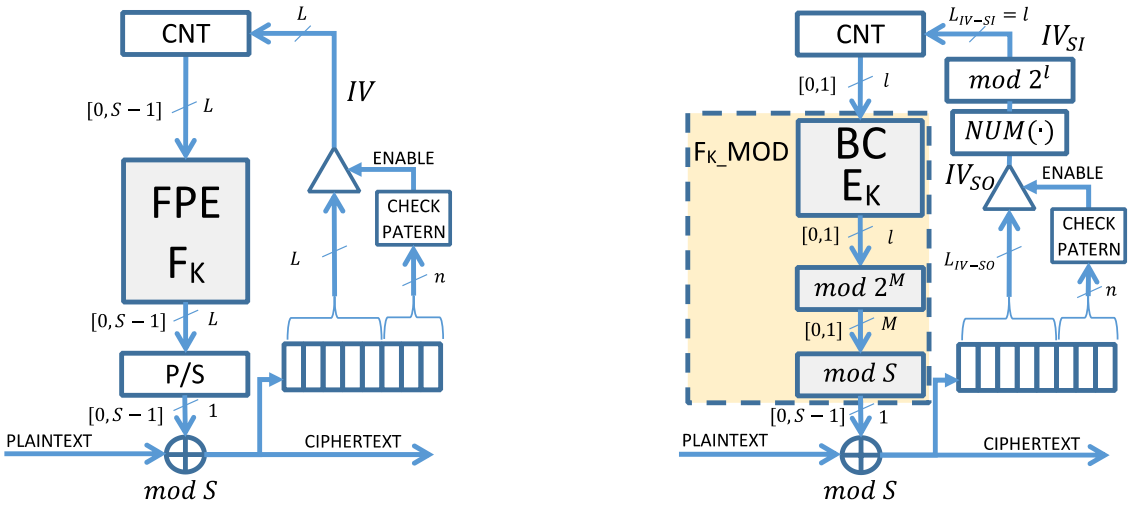



Fig.2-27. Esquemas implementados de tipo PSCFB para encriptación en 1000Base-X, (a) modo PSCFB-FPE, (b) modo PSCFB-MOD.

donde $ADV_{PSCFB-MOD}^{IND-CPA}(\mathrm{A})$ es la ventaja del adversario A sobre dicho modo, $ADV_{E}^{PRP}(B)$ es la ventaja de tipo PRP sobre $E_k$, $\mu$ es el número de bits encriptado, $T$ son los bits de información por cada símbolo, $l$ es el tamaño de bloque de $E_k$, $L_{IV-SO}$ es el tamaño del IV en número de símbolos e $I$ es la diferencia entre los bits de entrada a la operación modulo-S y $T$. Por otro lado, el término $P$ hace referencia al número de etapas total de procesado implementadas en la estructura $F_k\_MOD$, incluyendo las etapas del cifrador por bloque $E_k$ más las de la operación módulo, como si se trataran en conjunto de una única estructura en *pipeline*.

Para garantizar que el cifrador $E_k$ operando en este modo PSCFB logra al menos la misma seguridad que un cifrador tradicional sin autosincronización, en [PER20b] se compara la expresión de $ADV_{PSCFB-MOD}^{IND-CPA}(A)$ con la expresión de la ventaja de A sobre el modo CTR clásico con una longitud de bloque $l_{CTR}$. De esta forma se puede extraer alguna conclusión de cómo han de ser los parámetros del sistema propuesto. Concretamente en [PER20b] se llegó a la relación mostrada en (16) entre los parámetros mencionados anteriormente en la ecuación (14) y $l_{CTR}$.

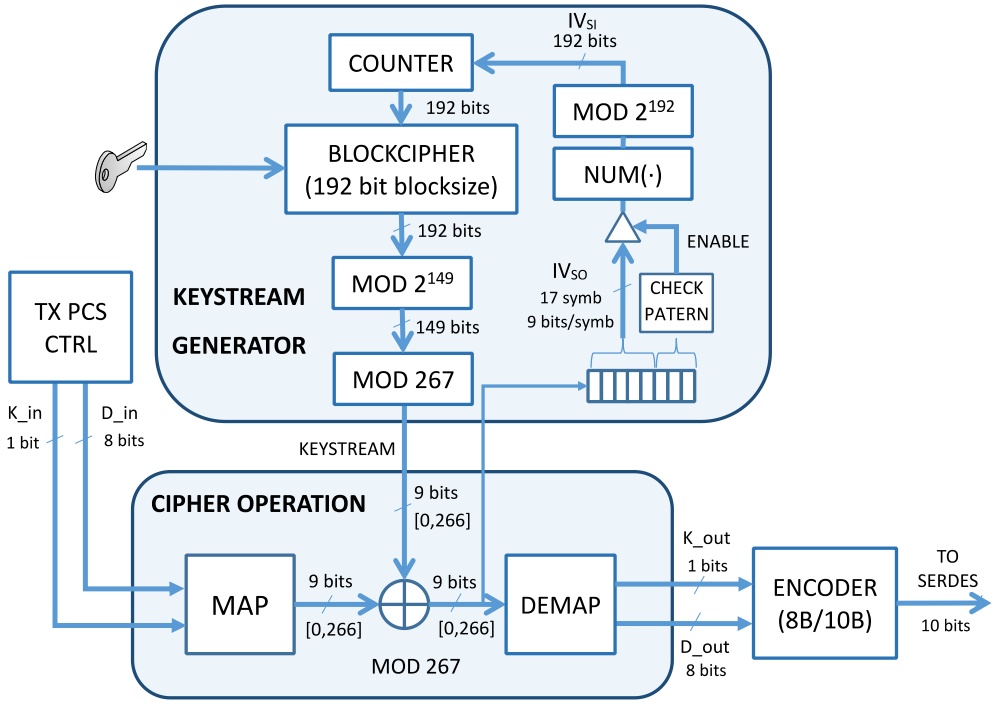



Fig.2-28. Estructura completa del sistema de encriptación por *streaming* empleando un cifrador funcionando en modo PSCFB-MOD para 1000Base-X.

$$\frac{1/2^{l}+1/S^{L_{IV-SO}}}{L_{IV-SO}} \leq \frac{T^{2}}{L_{CTR}^{2}\cdot 2^{L_{CTR}}}-\frac{T}{2^{I-1}}-\frac{1}{2^{l}} \tag{16}$$

Suponiendo que la estructura formada por el cifrador por bloque y la operación módulo-S es la misma que en [PER19b] se cumple que: $l = 192$, $S = 267$, $\mathrm{T} = \log_2 S \cong 8.06$, $I = M - T$ y $M = 149$. Lo que implica que para cumplir la condición en (16) $L_{IV-SO} \geq 17$. Es decir, el tamaño del IV en dicho esquema ha de ser de al menos 17 símbolos en base $S$.

Finalmente, de acuerdo con los valores de los parámetros citados, se llevó a cabo la implementación de la estructura mostrada en la Fig.2-28.

## 2.6 Implementación sobre FPGA

Para llevar a cabo el testeo de cada una de las soluciones de encriptación propuestas en esta tesis, estas han sido integradas en interfaces Ethernet e implementadas sobre una FPGA (Field Programmable Gate Array) Virtex 7 del

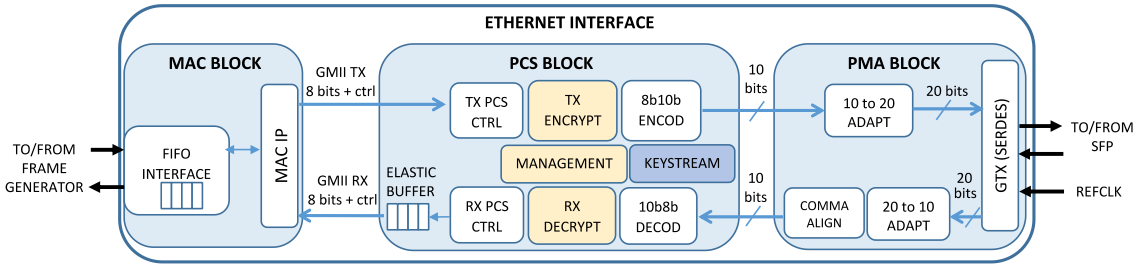



Fig.2-29. Interfaz Ethernet integrando el sistema de encriptación para 1000Base-X.

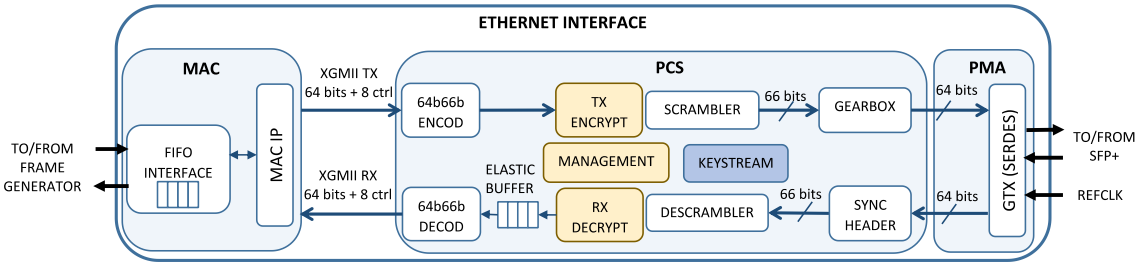



Fig.2-30. Interfaz Ethernet integrando el sistema de encriptación para 10GBase-R.

fabricante Xilinx. Los interfaces se componen de una capa física que incluye el sistema de encriptación junto a una capa MAC conectada a la misma, tal y como se muestra para cada uno de los estándares 1000Base-X y 10GBase-R en las Fig.2-29 y Fig.2-30, respectivamente.

En el *set-up* de test de cada uno de los estándares se implementaron dos interfaces. Por el lado de la MAC los interfaces fueron conectados a dos generadores de tramas Ethernet internos en la FPGA, mientras que por el lado de la capa física estos se conectaron a dos módulos ópticos externos SFP (Small Form Factor Pluggable) a su vez enlazados entre si gracias a un enlace de fibra óptica multimodo. El esquema del *set-up* de test y una fotografía del mismo se muestran en la Fig.2-31 y Fig.2-32, respectivamente. Gracias a los módulos generadores de tramas se pudo verificar el enlace encriptado utilizando tráfico real de datos. Este se configuró mediante ráfagas de tramas Ethernet de diferentes longitudes y espaciado IFG (Inter Frame Gap) entre ellas. Se pudo comprobar que el sistema de encriptación funcionaba sin interferir negativamente en la transferencia de paquetes, ya que en la recepción no se produjeron pérdidas de tramas ni errores en el CRC (Cyclic Redundancy Check) de las mismas.

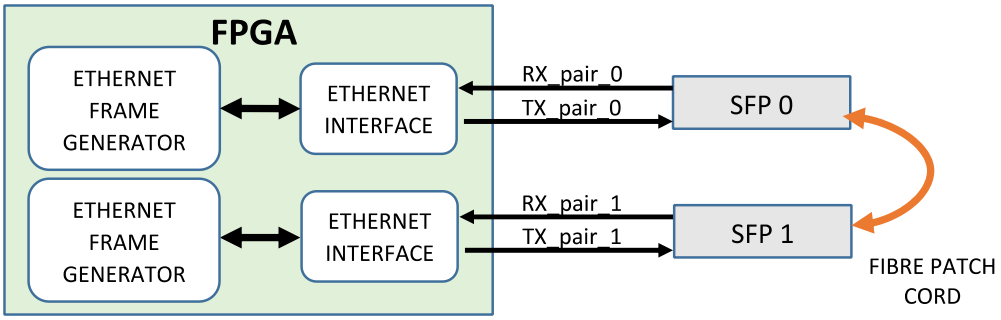



Fig.2-31. Esquema del *set-up* de test. Los interfaces Ethernet son conectados por el lado de la MAC a generadores de tramas, mientras que por el lado de la capa física se conectan a los módulos ópticos SFP.

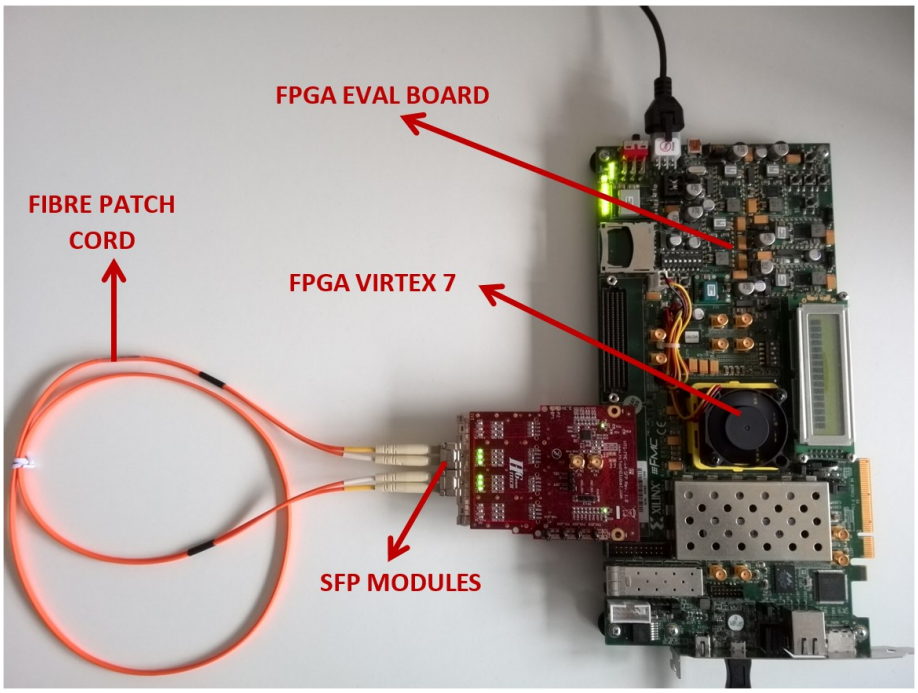



Fig.2-32. Foto del *set-up* de test. La FPGA forma parte de la placa de evaluación VC707 del fabricante Xilinx. Los módulos SFP son insertados en una placa de expansión conectada a la VC707 del fabricante Hitech Global.

En la Tabla 2-4 se observan los recursos *hardware* empleados en la implementación de cada una de las soluciones propuestas. Junto a estos se muestran también los valores de la latencia introducida en el camino de datos de la capa física debido a la operación de encriptación. Se puede comprobar que la latencia es la misma en las soluciones [PER19a], [PER19b], [PER20a] y [PER20b], ya que todas ellas introducen únicamente el retardo de la operación suma/resta modulo-267 y el mapeo/demapeo de los símbolos 8b/10b. En el caso de [PER19c] el retardo es debido a la operación XOR y el mapeo/demapeo de

| | SC-1G [PER19a] | CTR-FPE [PER19b] | CTR-MOD [PER20a] | SC-10G [PER19c] | PSCFB 10G [PER19d] | PSCFB FPE [PER19e] | PSCFB MOD [PER20b] |
|---|---|---|---|---|---|---|---|
| Registers | 6097 | 11127 | 8807 | 2271 | 19154 | 24979 | 10317 |
| LUTs | 13391 | 16978 | 10974 | 4680 | 17599 | 26455 | 12261 |
| 18K Block RAMs | 0 | 77 | 78 | 0 | 153 | 77 | 78 |
| DSP cells | 144 | 0 | 0 | 80 | 0 | 0 | 0 |
| Slices | 4110 | 5636 | 3844 | 1454 | 6794 | 9737 | 4355 |
| Encryption Rate (Mbps) | 1000 | 1000 | 1000 | 10000 | 10000 | 1000 | 1000 |
| Encryption Rate/Slice (Kbps/Slice) | 243.3 | 177.4 | 260.1 | 6878 | 1471.8 | 102.7 | 229.6 |
| Latency (ns) | 48 | 48 | 48 | 38.4 | 266 | 648 | 48 |

Tabla 2-4. Resumen de los recursos *hardware* empleados en la implementación de cada solución.

las cabeceras 64b/66b. Las soluciones [PER19d] y [PER19e], aparte del retardo de la operación de encriptación, introducen también el retardo debido a los *buffers* de adaptación de tasas para PSCFB, lo que incrementa sustancialmente la latencia respecto al resto de soluciones.

## 2.7 Análisis de la seguridad de las soluciones propuestas

Para realizar un análisis completo de la seguridad de las soluciones propuestas se ha de tener en cuenta tanto la seguridad de los cifradores por bloque o generadores de *keystream* subyacentes a los modos de encriptación, como la seguridad de tipo IND-CPA de cada uno de los modos propuestos.

En las soluciones estudiadas en [PER19a] y [PER19c] los generadores de *keystream* están construidos a partir de PRNG (Pseudo Random Number Generator) caóticos. Su seguridad fue sucintamente discutida en ambas propuestas. Por un lado la longitud de clave podría considerarse adecuada ya que supera los 128 bits, suficientes según la recomendación de ENISA (European Union Agency for Network and Information Security) respecto a la seguridad a medio o largo plazo de un criptosistema genérico. Por otro lado,

otras consideraciones como la sensibilidad o la posibilidad de ataques de reconstrucción de los mapas caóticos fueron tratadas tanto en [PER19a] como en [PER19c]. Sin embargo, dichas estructuras o PRNGs carecen de un criptoanálisis riguroso, lo que limita su seguridad.

En el resto de propuestas, [PER19b], [PER20a], [PER19d], [PER19e] y [PER20b] se optó por la utilización de cifradores por bloque ya diseñados que no tienen la carencia mencionada anteriormente y cuya seguridad es suficientemente reconocida, como AES, Rijndael o las estructuras FPE recomendadas por el NIST.

En cuanto a la seguridad de los modos propuestos, esta se midió en términos de la ventaja IND-CPA, tal y como se mide en los modos tradicionales como CTR o CBC. Una vez obtenida la expresión de la ventaja IND-CPA sobre cada modo, se pudo obtener un límite para ciertos parámetros dentro de la estructura de cada uno de ellos y así garantizar una seguridad igual o mayor que el modo tradicional tomado como referencia. En esta tesis, puesto que de los modos recomendados por el NIST el CTR es considerado el mejor en términos de ventaja IND-CPA, este ha sido el tomado como referencia.

En la Tabla 2-5 se muestra un resumen de las ventajas de cada modo y el valor necesario que han de tomar ciertos parámetros de su estructura para que logren al menos una seguridad de tipo IND-CPA mejor que el modo de referencia, es decir, CTR con 128 bits de tamaño de bloque.

Otro de los aspectos de la seguridad que se ha de remarcar es que las soluciones propuestas, además de conseguir la confidencialidad de la información, también son capaces de lograr la privacidad, entendida esta como la habilidad de ofuscar la presencia de las comunicaciones. En el caso del estándar 1000Base-X, al llevarse a cabo la encriptación de todo el flujo de símbolos 8b/10b incluyendo tanto los símbolos de datos como de control, no sólo la información de los paquetes de datos es encriptada, sino también los caracteres que sirven para señalización del inicio/fin de los mismos o de inactividad de la transmisión en el enlace. Concretamente, los símbolos 8b/10b se componen de un valor de 8 bits, junto con un bit de control, llamado *flag* K, que indica si el valor del símbolo es de datos o de control dependiendo de si este vale ‘0’ o ‘1’. Tanto el bus de datos como el *flag* K representan al símbolo 8b/10b que entra en el codificador 8b/10b

| Encryption Mode $\mathcal{SE}(E)$ | IND-CPA Advantage expression $ADV^{IND-CPA}_{\mathcal{SE}(E)}(A)$ | Security condition |
|---|---|---|
| SC-1G [PER19a] | $2 \cdot ADV_G^{PRG}(B) + \frac{\mu}{T \cdot 2^I}$ | $l \geq 148$ bits |
| CTR-FPE [PER19b] | $2 \cdot ADV_F^{PRP}(B) + \frac{\mu^2}{T^2 \cdot l^2 \cdot S^l}$ | $l \geq 16$ symbols |
| CTR-MOD [PER20a] | $2 \cdot ADV_E^{PRP}(B) + \frac{\mu^2}{T^2 \cdot 2^l} + \frac{\mu}{T \cdot 2^{I-1}}$ | $l \geq 149$ bits |
| SC-10G [PER19c] | $2 \cdot ADV_G^{PRG}(B)$ | |
| PSCFB-FPE [PER19e] | $2 \cdot ADV_E^{PRP}(B) + \frac{\mu^2}{l^2 \cdot S^l \cdot T^2} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right)$ | $l \geq 17$ symbols |
| PSCFB-MOD [PER20b] | $2 \cdot ADV_E^{PRP}(B) + \frac{\mu}{T \cdot 2^{I-1}} + \frac{\mu^2}{T^2} \cdot \left(\frac{1}{2^l} + \frac{P_{max_{r_i}}}{P - 1 + L_{IV-SO}}\right)$ | $L_{IV-SO} \geq 17$ symbols<br>$l = 192$ bits |
| PSCFB-10G [PER19d] | $2 \cdot ADV_E^{PRP}(B) + \frac{\mu^2}{l^2 \cdot 2^l} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right)$ | $l \geq 130$ bits |
| CTR | $2 \cdot ADV_E^{PRP}(B) + \frac{\mu^2}{l^2 \cdot 2^l}$ | $l = 128$ bits |

Tabla 2-5. Comparativa de la ventaja IND-CPA entre las diferentes soluciones. El parámetro $l$ es la longitud de bloque y $L_{IV-SO}$ el tamaño de IV en el modo PSCFB-MOD. La expresión de las ventajas IND-CPA en los casos de [PER19a] y [PER19c] ha sido calculada posteriormente al desarrollo de dichos trabajos y su razonamiento no está incluido en esta tesis. El término $ADV_G^{PRG}$ se refiere a la ventaja de tipo PRG (Pseudo Random Generator) de un adversario cualquiera sobre el generador de *keystream* ad-hoc $G$ empleado en el esquema de encriptación correspondiente.

para ser transformado en un valor de 10 bits. Durante la transmisión del *payload* de un paquete Ethernet el *flag* K permance a cero indicando que los bytes transmitidos son datos, sin embargo cuando no se transmiten paquetes, el enlace es mantenido mediante la transmisión de sets ordenados de tipo *idle*. Dichos sets están formados por dos símbolos consecutivos, el símbolo de control /K28.5/ (con *flag* K de valor 1) y un símbolo de datos (con *flag* K de valor 0). Esto hace que en un estado de *idle* o inactividad se produzcan transiciones continuas entre 1 y 0 en el *flag* K. En la Fig.2-33 se puede observar el patrón o forma que toma el *flag* K antes de la encriptación cuando el enlace se encuentra sin transmitir tramas Ethernet y cuando se encuentra transmitiendo una ráfaga de paquetes. En el momento en el que activamos la encriptación, los símbolos son completamente aleatorizados deshaciendo el patrón del *flag* K independientemente de que haya o no haya transmisión de datos en el enlace,

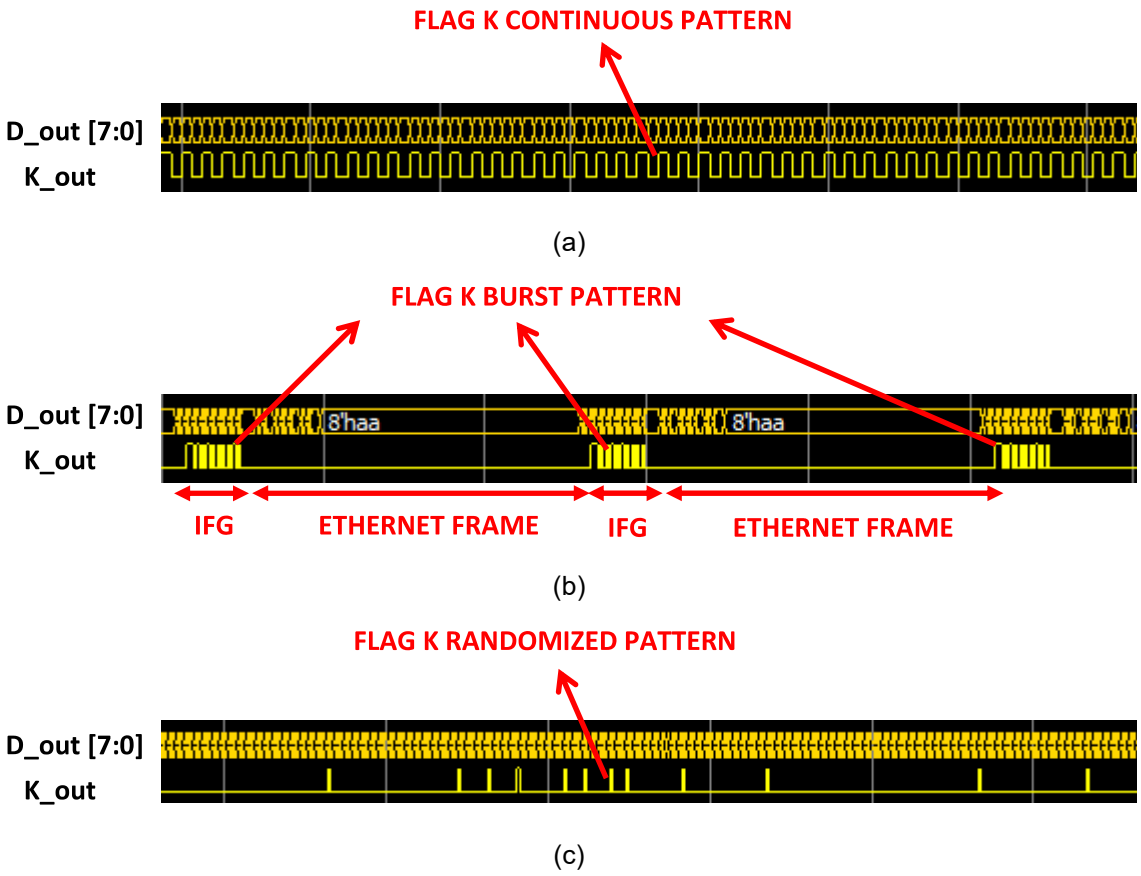



Fig.2-33. (a) Patrón del flag K sin encriptación cuando ninguna trama Ethernet es transmitida; (b) patrón del flag K sin encriptación cuando se transmite una ráfaga de tramas Ethernet, (c) patrón del flag K después de la encriptación independientemente de que haya o no transmisión de tramas Ethernet.

lo que hace indistinguible la presencia de la propia transmisión respecto a una situación de inactividad.

Esta monitorización del *flag* K fue llevada a cabo tanto en [PER19a] como en [PER19b]. Además en [PER19b] se midió matemáticamente la entropía de los símbolos 8b/10b con un enlace transmitiendo diferentes patrones de tráfico, incluyendo el caso en el que no hubiera transmisión de datos, es decir, solo *idles*.

La ecuación utilizada para el cálculo de la entropía se muestra en (17). Esta fue calculada agrupando los símbolos 8b/10b en tuplas $\beta_n$ de $n$ símbolos en base $S$ cada una y midiendo la probabilidad de aparición de cada posible valor de las mismas, $P(\beta_n)$. Se obtuvieron los valores para la entropía en los casos con $n$ igual a 1, 2 y 3 símbolos.

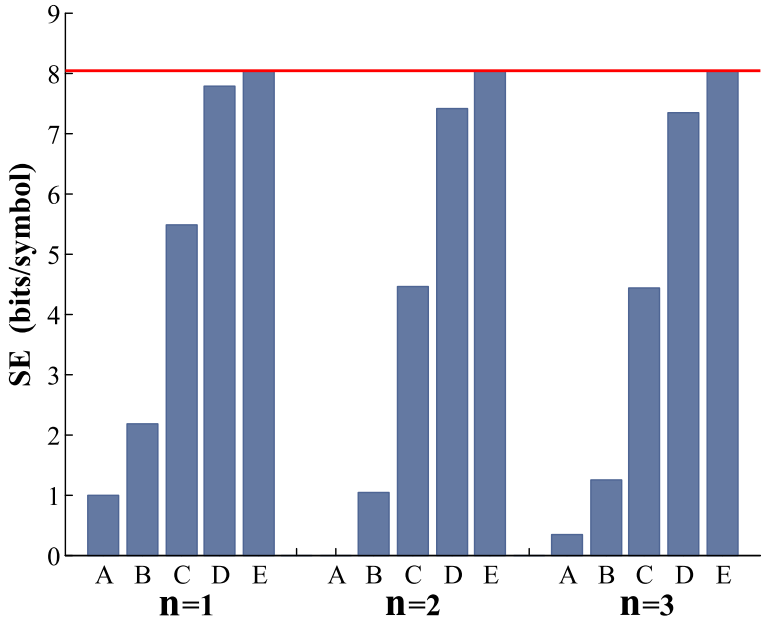



Fig.2-34. Entropía obtenida agrupando los símbolos 8b/10b en tuplas de 1, 2 y 3 símbolos. Los patrones en los que se midió fueron denominados A, B, C, D y E.

$$SE = -\frac{1}{n} \cdot \sum_{\beta_n < S^n} P(\beta_n) \cdot \log_2 P(\beta_n) \tag{17}$$

Con esta medida se obtuvo una gráfica como la mostrada en la Fig.2-34. En dicha figura, se muestra la entropía de cinco patrones: A, B, C, D y E. El patrón A corresponde con el caso de no transmisión de tramas, donde sólo los *idles* son transmitidos en el enlace. Los patrones B, C y D coinciden con una transmisión continua de tramas de longitud 1024 bytes y *payload* pseudoaleatorio con tasas de 10.2%, 50% y 91% del máximo alcanzable de la tasa de línea en 1000Base-X, respectivamente. El patrón E corresponde con la señal ya aleatorizada después de encriptar el patrón A, el peor en términos de entropía. Se observa claramente como el patrón E encriptado obtuvo la máxima entropía posible $SE = \log_2 S = \log_2 267 \cong 8.0606$, ya que en dicho caso $P(\beta_n)$ es igual para todas las $\beta_n$.

En el caso del estándar 10GBase-R la encriptación se llevó a cabo en todo el flujo de bloques 64b/66b. El bus de datos 64b/66b está formado por la cabecera de dos bits de sincronismo y el bloque de 64 bits de *payload*. Sin encriptación habilitada, la cabecera tomará valores de '10' cuando las tramas están transmitiéndose y '01' mientras no hay transmisión, por ejemplo durante el IFG

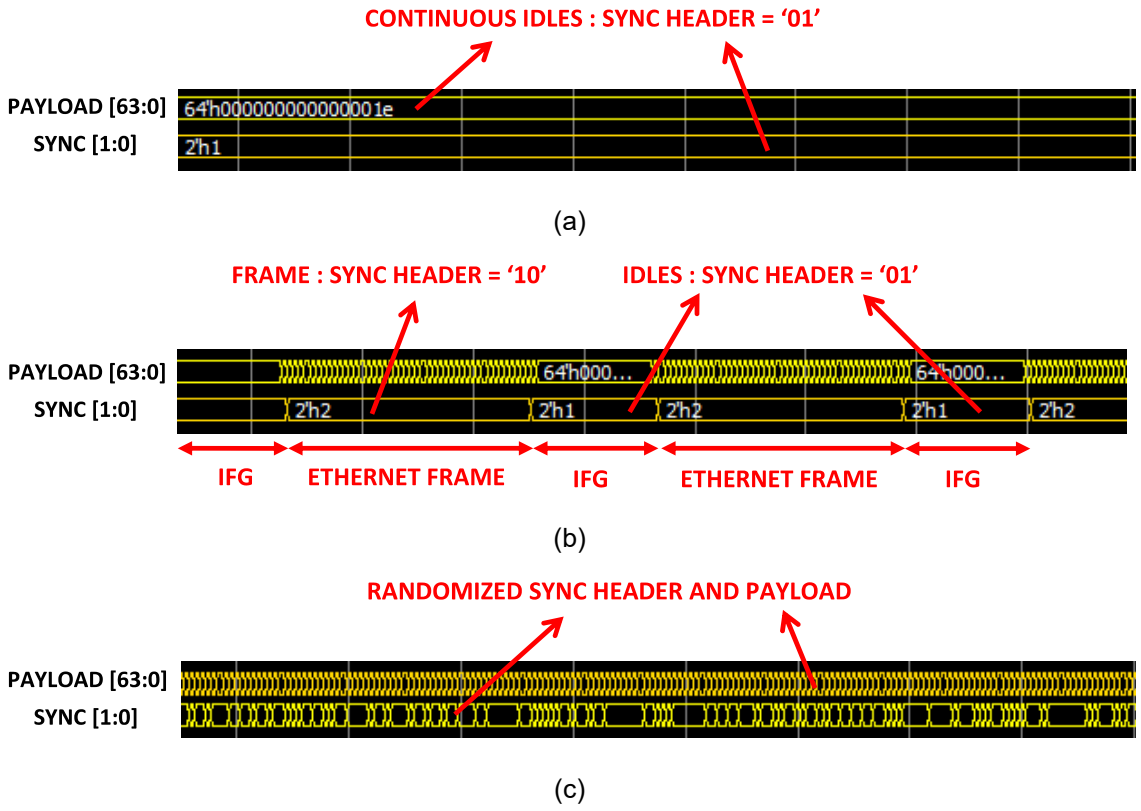



Fig.2-35. (a) Patrón de cabecera de sincronismo sin encriptación cuando no hay tramas Ethernet transmitidas; (b) patrón de cabecera de sincronismo sin encriptación cuando se transmiten paquetes Ethernet; (c) patrón de cabecera de sincronismo después de la encriptación independientemente de que se transmitan o no tramas Ethernet.

entre tramas. Sin embargo, cuando se activa la encriptación, a la entrada del *scrambler* se puede observar una cabecera que cambia de forma aleatoria entre los valores '01' y '10' y de la misma manera un valor de *payload* completamente aleatorizado. Esto ocurre independientemente de que se estén transmitiendo o no tramas Ethernet durante la encriptación. Estos tres casos fueron monitorizados tanto en [PER19c] como en [PER19d], y se pueden apreciar en la Fig.2-35.

Además en [PER19d] se realizó el cálculo matemático de la entropía tanto de la cabecera de sincronismo ya mapeada como del *payload* en diferentes patrones de tráfico, ambos con y sin encriptación. La expresión utilizada se muestra en (18).

$$SE = -\frac{1}{n} \cdot \sum_{\beta_n < 2^n} P(\beta_n) \cdot \log_2 P(\beta_n) \tag{18}$$

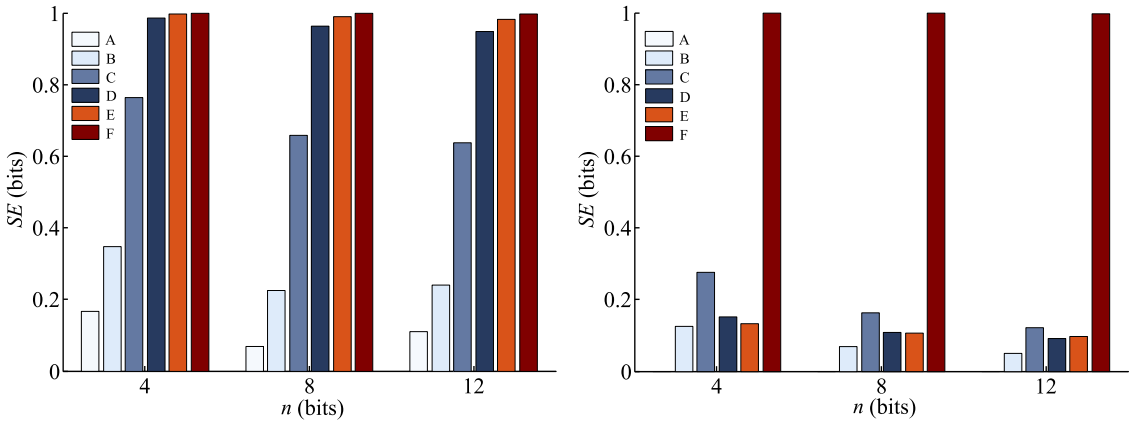



Fig.2-36. (a) Entropía de los *payloads* de los bloques 64b/66b con $n$ igual a 4, 8 y 12 en cada uno de los patrones de tráfico Ethernet; (b) Entropía de las cabeceras de sincronismo medidas con $n$ igual a 4, 8 y 12. Los patrones 64b/66b se han denominado A, B, C, D, E y F.

Ambos *bitstreams*, cabecera y *payload*, fueron agrupados en tuplas $\beta_n$ de longitud $n$ bits. Las entropías fueron medidas con $n$ igual a 4, 8 y 12 bits.

En la figura Fig.2-36 se muestran los resultados obtenidos en diferentes patrones de tráfico A, B, C, D, E y F. A se corresponde con el caso de no transmisión de tramas, es decir, donde sólo se envían bloques 64b/66b de control llenos de *idles*. Los casos B, C y D se corresponden con una transmisión continua de tramas de 1024 bytes con *payloads* pseudoaleatorios y tasas de 10.2%, 50% y 98% del máximo del ancho de banda de línea, respectivamente. El patrón E es igual que los tres anteriores pero empleando tramas de longitud aleatoria. Finalmente, el patrón F es el resultante de encriptar el A, que es el peor de todos ellos en términos de entropía. En este último caso se observa que gracias a la encriptación la entropía obtenida es máxima, $SE = \log_2 S = \log_2 2 = 1$.

## 2.8 Mecanismo de autenticación, integridad y refresco de claves

En cualquier sistema criptográfico, la confidencialidad y la privacidad no son las únicas características de seguridad que deben cumplirse. Desde una perspectiva de seguridad completa, se deben abordar algunos problemas como la integridad, la autenticidad, el refresco de los datos y otros aspectos, como la actualización

de claves. En [PER20b] se implementó un protocolo ligero, basado en pequeños paquetes formados por símbolos 8b/10b, capaz de desempeñar las funcionalidades mencionadas en un enlace de comunicaciones que emplee el estándar 1000Base-X.

La generación y configuración inicial de la clave maestra está fuera del alcance de este trabajo. Como en otros mecanismos de cifrado, esta tarea puede llevarse a cabo inicialmente por las capas superiores de la comunicación, como por ejemplo en el estándar MACsec con protocolos como IEEE 802.1X. Después de un establecimiento inicial de la clave maestra, $K_M$, en los dos terminales de la comunicación, se pueden generar diferentes claves de sesión, $K_S$, derivadas de dicha clave que servirán como claves del sistema al comienzo de cada sesión de encriptación. Esto implica que cada terminal, partiendo de su clave maestra inicial $K_M$, un índice $idx$ que identifica la sesión y una función $f(\cdot)$, será capaz de generar nuevas claves de sesión de la forma $K_S(idx) = f(K_S(idx\text{-}1))$ con $K_S(0) = K_M$, sin necesidad de llevar a cabo un intercambio de claves entre ambos.

A partir de $K_S(idx)$ los terminales calcularán tres claves distintas empleadas a lo largo de la sesión: clave de cifrado, $K_{CIPH}(idx)$, clave para el chequeo de integridad y autenticación de las tramas del protocolo, $K_{FRAME}(idx)$, y clave para el chequeo de la integridad y autenticación del flujo de símbolos en el enlace, $K_{ICM}(idx)$.

$K_{CIPH}$ se utiliza como clave de encriptación/desencriptación de los datos, es decir, se corresponde con la clave de 128 bits que configura los sistemas de cifrado descritos a lo largo de esta tesis. $K_{FRAME}$ es utilizada para realizar el cálculo del campo MAC (Message Authentication Code) necesario para verificar la integridad y autenticidad de cada paquete del protocolo de control. Finalmente, $K_{ICM}$ también se utiliza en el cálculo de un campo MAC, pero en este caso para verificar la integridad y autenticidad de todo el enlace, es decir de todo el flujo de símbolos 8b/10b que está siendo encriptado.

En este trabajo se han establecido unos roles de funcionamiento maestro/esclavo entre los terminales de la comunicación. El maestro es responsable de iniciar y finalizar la sesión de cifrado, que siempre se establece de forma bidireccional, lo que significa que cuando la sesión de cifrado comienza,

| | | | | |
|---|---|---|---|---|
| SCM | HEADER (2B) | STATE (1B) | COUNTER (5B) | MAC_F (8B) |
| $ICM_0$ | HEADER (2B) | COUNTER (6B) | | MAC_F (8B) |
| $ICM_1$ | HEADER (2B) | COUNTER (6B) | | MAC_I (8B) |
| $ICM_2$ | HEADER (2B) | COUNTER (6B) | | MAC_FI (8B) |
| SEM | HEADER (2B) | KEY_INDEX (2B) | COUNTER (4B) | MAC_F (8B) |
| SDM | HEADER (2B) | KEY_INDEX (2B) | COUNTER (4B) | MAC_F (8B) |
| EEM | HEADER (2B) | COUNTER (6B) | | MAC_F (8B) |
| EDM | HEADER (2B) | COUNTER (6B) | | MAC_F (8B) |

Fig.2-37. Estructura de los diferentes mensajes utilizados en el mecanismo de control de encriptación. El tamaño en bytes de cada campo de mensaje se indica entre paréntesis.

ambas direcciones de la comunicación, maestro a esclavo y esclavo a maestro, son encriptadas. Una vez que se establece la sesión de cifrado, la integridad y autenticidad de los datos se verifican periódicamente hasta que se finaliza la misma. Además, el maestro es responsable de realizar la actualización periódica de la clave mientras el enlace permanezca encriptado.

En la Fig.2-37 se muestra la estructura de los mensajes del protocolo de control propuesto para las tareas descritas. Su longitud es igual en todos ellos, de 16 bytes. Los dos primeros bytes contienen un encabezado de dos símbolos 8b/10b, uno de control y otro de datos, que indica el tipo de mensaje en cuestión.

En general, los mensjaes se clasifican en dos grupos según su funcionalidad. Por un lado, mensajes no periódicos, formados por SEM (Start Encryption Message), SDM (Start Decryption Message), EEM (End Encryption Message) y EDM (End Decryption Message), utilizados para controlar el inicio y el final de la sesión de cifrado. Por otro lado, mensajes periódicos, SCM (State Check Message) e ICM (Integrity Check Message), utilizados para refrescar la información del estado de cada terminal en el terminal remoto correspondiente y para verificar la integridad y autenticidad del enlace, respectivamente.

### 2.8.1 Establecimiento de la sesión de encriptación

El establecimiento de una sesión de cifrado comienza cuando el maestro envía un mensaje SEM al esclavo. En este mensaje, el campo KEY_INDEX, mostrado

en la Fig.2-37, representa al índice $idx$ a partir del cual se determina la clave de sesión $K_S(idx)$ y el resto de claves. Cuando el SEM es recibido en el esclavo, este responde con un SDM, que contiene el mismo índice de clave, hacia el maestro, activando inmediatamente después la encriptación en la dirección esclavo-maestro y utilizando como clave de cifrado el valor de $K_{CIPH}$ correspondiente al índice utilizado en KEY_INDEX. Por otro lado, el esclavo también comienza la transmisión periódica de mensajes SCM e ICM. Con el SCM actualiza su estado de cifrado en el maestro, mientras que con el ICM permite al maestro verificar la integridad y autenticidad del enlace en la dirección esclavo-maestro.

Después de la recepción del SDM, el maestro activa la desencriptación del enlace con el mismo valor de clave $K_{CIPH}$. A continuación verifica la recepción correcta de los SCM e ICM provenientes del esclavo. Una vez hecho esto, responde con un SDM al esclavo activando su cifrado con la misma clave y enviando también periódicamente los mensajes SCM e ICM. El esclavo realizará las mismas acciones que el maestro, activando la desencriptación después de la recepción SDM y verificando la recepción periódica de SCM e ICM en la dirección maestro-esclavo.

Un proceso análogo es llevado a cabo cuando el maestro finaliza la sesión de encriptación, pero utilizando los mensajes EEM y EDM. En la Fig.2-38 se muestra un diagrama de un establecimiento de sesión de cifrado. De acuerdo con esto, los únicos mensajes de control que no están encriptados son los mensajes SEM y SDM transmitidos al inicio, justo antes de activar la encriptación. El resto de mensajes transmitidos mientras la sesión de cifrado está en proceso se enviarán encriptados al igual que el flujo de símbolos 8b/10b.

### 2.8.2 Autenticación de mensajes y frescura de datos

Para garantizar que ningún adversario pueda inyectar o alterar mensajes de control falsos en el enlace, se ha agregado un campo MAC al final de cada mensaje denominado MAC_F. Además, para conseguir la frescura de los mensajes evitando la posible retransmisión de los mismos, se ha utilizado un

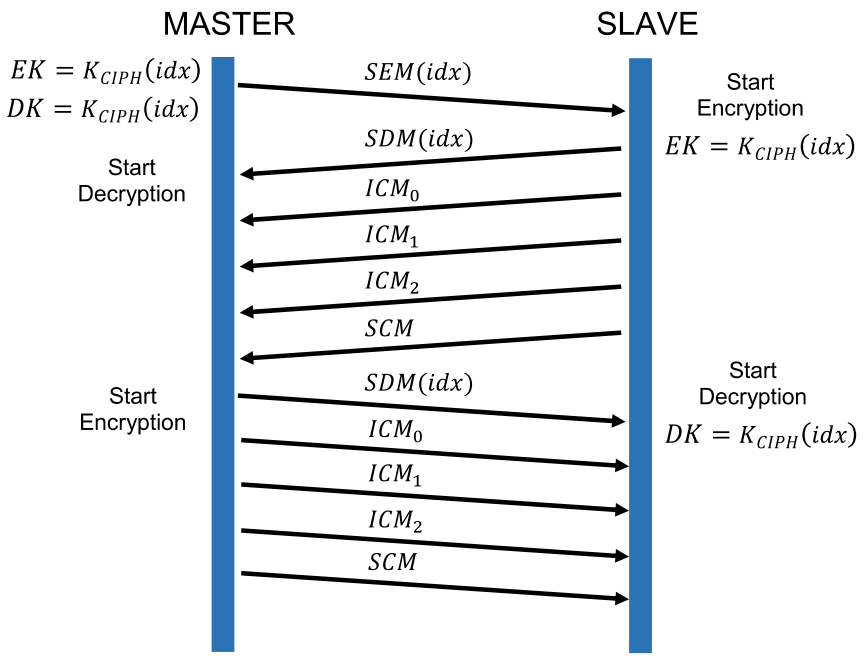



Fig.2-38. Diagrama del establecimiento de una sesión de encriptación. Al inicio de la sesión el maestro configura sus claves de cifrado y descifrado, $EK$ y $DK$, respectivamente, con el valor de la clave $K_{CIPH}$. Esta depende a su vez de la clave de sesión $K_S$. El campo KEY_INDEX de los mensajes SEM y SDM se configurará con el valor de $idx$.

campo de contador COUNTER de varios bytes de anchura, tal y como se muestra en la Fig.2-37. El campo MAC_F se calcula de la siguiente manera:

$$MAC\_F = CMAC(K_{FRAME}, H|D|CNT, 64) \tag{19}$$

donde $CMAC(K, M, Tlen)$ es una función descrita en [NIS05] que consiste en un cifrador por bloque AES con una clave de 128 bits trabajando en modo CMAC (Cipher-based Message Authentication Code). Esta función calcula el campo MAC de tamaño $Tlen$ bits para un mensaje $M$, con una clave $K$. En (19) $MAC\_F$ es un campo de 64 bits de longitud y $K_{FRAME}$ es la clave utilizada para el cálculo de $MAC\_F$ y que se deriva de la clave de sesión $K_S$. El campo $H|D|CNT$ representa el contenido del mensaje. Este es la concatenación de los campos $H$, $D$ y $CNT$, siendo estos la cabecera HEADER, los datos específicos del mensaje (como KEY_INDEX en SEM o el estado STATE en SCM) y el valor del campo COUNTER del mensaje, respectivamente.

Por cada mensaje recibido, tanto el maestro como el esclavo calculan el código MAC y lo comparan con el valor de campo $MAC\_F$ del mismo mensaje. Si estos valores no coinciden el mensaje es descartado.

En cuanto al valor del campo COUNTER, se han utilizado dos contadores internos en cada terminal, uno para los mensajes no periódicos y otro para los periódicos. El contador de los mensajes no periódicos en el transmisor se incrementa cada vez que un mensaje del tipo SEM, SDM, EEM o EDM es transmitido. En cuanto a los mensajes periódicos, estos son enviados en conjuntos formados por varios mensajes ICMs y un SCM. Todos ellos son transmitidos con el mismo valor de contador, que es incrementado por cada transmisión de dicho grupo de mensajes.

En recepción, cualquier mensaje recibido con un contador inferior al esperado se descarta. Esto evitará que un observador malintencionado con capacidad de capturar mensajes pueda realizar reenvíos de mensajes antiguos ya transmitidos, por ejemplo, en sesiones de encriptación previas.

### 2.8.3 Chequeo de la integridad del enlace

La integridad del enlace se evalúa periódicamente durante la sesión de cifrado que se divide en periodos de tiempo de aproximadamente 100 μs cada uno. A su vez, cada periodo es dividido en cuatro subintervalos y se delimita por la transmisión del mensaje $ICM_0$ al comienzo del primer subintervalo. En el segundo y tercer subintervalos se envían los mensajes $ICM_1$ e $ICM_2$, respectivamente.

El transmisor calcula el valor del código MAC de todo el flujo de símbolos 8b/10b encriptado entre dos mensajes $ICM_0$ para que el receptor pueda periódicamente chequear la integridad del enlace. El mensaje $ICM_1$ se transmite al comienzo del segundo subintervalo y lleva el campo MAC_I correspondiente al código MAC calculado durante el periodo anterior, tal y como se muestra en la Fig.2-39. Dicho valor MAC se calcula de acuerdo a la siguiente ecuación:

$$MAC\_I(i) = CMAC(K_{ICM}, S\_dat(i\text{-}1),64) \tag{20}$$

donde $MAC\_I(i)$ es el valor del código MAC calculado durante el periodo $i$-1 y transmitido en el $i$-ésimo periodo, $S\_dat(i\text{-}1)$ es la concatenación de todos los símbolos 8b/10b del periodo anterior y $K_{ICM}$ es la clave utilizada para el cálculo de la MAC y que se deriva de la clave de sesión $K_S$.

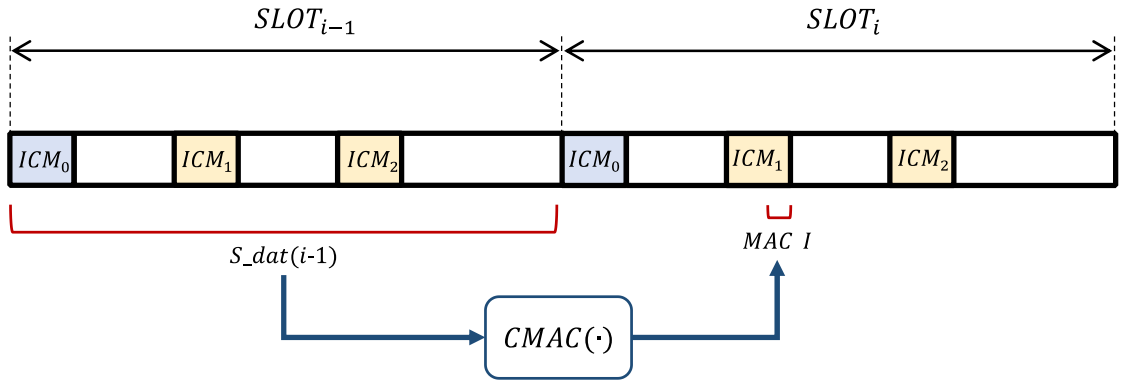



Fig.2-39. Esquema de dos periodos completos de ICMs. $S\_dat(i\text{-}1)$ representa los octetos del periodo $i$-1, incluyendo los mensajes ICM enviados durante dicho periodo.

Respecto a la integridad de los mensajes $ICM_0$ e $ICM_1$, tal y como se muestra en la Fig.2-37, el mensaje $ICM_0$ lleva su propio código de autenticación $MAC\_F$, sin embargo los mensajes $ICM_1$ no llevan ningún campo para su chequeo ya que han de transportar el valor de $MAC\_I$. Para chequear la integridad de estos se transmite un último mensaje $ICM_2$ al principio del tercer subintervalo. Este mensaje transporta el código de autenticación MAC calculado a partir del contenido de $ICM_1$ e $ICM_2$ tal y como se muestra en (21).

$$MAC\_FI = CMAC(K_{FRAME}, ICM\_dat_1 | ICM\_dat_2, 64) \tag{21}$$

donde $ICM\_dat_1 = H_{ICM1} | CNT_{ICM} | MAC\_I$, $ICM\_dat_2 = H_{ICM2} | CNT_{ICM}$, $H_{ICM1}$ y $H_{ICM2}$ son las cabeceras de $ICM_1$ e $ICM_2$, respectivamente, y $CNT_{ICM}$ el contador del set de mensajes ICM.

Se pueden detectar tres tipos de errores con respecto a los ICM. Un conjunto de ICM mal formado por pérdida de alguno de los tres mensajes $ICM_i$, un campo de contador de ICM incorrecto porque el contador del $ICM_0$ recibido es inferior al esperado o porque los mensajes $ICM_i$ en el conjunto no tienen el mismo valor de contador, y por último la recepción de valores erróneos de los códigos MAC asociados al set: el campo $MAC\_F$ del $ICM_0$, el $MAC\_I$ en el $ICM_1$ o el $MAC\_FI$ en el $ICM_2$. Cuando se reciben varios grupos ICM completos con un valor adecuado de contador y de los campos MAC durante varios periodos, el receptor considera que la integridad y autenticidad del enlace es correcta.

### 2.8.4 Refresco de claves

Para llevar a cabo la actualización de las claves del sistema el terminal maestro lleva a cabo la inicialización de una nueva sesión de encriptación empleando un nuevo índice de sesión. En primer lugar, el maestro incrementa el índice interno de sesión, $idx$ y obtiene la nueva clave de sesión, $K_S$, y de esta los nuevos valores de $K_{CIPH}$, $K_{FRAME}$ y $K_{ICM}$.

Una vez calculados estos nuevos valores lleva a cabo el envío de un nuevo SDM sin necesidad de finalizar la sesión actual de encriptación, es decir sin mandar un EEM ni EDM. En el nuevo SDM el campo KEY_INDEX es configurado con el nuevo valor $idx$, de forma que con este nuevo índice el esclavo también será capaz de calcular los valores para $K_S$, $K_{CIPH}$, $K_{FRAME}$ y $K_{ICM}$.

En ambos terminales no es necesario detener el proceso de encriptación ni desencriptación para actualizar las claves. El maestro configurará su hardware con las nuevas claves posteriormente a la transmisión del SDM, mientras que el esclavo lo hará una vez que lo haya recibido. Este procedimiento es llevado a cabo de forma sincronizada entre ambos terminales, de forma que el enlace permanece encriptado en todo momento además de que se sigue analizando su integridad y autenticidad.

Después de recibir el SDM, el terminal esclavo envía un nuevo SDM hacia el maestro utilizando el mismo índice de clave y, por lo tanto, realiza el mismo proceso que el mencionado en la dirección maestro-esclavo.

Los tres tipos de claves $K_{CIPH}$, $K_{FRAME}$ y $K_{ICM}$ se obtienen gracias a la clave de sesión $K_S$, y esta gracias a la clave maestra $K_M$ y al índice $idx$ de sesión. Para realizar el cálculo de estas claves se ha implementado el algoritmo KDF (Key Derivation Function) en modo contador como se especifica y analiza en [NIS09] y [ABD00], respectivamente.

El modo contador del algoritmo KDF se corresponde con el generador de claves paralelo descrito en [ABD00]. Con este método de generación de claves, a partir de una clave inicial $K$ y una PRF $F_K(\cdot)$ se generan las sucesivas claves $K_i$ de la forma $K_i = F_K(i)$ , siendo $i$ un contador tal que $1 \leq i \leq n$ y $n$ el número de claves que se desea generar. Según [ABD00] se puede considerar este método seguro

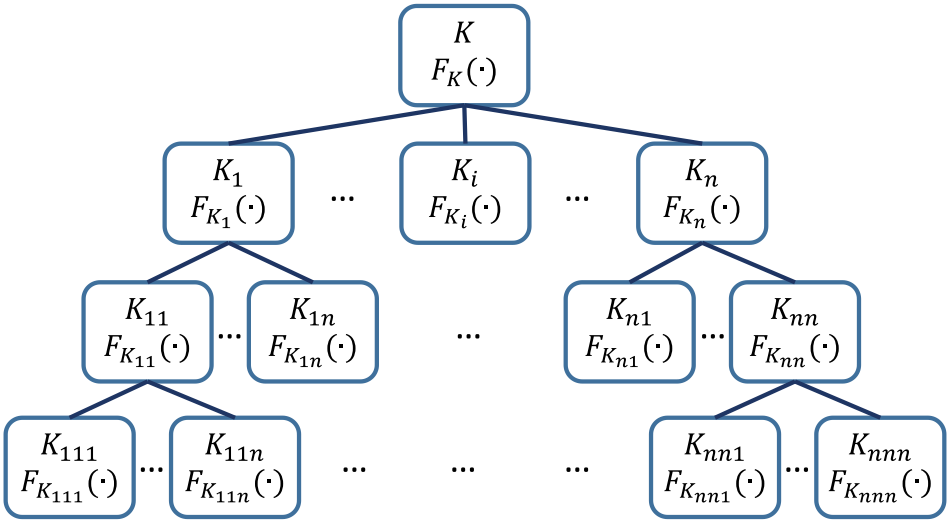



Fig.2-40. Árbol genérico de generación de claves.

en función del valor de la ventaja de tipo PRF $ADV_F^{PRF}$ sobre $F_K(\cdot)$. Es decir, si la PRF empleada para la generación de claves tiene una ventaja $ADV_F^{PRF}$ despreciable, el algoritmo de generación de claves se podrá considerar un algoritmo seguro.

Además, en [ABD00] se analiza la seguridad de un mecanismo de generación de claves basado en una estructura de árbol que también es incluido como recomendación en [NIS09]. Según este mecanismo cada una de las claves $K_i$ generadas gracias a $F_K(\cdot)$ pueden ser empleadas como claves a partir de las cuales se deriven otras nuevas. Es decir, si a partir de una clave $K$ y una función $F_K(\cdot)$ se derivan claves $K_i$ con $1 \leq i \leq n$, entonces se podrán establecer $n$ nuevos generadores de claves basados en las PRFs $F_{K_i}(\cdot)$. Realizando este proceso de forma iterativa se obtendría una estructura en árbol como la mostrada en la Fig.2-40.

En este trabajo, el algoritmo KDF en modo contador es ejecutado para obtener un material de clave con longitud $L_{MAT}$ igual a 512 bits, que representa la concatenación de cuatro claves de 128 bits cada una.

Cada una de las claves es obtenida de acuerdo a las expresiones mostradas en (22). Estas ecuaciones son equivalentes a lanzar una iteración del algoritmo KDF configurado para obtener $L_{MAT}$ bits.

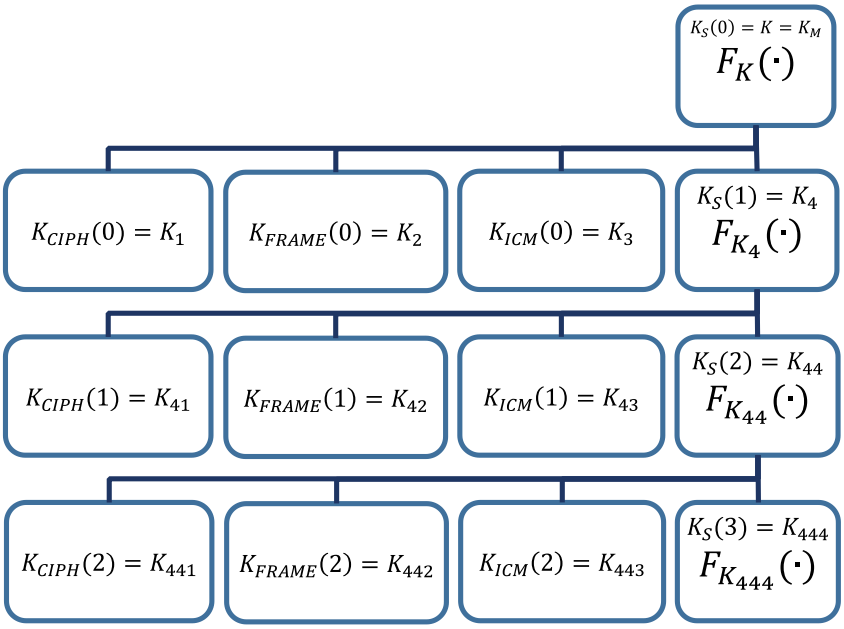



Fig.2-41. Árbol de generación de claves implementado en tres sesiones de encriptación. En la sesión con índice $idx = 0$, la clave de sesión es la clave maestra, $K_S = K_M$. A partir de $K_S(idx)$ se calculan las claves $K_{CIPH}$, $K_{FRAME}$, $K_{ICM}$ y la clave $K_S(idx + 1)$ de la siguiente sesión.

$$\begin{aligned} K_{CIPH}(idx) &= PRF\_func(K_S(idx), 0x01|VAL|L_{MAT}) \\ K_{FRAME}(idx) &= PRF\_func(K_S(idx), 0x02|VAL|L_{MAT}) \\ K_{ICM}(idx) &= PRF\_func(K_S(idx), 0x03|VAL|L_{MAT}) \\ K_S(idx+1) &= PRF\_func(K_S(idx), 0x04|VAL|L_{MAT}) \end{aligned} \tag{22}$$

El valor inicial de $K_S(idx)$ es la clave maestra del sistema, esto es $K_S(0) = K_M$, y se establece después de adquirir dicha clave al inicio de la primera sesión de encriptación. $PRF\_func$ es una PRF utilizada en la definición de KDF. Esta se puede identificar con la PRF empleada para el generador de claves paralelo descrito en [ABD00], tal que $PRF\_func(K_S(idx),\cdot) = F_{K_S(idx)}(\cdot)$. En nuestro caso esta función se corresponde con un cifrador AES con clave de 128 bits trabajando en modo CMAC. El segundo argumento de la función $PRF\_func$ ha sido definido como en la especificación del KDF: la concatenación de un contador (que en este caso se incrementa tomando valores entre 0x01 y 0x04), una constante $VAL$ que en nuestro caso se ha dejado a cero y la longitud del material de clave $L_{MAT}$. El árbol de generación de claves quedaría finalmente como en la Fig.2-41.

### 2.8.5 Inserción de los mensajes de control en el flujo 8b/10b

Tal y como se muestra en la Fig.2-37 el tamaño de los mensajes de control es constante e igual a 16 bytes. Para insertar estos mensajes se ha utilizado un mecanismo similar al presentado en [PER19a] y representado en la Fig.2-8. En este mecanismo, antes de llevar a cabo la encriptación, se analizan los símbolos 8b/10b buscando la presencia de sets de tipo *idle*. Cuando se detectan al menos ocho sets *idle* continuos, el mensaje a enviar se inserta en el flujo de símbolos 8b/10b reemplazando estos *idles* por los símbolos que forman el propio mensaje. En el receptor, después de la desencriptación se realiza el proceso inverso, extrayendo el mensaje de control y reemplazándolo por ocho *idles*.

Los *idles* están formados por dos símbolos 8b/10b, uno de control y otro de datos. Están presentes continuamente en el enlace cuando no se transmite ninguna trama. Cuando las tramas se transmiten continuamente, los *idles* estarán presentes solo durante el IFG entre ellas. Si el ancho de banda ocupado por el tráfico de las tramas es suficientemente bajo y los IFG son lo suficientemente largos, al menos equivalentes a ocho *idles* consecutivos, entonces será posible la inserción de los mensajes de control. Sin embargo, si el ancho de banda ocupado es el máximo posible, el IFG será mínimo y, según lo especificado por el estándar, la longitud mínima de los IFGs ha de ser al menos de 96 μs, lo que equivale a seis *idles* consecutivos. En este caso, será imposible insertar los mensajes de control a menos que no reduzcamos el ancho de banda empleado por las tramas de datos. Para evitar este problema y mantener el ancho de banda de datos al máximo en la solución propuesta, el preámbulo de las tramas transmitidas se ha reducido en seis *bytes*, lo que amplía el IFG mínimo hasta nueve *idles,* que es suficiente para nuestros propósitos. En el receptor, después de descifrar y extraer los mensajes de control, el preámbulo de las tramas Ethernet se restaura a su tamaño original antes de que estas lleguen a la capa MAC.

### 2.8.6 Implementación sobre FPGA

El mecanismo de chequeo de autenticación, integridad y refresco de claves fue implementado en [PER20b] junto con el esquema de encriptación PSCFB-MOD

descrito en el Aparado 2.5.3. Al igual que las implementaciones explicadas en el Apartado 2.6, el sistema descrito fue integrado en un interfaz Ethernet como el de la Fig.2-29, donde en este caso el bloque MANAGEMENT también es el encargado de generar y procesar los mensajes de control y de llevar a cabo el cálculo y refresco de las claves del sistema. Dos interfaces Ethernet fueron enfrentados tal y como se muestra en el diagrama de la Fig.2-42, haciendo uso también de dos generadores de tramas Ethernet para el chequeo del tráfico de datos. En este caso dos placas de evaluación VC707 fueron configuradas con los roles de maestro y esclavo, respectivamente, como muestra en la foto del *set-up* de test de la Fig.2-43.

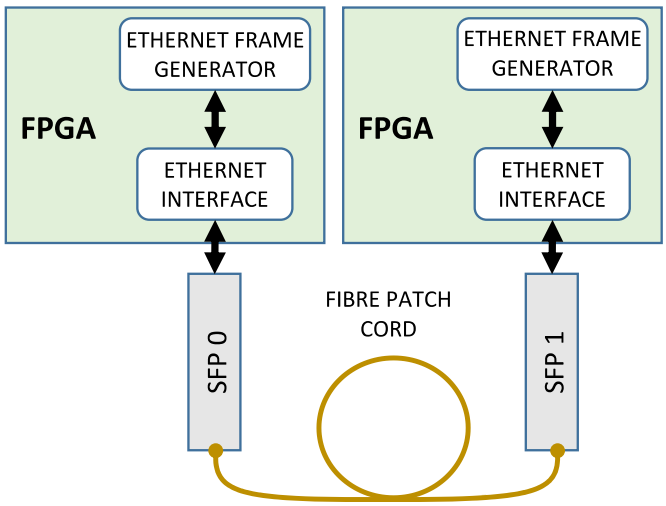



Fig.2-42. Esquema del *set-up* de test para verificar el protocolo de control.

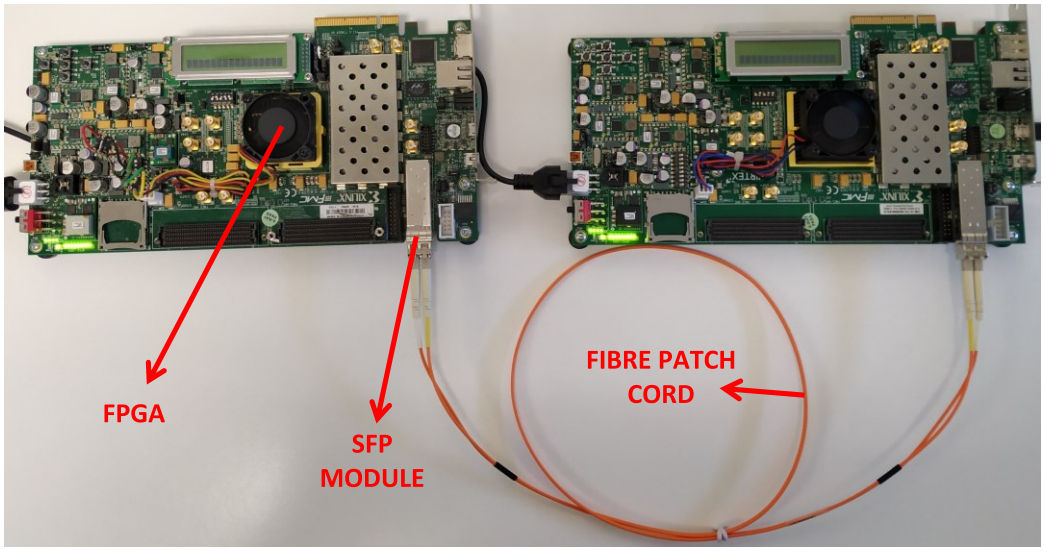



Fig.2-43. Foto del *set-up* de test con dos placas de evaluación VC707.

# 3

# Conclusiones



## 3.1 Conclusiones generales

Hoy en día, Ethernet óptico es un estándar de comunicaciones ampliamente utilizado en aplicaciones donde son necesarias altas tasas de transmisión de datos. Desde redes en entornos industriales con velocidades de hasta 1 Gbps, a enlaces de acceso o transporte de datos en redes de banda ancha, donde se llegan a superar tasas de 100 Gbps.

Respecto a la seguridad en la transmisión de datos, existen diferentes mecanismos o protocolos estandarizados en distintas capas de comunicación, como por ejemplo IPsec o MACsec. Dichos mecanismos son capaces de garantizar tanto la confidencialidad, autenticidad como la integridad de la comunicación. Normalmente introducen un *overhead* en los datos, lo que disminuye la tasa neta de transmisión al mismo tiempo que incrementa su latencia. Por otro lado, la encriptación es llevada a cabo a nivel de paquete logrando su confidencialidad, sin embargo no logra su privacidad, es decir, no permite ofuscar la presencia de una transmisión. Este hecho implica que aunque un observador malintencionado no sea capaz de interpretar el contenido de los paquetes, sí que pueda ser capaz de detectar la presencia de los mismos y

analizar el patrón o comportamiento de las comunicaciones. Todo esto gracias al procesado de información como la longitud de los paquetes o la duración y frecuencia de las transmisiones.

En el caso de las comunicaciones sobre Ethernet óptico, hasta la fecha no se han propuesto mecanismos que logren la mencionada privacidad al mismo tiempo que la confidencialidad, sin que además introduzca un *overhead* o una latencia indeseada.

En la presente tesis se han realizado propuestas viables de modificación en dos de los principales estándares ópticos Ethernet, 1000Base-X y 10GBase-R, de forma que se pueda llevar a cabo la encriptación en su capa física. Dichas modificaciones han sido diseñadas de forma que mantengan la compatibilidad con el hardware electrónico más dependiente del medio de transmisión en dichos estándares, como los módulos ópticos SFP, los SERDES o los circuitos de recuperación de reloj y datos.

Por otro lado, se ha realizado un estudio de posibles esquemas de encriptación por *streaming* que sean capaces de cifrar datos a velocidades superiores a 1 Gbps y adaptarlos a las arquitecturas propuestas. Además se han propuesto posibles mecanismos para llevar a cabo la sincronización de los módulos de encriptación entre dos terminales remotos.

Una vez implementadas, se ha conseguido que las soluciones propuestas lleven a cabo la encriptación introduciendo la menor latencia posible, al menos en un orden de magnitud igual o inferior al de soluciones en otros estándares de comunicaciones como OTN.

Para evaluar la confidencialidad de los diferentes mecanismos de encriptación se ha llevado a cabo el estudio de su ventaja de tipo IND-CPA extrayendo su expresión de seguridad. Además se han establecido unas condiciones de diseño que permitan lograr al menos una seguridad mayor o igual que con el esquema de referencia CTR funcionando con un cifrador por bloque con un tamaño de bloque estándar de 128 bits.

Respecto a la privacidad, esta se ha podido analizar mediante la medida de la entropía de diferentes flujos de información correspondientes a diferentes patrones de datos sin encriptar y encriptados. Una vez calculada la entropía se

ha comprobado que los patrones encriptados adquieren el máximo valor posible independientemente del tipo de tráfico al que representen, incluyendo los patrones correspondientes al estado de no transmisión, no siendo así cuando no hay encriptación activa. Esto permite concluir que las situaciones de transmisión y no transmisión de datos son indistinguibles gracias al uso de los mecanismos propuestos, logrando así la privacidad mencionada.

Además de la confidencialidad y privacidad, también se ha implementado una propuesta para lograr la integridad, autenticación y refresco de claves a nivel de capa física en el caso del estándar 1000Base-X.

Todas las soluciones mencionadas han sido implementadas físicamente en dispositivos tipo FPGA para poder evaluar su viabilidad en un entorno de comunicaciones real empleando elementos *hardware* comerciales.

## 3.2 Líneas de investigación futuras

En este trabajo se han propuesto e implementado diversas soluciones para conseguir la encriptación a nivel de capa física en dos de los estándares ópticos Ethernet más empleados, 1000Base-X y 10GBase-R. Aunque las propuestas atañen sólo a estos dos estándares ambos están basados en las principales codificaciones para transmisiones de alta velocidad, 8b/10b y 64b/66b. Esto permitiría escalar o adaptar estas soluciones de encriptación a estándares Ethernet con mayores tasas de transmisión, como 40 Gbps o 100 Gbps, ya que estos se basan en las mismas codificaciones. Lo mismo podría concluirse con otros estándares no Ethernet como Fibre channel o PCI-express, basados también en estas mismas codificaciones u otras similares.

# A

# Referencias


[ABD00] Michel Abdalla and Mihir Bellare. "Increasing the Lifetime of a Key: A Comparative Analysis of the Security of Re-keying Techniques". In Tatsuaki Okamoto, editor, *Advances in Cryptology — ASIACRYPT 2000. Lecture Notes in Computer Science, vol 1976*, pages 546–559, Berlin, Heidelberg, 2000. Springer-Verlag.

[ALK02] Ammar Alkassar, Alexander Geraldy, Birgit Pfitzmann, and Ahmad-Reza Sadeghi. "Optimized Self-Synchronizing Mode of Operation". In Mitsuru Matsui, editor, *Fast Software Encryption. FSE 2001. Lecture Notes in Computer Science, vol 2355*, pages 78–91. Springer Berlin Heidelberg, 2002.

[BEL05a] M. Bellare and P. Rogaway. "Introduction to modern cryptography", May 2005. [Online]. Available: http://web.cs.ucdavis.edu/ rogaway/classes/227 /spring05/book/main.pdf [Accessed 17 Aug. 2017].

[BEL05b] M. Bellare and P. Rogaway. "Chapter 5.4 Indistinguishably under chosen-plaintext attack". In *Introduction to modern cryptography*, May 2005. [Online]. Available: http://web.cs.ucdavis.edu/ rogaway/classes/227/spring05/book /main.pdf [Accessed 17 Aug. 2017].

[BEL05c] M. Bellare and P. Rogaway. "Chapter 5.7 Security of CTR Modes". In *Introduction to modern cryptography*, May 2005. [Online]. Available: http://web.cs.ucdavis.edu/ rogaway/classes/227/spring05/book/main.pdf [Accessed 17 Aug. 2017].

[BEL97] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway. "A Concrete Security Treatment of Symmetric Encryption: Analysis of the DES Modes of Operation". In *Proceedings 38th Annual Symposium on Foundations of Computer Science (FOCS)*, pages 394–403, Oct 1997.

[CIS16] Cisco Systems. "Cisco Global Cloud Index: Forecast and Methodology. 2015-2020", 2016. [Online]. Available: https://www.cisco.com/c/dam/en /us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf. [Accessed 17 Aug. 2017].

[CIS18] Cisco Systems. "Cisco 2018 Annual Cybersecurity Report", Feb. 2018. [Online]. Available: https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf [Accessed 17 Aug. 2017].

[DAE05] J. Daemen, J. Lano, and B. Preneel. "Chosen Ciphertext Attack on SSS", 2005. [Online]. Available: http://www.ecrypt.eu.org/stream/papersdir/044.pdf [Accessed 17 Aug. 2017].

[ELK13] D. Elkouss, J. Martinez-Mateo, A. Ciurana, and V. Martin. "Secure Optical Networks Based on Quantum Key Distribution and Weakly Trusted Repeaters". *IEEE/OSA Journal of Optical Communications and Networking*, 5(4):316–328, April 2013.

[FOK11] M. P. Fok, Z. Wang, Y. Deng, and P. R. Prucnal. "Optical Layer Security in Fiber-Optic Networks". *IEEE Transactions on Information Forensics and Security*, 6(3):725–736, Sep. 2011.

[FUR14] M. Furdek, N. Skorin-Kapov, S. Zsigmond, and L. Wosinska. "Vulnerabilities and Security Issues in Optical Networks". In *2014 16th International Conference on Transparent Optical Networks (ICTON)*, pages 1–4, Graz, Austria, July 2014.

[GAR17a] M. Garcia-Bosque, C. Sánchez-Azqueta, A. Pérez, A. D. Martínez, and S. Celma. "Fast and Secure Chaotic Stream Cipher with a MEMS-based Seed Generator". In *2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6, May 2017.

[GAR17b] M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, and S. Celma. "Application of a MEMS-Based TRNG in a Chaotic Stream Cipher". *Sensors*, 17(3):646, Mar. 2017.

[GUA16] K. Guan, J. Kakande, and J. Cho. "On Deploying Encryption Solutions to Provide Secure Transport-as-a-Service (TaaS) in Core and Metro Networks". In *ECOC 2016; 42nd European Conference on Optical Communication*, pages 18–22, Sep. 2016.

[HEY01] H. M. Heys. "An Analysis of the Statistical Self-Synchronization of Stream Ciphers". In *Proceedings IEEE INFOCOM 2001. Conference on Computer Communications. Twentieth Annual Joint Conference of the IEEE Computer and Communications Society (Cat. No.01CH37213)*, volume 2, pages 897–904, Anchorage, USA, April 2001.

[HEY03] H. M. Heys. "Analysis of the Statistical Cipher Feedback Mode of Block Ciphers". *IEEE Transactions on Computers*, 52(1):77–92, Jan 2003.

[HEY11] H. M. Heys and L. Zhang. "Pipelined Statistical Cipher Feedback: A New Mode for High-Speed Self-Synchronizing Stream Encryption". *IEEE Transactions on Computers*, 60(11):1581–1595, Nov 2011.

[HIZ10] J. Hizanidis, S. Deligiannidis, A. Bogris, and D. Syvridis. "Enhancement of Chaos Encryption Potential by Combining All-Optical and Electrooptical Chaos Generators". *IEEE Journal of Quantum Electronics*, 46(11):1642–1649, Nov 2010.

[IEE06] "IEEE Standard for Local and metropolitan area networks-Media Access Control (MAC) Security". *IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006)*, pages 1–239, Dec. 2018.

[IEE16] "IEEE Standard for Ethernet". *IEEE Std 802.3-2015 (Revision of IEEE Std 802.3-2012)*, pages 1–4017, March 2016.

[JI17] J. Ji, G. Zhang, W. Li, L. Sun, K. Wang, and M. Xu. "Performance Analysis of Physical-Layer Security in an OCDMA-based Wiretap Channel". *IEEE/OSA Journal of Optical Communications and Networking*, 9(10):813–818, Oct 2017.

[JOU06] Antoine Joux and Frédéric Muller. "Chosen-Ciphertext Attacks Against MOSQUITO". In Matthew Robshaw, editor, *Fast Software Encryption. FSE 2006. Lecture Notes in Computer Science, vol 4047*, pages 390–404. Springer Berlin Heidelberg, 2006.

[JUN99] Oliver Jung and Christoph Ruland. "Encryption with Statistical Self-Synchronization in Synchronous Broadband Networks". In Çetin K. Koç and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems. CHES 1999. Lecture Notes in Computer Science, vol 1717*, pages 340–352. Springer Berlin Heidelberg, 1999.

[KEN05] S. Kent and K. Seo. "RFC 4301: Security Architecture for the Internet Protocol", 2005. [Online]. Available: https://tools.ietf.org/html/rfc4301 [Accessed 17 Aug. 2017].

[KLE13] Andreas Klein. *Stream Ciphers*. Springer-Verlag, 2013.

[KNU97] Donald E. Knuth. *The Art of Computer Programming, Volume 2: Seminumerical Algorithms*. Addison-Wesley, Boston, third edition, 1997.

[KOL13] S. S. Kolahi, Y. R. Cao, and H. Chen. "Bandwidth-IPSec Security Trade-off in IPv4 and IPv6 in Windows 7 Environment". In *Second International Conference on Future Generation Communication Technologies (FGCT 2013)*, pages 148–152, Nov 2013.

[LEC07] Pierre L'Ecuyer and Richard Simard. "TestU01: A C Library for Empirical Testing of Random Number Generators". *ACM Trans. Math. Softw.*, 33(4):22:1–22:40, August 2007.

[LEE06] K. C. Lee, S. Lee, and M. H. Lee. "Worst Case Communication Delay of Real-Time Industrial Switched Ethernet With Multiple Levels". *IEEE Transactions on Industrial Electronics*, 53(5):1669–1676, Oct 2006.

[LUB88] M. Luby and C. Rackoff. "How to Construct Pseudorandom Permutations from Pseudorandom Functions". *SIAM Journal on Computing*, 17(2):373–386, 1988.

[MAR97] G. Marsaglia. *Diehard: A Battery of Tests of Randomness*. 1997. [Online]. Available: http://webhome.phy.duke.edu˜rgb/General/dieharder.php [Accessed 17 Aug. 2017].

[MET09] Metro Ethernet Forum. "MEF White Paper: Delivering Ubiquitous Ethernet Services using an Array of Access Technologies", 2009. [Online]. Available: https://wiki.mef.net/display/CESG/MEF+White+Paper+-+Ethernet+Services+and+Access+Technologies [Accessed 17 Aug. 2017].

[MIC16] MicroSemi. "In-flight Encryption in Service Provider Networks, No: PMC-2150716, Issue 2", 2016. [Online]. Available: https://pmcs.com/cgi-bin/document.pl?docnum=2150716. [Accessed 17 Aug. 2017].

[NIS01] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. NIST Special Publication 800-38A, Gaithersburg, MD, USA, 2001.

[NIS05] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication*. NIST Special Publication 800-38B, Gaithersburg, MD, USA, 2005.

[NIS09] L. Chen. *Recommendation for Key Derivation Using Pseudorandom Functions*. NIST Special Publication 800-108, Gaithersburg, MD, USA, 2009.

[NIS10] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, and A. Heckert. *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications*. NIST Special Publication 800-22 Rev.1a, Gaithersburg, MD, USA, 2010.

[NIS15] E. Barker and J. Kesley. *Recommendation for Random Number Generation Using Deterministic Random Bit Generators*. NIST Special Publication 800-90A Rev.1v, Gaithersburg, MD, USA, 2015.

[NIS16] M. Dworkin. *Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption*. NIST Special Publication 800-38G, Gaithersburg, MD, USA, 2016.

[PER19a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems". *IEEE Transactions on Instrumentation and Measurement*, 68(12):4876–4886, Dec 2019.

[PER19b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Physical Layer Encryption for Industrial Ethernet in Gigabit Optical Links". *IEEE Transactions on Industrial Electronics*, 66(4):3287–3295, April 2019.

[PER19c] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaotic Encryption for 10-Gb Ethernet Optical Links". *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(2):859–868, Feb. 2019.

[PER19d] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-Synchronized Encryption for Physical Layer in 10Gbps Optical Links". *IEEE Transactions on Computers*, 68(6):899–911, June 2019.

[PER19e] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-Synchronized Encryption Using an FPE Block Cipher for Gigabit Ethernet". In *2019 15th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, pages 81–84, Lausanne, Switzerland, July 2019.

[PER20a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "A New Method for Format Preserving Encryption in High-Data Rate Communications". *IEEE Access*, 8:21003–21016, 2020.

[PER20b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-synchronized Encryption for Physical Layer in 1Gbps Ethernet Optical Links". *IEEE Access*, Pending Acceptance.

[ROB08] Matthew Robshaw and Olivier Billet. *New Stream Cipher Designs: The eSTREAM Finalists*. Springer-Verlag, Berlin, Heidelberg, 2008.

[ROG11] P. Rogaway. "Evaluation of some Blockcipher Modes of Operation". In *Technical report, Cryptography Research and Evaluation Committees (CRYPTREC) for the Goverment of Japan*, Feb. 2011.

[SAU10] T. Sauter. "The Three Generations of Field-Level Networks—Evolution and Compatibility Issues". *IEEE Transactions on Industrial Electronics*, 57(11):3585–3595, Nov 2010.

[SAU11] T. Sauter, S. Soucek, W. Kastner, and D. Dietrich. "The Evolution of Factory and Building Automation". *IEEE Industrial Electronics Magazine*, 5(3):35–48, Sep. 2011.

[SHA45] C. E. Shannon. "A Mathematical Theory of Cryptography". *Bell Telephone Labs*, Sept 1945.

[SKO16] N. Skorin-Kapov, M. Furdek, S. Zsigmond, and L. Wosinska. "Physical-Layer Security in Evolving Optical Networks". *IEEE Communications Magazine*, 54(8):110–117, Aug. 2016.

[STI02] Douglas Stinson. *Cryptography: Theory and Practice*. CRC/C&H, Boca Raton, FL, USA, 2nd edition, 2002.

[TRO05] L. Troell, J. Burns, K. Chapman, D. Goddard, M. Soderlund, and C. Ward. "Converged vs. Dedicated IPSec Encryption Testing in Gigabit Ethernet Networks, Technical Report", 2005. [Online]. Available: http://scholarworks.rit.edu/article/1743 [Accessed 17 Aug. 2017].

[XEN06] C. Xenakis, N. Laoutaris, L. Merakos, and I. Stavrakakis. "A Generic Characterization of the Overheads Imposed by IPsec and Associated Cryptographic Algorithms". *Computer Networks*, 50(17):3225 – 3241, 2006.

[YAN04] F. Yang and H. M. Heys. "Comparison of Two Self-Synchronizing Cipher Modes". In *Queen's 22nd Biennial Symposium on Communications*, 2004.

[ZAF11] M. Zafar Iqbal, H. Fathallah, and N. Belhadj. "Optical Fiber Tapping: Methods and Precautions". In *8th International Conference on High-capacity Optical Networks and Emerging Technologies (HONET)*, pages 164–168, Rihad, Dec 2011.

# B

# Copia de los trabajos presentados en la tesis

# Chaotic Encryption for 10-Gb Ethernet Optical Links

Adrián Pérez-Resa, Miguel Garcia-Bosque, Carlos Sánchez-Azqueta, and Santiago Celma

***Abstract*— In this paper, a new physical layer encryption method for optical 10-Gb Ethernet links is proposed. Necessary modifications to introduce encryption in Ethernet 10GBase-R standard have been considered. This security enhancement has consisted of a symmetric streaming encryption of the 64b/66b data flow at physical coding sublayer level thanks to two keystream generators based on a chaotic algorithm. The overall system has been implemented and tested in a field programmable gate array. Ethernet traffic has been encrypted, transmitted, and decrypted over a multimode optical link. Experimental results are analyzed concluding that it is possible to cipher traffic at this level and hide the complete Ethernet traffic pattern from any passive eavesdropper. In addition, no overhead is introduced during encryption, getting no losses in the total throughput.**

***Index Terms*— Ethernet, 10GBASE-R, cryptography, stream cipher, skew tent map.**

## I. Introduction

NOWADAYS, thanks to advances in optical communication technologies it is possible to meet the ever-increasing data traffic demand of modern networks. It is estimated that IP traffic in data centers will reach 15.3 Zettabytes per year by 2020 [1].

On the other hand, security and confidentiality in communications networks are important fields of study. In the specific case of optical networks, some studies were carried out in the past to discern whether the degradation of a link or failure of a service is due to a natural network defect or an external attack [2], [3]. In fact, threat analysis in the physical layer is considered crucial for secure communications [4], [5]. However, nowadays there are simple methods for intercepting the signal without appreciably interfering in optical communication [6]. Such methods use simple fiber coupling devices and electro-optical converters and numerous examples are easily available for the public on the Internet.

In a layered communication model, such as OSI (Open System Interconnection) or TCP/IP (Transmission Control Protocol/Internet Protocol), 'in-flight' encryption can be carried out at different communication layers. For example, well known encryption methods for layer 2 and layer 3 are the protocols MACsec [7] and IPsec [8] respectively.

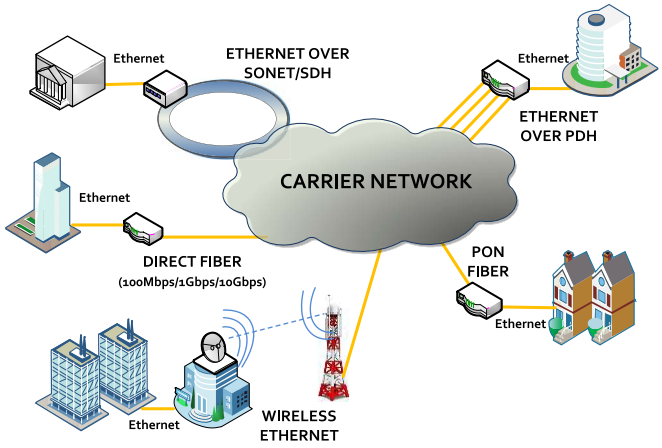



Fig. 1. Simple example of different access technologies for a CEN (Carrier Ethernet Network).

At physical layer, there are security solutions directly related to the optical technology such as QKD (Quantum Key Distribution) [9] or OCDM (Optical Code-Division Multiplexing) [10], or related to payload encryption in some optical protocols as OTN (Optical Transport Network) standard [11]. Encryption at physical layer brings some benefits such as minimum latency and zero overhead, achieving maximum link efficiency. In fact, commercial OTN equipment performing encryption at line rate is able to achieve a 100% throughput in contrast with encryption at other communication layers [12]. As an example, in IPsec the inherent overhead introduced during encryption reduces the overall throughput between 20% and 90% of the maximum achievable [13].

Today one of the most used technologies for the access to optical transport networks is Ethernet. As shown in Fig. 1, some access technologies in CENs (Carrier Ethernet Networks) 'last mile' are Ethernet over Fiber (Direct Fiber, SONET/SDH, PON), Ethernet over PDH, Wireless Ethernet, etc. [14].

10GBase-R is one of the most widely used physical layer standards for Ethernet at 10 Gbps in optical links, but nowadays there is no mechanism or standard providing physical layer security on it. For this reason, the main motivation of this work is to propose and implement the necessary modifications

Manuscript received February 13, 2018; revised July 10, 2018 and August 24, 2018; accepted August 25, 2018. Date of publication September 26, 2018; date of current version January 18, 2019. This work was supported in part by MINECO-FEDER under Grant TEC2014-52840-R and Grant TEC2017-85867-R. The work of M. Garcia-Bosque was supported by FPU Fellowship under Grant FPU14/03523. This paper was recommended by Associate Editor G. Masera. *(Corresponding author: Adrián Pérez-Resa.)*

The authors are with the Electronic and Communications Engineering Department, University of Zaragoza, 50009 Zaragoza, Spain (e-mail: aprz@unizar.es; mgbosque@unizar.es; csanaz@unizar.es; scelma@unizar.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TCSI.2018.2867918

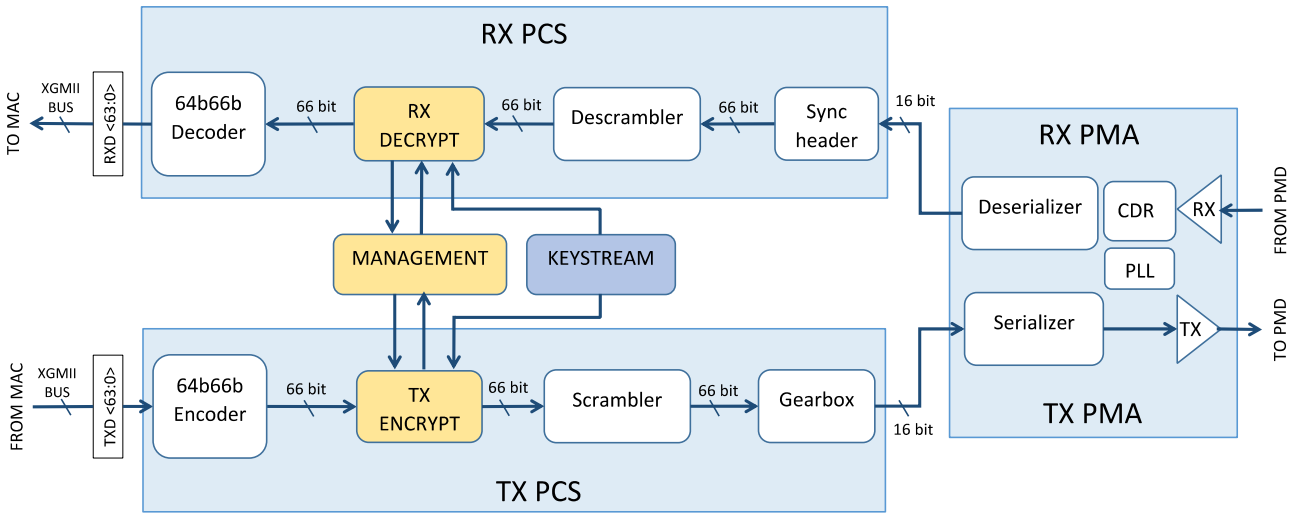



Fig. 2. PCS structure and the proposed encryption function 10G-PHYsec.

to introduce encryption in its PCS (Physical Coding Sublayer) level. We have called this method 10G-PHYsec.

An encryption system at physical layer of Ethernet provides the mentioned advantages such as no overhead, low latency and also obfuscation of customer data traffic patterns, which results in an overall improvement of security. An example of the same property is proved at a lower transmission speed in an Ethernet interface using 1000Base-X physical layer [15]. In this case the encryption mechanism is completely different, as 8b/10b encoding is used, consisting of an additive stream cipher where modulo-267 addition is performed between the keystream and plaintext.

The article is organized as follows: Section II explains the general structure of the encryption system in the Ethernet physical layer 10GBase-R, Section III explains the encryption infrastructure composed by the stream cipher operation and the synchronization mechanism between TX (transmitter) and RX (receiver), Section IV deals with keystream generation of the proposed stream cipher; Section V details the system implementation and test results and finally in Section VI conclusions obtained in this work are given.

## II. PCS Layer Encryption

In Ethernet standards, the physical layer is usually divided into three sublayers with different functionalities: PCS (Physical Coding Sublayer), PMD (Physical Medium Dependent) and PMA (Physical Medium Attachment). PCS sublayer performs functions such as link establishment, data encoding, scrambling, synchronization, clock rate adaptation, etc.

As many high speed standards in optical Ethernet, a baseband serial data transmission is carried out while clock frequency information is embedded in the serial bitstream itself. At PMA sublayer a CDR (Clock and Data Recovery) circuit is responsible for extracting timing information from the data stream. As a critical part of a typical transceiver for high speed communications many researches have been carried out about this kind of circuits [16], [17]. In order to facilitate the work of the CDR circuit, information is usually encoded in such a way that a good transition density and short run length are achieved. Also a DC-balanced serial data stream must be guaranteed by getting a good disparity, which is important for some transmission media, as optical links.

In the case of 10GBase-R standard, 64b/66b encoding is used at PCS sublayer. This type of block-line encoding achieves the properties set out before in a statistical way thanks to the scrambling of the bitstream. The basic structure of the PCS sublayer in 10GBase-R standard is shown in Fig. 2, where its main function blocks such as 64b/66b encoder/decoder and scrambler/descrambler are shown.

In order to encrypt physical layer when block-line encoding is used, it is necessary to preserve the mentioned encoder properties, therefore the location of the cipher in the datapath must be taken into account. In this work we propose to carry out encryption at the input of the scrambler and decryption at the output of the descrambler as shown in Fig. 2

Unlike other encryption mechanisms such as the MACsec, where the encrypted data are directly the Ethernet packets, in this work, the 64b/66b data flow is directly encrypted thanks to a stream cipher based on a chaotic algorithm. The main advantages of this method is that there is no throughput loss and that not only data packet content is encrypted but also the link activity or Ethernet data traffic pattern. By encrypting at 64b/66b block level, both data and control blocks are obfuscated, such as start and end frame blocks or control blocks with IDLE control characters inside, making frame pattern indecipherable.

## III. Encryption Infrastructure

The encryption infrastructure is shown in Fig. 3. This structure is composed of three blocks, the TX_ENCRYPT, RX_DECRYPT and the MANAGEMENT module.

### A. Encryption Function

The stream cipher operation is performed by the CIPHER_OP_TX and CIPHER_OP_RX modules in Fig. 3.

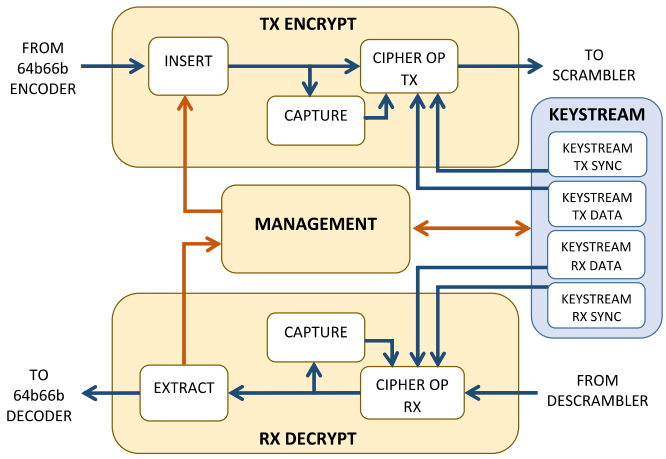



Fig. 3. Encryption infrastructure for 10G-PHYsec function.

The encoded blocks after the 64b/66b encoder are composed of a two-bit synchronization header plus a 64-bit payload, forming a 66-bit block. On the one hand, the 66-bit block type depends on the type field inside the block payload and the value of the 2-bit block synchronization header, then both, block payload and synchronization header are necessary to be encrypted to mask the block type. On the other hand, the main purpose of the synchronization header is to guarantee 66-bit block alignment while limiting the run length to 66 bits. This header only has two possible values ('01' or '10'), which produces a transition between 0 and 1 every 66 bits. Thus, after encryption, it is necessary to preserve this bit transition.

The stream cipher operation is shown in Fig. 4. Both modules CIPHER_OP_TX and CIPHER_OP_RX perform the same operation. Firstly, the 66-bit block, D_IN, is split in two parts, the synchronization header and the block payload. The block payload is directly encrypted by performing an XOR operation between its 64 bits and a 64-bit keystream data sequence, called 'keystream data' in Fig. 4. The 2-bit synchronization header is mapped to values '0' or '1' depending, respectively, on whether it is equal to '01' or '10'. Then, the mapped value is encrypted performing an XOR operation with a 1-bit keystream sequence called 'keystream sync'. After encryption, this new value is reverse mapped resulting in a new header '01' or '10' depending on whether it is '0' or '1'. In this way, the transition between 0 and 1 every 66 bits is guaranteed. Finally, the new synchronization header is concatenated with the encrypted block payload resulting in the output D_OUT, which is sent to the scrambler when encrypting or to the 64b/66b decoder when decrypting.

### *B. Encryption Synchronization*

It is necessary to activate and deactivate the keystream generators and the encryption operation synchronously between transmitter and receiver. For that purpose, the MANAGEMENT module in Fig. 3 performs the insertion and extraction of two management sequences into the 64b/66b block stream. One of these sequences is used for the encryption activation and the other one for deactivation.

TABLE I
64B/66B BLOCK FORMATS IN 10GBASE-R STANDARD

| Input Data | Sync | Block Payload | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| Bit Position: / Data Block Format: | 0 1 | 2 | | | | | | | 65 |
| $D_0\,D_1\,D_2\,D_3/D_4\,D_5\,D_6\,D_7$ | 01 | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | $D_7$ |
| **Control Block Formats:** | | **Block Type** | | | | | | | |
| $C_0\,C_1\,C_2\,C_3/C_4\,C_5\,C_6\,C_7$ | 10 | 0x1e | $C_0$ | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $C_0\,C_1\,C_2\,C_3/O_4\,D_5\,D_6\,D_7$ | 10 | 0x2d | $C_0$ | $C_1$ | $C_2$ | $C_3$ | $O_4$ | $D_5$ | $D_6$ | $D_7$ |
| $C_0\,C_1\,C_2\,C_3/S_4\,D_5\,D_6\,D_7$ | 10 | 0x33 | $C_0$ | $C_1$ | $C_2$ | $C_3$ | | $D_5$ | $D_6$ | $D_7$ |
| $O_0\,D_1\,D_2\,D_3/S_4\,D_5\,D_6\,D_7$ | 10 | 0x66 | $D_1$ | $D_2$ | $D_3$ | $O_0$ | | $D_5$ | $D_6$ | $D_7$ |
| $O_0\,D_1\,D_2\,D_3/O_4\,D_5\,D_6\,D_7$ | 10 | 0x55 | $D_1$ | $D_2$ | $D_3$ | $O_0$ | $O_4$ | $D_5$ | $D_6$ | $D_7$ |
| $S_0\,D_1\,D_2\,D_3/D_4\,D_5\,D_6\,D_7$ | 10 | 0x78 | $D_1$ | $D_2$ | $D_3$ | $D_4$ | | $D_5$ | $D_6$ | $D_7$ |
| $O_0\,D_1\,D_2\,D_3/C_4\,C_5\,C_6\,C_7$ | 10 | 0x4b | $D_1$ | $D_2$ | $D_3$ | $O_0$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $T_0\,C_1\,C_2\,C_3/C_4\,C_5\,C_6\,C_7$ | 10 | 0x87 | | $C_1$ | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $D_0\,T_1\,C_2\,C_3/C_4\,C_5\,C_6\,C_7$ | 10 | 0x99 | $D_0$ | | $C_2$ | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $D_0\,D_1\,T_2\,C_3/C_4\,C_5\,C_6\,C_7$ | 10 | 0xaa | $D_0$ | $D_1$ | | $C_3$ | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $D_0\,D_1\,D_2\,T_3/C_4\,C_5\,C_6\,C_7$ | 10 | 0xb4 | $D_0$ | $D_1$ | $D_2$ | | $C_4$ | $C_5$ | $C_6$ | $C_7$ |
| $D_0\,D_1\,D_2\,D_3/T_4\,C_5\,C_6\,C_7$ | 10 | 0xcc | $D_0$ | $D_1$ | $D_2$ | $D_3$ | | $C_5$ | $C_6$ | $C_7$ |
| $D_0\,D_1\,D_2\,D_3/D_4\,T_5\,C_6\,C_7$ | 10 | 0xd2 | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | | $C_6$ | $C_7$ |
| $D_0\,D_1\,D_2\,D_3/D_4\,D_5\,T_6\,C_7$ | 10 | 0xe1 | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | | $C_7$ |
| $D_0\,D_1\,D_2\,D_3/D_4\,D_5\,D_6\,T_7$ | 10 | 0xff | $D_0$ | $D_1$ | $D_2$ | $D_3$ | $D_4$ | $D_5$ | $D_6$ | |

TABLE II
LINK FAULT SIGNALING ORDERED SETS AND NEW PROPOSED ORDERED SETS

| LANE 0 | LANE 1 | LANE 2 | LANE 3 | DESCRIPTION |
|---|---|---|---|---|
| Sequence | 0x00 | 0x00 | 0x00 | Reserved |
| Sequence | 0x00 | 0x00 | 0x01 | Local Fault |
| Sequence | 0x00 | 0x00 | 0x02 | Remote Fault |
| Sequence | 0x00 | 0x00 | 0x03 | Link Interruption |
| **Sequence** | **0x00** | **0x00** | **0x04** | **Cipher ON** |
| **Sequence** | **0x00** | **0x00** | **0x05** | **Cipher OFF** |
| Sequence | ≥0x00 | ≥0x00 | ≥0x06 | Reserved |

In order to maintain the concordance with the standard, the format of the implemented control sequence is equivalent to that of an ordered set. Among available 64b/66b block types, the one selected for this work is 0x55. As shown in Table I, inside this block type two consecutive ordered sets can be transmitted.

On the other hand, the possible ordered sets established in the standard are between values 0x00 and 0x03 and they are used for link fault signaling. Since values above 0x03 are out of use and are reserved for future standardization, in this work the new ordered sets for encryption control signaling have been defined as shown in Table II. They have been called 'Cipher ON' and 'Cipher OFF', and their values are 0x04 and 0x05, respectively.

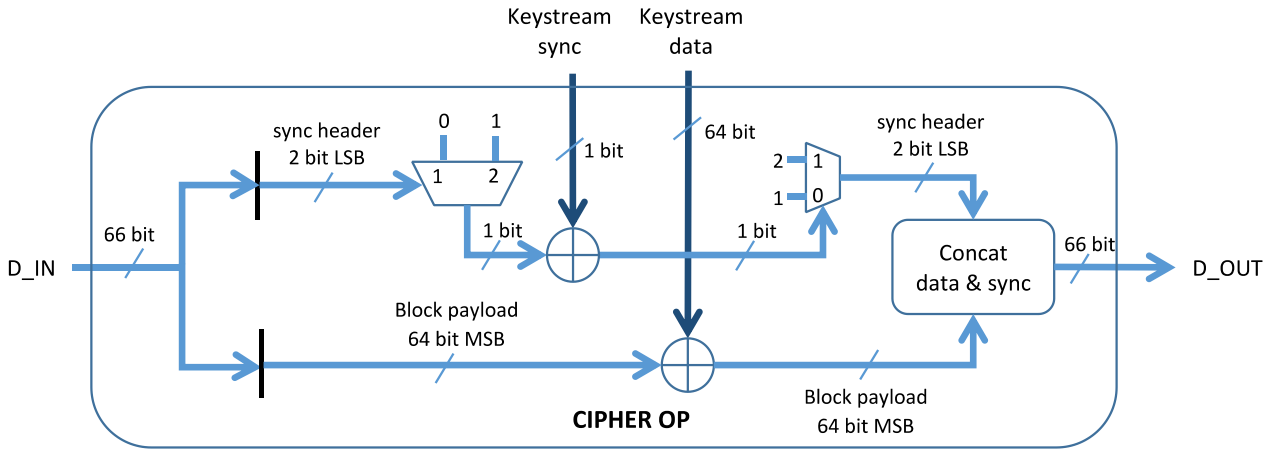



Fig. 4. Stream cipher operation.

The final encryption sequence for enabling the encryption consists of the 0x55 block type filled with two consecutive 'Cipher ON' ordered sets. In this work this block is called 'Cipher_ON block'. For disabling encryption the sequence is the same but using two 'Cipher OFF' ordered sets and the resulting block is called 'Cipher_OFF block'.

For carrying out the insertion of these new control sequences the structure shown in Fig. 3 has been used. This infrastructure allows carrying out the generation, insertion and capture of a future set of messages that extends the encryption management functionality, such as key refresh or other options between transmitter and receiver. The basic operation of this infrastructure is described below.

The CIPHER_OP_TX module is responsible for implementing the encryption operation. Initially, it is disabled, allowing the 64b/66b blocks to be transparently passed from the encoder to the scrambler. To start the encryption of the 64b/66b block stream, the MANAGEMENT module acts on the INSERT module to send the encryption start message. This message is inserted in 64b/66b block stream by replacing one 0x1E type block filled with IDLE control characters /I/ with the new Cipher_ON block. When CAPTURE module detects the presence of a Cipher_ON block it enables the CIPHER_OP_TX module and TX keystream generators (KEYSTREAM TX SYNC and KEYSTREAM TX DATA in Fig. 3), which starts the encryption process after Cipher_ON block has been transmitted.

In the receiver, the module CIPHER_OP_RX is in charge of performing the decryption operation. Like the transmitter, it is initially inactive, enabling 64b/66b blocks to be transparently passed from the descrambler to the decoder. When the CAPTURE module receives the Cipher_ON block, CIPHER_OP_RX module and RX keystream generator modules (KEYSTREAM RX SYNC and KEYSTREAM RX DATA in Fig. 3) are enabled, starting to decrypt the 64b/66b data flow after Cipher_ON block. Subsequently, the control sequence Cipher_ON is extracted from the data stream in the EXTRACT module, which replaces it with a 0x1e type block filled with /I/ control characters (reverse operation of the INSERT module). Thanks to this procedure, the keystream generators in TX and RX are synchronized and data can be deciphered correctly.

In order to disable the encryption, the same process is used, but sending Cipher_OFF blocks instead of Cipher_ON.

## IV. Encryption Keystream

The keystream generator is responsible for generating a pseudorandom sequence to carry out the encryption of the signal. In this work the generation of this pseudorandom sequence is based on a chaotic algorithm.

As the encryption operation consists of two XOR operations, one for block payload and another one for the synchronization header, two keystream generators are necessary. Both generators have different output widths, the generator for block payload encryption produces a 64-bit sequence, while the keystream generator for synchronization header is only 1-bit wide, as shown in Fig. 4.

Next, a description of the keystream generator is given, and also some security considerations are discussed, such as the achieved keystream randomness or the key space used.

### A. Chaotic Pseudorandom Bitstream Generator

In the last decades chaotic systems have been taken into account for the design of new cryptographic systems. Their high sensitivity to the initial conditions or control parameters (also called 'butterfly effect') and their unpredictable pseudo-random orbits are the most well-known characteristics of chaotic systems. These characteristics are very similar to those that a good cryptographic algorithm should have [18]. All these considerations have given rise to new private and public encryption proposals [19], [20].

A variant of the chaotic map called STM (Skew-Tent-Map) has been used in this work to generate the pseudorandom bitstream, used as keystream of the proposed cryptographic system. Based on previous work by our group, this chaotic map has already been theoretically studied in [21] and [22]. The hardware block diagram of the basic STM algorithm is shown in Fig. 5 and its equations are shown below:

$$f(x_i) = x_{i+1} = \begin{cases} x_i/\gamma, & x_i \in [0, \gamma] \\ (1 - x_i)/(1 - \gamma), & x_i \in (\gamma, 1] \end{cases} \quad (1)$$

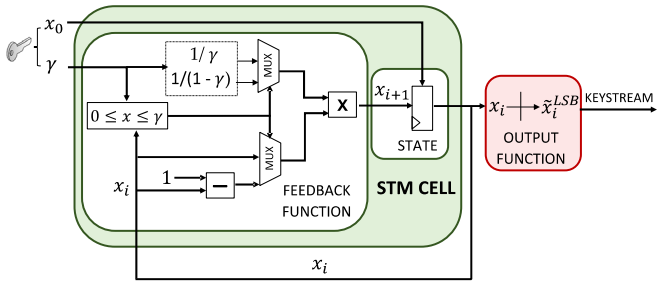



Fig. 5. Skew Tent Map block diagram.

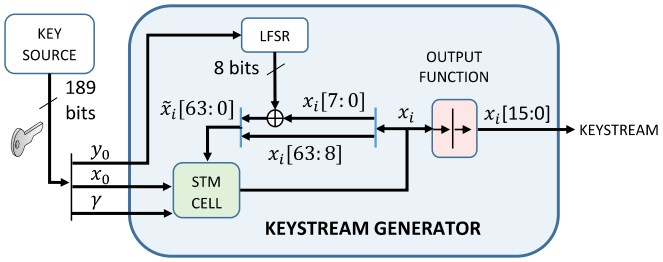



Fig. 6. Basic keystream generator. The 189-bit key is the concatenation of the three parameters $(y_0, x_0, \gamma)$.

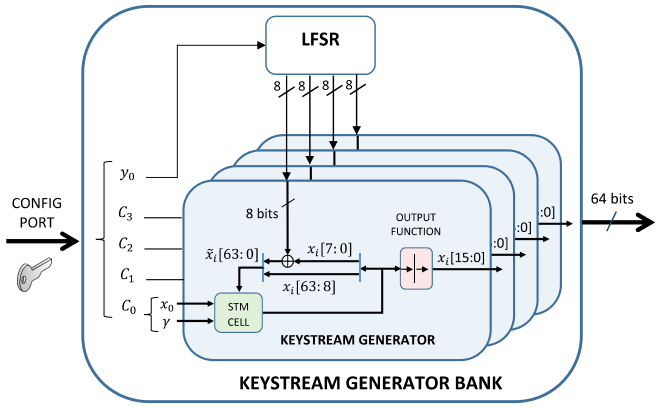



Fig. 7. 64-Bit Keystream Generator for 64b/66b block payload.

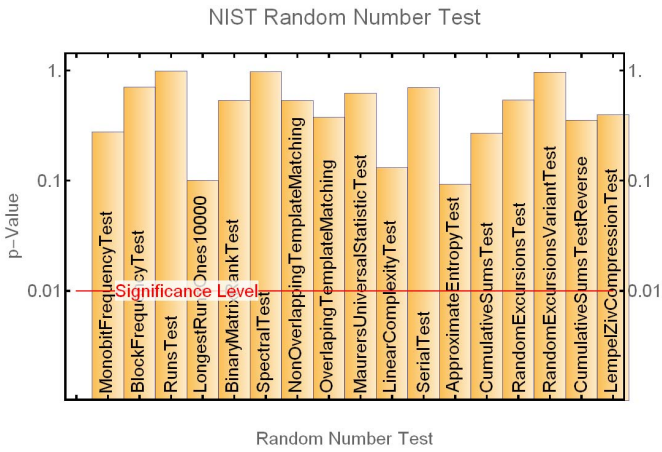



Fig. 8. NIST test results for a bitstream generated using STM-LFSR algorithm.

where its control parameter $\gamma$ and initial state $x_0$ are real numbers whose values are included in the interval (0, 1). One of the main advantages of this chaotic map is that it maintains its chaoticity for all values of $\gamma$ and $x_0$, which makes it more adequate for cryptographic applications than other maps whose parameter space contains periodic windows [18].

One important problem of digital implementations of chaotic maps under finite-precision and fixed-point arithmetic is that their dynamical properties are degraded with respect to their original continuous versions [23]. As studied in [21] and [22] an LFSR (Linear Feedback Shift Register) can be used to improve the digital implementation of the chaotic map shown in (1), reducing the mentioned dynamical degradation. Thanks to this modification it is possible to obtain an important increase in the keystream period and therefore achieve better randomness [24].

The basic keystream generator used in this work is shown in Fig. 6. A 61-step LFSR has been added to the Skew Tent Map module. The key for this basic keystream generator consists of the initial state and control parameter $(\gamma, x_0)$, of the STM cell and the initial state of the LFSR $(y_0)$.

The STM cell has been implemented with an internal 64-bit state whose output is the vector $x_i[63:0]$. The improvement related to the LFSR is to perform the XOR operation between the least significant 8 bits of LFSR and the STM cell outputs. The state returned to the STM cell will no longer be the previous state calculated by it, but a new state, $\tilde{x}_i$ to which a small noise generated by the LFSR has been added thanks to the XOR operation. The generator output function consists of taking the 16 least significant bits of $x_i$ and discarding the rest.

For the payload block encryption, a 64-bit keystream sequence is necessary, thus a bank of basic keystream generators has been built. The bank consists of four 16-bit generators whose outputs are concatenated to give a 64-bit output. This structure is shown in Fig. 7.

For the synchronization header encryption, as only a 1-bit keystream sequence is needed the basic STM cell in Fig. 6 has been used, but in this case only the least significant bit of $x_i$ is taken as output.

### *B. Keystream Output Analysis*

The keystream obtained at the output of both generators must be analyzed in order to test their randomness, as one important requirement for a secure stream cipher is that their keystream must be undistinguishable from a truly random sequence. In order to check this property, sequences generated by the basic generator in Fig. 6 were subjected to the NIST (National Institute of Standards and Technology) SP 800-22 battery of test [25]. Both generators passed these tests. An example is shown in Fig. 8, where a particular sequence is tested.

### *C. Security Considerations: Key Space*

One of the security considerations that has to be taken into account is the key space size [18]. Some recommendations establish 112 bits as minimum key size, as NIST

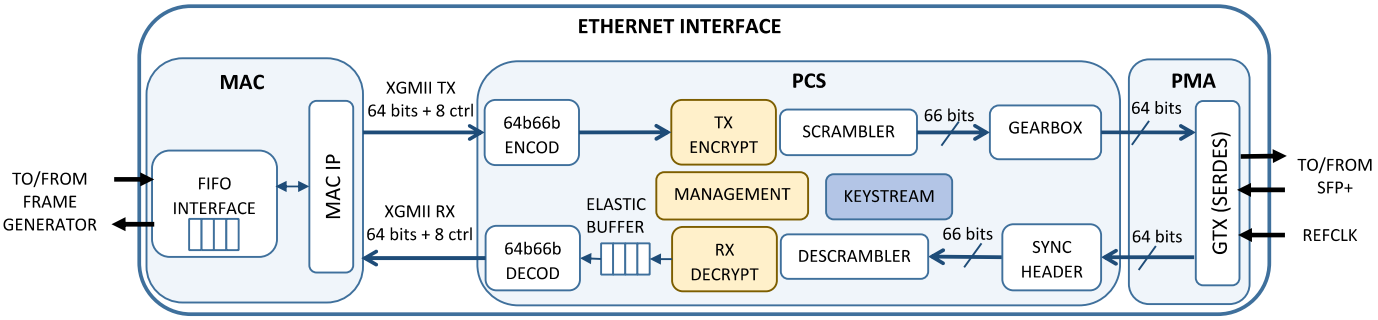



Fig. 9. Scheme of FPGA implementation for 10G Ethernet interface with encryption function.

recommendation [26], others as ENISA (European Union Agency for Network and Information Security) report [27] recommends 128 or 256 bits for mid-term and long-term security respectively.

For the basic keystream generator in Fig. 6, the complete key is composed by two 64-bit parameters $(\gamma, x_0)$ and one 61-bit parameter $y_0$. The resulting key is 189-bit length, which can be considered good enough to guarantee security against brute-force attacks.

### D. Security Considerations: Sensitivity

Sensitivity to the initial conditions and control parameters is a well-known characteristic of chaotic maps. It leads to the desired diffusion property that any cryptosystem must have under small changes in the secret key. While in the proposed cryptosystem, the STM cell meets this property, sensitivity to plaintext and ciphertext changes is not achieved. It means that as other stream ciphers, a single bit change on the plaintext only changes one bit on the ciphertext and vice versa. Therefore if the keystream generator was restarted with the same key a successful differential known-plaintext attack could be applied. This fact must be taken into account, above all when continuous IDLE sets are being transmitted over the link.

### E. Security Considerations: Map Reconstruction Attack

One important security problem related with chaotic systems is the possibility of reconstructing the map and inferring the control parameters. This type of attack can be performed by analyzing the output sequence and plotting $x_{i+1}$ versus $x_i$. As an example, if the attacker knows the value of $x_i$ and $x_{i+1}$ , he could use (1) for $f(x_{i-1})$ and $f(x_i)$ to calculate the values of $\gamma$ and $x_{i-1}$. Solutions for this kind of attacks are to shuffle/truncate the chaotic orbit before using it for encryption [28].

The fact that we are using only the least 16 significant bits from each $x_i$ for the encryption, limits the information revealed to an attacker. If an attacker could know the least significant 16 bits of $x_i$ and $x_{i+1}$, there would still be $2^{48}$ possible values for each $x_i$ or $x_{i+1}$, then $2^{96}$ for the combination of both.

In addition, the eight least significant bits of each $x_i$ are XORed with the LFSR output which makes the map reconstruction more difficult.

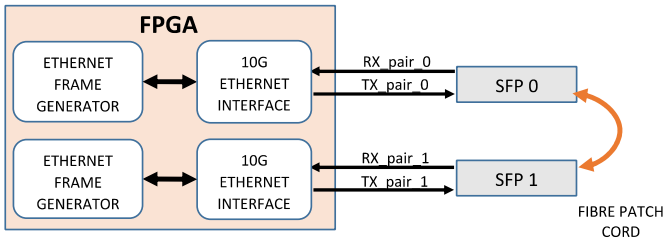



Fig. 10. Test setup scheme.

### F. Security Considerations: Other Considerations

Initial key configuration and key refreshing are important mechanisms in any cryptographic system. In this work, keys have been preconfigured in transmitter and receiver thanks to the FPGA debug system. The implementation of the key management procedure based on 64b/66b control blocks has been let as future work.

As mentioned in Section III, a future set of control messages could be expanded thanks to encryption infrastructure shown in Fig. 3. In order to avoid malicious manipulation or generation of these kinds of messages, a MAC (Message Authentication Control) functionality also should be implemented for the future protocol.

## V. System Implementation

### A. System Description

The overall system has been implemented in a Xilinx Virtex 7 FPGA. In the setup for test, the FPGA has been connected to two SFP+ (Small Form-Factor Pluggable) modules capable of transmitting at a rate of 10.3125 Gbps at 850 nm over multimode fiber. The FPGA design consists of two 10G Ethernet interfaces including the 10G-PHYsec function and two Ethernet Frame Generator modules connected to them. The structure of the 10G Ethernet Interfaces is shown in Fig 9. It includes the MAC module and the PHY (PCS and PMA blocks). The PHY is connected directly to the SFP+ modules thanks to the FPGA SERDES (Serializer-Deserializer) circuit. In the MAC side, Ethernet Interface is connected to the Ethernet Frame Generator to test the encrypted link with real traffic. It allows to verify that no frames are lost and no CRC (Cyclic Redundancy Check) errors are produced during encryption.

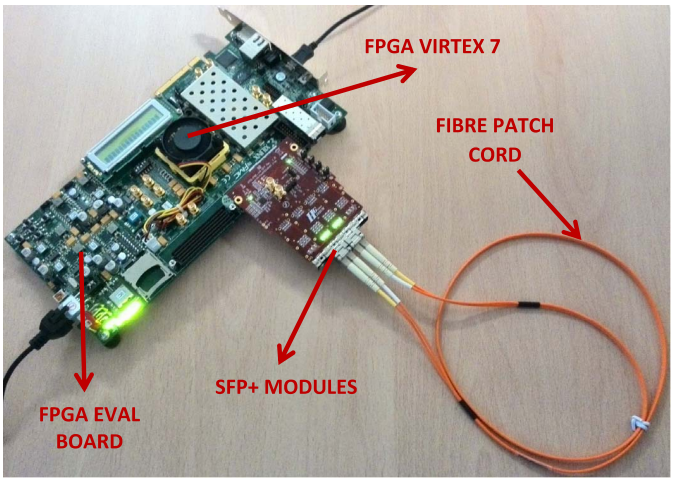



Fig. 11. Test setup photo with SFP+ modules working at 10 Gbps rate.

In Fig. 10 and Fig. 11 the diagram and a photo of the hardware test setup are shown, respectively. Since the key can be configured separately for TX and RX encryption modules in each interface, it is possible to test the bidirectional link with different keys in each direction.

### *B. Encryption Results*

The conclusions drawn from simulation and hardware debugging can be summarized in the following points:

1) Encryption/Decryption works correctly and synchronously without harming data traffic or link establishment between 10G Ethernet interfaces. Thanks to the frame and CRC counters inside Ethernet Frame Generators it has been possible to check that neither frame losses nor CRC errors are produced. Particularly, bursts of 1024-byte length frames were tested with a duration up to $10^6$ frames and a throughput between 10% and 98% of the maximum line rate.
2) Transmitted frames are indecipherable thanks to encryption. When encryption keys are different between transmitter and receiver, no valid frames are received in the receiver because it is impossible to recover the right 64b/66b block flow.
3) When encrypting, data traffic pattern is indistinguishable from a continuous IDLE pattern, then it is able to hide the pattern of Ethernet traffic from any malicious observer.

For this last capability it is interesting to know the signal behavior at the input of the scrambler and output of the descrambler. Particularly, the block synchronization header can give information about the transmission state.

If the encryption is not enabled, the synchronization header takes the value "10" when transmitting Ethernet frames and "01" during the IFG (Inter Frame Gap) periods and frame boundaries. If no frame is transmitted, the synchronization header is a continuous value "01", as IDLE control characters are being continuously transmitted.

When encryption is enabled, the synchronization header switches randomly between "01" and "10" and the same happens with the 64b/66b block payload that is randomized thanks to the keystream. This happens in both situations, with or without Ethernet traffic being transmitted over the link. This effect makes the data traffic pattern indistinguishable.

TABLE III
FPGA RESOURCES USED BY PCS MODULE

| MODULE | Slice LUTs | Slice Registers | DSPs |
|---|---|---|---|
| PCS (without cipher) | 3601 | 3726 | 0 |
| PCS (with cipher) | 13701 | 8641 | 160 |

TABLE IV
FPGA RESOURCES USED BY ENCRYPTION INFRASTRUCTURE AND KEYSTREAM GENERATORS IN RX AND TX

| MODULE | Slice LUTs | Slice Registers | DSPs |
|---|---|---|---|
| ENCRYP INFRASTR | 276 | 293 | 0 |
| KEYSTREAM RX | 4687 | 2271 | 80 |
| KEYSTREAM TX | 4680 | 2271 | 80 |

TABLE V
FPGA RESOURCES USED IN KEYSTREAM GENERATOR SUBMODULES

| MODULE | Slice LUTs | Slice Registers | DSPs |
|---|---|---|---|
| KEYSTREAM_GEN | 4680 | 2271 | 80 |
| LFSR | 202 | 186 | 0 |
| STM_BANK | 3534 | 1569 | 64 |
| STM_1BIT | 948 | 516 | 16 |

### *C. Implementation Results*

Regarding the resources used by the PCS layer, it is interesting to study the increase in resources needed to implement the encryption features. In Table III, post-synthesis resource estimation is shown. When encryption is included it is possible to see a large increase in LUTs (Look-Up Table), registers and DSP (Digital Signal Processing) cells. Approximately this increment consists of 10100 LUTs, 4915 registers and 160 DSP cells and is distributed among the RX/TX keystream generators and the encryption infrastructure, as shown in Table IV. In Table V, resources for keystream generator (KEYSTREAM GEN) are decomposed into the 64- bit keystream generator (STM BANK) plus LFSR module used in block payload encryption and the 1-bit keystream generator (STM_1BIT) for block synchronization header encryption.

In the case of DSP cells, 16 of them are necessary to implement the multiplications inside each STM_CELL, as the chaotic map implementation requires it. A total of 80 DSP cells are necessary for the complete keystream generator.

### *D. Comparison With Other Solutions*

In Table VI a resource and throughput comparison with other chaotic encryption solutions is shown. This comparison

TABLE VI
CHAOTIC CELL COMPARISON WITH OTHER SOLUTIONS

| | Application | Platform | FF | LUTs | DSP cells | Max Freq. (MHz) | Output bits/cycle | Internal state (bits) | Key size (bits) | Encryption Rate (Mbps) | Encryption Rate/Register (Mbps/Register) | Encryption Rate/LUT (Mbps/LUT) | Encryption Rate/DSP (Mbps/DSP) |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **LM [30]** | PRNG | Virtex-5 | 64 | 129 | 16 | 26.9 | 1 | 64 | 64 | 26.9 | 0.42 | 0.21 | 1.7 |
| **MLM [29]** | Encryption | Virtex-6 | 160 | 643 | 16 | 93 | 16 | 64 | 192 | 1488 | 9.3 | 2.31 | 93 |
| **GFLM [33]** | Encryption | Virtex-5 | 28 | 309 | 3 | 58.4 | 20 | 20 | 60 | 1168* | 41 | 3.77 | 389 |
| **Bernoulli [31]** | PRNG | Spartan 3E | 108 | 575 | 9 | 36.9 | 1/5 | 52 | 156 | 7.38 | 0.07 | 0.01 | 0.82 |
| **Chua [32]** | PRNG | Artix-7 | 787 | 986 | 64 | 80 | 96 | 96 | - | 7685* | 9.76 | 7.79 | 120 |
| **CIPRNG-XOR [34]** | PRNG | Artix-7 | 582 | 345 | 0 | 257.5 | 32 | 32 | - | 8240* | 14.1 | 23.88 | NA |
| **GCIPRNG-F2 [35]** | PRNG | Artix-7 | 444 | 415 | 0 | 205.7 | 32 | 32 | - | 6580* | 14.8 | 15.85 | NA |
| **This Work** | Encryption | Virtex-7 | 64 | 858 | 16 | 175.4 | 16 | 64 | 189 | 2806 | 43.8 | 3.27 | 175 |

*These solutions achieve their Encryption_rate at expense of using their complete state as output, which can reduce the security.

is only illustrative as the FPGA device used in each implementation is different. In this work Virtex-7 is used, however in the other solutions different devices are used, such as Artix-7, Virtex-6, Virtex-5 and Spartan-3E. While CLB structure in Artix-7, Virtex-6 and Virtex-7 is similar in terms of LUTs and registers, with four 6-input LUTs and 8 registers per slice, in Virtex-5 each slice contains four 6-input LUTs and 4 registers. The difference is higher with Spartan-3E, a much older device with only 2 LUTs and 2 register per slice. For this reason the comparison is made in terms of *Encryption_rate/register* and *Encryption_rate/LUT*.

As a comparison among different chaotic cells is a difficult task, we have included in Table VI not only their hardware resources and encryption throughput but also some parameters relative to their structure, such as the internal state and output bit width, and the total key/parameters length. Assuming that every chaotic cell passes the randomness tests, authors consider that the mentioned parameters are important as a measure of the overall security provided by them. Indeed, some of the solutions shown in Table VI are proposed as encryption algorithms while others are used only as PRNGs (Pseudo Random Number Generator) for cryptographic applications, such as key generation.

On the one hand, key size shows how secure can be a cell against brute-force attacks, while the bit width difference between the internal state and output shows how secure can be against a map reconstruction attack. Both parameters affect the results in the hardware resources and encryption rate shown in Table VI. In particular, a longer key and internal state mean that the overall hardware resources are increased, since the width of arithmetic operands inside the cell also increases. In the same way the maximum operation frequency achieved by the cell is decreased, which reduces the final throughput. On the other hand, the opposite effect happens with the output width, a longer output means higher throughput, as more bits per cycle are available for being used in the encryption process.

For example, in [29] the resulting chaotic cell with a 192-bit key obtains its output from the 16 least significant bits of its 64-bit internal state, while in [30], the chaotic cell, also with a 64-bit state, uses only as output one of its bits, however it has a key of only 64 bits. In [31], among the several maps that are implemented, we have selected Bernoulli for the comparsion, it has a 52-bit state and only outputs one bit every 5 clock cycles. Other implementations, as [32], and [33] use their complete state as output, with 96 and 20 bits, respectively. Regarding [34] and [35], they use a chaotic iteration post-processing technique to improve the randomness of linear PRNGs. These do not need DSP cells, however the complete output of the chaotic post-processing is used as output.

According to these parameters, key size, internal state and output bits, the proposed work, together with [29] and [31], presents the best security, with a key of more than 128 bits and less than the 25% of its internal state bits revealed.

In terms of *Encryption_rate* and *Encryption_rate/resource* the implementation in this work clearly achieves a good result. Although [32], [34] and [35] present better *Encryption_rate*, it is at expense of using their complete state as output, which reduces the security. Indeed they are mainly proposed as PRNGs.

Regarding to the DSP cells used, the proposed solution needs a higher quantity than others, as in [31] and [33]. However, it is also at the expense of reducing other parameters. For example in both solutions *Encryption_rate* is lower than this work. Moreover, in [33] the complete state is taken as output and the key size is only 60 bits, which disminishes the security achieved by this solution.

Finally, if we compare this work with [29], the most similar in terms of security, this work achieves better *Encryption_rate* figures. However, it is important to remark that the FPGA used in [29] is a Virtex-6, a 40 nm device, while Virtex-7 used in this work is a 28 nm device. Because of that, the proposed STM cell has been also implemented configuring the target device as the same in [29]. In this case the obtained hardware

resources are similar to those used in Virtex-7 but resulting in a lower maximum operation frequency of 154.2 MHz. This fact reduces the *Encryption_rate* but anyway it is still higher that in [29].

## VI. Conclusion

As far as the authors are aware, this is the first time that a 10GBase-R Ethernet physical layer encryption method has been proposed and developed. The proposed new encryption function 10G-PHYsec consists of symmetric ciphering at PCS sublayer of the 64b/66b block stream transmitted over an optical link. Encryption based on an original chaotic cipher has been tested with real Ethernet traffic and it has been concluded that the proposed system works correctly without harming data traffic or link establishment, making Ethernet frames indecipherable and obfuscating completely the data traffic patterns. These features improve the security at physical level with no throughput losses, null overhead and low latency, and it is compatible with other solutions at other layers where data packets must be modified by adding extra overhead and delay during their transmission.

Another advantage of the proposed method is that by preserving 64b/66b coding properties as DC-balance, short run length, and transmission density, physical layer encryption is achieved without the necessity of change any of subsequent hardware elements in the TX/RX chain, as SERDES (Serializer/Deserializer), CDR and driver circuits or optical components as commercial SFP modules. In addition, as this method is based in the 64b/66b encoding, it could be scaled to other standards with higher transmission rates based also in this encoding method, such as in 40 or 100 Gigabit Ethernet.

Although it is necessary an increment of FPGA resources for the proposed physical layer modifications, it mainly lies with the keystream generator module and entails an encryption throughput better than other solutions based on chaotic maps.

Finally, we are working on improving the encryption protocol to add functionalities such as new control messages for temporary key refresh and some mechanism for message authentication control.

## References


[1] *Cisco Global Cloud Index: Forecast and Methodology*, Cisco Systems, San Jose, CA, USA, 2016, pp. 2015–2020.

[2] S. V. Kartalopoulos, "Discriminating between faults and attacks in secure optical networks," in *Proc. MILCOM*, Orlando, FL, USA, Oct. 2007, pp. 1–5.

[3] S. V. Kartalopoulos, "Optical network security: Sensing eavesdropper intervention," in *Proc. IEEE GLOBECOM*, San Francisco, CA, USA, Nov./Dec. 2006, pp. 1–6.

[4] N. Skorin-Kapov, M. Furdek, S. Zsigmond, and L. Wosinska, "Physical-layer security in evolving optical networks," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 110–117, Aug. 2016.

[5] M. Furdek, N. Skorin-Kapov, S. Zsigmond, and L. Wosinska, "Vulnerabilities and security issues in optical networks," in *Proc. 16th Int. Conf. Transparent Opt. Netw. (ICTON)*, Graz, Austria, Jul. 2014, pp. 1–4.

[6] M. Zafar, H. Fathallah, and N. Belhadj, "Optical fiber tapping: Methods and precautions," in *Proc. 8th Int. Conf. High-Capacity Opt. Netw. Emerg. Technol.*, Riyadh, Saudi Arabia, Dec. 2011, pp. 164–168.

[7] *IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security*, IEEE Standard 802.1AE, 2006.

[8] S. Kent and K. Seo. (2005). *RFC 4301: Security Architecture for the Internet Protocol*. [Online]. Available: https://tools.ietf.org/html/rfc4301

[9] D. Elkouss, J. Martinez-Mateo, A. Ciurana, and V. Martin, "Secure optical networks based on quantum key distribution and weakly trusted repeaters," *J. Opt. Commun. Netw.*, vol. 5, no. 4, pp. 316–328, Apr. 2013.

[10] J. Ji, G. Zhang, W. Li, L. Sun, K. Wang, and M. Xu, "Performance analysis of physical-layer security in an OCDMA-based wiretap channel," *J. Opt. Commun. Netw.*, vol. 9, no. 10, pp. 813–818, Oct. 2017.

[11] K. Guan, J. Kakande, and J. Cho, "On deploying encryption solutions to provide secure transport-as-a-service (TaaS) in core and metro networks," in *Proc. 42nd Eur. Conf. Opt. Commun.*, vol. 22, Sep. 2016, pp. 1–3.

[12] *In-Flight Encryption in Service Provider Networks*, document PMC-2150716, MicroSemi, 2016, no. 2.

[13] L. Troell, J. Burns, K. Chapman, D. Goddard, M. Soderlund, and C. Ward, "Converged vs. dedicated IPSec encryption testing in gigabit Ethernet networks," Rochester Inst. Technol., Rochester, NY, USA, Tech. Rep. 1743, 2005. [Online]. Available: http://scholarworks.rit.edu/article/1743

[14] "Delivering ubiquitous Ethernet services using an array of access technologies," Metro Ethernet Forum, Los Angeles, CA, USA, Tech. Rep., 2009.

[15] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma, "Using a chaotic cipher to encrypt Ethernet traffic," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Florence, Italy, May 2018, pp. 1–5.

[16] S. Choi *et al.*, "A 0.65-to-10.5 Gb/s reference-less CDR with asynchronous baud-rate sampling for frequency acquisition and adaptive equalization," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 2, pp. 276–287, Feb. 2016.

[17] K. Lee and J.-Y. Sim, "A 0.8-to-6.5 Gb/s continuous-rate reference-less digital CDR with half-rate common-mode clock-embedded signaling," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 63, no. 4, pp. 482–493, Apr. 2016.

[18] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. Bifurcation Chaos*, vol. 16, no. 8, pp. 2129–2151, 2006.

[19] R. Bose and S. Pathak, "A novel compression and encryption scheme using variable model arithmetic coding and coupled chaotic system," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 53, no. 4, pp. 848–857, Apr. 2006.

[20] P. Bergamo, P. D'Arco, A. De Santis, and L. Kocarev, "Security of public-key cryptosystems based on chebyshev polynomials," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 52, no. 7, pp. 1382–1393, Jul. 2005.

[21] M. Garcia-Bosque, C. Sánchez-Azqueta, A. Pérez, A. D. Martínez, and S. Celma, "Fast and secure chaotic stream cipher with a MEMS-based seed generator," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, May 2017, pp. 1–6.

[22] M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, and S. Celma, "Application of a MEMS-based TRNG in a chaotic stream cipher," *Sensors*, vol. 17, no. 3, p. 646, 2017.

[23] S. Li, G. Chen, and X. Mou, "On the dynamical degradation of digital piecewise linear chaotic maps," *Int. J. Bifurcation Chaos*, vol. 15, no. 10, pp. 3119–3151, 2005.

[24] L. Kocarev and S. Lian, "Digitized chaos for pseudo-random number generation in cryptography," in *Chaos-Based Cryptography*. Berlin, Germany: Springer, 2009, pp. 67–97.

[25] A. Rukhin *et al.*, "A statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST-Inf. Technol., Gaithersburg, MD, USA, Tech. Rep. NIST SP 800-22 Rev.1a, 2010.

[26] E. Barker and A. Roginsky, *Transitions: Recommendation for Transitioning the Use of Cryptographic Algorithms and Key Lengths*. Gaithersburg, MD, USA: NIST Special Publication 800-131A, 2015.

[27] ENISA. (Nov. 2014). *Algorithms, Key Size and Parameters Report*. Accessed: Dec. 11, 2017. [Online]. Available: www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014

[28] G. Alvarez, J. M. Amigó, D. Arroyo, and S. Li, "Lessons learnt from the cryptanalysis of chaos-based ciphers," in *Chaos-Based Cryptography*. Berlin, Germany, Springer-Verlag, 2011, pp. 257–295.

[29] A. Pande and J. Zambreno, "A chaotic encryption scheme for real-time embedded systems: Design and implementation," *Telecommun. Syst.*, vol. 52, no. 2, pp. 551–561, 2013.

[30] P. Dabal and R. Pelka, "A chaos-based pseudo-random bit generator implemented in FPGA device," in *Proc. Int. Symp. Design Diagnosis Electron. Circuits Syst. (DDECS)*, Cottbus, Germany, Apr. 2011, pp. 151–154.

[31] L. De la Fraga, E. Torres-Pérez, E. Tlelo-Cuautle, and C. Mancillas-López, "Hardware implementation of pseudo-random number generators based on chaotic maps," *Nonlinear Dyn.*, vol. 90, pp. 1661–1670, Nov. 2017.
[32] A. J. A. El-Maksoud *et al.*, "FPGA implementation of fractional-order Chua's chaotic system," in *Proc. 7th Int. Conf. Mod. Circuits Syst. Technol. (MOCAST)*, Thessaloniki, Greece, May 2018, pp. 1–4.
[33] S. M. Ismail *et al.*, "Generalized fractional logistic map encryption system based on FPGA," *AEU—Int. J. Electron. Commun.*, vol. 80, pp. 114–126, Oct. 2017.
[34] M. Bakiri, J.-F. Couchot, and C. Guyeux, "CIPRNG: A VLSI family of chaotic iterations post-processings for F2 -linear pseudorandom number generation based on Zynq MPSoC," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 65, no. 5, pp. 1628–1641, May 2018.
[35] M. Bakiri, C. Guyeux, J.-F. Couchot, L. Marangio, and S. Galatolo, "A hardware and secure pseudorandom generator for constrained devices," *IEEE Trans. Ind. Informat.*, vol. 14, no. 8, pp. 3754–3765, Aug. 2018.

**Adrián Pérez-Resa** was born in San Sebastián, Spain. He received the M.Sc. degree in telecommunications engineering from the University of Zaragoza, Zaragoza, Spain, in 2005.

He is currently pursuing the Ph.D. degree with the Group of Electronic Design, Aragon Institute of Engineering Research, University of Zaragoza.

He was an R&D engineer with telecommunications industry for over 10 years. He is also a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include high-speed communications and cryptography applications.

**Miguel Garcia-Bosque** was born in Zaragoza, Spain. He received the B.Sc. and M.Sc. degrees in physics from the University of Zaragoza, Zaragoza, in 2014 and 2015, respectively, where he is currently pursuing the Ph.D. degree with the Group of Electronic Design, Aragon Institute of Engineering Research.

He is also a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include chaos theory and cryptography algorithms.

**Carlos Sánchez-Azqueta** was born in Zaragoza, Spain. He received the B.Sc., M.Sc., and Ph.D. degrees in physics from the University of Zaragoza, Zaragoza, in 2006, 2010, and 2012, respectively, and the Dipl.-Ing. degree in electronic engineering from the Complutense University of Madrid, Madrid, Spain, in 2009.

He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include mixed-signal integrated circuits, high-frequency analog communications, and cryptography applications.

**Santiago Celma** was born in Zaragoza, Spain. He received the B.Sc., M.Sc., and Ph.D. degrees in physics from the University of Zaragoza, Zaragoza, Spain, in 1987, 1989, and 1993, respectively.

He is currently a Full Professor with the Group of Electronic Design, Aragon Institute of Engineering Research, University of Zaragoza. He has co-authored over 100 technical papers and 300 international conference contributions. He has co-authored four technical books and holds four patents. He appears as a principal investigator in over 30 national and international research projects. His research interests include circuit theory, mixed-signal integrated circuits, high-frequency communication circuits, wireless sensor networks, and cryptography for secure communications.

# Physical Layer Encryption for Industrial Ethernet in Gigabit Optical Links

Adrián Pérez-Resa, Miguel Garcia-Bosque, Carlos Sánchez-Azqueta, and Santiago Celma

***Abstract*—Industrial Ethernet is a technology widely spread in factory floors and critical infrastructures where a high amount of data need to be collected and transported. Fiber optic networks at gigabit rates fit well with that type of environment, where speed, system performance, and reliability are critical. In this paper, a new encryption method for high-speed optical communications suitable for such kinds of networks is proposed. This new encryption method consists of a symmetric streaming encryption of the 8b/10b data flow at physical coding sublayer level. It is carried out thanks to a format preserving encryption block cipher working in CTR (counter) mode. The overall system has been simulated and implemented in a field programmable gate array. Thanks to experimental results, it can be concluded that it is possible to cipher traffic at this physical level in a secure way. In addition, no overhead is introduced during encryption, getting minimum latency and maximum throughput.**

***Index Terms*—Cryptography, Industrial Ethernet, optical communications, stream cipher.**

## I. Introduction

THANKS to advances in Information Technology during the last decades, communication solutions at automation level in the industrial control systems (ICS) have evolved from legacy field buses to Ethernet technology [1]–[3]. Ethernet has increased significantly its use as communication solution for supervisory control and data acquisition (SCADA) networks. On the other hand, not only data from industrial equipment are supported in SCADA networks, but also data traffic from surveillance video security and future Internet of Things applications. Both kinds of applications demand a high-transmission bandwidth. Therefore, telecommunication equipment adapted for industrial environments with rates up to 1 Gbps and beyond are available from many vendors. A simple scheme of a SCADA network is shown in Fig. 1.

Among different transmission media in Ethernet standards, optical fiber provides the best bandwidth, less signal loses, and more immunity to electromagnetic interference than other wired systems. Owing to these advantages, a lot of research related with multigigabit optical transceivers has been carried out for applications in industrial environments [4], [5].

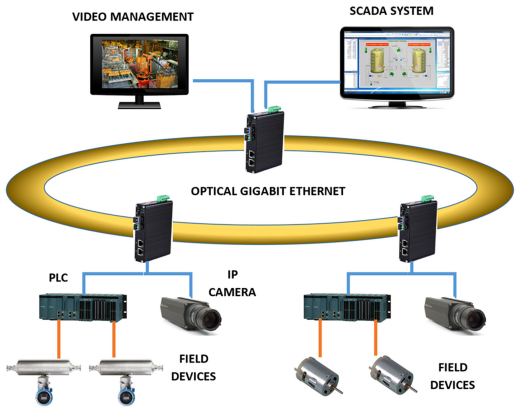



Fig. 1. Simple example of a SCADA network sharing field device traffic and video security.

With respect to wireless systems, optical communications can be considered safer as they do not emit any radiation that can be captured by an intruder. However, the optical signal can be intercepted easily, thanks to fiber coupling devices and electro-optical converters [6]. Moreover, it can be performed without perceptibly interfere in communications.

In layered communication models like OSI (Open Systems Interconnection) or TCP/IP, several encryption techniques can be used to avoid passive eavesdropping. It depends on the communication layer where confidentiality is needed. Many solutions proposed for industrial Ethernet encryption are usually for layers 3 or 4, such as IPsec or transport layer security standards, respectively. These offer security when field buses are attached to IP networks by means of a gateway [7]. Other solutions are proposed for layer 2, such as MACsec standard [8]. Moreover, industrial communication equipment, with layer 2 and layer 3 encryption capabilities, is nowadays available from some vendors, and is used to protect ICS from both internal and external cyberattacks.

Regarding layer 1 encryption in optical communications, there is no proposal for Ethernet. However, there are solutions related with optical technology, such as optical code-division multiplexing [6], or with physical layer protocols, such as the encryption of optical transport network (OTN) frame payloads [9]. The latter is mainly used by service providers to grant security on the carrier telecommunication networks.

Manuscript received February 6, 2018; revised April 13, 2018 and May 15, 2018; accepted June 5, 2018. Date of publication June 21, 2018; date of current version November 30, 2018. This work was supported in part by MINECO-FEDER under Grant TEC2014-52840-R and Grant TEC2017-85867-R and in part by the FPU Fellowship Program to M. Garcia-Bosque under Grant FPU14/03523. *(Corresponding author: Adrián Pérez-Resa.)*

The authors are with the Group of Electronic Design, Aragón Institute of Engineering Research (I3A), University of Zaragoza, 50009 Zaragoza, Spain (e-mail: aprz@unizar.es; mgbosque@unizar.es; csanaz@unizar.es; scelma@unizar.es).

Color versions of one or more of the figures in this paper are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIE.2018.2847670

One of the advantages of performing encryption at layer 1 is that no extra data fields are introduced on ciphering packets, unlike protocols at other layers do [10]. Thanks to this, no throughput is wasted for transmitting this extra overhead. For example, OTN commercial equipment is usually able to perform encryption at line rate, achieving a 100% throughput and introducing very low latency [11], in the range of nanoseconds. It contrasts with other encryption methods, such as IPsec, which usually introduces latencies in the range of milliseconds.

Regarding to Ethernet, 1000Base-X is one of the most widely used physical layer standards at 1 Gbps rate in optical communications, and there is no mechanism providing layer 1 security on it. On the other hand, Gigabit optical Ethernet is widely used in industrial Ethernet networks, where for certain applications a high level of determinism is needed, and there can be strict requirements for network delay and jitter [12]. A physical layer encryption mechanism could provide the mentioned advantages such as zero overhead and low latencies. This kind of 'in-flight' encryption could be useful with real time protocols, such as EtherCAT or PROFINET IRT [12], [13], enabling encryption to be performed at line rate.

The main motivation of this paper is to propose and implement an encryption method suitable for the physical coding sublayer (PCS) of the 1000Base-X standard. The physical coding of this Ethernet layer is the well-known 8b/10b encoding. To get "in-flight" encryption of the 8b/10b symbol stream, a stream cipher that preserves the format of this codification is necessary.

As far as the authors are concerned, there is no standardized or recommended format preserving stream cipher. However, recently, several format preserving encryption (FPE) modes of operation have been approved by National Institute of Standards and Technology (NIST) [14]. The structures described in these modes can be understood as a kind of FPE "block ciphers" [15] that are able to encrypt data in the desired format. Its use in a like of electronic code book mode is proposed to provide security in databases with an arbitrary format [15] or in legacy ICS protocols [16].

In this paper, CTR (counter) mode is proposed to be used with "FPE block ciphers" to achieve security in high-throughput applications where data format must be preserved. For example, the Gigabit Ethernet 1000Base-X physical layer.

This paper is divided into the following sections. In Section II, an introduction to PCS layer encryption necessities is given; in Section III, an introduction to the FPE NIST recommendations is made. In Section IV, security considerations of the proposed structure are given and the analysis of the keystream output is carried out. Subsequently, Section V deals with the practical case for Gigabit Ethernet 1000Base-X layer and the overall structure of the proposed encryption system. In Section VI, the hardware implementation of the cipher is described and results of the encryption are explained. Finally, in Section VII, conclusions are given.

## II. Coding Preserving Encryption

Physical coding sublayer is part of the Ethernet 1000Base-X standard and performs functions such as autonegotiation, link establishment, clock rate adaptation, and data encoding.

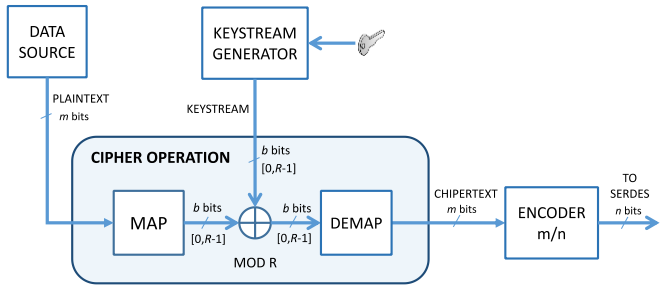



Fig. 2. Location and generic structure of a stream cipher in a physical layer with block line encoding.

As other high-speed standards, a baseband serial data transmission is carried out while clock frequency information is embedded in the serial bitstream itself. Thanks to bit transitions in the data stream, the clock recovery circuits are able to extract the frequency information at the receiver.

Consequently, serial data sampling is made at the appropriate time. In order to facilitate the work of the clock and data recovery circuit, information must be encoded in such a way that a good transition density and a short run length are achieved. Also a dc-balanced serial data stream must be guaranteed by getting a good disparity.

Block line encoders group input bits into $m$-bit blocks and map these blocks into $n$-bit blocks, where $n$ is greater than $m$. Thanks to the redundancy introduced in data, block encoders achieve the attributes mentioned previously [17] (transition density, short run length, and dc-balance). In the case of 1000Base-X, 8b/10b encoding is used.

Normally, an additive stream cipher is implemented by carrying out the XOR operation between the plaintext and a keystream obtained from a secure pseudorandom generator. In case of encrypting physical layer when block-line encoding is used, it is necessary to preserve the mentioned encoder properties; therefore, the location of the encryptor in the datapath must be taken into account. In the case of a block line code such as 8b/10b, encryption should be carried out before the encoder and the XOR operation would not be suitable. An $m$-bit symbol should be encrypted giving rise to another $m$-bit symbol, within the alphabet of symbols supported by the encoding standard. If $R$ is the number of possible symbols, then the operation performed by the stream cipher should be a modulo-$R$ addition instead of an XOR. Moreover, $m$-bit symbols should be mapped in a value between 0 and $R-1$ before this addition, and the resulting value should be reverse-mapped to the corresponding new $m$-bit symbols that are finally encoded.

In Fig. 2, the generic mechanism is shown with such kind of encoding. The $m$-bit symbols are mapped to $b$-bit values that are encrypted, reverse-mapped, and, finally, encoded into $n$-bit words. In addition to these operations, the keystream generator must be able to produce a uniformly distributed keystream in the range of the available alphabet of symbols. To achieve this, in this paper, CTR mode is proposed to be used with FPE block ciphers. FPE modes are introduced in Section III, and security considerations of the overall structure are given in Section IV.

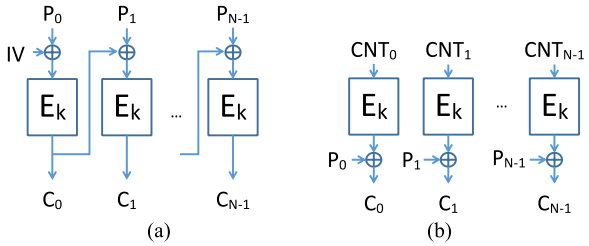



Fig. 3. Scheme of block cipher operation modes. (a) CBC. (b) CTR. Plaintext is split in $N-1$ blocks of $B$ bits $\{P_0, P_1, \ldots, P_{N-1}\}$ to get $N-1$ blocks for ciphertext $\{C_0, C_1, \ldots, C_{N-1}\}$. IV is the initial value for CBC and CNT the counter for CTR mode, both are $B$-bit wide. $E_K$ is the underlying block cipher and $B$ its blocksize in bits.

## III. FPE Cipher Modes

FPE encrypts plaintext in a ciphertext, preserving its original format and length. Some examples where this type of encryption is useful are the primary account numbers or social security numbers for which standard block ciphers or their operating modes would not preserve the format. Currently, two operation modes for FPE are recommended by NIST, FF1, and FF3 [14]. Both "ciphers" (actually, they are considered operation modes) have a scheme based on a nonbinary Feistel structure. In NIST recommendations, the underlying round function of the Feistel network consists of an advanced encryption standard (AES) block cipher, although in its original specifications this pseudorandom function (PRF) can be also implemented by a hash function based on hash-based message authentication code [18].

The main differences between both modes, FF1 and FF3, are the number of rounds in the Feistel structure (with a smaller number in FF3) and the way to achieve the arbitrary length message encryption. In the case of FF1, its specification allows message lengths up to $2^{32}$ words, whereas in FF3 message length is more limited and it is constrained by the input data radix. However, in the original proposal for FF3, also called Brier, Peyrin, and Stern (BPS), the basic block cipher component BPS-BC is proposed to be used in cipher block chaining (CBC) mode for encryption of arbitrary length messages [14]. In spite of this, CBC operation mode properties result in frequent misuse among classical confidentiality block cipher modes. Indeed, CTR can be considered the best and most modern way to achieve privacy-only encryption [19]. The basic operation for both modes is shown in Fig. 3.

As far as the authors know, there is no stream cipher proposal able to preserve the format of the plaintext symbols. The main stream cipher proposals are oriented to generate binary keystreams, as the finalists of eSTREAM project [20] or the well-known solution formed by secure block ciphers in CTR mode. Due to this absence, we consider that the use of the recent FPE modes approved by NIST in conjunction with CTR mode could be a good solution.

## IV. Security Considerations

### A. FPE With CTR Mode

Assuming that the plaintext is a stream of symbols defined with radix $R$, we can define the CTR mode for our purpose as described by NIST [21] but using a modulo-$R$ addition as encryption operation instead of an XOR. For a family of functions $F$ such that

$$F : K \times \{0, 1, \ldots, R-1\}^B \rightarrow \{0, 1, \ldots, R-1\}^B,$$

where $B$ is the block size and $K$ the keyspace, and given a plaintext $P = \{P_0, P_1, \ldots, P_{l-1}\}$ with $l > B$, $P$ is divided into $N$ blocks $P_i$: $N-1$ blocks with $B$ symbols plus one block with the rest. The $N$ blocks of the ciphertext $C_i$ are obtained applying Algorithm 1, where $N$ $K_i$ blocks are the keystream obtained from $N$ successively counter values $\mathrm{CNT}_i$. $\mathrm{MSB}(K_{N-1})$ are the most significant symbols of the last keystream block that operate with the remaining symbols in the last plaintext block $P_{N-1}$. The modulo-$R$ addition is performed symbol by symbol between each plaintext and keystream blocks. In Fig. 4, the proposed scheme is shown.

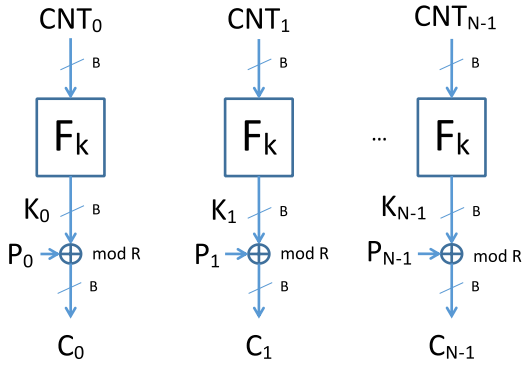



Fig. 4. Scheme of CTR mode using a FPE block cipher $F_K$. Plaintext is split in $N-1$ blocks of $B$ symbols with radix $R$ $\{P_0, P_1, \ldots, P_{N-1}\}$ to get $N-1$ blocks for ciphertext $\{C_0, C_1, \ldots, C_{N-1}\}$. Counter values $\{\mathrm{CNT}_0, \mathrm{CNT}_1, \ldots, \mathrm{CNT}_{N-1}\}$ and keystream values $\{K_0, K_1, \ldots, K_{N-1}\}$ are also blocks of $B$ symbols with radix $R$. Modulo-$R$ addition is done symbol by symbol inside each block between keystrem and plaintext.

**Algorithm 1:** CTR mode for FPE encryption.

$\mathrm{CNT}_0 = \mathrm{INIT\_CNT}$
$K_i = F_K[\mathrm{CNT}_i]$ for $i = 0, 1, \ldots, N-1$
$\mathrm{CNT}_{i+1} = \mathrm{CNT}_i + 1$ for $i = 0, 1, \ldots, N-1$
$C_i = (P_i + K_i) \bmod R$ for $i = 0, 1, \ldots, N-2$
$C_{N-1} = (P_{N-1} + \mathrm{MSB}(K_{N-1})) \bmod R$

The decryption algorithm will be as Algorithm 1, but replacing the plaintext $P$ with the ciphertext $C$ and the modulo-$R$ addition by a modulo-$R$ subtraction.

According to [22] and posterior studies, it can be shown that in the sense of IND-CPA (indistinguishability against chosen-plaintext attacks) security, given an adversary $X$ attacking a CTR scheme, it's possible to built an adversary $Y$ attacking the PRF security of the underlying function $F_K$ of that CTR scheme. The advantages, ADV, of such adversaries are related as

$$\mathrm{ADV}^{\mathrm{IND-CPA}}_{\mathrm{CTR}}(X) \leq 2 \cdot \mathrm{ADV}^{\mathrm{PRF}}_{F}(Y). \quad (1)$$

Also according to its proof, it can be concluded that it holds independently of the radix $R$ on which the CTR mode operates. On the other hand, according to the PRF–PRP switching lemma [22] and since the underlying function $F_K$ is a block cipher

that can be considered a pseudorandom permutation (PRP), it is well-known that the cited advantage is degraded by a term of $Q^2/2^{B+1}$, as

$$\mathrm{ADV}_{\mathrm{CTR}}^{\mathrm{IND-CPA}}(X) \leq 2 \cdot (\mathrm{ADV}_{F}^{\mathrm{PRP}}(Y) + Q^2/2^{B+1}) \quad (2)$$

where $Q$ is the number of queries made by the adversary $Y$ and $B$ is the blocksize of the blockcipher. In the case of an FPE block cipher with radix $R$, this factor will be $Q^2/2R^B$. Indeed, due to the birthday paradox, it could be considered insecure to encrypt more than $Q = \sqrt{R^B}$ blocks with the same key.

It is possible to conclude that, in the same way that the use of CTR mode with a standard block cipher is considered safe, we can safely use the CTR mode with an FPE block cipher. On the other hand, it will be important to take into account that the block size $B$ is a configurable parameter for FPE block ciphers, and it affects the overall security of the CTR scheme.

One requisite for $B$ could be to provide the same security limit as a recommended block cipher with a 128-bit blocksize. For AES working in CTR mode, $Q = 2^{64}$, then the data limit to be transmitted before key refreshing should be lower than $L = Q \cdot 128 = 2^{71}$ bits. As the proposed system in this paper should have at least the same bound, we have taken as design constraint the requisite

$$L = Q \cdot B \cdot b \geq 2^{71} \rightarrow \sqrt{R^B} \cdot B \geq \frac{2^{71}}{b} \quad (3)$$

where $B$ is the blocksize in symbols defined with radix $R$ and $b$ is the information bits represented by each symbol in the encoding standard. This constraint gives a lower bound for $B$ and it is taken into account in Section V, where the cipher parameters are explained.

### B. Keystream Analysis

It is well-known that one of the security criteria of a stream cipher is based on the fact that the generated keystream is indistinguishable from a truly random sequence. Therefore, it is useful to check the randomness of the obtained keystream sequence $\{K_0, K_1, \ldots, K_{N-1}\}$ in Fig. 4.

Among the most used tests for randomness, we can highlight some of them such as NIST [23], Diehard [24], or TestU01 [25]. All of them need as input a binary stream or a sequence of integers within a range that is a power of two. In our case, the keystream values are not included in a range like that due to the encoding used. The range of possible values for the keystream will be explained later in Sections V and VI. As far as the authors know, there is no standardized set of tests able to check the randomness of a sequence of integer numbers that are not included in a power of two ranges. However, the battery of tests proposed by Knuth [26] does include different tests that are suitable for our purpose, as they do not have the mentioned constraint.

In this paper, the keystream resulting from our FPE implementation was evaluated applying the following tests described in [26]: frequency test, serial test for pairs and triples of symbols, poker test, run and serial correlation test.

Regarding frequency and serial test, chi-square goodness-of-fit test was applied successfully with 5% and 1% of significance

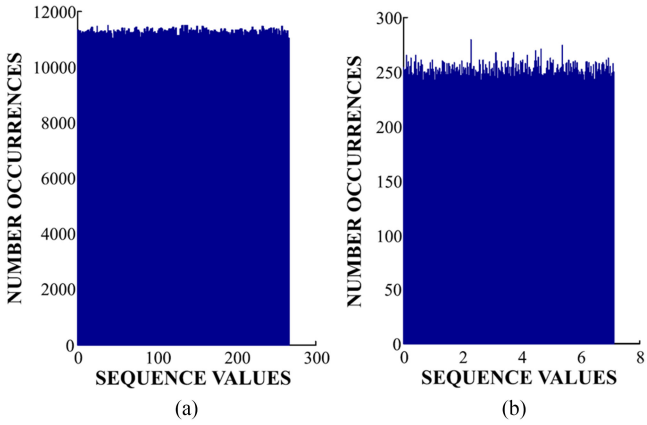



Fig. 5. Histograms obtained. (a) Frequency test. (b) Serial test. Sequence values units in (b) are given in thousands.

level. Histograms for the keystream sequence organized in tuples of one and two symbols are shown in Fig. 5(a) and (b) for sequences of 3 and 15 millions of tuples, respectively.

As for the poker test, chi-square goodness-of-fit test was also applied using the five categories described by Knuth [26].

With respect to the run test, it was applied to runs up and down successfully for sequences of 5000 values.

Correlation coefficient was calculated for a window of 25 000 samples and shifts up to 100 000, giving a result between the bounds recommended for this test.

Although the fulfillment of a statistical test does not guarantees that the sequence is truly random, we can conclude that the generated sequence is suitable as keystream for the proposed stream cipher.

## V. Application Case: Ethernet 1000Base-X

In this paper, we have focused on the case of high-speed communications, particularly in the 1000Base-X standard used in optical Gigabit Ethernet links. In this standard, the PCS level is responsible for generating and encoding control and data symbols that are transmitted to the optical line. As the block line coding is 8b/10b, the purpose of the encryption will be to cipher the complete 8b/10b symbol flow as shown in Fig. 2.

The main advantage of the proposed method is that there is no throughput loss because no extra data fields are added to the packets when they are encrypted, then no overhead is introduced. Moreover, it not only encrypts the contents of the packets, but also the activity or data traffic pattern. This is because by encrypting at 8b/10b symbol level, control symbols and ordered sets are also encrypted, such as packet start and end symbols or IDLE sets in the interframe gap (IFG). This last capacity could improve security as it prevents passive eavesdroppers from performing traffic analysis attacks. It would be useful in scenarios where traffic pattern analysis could reveal sensitive information about the behavior of a critical infrastructure or facility.

The dataset to be encrypted is a limited set, since the valid 8b/10b symbols are composed of 256 data symbols plus 12 control symbols, a total of 268 symbols that do not generate code errors. On the other hand, the special control symbol /K28.7/

used only in diagnostic functions is capable of generating an undesired Ethernet "comma" sequence if it coincides sequentially with some particular symbols [27]. A comma across any two adjacent code-groups may cause code-group realignment. Symbol /K28.7/ is not used for standard data communication, and in order to avoid the accidental generation of it in the encryption of any 8b/10b symbol it has been excluded from the encryption symbol mapping.

Therefore, the valid set of symbols consists of 267 possible values to be encrypted. As in Fig. 2, in order to carry out the encryption, the 267 possible symbols are mapped in the range of 0 to 266, giving a value of 267 for radix parameter $R$. After symbol mapping, modulo-$R$ addition is performed with the keystream. Once the cipher operation is done, the resulting values are reverse-mapped to the corresponding new 8b/10b symbol. The new symbols will generate neither code error nor realignment as they exist in the set of 267 possible symbols and /K28.7/ has been excluded from it.

To generate the keystream, an FPE block cipher in CTR mode has been built. Among the two NIST recommendations, FF1 and FF3, we have selected the latter. FF1 mode is not as limited in block length as FF3, since its specification says that it can reach up to $2^{32}$ symbols; therefore, a greater security could be achieved. However, since for FF3 mode fewer rounds are required in the Feistel network, the cost in hardware resources is lower. In this mode, the block size is limited according to the radix used, mathematically expressed as follows:

$$\begin{aligned} R &\in \left[2 \ldots 2^{16}\right] \\ R^{\text{minlen}} &\geq 100 \\ 2 \leq \text{minlen} &\leq \text{maxlen} \leq 2\left\lfloor \log_R\left(2^{96}\right)\right\rfloor \end{aligned} \tag{4}$$

where $R$ is the radix, and minlen and maxlen are the bounds for the block size $B$. Since radix is 267, according to (4), block size $B$ is between 2 and 22.

On the other hand, according to (1), for $R = 267$ and $b = 8$ information bits/symbol, $B$ should be greater than or equal to 16; therefore, it is possible to achieve this value using FF3 mode, as 22 is the maximum allowable value for $B$.

In this paper, the selected value for block size is the maximum, it means 22. The maximum block size means a greater number of cycles available per stage for a possible pipelined architecture, which allows for a better reuse of the resources consumed by the hardware.

Given these parameters, the structure of the full streaming encryption system is shown in Fig. 6. It is similar to Fig. 2, but the keystream generator module has been replaced with the final CTR structure based on FPE block cipher.

In Fig. 6, TX_PCS_CTRL module represents the PCS controller that generates the 8b/10b symbol flow. Each symbol is formed by its 8-bit value $D\_\text{in}$ and the control flag $K\_\text{in}$. This control flag signals if the 8b/10b symbol is a control or data one, depending if its value is "1" or "0", respectively. Both signals $D\_\text{in}$ and $K\_\text{in}$ are the input to the cipher. The encrypted output is a new 8b/10b symbol formed by $D\_\text{out}$ and $K\_\text{out}$, and it is the input of the encoder.

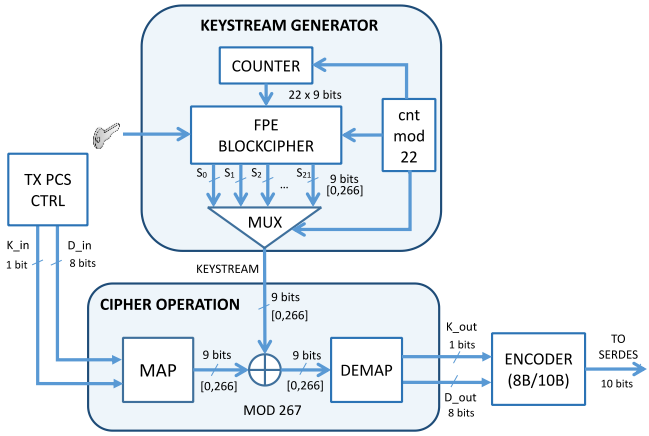



Fig. 6. Overall structure for the streaming encryption system in a physical layer with 8b/10b encoding. Decryption will be as encryption but using a modulo-267 subtraction instead of an addition.

In this structure, for each clock cycle, a modulo-22 counter (called "cnt_mod_22" in Fig. 6) selects one of the 22 symbols from the output block of the FPE block cipher. In Fig. 6, these symbols are represented by the set $\{S_0, S_1, \ldots, S_{21}\}$. The selected output symbol will be added to the incoming plaintext, thanks to the modulo-267 addition. The FPE output block is refreshed every 22 cycles. The same happens with its input block that comes from CTR counter (called COUNTER in Fig. 6). This COUNTER will also increment its value every 22 cycles.

## VI. System Implementation

### A. System Description

FF3 encryption algorithm is described in [14], and it implements a nonbinary Feistel network. In this paper, the cipher tweak value has been set to zero, and only the key is configurable. By taking into account the parameters that we need for our system (radix $R = 267$ and block size $B = 22$), the hardware structure for the FPE block cipher can be particularized and represented as shown in Fig. 7, where the main functions are implemented in modules REV, NUM, STR, AES, and mod $267^{11}$, all of them are described in [14]. In this implementation, AES module has a key size of 128 bits.

The width of the data bus in each branch of the Feistel network is 11 symbols, i.e., half of the block size.

This design has a pipelined architecture to achieve the required throughput; as blocksize is 22, for every 22 cycles 22 symbols are generated in the block cipher output and each stage in the pipeline can last at most 22 cycles. Each round can be implemented in several stages and its number depends on the functions that it carries out. Particularly, in this implementation, function REV takes zero stages, NUM and AES take one stage, STR takes 10 stages, and mod $267^{11}$ takes two stages.

### B. Implementation Results

The system described in Fig. 7 has been implemented in a Xilinx Virtex 7 field programmable gate array (FPGA). Regarding the resources used for the FPE block cipher, these are shown

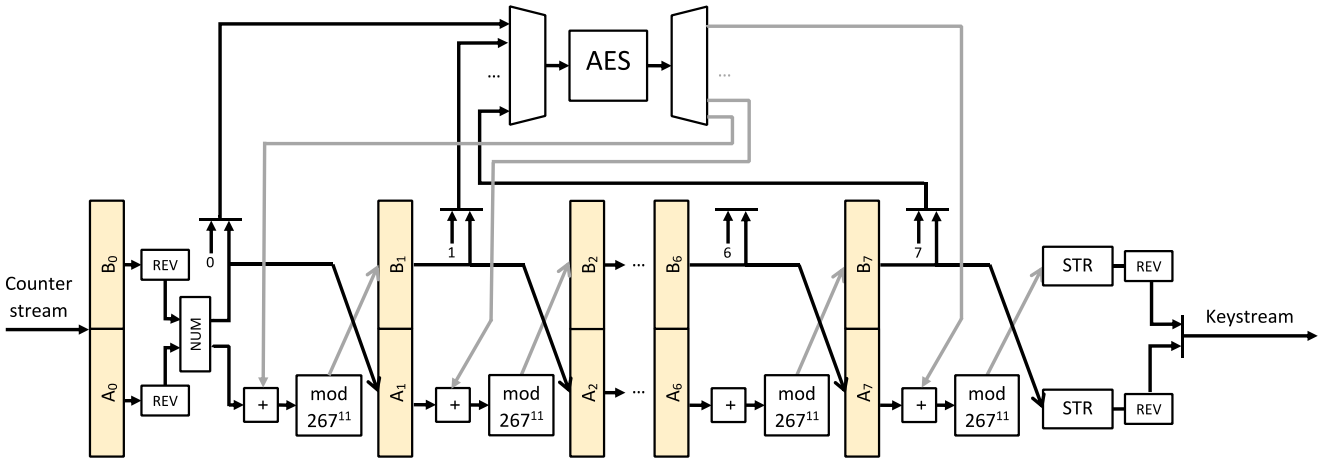



Fig. 7. Hardware structure of the implemented FPE block cipher of Fig. 6. Its input is the stream of counter values and its output the keystream.

TABLE I
COMPARISON WITH OTHER SOLUTIONS

| | FF1 [16] | FF3 [16] | This Work |
|---|---|---|---|
| Slice Registers | 11285 | 5592 | 11127 |
| Slice LUTs | 7426 | 3587 | 16978 |
| 18K Block RAMs[1] | 343 | 170 | 77 |
| Slices[2] | 3268 | 1596 | 5636 |
| Operation. Freq. (MHz) | 279.6 | 283.5 | 125 |
| Cycles/Encryption | 707 | 269 | 1 |
| Bytes/Encryption | 13 | 13 | 1 |
| Encryption Rate (Mbps) | 41.1 | 109.6 | 1000 |
| Encryption Rate/Slice (Kbps/Slice) | 12.57 | 68.7 | 177.4 |

[1]77 Block RAMs are used in this work is due to the AES core.[2]Slices are estimated from the number of register and LUTs, assuming they are not packed together.

in Table I in terms of look-up tables (LUTs), registers, and block RAMs. In this design, no digital signal processing cell has been used.

Table I also shows a resource comparison with other implementations, particularly with [16], where FF1 and FF3 modes are implemented. Such implementations use an iterative looping architecture that saves resources but increments the number of cycles to carry out one block encryption. Although it is difficult to establish a comparison, it has been made in terms of throughput/resources. Among the three modes studied in [16], only FF1 and FF3 modes have been selected for the comparison, as FF2 was removed from NIST recommendation.

As shown in Table I, the proposed system in this paper entails a throughput/resources ratio better than that of the existing FPE implementations. This comparison is only orientative as FPGA devices used in both implementations are different. While in [16], implementation is made over a Virtex-6 device, in this paper, Virtex-7 is used. Anyways, in both devices, the CLB structure is similar in terms of LUTs and registers, with four six-input LUTs and eight registers per slice.

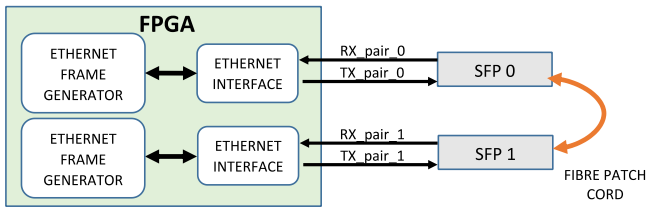



Fig. 8. Test setup scheme.

Also, it is important to remark that in this paper the necessary radix is 267. It implies that mod $267^{11}$ and STR modules are more complex to implement than the cases where radix is a power of two. In the latter division and modulo, operations could simply be reduced to shifting and slicing operations over bit vectors.

### C. Encryption Setup

To test the proposed encryption system, the overall system described in Fig. 6 has been integrated in a 1000Base-X Ethernet interface linked to an Ethernet frame generator. Two chains, each consisting of an Ethernet interface and Ethernet frame generator, have been implemented over a Xilinx Virtex 7 FPGA. In the setup for test, the Ethernet interfaces have been connected to two small form-factor pluggable (SFP) modules suitable for Gigabit Ethernet standard over multimode fiber. The overall setup scheme and a photo of the hardware system are shown in Figs. 8 and 9, respectively. In Fig. 10, the structure of the Ethernet interface is shown. Ethernet frame generators have been used to check the encrypted link with real traffic and verify that no frames are lost and no cyclic redundancy check (CRC) errors are produced during encryption.

The final PCS structure, shown in Fig. 10, contains the encryption and decryption modules. In the 8b/10b TX datapath, the latency introduced by the encryption module TX_ENCRYPT is 192 ns, and approximately the same for the RX datapath. In addition, no overhead is introduced in the encryption process; thus, a 100% throughput is achieved.

These values of latency and throughput are a doubtless improvement with respect to other encryption mechanisms, such

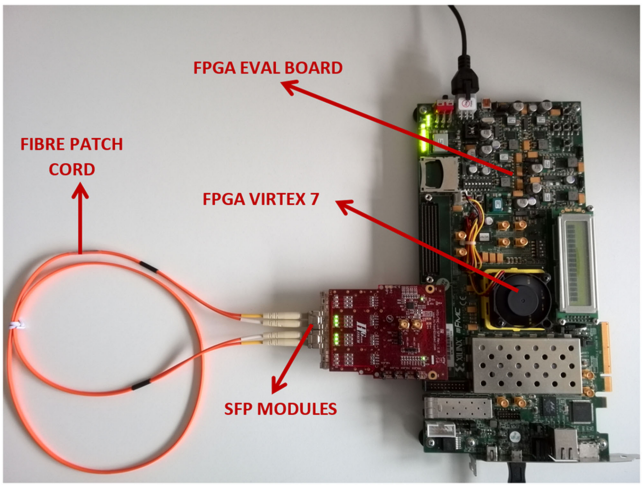



Fig. 9. Test setup photo with SFP modules working at 1 Gbps.

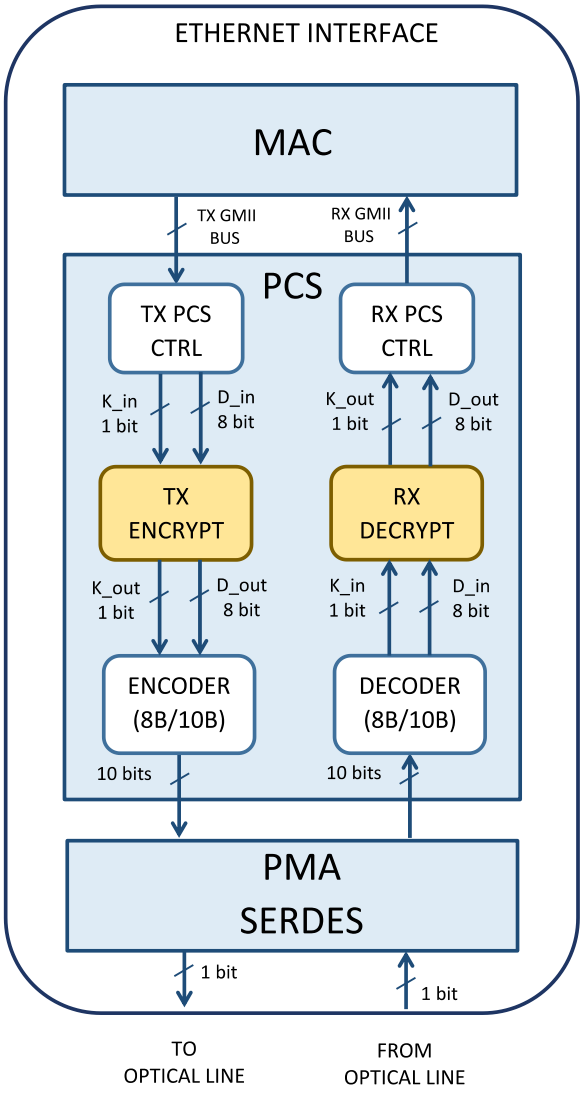



Fig. 10. Structure of the Ethernet Interface. It is composed by the MAC module, the PCS and the SERDES. In the PCS layer, TX_ENCRYPT and RX_DECRYPT are the encryption/decryption modules. Both include the CIPHER_OPERATION and KEYSTREAM_GENERATOR modules shown in Fig. 6. TX_PCS_CTRL and RX_PCS_CTRL are the PCS controllers in charge of generating and receiving the 8b/10b symbols.

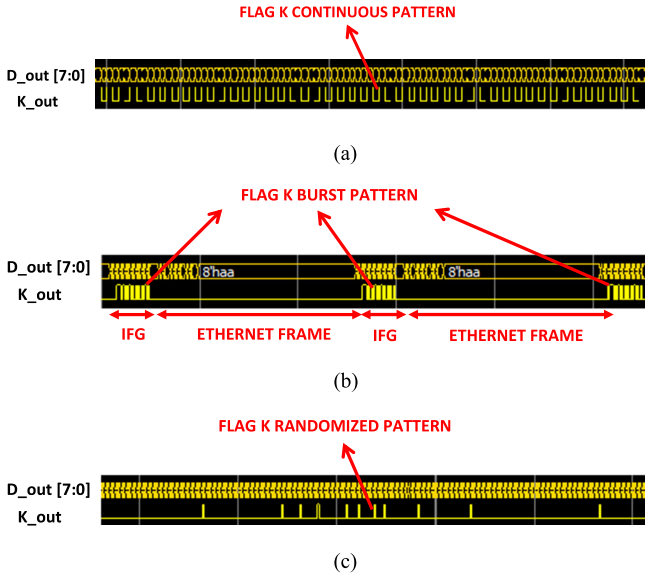



Fig. 11. (a) K flag pattern without encryption when no Ethernet frame is transmitted. (b) K flag pattern without encryption when transmitting an Ethernet frame burst. (c) K flag pattern after encryption regardless of the transmission or nontransmission of Ethernet frames.

as IPsec. As an example, in [28] latency measurements made over commercial equipment that implements IPsec give values between 50 and 350 ms. Moreover, the inherent overhead introduced by IPsec during encryption limits the throughput of these equipment to values between 20% and 90% of the maximum achievable.

### D. Encryption Results

Different Ethernet traffic patterns have been tested. Particularly, no frame transmission and transmission of frames with randomized payload at different rates have been encrypted. Patterns have been named A, B, C, and D. A corresponds with the case of no frame transmission, where only IDLES are transmitted over the link. B, C, and D correspond to continuous frame transmission of 1024-bytes length at rates of 10.2%, 50%, and 91% of the maximum Gigabit line rate, respectively.

As mentioned previously, the proposed encryption system is able to make indistinguishable data traffic patterns. Regarding to this capability, it is interesting to monitor the signal waveform at the input of the 8b/10b encoder. As shown in Figs. 6 and 10, the output of the encryption is the input of the encoder, and it is formed by the 8b/10b symbol value $D\_\mathrm{out}$ and its K control flag (named $K\_\mathrm{out}$). Each 8b/10b symbol is a control or data one depending on whether its K flag is "1" or "0," respectively. $D\_\mathrm{out}$ and $K\_\mathrm{out}$ are used by the 8b/10b encoder to encode an 8-bit symbol into a 10-bit one.

K control flag pattern has been monitored for each of the mentioned traffic patterns (from A to D) with the encryption activated and deactivated.

When the encryption is not enabled in the case of A pattern, where no frames are transmitted, K control flag waveform is a signal that switches continuously between "0" and "1," as shown in Fig. 11(a). It is because when no frame is

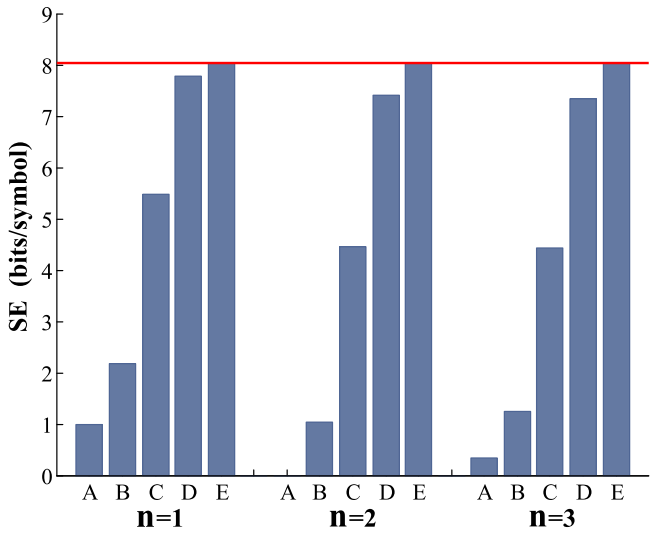



Fig. 12. SE measured with *n* from 1 to 3, and for all mentioned Ethernet traffic patterns, A, B, C, D, and E.

transmitted, the physical layer controller (TX_PCS_CTRL module in Fig. 10) transmits continuously IDLE sets to the 8b/10b encoder. These IDLE sets are composed by two consecutive symbols, the control symbol /K28.5/ (with K flag equals to "1") plus a data symbol (with K flag equals to "0").

However, with encryption disabled for patterns B, C, and D, the K flag waveform seems a blast signal, as IDLE transmission only occurs in the IFG between frames, and during frame transmission only 8b/10b data symbols are transmitted (setting K flag continuously to "0"). It is shown in Fig. 11(b).

On enabling encryption, the K flag waveform is completely randomized in all cases, A, B, C, and D, making indistinguishable which pattern is being transmitted, as shown in Fig. 11(c).

As an overall result of encryption, Shannon entropy (SE) has been measured for each of the mentioned patterns before and after encryption as

$$\mathrm{SE} = -\frac{1}{n} \cdot \sum_{\beta_n \epsilon R^n} P\left(\beta_n\right) \cdot \log_2 P\left(\beta_n\right). \tag{5}$$

The 8b/10b symbol stream for each traffic pattern, mapped between 0 and $R-1$, has been grouped in tuples of $n$ symbols, called $\beta_n$, and the probability for each tuple $P(\beta_n)$ has been calculated. Particularly, SE has been measured for values of $n$ from 1 to 3 in each of the mentioned patterns A, B, C, and D. For each $n$, 1, 2, and 3, the length of the used sequences has been 2.67, 14.26, and 571 M samples, respectively.

Because in (5) logarithms with base 2 are used, the SE result is given in bits/symbol. Ideally, if every $n$-tuple $(\beta_n)$ is equally likely with probability $P(\beta_n) = p = R^{-\mathrm{n}}$, the value of SE for every $n$ is given as

$$-\frac{1}{n} \cdot R^n \cdot p \cdot \log_2 p = \log_2 R = \log_2 267 \cong 8.0606. \tag{6}$$

Owing to the limited memory in FPGA hardware resources, measurements for each pattern have been calculated at simulation stage, but this fact does not invalidate experimental results.

In Fig. 12, values of SE with $n$ from 1 to 3 and for patterns A, B, C, and D are shown. These values are compared with a fifth case named E, which corresponds to the randomized signal after encryption of pattern A, which can be considered the worst case in terms of entropy. In this last pattern E, it is possible to notice that SE is, as expected, almost the ideal value, i.e., 8.0606. However, as the transmission rate decreases from D to A, the SE value also decreases. It is a logical result, as when a lower bandwidth transmission is used, IFGs full of IDLES take more bandwidth percentage versus the random payloads of the transmitted frames.

Thank to this result, it is possible to conclude that encryption works and makes indistinguishable data traffic patterns, which permits hiding of the pattern and contents of Ethernet traffic from any passive eavesdropper.

## VII. Conclusion

As far as the authors are aware, this paper was the first time that an encryption mechanism was proposed for ciphering physical layer communications based on 8b/10b coding. This new encryption system consists of symmetric ciphering of the complete 8b/10b symbol stream. Encryption based on an FPE block cipher working in CTR mode was simulated and implemented. Also, security considerations were taken into account. The proposed system was able to obfuscate the data traffic pattern, which could improve the overall security, with no throughput losses, null overhead, and low latency.

These properties make this kind of "in-flight" encryption suitable for protocols where delay and jitter are critical, such as the real-time Ethernet in industrial environments, which would improve security in modern optical SCADA networks.

In addition to this, by preserving coding properties such as dc-balance, short run length, and transition density, physical layer encryption was easily achieved without making changes in the subsequent hardware elements or medium-dependent circuitry. As an example, in Ethernet 1000Base-X standard, SERDES (Serializer/Deserializer), and commercial SFP, optical modules could be compatible with the proposed encryption system.

Regarding its implementation, although in the proposed system there was an increment in the hardware resources with respect a standard AES block cipher, this system entails a throughput/resources ratio better than that of the existing FPE implementations.

Finally, as future work, authors consider studying the implementation of physical layer encryption for other Ethernet standards with higher transmission rates such as 10G-BaseR.

## References

[1] T. Sauter, "The three generations of field-level networks-evolution and compatibility issues," *IEEE Trans. Ind. Electron.*, vol. 57, no. 11, pp. 3585–3595, Nov. 2010.

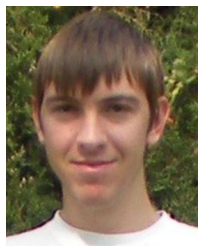
[2] K. C. Lee, S. Lee, and M. Lee, "Worst case communication delay of real-time industrial switched Ethernet with multiple levels," *IEEE Trans. Ind. Electron.*, vol. 53, no. 5, pp. 1669–1676, Oct. 2006.

[3] T. Sauter, S. Soucek, W. Kastner, and d. Dietrich, "The evolution of factory and building automation," *IEEE Ind. Electron. Mag.*, vol. 5, no. 3, pp. 35–48, Sep. 2011.

[4] J. Aguirre, D. Bol, D. Flandre, C. Sánchez-Azqueta, and S. Celma, "A robust 10-Gb/s duobinary transceiver in 0.13-$\mu$m SOI CMOS for short-haul optical networks," *IEEE Trans. Ind. Electron.*, vol. 65, no. 2, pp. 1518–1525, Feb. 2018.
[5] C. Gimeno, E. Guerrero, C. Sánchez-Azqueta, G. Royo, C. Aldea, and S. Celma, "Continuous-time linear equalizer for multigigabit transmission through SI-POF in factory area networks," *IEEE Trans. Ind. Electron.*, vol. 62, no. 10, pp. 6530–6532, Oct. 2015.
[6] N. Skorin-Kapov, M. Furdek, S. Zsigmond, and L. Wosinska, "Physical-layer security in evolving optical networks," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 110–117, Aug. 2016.
[7] T. Sauter and M. Lobashov, "How to access factory floor information using internet technologies and gateways," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 699–712, Nov. 2011.
[8] J. Lázaro, A. Astarloa, J. Araujo, N. Moreira, and U. Bidarte, "MACsec layer 2 security in HSR rings in substation automation systems," *Energies*, vol. 10, no. 2, pp. 1–15, Jan. 2017.
[9] K. Guan, J. Kakande, and J. Cho, "On deploying encryption solutions to provide secure transport-as-a-service (TaaS) in core and metro networks," in *Proc. Eur. Conf. Exhib. Opt. Commun.*, Düsseldorf, Sep. 18–22, 2016, pp. 1–3.
[10] C. Xenakis, N. Laoutaris, L. Merakos, and I. Stavrakakis, "A generic characterization of the overheads imposed by IPsec and associated cryptographic algorithms," *Comput. Netw.*, vol. 50, no. 17, pp. 3225–3241, 2006.
[11] MicroSemi, "In-flight Encryption in Service Provider Networks, No: PMC-2150716, Issue 2," 2016. [Online]. Available: https://pmcs.com/cgi-bin/document.pl?docnum= 2150716. Accessed on: Aug. 17, 2017.
[12] X. Wu, L. Xie, and F. Lim, "Network delay analysis of EtherCAT and PROFINET IRT Protocols," in *Proc. Ind. Electron. Soc., IECON*, Dallas, TX, USA, 2014 , pp. 2597–2603.
[13] J.-D. Decotignie, "Ethernet-based real-time and industrial communications," *Proc. IEEE*, vol. 93, no. 6, pp. 1102–1117, Jun. 2005.
[14] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption*. Gaithersburg, MD, USA: NIST Special Publication 800-38G, 2016.
[15] P. Rogaway, "A synopsis of format-preserving encryption," *University of California*, Davis, CA, USA, Mar. 2010. Online. Available: http://web.cs.ucdavis.edu∼rogaway/papers/synopsis.pdf
[16] R. Agbeyibor, J. Butts, M. Grimaila, and R. Mills, "Evaluation of format preserving encryption algorithms for critical infrastructure protection," in *Critical Infrastructure Protection VIII*. New York, NY, USA: Springer-Verlag, 2014, pp. 245–261.
[17] D. R. Smith, *Digital Transmission Systems*, New York, NY, USA: Springer-Verlag, 2004.
[18] "FIPS 198-1. The keyed-hash message authentication code (HMAC)," National Institute of Standards and Technology, Gaithersburg, MD, USA, Jul. 2008.
[19] P. Rogaway, "Evaluation of some blockcipher modes of operation," in *Proc. Tech. Rep., Cryptography Res. Eval. Committees Gov. Jpn.*, Feb. 2011, pp. 1–159.
[20] M. Robshaw and O. Billet, *New Stream Cipher Designs: The eSTREAM Finalists*. New York, NY, USA: Springer-Verlag, 2008.
[21] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Methods and Techniques*. Gaithersburg, MD, USA: NIST Special Publication 800-38G, 2001.
[22] M. Bellware, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption: Analysis of the DES modes of operation," in *Proc. Symp. Found. Comput. Sci.*, 1997, pp. 349–403.
[23] A. Rukhin *et al.*, *A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications.* Gaithersburg, MD, USA: NIST Special Publication 800-22 Rev.1a, 2010.
[24] G. Marsaglia, *Diehard: A Battery of Tests of Randomness*, 1997. Online. Available: http://webhome.phy.duke.edu˜rgb/General/dieharder.php
[25] P. L'Ecuyer and R. Simard, "TestU01: A C library for empirical testing of random number generators," *ACM Trans. Math. Softw.*, vol. 33, no. 4, Aug. 2007.
[26] D. Knuth, *The Art of Computer Programming, vol. 2, Seminumerical algorithms*. Reading, MA, USA: Addison-Wesley, 1981.
[27] "IEEE Standard for Ethernet, IEEE Std 802.3-2015, Section 3, 36.2.4.9," Sep. 2015.
[28] L. Troell, J. Burns, K. Chapman, D. Goddard, M. Soderlund, and C. Ward, "Converged vs. dedicated IPSEC encryption testing in gigabit ethernet networks, technical report," 2005. [Online]. Available: http://scholarworks.rit.edu/article/1743

**Adrián Pérez-Resa** was born in San Sebastián, Spain. He received the M.Sc degree in telecommunications engineering from the University of Zaragoza, Zaragoza, Spain, in 2005. Currently, he is working toward the Ph.D degree in telecommunications engineering from the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza.

He was an R&D Engineer with the Telecommunications industry for more than ten years. He is a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include high speed communications and cryptography applications.



**Miguel Garcia-Bosque** was born in Zaragoza, Spain. He received the B.Sc. and M.Sc. degrees in physics from the University of Zaragoza, Zaragoza, Spain, in 2014 and 2015, respectively.

He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include chaos theory and cryptography algorithms.

**Carlos Sánchez-Azqueta** was born in Zaragoza, Spain. He received the B.Sc., M.Sc., and Ph.D. degrees in physics from the University of Zaragoza, Zaragoza, Spain, in 2006, 2010, and 2012, respectively, and the Dipl.-Ing. degree in electronic engineering from the Complutense University of Madrid, Madrid, Spain, in 2009.

He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include mixed-signal integrated circuits, high-frequency analog communications, and cryptography applications.

**Santiago Celma** was born in Zaragoza, Spain. He received the B.Sc., M.Sc., and Ph.D. degrees in physics from the University of Zaragoza, Zaragoza, Spain, in 1987, 1989, and 1993, respectively.

He is currently a Full Professor in the Group of Electronic Design, Aragon Institute of Engineering Research, University of Zaragoza. He has coauthored more than 100 technical papers and 300 international conference contributions. He is coauthor of four technical books and the holder of four patents. He is a Principal Investigator in more than 30 national and international research projects. His research interests include circuit theory, mixed-signal integrated circuits, high-frequency communication circuits, wireless sensor networks, and cryptography for secure communications.

# Self-Synchronized Encryption for Physical Layer in 10Gbps Optical Links

Adrián Pérez-Resa, Miguel Garcia-Bosque, Carlos Sánchez-Azqueta, and Santiago Celma

**Abstract**—In this work a new self-synchronized encryption method for 10 Gigabit optical links is proposed and developed. Necessary modifications to introduce this kind of encryption in physical layers based on 64b/66b encoding, such as 10 GBase-R, have been considered. The proposed scheme encrypts directly the 64b/66b blocks by using a symmetric stream cipher based on an FPE (Format Preserving Encryption) block cipher operating in PSCFB (Pipelined Statistical Cipher Feedback) mode. One of the main novelties in this paper is the security analysis done for this mode. For the first time, an expression for the IND-CPA (Indistinguishability under Chosen-Plaintext Attack) advantage of any adversary over this scheme has been derived. Moreover, it has been concluded that this mode can be considered secure in the same way of traditional modes are. In addition, the overall system has been simulated and implemented in an FPGA (Field Programmable Gate Array). An encrypted optical link has been tested with Ethernet data frames, concluding that it is possible to cipher traffic at this level, getting maximum throughput and hiding traffic pattern from passive eavesdroppers.

**Index Terms**—Optical communications, Ethernet, self-synchronous encryption, pipeline statistical cipher feedback

## 1 Introduction

Today, high speed optical networks are a reality. Thanks to the advances in these technologies it is possible to afford the bandwidth growth that nowadays modern applications demand [1], such as cloud computing and big data. In addition, information security has become an important issue as the volume of threat events has increased over the last years [2]. Failures in security can lead to the malfunction of a service or the confidentiality loss in customer critical information.

In a layered communication system, such as OSI (Open System Interconnection) or TCP/IP (Transmission Control Protocol/Internet Protocol), passive or active attacks can be carried out at different communication levels. Depending on the communication layer, different approaches are used for getting information confidentiality. For example, standardized protocols such as MACsec [3] or IPsec [4] are usually used at layer 2 (Data link layer) and layer 3 (Network layer), respectively. In these cases, encryption is carried out in each frame individually.

For the particular case of optical networks, the threat analysis in its physical layer is also considered critical to guarantee secure communications. [5], [6]. Among the most important attacks at this level, signal splitting attacks must be taken into consideration. Nowadays thanks to low-cost tapping techniques it is possible to intercept the optical signal without the need to perceptibly interfere in communications or create visible side-effects [7].

To deal with these threats and protect data confidentiality, several physical layer mechanisms related with photonic technologies have been proposed [8], for example OCDM (Optical Code Division Multiplexing) [9], SCOC (Secure Communications using Optical Chaos) [10] or QKD (Quantum Key Distribution) [11]. Other techniques, related with physical layer protocols, cipher the information at bit level, for example the encryption of OTN (Optical Transport Network) frame payloads [12]. Some of the advantages claimed by these techniques are that they achieve in-flight encryption introducing null overhead and a very low latency (in the range of nanoseconds) in data packets [12]. In fact, OTN communication equipment performing encryption at line rate and getting a 100 percent throughput are already available on the market [13]. This contrasts with what protocols at other layers do [14], [15]. For example IPsec usually introduces latencies in the range of milliseconds. Moreover, the overhead introduced by IPsec during encryption limits the total throughput to values between 20 and 90 percent of the maximum achievable [16], [17].

Some applications for 10 Gigabit Ethernet standards are shown in Fig. 1. Nowadays one of the most used technologies for the access to optical transport networks is Ethernet, and for high data rates 10 Gigabit Ethernet is widely deployed in MAN (Metro Area Networks) and WAN (Wide Area Networks) environments. Optical 10 Gigabit Ethernet standards are also available for EFM (Ethernet in the First Mile) applications, allowing customers the access to the provider network through a PON (Passive Optical Network) infrastructure.

Regarding the physical layer security in these standards, a new mechanism was proposed in [18]. The encryption solution consisted of a symmetric chaotic stream cipher, suitable

• *The authors are with the Group of Electronic Design, (GDE), Aragón Institute of Engineering Research (I3A), Zaragoza University, CP 50009, Spain. E-mail: {aprz, mgbosque, csanaz, scelma}@unizar.es.*

*Manuscript received 18 July 2018; revised 21 Dec. 2018; accepted 22 Dec. 2018. Date of publication 2 Jan. 2019; date of current version 16 May 2019. (Corresponding author: Adrián Pérez-Resa.)*
*Recommended for acceptance by W. Liu.*
*For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.*
*Digital Object Identifier no. 10.1109/TC.2018.2890259*

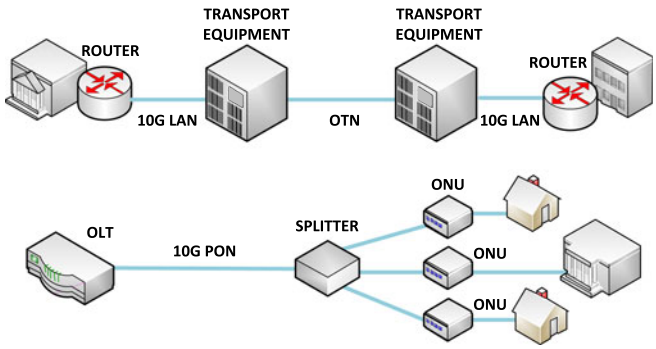



Fig. 1. Example of two different 10 Gigabit Ethernet standards in optical networks for WAN and EFM applications: 10GBase-R (upper) and 10GBase-PR (lower).

for the encryption of 64b/66b blocks. This kind of encryption could provide the mentioned advantages such as maximum throughput and low latency. However, although stream ciphers are suitable for high speed applications, their cryptanalysis and design criteria are less understood than block ciphers [19]. Indeed a stream cipher application can be implemented easily thanks to a secure block cipher such as AES working in CTR mode, considered secure thanks to its formal security proof [20]. Furthermore, in [18] the implemented stream cipher is based on a chaotic structure whose cryptanalysis could be not clear enough and it is mainly based on its randomness analysis.

On the other hand, the encryption system in [18] is not self-synchronized. To get synchronization, a mechanism was implemented based on the usage of new ordered sets in a specific 64b/66b block type, which increments the complexity of the overall system. In addition, in case of missing the synchronization in the middle of an encryption session there is a lack of a protocol to recover it.

In this work, a complete solution to overcome the mentioned disadvantages is proposed.

Regarding self-synchronization, in Ethernet data stream at PCS level there is no possibility to synchronize TX and RX stream ciphers thanks to standardized data fields or structures, as in OTN, where data stream is composed of continuous and periodic data containers.

To get synchronization in the symmetric encryption scheme, a self-synchronized operating mode called PSCFB has been analyzed and implemented. Moreover, a formal security expression for this operation mode has been deduced, concluding that it can be considered secure in the same way as other traditional modes.

The paper is divided into the following sections. Section 2 explains PCS layer encryption necessities when using 64b/66b encoding, in Section 3 an introduction to the PSCFB (Pipeline Statistical Cipher Feedback) mode of operation is made. In Section 4 IND-CPA (Indistinguishability under Chosen-Plaintext Attack) advantage expression for this operation mode is proposed. Subsequently, Section 5 deals with the practical case of 10 Gigabit Ethernet, particularly with the standard 10 GBase-R, and the overall scheme of the proposed encryption system. In Section 6, the hardware implementation of the cipher is described while in Section 7 results of the encryption are explained. Other security considerations as key distribution are taken into account in Section 8. Finally, in Section 9 conclusions are given.

## 2 Coding Preserving Encryption

Typically, in Ethernet standards, the physical layer is divided into three sublayers, PCS (Physical Coding Sublayer), PMD (Physical Medium Dependent) and PMA (Physical Medium Attachment). The Physical Coding Sublayer carries out functions such as link establishment, clock rate adaptation, data encoding and scrambling.

Optical Ethernet standards are high-speed communication systems where a baseband serial data transmission is carried out while clock frequency information is embedded in the serial bitstream itself. At the receiver, the clock recovery circuits must be able to extract the frequency information thanks to the bit transitions in the data stream. After that, serial data sampling can be made at the appropriate time. In order to facilitate the work of the CDR (Clock and Data Recovery) circuit, information must be encoded in such a way that a good transition density and a short run length are achieved. Also a DC-balanced serial data stream must be guaranteed, which is important for some transmission media, such as optical links.

In the case of PCS sublayers using a dense coding such as 64b/66b the mentioned properties, DC-balance and transition density, are achieved in a statistical way thanks to the scrambling of the bitstream. On the other hand, the short run length is guaranteed thanks to a synchronization header at the beginning of each 66-bit block, whose only two possible values are ‘10’ or ‘01’.

Usually stream ciphers are implemented by carrying out the XOR operation between the plaintext and a keystream obtained from a secure pseudorandom generator. In case of using physical layer encryption in the PCS sublayer, it is necessary to preserve the properties of the block line encoder, therefore the location of the XOR operation in the datapath must be taken into account. For 64b/66b encoding where first the blocks of bits are formatted and finally processed by a scrambler, the stream cipher should implement the XOR operation before the scrambler to guarantee that the scrambler transfers its statistical properties to the resulting bitstream before being transmitted.

As mentioned before, the synchronization header of each 66-bit block is composed of a pair of bits with two possible values ‘01’ or ‘10’. On the one hand, it is necessary for avoiding long runs of zeros or ones, which limits the run length to a maximum of 66 bits. On the other hand, it is used to detect the 66-bit block boundary in the receiver and perform the decoding process properly. For this reason, this 2-bit header is kept untouched and is not scrambled as the rest of the 66 bit block.

To preserve the two-bit transition of the sync header, the XOR operation must be carried out as shown in Fig. 2. The 2-bit header must be mapped to values ‘0’ or ‘1’ depending on whether it is equal to ‘01’ or ‘10’, respectively. Then the mapped value is concatenated to the 64-bit block payload. These two operations are performed in the HEADER_MAP block. The resulting output is a 65-bit word that is XORed with the keystream, giving a new 65-bit word. The first bit of this word will be reverse mapped giving the new 2-bit header while the subsequent 64 bits will be directly the new payload. The concatenation of both data fields will result on the new ciphered 66-bit block. These last operations are performed in the HEADER_REVERSE_MAP module.

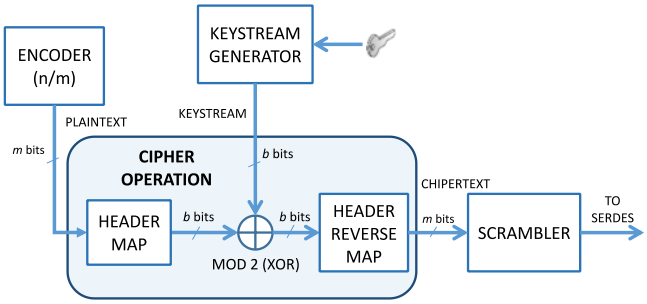



Fig. 2. Location and generic structure of a stream cipher in a physical layer with a dense block line encoding. In the case of 64b/66b encoding, parameters $n$, $m$, and $b$ are 64, 66 and 65 bits, respectively. The output of the encoder is a 66-bit block whose 2-bit header is mapped to a value '0' or '1' and concatenated with the rest of the block in the HEADER MAP module. The result is a 65-bit block that is encrypted in a one-time pad fashion thanks to the XOR operation.

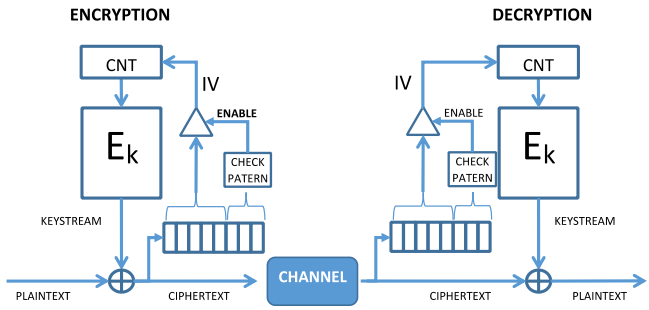



Fig. 3. Structure of PSCFB mode for encryption and decryption. The block cipher $E_K(\cdot)$ has a block size of $L$ bits, and is implemented using $P$ internal stages. The difference between the encryption and decryption is that the sync pattern scan is performed after the XOR operation in the transmitter and before the XOR in the receiver.

## 3 Self-Synchronized Encryption

### 3.1 Self-Synchronized Stream Ciphers

Self-synchronized stream ciphers usually generate the keystream as a function of the key and the preceding ciphered bitstream. In spite of its self-synchronizing properties, this kind of ciphers are less understood and their security analysis is more difficult than typical synchronized stream ciphers. Indeed, there are few proposals of these algorithms. For example, only two of the proposed stream ciphers in eSTREAM project, SSS and Mosquito were self-synchronized [21]. However, they were dismissed owing to their vulnerabilities [22], [23].

On the other hand, stream ciphers can also be based on different operating modes of block ciphers, such as OFB (Output Feedback), CFB (Cipher Feedback) or CTR (Counter) modes [24]. For self-synchronized purposes CFB is the only one recommended by the NIST (National Institute of Standards and Technology). However, to achieve synchronization with a loss of an arbitrary number of bits, CFB mode must only feedback one ciphertext bit for every block cipher operation. For this reason, if the block size of the underlying block cipher working in CFB mode is $L$, the CFB resulting throughput would be $1/L$ of the underlying block cipher.

To solve this throughput limitation, SCFB (Statistical Cipher Feedback) [25] and OCFB (Optimized Cipher Feedback) [26] modes were proposed. Particularly, SCFB was deeply analyzed in [27], [28], and in [29] was compared with OCFB, resulting in better properties for high-speed physical layer security. In spite of this advantage, it is recommended that an implementation of conventional SCFB be constrained to 50 percent of the throughput of its underlying block cipher. This constraint is necessary to ensure that no bits are lost due to queue overflow in the SCFB system.

To overcome this limitation, PSCFB (Pipelined Statistical Cipher Feedback) was proposed [30]. This mode allows an efficient utilization of a block cipher using a pipeline architecture, which results in implementations with a throughput near to 100 percent.

### 3.2 PSCFB Mode of Operation

The PSCFB mode of operation is essentially a hybrid of CFB and CTR modes, where the underlying block cipher is implemented with a pipelined architecture of $P$ stages and a block size of $L$ bits. Let us assume that the cipher is configured with a key $K$ and it has an encryption function $E_K(\cdot)$. The cipher operates in conventional CTR mode while scanning the ciphertext looking for a special $n$-bit length synchronization pattern. Assuming that the underlying block cipher is a good PRP (Pseudo Random Permutation), each value of the keystream block cipher output can be seen as randomly and independently chosen, giving also random and independent values of ciphertext blocks after performing the XOR. Therefore, the sync pattern will be observed at a statistically random point in the ciphertext stream. When this pattern is detected, the next $L$ bits are captured and used as an initialization vector ($IV$) that feeds back the counter value at the block cipher input. Therefore, it can be considered that the block cipher temporarily works in CFB mode. On the other hand, it is necessary to disable the sync pattern scanning since the $IV$ is captured until $E_K(IV)$ is available as new keystream block. This interval is called the blackout period.

In Fig. 3 and Fig. 4 the structure of PSCFB mode and the complete synchronization cycle are shown, respectively.

## 4 Security Considerations

### 4.1 Background of PSCFB Security

Regarding to the security of SCFB and PSCFB modes, the probability of generating repeated keystream blocks has been analyzed in [22] and [27]. If the counter reaches some value already used in previous synchronization cycles, the keystream will be repeated until next sync pattern detection. This issue would compromise the security of these modes.

The conclusion of these analyses is that the probability of repeated keystream is very low for both modes with typical size parameters, such as 128-bit block size. However,

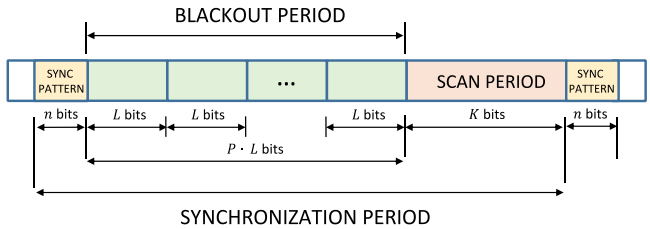



Fig. 4. Structure of the synchronization period in PSCFB mode. A complete synchronization period is formed by the sync pattern, blackout period and scan period. As the cipher has $P$ stages, the encrypted value of the $IV$, $E_K(IV)$ is used as keystream $P$ blocks later, when it has been fed back as a new counter value.

although PSCFB mode is built from two secure modes, CTR and CFB, with formal security proofs [20] [26], no formal security proof exists for PSCFB, and it has been let as an open problem [30].

In this section we discuss the security for PSCFB mode. This mode can be considered similar to other one called CTR$, for which a security proof exists. Based on this proof it is possible to derive a formal security expression for PSCFB for the first time and determine under which conditions it can be considered, at least, as secure as other traditional modes, e.g., CTR.

In the next subsection, CTR$ mode and its formal security proof are introduced. Authors consider that this explanation is convenient for the understanding of Section 4.3, where the IND-CPA advantage for PSCFB is derived.

### 4.2 CTRC and CTR$ Modes

Concrete security analysis for CTRC and CTR$ were originally established in [20]. The CTRC scheme is a stateful (counter based and deterministic) while the CTR$ is a stateless (randomized) variant of CTRC.

Let us consider a family of functions $F$ such that: $F : K \times \{0,\ 1\}^l \rightarrow \{0,\ 1\}^L$ where $L$ is the block size and $K$ the keyspace. Given a plaintext $M$ formed by $m$ $L$-bit blocks $\{M_0, M_1, \ldots,\ M_{\mathrm{m}-1}\}$, then the $m$ blocks of the output ciphertext $C_i$ are obtained in each S$\mathcal{E}$ (encryption scheme) CTRC and CTR$, applying their encryption functions. Encryption function for CTR$ is shown in Algorithm 1.

**Algorithm 1.** Function $\mathcal{E} - CTR\$^F(M)$

$R \leftarrow^{\$} \{0,\ 1\}^l$
$CNT_0 = R + 1$
$K_i = F_K\ [CNT_i]\ for\ i = 0, 1, \ldots, m-1$
$CNT_{i+1} = CNT_i + 1\ for\ i = 0,\ 1, \ldots, m-2$
$C_i = (M_i \oplus K_i)\ for\ i = 0,\ 1, \ldots, m-1$
Return $\{C_0, C_1, \ldots,\ C_{\mathrm{m}-1}\}$

In this algorithm $CNT_i$ and $K_i$ are the values of the counter and keystream block in each encryption step. $F_K$ is the underlying encryption function and $R$ is an $l$-bit random value. In CTR$ the counter is set to a random value $R$ at the beginning of each message encryption.

Usually the security of these modes is studied in the sense of IND-CPA (Indistinguishability under Chosen-Plaintext Attack) security [31]. An advantage expression is obtained thanks to a game between an active adversary $A$ and an encryption oracle performing the encryption scheme S$\mathcal{E}$, configured with a key $K$ and an experiment bit $b$.

It is demonstrated in [32] that an adversary $B$ attacking the PRF security of $F_K$ can be built thanks to the adversary $A$ and their advantages are related as follows:

$$ADV^{IND-CPA}_{S\mathcal{E}(F)}(A) = 2 \cdot ADV^{PRF}_{F}(B) + ADV^{IND-CPA}_{S\mathcal{E}(Func)}(A), \tag{1}$$

where $ADV^{IND-CPA}_{SE(F)}(A)$ is the advantage of $A$ attacking S$\mathcal{E}$ when its underlying encryption function is a PRF $F_K$, $ADV^{IND-CPA}_{SE(Func)}(A)$ is the advantage of $A$ over S$\mathcal{E}$ when the underlying encryption function is a random function $Func(l, L)$ and $ADV^{PRF}_{F}(B)$ is the prf-advantage of $B$ as defined in [31].

In the formal security proofs of CTRC and CTR$ modes the term $ADV^{IND-CPA}_{SE(Func)}(A)$ is obtained [32], then allowing to reach the final advantage expression. It is proven that:

$$ADV^{IND-CPA}_{S\mathcal{E}(Func)}(A) \leq \Pr(col) \tag{2}$$

where $\Pr(col)$ is the probability of a collision among the counter values used during the game.

Given the experiment bit $b$, during the attacking game the adversary performs $q$ queries of a pair of messages. For each pair $(M_i^0, M_i^1)$ it receives from the oracle the ciphertext $C_i$ corresponding to the message $M_i^b$. Assuming that each encrypted message $M_i^b$ has a length of $m_i$ blocks, the counters used during the game session can be represented as in the following table:

$$\begin{array}{lll} r_1 + 1, & r_1 + 2, \ \ldots, & r_1 + m_1 \\ r_2 + 1, & r_2 + 1, \ \ldots, & r_2 + m_2 \\ \ldots & \ldots & \ldots \\ r_q + 1, & r_q + 1, \ \ldots, & r_q + m_q, \end{array} \tag{3}$$

where $r_i$ is the randomized counter value loaded at the beginning of the message $M_i^b$ with length $m_i$. The subsequent counters from $r_i$ in advance will be incremented up to $r_i + m_i$. According to (3), the probability of collision of every counter value, $\Pr(col)$, can be bounded with the following expression:

$$\Pr(col) \leq \frac{(q-1) \cdot \sum_{i\ =\ 1}^{i\ =\ q} m_i}{2^l}, \tag{4}$$

Thanks to (4) it is possible to derive an upper bound for $ADV^{IND-CPA}_{S\mathcal{E}(Func)}(A)$, and by replacing it in (1), the final $ADV^{IND-CPA}_{S\mathcal{E}(F)}(A)$ expression of CTR$ mode.

### 4.3 PSCFB vs CTR$

In PSCFB, the sync pattern will be observed at a statistically random point in the keystream. On the other hand, if the block cipher is considered a good PRF and there are no collisions among the counter values, ciphertext blocks can be considered random and independent. Therefore, we can consider the new $IV$ as a random value. As we consider the block cipher $F$ such that: $F : K \times \{0,\ 1\}^l \rightarrow \{0,\ 1\}^L$, in this section we let the $IV$ be an $l$-bit word and the blackout period will have a length of $(P-1) \cdot L + l$ bits.

Let us assume that the adversary tries a game over an oracle performing a PSCFB encryption scheme. The adversary sends $q$ queries to the oracle, but now, unlike the CTR$ mode, the counter of the PSCFB scheme is not reinitiated randomly at the beginning of each message. The counter could be reseeded at a random point of each message. Depending on the length of the messages this initialization could happen more or fewer times. For example if the length of a message is shorter than the mean synchronization period, possibly in that message only a new sync pattern is received and then the counter is reinitiated only once to a random $IV$. However, for longer messages this could happen more times.

In general, in CTRC, CTR$ and PSCFB we can understand that the counter behaves in a cyclic fashion during the encryption session. We call this type of cycle a counter

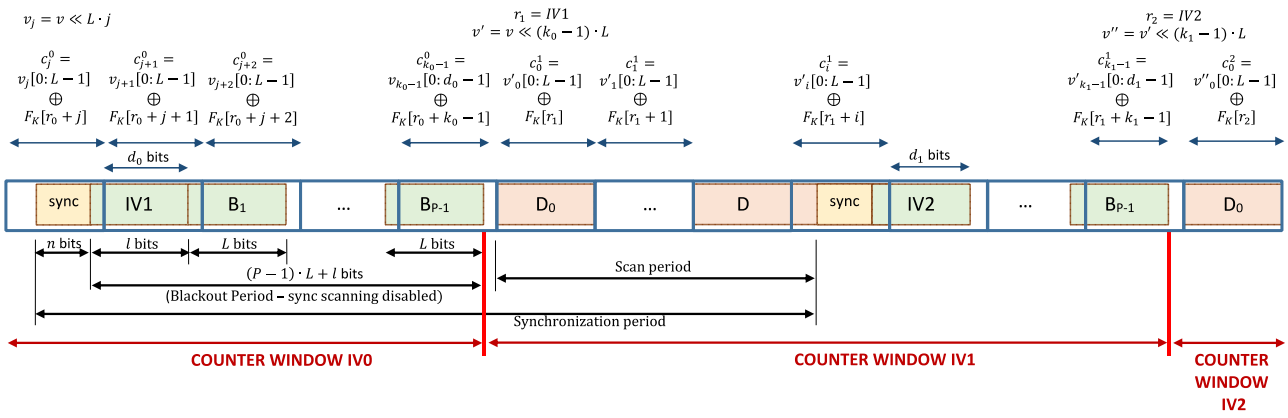



Fig. 5. Counter windows in PSCFB mode. Each $IV$ used in each counter window is captured in the previous one. For example, $IV_1$ is captured at the beginning of the blackout period in counter window $IV_0$. Then its encrypted value $F_K(IV_1)$ is used as the first keystream block in next scan period (in counter window $IV_1$). In counter window $IV_i$ the counter $r_i$ takes values from $IV_i$ to $IV_i + k_i - 1$ where $k_i = \mu_i/L$ , $\mu_i$ is the length in bits of this counter window and $L$ is the block size.

window. Inside each window, the counter is incremented and not repeated. In the case of CTRC there is only one counter window, which means that the counter is initialized only once, at the beginning of the first message, and never repeated. In CTR$ there are so many windows as encrypted messages because the counter is reinitialized randomly at the beginning of each message. In the same way in PSCFB there are so many windows as synchronization cycles are produced during the whole session, as the counter is reseeded at the beginning of each blackout period. The counter window will start at the beginning of a scan period and will finish at the end of the next blackout period, as shown in Fig. 5.

In addition, the counter window length is different in each mode. In CTRC and CTR$ the length is a number of bits that is multiple of the block size, while in PSCFB it is multiple of the block size plus a random number of bits between 0 and the block size. This fact depends on when the end of the blackout period happens, as it could be not aligned with the end of an encrypted block of bits. An example is shown in Fig. 5, where the counter windows $IV_0$ and $IV_1$ do not finish exactly in block boundaries.

Starting from (1), which can be considered generic for any counter encryption scheme we will have:

$$ADV^{IND-CPA}_{PSCFB(F)}(A) = 2 \cdot ADV^{PRF}_{F}(B) + ADV^{IND-CPA}_{PSCFB(Func)}(A). \tag{5}$$

As we can consider PSCFB a counter mode, it is possible to make the same assumptions as in [32] to reach equation (6), that is, if the block cipher is a good PRF and there are no repeated counter values during the game session, then the adversary has zero advantage in winning the game and the encryption scheme behaves as a one-time pad. Then next condition is fulfilled:

$$ADV^{IND-CPA}_{PSCFB(Func)}(A) \leq \Pr(col). \tag{6}$$

During the game session the adversary sends $q$ messages of length $m_i$ blocks. The block size is $L$ bits and the total number of bits in the session is $\mu$. Let us assume that in the case of PSCFB $N$ sync cycles happen during the whole session. Therefore we can consider that the counter table (3) for PSCFB can be represented as:

$$\begin{array}{llll} r_1, & r_1+1, & \ldots, & r_1+k_1-1 \\ r_2, & r_2+1, & \ldots, & r_2+k_2-1 \\ \ldots & \ldots & & \ldots \\ r_N, & r_N+1 & \ldots, & r_N+k_N-1. \end{array} \tag{7}$$

where $k_i$ is the length in data blocks for the $i$th counter window. As the bit length of a counter window could not be a multiple of the block size, the length $k_i$ will be: $k_i = \lceil \mu_i/L \rceil$ , where $\mu_i$ is its length in bits and the operator $\lceil \cdot \rceil$ means its rounded up value.

As in (4), $\Pr(col)$ for PSCFB case is obtained:

$$\Pr(col) \leq \frac{(N-1) \cdot \sum_{i=1}^{i=N} k_i}{2^l}. \tag{8}$$

Since $k_i = \lceil \mu_i/L \rceil \leq \mu_i/L + 1$, then:

$$\Pr(col) \leq \frac{(N-1) \cdot \sum_{i=1}^{i=N} (\mu_i/L + 1)}{2^l} \leq \frac{N \cdot \left(\frac{\mu}{L} + N\right)}{2^l}. \tag{9}$$

Let us consider $N_{max}$ the maximum number of sync cycles in $\mu$ bits and $C_{min}$ the minimun posible size of a sync cycle. Therefore $N_{max} = \mu/C_{min}$. Let $C = (P-1) \cdot L + l$, since $C_{min} = (P-1) \cdot L + l + n$, then $N_{max} = \mu/C_{min} \leq \mu/C$. Therefore, it is possible to rewrite equation (9) as:

$$\Pr(col) \leq \frac{\frac{\mu}{C} \cdot \left(\frac{\mu}{L} + \frac{\mu}{C}\right)}{2^l} = \frac{\mu^2}{2^l} \cdot \frac{C+L}{C^2 \cdot L}. \tag{10}$$

Finally, according to (5), the IND-CPA advantage of adversary $A$ against PSCFB can be expressed as:

$$ADV^{IND-CPA}_{PSCFB(F)}(A) \leq 2 \cdot ADV^{PRF}_{F}(B) + \frac{\mu^2}{2^l} \cdot \frac{C+L}{C^2 \cdot L}. \tag{11}$$

Although block ciphers are analyzed as PRFs, their input and output size are equal, therefore if we consider that $L = l, C = P \cdot L$, then the advantage result is:

$$ADV^{IND-CPA}_{PSCFB(F)}(A) \leq 2 \cdot ADV^{PRF}_{F}(B) + \frac{\mu^2}{L^2 2^L} \cdot \frac{P+1}{P^2}, \tag{12}$$

Supposing that the term $ADV^{PRF}_{F}(B)$ is negligible, $ADV^{PRF}_{PSCFB}(A)$ will be negligible when the following condition is accomplished:

$$L^2 2^L \cdot \left(\frac{P^2}{P+1}\right) \gg \mu^2 \tag{13}$$

We can conclude that the PSCFB mode can be considered secure under certain conditions, in the same way that happens with CTR$ mode. According to (12), the advantage of an adversary over PSCFB will be reduced if the size of the blackout period $P$ is lengthened, which is directly related with the number of pipeline stages with which the block cipher has been implemented. It is a coherent result, because the longer the sync period the fewer random counter initializations will be produced, consequently reducing the probability of a collision and increasing the security. However, as proved in [30] longer $P$ means worst values of SRD (Synchronization Recovery Delay) and EPF (Error Propagation Factor) in the PSCFB system. It is possible to conclude that there is a compromise between security and inherent properties of PSCFB measured by SRD and EPF.

### 4.4 PSCFB vs CTR

As mentioned before, usually block ciphers are analyzed as PRFs, however it is well known that the PRPs (Pseudo Random Permutation) are what best models them. It is known that the prp-security and prf-security of a block cipher are related thanks to the PRF-PRP switching lemma [20]. Due to this, the overall advantage of the adversary $A$ over the CTRC scheme is degraded by an amount given by the birthday attack, it is $\mu^2/L^2 2^{L+1}$, which means that its advantage can be written as:

$$ADV^{IND-CPA}_{CTRC(F)}(A) \leq 2 \cdot ADV^{PRP}_{F}(B) + \frac{\mu^2}{L^2 2^L}. \tag{14}$$

The same consideration can be taken for PSCFB, for which advantage can also be rewritten as:

$$ADV^{IND-CPA}_{PSCFB(F)}(A) \leq 2 \cdot ADV^{PRP}_{F}(B) + \frac{\mu^2}{L^2 2^L} + \\ + \frac{\mu^2}{L^2 2^L} \cdot \left(\frac{1}{P} + \frac{1}{P^2}\right). \tag{15}$$

which means that:

$$ADV^{IND-CPA}_{PSCFB(F)}(A) \leq 2 \cdot ADV^{PRP}_{F}(B) + \\ + \frac{\mu^2}{L^2 2^L} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right). \tag{16}$$

In Table 1, a comparison between the derived advantage of PSCFB and other operation modes is shown. As mentioned in [33] CTRC can be considered the best and most modern way to achieve privacy-only encryption. For this reason it is useful to make a comparison between the IND-CPA advantages between this mode and PSCFB.

Let us suppose that the underlying block ciphers for the two modes, CTRC and PSCFB, are good PRPs and have the same prp-security, then we can stablish under what values of block size and pipelining the block cipher in PSCFB mode has at least the same security as that in CTRC when encrypting the same amount of information. For this purpose we should compare the second terms of (14) and (15) as follows:

TABLE 1
IND-CPA Advantage Comparison of Different Modes

| Encryption Mode $\mathcal{SE}(F)$ | IND-CPA Advantage expression[1] $ADV^{IND-CPA}_{\mathcal{SE}(F)}(A)$ |
|---|---|
| PSCFB | $2 \cdot ADV^{PRP}_{F}(B) + \frac{\mu^2}{L^2 2^L} \cdot (1 + \frac{1}{P} + \frac{1}{P^2})$ |
| CTRC | $2 \cdot ADV^{PRP}_{F}(B) + \frac{\mu^2}{L^2 2^L}$ |
| CTR$ | $2 \cdot ADV^{PRP}_{F}(B) + \frac{2\mu^2}{L^2 2^L}$ |
| CBC | $2 \cdot ADV^{PRP}_{F}(B) + \frac{2\mu^2}{L^2 2^L}$ |
| CFB[2] | $2 \cdot ADV^{PRF}_{F}(B) + \frac{\mu^2}{m^2 2^{L+1}}$ |

[1] *In each expression L is the block size and $\mu$ the number of encrypted bits.*
[2] *The term $ADV^{PRF}_{F}$ refers to the prf-advantage where $F_K$ is the function $select(E_K(\cdot)).E_K(\cdot)$ is the block cipher with blocksize L and $select(\cdot)$ is a function that outputs m fixed bits from its input.*

$$\frac{\mu^2}{L^2_{PSCFB} \cdot 2^{L_{PSCFB}}} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right) \leq \frac{\mu^2}{L^2_{CTR} \cdot 2^{L_{CTR}}}, \tag{17}$$

where $L_{PSCFB}$ and $L_{CTR}$ are the block sizes of the underlying block ciphers in PSCFB and CTRC modes respectively. According to (17), we can deduce the block size that a secure pipelined block cipher should have in case of being used in PSCFB mode if we want to provide the same security as another one with the same prp-security working in CTRC mode.

For example, let us assume that we have an AES (Advance Encryption Standard) block cipher working in CTRC mode, then $L_{CTR} = 128$ bits. We can stablish the value of $L_{PSCFB}$ provided that:

$$L^2_{PSCFB} \cdot 2^{L_{PSCFB}} \geq L^2_{CTR} \cdot 2^{L_{CTR}} \left(1 + \frac{1}{P} + \frac{1}{P^2}\right) = \\ 2^{142} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right). \tag{18}$$

As $P \geq 1$ the right term will be maximum with $P = 1$, then

$$L^2_{PSCFB} \cdot 2^{L_{PSCFB}} \geq 2^{142} \cdot 3. \tag{19}$$

This inequality is fulfilled with $L_{PSCFB} \geq 130$ bits. For $P > 1$ the condition is fulfilled with $L_{PSCFB} > 128$.

Therefore, we can conclude that given two block ciphers with the same prp-security that work in two different modes, CTRC and PSCFB, if the block size of the one working in CTRC is 128 bits, then the other must have a block size larger than 128 to get the same or better IND-CPA security when encrypting the same amount of data.

In case of setting the same block size for both block ciphers, to get better IND-CPA security in PSCFB mode, less amount of data could be encrypted per key session.

## 5 Application Case: Ethernet 10 GBase-R

In this paper we have focused on the case of high-speed communications, particularly in the 10 GBase-R standard used in 10 Gigabit Ethernet optical links. In this standard, the PCS level is responsible for generating, encoding and scrambling the control and data blocks that are transmitted to the optical line. As block line coding is 64b/66b, the purpose of the encryption will be to cipher the complete 64b/66b block flow as shown in Fig. 2.

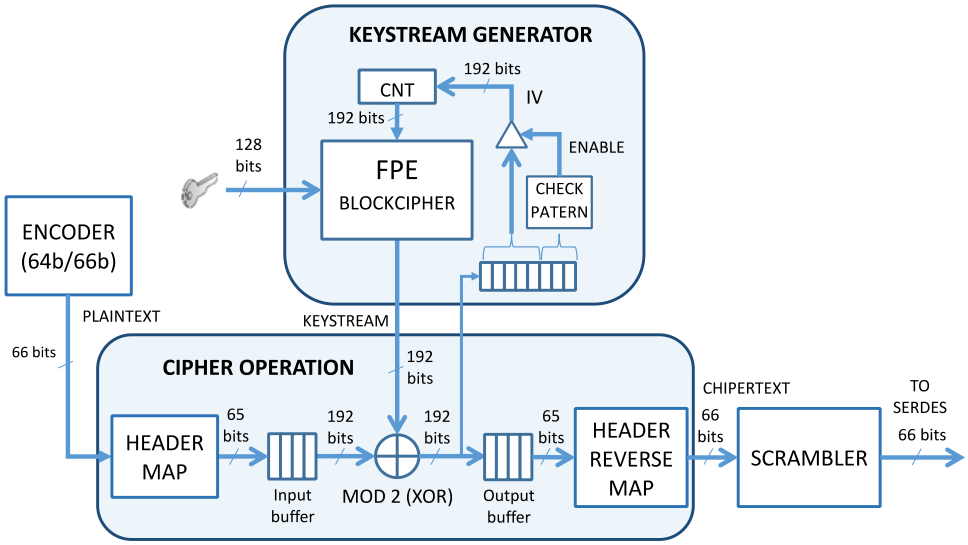



Fig. 6. Overall structure of the PSCFB system in the encryption module. In the decryption side the scheme is the same, however the check pattern input is taken directly from the input buffer output, before performing the XOR operation.

Because no extra data fields are added to the packets when they are encrypted, then no overhead is introduced and no throughput loss is produced. Moreover it not only encrypts the contents of the packets but also the activity or data traffic pattern. This is because by encrypting at 64b/66b block level, control blocks are also encrypted, such as packet start and end blocks or control blocks full of idle characters when no traffic is transmitted or during the IFGs (Inter Frame Gaps). Thanks to this last property, security could be improved, as passive eavesdroppers would be prevented from performing traffic analysis attacks. It would be useful in scenarios where traffic pattern analysis could reveal sensitive information about the behavior of a critical infrastructure or facility.

In this work the keystream generator of Fig. 2 has been implemented thanks to a block cipher working in PSCFB mode. As concluded in Section 4.4, to provide the same security level as a 128-bit block cipher working in CTRC mode when transmitting the same amount of data, it is necessary to use a block cipher with larger block size. In particular, for $P > 1$ this block size must be larger than 128 bits.

A possible block cipher candidate could be the well-known cipher Rijndael [34]. The main difference between Rijndael and AES is the range of configuration values for the block size and key length. Particularly, AES is a subset of Rijndael that uses a fixed block size of 128 bits and a key length of 128, 192 or 256 bits. On the contrary, the original specification of Rijndael also included 192 and 256 bits as possible block sizes. However, only the subset corresponding to the current version of AES was standardized by NIST, and as far as the authors know, there is no recommended block cipher with a block size greater than 128 bits.

Other solution for building a suitable and standardized block cipher with more than 128-bits block size is the use of the recent FPE (Format Preserving Encryption) modes approved by NIST [35]. FPE modes encrypt plaintext in a ciphertext preserving its original format and length. Typical applications of this modes are the encryption of PANs (Primary Account Numbers) or SSNs (Social Security Numbers) where a standard block cipher would not preserve their format.

Currently, two FPE modes are recommended by NIST, FF1 and FF3 [35]. Both modes are based on a non-binary Feistel structure, whose underlying round function consists of an AES block cipher. These are considered AES modes allowing to configure the block size and data radix of the resulting FPE block cipher.

Between the two NIST recommendations, we have selected FF3, as it is built with less rounds in its Feistel network and the cost in hardware resources is lower. In this mode the block size is limited according to the radix used, as shown in (20):

$$\begin{aligned} R &\in [2 \dots 2^{16}] \\ R^{minlen} &\geq 100 \\ 2 \leq minlen \ &\leq maxlen \leq 2\lfloor log_R(2^{96}) \rfloor . \end{aligned} \tag{20}$$

where $R$ is the radix and $minlen$ and $maxlen$ the bounds for the block size. With $R = 2$, the block size is between 2 and 192.

In this work the selected value for block size is the maximum, $L = 192$. With this size it is possible to get a major margin of cycles available per stage in a possible pipelined architecture. This fact allows a better reuse of the hardware resources.

According to this parameter, the final structure of the self-synchronized encryption system is shown in Fig. 6. It is similar to Fig. 2, but the keystream generator has been replaced with the final self-synchronous PSCFB structure based on AES in FPE mode. Also, in this figure it is shown that buffers are required at the input and output sides of the encryption structure. They are necessary to achieve 100 percent efficiency of the final encryption scheme.

## 6 System Implementation

### 6.1 PSCFB System Implementation

The underlying block cipher of the implemented PSCFB mode has been built using an FF3 structure. FF3 algorithm is described in [35]. In this work the cipher tweak value has been set to zero, while only the key is configurable. Taking

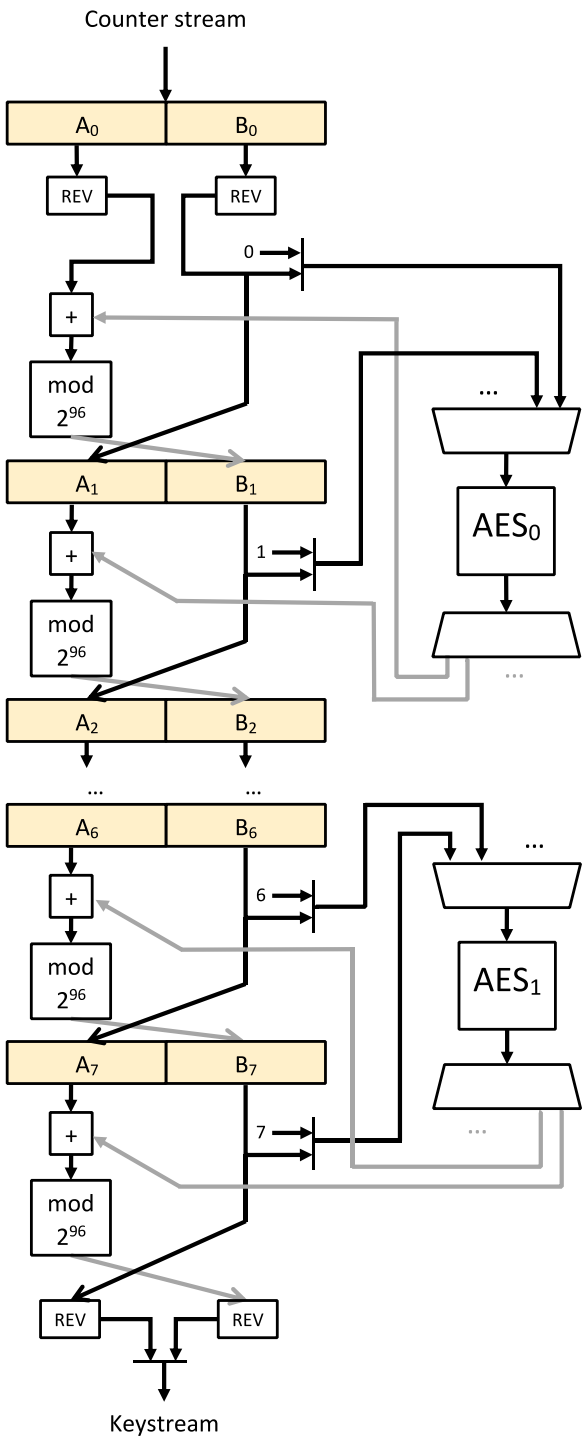



Fig. 7. Structure of the Feistel network for the 192-bit FPE cipher.

into account the selected parameters for our FF3 implementation, $R = 2$ and $L = 192$, its structure is shown in Fig. 7 where one AES core is used each four stages of the Feistel network. The final latency introduced in its pipelined structure is 58 cycles.

Regarding to the system throughput, the efficiency $\eta$ of an encryption scheme represents the amount of ciphered information that it can produce relative to the number of bits generated by its underlying block cipher. In the case of PSCFB, as the sync cycle length can be different to a multiple of the block size in bits, then not all blocks produced by the block cipher will be used completely for encryption. Indeed, the end of the blackout period could not be aligned with a kesytream block ending. It means that in the last block cipher operation of the blackout period the block cipher output could be used partially producing less ciphertext bits than the block size.

As explained in [30], it can be shown that the efficiency can be lower bounded by $\eta = P/(P+1)$, which means that for $T$ bits produced by the block cipher output at least $T \cdot P/(P+1)$ ciphertext bits of PSCFB are generated. Therefore, to generate the ciphertext stream at the 100 percent throughput rate of a 10 Gbps Ethernet system, it is necessary to overclock the PSCFB subsystem with respect to the PCS sublayer. For this reason the two buffers shown in Fig. 6 have been introduced, to isolate the different clock domains of PSCFB and PCS.

On the other hand, these buffers also work as the input/output queues defined in the original specification of PSCFB, necessary to store information temporarily during periods of resynchronization, where partial block cipher outputs are used to encrypt data due to the fact that the sync pattern is not aligned with the end of the block cipher output.

Taking into account that the bus widths for PSCFB and PCS are 192 and 65 bits respectively, and equating the PCS 64b/66b bit throughput with that of the PSCFB, the following condition must be achieved:

$$T_{PSCFB} \cdot 192 = \eta \cdot T_{FPE} \cdot 192 = T_{PCS} \cdot 65 \tag{21}$$

where $T_{PSCFB}$, $T_{FPE}$ and $T_{PCS}$ are the word throughputs of PSCFB, FPE and PCS domains, respectively, measured in words per second.

As in our FF3 implementation we are using two AES cores that have to attend to four stages each one, the total throughput of each AES core, $T_{AES}$, will be four times that of the FF3. Then (21) can be rewritten as:

$$\frac{P}{P+1} \cdot \frac{T_{AES}}{4} \cdot 192 = T_{PCS} \cdot 65 \tag{22}$$

As $P = 58$ and the PCS word rate is 156.25 MHz, the resulting clock frequency at which AES cores should work for getting a 100 percent throughput is 215.24 MHz. In this work the total frequency used for PSCFB system has been set to 217 MHz.

The described system in Fig. 6 has been implemented in a Xilinx Virtex 7 FPGA (Field Programmable Gate Array). In Table 2, the hardware resources of the PSCFB system are shown. It includes the resources used by the main modules in Fig. 6, KEYSTREAM_GENERATOR and CIPHER_OPERATION. Moreover, in Table 2, a comparison in terms of LUTs (Look-Up Tables), registers and BRAMs (Block RAMs) is made between this work and other implementations [36], [37]. Particularly, in [36] the FPGA used is a Virtex-6 device, different model than in this work, however the CLB structure in both devices is similar in terms of LUTs and registers, with four six-input LUTs and eight registers per slice. Although the implementation in this work entails more hardware resources, the ratio *Encryption_Rate/Slice* is clearly superior.

In general, although the inherent structure of an AES in FPE mode consumes more resources than the AES core,

TABLE 2
Comparison with Other Solutions

| | FF1 [36] | FF3 [36] | FF3 [37] | This work |
|---|---|---|---|---|
| Slice Registers | 11285 | 5592 | 11127 | 19154 |
| Slice LUTs | 7426 | 3587 | 16978 | 17599 |
| 18K Block RAMs[1] | 343 | 170 | 77 | 153 |
| Slices[2] | 3268 | 1596 | 5636 | 6794 |
| Operation Freq. (MHz) | 279.6 | 283.5 | 125 | 217 |
| Cycles/Encryption | 707 | 269 | 1 | 1 |
| Bytes/Encryption | 13 | 13 | 1 | 192 |
| Encryption Rate (Mbps) | 41.1 | 109.6 | 1000 | 10000 |
| Encryption Rate/Slice (Kbps/Slice) | 12.57 | 68.7 | 177.4 | 1471.8 |

[1]*The 153 Block RAMs used in this work are due to the AES cores.*
[2]*Slices are estimated from the number of register and LUTs, assuming they are not packed together.*

TABLE 3
Optical Link Parameters

| | Parameter | Value |
|---|---|---|
| Transmitter | Average Launch Power (dBm) | −1 |
| | Optical Wavelength (nm) | 850 |
| | RMS Spectral Width (dB) | 0.45 |
| | Optical Extinction Ratio (dB) | 5.5 |
| Receiver | Receiver Sensitivity (dBm) | −11.1 |
| | RX Wavelength Range (nm) | 840-860 |
| Fiber Link | Link Length (m) | 1 |
| | Fiber diameter - core/cladding (nm/$\mu$m) | 62.5/125 |
| | Modal Bandwidth (MHz x km) | 200 |
| | Attenuation (dB/km) | 3 |

authors have considered this option to grant that the security of the PSCFB system is not degraded in respect a typical CTRC implementation of AES. On the other hand, although implementations of a 192-bit block cipher such as Rijndael could have been possible, they are not recommended by NIST, as the FPE option is.

### 6.2 Test Setup

To carry out the test of the system, the proposed encryption scheme has been integrated in a 10 GBase-R Ethernet interface. In Fig. 8, the complete PCS structure is shown. Apart from the 64b/66b encoding and scrambling functions it contains the encryption and decryption modules.

Moreover, a traffic generator has been also implemented and linked to the Ethernet interface to test it with real data frames. Two chains composed each one by the 10 GBase-R Ethernet interface and a frame generator have been implemented over the Xilinx Virtex 7 FPGA. Both Ethernet interfaces have been connected to two SFP+ (Small Form-factor Pluggable) modules configured to work at 10 Gbps speed and linked between them with a multimode fiber patch cord. Some of the parameters of the transmitter/receiver and the fiber link are shown in Table 3.

A scheme of the setup for test is shown in Fig. 9. Thanks to the Ethernet frame generators, it has been possible to check the encrypted link with real data packets without producing any frame loss or CRC (Cyclic Redundancy Check) errors.

The extra hardware resources introduced in the PCS datapath generate an extra latency of 266 ns for both transmission and reception. It is noteworthy that the proposed encryption system gets values comparable to those achieved by OTN equipment [12]. Moreover, no overhead is introduced in the encryption process allowing data be encrypted at line rate.

This is a clear advantage over other encryption mechanisms whose inherent overhead limits the total throughput to values lower than 100 percent [16].

## 7 Encryption Results

### 7.1 Encryption Properties

The encrypted link has been tested with different Ethernet traffic flows. Thanks to the Ethernet frame generators four different traffic patterns have been encrypted. These have been named A, B, C and D. Pattern A corresponds with the case of no frame transmission, where only 64b/66b control blocks full of idle characters are transmitted over the link. Patterns B, C and D correspond to continuous frame transmission of 1024-bytes length at rates of 10.2, 50 and 98 percent of the maximum 10 Gigabit line rate, respectively, and with random payloads.

Two conclusions arise from simulation and hardware debugging. On the one hand, encryption and decryption work correctly and synchronously without harming data traffic or link establishment between 10G Ethernet interfaces. No CRC errors are produced when transmitting the mentioned traffic patterns. On the other hand encrypted traffic patterns are masked, which can improve the overall security against passive eavesdroppers.

Regarding this capability, it is interesting to monitor signal waveforms after the encryption module, at the input

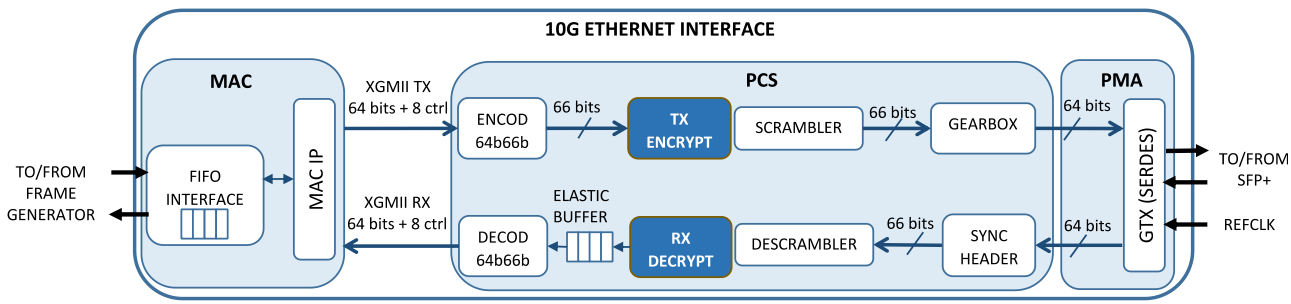



Fig. 8. Structure of the 10G Ethernet interface including the encryption function. It is composed by the MAC module, the PCS and the SERDES. In the PCS layer, TX_ENCRYPT and RX_DECRYPT are the encryption/decryption modules. Both include the CIPHER_OPERATION and KEYSTREAM_GENERATOR modules shown in Fig. 8.

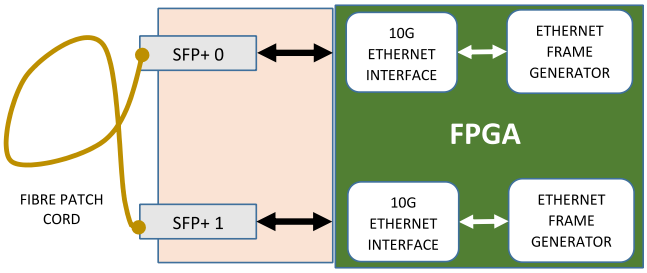



Fig. 9. Test setup scheme with SFP+ modules working at 10Gbps rate.

of the scrambler. The 64b/66b data bus is formed by the 2-bit sync header and the 64-bit block payload. If encryption is disabled, the synchronization header takes the value '10' when frames are transmitted and '01' during the IFG between frames. In t-he case of pattern A, as no frames are transmitted, sync header is always equal to '01'. However when encryption is enabled, the sync header takes random values between '10' and '01'. Also, the 64-bit block payload is randomized. In this way the traffic pattern is indistinguishable, independently of whether frames are being transmitted or not. In Fig. 10 waveforms of an encrypted pattern are shown.

In order to check this masking property, the randomness test suite proposed by NIST [38] has been used to evaluate encrypted patterns. Also SE (Shannon Entropy) has been measured and compared among the mentioned non-encrypted patterns and the encrypted signal.

Owing to the limited memory in FPGA hardware resources, these tests have been performed at simulation stage, but this fact does not invalidate experimental results.

Regarding the NIST tests, sync header and block payload have been evaluated separately with these tests concluding that both can be considered random sequences after encrypting any of the mentioned patterns. As an example, results for NIST test applied to the patterns D and C before and after encryption are shown in Fig. 11. It is possible to conclude that although they are very different patterns, after encryption they are transformed into a sequence that passes these tests, which makes them indistinguishable from a random stream.

As for SE measurement, it has also been calculated for the mapped sync header and block payload separately as defined in (23).

$$SE = -\frac{1}{n} \cdot \sum_{\beta_n \epsilon 2^n} P(\beta_n) \cdot \log_2 P(\beta_n) \tag{23}$$

In both cases the bit stream has been grouped in tuples of $n$ bits called $\beta_n$ and the probability for each tuple, $P(\beta_n)$, has been calculated. SE has been measured for values of $n$ equal to 4, 8 and 12 bits in each of the mentioned non-encrypted patterns A, B, C and D, all of them with fixed length frame and random payloads. In addition two more patterns have been added: E and F. Pattern E corresponds to continuous frame transmission with random payloads and random length between 64 and 1516 bytes while pattern F corresponds to the randomized signal after encryption of pattern A, which can be considered the worst case in terms of entropy. In Figs. 12 and 13 the comparison among the measured SE is shown. It is possible to notice that SE in pattern F is as expected, almost the ideal value of 1 in both cases, block payload and sync header, as it is encrypted. In the rest of non-encrypted patterns SE of block payloads decreases as the transmission rate decreases from E to A. This effect is owing to the ratio of the bandwidth that is used by the IFGs (Inter Frame Gaps). As when lower bandwidth is used, the IFGs full of idle sets takes more bandwidth percentage versus the random payloads of the transmitted frames, resulting in a lower SE. In the case of sync header entropy, clearly all non-encrypted patterns achieve a very low value.

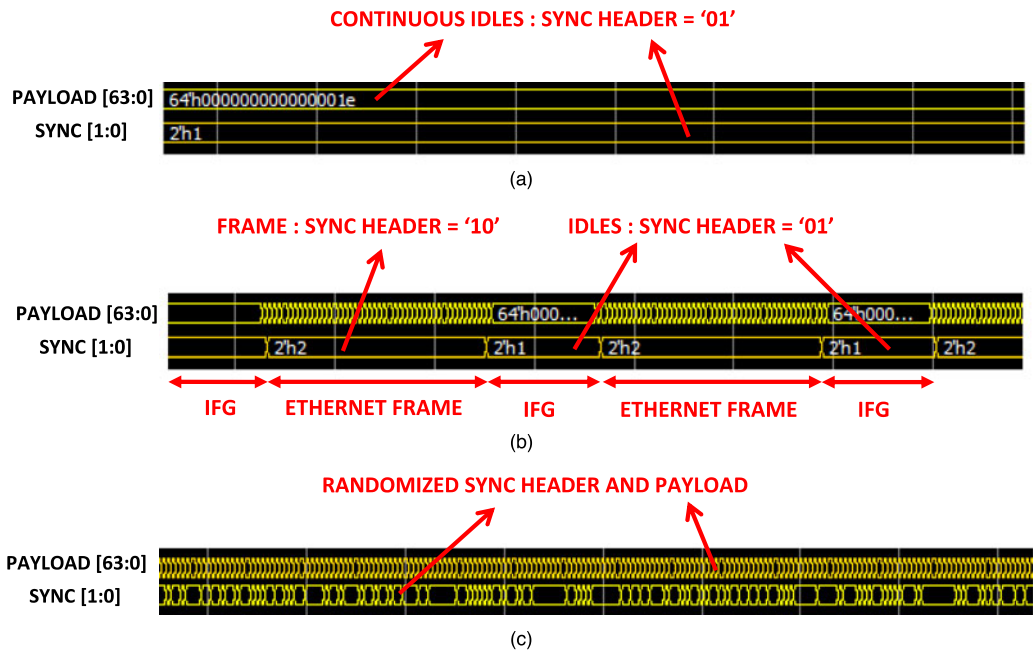



Fig. 10. (a) SYNC header pattern without encryption when no Ethernet frame is transmitted; (b) SYNC header pattern without encryption when transmitting an Ethernet frame burst; (c) SYNC header pattern after encryption regardless of the transmission or non-transmission of Ethernet frames.

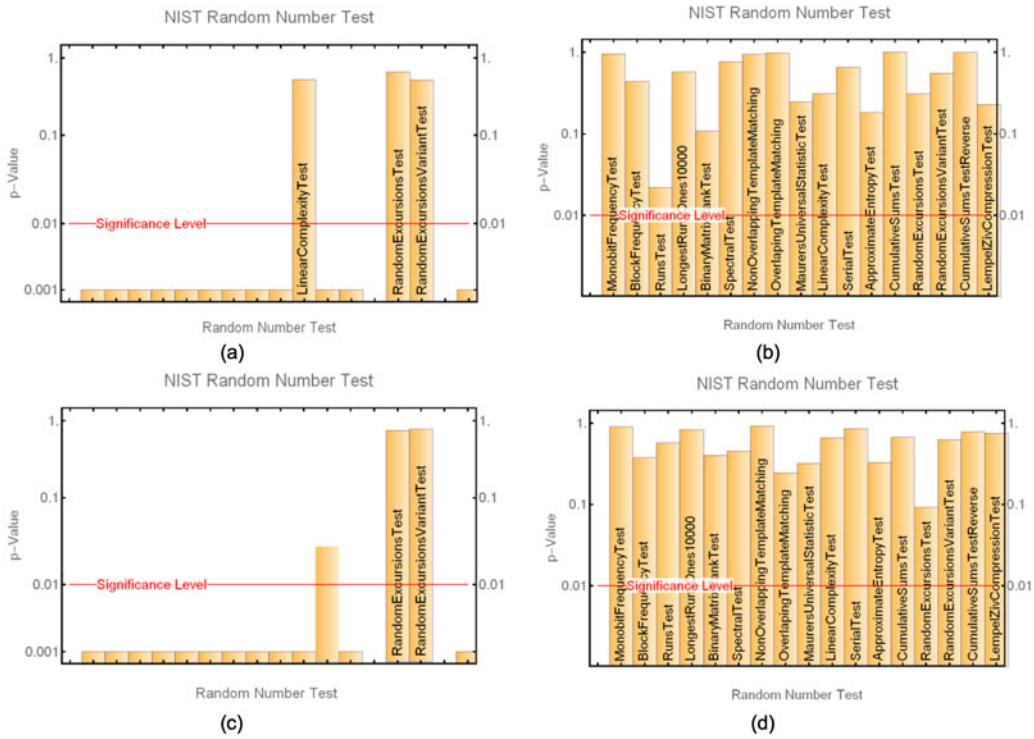



Fig. 11. NIST test results for the 64b/66b block payload of D pattern, (a) before encryption and (b) after it. NIST test results for the 64b/66b block payload of C pattern, (c) before encryption and (d) after it.

Thank to this result it is possible to conclude that encryption makes indistinguishable data traffic pattern, as the maximum value for SE is obtained and NIST tests are passed successfully when traffic is encrypted.

### 7.2 Self-Synchronization

The SRD (Synchronization Recovery Delay) is the metric used to examine the resynchronization properties of SCFB and PSCFB modes [27]. It is defined as the expected number of bits following a sync loss before synchronization is reestablished. According to [30], upper and lower bounds for SRD depend on the block size $L$, number of pipelines $P$ and size of the synchronization pattern $n$. Taking into account that in this system $L = 192$ and $P = 58$, upper and lower SRD bounds calculated with different values of $n$ are shown in Fig. 14. As for $n$ below 11 the upper bound grows, those values have not been shown. In this work $n = 12$ has been used, limiting the upper bound of SRD to 304 encrypted 64b/66b blocks.

Experimental results and simulation show that self-synchronization works correctly. By removing the optical fiber with encryption activated between Ethernet interfaces, both link status and encryption synchronization are lost. However, when restoring the optical fiber encryption synchronization is always recovered after PCS link status is achieved. Traffic bursts were correctly tested after synchronization recovering to check the integrity of the link.

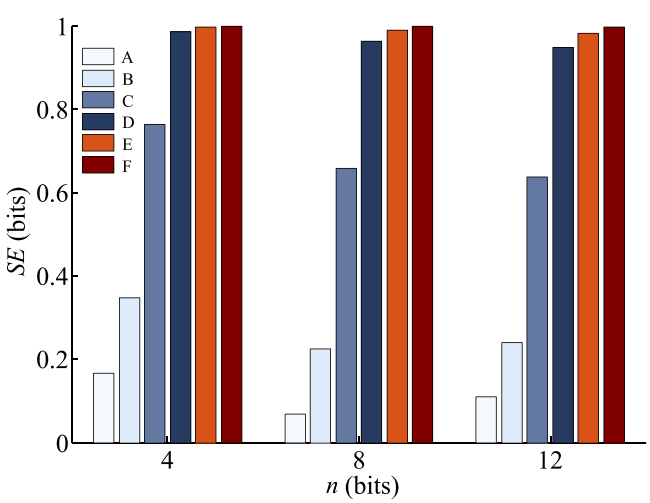



Fig. 12. Shannon Entropy of block payloads measured with n equal to 4, 8 and 12 in Ethernet traffic patterns from A to F.

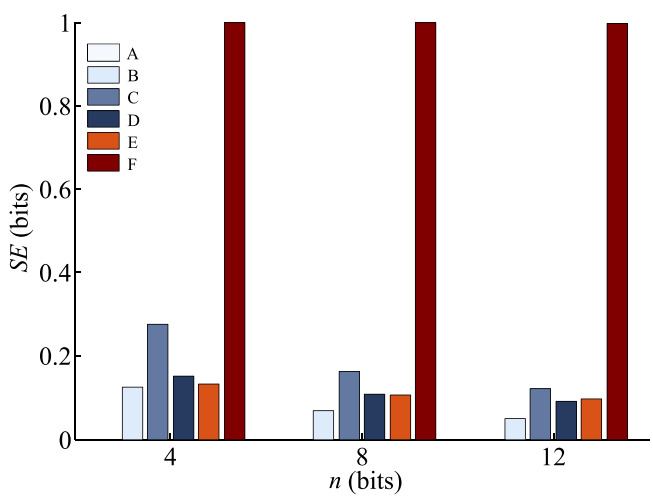



Fig. 13. Shannon Entropy of mapped sync header measured with n equal to 4, 8 and 12 in Ethernet traffic patterns from A to F. In the case of A pattern, sync header has a continuous value, which means that its entropy is zero. Because of this, A bar is zero in this figure.

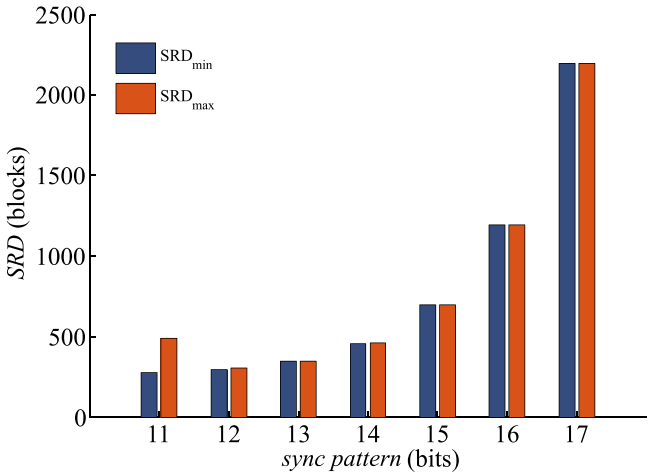



Fig. 14. Theoretical maximum and minimum SRD values for different sync pattern sizes.

## 8 Other Security Considerations

Although in this paper physical layer encryption mechanism is proposed and developed, other security considerations must be taken into account, especially regarding the key configuration and refreshing. One possible solution could be the use of classical public key encryption schemes, where the interchange of the symmetric master key is performed thanks to algorithms such as RSA used by protocols at higher communication layers. As an example, in the layer 2 encryption protocol MACsec, the master key negotiation is outside of its specification and is carried out by IEEE 802.1X standard. Therefore, the same idea could be considered for this work.

Other possible solutions for key distribution are optical mechanisms such as QKD (Quantum Key Distribution) that is based on the laws of physics rather than classic asymmetric cryptography algorithms. In spite of the crosstalk that classical signals can introduce to QKD channels, successful experiments where QKD channels coexist with classical data traffic are a reality [39], [40]. Thanks to WDM (Wavelength-Division-Multiplexing) techniques it is possible to reduce the crosstalk noise to a tolerable level, which means that QKD could also be considered for its use with the encryption mechanism presented in this work.

## 9 Conclusion

To the authors best knowledge, this is the first time that a self-synchronous encryption method is proposed for ciphering physical layer communications based on 64b/66b encoding. The new encryption system consists of a self-synchronous symmetric ciphering of the complete 64b/66b block stream. The encryption is based on the PSCFB mode, and it has been simulated and implemented over an FPGA. Also security considerations for this mode have been taken into account, deriving a formal security expression similar to that known for other operation modes.

This new mechanism is able to perform encryption at a line-rate introducing a latency in the range of nanoseconds, while the complete data traffic pattern is masked, improving the overall security.

Although this mechanism is proposed for 10 Gbps Ethernet links, 64b/66b encoding is used in other standards at higher rates, as 100 Gigabit Ethernet. It means that the same encryption scheme could be applied not only to the access networks but also to long-haul optical links in transport networks.

In addition to this, by preserving coding properties such as short run length and transition density, physical layer encryption is achieved without making changes in the subsequent circuitry. For example, commercial SFP+ modules or SERDES at 10 Gbps rate are compatible with the proposed encryption scheme.

## Acknowledgments

This work has been supported in part by the MINECO-FEDER under Grant TEC2014-52840-R and Grant TEC2017-85867-R and in part by the FPU fellowship program to M. García-Bosque under Grant FPU14/03523.

## References

[1] Cisco global cloud index: Forecast and methodology. 2015-2020, Cisco Systems, 2016, https://www.cisco.com/c/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.html, Accessed on: 1 Dec. 2018.

[2] Cisco 2018 annual cybersecurity report, Cisco Systems, Feb. 2018, https://www.cisco.com/c/dam/m/hu_hu/campaigns/security-hub/pdf/acr-2018.pdf Accessed on: 1 Dec. 2018.

[3] "IEEE standard for local and metropolitan area networks-Media Access Control (MAC) Security, IEEE Std 802.1AE-2018 (Revision of IEEE Std 802.1AE-2006), pp.1–239, 26 Dec. 2018, doi: 10.1109/IEEESTD.2018.8585421.

[4] S. Kent and K. Seo, "RFC 4301: Security architecture for the internet protocol," 2005. [Online]. Available: https://tools.ietf.org/html/rfc4301.

[5] N. Skorin-Kapov, M. Furdek, S. Zsigmond, and L. Wosinska, "Physical-layer security in evolving optical networks," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 110–117, Aug. 2016.

[6] M. Furdek, N. Skorin-Kapov, S. Zsigmond and L. Wosinska, "Vulnerabilities and security issues in optical networks," in *Proc. Int. Conf. Transparent Optical Netw.*, 2014, pp. 1–4.

[7] M. Zafar, H. Fathallah, and N. Belhadj, "Optical fiber tapping: Methods and precautions," in *Proc. High Capacity Optical Netw. Enabling Technol.*, 2011, pp. 164–168.

[8] M. P. Fok, Z. Wang, Y. Deng, and P. R. Prucnal, "Optical layer security in fiber-optic networks," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 725–736, Sep. 2011.

[9] J. Ji, G. Zhang, W. Li, L. Sun, K. Wang, and M. Xu, "Performance analysis of physical-layer security in an OCDMA-based wiretap channel," *J. Opt. Commun. Netw.*, vol. 9, no. 10, pp. 813–818, 2017.

[10] J. Hizanidis, S. Deligiannidis, A. Bogris and D. Syvridis, "Enhancement of chaos encryption potential by combining all-optical and electrooptical chaos generators," *IEEE J. Quantum Electron.*, vol. 46, no. 11, pp. 1642–1649, Nov. 2010.

[11] D. Elkouss, J. Martinez-Mateo, A. Ciurana, and V. Martin, "Secure optical networks based on quantum key distribution and weakly trusted repeaters," *J. Opt. Commun. Netw*, vol. 5, no. 4, pp. 316–328, 2013.

[12] K. Guan, J. Kakande, and J. Cho, "On deploying encryption solutions to provide secure transport-as-a-service (TaaS) in core and metro networks," in *Proc. Eur. Conf. Exhib. Optical Commun.*, Sep. 18-22, 2016, pp. 1–3.

[13] In-flight Encryption in service provider networks document number: PMC-2150716, no. 2 MicroSemi, 2016, https://pmcs.com/cgi-bin/document.pl?docnum=2150716, Accessed on: 1 Dec. 2018.

[14] S. Salehi, Y. Rico Cao, and H. Chen, "Bandwidth-IPSec security trade-off in IPv4 and IPv6 in windows 7 environment," in *Proc. Int. Conf. Future Generation Commun. Technol.*, Nov. 2013, pp. 148–152.

[15] C. Xenakis, N. Laoutaris, L. Merakos, and I. Stavrakakis, "A generic characterization of the overheads imposed by IPsec and associated cryptographic algorithms," *Comput. Netw.*, vol. 50, pp. 3225–3241, 2006.

[16] L. Troell, J. Burns, K. Chapman, D. Goddard, M. Soderlund, and C. Ward, Converged vs. dedicated IPSec encryption testing in gigabit ethernet networks, Rochester Institute of Technology (2005), https://scholarworks.rit.edu/article/1743, Accessed on: 1 Dec. 2018.
[17] S. Salehi, Y. Rico Cao, and H. Chen, "Bandwidth-IPSec security trade-off in IPv4 and IPv6 in windows 7 environment," in *Proc. Int. Conf. Future Generation Commun. Technol.*, Nov. 2013, pp. 148–152.
[18] A. Pérez-Resa, M. Garcia-Bosque, C. Sanchez-Azqueta, S. Celma, "Chaotic encryption for 10-Gb ethernet optical links," *IEEE Tans. Circuits Syst.–I: Regular Papers*, 2018, doi: 10.1109/TCSI.2018.2867918.
[19] A. Klein, *Stream Ciphers*, London, United Kingdom: Springer-Verlag, 2013.
[20] M. Bellware, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatement of symmetric encryption: Analysis of the DES modes of operation," in *Proc. Symp. Found. Comput. Sci.*, 1997, pp. 394–403.
[21] M. Robshaw and O. Billet, *New Stream Cipher Designs: The eSTREAM Finalists*, Berlin, Germany: Springer, 2008.
[22] J. L. J. P. B. Daemen, "Chosen ciphertext attack on SSS," 2005. [Online]. Available: http://www.ecrypt.eu.org/stream/papersdir/044.pdf, Art. no. 235.
[23] A. Joux and F. Muller, "Chosen-ciphertext attacks against MOSQUITO," in M. Robshaw(ed.) *Fast Software Encryption*, Berlin, Germany: Springer, 2006, pp. 390–404.
[24] M. J. Dworkin, SP 800-38A 2001 edn. Recommendation for block cipher modes of operation: methods and techniques, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2001.
[25] O. Jung and C. Ruland, "Encryption with statistical self-synchronization in synchronous broadband networks," in *Proc. Conf. Cryptographic Hardware Embedded Syst.*, pp. 340–352, 1999.
[26] A. Alkassar, A. Geraldy, B. Pfitzmann, and A.-R. Sadeghi, "Optimized self-synchronizing mode of operation," in *Proc. Conf. Fast Softw. Encryption*, Apr. 2001, pp. 78–91.
[27] H. M. Heys, "Analysis of the statistical cipher feedback mode of block ciphers," *IEEE Trans. Comput.*, vol. 52, no. 1, pp. 77–92, Jan. 2003.
[28] H. M. Heys, "An analysis of the statistical self-synchronization of stream ciphers," in *Proc. IEEE INFOCOM Conf. Comput. Commun. 20th Annu. Joint Conf. IEEE Comput. Commun. Soc.*, 2001, pp. 897–904.
[29] F. Yang and H. M. Heys, "Comparison of two self-synchronizing cipher modes," in *Proc. Queen's 22nd Biennial Symp. Commun.*, 2004.
[30] H. M. Heys and L. Zhang, "Pipelined statistical cipher feedback: A new mode for high-speed self-synchronizing stream encryption," *IEEE Trans. Comput.*, vol. 60, no. 11, pp. 1581–1595, Nov. 2011.
[31] M. Bellware and P. Rogaway, "Introduction to modern cryptography," May 2005. [Online]. Available: http://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf.
[32] M. Bellware and P. Rogaway, "Chapter 5.7 security of CTR modes," in *Introduction to Modern Cryptography*, May 2005, http://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf
[33] P. Rogaway, "Evaluation of some blockcipher modes of operation," in *Technical report*, Cryptography Research and Evaluation Committees (CRYPTREC) for the Goverment of Japan, pp. 45–52, Feb. 2011.
[34] J. Daemen and V. Rijmen, *The Design of Rijndael, AES - The Advanced Encryption Standard*, Berlin Germany and New York: Springer-Verlag, 2002.
[35] M. J. Dworkin, SP 800-38G recommendation for block cipher modes of operation: methods for format-preserving encryption, National Institute of Standards & Technology, Gaithersburg, MD, United States, 2016.
[36] R. Agbeyibor, J. Butts, M. Grimaila and R. Mills, "Evaluation of format preserving encryption algorithms for critical infrastructure protection," in *Critical Infrastructure Protection VIII*, Berlin, Geramny: Springer, 2014, pp. 245–261.
[37] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma, "Physical layer encryption for industrial ethernet in Gigabit optical links," *IEEE Trans. Ind. Electron.*, vol. 66, no. 4, pp. 3287–3295, Jun. 2018.
[38] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, S. Leigh, M. Levenson, M. Vangel, D. Banks, and A. Heckert, SP 800-22 Rev.1a, A Statistical Test Suite for Random and Pseudorandom Number Generators for Cryptographic Applications: National Institute of Standards & Technology, Gaithersburg, MD, United States, 2010.
[39] Y. E. A. Mao, "Integrating quantum key distribution with classical communications in backbone fiber network," *Opt. Express*, vol. 26, no. 5, pp. 6010–6020, 2018.
[40] A. E. A. Slavisa, "Perspectives and limitations of QKD integration in metropolitan area networks," *Opt. Express*, vol. 23, no. 8, pp. 10359–10373, 2015.

**Adrián Pérez-Resa** received the MSc degree in telecommunications engineering from the University of Zaragoza, Zaragoza, Spain, in 2005. Currently he is working toward the PhD degree from the Group of Electronic Design, Aragón Institute of Engineering Research, the University of Zaragoza. He was an R&D engineer with the Telecommunications Industry for more than ten years. He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include high speed communications and cryptography applications.

**Miguel Garcia-Bosque** received the BSc and MSc deghrees in physics from the University of Zaragoza, Zaragoza, Spain in 2014 and 2015 respectively. He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include chaos theory and cryptography algorithms.

**Carlos Sánchez-Azqueta** received the BSc, MSc, and PhD degrees from the University of Zaragoza, Zaragoza, Spain, in 2006, 2010, and 2012, respectively, all in Physics, and the Dipl.-Ing. Degree in electronic engineering from the Complutense University of Madrid, Madrid, Spain, in 2009. He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include mixed-signal integrated circuits, high-frequency analog communications, and cryptography applications.

**Santiago Celma** received the BSc, MSc, and PhD degrees in physics from the University of Zaragoza, Zaragoza, Spain, in 1987, 1989, and 1993, respectively. He is currently a full professor in the Group of Electronic Design, Aragon Institute of Engineering Research, University of Zaragoza. He has coauthored more than 100 technical papers and 300 international conference contributions. He is coauthor of four technical books and the holder of four patents. He appears as principal investigator in more than 30 national and international research projects. His research interests include circuit theory, mixed-signal integrated circuits, high-frequency communication circuits, wireless sensor networks and cryptography for secure communications.

# Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems

Adrián Pérez-Resa, Miguel Garcia-Bosque, Carlos Sánchez-Azqueta, and Santiago Celma

***Abstract*— In the past decades, Ethernet has become an alternative technology for the field buses traditionally used in industrial control systems and distributed measurement systems. Among different transmission media in Ethernet standards, optical fiber provides the best bandwidth, excellent immunity to electromagnetic interference, and less signal loses than other wired media. Due to the absence of a standard that provides security at the physical layer of optical Ethernet links, the main motivation of this paper is to propose and implement the necessary modifications to introduce encryption in Ethernet 1000Base-X standard. This has consisted of symmetric streaming encryption of the 8b10b symbols flow at physical coding sublayer level, thanks to a keystream generator based on chaotic algorithm. The overall system has been implemented and tested in an field programmable gate array and Ethernet traffic has been encrypted and transmitted over an optical link. The experimental results show that it is possible to cipher traffic at this level and hide the complete Ethernet traffic pattern from passive eavesdroppers. In addition, no space overhead is introduced in data frames during encryption, achieving the maximum throughput.**

***Index Terms*— 1000Base-X, chaos, cryptography, Ethernet, field-programmable gate array (FPGA), stream cipher.**

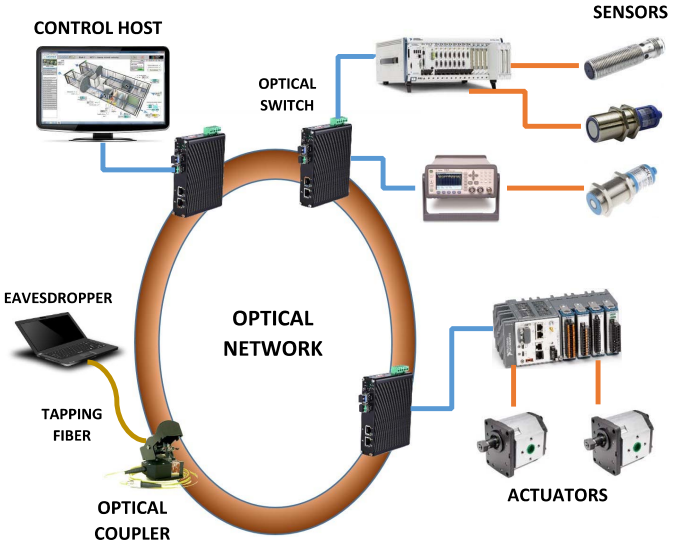



Fig. 1. Simple example of a distributed control and data acquisition network and an eavesdropper attack with a tapping fiber.

## I. Introduction

In RECENT decades, Ethernet has expanded widely in industrial control systems and critical infrastructures, replacing the traditional communication field buses [1], [2]. Ethernet has proved to be an efficient technology to create distributed acquisition systems such as supervisory control and data acquisition networks. At present, this kind of networks not only supports measurement and control systems but also data traffic for surveillance video security and future Internet of Things (IoT) applications. Both, distributed measurement platforms with integrated Ethernet ports and industrial network equipment up to 1-Gb/s rates and beyond, are available from many vendors. A simple scheme of a distributed control and data acquisition network is shown in Fig. 1.

Manuscript received October 2, 2018; revised January 18, 2019; accepted January 21, 2019. Date of publication February 25, 2019; date of current version November 8, 2019. The work of M. Garcia-Bosque was supported in part by MINECO-FEDER under Grant TEC2014-52840-R and Grant TEC2017-85867-Rand in part by FPU Fellowship under Grant FPU14/03523. The Associate Editor coordinating the review process was Huang-Chen Lee. *(Corresponding author: Adrián Pérez-Resa.)*

The authors are with the Electronic and Communications Engineering Department, Zaragoza University, 50009 Zaragoza, Spain (e-mail: aprz@unizar.es; mgbosque@unizar.es; csanaz@unizar.es; scelma@unizar.es).

Color versions of one or more of the figures in this article are available online at http://ieeexplore.ieee.org.

Digital Object Identifier 10.1109/TIM.2019.2896550

Optical Ethernet is widely used in industrial environments. Optical fiber has some advantages over other wired methods, such as higher bandwidth, less signal losses, and more immunity to electromagnetic interference. In addition, as it does not emit any radiation, it is safer than wireless systems that are more exposed to eavesdropping.

However, vulnerability and threat analysis in the physical layer of optical systems is critical to guarantee secure communications [3], [4]. One of the most important attacks is the splitting attack. It is normally used for either eavesdropping or signal degradation and can be easily performed with tapping techniques. At present, low-cost methods for intercepting the optical signal through fiber coupling devices and electrooptical converters are available without the need to perceptibly interfere in communications [5]. To avoid or detect eavesdropping, encryption and intrusion detection systems have been proposed as solutions [3].

In a layered communication model, encryption methods can be implemented at different communication levels. It depends on the communication layer where confidentiality is needed. In the particular case of industrial Ethernet, solutions are usually proposed for network and transport layers (layers 3 and 4), such as IPsec or transport layer security protocols [6], [7]. Other solutions are proposed for the data link layer (layer 2), such as MACsec standard [8].

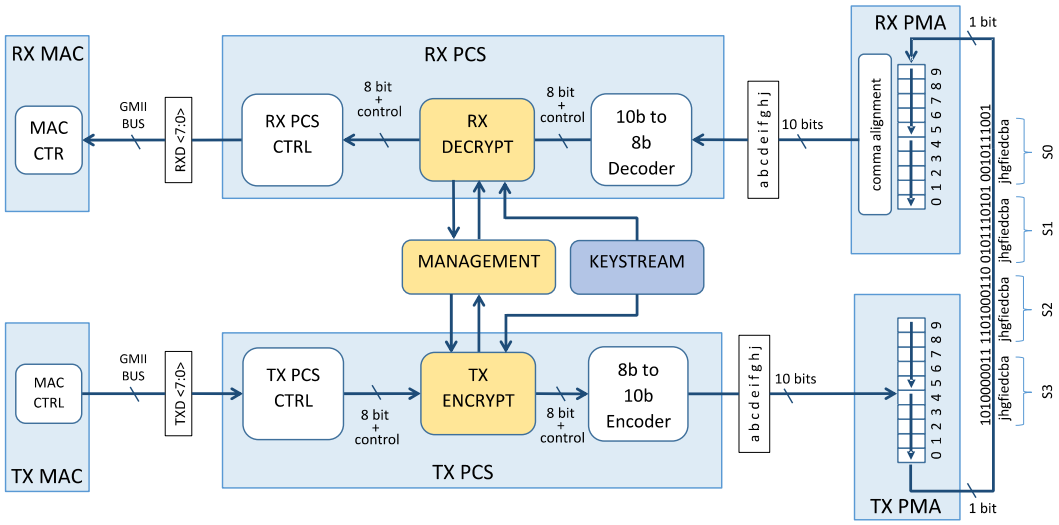



Fig. 2. PCS structure with the proposed encryption function PHYsec included.

Regarding physical layer encryption in optical communications, there are some solutions for protocols such as optical transport network [9]. However, they are only oriented to telecommunication networks. One advantage of encryption at the physical layer is that it can be performed at a line-rate introducing a very low latency. In the particular case of industrial Ethernet networks, high level of determinism is usually needed, and for certain applications, there can also be strict requirements for network delay and jitter [10]. A physical layer encryption mechanism could be useful for encrypting traffic of real-time protocols such as EtherCAT or PROFINET IRT [10], [11], enabling encryption to be performed at line rate.

As 1000Base-X standard is one of the most widely used physical layer standards at 1 Gb/s, the main motivation of this paper is to propose and implement an encryption method suitable for the physical coding sublayer (PCS) of this standard. As the main coding in 1000Base-X is the well-known 8b10b encoding, a stream cipher that preserves the format of this codification is needed.

In general, two approaches are usually used to build stream ciphers: *ad hoc* stream ciphers, as those proposed in [12], and secure blockciphers, such as advanced encryption standard (AES), working in a specific operation mode, e.g., counter mode (CTR). These two approaches are focused on plaintext in binary format, which are not suitable for our purposes. Among different *ad hoc* stream ciphers solutions, there are those based on chaotic algorithms. The characteristics of these are very similar to those that a good cryptographic algorithm should have. They have been analyzed from several points of view [13], [14] and have given rise to new encryption proposals related with several areas, such as IoT applications [15], communications [16], or image encryption [17], [18]. On the other hand, chaotic encryption has been applied with different technologies, using analog and digital electronic circuits [19], [20] or optical and electrooptical mechanisms [21], [22].

To encrypt data preserving its format with a stream cipher, a solution was initially proposed in [23], where the keystream generator was based on a format preserving encryption (FPE) blockcipher working in CTR mode.

This paper is an extension of [24], where a new alternative to [23] is proposed. In this paper, this encryption method has been named PHYsec and the keystream generator is now based on a chaotic stream cipher.

An electronic implementation for the chaotic algorithm has been carried out in a digital programmable platform as in [23]. Thanks to the proposed structure, it is possible to save hardware resources while reaching the same encryption throughput. Moreover, the power consumption is reduced by more than 50%.

In addition, this paper deals with another important issue relative to the encryption system such as the keystream synchronization, a topic that is not addressed in [23].

This paper is organized as follows. Section II explains the overall structure of the encryption mechanism in the PCS sublayer. Sections III and IV deal with the encryption operation for PHYsec and the synchronization mechanism between TX (transmitter) and RX (receiver), respectively. Section V explains the keystream generation and some considerations about the key space. Section VI details the system implementation and test results. Finally, Section VII summarizes the conclusions obtained in this paper.

## II. PCS Layer Encryption

The physical layer or PHY is responsible for carrying out the lower level functions in the transmission. It defines the electrical, mechanical, and functional specifications to activate, maintain, and deactivate the physical link. In the case of Ethernet standards, it is usually divided into three other sublayers with different functionalities: PCS, physical medium attachment (PMA), and physical medium dependent. PCS layer performs functions such as autonegotiation, link establishment, 8b10b data encoding, symbol synchronization, clock rate adaptation, and so on.

The basic structure of the PCS layer and its interface with the medium access control (MAC) and the PMA levels are shown in Fig. 2. The coding function is separated into encoder and decoder blocks while the rest of the functions of the PCS layer would reside in the RX_PCS_CTRL and TX_PCS_CTRL modules.

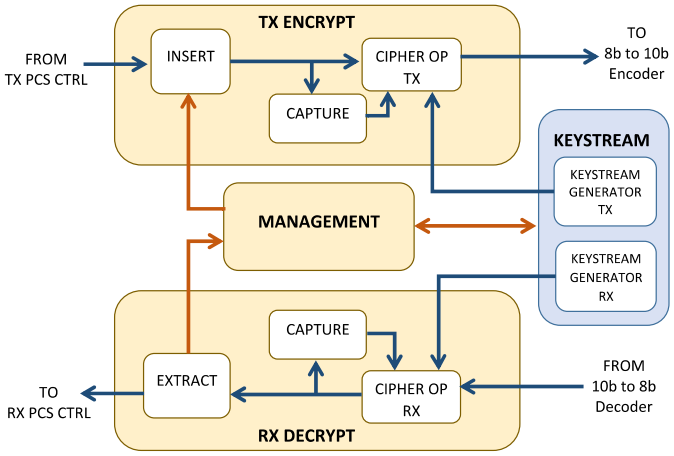



Fig. 3. Encryption infrastructure for PHYsec function.

Unlike other encryption mechanisms such as IPsec or MACsec, where the data packets are encrypted individually, in this paper, the 8b10b symbol flow is directly and continuously encrypted. One advantage of this method is that no extra data field is added to packets, which would reduce the overall data throughput. In addition, by encrypting the complete 8b10b symbol flow, it also achieves to mask the activity or data traffic pattern. This is because by encrypting at 8b10b symbol level, control symbols and ordered sets are also encrypted, such as packet start and end symbols or idle sets in the interframe gap (IFG). Some important properties of the 8b10b encoder are the transition density and the short run length introduced in the encoded serial data. These are necessary to facilitate the operation of the remote clock and data recovery (CDR) circuit. To preserve these properties, it is proposed to perform the encryption operations at the input of the encoder and decryption at the output of the decoder, as shown in Fig. 2.

One advantage of performing the encryption preserving the coding properties and maintaining the rest of physical layer features untouched is that the physical layer encryption maintains compatibility with the subsequent hardware elements or medium-dependent circuitry. For example, commercial optical modules as small form-factor pluggable (SFP) and electronic circuits as CDR or serializer/deserializer (SERDES) for 1000Base-X standard would be also compatible with the proposed encryption method.

In the proposed system, encryption is carried out by a symmetrical stream cipher, as it fits well with a continuous data flow such as 8b10b symbol transmission. We divide the system into three functional parts. The first consists of the encryption and decryption blocks, which includes the mathematical stream cipher operation. The second is the synchronization mechanism between TX and RX. Finally, the third is the keystream generator used in the cipher function. These parts are explained in Sections III, IV, and V, respectively.

## III. Encryption and Decryption Operations

The stream cipher operation is performed in the CIPHER_OP_TX and CIPHER_OP_RX modules in Fig. 3. The data set to be encrypted is a limited set, since the valid 8b10b symbols are composed of 256 data symbols plus 12 control symbols, a total of 268 symbols which do not generate code errors.

Nevertheless, one of the 12 control symbols (/K28.7/) is not used for standard data communication [25], and to avoid its accidental generation in the encryption of any 8b10b symbol, it has been excluded from the encryption symbol mapping, explained in the following. Therefore, the number of valid symbols is 267.

As the encryption is performed before the encoding and it is necessary to preserve the coding properties, the encrypted data must be valid 8b10b symbols inside the 267 possible symbols. To achieve this goal, the symbols are mapped to an integer value in the range of 0–266. After the mapping, stream cipher operation is performed. This operation consists of a modulo-267 addition between the mapped symbols and the keystream, which also takes values uniformly distributed between 0 and 266. Once the cipher operation is done, the resulting values are reverse-mapped to the corresponding new 8b10b symbol.

The decryption process is the same, only that the decryption operation is a modulo-267 subtraction. In Fig. 4, the structure of the module CIPHER_OP_TX and CIPHER_OP_RX is shown.

## IV. Encryption Synchronization

### A. Management Module

MANAGEMENT module in Fig. 3 is the part of the system that configures, controls, and reports the encryption status between both PHYs. It implements the initial synchronization procedure and collects the alarms relative to the synchronization status, by which the user can perform the necessary actions for achieving a coherent communication between receiver and transmitter.

For example, in case of a mismatch between encryption status (one PHY encrypting and the other not), or a bad synchronization status (misaligned keystream generators between remote PHYs), several alarms can be triggered and latched in the MANAGEMENT module, which reports them to the user, thanks to the field-programmable gate array (FPGA) debug system. The system is able to detect the loss of synchronization 2.136 $\mu$s after the misalignment is produced. After this notification, the user is able to stop keystream generators and restart encryption synchronization procedure in a way that a new coherent encryption status between the two PHYs is achieved. In the long term, under normal working conditions, we could consider that the probability of a synchronization loss is negligible, as the implemented system uses commercial components compatible with the standard. For example, SFP modules, optical fiber, or the transceiver circuitry. The standard guarantees a very low bit error rate, under $10^{-12}$, and the ability of clock recovering with a clock tolerance of $\pm 100$ ppm.

In order to maintain the concordance with the standard, communication between the MANAGEMENT modules of remote PHYs has been planned to be based on control messages formatted like 1000Base-X ordered sets. The insertion/extraction of the new and future control messages in the 8b10b data flow is performed by MANAGEMENT module,

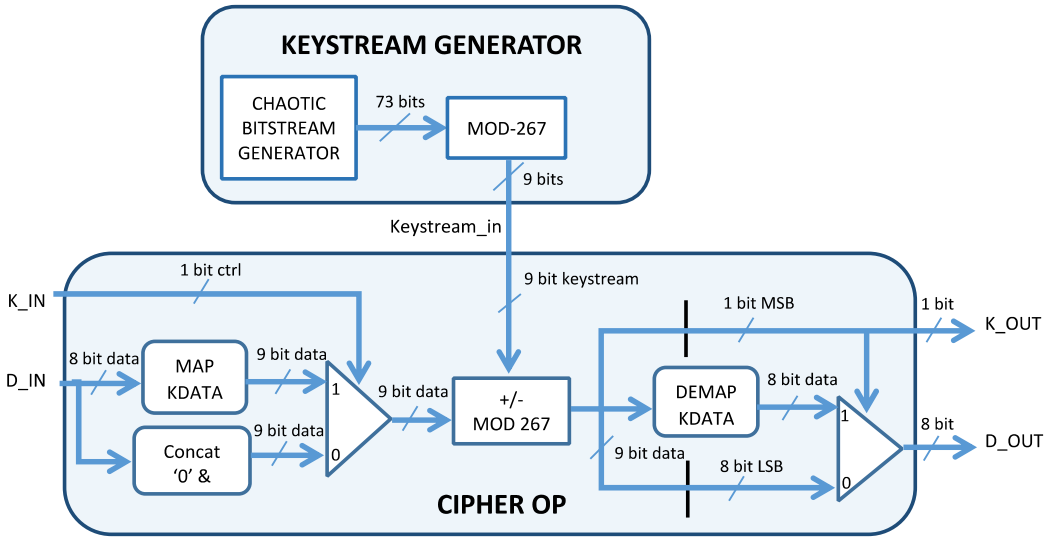



Fig. 4. Stream cipher operation performed in CIPHER_OP module next to one of the keystream generators (RX or TX).

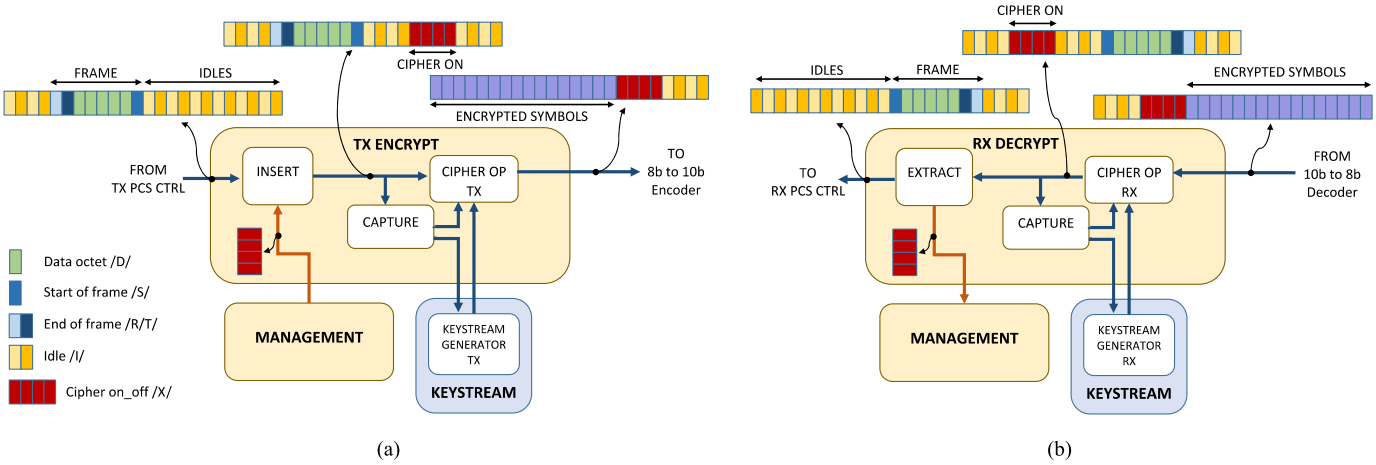



Fig. 5. Initial synchronization procedure, based on the proposed /X/ ordered set. (a) Transmission. (b) Reception.

thanks to INSERT and EXTRACT blocks in Fig. 3, and it is explained in Section IV-B.

### *B. Initial Synchronization Procedure*

To achieve encryption synchronization with a stream cipher, it is necessary that the keystream sequence generated at RX side is exactly aligned with the incoming keystream sequence embedded into the ciphertext and generated at TX side. The initial synchronization procedure is in charge of this mechanism. For that purpose, MANAGEMENT module in Fig. 3 performs the insertion and extraction of a control sequence into the 8b10b symbol stream to indicate the remote PHY when to start/stop its decryption process. This sequence is used for both the activation and deactivation of the encryption and decryption.

The format of the implemented control sequence is similar to an ordered set one. In Table I, this sequence is called /X/ and it is shown along with those that already exist in the standard. The first character in the new set is /K28.1/ and as /K28.5/, it includes the comma sequence.

The basic operation of this procedure is described next and is shown in Fig. 5(a) and (b). In Fig. 5(a), the CIPHER_OP_TX module, responsible for performing the encryption operation, is initially disabled, allowing the 8b10b symbols to be transparently passed from the TX_PCS_CTRL to the 8b10b encoder. In order to start encrypting the 8b10b symbol stream, the MANAGEMENT module acts on the INSERT block to insert a /X/ encryption start message in the 8b10b symbol flow. As shown in Fig. 5(a), this message is represented as a set of four symbols and it is inserted in the line by replacing idle ordered sets with the octets that form the message itself. Before passing through the CIPHER_OP_TX module, /X/ is detected by the CAPTURE module. Then the CAPTURE module enables CIPHER_OP_TX and the keystream generator (KEYSTREAM_GENERATOR TX), which initiates the encryption process after /X/ has been transmitted. As shown in Fig. 5(a), at the output of the CIPHER_OP_TX module, every 8b10b symbol after /X/ is encrypted, including idle sets and complete Ethernet frames.

TABLE I
IEEE 802.3 ORDERED SETS [25] AND PROPOSED ORDERED SET /X/

| Code | Ordered_Set | Number of Code-Groups | Encoding |
|---|---|---|---|
| /C/ | CONFIGURATION | | Alternating /C1/ and /C2/ |
| /C1/ | Configuration 1 | 4 | /K28.5/D21.5/Config_Reg |
| /C2/ | Configuration 2 | 4 | /K28.5/D2.2/Config_Reg |
| /I/ | IDLE | | Correcting /I1/, Preserving /I2/ |
| /I1/ | Idle 1 | 2 | /K28.5/D5.6/ |
| /I2/ | Idle 2 | 2 | /K28.5/D16.2/ |
| | ENCAPSULATION | | |
| /R/ | Carrier_extend | 1 | /K23.7/ |
| /S/ | Start_of_Packet | 1 | /K27.7/ |
| /T/ | End_of_Packet | 1 | /K29.7/ |
| /V/ | Error_Propagation | 1 | /K30.7/ |
| | **ENCRYPTION** | | **Enable/Disable** |
| **/X/** | **Cipher_on_off** | **4** | **/K28.1/D21.5/D21.2/D21.2/** |

At the receiver, as shown in Fig. 5(b), the module CIPHER_OP_RX is in charge of performing the decryption operation. As in the transmitter, it is initially inactive, enabling 8b10b symbols to be transparently passed from the 8b10b decoder to the controller RX_PCS_CTRL. When the CAPTURE module detects /X/ in reception, decipher module (CIPHER_OP_RX) and the keystream generator (KEYSTREAM_GENERATOR RX) are enabled, starting to decrypt the 8b10b stream after the /X/ set. Subsequently, the control message /X/ is extracted from the data stream in the EXTRACT module, which replaces it with idle ordered sets in the 8b10b symbol flow (reverse operation of the INSERT module).

Thanks to this procedure, the beginning of the ciphertext is detected at the receiver allowing to generate a keystream sequence that is aligned with the incoming data, then performing decryption properly.

The way to insert set /X/ is shown in Fig. 6. INSERT module contains a buffer where the MANAGEMENT module can write messages while there is enough space. These messages can only be read from the buffer when the INSERT_MACHINE block in Fig. 6 indicates that it is possible to do so.

The INSERT_MACHINE block monitors a pipeline where the 8b10b symbol stream passes through. Initially, the output of the INSERT module is selected as the output of the pipeline. When a number of idle ordered sets, equivalent to the size of the next message to be read, is detected inside the pipeline, the output of INSERT module is switched to the data coming from the message buffer. Then the pending message is read and the pipeline is emptied from the stored idles. Once the message has been transmitted, the output of INSERT module switches to the pipeline output again.

## V. KEYSTREAM GENERATOR

The keystream generator is responsible for calculating pseudorandom numbers to carry out the encryption of the signal. As the encryption operation consists of a modulo-267 addition, the values of the pseudorandom sequence must be between 0 and 266. In order to build that sequence two

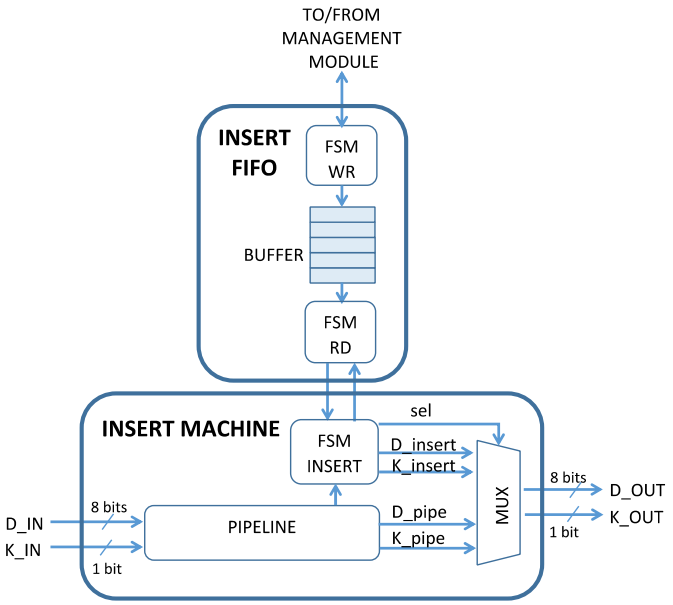



Fig. 6. INSERT module. Several finite-state machines (FSMs) govern this module. FSM_WR and FSM_RD control the write and read of message buffer and FMS_INSERT the message insertion on the line.

steps are carried out in each keystream generator (TX and RX), first, a pseudorandom bitstream is generated based on a chaotic algorithm, which is considered the core of this stream cipher. Second, modulo-267 operation is applied to the output of the chaotic algorithm. The result is a pseudorandom sequence composed of 9-bit words with values between 0 and 266. The keystream generator structure is shown in Fig. 4, where the CHAOTIC_BITSTREAM_GENERATOR generates the pseudorandom bitstream and MOD-267 block carries out the modulo operation.

### A. Modulo Operation

The modulo operation must be applied to words with 9 bit width or more to result in a 9-bit-width output. When doing such operation, a bias is introduced in the distribution of the resulting number sequence. According to the National Institute of Standards and Technology (NIST) recommendation [26], in order to make this bias negligible, the input width of the modulo operation shall be at least 64 bits longer than the output. As in our case 9-bit numbers are necessary, the pseudorandom bitstream generator output and modulo operator input have to be at least 73 bits width.

The implementation of modulo-267 operation has been based on [27], which presents a high-speed hardware implementation for a generic operation "$x$ mod $z$" that can be fragmented into a pipeline of $n - m + 1$ stages, where $x$ is represented by $n$ bits and $z$ by $m$ bits. In our case, $z$ has been taken as a constant value equal to 267. According to this, the resulting hardware structure should have 65 stages as shown in Fig. 7. However, by implementing an overclocked structure, this number can be reduced to 33.

### B. Pseudorandom Bitstream Generator

The study of chaos and the systems derived from it have aroused interest as possible design solutions of new

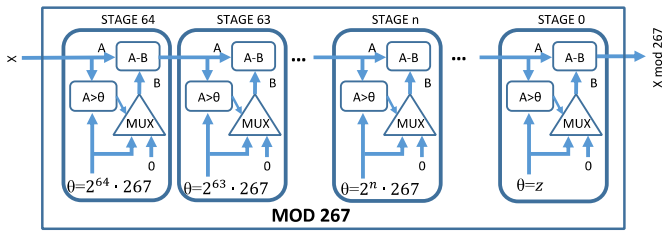



Fig. 7. Modulo 267 hardware.

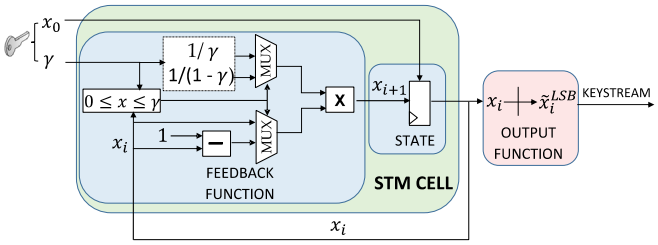



Fig. 8. STM block diagram.

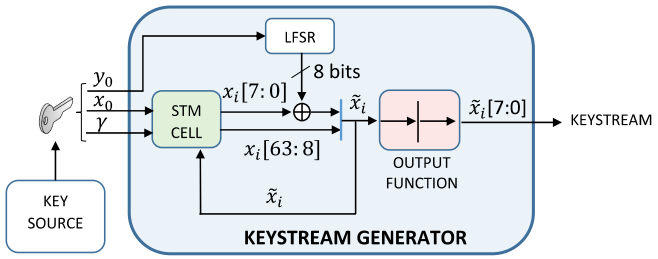



Fig. 9. Basic keystream generator.

cryptographic systems. From a theoretical point of view, a good cryptographic algorithm should have properties such as confusion, diffusion, and sensitivity to changes in plaintext and secret key. Apart from that, the most well-known characteristics of chaotic systems are the high sensitivity to the initial conditions or control parameters (also called "butterfly effect") and the unpredictable pseudorandom orbits generated by their algorithms.

Among chaos-based cryptosystems, there are two main approaches for their design. On the one hand, analog chaotic cryptosystems based on chaos synchronization, where two chaotic systems can synchronize under the driving of scalar signals sent from one system to another. These kinds of cryptosystems are usually implemented in the analog domain and need very accurate components to ensure information recovery.

On the other hand, a second approach is the discrete chaos-based cryptosystems, where one or more chaotic maps are implemented using finite precision and they do not depend on chaos synchronization at all. Many hardware implementations have been proposed for a wide variety of chaotic maps such as in [28], [29], or [30] where the well-known modified logistic map, Hennon map, and Bernoulli map, respectively, were built. Although there are analog solutions that have been digitized, as in [20], the discrete chaos-based approach is more suitable for digital hardware platforms as FPGAs.

In this paper, for generating the pseudorandom bitstream, a variant of the chaotic map called skew tent map (STM) has been used. This chaotic map has already been extensively studied in [31] and [32], based on the previous work by our group.

The equations of this map are shown in the following equation:

$$f(x_i) = x_{i+1} = \begin{cases} x_i/\gamma, & x_i \in [0, \gamma\,] \\ (1 - x_i)/(1 - \gamma), & x_i \in (\gamma, 1] \end{cases} \quad (1)$$

where its control parameter and initial state are $\gamma$ and $x_0$, respectively, and their values are included in the interval (0, 1). One important characteristic of STM is that it is a piecewise linear map and has positive Lyapunov exponent for any selected value of $\gamma$ and $x_0$. This fact can satisfy always the chaoticity of this map [13] and makes it more adequate than others for cryptographic applications. In Fig. 8, the hardware block diagram of the basic STM algorithm is shown.

In this paper, a fixed-point implementation of the STM algorithm has been carried out. A typical problem of this kind of implementations is that due to the finite word length, degradation is produced in the dynamical properties of the chaotic map. When the STM is implemented with a presicion of $n$ bits, there are only $2^n$ possible values for each $x_i$, giving a maximum period for the keystream of $2^n$ possible samples. In spite of this, cycle lengths are found empirically to be much shorter. Owing to this fact, usually some randomness test applied to the chaotic sequences fail. One recommended solution for increasing this period is the use of a pseudorandom number generator to perturb the chaotic orbits of this kind of algorithms [33].

For this purpose, the structure in Fig. 8 has been improved by using a linear feedback shift register (LFSR), obtaining a significant increase in the keystream period and, therefore, better properties as a pseudorandom sequence [32], [34]. According to [32], by selecting an LFSR with 61 steps, it is possible to guarantee an enough length for the chaotic sequence that makes it pass the randomness tests.

Fig. 9 shows the structure of the basic keystream generator used in this paper. A 61-step LFSR has been added to the STM module. The cryptographic key used to configure this basic generator will consist of the initial state and control parameter ($\gamma$ , $x_0$) of STM cell and the initial state of the LFSR ($y_0$).

The STM cell has been implemented with an internal 64-bit state whose output is the vector $x_i[63 : 0]$. The improvement related to LFSR is to perform the XOR operation between the least significant 8 bits of LFSR output and the STM cell before applying the generator output function. The state returned to the STM cell will no longer be the previous state calculated by it, but a new state $\tilde{x}_i$ to which a small noise generated by the LFSR has been added.

The generator output function in Fig. 9 is to take the 8 least significant bits of $\tilde{x}_i$. Since the output bus of the keystream generator must be 73 bits width, a bank of basic keystream generators has been built. The bank consists of eight 8-bit-width generators and one 9-bit-width generator; their outputs are concatenated to give a 73-bit output. The final structure for the keystream generator is shown in Fig. 10.

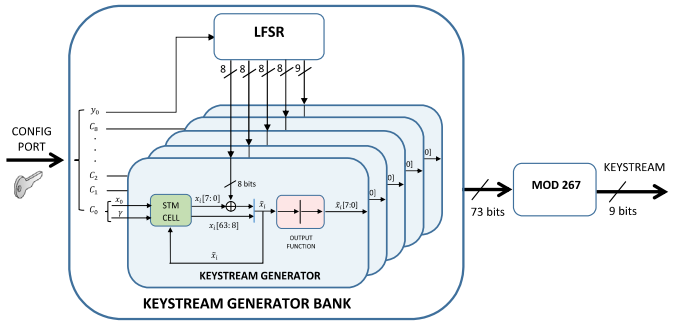



Fig. 10. Keystream generator.

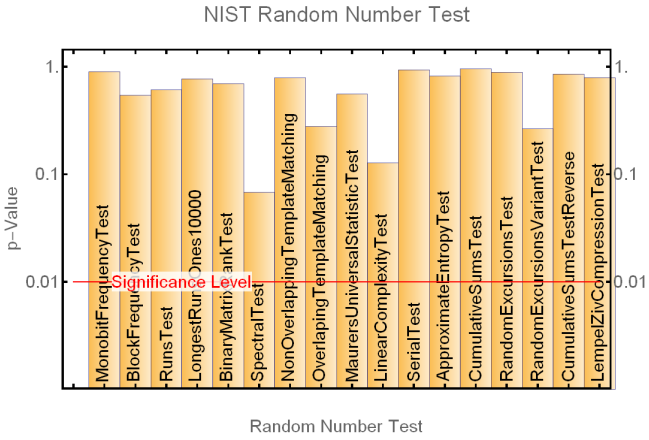



Fig. 11. NIST test results for a bitstream generated using STM-LFSR algorithm.

### C. Keystream Output Analysis

One of the main requirements of a secure stream cipher is that its keystream should be undistinguishable from a truly random sequence. On the one hand, it is necessary to test the randomness of the chaotic bitstream, and on the other hand, it is necessary to check that the bias introduced by module 267 operator is negligible.

In order to test the randomness of chaotic bitstream, sequences generated by the basic generator in Fig. 9 were subjected to the NIST SP 800-22 battery of tests [35]. The generator passed these tests, and an example of the results for a particular sequence is shown in Fig. 11.

In order to check the uniformity between 0 and 266 of the resulting keystream after modulo operation, histograms for different sequences were obtained. In Fig. 12, two cases are shown for a sequence of 1 310 720 values, one of them [Fig. 12(b)] is the histogram obtained when NIST recommendation [26] is applied and the input to module 267 operator is 73 bit width. The other case [Fig. 12(a)] is when only a 9-bit-width input is used in the module operation. Clearly, a bias appears in the second case, but it seems negligible in the first.

Moreover, chi-square goodness of fit test has been used to determine if the resulting keystream obtained when applying [26] comes from a uniform distribution between 0 and 266. The test has not rejected this hypothesis at 5% of significance level. However, this kind of tests for uniform distributions does not guarantee the randomness for the final keystream.

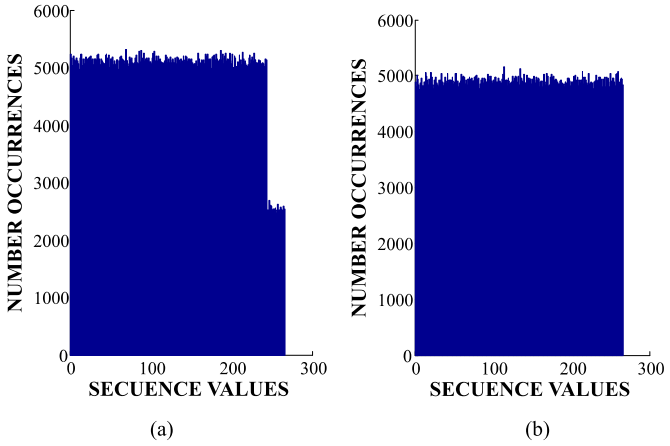



Fig. 12. Histogram of output keystream (a) Without applying NIST recommendation and (b) Applying it.

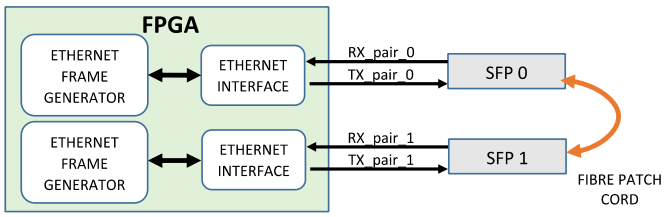



Fig. 13. Test setup scheme.

The NIST battery of test is suitable for binary streams of data; however, it is not directly applicable to nonbinary data sets when they are not power of two. As far as we are concerned, there is not a standardized set of test for checking randomness of nonbinary sources.

In this paper, some tests suitable for nonbinary sources described in [36] have been applied. Particularly, frequency test, serial test, and poker test were used successfully. As for frequency and serial test, the chi-square goodness of fit test was successfully passed to sequences of 3 and 15 millions of tuples, respectively. Regarding the poker test, also the chi-square goodness of fit test was carried out by using the five categories described by Knuth [36].

### D. Key Space

In the proposed system, the key space size of a single cell would be given by all the possible values of $\gamma$ and $x_0$, with 64 bits each one, and the initial 61-bit-length LFSR state, $y_0$. This gives a total key space size of $2^{64+64+61} = 2^{189}$. Some guidelines recommend that in order to be secure for the coming years, the key space size should be greater than $2^{112}$ or $2^{128}$ [37], [38]. Therefore, the key space size of this algorithm is big enough to prevent both key reusing as well as brute-force attacks.

## VI. System Implementation

### A. System Description

The complete system has been implemented in a Xilinx Virtex 7 FPGA. In the setup for test, the FPGA has been connected to two SFP modules capable of transmitting at a rate of 1.25 Gb/s at 850 nm over multimode fiber.

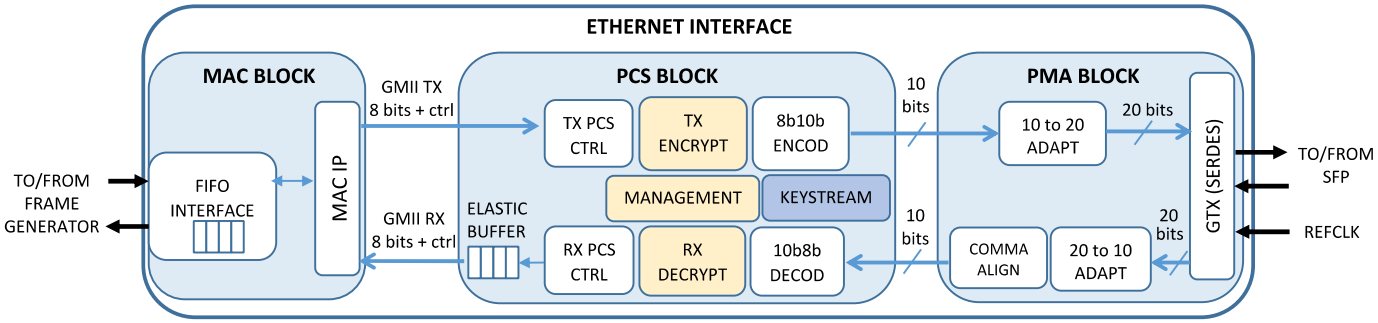



Fig. 14. Scheme of FPGA implementation for Ethernet interface with the encryption function. TX and RX PCS datapaths work with different clock domains. In TX datapath, the system clock is used. However, for the RX datapath, is necessary to use the recovered clock from PMA block. This clock is used on the RX side of the PCS block until the elastic buffer. Elastic buffer is used for rate adaptation between MAC and RX interface of PCS block.

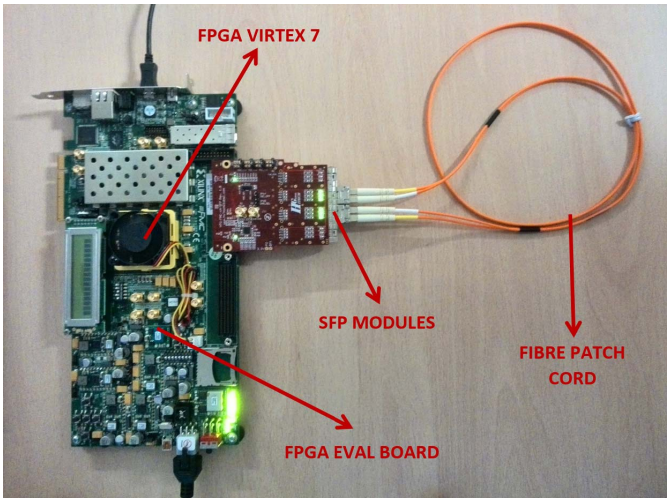



Fig. 15. Test setup photograph.

The FPGA design consists of two Ethernet interfaces with PHYsec function and two Ethernet frame generator modules connected to them. Each Ethernet interface contains the MAC module and the PHY (PCS and PMA blocks) including the encryption system explained along this paper. The scheme of the overall hardware system and the structure of the Ethernet interfaces are shown in Figs. 13 and 14, respectively. In Fig. 15, a photograph of the test setup is also shown.

The PHY is connected directly to the SFP modules, thanks to the FPGA SERDES circuit. In the MAC side, Ethernet interface is connected to the Ethernet frame generator to test the encrypted link with real traffic and verify that no frames are lost and no cyclic redundancy check (CRC) errors are produced during encryption.

It is important to remark that the initial PCS structure of this paper (without the encryption mechanism) parts from an implementation compatible with the standard. The PHYsec functionality has been developed and incorporated to this initial PCS sublayer.

Moreover, the final PCS structure, including PHYsec functionality, introduces an extra latency in the 8b10b TX datapath of 192 ns, in respect the baseline implementation, and approximately the same in the RX datapath. This extra latency is due to the new hardware modules added to PCS sublayer and shown in Fig. 3. For example, in the TX direction, 8b10b symbols have to traverse INSERT and CIPHER_OP_TX modules before being encoded. In the INSERT module, detailed in Fig. 6, the pipeline introduces a latency of 18 clock cycles, while the CIPHER_OP_TX operations (mapping, modulo-267 addition, and reverse-mapping) delays only take six clock cycles. The total latency introduced in TX datapath is 24 cycles at a clock rate of 125 MHz (8 ns of period), which is the operation frequency at which the system works.

### *B. Encryption Results*

The conclusions drawn from simulation and hardware debugging can be summarized in the following points.

1) Encryption/decryption works correctly and synchronously without harming data traffic or link establishment between Ethernet interfaces. Maximum data rate is achieved without frame losses or CRC errors. It has been checked, thanks to frame and CRC counters inside Ethernet frame generators. Traffic bursts were tested with a duration of $10^6$ frames with length of 1500 bytes. The throughput of the traffic was configured between 10% and 98% of the maximum line rate.
2) Encryption allows making the transmitted frames indecipherable. When encryption keys are different between transmitter and receiver, no valid frames are received in RX interface as it is impossible to detect the right8b10b symbol flow.
3) Encryption makes indistinguishable a data traffic pattern from a continuous idle transmission, and then it is able to hide the pattern of Ethernet traffic from passive eavesdroppers.

For this last capability, it is interesting to monitor the signal waveforms at the input of the encoder and output of the decoder. Particularly, K control flag can give information about the transmission state. This flag is generated next to each 8b10b symbol by the PCS_TX_CTRL controller in the PCS sublayer. Both K control flag and 8b10b symbols are the input to the encoder and the output of the decoder. Each 8b10b symbol is a control or data one depending whether its K flag is "1" or "0," respectively.

If the encryption is not enabled, when transmitting no frames, K control flag pattern is a signal that switches continuously between "0" and "1." It is because when no frame is transmitted, idle ordered sets are continuously sent, and they

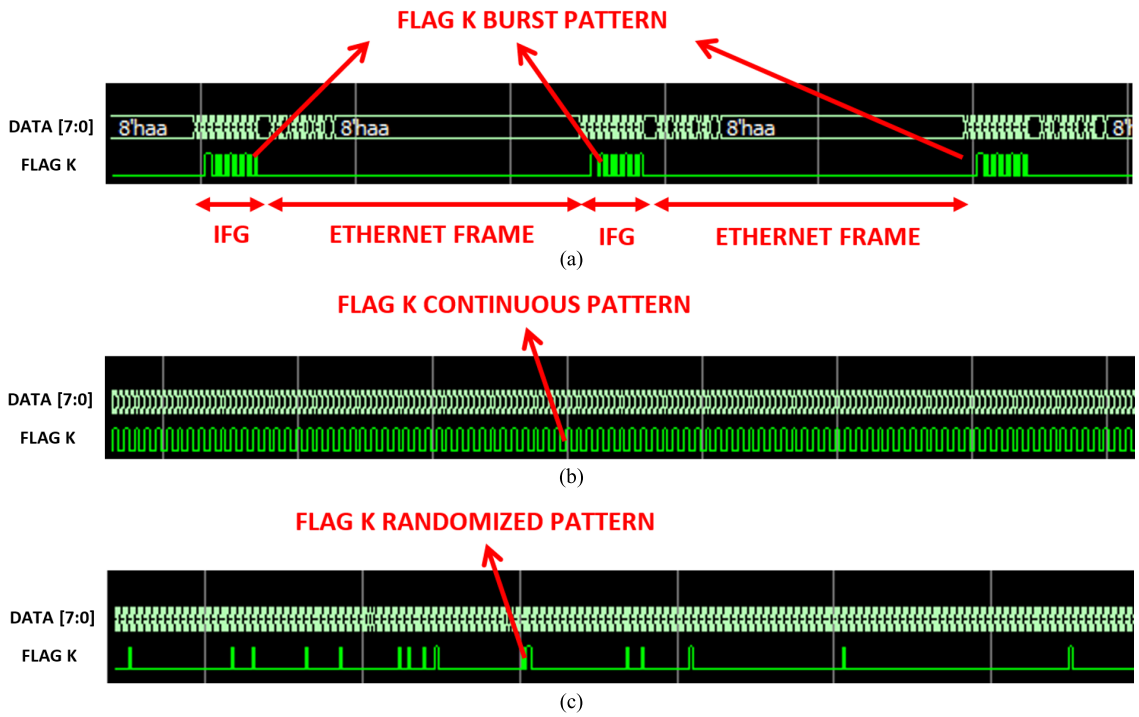



Fig. 16. (a) K flag pattern before encryption when transmitting an Ethernet frame burst. (b) K flag pattern before encryption when no Ethernet frame is transmitted. (c) K flag pattern after encyption regardless of the transmission or nontranmission of Ethernet frames.

TABLE II
FPGA RESOURCES USED IN KEYSTREAM GENERATOR SUBMODULES

| MODULE | Slice LUTs | Slice Registers | MULT cells |
|---|---|---|---|
| KEYSTREAM_GEN[1] | 10943 | 3629 | 144 |
| LFSR | 210 | 195 | 0 |
| STM_BANK | 6606 | 589 | 144 |
| MOD-267 | 4127 | 2845 | 0 |

[1]KEYSTREAM_GEN refers to the hardware resources of the Keystream Generator, including LFSR, STM_BANK and MOD-267 operation.

TABLE III
COMPARISON WITH OTHER SOLUTIONS

| | FPE [23] | This Work |
|---|---|---|
| Slice Registers | 11127 | 3629 |
| Slice LUTs | 16978 | 10943 |
| 18K Block RAMs | 77 | 0 |
| MULT cells | 0 | 144 |
| Slices[1] | 5636 | 3190 |
| Operation Freq. (MHz) | 125 | 125 |
| Encryption Rate (Mbps) | 1000 | 1000 |
| Encryption Rate/Slice (Kbps/Slice) | 177.4 | 313.4 |
| Power consumption (W) | 0.775 | 0.332 |

[1]Slices are estimated from the number of register and LUTs, assuming they are not packed together.

are composed by two consecutive symbols, the control symbol /K28.5/ plus a data symbol (/D5.6/ or /D16.2/).

However, when transmitting Ethernet frames, the K flag pattern seems a burst pattern, as idle transmission only occurs in the IFG periods and only data symbols are transmitted between frame boundaries (setting K flag to zero).

When encryption is enabled, K flag pattern seems completely random in both situations, with or without Ethernet traffic being transmitted. This effect makes indistinguishable the data traffic pattern. In Fig. 16, simulation screenshot shows how the K flag pattern in the ciphertext [Fig. 16(c)] is randomized regardless of the traffic pattern of the plaintext [Fig. 16(a) and (b)].

### C. Implementation Results

Regarding the resources used by the proposed solution, in Table II, the main contribution for each block inside each keystream generator (KEYSTREAM GEN) is shown. Modulo 267 (MOD-267) operation and keystream generator bank (STM_BANK) take up the largest amount of FPGA resources.

In the case of multiplication (MULT) cells, 16 of them are necessary to implement the multiplications inside each STM_CELL, as the chaotic map implementation requires it. As there are nine STM cells per STM_BANK, then a total of 144 MULT cells are necessary for each keystream generator.

Moreover, a comparison in terms of hardware resources with the solution proposed in [23] is shown in Table III. The main difference between them is the number of block RAM (BRAM) and MULT cells. While this solution uses MULT cells, in [23], no cell of this type is used. However, the opposite happens with the number of BRAMs.

In addition, power consumption has been estimated in both solutions. First, postsynthesis simulation of both mechanisms

TABLE IV
CHAOTIC CELL COMPARISON WITH OTHER SOLUTIONS

| | LM [39] | MLM [28] | Bernoulli [30] | Hennon [29] | This Work |
|---|---|---|---|---|---|
| Platform | Virtex 5 | Virtex 6 | Spartan 3E | Virtex 6 | Virtex 7 |
| Slice Registers | 64 | 160 | 108 | 64 | 65 |
| Slice LUTs | 129 | 643 | 575 | 1600 | 734 |
| MULT cells | 16 | 16 | 9 | 16 | 16 |
| Slices[1] | 49 | 128 | 342 | 275 | 131 |
| Max. Freq. (MHz) | 26.9 | 93 | 36.9 | 25.7 | 174 |
| Output width | 1 | 16 | 1 | 1 | 8 |
| Encryption Rate (Mbps) | 26.9 | 1488 | 7.38 | 25.7 | 1392 |
| Encryption Rate/ (Slice×Output width) (Mbps/(slice×bit)) | 0.21 | 0.73 | 0.02 | 0.1 | 1.32 |

[1]Slices are estimated from the number of register and LUTs, assuming they are not packed together.

has been performed to obtain the switching activity interchange format files, where port and signal switching rates are recorded. With this information and thanks to the power estimation tools, provided by the FPGA manufacturer, the dynamic power figures have been obtained. The implementation of this paper achieves clearly better figures in *Encryption_Rate/Slice* and power consumption as shown in Table III.

Thanks to this comparison, it is possible to conclude that the structure proposed in this paper, based on a stream cipher next to a modulo operation, is more efficient than an FPE blockcipher working in CTR mode [23].

However, as the stream cipher structure is based on a chaotic map, it is also interesting to make a comparison of the presented chaotic cell with other chaotic solutions. In Table IV, a comparison in terms of hardware resources is made with other chaotic implementations. Particularly, solutions implementing the logistic map [39], modified logistic map [28], Bernoulli [30], and Hennon [29] maps are shown. While in Tables II and III, the overall hardware resources of this solution are shown, and in Table IV, only the hardware relative to the STM cell is taken into account.

As each implementation takes from its internal state different number of bits as output, the comparison has been made in terms of the *Encryption_rate* per slice and output bit. According to this, the proposed solution clearly achieves a better result.

## VII. CONCLUSION

As far as the authors are aware, this is the first time that a chaotic solution for encrypting 1000Base-X Ethernet physical layer has been proposed and developed. The new encryption function PHYsec consists of symmetric ciphering at PCS sublayer of the 8b10b symbol stream transmitted over an optical link. Encryption based on an original chaotic cipher has been tested with real Ethernet traffic and it has been concluded that the proposed system works correctly without harming data traffic or link establishment. In this paper, not only Ethernet frames are ciphered but also the data traffic patterns are masked. These features could improve the security at physical level with no throughput losses, zero space overhead, and low latency.

On the other hand, the proposed keystream generator module entails ratio throughput/resources better than existing FPE implementations. Moreover, other chaotic or nonchaotic secure stream ciphers could be compatible in the proposed scheme.

Finally, as the proposed PHYsec method is suitable for PCS sublayers using 8b10b encoding, the same idea could be extended to other high-speed standards based on this codification, such as Fiber Channel or peripheral component interconnect-express.

## REFERENCES

[1] T. Sauter, S. Soucek, W. Kastner, and D. Dietrich, "The evolution of factory and building automation," *IEEE Ind. Electron. Mag.*, vol. 5, no. 3, pp. 35–48, Sep. 2011.
[2] A. Depari, P. Ferrari, A. Flammini, D. Marioli, and A. Taroni, "A new instrument for real-time Ethernet performance measurement," *IEEE Trans. Instrum. Meas.*, vol. 57, no. 1, pp. 121–127, Jan. 2008.
[3] N. Skorin-Kapov, M. Furdek, S. Zsigmond, and L. Wosinska, "Physical-layer security in evolving optical networks," *IEEE Commun. Mag.*, vol. 54, no. 8, pp. 110–117, Aug. 2016.
[4] M. Furdek, N. Skorin-Kapov, S. Zsigmond, and L. Wosinska, "Vulnerabilities and security issues in optical networks," in *Proc. Int. Conf. Transparent Opt. Netw. (ITCON)*, Graz, Austria, 2014, pp. 1–4.
[5] M. Zafar, H. Fathallah, and N. Belhadj, "Optical fiber tapping: Methods and precautions," in *Proc. High Capacity Opt. Netw. Enabling Technol. (HONET)*, Rihad, Saudi Arabia, 2011, pp. 164–168.
[6] T. Sauter and M. Lobashov, "How to access factory floor information using Internet technologies and gateways," *IEEE Trans. Ind. Informat.*, vol. 7, no. 4, pp. 699–712, Nov. 2011.
[7] D. V. Bhatt, S. Schulze, and G. P. Hancke, "Secure Internet access to gateway using secure socket layer," *IEEE Trans. Instrum. Meas.*, vol. 55, no. 3, pp. 793–800, Jun. 2006.
[8] J. Lázaro, A. Astarloa, J. Araujo, N. Moreira, and U. Bidarte, "MACsec layer 2 security in HSR rings in substation automation systems," *Energies*, vol. 10, no. 2, p. 162, 2017.
[9] K. Guan, J. Kakande, and J. Cho, "On deploying encryption solutions to provide secure transport-as-a-service (TaaS) in core and metro networks," in *Proc. Eur. Conf. Exhib. Opt. Commun.*, Düsseldorf, Germany, 2016, pp. 1–3.
[10] X. Wu, L. Xie, and F. Lim, "Network delay analysis of EtherCAT and PROFINET IRT protocols," in *Industrial Electronics Society*. Dallas, TX, USA: IECON, 2014.
[11] J. D. Decotignie, "Ethernet-based real-time and industrial communications," *Proc. IEEE*, vol. 93, no. 6, pp. 1102–1117, Jun. 2005.
[12] M. Robshaw and O. Billet, *New Stream Cipher Designs: The eSTREAM Finalists*. New York, NY, USA: Springer-Verlag, 2008.
[13] G. Alvarez and S. Li, "Some basic cryptographic requirements for chaos-based cryptosystems," *Int. J. Bifurcation Chaos*, vol. 16, no. 8, pp. 2129–2151, 2006.
[14] Y. Luo, D. Zhang, J. Liu, and Y. Liu, "Cryptanalysis of chaos-based cryptosystem from the hardware perspective," *Int. J. Bifurcation Chaos*, vol. 28, no. 9, p. 1850114, 2018.
[15] N. Mekki, M. Hamdi, T. Aguili, and T. Kim, "A real-time chaotic encryption for multimedia data and application to secure surveillance framework for IoT system," in *Proc. Int. Conf. Adv. Commun. Technol. Netw. (CommNet)*, Marrakesh, Morocco, 2018, pp. 1–10.
[16] G. Bhatnagar and Q. J. Wu, "Chaos-based security solution for fingerprint data during communication and transmission," *IEEE Trans. Instrum. Meas.*, vol. 61, no. 4, pp. 876–887, Apr. 2012.
[17] Z. Hua, F. Jin, B. Xu, and H. Huang, "2D logistic-sine-coupling map for image encryption," *Signal Process.*, vol. 149, pp. 148–161, Aug. 2018.
[18] Z. Hua, S. Yi, and Y. Zhou, "Medical image encryption using high-speed scrambling and pixel adaptive diffusion," *Signal Process.*, vol. 144, pp. 134–144, Mar. 2018.
[19] K. Cuomo, A. Oppenheim, and S. Strogatz, "Circuit implementation of synchronized chaos with applications to communications," *IEEE Trans. Circuits Syst. II, Analog Digit. Signal Process.*, vol. 40, no. 10, pp. 626–633, Oct. 1993.

[20] M. Azzad, C. Tanougast, S. Sadoudi, and A. Dandache, "Real-time FPGA implementation of Lorenz's chaotic generator for ciphering telecommunications," in *Proc. IEEE Int. Circuits Syst. TAISA Conf.*, Toulouse, France, Jun. 2009, pp. 1–4.
[21] J. Hizanidis, S. Deligiannidis, A. Bogris, and D. Syvridis, "Enhancement of chaos encryption potential by combining all-optical and electrooptical chaos generators," *IEEE J. Quantum Electron.*, vol. 46, no. 11, pp. 1642–1649, Nov. 2010.
[22] M. Grapinet, V. Udaltsov, L. Larger, and J. Dudley, "Synchronization and communication with regularly clocked optoelectronic discrete time chaos," *Electron. Lett.*, vol. 44, no. 12, pp. 764–766, 2008.
[23] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma, "Physical layer encryption for industrial ethernet in gigabit optical links," *IEEE Trans. Ind. Electron.*, vol. 66, no. 4, pp. 3287–3295, Apr. 2019.
[24] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma, "Using a chaotic cipher to encrypt Ethernet traffic," in *Proc. IEEE Int. Symp. Circuits Syst. (ISCAS)*, Florence, Italy, May 2018, pp. 1–5.
[25] *IEEE Standard for Ethernet*, IEEE Standard 802.3-2015, Section 3, 36.2.4.9, 2016.
[26] E. Barker and J. Kesley, "Recommendation for random number generation using deterministic random bit generators," NIST-Inf. Technol. Lab., Gaithesburg, MD, USA, Tech. Rep. SP 800-90A Rev.1v, 2015, p. 70.
[27] J. Buttler and T. Sasao, "Fast hardware computation of x mod z," in *Proc. Parallel Distrib. Process. Workshops Phd Forum (IPDPSW)*, Shanghai, China, 2011, pp. 294–297.
[28] A. Pandle and J. Zambreno, "A chaotic encryption scheme for real-time embedded systems: Design and implementation," *Telecommun. Syst.*, vol. 52, no. 2, pp. 515–561, 2013.
[29] P. Dabal and R. Pelka, "FPGA implementation of chaotic pseudo-random bit generators," in *Proc. 19th Int. Conf. Mixed Design Integr. Circuits Syst. (MIXDES)*, Warsaw, Poland, May 2012.
[30] L. De la Fraga, E. Torres-Pérez, E. Tlelo-Cuautle, and C. Mancillas-López, "Hardware implementation of pseudo-random number generators based on chaotic maps," *Nonlinear Dyn.*, vol. 90, pp. 1661–1670, Nov. 2017.
[31] M. García-Bosque, C. Sánchez-Azqueta, A. Pérez, A. Martínez, and S. Celma, "Fast and secure chaotic stream cipher with a MEMS-based seed generator," in *Proc. IEEE Int. Instrum. Meas. Technol. Conf. (I2MTC)*, Turin, Italy, May 2017, pp. 1–6.
[32] M. García-Bosque, A. Pérez, C. Sánchez-Azqueta, and S. Celma, "Application of a MEMS-based TRNG in a chaotic stream cipher," *Sensors*, vol. 17, no. 3, p. E646, 2017.
[33] S. Li, G. Chen, and X. Mou, "On the dynamical degradation of digital piecewise linear chaotic maps," *Int. J. Bifurcation Chaos*, vol. 15, no. 10, pp. 3119–3151, 2005.
[34] L. Kocarev and S. Lian, "Digitized chaos for pseudo-random number generation in cryptography," in *Chaos-Based Cryptography*. Berlin, Germany: Springer, 2009, p. 88.
[35] A. Rukhin *et al.*, "A Statistical test suite for random and pseudorandom number generators for cryptographic applications," NIST-Inf. Technol., Gaithersburg, MD, USA, Tech. Rep. 800-22 Rev.1a, 2010.
[36] D. Knuth, *Art of Computer Programming: Seminumerical Algorithms*, vol. 2. Reading, MA, USA: Addison-Wesley, 1981.
[37] E. Barker and A. Roginsky, "Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths," NIST, Gaithersburg, MD, USA, Tech. Rep. 800-131A, Nov. 2015.
[38] ENISA. (Nov. 2014). *Algorithms, Key Size and Parameters Report*. Accessed: Apr. 9, 2018. [Online]. Available: www.enisa.europa.eu/activities/identity-and-trust/library/deliverables/algorithms-key-size-and-parameters-report-2014
[39] P. Dabal and R. Pelka, "A chaos-based pseudo-random bit generator implemented in FPGA device," in *Proc. Int. Symp. Design Diagnosis Electron. Circuits Syst. (DDECS)*, Cottbus, Germany, Apr. 2011, pp. 151–154.

**Adrián Pérez-Resa** was born in San Sebastián, Spain. He received the M.Sc. degree in telecommunications engineering from the University of Zaragoza, Zaragoza, Spain, in 2005, where he is currently pursuing the Ph.D. degree with the Group of Electronic Design, Aragón Institute of Engineering Research.

He was a Research and Development Engineer with Telnet-RI, Zaragoza. He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His current research interests include high-speed communications and cryptography applications.

**Miguel Garcia-Bosque** was born in Zaragoza, Spain. He received the B.Sc. and M.Sc. degrees in physics from the University of Zaragoza, Zaragoza, in 2014 and 2015, respectively.

He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His current research interests include chaos theory and cryptography algorithms.

**Carlos Sánchez-Azqueta** was born in Zaragoza, Spain. He received the B.Sc., M.Sc., and Ph.D. degrees in physics from the University of Zaragoza, Zaragoza, in 2006, 2010, and 2012, respectively, and the Dipl.-Ing. degree in electronic engineering from the Complutense University of Madrid, Madrid, Spain, in 2009.

He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His current research interests include mixed-signal integrated circuits, high-frequency analog communications, and cryptography applications.

**Santiago Celma** was born in Zaragoza, Spain. He received the B.Sc., M.Sc., and Ph.D. degrees in physics from the University of Zaragoza, Zaragoza, in 1987, 1989, and 1993, respectively.

He is currently a Full Professor with the Group of Electronic Design, Aragon Institute of Engineering Research, University of Zaragoza. He is a Principal Investigator for more than 30 national and international research projects. He has co-authored more than 100 technical papers, 300 international conference contributions, and four technical books. He holds four patents. His current research interests include circuit theory, mixed-signal integrated circuits, high-frequency communication circuits, wireless sensor networks, and cryptography for secure communications.

IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 8, 2020, accepted January 19, 2020, date of publication January 22, 2020, date of current version February 4, 2020.

*Digital Object Identifier 10.1109/ACCESS.2020.2968816*

# A New Method for Format Preserving Encryption in High-Data Rate Communications

**ADRIÁN PÉREZ-RESA, MIGUEL GARCIA-BOSQUE, CARLOS SÁNCHEZ-AZQUETA, AND SANTIAGO CELMA**

Electronic and Communications Engineering Department, University of Zaragoza, 50009 Zaragoza, Spain

Corresponding author: Adrián Pérez-Resa (aprz@unizar.es)

This work was supported in part by the Ministry of Economy and Competitiveness of Spain (MINECO)-European Regional Development Fund (FEDER) under Grant TEC2017-85867-R.

**ABSTRACT** In some encryption systems it is necessary to preserve the format and length of the encrypted data. This kind of encryption is called FPE (Format Preserving Encryption). Currently, only two AES (Advanced Encryption Standard) modes of operation recommended by the NIST (National Institute of Standards and Technology) are able to implement FPE algorithms, FF1 and FF3. These modes work in an electronic codebook fashion and can be configured to encrypt databases with an arbitrary format and length. However, there are no stream cipher proposals able to implement FPE encryption for high data rate information flows. The main novelty of this work is a new block cipher operation mode proposal to implement an FPE algorithm in a stream cipher fashion. It has been called CTR-MOD and it is based on a standard block cipher working in CTR (Counter) mode and a modulo operation. The confidentiality of this mode is analyzed in terms of its IND- CPA (Indistinguishability under Chosen Plaintext Attack) advantage of any adversary attacking it. Moreover, the encryption scheme has been implemented on an FPGA (Field Programmable Gate Array) and has been integrated in a Gigabit Ethernet interface to test an encrypted optical link with a real high data rate traffic flow.

**INDEX TERMS** FPE (format preserving encryption), stream cipher, FPGA (field programmable gate array), Ethernet.

## I. INTRODUCTION

Format Preserving Encryption, FPE, is a kind of encryption used to cipher a plaintext preserving its original length and format [1]–[3]. In the past, some of the first FPE solutions [4], [5], were based mainly on the use of a standard binary block cipher working in a known operation mode. According to them, if the plaintext is in radix $S$, it must be added modulo-$S$ to the block cipher output to produce the ciphertext. Although these techniques are based on standard modes of operation, also used to build stream ciphers, no argument for their security has been given. In addition, in some of them it is necessary to use an unbiasing operation when $S$ is not a power of two [4].

There have been many other proposals for this type of encryption [6], but the only ones approved by the NIST (National Institute of Standards and Technology) are the modes FF1 and FF3 [7]. FF1, originally called FFX (Format-preserving Feistel-based Encryption), was proposed by Bellare *et al.* [8], whereas FF3 corresponds to the BPS-BC component proposed by Brier *et al.* [9]. Both operation modes are based on a non-binary Feistel structure, similar to that shown in Fig. 1, and they are able to encrypt blocks of data with an arbitrary format in an ECB (Electronic Code Book) fashion. In fact, although they are operation modes using AES as the underlying block cipher, they can be considered directly as a kind of FPE block ciphers.

Some application examples for FPE are the encryption of databases with an arbitrary format [6], [10]–[12] such as PANs (Primary Account Numbers) or SSNs (Social Security Numbers), which are not in binary format. Also, FPE can be used in communication systems when it is necessary to encrypt certain protocols, for example, in military or industrial environments [13], [14], or when encrypting some image formats [15].

Regarding the performance of FPE encryption methods, some studies have been done, however they are mainly related to software implementations [14], [16]–[18]. For performance benefits, a hardware implementation could be considered as in [13], [19] or [20].

The associate editor coordinating the review of this manuscript and approving it for publication was Sedat Akleylek.

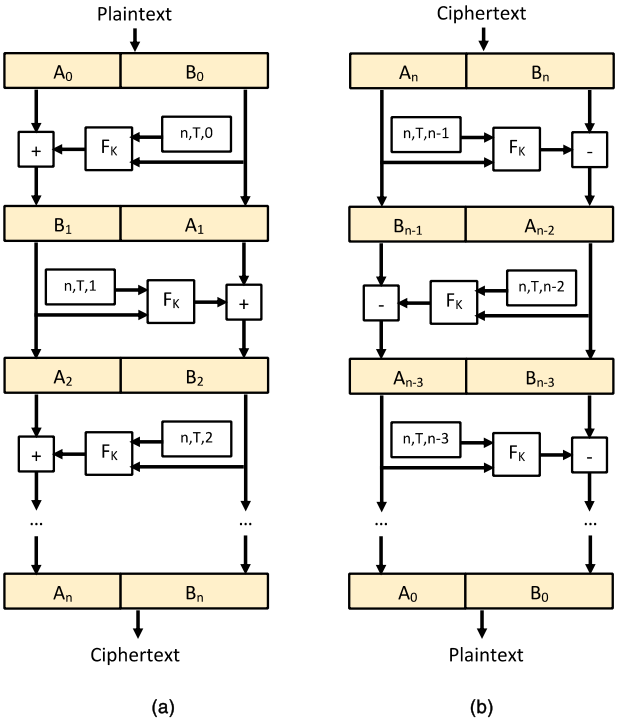



**FIGURE 1.** Generic cipher structure for format-preserving Feistel-based encryption: (a) ciphering (b) deciphering.

A hardware implementation of an FPE stream cipher could be advantageous for that cases where a high encryption rate is important and the plaintext format preservation is mandatory. A tentative solution for this cases could be also the usage of a standard stream cipher, however this would not be a valid solution. As an example, let us to imagine a plaintext formed by symbols in decimal radix. Each symbol will be represented with 4 bits. Then we could try to encrypt the plaintext thanks to a standard stream cipher generating a keystream formed by 4-bit words, which would produce 4-bit ciphertext symbols. As the keystream generator output could be considered random and uniformly distributed, then the XOR operation between the keystream and the plaintext would not guarantee a ciphertext also in decimal radix. For example, if the XOR operation between a symbol of the plaintext and the keystream produced a ciphertext value between 11 and 15, then the resulting value would not maintain the original format. If we needed to preserve the format (length and radix) of the plaintext, the described encryption mechanism with a standard stream cipher would not be valid.

An application example for the utility of FPE stream ciphers could be in [20] or [21], where a Gigabit Ethernet data flow must be encrypted at line rate preserving the 8b/10b encoding properties, which means preserving its format.

Some FPE stream ciphers have been proposed. For example, in the FF3 mode [9], the basic FPE block cipher component BPS-BC is proposed to be used in CBC (Cipher Block Chaining) mode, while in [20], it is proposed to be used in CTR (Counter) mode, as CTR can be considered the best and most modern way to achieve confidentiality-only

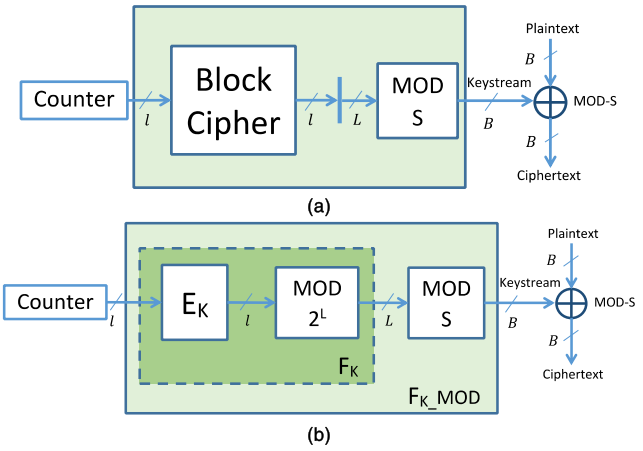



**FIGURE 2.** (a) Structure of the proposed keystream generator using a binary block cipher in CTR mode and a modulo-S operation. (b) Structure of PRF $F_{K\_MOD}$ decomposed in $F_K$ and modulo-S operation. The output of $F_K$ is the least significant L bits taken from the output of block cipher $E_K$ which has a block size of l bits.

encryption [22]. In [21], another proposed solution consisted of using a conventional stream cipher whose output was subjected to a modulo-$S$ operation to encrypt a plaintext in radix $S$. However, in this case the security is not clear as the bias introduced by the modulo-$S$ operation is not analyzed.

The main novelty of this work is the proposal of an FPE stream cipher solution that reduces the hardware complexity of possible solutions based on FPE modes (FF1 and FF3) and is based on a recommended binary block cipher. Moreover, by means of a new operation mode that could use a standard block cipher, such as AES, it is possible to develop a formal security proof, in the same way that is usually done with traditional confidentiality-only operation modes, as in CTR or CBC. The formal security proof consists in the analysis of the IND-CPA (Indistinguishability under Chosen Plaintext Attack) advantage expression of any adversary attacking the proposed mode.

The proposed encryption mode in this work has been called CTR-MOD. To parametrize its resulting structure, a comparison of this mode with other taken as reference has been done in terms of their IND-CPA advantage expressions. The idea is to establish the condition under which CTR-MOD has at least the same or greater security than the reference mode when encrypting the same amount of information. Particularly, the mode used as reference has been CTR, since, as mentioned before, it can be considered the best to achieve confidentiality-only encryption [22].

CTR-MOD mode consists of a standard block cipher working in CTR mode plus a modulo-$S$ operation applied to its output, which is added modulo-$S$ to the plaintext. As shown in Fig. 2a, the keystream, plaintext and ciphertext will be also in radix $S$. In this figure the block size of the block cipher is $l$ bits, from which $L$ are taken and used as input for the modulo-$S$ operator. The output values from the modulo-$S$ operator will be in the range $\{0, \ldots, S-1\}$ and will be represented with $B$ bits where $B = \lceil \log_2 S \rceil$.

**Algorithm 1** $\{C, next\_ctr\} = CTR(M, ctr)$

$CNT_0 = ctr + 1$
Split $M$ into $m$ $L$-bit blocks $\{MB_0, MB_1, \ldots, MB_{m-1}\}$
$KB_i = F_K\,[CNT_i]$ *for* $i = 0, 1, \ldots, m-1$
$CNT_{i+1} = CNT_i + 1$ *for* $i = 0, 1, \ldots, m-2$
$CB_i = (MB_i \oplus KB_i)$ *for* $i = 0, 1, \ldots, m-1$
$next\_ctr = CNT_{m-1}$
$C = \{CB_0, CB_1, \ldots, CB_{m-1}\}$
Return $\{C, next\_ctr\}$

**Algorithm 2** $\{C, next\_ctr\} = CTR - MOD(M, ctr)$

$CNT_0 = ctr + 1$
Split $M$ into $m$ symbols in radix $S$ $\{MB_0, MB_1, \ldots, MB_{m-1}\}$
$KB_i = F_K\_MOD\,[CNT_i]$ *for* $i = 0, 1, \ldots, m-1$
$CNT_{i+1} = CNT_i + 1$ *for* $i = 0, 1, \ldots, m-2$
$CB_i = ((MB_i \oplus KB_i) mod S)$ *for* $i = 0, 1, \ldots, m-1$
$next\_ctr = CNT_{m-1}$
$C = \{CB_0, CB_1, \ldots, CB_{m-1}\}$
Return $\{C, next\_ctr\}$

The paper is divided in eight sections. In Section II both modes CTR and CTR-MOD are introduced, Section III details the IND-CPA advantage expression for CTR-MOD mode. Subsequently, Section IV makes a comparison between the security expression of CTR and CTR-MOD modes to parametrize the resulting structure of the proposed mode. In Section V the application case where the proposed mode has been applied, optical Gigabit Ethernet communication, is described. The implementation and some encryption results of CTR-MOD in the mentioned application case are shown in Sections VI and VII, respectively. Finally, in Section VIII conclusions are given.

## II. CTR AND CTR-MOD MODES

Concrete security analysis for CTR mode was originally established in [23]. This operation mode is a stateful (counter based and deterministic) encryption scheme.

Let us consider a family of PRF (Pseudo Random Function) functions $F$ such that $F : \mathcal{K} \times \{0,1\}^l \rightarrow \{0,1\}^L$ where $L$ is the block size, $\mathcal{K}$ is the keyspace and $F_K$ is a PRF from this family configured with a random key $K$ taken randomly from $\mathcal{K}$ ($K \epsilon \mathcal{K}$). The plaintext is formed by a group of $m$ $L$-bit blocks $M = \{MB_0, MB_1, \ldots, MB_{m-1}\}$ and it is encrypted resulting in a ciphertext formed also by $m$ $L$-bit blocks $C = \{CB_0, CB_1, \ldots, CB_{m-1}\}$ thanks to its encryption function $CTR(M, ctr)$, as described in Algorithm 1.

The inputs of $CTR(M, ctr)$ are the message $M$ itself and the initial value of the counter $ctr$, which is considered the state of this algorithm. $CNT_i$ and $KB_i$ are the values of the counter and keystream block in each encryption step. The $l$-bit counter values are encrypted thanks to the underlying encryption function $F_K$ giving rise to the $L$-bit keystream blocks. The last counter value $CNT_{m-1}$ will be used as next initial $ctr$ value for the next invocation of $CTR(M, ctr)$.

The new proposed structure for the mode CTR-MOD of Fig. 2a can be decomposed as shown in Fig. 2b, where the block cipher in Fig. 2a has been modeled as a PRF $E_K$. The least significant $L$ output bits of $E_K$ are taken as input of the modulo-$S$ operation, which is equivalent to perform the modulo-$2^L$ operation at the output of $E_K$. In this proposed mode we have called $F_K$ and $E_K\_MOD$ to the functions such that $F_K(x) = E_K\_MOD(x) = \left(E_K(x) \bmod 2^L\right)$. In this case, as in $CTR(M,ctr)$ algorithm, $F_K$ in Fig. 2b can be considered a PRF such that $F : \mathcal{K} \times \{0,1\}^l \rightarrow \{0,1\}^L$, which means that it maps the space of values $\{0, \ldots, 2^l - 1\}$ to the range $\{0, \ldots, 2^L - 1\}$. In Fig. 2b the whole module formed by $F_K$ and the modulo-$S$ operation has been called $F_K\_MOD$, which means that $F_K\_MOD(x) = F_K(x) \bmod S$. We can also consider $F_K\_MOD$ a PRF such that $F_K\_MOD : \mathcal{K} \times \{0,1\}^l \rightarrow \{0, \ldots, S-1\}$, as it maps integer values from $\{0, \ldots, 2^l - 1\}$ to $\{0, \ldots, S-1\}$, where $S$ is not necessarily a power of two.

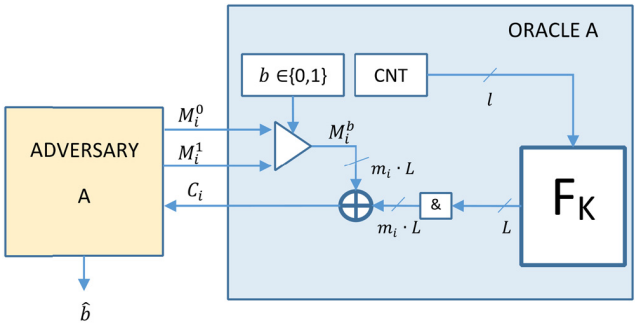



**FIGURE 3.** Scheme of the attack game between the adversary *A* and its oracle performing a CTR encryption scheme. Configuration bit *b* determines which message is encrypted during the game. After *s* queries the adversary outputs bit $\hat{b}$, meant as a guess at *b*.

By taking into account Fig. 2b and the definition of $F_K\_MOD$, the encryption function of CTR-MOD scheme is described in Algorithm 2. This algorithm is similar to Algorithm 1 but using $F_K\_MOD$ function instead of $F_K$ and the addition modulo-$S$ instead of the XOR operation (addition modulo-2). Also, the plaintext, ciphertext and keystream are formed by $m$ symbols in radix $S$ instead of $m$ $L$-bit blocks.

## III. CTR-MOD IND-CPA SECURITY

Usually the security of traditional operation modes for only confidentiality is studied in the sense of IND-CPA security [24]. The attack model is a game between an active adversary $A$ and an encryption oracle performing the encryption scheme $S\mathcal{E}$ configured with a key $K$ and a configuration bit $b$.

During the game the adversary chooses a sequence of $s$ pairs formed by two equal length messages $(M_1^0, M_1^1), \ldots, (M_s^0, M_s^1)$. For each pair of messages $(M_i^0, M_i^1)$ the adversary receives from the oracle the ciphertext $C_i$ corresponding to the message $M_i^b$. Finally the adversary must guess whether $(M_1^0, \ldots, M_s^0)$ or $(M_1^1, \ldots, M_s^1)$ were encrypted during the game. It means that the adversary has to guess the value of the configuration bit $b$ after performing the $s$ queries. Supposing that the $S\mathcal{E}$ is CTR, in Fig. 3 a scheme of the game is shown.

**TABLE 1. Game definitions.**

| Game A: $E_{F'}^{nb}(A)$ | Game B: $E_{F'}^{n}(A)$ | Game C: $E_{F}^{nb}(A)$ | Game D: $E_{F}^{n}(A)$ |
|---|---|---|---|
| $K \xleftarrow{\$} \{0,1\}^k$ | $K \xleftarrow{\$} \{0,1\}^k$ | $K \xleftarrow{\$} \{0,1\}^k$ | $K \xleftarrow{\$} \{0,1\}^k$ |
| $f \xleftarrow{\$} Func(l,L)$ | $f \xleftarrow{\$} Func(l,L)$ | $f \xleftarrow{\$} Func(l,l)$ | $f \xleftarrow{\$} Func(l,l)$ |
| $Fun\_MOD \leftarrow f \bmod S$ | $Func\_MOD \leftarrow f \bmod S$ | $Func'\_MOD \leftarrow f \bmod 2^L$ | $Func'\_MOD \leftarrow f \bmod 2^L$ |
| $case\ (n)\ is$ | $case\ (n)\ is$ | $case\ (n)\ is$ | $case\ (n)\ is$ |
| $0: r \leftarrow Func\_MOD$ | $0: g \leftarrow Func\_MOD$ | $0: r \leftarrow Func'\_MOD$ | $0: g \leftarrow Func'\_MOD$ |
| $1: r \leftarrow F_K\_MOD$ | $1: g \leftarrow F_K\_MOD$ | $1: r \leftarrow E_K\_MOD$ | $1: g \leftarrow E_K\_MOD$ |
| $end\ case$ | $end\ case$ | $end\ case$ | $end\ case$ |
| $case\ (b)\ is$ | $\hat{n} \leftarrow A(g)$ | $case\ (b)\ is$ | $\hat{n} \leftarrow A(g)$ |
| $0: g \xleftarrow{\$} Func(l,T)$ | Return $\hat{n}$ | $0: g \xleftarrow{\$} Func(l,L)$ | Return $\hat{n}$ |
| $1: g \leftarrow r$ | | $1: g \leftarrow r$ | |
| $end\ case$ | | $end\ case$ | |
| $\hat{b} \leftarrow A(g)$ | | $\hat{b} \leftarrow A(g)$ | |
| Return $\hat{b}$ | | Return $\hat{b}$ | |

In this figure, for each message $M_i^b$ with a length of $m_i$ $L$-bit blocks the oracle performs the CTR encryption as in Algorithm 1, generating $m_i$ keystream blocks $KB$ of length $L$ bits. Symbol '&' represents the concatenation of the $m_i$ keystream blocks that have to be XORed with $M_i^b$.

To measure the success of the adversary in breaking a symmetric encryption scheme S$\mathcal{E}$, the adversary advantage is defined in [25] as in the following equation:

$$ADV_{SE}^{IND-CPA}(A) = 2 \cdot \Pr\left(\hat{b} = b\right) - 1 \tag{1}$$

where the $ADV_{SE}^{IND-CPA}(A)$ is the IND-CPA advantage of the adversary $A$ over the encryption scheme S$\mathcal{E}$, and $\Pr\left(\hat{b} = b\right)$ is the probability of the adversary $A$ of guessing the correct value of configuration bit $b$. The advantage of $A$ can be understood as the excess of this probability over 1/2. When the 'guess' probability is almost 1/2 and then the adversary advantage is negligible the encryption scheme S$\mathcal{E}$ can be considered secure.

IND-CPA advantage for CTR mode can be expressed as in Theorem 1, which is proven in [23].

*Theorem 1:* Let $F_K : \mathcal{K} \times \{0,1\}^l \rightarrow \{0,1\}^L$ be the underlying function of the encryption scheme S$\mathcal{E}$ that corresponds with CTR symmetric encryption mode. Let $A$ be an adversary attacking the IND-CPA security of S$\mathcal{E}$ that asks at most $s$ queries formed each one by a pair of messages $(M_i^0, M_i^1)$ with a length of $m_i$ blocks with $L$-bit length each one. The $s$ message queries will produce a total number of $qL$-bit encrypted blocks which means that $q = \sum_{i=1}^{i=s} m_i$. Then an adversary $B$ (attacking the PRF security of $F_K$ and performing $q$ queries) can be built thanks to $A$, such that:

$$ADV_{CTR}^{IND-CPA}(A) \leq 2 \cdot ADV_{F_K}^{PRF}(B) \tag{2}$$

where $ADV_{F_K}^{PRF}(B)$ is the prf-advantage [24] of any adversary $B$ over $F_K$ and $ADV_{CTR}^{IND-CPA}(A)$ is the IND-CPA advantage of $A$ over the encryption scheme CTR.

Our purpose is to obtain the IND-CPA advantage for CTR-MOD to compare its security with the typical CTR scheme and in this way extract the necessary conditions to achieve at least the same or greater security. It can be proved that this advantage can be expressed as in the following theorem:

*Theorem 2:* Let $F_K\_MOD : \mathcal{K} \times \{0,1\}^l \rightarrow \{0,\ldots,S-1\}$ be the underlying function of the encryption scheme S$\mathcal{E}$ that corresponds with CTR-MOD symmetric encryption mode. Let $A$ be an adversary attacking the IND-CPA security of S$\mathcal{E}$ that asks at most $s$ queries formed each one for a pair of messages $(M_i^0, M_i^1)$ with a length of $m_i$ symbols of radix $S$. The $s$ message queries will produce a total number of $q$ encrypted symbols, which means that $q = \sum_{i=1}^{i=s} m_i$.

Then it is possible to build an adversary $D$ (attacking the PRF security of $E_K$ and performing $q$ queries) such that:

$$ADV_{CTR-MOD}^{IND-CPA}(A) \leq 2 \cdot ADV_{E_K}^{PRF}(D) + \frac{q}{2^{I-1}} \tag{3}$$

where $ADV_{E_K}^{PRF}(D)$ is the prf-advantage of any adversary $D$ over $E_K$ as defined in [24], $E_K$ corresponds to the block cipher that is part of the $F_K\_MOD$ function and $I$ is the difference between $L$ (the input bit length of modulo-$S$ operation in $F_K\_MOD$) and $T$, with $T = log_2 S$. The proof of this theorem is developed in the Appendix A thanks to the games described in Table 1 and the definition of the prf-advantage term $ADV_{F_K}^{PRF}$. The explanation of these games and the term $ADV_{F_K}^{PRF}$ are described in Appendix B.

## IV. SECURITY ANALYSIS: CTR-MOD VS CTR

Although usually block ciphers are analyzed as PRFs, PRPs (Pseudo Random Permutation) are what best models them. Thanks to the PRF-PRP switching lemma [23] it is possible to relate the PRF and PRP advantages of an adversary against a block cipher as shown in (4).

$$ADV_{E_K}^{PRF}(A) \leq ADV_{E_K}^{PRP}(A) + \frac{q^2}{2^{l+1}} \tag{4}$$

**TABLE 2. IND-CPA Advantage comparison of different modes.**

| Encryption Mode $\mathcal{SE}$ | IND-CPA Advantage expression[1] $ADV^{IND-CPA}_{\mathcal{SE}}(A)$ |
|---|---|
| CTR-MOD | $2 \cdot ADV^{PRP}_{E}(B) + \frac{\mu^2}{T^2 2^l} + \frac{\mu}{T \cdot 2^{l-1}}$ |
| CTR | $2 \cdot ADV^{PRP}_{E}(B) + \frac{\mu^2}{l^2 2^l}$ |
| CTR$ | $2 \cdot ADV^{PRP}_{E}(B) + \frac{2\mu^2}{l^2 2^l}$ |
| CBC | $2 \cdot ADV^{PRP}_{E}(B) + \frac{2\mu^2}{l^2 2^l}$ |
| CFB[2] | $2 \cdot ADV^{PRF}_{F}(B) + \frac{\mu^2}{m^2 2^{l+1}}$ |

[1]In each expression $l$ is the block size, $\mu$ the number of encrypted bits and $T$ the bits mapped per symbol in CTR-MOD mode.

[2]The term $ADV^{PRF}_{F}$refers to the prf-advantage where $F_K$ is the function $select(E_K(\cdot)).E_K(\cdot)$ is the block cipher with blocksize $l$ and $select(\cdot)$ is a function that outputs $m$ fixed bits from its input.

where $ADV^{PRF}_{E_K}(A)$ and $ADV^{PRP}_{E_K}(A)$ are the prf-advantage and prp-advantage of adversary $A$ against block cipher $E_K$, respectively. The block size is $l$ and the number of encryption queries performed by the adversary during the prf-advantage game is $q$.

According to (4), equations (2) and (3) of Theorems 1 and 2 can be rewritten as:

$$ADV^{IND-CPA}_{CTR}(A) \leq 2 \cdot ADV^{PRP}_{E_K}(B) + \frac{q^2_{CTR}}{2^{l_{CTR}}} \tag{5}$$

$$ADV^{IND-CPA}_{CTR-MOD}(A) \leq 2 \cdot ADV^{PRP}_{E_K}(D) + \frac{q^2_{CTR-MOD}}{2^{l_{CTR-MOD}}} + \frac{q_{CTR-MOD}}{2^{I-1}} \tag{6}$$

where $E_K$ is the underlying block cipher used in both modes, $q_{CTR}$ and $q_{CTR-MOD}$ are the number of encrypted blocks and symbols during the IND-CPA games of each mode, and $l_{CTR}$ and $l_{CTR-MOD}$ are the block sizes of $E_K$ in CTR and CTR-MOD, respectively. By each encrypted block CTR mode encrypts $l_{CTR}$ information bits, while by each encrypted symbol CTR-MOD encrypts $T$ information bits ($T = log_2 S$). Therefore the total number of encrypted bits in each mode during the IND-CPA game is $\mu_{CTR} = q_{CTR} \cdot l_{CTR}$ and $\mu_{CTR-MOD} = q_{CTR-MOD} \cdot T$, respectively.

According to this, it is possible to express the advantages of equations (5) and (6) in terms of the encrypted bits and compare them with the expressions of other well-known operation modes, as shown in Table 2.

As mentioned in Section I, we want to parametrize CTR-MOD and establish under what condition it has at least the same security as the classical CTR scheme when encrypting the same amount of information, with $\mu_{CTR} = \mu_{CTR-MOD} = \mu$. It means that we want to know the constraints needed to get the following condition:

$$ADV^{IND-CPA}_{CTR-MOD}(A) \leq ADV^{IND-CPA}_{CTR}(A) \tag{7}$$

If we assume that $E_K$ is in both modes a secure and a recommended cipher we can consider that it is a good PRP and has a great prp-security, which means that the term $ADV^{PRP}_{E_K}$ is negligible for both expressions (5) and (6). Then it is only needed to compare the second terms of both expressions to meet (7), as shown in the following equation:

$$\frac{q^2_{CTR-MOD}}{2^{l_{CTR-MOD}}} + \frac{q_{CTR-MOD}}{2^{I-1}} \leq \frac{q^2_{CTR}}{2^{l_{CTR}}} \tag{8}$$

As $q_{CTR} = \mu_{CTR}/l_{CTR}$, $q_{CTR-MOD} = \mu_{CTR-MOD}/T$, and $\mu_{CTR} = \mu_{CTR-MOD} = \mu$ then (8) can be rewritten as:

$$\frac{\mu^2}{T^2 \cdot 2^{l_{CTR-MOD}}} + \frac{\mu}{T \cdot 2^{I-1}} \leq \frac{\mu^2}{l^2_{CTR} \cdot 2^{l_{CTR}}} \tag{9}$$

It is possible to rewrite (9) as:

$$\frac{1}{\mu} \leq T \cdot 2^{I-1} \cdot \left( \frac{1}{l^2_{CTR} \cdot 2^{l_{CTR}}} - \frac{1}{T^2 \cdot 2^{l_{CTR-MOD}}} \right) \tag{10}$$

In (10) $I = L - T$, where $L$ is the input bit length of modulo-$S$ operation in CTR-MOD mode. If we define the difference in bits between the output of the block cipher $E_K$ and the input to the modulo-$S$ operation as $P = l_{CTR-MOD} - L$, it is possible to rewrite (10) as:

$$L \geq T + 1 + \log_2 \left( \frac{l^2_{CTR} \cdot 2^{l_{CTR}}}{T} \cdot \left( \frac{1}{\mu} + \frac{1}{T \cdot 2^{P+T+1}} \right) \right) \tag{11}$$

As $\mu \geq 1$ and $P \geq 0$, then the lowest bound for $L$ that always meets (11) is:

$$L \geq T + 1 + \log_2 \left( \frac{l^2_{CTR} \cdot 2^{l_{CTR}}}{T} \cdot \left( 1 + \frac{1}{T \cdot 2^{T+1}} \right) \right) \tag{12}$$

It is possible to conclude that if the underlying block ciphers used in CTR and CTR-MOD modes have the same prp-security, and the block size $l_{CTR-MOD} \geq L$, then CTR-MOD scheme can have equal or better IND-CPA security than CTR when encrypting the same amount of information. It is important to notice that the block size $l_{CTR-MOD}$ used in CTR-MOD will depend on the block size $l_{CTR}$ of the CTR mode used as reference and the radix $S$ of the plaintext, as $T = log_2 S$. Expression in (12) will be the constraint necessary to achieve condition (7).

## V. APPLICATION CASE: ETHERNET 1000BASE-X

As we have mentioned in Section I, as far as the authors are concerned, there are no standardized solutions for FPE stream ciphers and its usage could be relevant in the cases where a high encryption rate is necessary. For example, in the case of the encryption in 1000Base-X standard for Gigabit Ethernet optical links [20] .

The encryption in a layered communication model such as TCP/IP can be performed at different levels of the communication, such as in layers 2 or 3 with MACsec or IPsec standards, respectively. Although encryption in physical layer (layer 1) is less usual than in other layers some proposals

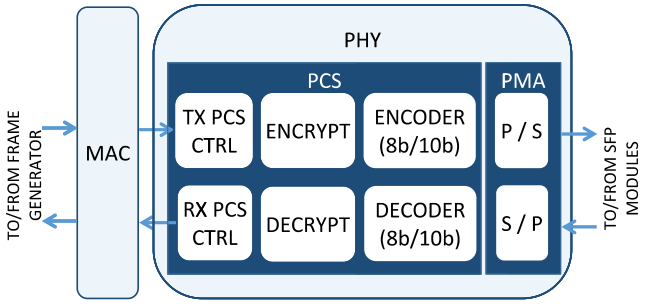



**FIGURE 4.** Scheme of the Ethernet Interface formed by the PHY and MAC (Medium Access Control) modules. MAC layer builds the Ethernet packets transmitted to the PHY, which includes the PCS and PMA (Physical Medium Attachment) sublayers. ENCRYPT and DECRYPT modules perform the format preserving encryption/decryption of 8b/10b symbols at the PCS sublayer. P/S and S/P modules are Parallel to Serial and Serial to Parallel modules, that transmit and receive the bitstream from the optical link.

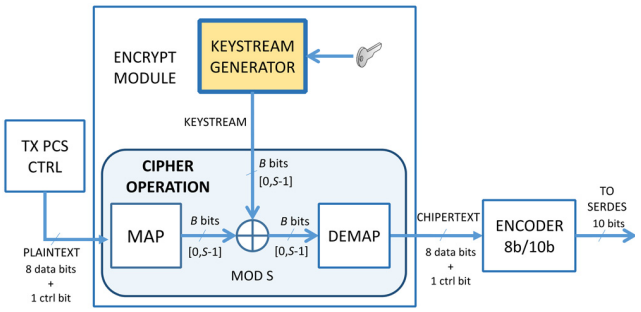



**FIGURE 5.** Location and generic structure of a stream cipher in a physical layer with 8b/10b line encoding. 8b/10b symbols are formed by eight data bits and one control bit. These symbols are mapped in CIPHER_OPERATION block thanks to MAP and DEMAP modules. The mapped symbols are represented with $B = \lceil \log_2 S \rceil$ bits.

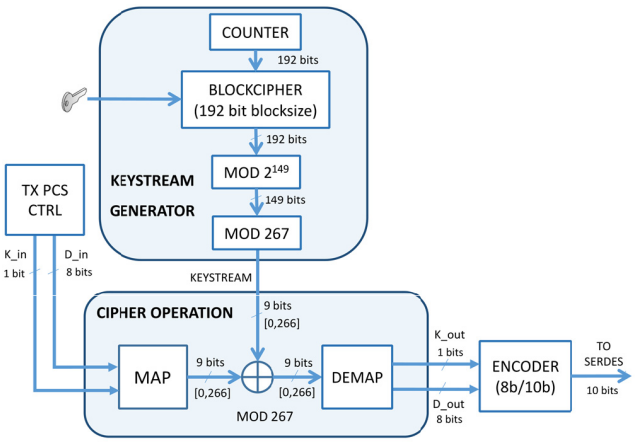



**FIGURE 6.** Overall structure for the streaming encryption system in a physical layer with 8b/10b encoding using CTR-MOD mode. Decryption will be as encryption but using a modulo-267 subtraction instead of an addition. 8b/10b symbols are formed by 8 data bits, D_in, and one control bit K_in. The symbols are mapped to 9-bit values (B = 9) in the range [0, 266] as S = 267.

have been made, for example related with photonic [26]–[29] or radio [30], [31] technologies or with the physical layer protocols [20], [21], [32].

One of the key benefits of performing the encryption at layer 1 is the possibility of masking the data traffic pattern, then achieving additional privacy, which permits to hide the existence of data transmission as in [26] with the optical steganography or in [20] with the encryption at encoding sublayer.

Our application case is the standard 1000Base-X, as in [20], [21]. In both cases the encryption is performed in one of the sublayers of physical layer, where the 8b/10b encoding is performed. This sublayer is called PCS (Physical Coding Sublayer). The 8b/10b encoding at PCS is used to provide certain properties to the bitstream that is transmitted through the Gigabit Ethernet optical link, such as DC balance, high transition density and short run length. Ciphering at PCS level must be performed in a way that the 8b/10b encoding properties are preserved, which means that the encryption method must preserve the same format in the plaintext and the ciphertext.

On the other hand, encryption and decryption modules must be located in the 1000Base-X datapath as shown in Fig. 4, where an optical Ethernet interface is shown.

In order to preserve the coding properties, the encryption of an 8b/10b symbol must give as result another valid 8b/10b symbol, which means to perform an FPE encryption. Ciphered symbols must be within the alphabet of symbols supported by the encoding standard. For this reason, the generic structure of the ENCRYPT module is built as shown in Fig. 5.

In Fig. 5, since $S$ is the possible number of valid 8b/10b symbols, these are mapped to an integer value in the range $\{0, \ldots, S-1\}$, giving rise to a plaintext in radix $S$. Then a modulo-$S$ addition is performed between the mapped symbols and a keystream also represented with values in the range $\{0, \ldots, S-1\}$. After that, the resulting ciphered values are reverse-mapped to its corresponding 8b/10b symbols which are finally encoded to 10-bit values and sent to the serializer.

In the 1000Base-X standard only 267 possible symbols are valid in the 8b/10b encoding, which means that $S$ =267.

In this work, CTR-MOD operation mode has been used to perform the keystream generation and the modulo-$S$ addition of Fig. 5, in the same way as Algorithm 2 of Section II.

## VI. SYSTEM IMPLEMENTATION

We assume that we want to achieve at least the same or better IND-CPA security than a recommended block cipher working in CTR mode. Let be the block and key size of this reference block cipher a standard length of 128 bits. According to (12) if $l_{CTR} = 128$ bits, and $T = log_2 S \cong 8.06$ bits, then $l_{CTR-MOD}$ must be $l_{CTR-MOD} \geq L \geq 149$ bits to achieve the security that meets condition (7). By taking into account these parameters, proposed CTR-MOD structure has been adapted to the generic scheme of Fig. 5. The resulting encryption scheme is shown in Fig. 6.

As the underlying block cipher must have a block size greater than 128 bits because $l_{CTR-MOD} \geq 149$, the well-known Rijndael [33] cipher has been used. The main difference between Rijndael and AES (Advance Encryption Standard) is the range of configuration values for the block

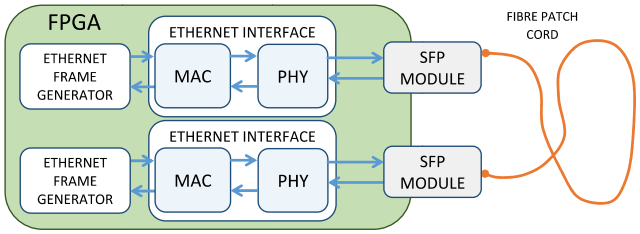



**FIGURE 7.** Test set-up scheme.

**TABLE 3.** Comparison with other hardware solutions.

| Resource type | FF1 [13] | FF3 [13] | FPE [20] | SC[2] [21] | This Work |
|---|---|---|---|---|---|
| Slice Registers | 11285 | 5592 | 11127 | 6097 | 8807 |
| Slice LUTs | 7426 | 3587 | 16978 | 13391 | 10974 |
| 18K Block RAMs | 343 | 170 | 77 | 0 | 78 |
| DSP cells | 0 | 0 | 0 | 144 | 0 |
| Slices[1] | 3268 | 1596 | 5636 | 4110 | 3844 |
| Encryption Rate (Mbps) | 41.1 | 109.6 | 1000 | 1000 | 1000 |
| Encryption Rate/Slice (Kbps/Slice) | 12.57 | 68.7 | 177.4 | 243.3 | 260.1 |

[1]Slices are estimated from the number of register and LUTs, assuming they are not packed together.
[2]For the particular case of [21] the amount of resources has been calculated supposing the same input width in the modulo-267 operation.

size and the key length, in fact AES is a subset of Rijndael. While in AES the block size is fixed to 128 bits, in Rijndael it can take three values, 128, 192 and 256 bits. In this work Rijndael has been configured with 192 bit block size and a 128 bit key length, which means $l_{CTR-MOD} = 192$.

The block MOD_$2^{149}$ of Fig. 6 is simply to take the 149 least significant bits of the Rijndael output. However, the second modulo operation, the MOD_267 module, takes more resources as 267 is not a power of two. Its implementation has been based on [34], which presents a high-speed hardware structure for a generic operation '*x mod z*'.

The final structure shown in Fig. 6 has been synthetized using a Xilinx Virtex 7 FPGA. Regarding the hardware resources used, they have been compared in Table 3 with other solutions based in the mentioned FF1 and FF3 modes for FPE [13], [20], and an ad-hoc stream cipher [21]. In this table, the amount of registers, LUTs (Look Up Tables) and BRAMs (Block RAMs) are shown. According to these results it is possible to conclude that the solution in this work achieves a better figure in *Encryption_Rate/Slice* than the others.

Finally, to test the encryption mechanism with real traffic, the KEYSTREAM_GENERATOR and CIPHER OPERATION modules of Fig. 6 have been integrated in the ENCRYPT and DECRYPT modules of the Ethernet Interface in Fig. 4. Two Ethernet Interfaces have been implemented in the Xilinx FPGA platform as shown in Fig.7. The PHY sides of both interfaces have been connected to optical SFP (Small Form Factor Pluggable) modules able to transmit at 1 Gbps data rate through a fiber link. Also two Ethernet Frame Generators have been connected to the MAC sides of both Ethernets Interfaces to test the encrypted link with real traffic. With these generators it is possible to produce Ethernet frame flows configured with different frame size, payload and interframe gap.

Note that in Table 3 we have only compared the solution in this work with other FPE solutions. A comparison of the proposed system with other well-known binary stream cipher implementations could be done. However, as mentioned in Section I this kind of ciphers cannot be a valid solution to preserve the format of the plaintext. In addition, although stream ciphers are suitable for high speed applications, their cryptanalysis and design criteria are less understood than block ciphers [35]. Indeed, a stream cipher application can be implemented easily thanks to a secure block cipher such as AES working in CTR mode, considered secure thanks to its formal security proof [23]. In the same way CTR-MOD mode can be considered enough safe as an FPE stream cipher structure.

## VII. ENCRYPTION RESULTS

As mentioned in previous Section, one of the key benefits of performing the encryption at layer 1 is the possibility of masking the data traffic pattern, achieving additional privacy, as possible passive attackers are not able to detect the presence of current communications.

In the particular case of 1000Base-X, the transmission of 8b/10b symbols is carried out constantly, including in the case no frame is being transmitted or during the gap between frames. In these situations, the PHY always transmits idle sets of 8b/10b symbols, whose purpose is to maintain the synchronization between remote terminals.

To check the masking capability of the encryption, the SE (Shannon Entropy) in (13) has been measured for different encrypted and non-encrypted frame traffic patterns. The 8b/10b symbol stream for each traffic pattern, mapped between 0 and $S-1$, has been grouped in tuples of $t$ symbols called $\beta_t$, and the probability for each tuple, $P(\beta_t)$, has been calculated. Particularly, $SE$ has been measured for values of $t$ equals to 1 and 2.

$$SE = -\frac{1}{t} \cdot \sum_{0 \le \beta_t < S^t} P(\beta_t) \cdot \log_2 P(\beta_t) \qquad (13)$$

Ideally, if every $t$-tuple $(\beta_t)$ is equally likely with probability $P(\beta_t) = p = S^{-t}$ the value of Shannon Entropy for every $t$ should be equals to $SE = \log_2 S = \log_2 267 \cong 8.0606$.

In Fig. 8 the $SE$ measured for different traffic patterns is shown. Pattern A corresponds with no frame transmission, where only idle sets are transmitted over the link. Patterns B, C and D correspond to continuous frame transmission of 1024-bytes length with random payload at rates of 10.2%, 50% and 91% of the maximum Gigabit line rate. Pattern E corresponds to continuous frame transmission of random

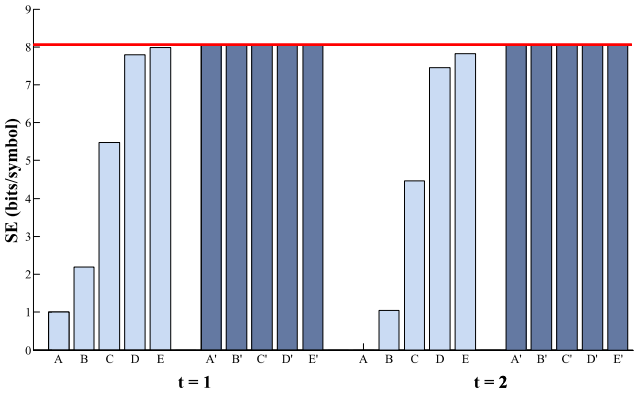



**FIGURE 8.** Shannon Entropy measured for different traffic flows of 8b/10b symbols grouped in tuples of t symbols with t from 1 to 2. Ethernet traffic patterns are called A, B, C, D and E and their encrypted versions, A', B', C', D' and E'. The red line marks the maximum entropy achievable with an 8b/10b data flow.

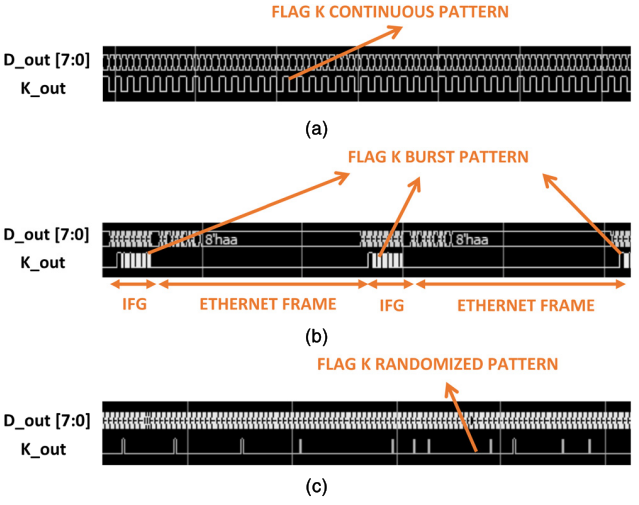



**FIGURE 9.** (a) K_out flag pattern without encryption when no Ethernet frame is transmitted; (b) K_out flag pattern without encryption when transmitting an Ethernet frame burst; (c) K_out flag pattern after encryption regardless of the transmission or non-transmission of Ethernet frames.

length and payload and minimum gap between frames. The non-encrypted versions of these patterns have different *SE*, however every encrypted version has the maximum possible *SE*, which makes them indistinguishable from each other and therefore proves the masking property, as shown in Fig. 8.

Moreover, to appreciate graphically this masking property is interesting to show the signal waveforms of 8b/10b flows after the encryption module. As shown in Fig. 6, 8b/10b symbols are formed by one control bit and eight data bits. In Fig. 6, after the encryption operation, these are called *K_out* and *D_out*, respectively, and they are used by the 8b/10b encoder to generate 10 bits. Each 8b/10b symbol is a control or data one depending whether its *K_out* flag is '1' or '0', respectively. When no frames are transmitted (pattern A) and encryption is disabled, Ethernet physical layer always transmit continuously idle sets composed by two consecutive symbols, one control symbol (with *K_out* equals to '1') and a data symbol (with *K_out* equals to '0'). It means that the *K_out* pattern in this situation is a signal that switches continuously between '0' and '1', as shown in Fig. 9a. When frames are transmitted (patterns B, C, D, E), the *K_out* flag will behave as a blast signal, as the idle sets are only transmitted in the space between frames. During frame transmission only 8b/10b data symbols are transmitted which will make *K_out* flag remain to '0' as shown in Fig. 9b.

Finally, by enabling encryption, 8b/10b symbols are ciphered and *K_out* flag and *D_out* are randomized (in every flow A, B, C, D or E) making indistinguishable which pattern is being transmitted, as shown in Fig. 9c.

## VIII. CONCLUSION

In this work the authors have presented a new block cipher operation mode able to preserve the format of the plaintext when encrypting a high speed data stream. A security analysis has been made proving that it is possible to get at least the same or better security than a traditional CTR mode working with a standard 128-bits block cipher. The proposed solution permits the use of classical block ciphers instead of ad-hoc stream ciphers whose cryptanalysis and design criteria are less understood, or FPE structures whose architecture is more complex and entails larger hardware resources.

A parametrization and implementation of this FPE proposal has been carried out for the specific case of optical Gigabit Ethernet communications, achieving an *Encryption_Rate/Slice* better than in other existing FPE solutions, including FF1 and FF3 modes.

Finally, the masking property of the encryption at physical layer in 1000Base-X standard has been tested with the proposed operation mode, checking that different data patterns can be made indistinguishable from each other, including from the situation of no frame transmission.

## APPENDICES

## APPENDIX A

## PROOF OF THEOREM 2

According to [23], and taking into account that CTR-MOD is a CTR-like mode but using a modulo-$S$ addition (instead of an XOR operation) and an underlying PRF function $F_K\_MOD$ that maps integer values from $\{0,\ldots,2^l-1\}$ to $\{0,\ldots,S-1\}$ instead from $\{0,\ldots,2^l-1\}$ to $\{0,\ldots,2^L-1\}$ as $F_K$, it is possible to express the IND-CPA security of CTR-MOD in a similar way as in (2) such that:

$$ADV^{IND-CPA}_{CTR-MOD}(A) \leq 2 \cdot ADV^{PRF}_{F_K\_MOD}(B) \qquad (14)$$

where $ADV^{PRF}_{F_K\_MOD}(B)$ is the prf-advantage of an adversary $B$ over $F_K\_MOD$. Therefore, by obtaining this advantage it will be possible to get the expression for the IND-CPA advantage over the proposed scheme. The generic prf-advantage $ADV^{PRF}_{F_K}$ of any adversary over a PRF $F_K$ is defined in [24] and an explanation about it is detailed in the Appendix B.

Thanks to the games defined in Table 1 it is possible to express the term $ADV^{PRF}_{F_K\_MOD}(A)$ according the following lemma:

*Lemma 1:* Let be $F_K$ and $E_K$ PRFs taken from the families of functions $F : \mathcal{K} \times \{0,1\}^l \rightarrow \{0,1\}^L$ and $E : \mathcal{K} \times \{0,1\}^l \rightarrow \{0,1\}^l$. Let be *Func_MOD* and *Func′_MOD* two functions defined as in Table 1. On the one hand, $Func\_MOD(x) = f(x) \, mod S$, where $f$ is a random function taken from the set of functions $Func(l, L) : \{0,1\}^l \rightarrow \{0,1\}^L$. On the other hand $Func'\_MOD(x) = f(x) \, mod 2^L$, where $f$ is a random function taken from the set of functions $Func(l, l) : \{0,1\}^l \rightarrow \{0,1\}^l$. Then their prf-advantages can be related as in the following equation:

$$ADV^{PRF}_{F_K\_MOD}(A) = ADV^{PRF}_{E_K}(D) + ADV^{PRF}_{Func'\_MOD}(A) + ADV^{PRF}_{Func\_MOD}(A) \quad (15)$$

In addition, it is possible to express the prf-advantages of any adversary $A$ over the functions *Func_MOD* and *Func′_MOD* according to the Lemmas 2 and 3, respectively:

*Lemma 2:* Let be *Func_MOD* a function defined as in Lemma 1. Then any adversary $A$ making $q$ oracle queries when attacking the prf-security of *Func_MOD* will obtain an advantage bounded by the following expression:

$$ADV^{PRF}_{Func\_MOD}(A) \leq \frac{q}{2^I} \quad (16)$$

where $I = L - T$ and $T = log_2 S$.

*Lemma 3:* Let be *Func′_MOD* a function defined as in Lemma 1. Then any adversary $A$ making $q$ oracle queries when attacking the prf-security of *Func′_MOD* will obtain an advantage such that:

$$ADV^{PRF}_{Func'\_MOD}(A) = 0 \quad (17)$$

By taking into account Lemma 2 and Lemma 3 it is possible to derive the following expression from equation (15):

$$ADV^{PRF}_{F_K\_MOD}(A) \leq ADV^{PRF}_{E_K}(D) + \frac{q}{2^I} \quad (18)$$

Finally, by substituting the expression of the term $ADV^{PRF}_{F_K\_MOD}(A)$ in equation (14) it is possible to derive (3) which proves Theorem 2. For clarity purposes, proofs of Lemmas 1, 2 and 3 are given in the Appendices C, D and E using the games definitions of Table 1. Game definitions are explained in Appendix B.

## APPENDIX B
## GAME DEFINITIONS

In this Section the definition of term $ADV^{PRF}_{F_K}$ and the explanation of games in Table 1 are described. These are necessary to develop the proof of Theorem 2 and Lemmas 1, 2 and 3.

### A. PRF ADVANTAGE $ADV^{PRF}_{F_K}$

The generic prf-advantage $ADV^{PRF}_{F_K}(A)$ of any adversary $A$ over a PRF $F_K$ is defined in [24] according to the following definition:

*Definition 1:* Let $F : \mathcal{K} \times \{0,1\}^l \rightarrow \{0,1\}^L$ be a family of functions, and let adversary $A$ be an algorithm that takes from an oracle a function $g: \{0,1\}^l \rightarrow \{0,1\}^L$, that can be configured as a random function or a PRF $F_K$ depending on a bit $b$, and tries to distinguish which function has been taken. The function $g$ is passed to the adversary as an argument and it returns a result bit $\hat{b}$ ($\hat{b} \leftarrow A(g)$) meant as a guess at $b$. According to the game $EX^b_F$ in Table 4 the prf-advantage of $A$ over $F_K$ is defined as:

$$ADV^{PRF}_{F_K}(A) = P\left[EX^1_F(A) = 1\right] - P\left[EX^0_F(A) = 1\right] \quad (19)$$

where $P\left[EX^b_F(A) = 1\right]$ is the probability that the result of the game $EX^b_F(A)$ is $\hat{b} = 1$.

**TABLE 4.** Game $EX^b_F$.

| Game $EX^b_F(A)$ |
|---|
| $K \xleftarrow{\$} \{0,1\}^k$; $f \xleftarrow{\$} Func(l, L)$ |
| *case* $(b)$ *is* |
| 0: $g \leftarrow f$ |
| 1: $g \leftarrow F_K$ |
| *end case* |
| $\hat{b} \leftarrow A(g)$; Return $\hat{b}$ |

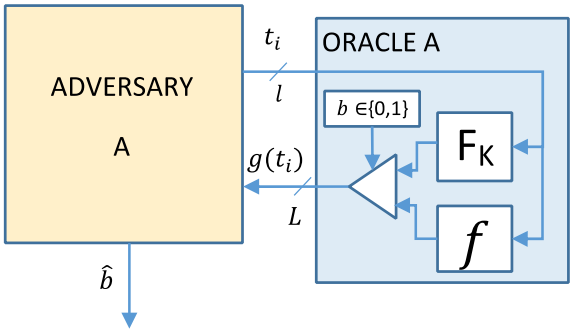



**FIGURE 10.** Scheme of the game $EX^b_F(a)$ between the adversary $A$ and its oracle. Configuration bit $b$ determines which function is being implemented by the oracle, $F_K$ or $f$. The outputs of the resulting function $g$ are analyzed by $A$ to return $\hat{b}$, meant as a guess at $b$.

The game $EX^b_F(A)$ is configured with the bit $b$. If $b = 1$ a key of length $k$ bits is selected randomly from the keyspace $\mathcal{K}$. It is expressed as $K \xleftarrow{\$} \{0,1\}^k$, where the operator $\xleftarrow{\$}$ means a 'random selection'. In addition, the argument $g$ for the adversary $A$ is selected to be the PRF $F_K$ configured with key $K$. On the other hand, when the game $EX^b_F(A)$ is configured with $b = 0$ the argument for $A$ is a random function $f$, taken randomly from the whole set of functions $Func(l, L) : \{0,1\}^l \rightarrow \{0,1\}^L$, which is expressed as $f \xleftarrow{\$} Func(l, L)$. The value of $ADV^{PRF}_{F_K}(A)$ measures how well $A$ is doing when distinguishing between $f$ and $F_K$. Game $EX^b_F$ can be represented graphically as in Fig. 10, where the oracle implements the function $g$, that can be $F_K$ or $f$ depending on the value of $b$. The adversary $A$ interacts with $g$ by performing $q$ queries of $l$-bit values $t_i$, $\{t_0, t_1, \ldots, t_{q-1}\}$, and the oracle responds to the adversary queries with $qL$-bit output values $g(t_i)$. Finally $A$ returns $\hat{b}$ meant as guess at $b$.

In order to obtain the expression for the $ADV^{PRF}_{F_K\_MOD}(A)$ we have defined a set of games shown in Table 1. In these games, as in game $EX^b_F$, the operator $\xleftarrow{\$}$ means a 'random selection', and $Func(l, L)$, $Func(l, l)$

and $Func(l, T)$ represent sets of functions such that $Func(l, L) : \{0, 1\}^l \rightarrow \{0, 1\}^L$, $Func(l, l) : \{0, 1\}^l \rightarrow \{0, 1\}^l$ and $Func(l, T) : \{0, 1\}^l \rightarrow \{0, \ldots, S-1\}$ with $T = log_2 S$.

In the same way as in game $EX_F^b$, games in Table 1 can measure the ability for an adversary $A$ to distinguish between two functions depending on the configuration of the bits $b$ and $n$. Moreover, in each game $A$ performs $q$ queries of values $t_i$, $\{t_0, t_1, \ldots, t_{q-1}\}$, to an oracle implementing a function $g$. Adversary $A$ receives from the oracle $q$ output values $g(t_i)$ and returns a result bit as in $EX_F^b$.

### B. GAMES B AND D

Games $E_{F'}^n(A)$ and $E_F^n(A)$ are games B and D respectively in Table 1, they are only configured with the configuration bit $n$, and at the beginning of both a random key $K$ is obtained, such that $K \xleftarrow{\$} \{0, 1\}^k$.

In game B, $E_{F'}^n(A)$, adversary $A$ tries to distinguish between functions $F_K\_MOD$ and $Func\_MOD$. $F_K\_MOD$ is defined as in Section II, where $F_K\_MOD(x) = F_K(x) \, mod \, S$, and $F_K$ is the PRF configured with the random key $K$. $Func\_MOD$ is a function obtained after applying modulo-$S$ operation to the output of the function $f$, such that $Func\_MOD(x) = f(x) \, mod \, S$. Function $f$ is selected randomly from the set of functions $Func(l, L)$.

In game D, $E_F^n(A)$, adversary $A$ tries to distinguish between functions $E_K\_MOD$ and $Func'\_MOD$. $E_K\_MOD$ is defined as in Section II, where $E_K\_MOD(x) = \left(E_K(x) \, mod 2^L\right)$, and $E_K$ is the PRF function that represents the block cipher of Fig. 2b configured with the random key $K$. $Func'\_MOD$ is a function obtained after applying a modulo-$2^L$ operation to the output of a function $f$ selected randomly from the set of functions $Func(l, l)$.

As summary, in $E_{F'}^n(A)$, adversary $A$ tries to distinguish between a PRF and a random function with different input and output bit lengths, both followed by a modulo-$S$ operation, while in $E_F^n(A)$ adversary $A$ tries to distinguish between a PRF and a random function with the same input and output bit lengths, both followed by a modulo-$2^L$ operation.

### C. GAMES A AND C

Games $E_{F'}^{nb}(A)$ and $E_F^{nb}(A)$ are games A and C respectively, they are configured with the mode bit $n$ and the configuration bit $b$, and at the beginning of both a random key $K$ is obtained, such that $K \xleftarrow{\$} \{0, 1\}^k$.

In $E_{F'}^{nb}(A)$ adversary $A$ will try to distinguish between a random function selected from the set of functions $Func(l, T)$ and a function $r$ that depends on the value of the mode bit $n$. When $n = 1$, this function is configured as $r = F_K\_MOD$ as defined in Section II, while with $n = 0$, $r = Func\_MOD$ that is defined as in the previous subsection.

In $E_F^{nb}(A)$ adversary $A$ will try to distinguish between a random function selected from the set of functions $Func(l, L)$ and a function $r$ that depends on the value of the mode bit $n$. With $n = 1$, $r = E_K\_MOD$ as defined in Section II, and when $n = 0$, $r$ is configured as $r = Func'\_MOD$ as defined in the previous subsection.

## APPENDIX C
## PROOF OF LEMMA 1

According to game A in Table 1 and the definition of the prf-advantage in (19), we can define the following prf-advantages of adversary $A$ over $F_K\_MOD$ and $Func\_MOD$:

$$ADV^{PRF}_{F_K\_MOD}(A) = P\left[E^{11}_{F'}(A) = 1\right] - P\left[E^{10}_{F'}(A) = 1\right] \tag{20}$$

$$ADV^{PRF}_{Func\_MOD}(A) = P\left[E^{01}_{F'}(A) = 1\right] - P\left[E^{00}_{F'}(A) = 1\right] \tag{21}$$

where $P\left[E^{nb}_{F'}(A) = 1\right]$ is the probability that the result of the game $E^{nb}_{F'}(A)$ is $\hat{b} = 1$. In the case of (20) we are measuring the advantage of $A$ over $F_K\_MOD$ as we are performing the game $E^{1b}_{F'}(A)$, and in this case $A$ tries to distinguish between the PRF $F_K\_MOD$ and a random function taken from the set of functions $Func(l, T)$. Both $F_K\_MOD$ and $Func(l, T)$ map values between $\{0, 1\}^l$ and $\{0, \ldots, S-1\}$. In the case of (21) we are measuring the advantage of $A$ over $Func\_MOD$ as we are performing game $E^{0b}_{F'}(A)$ where $A$ tries to distinguish between $Func\_MOD$ and a random function taken from $Func(l, T)$.

According to Table 1, if $b = 0$ the input argument for adversary $A$ is always the same in games $E^{00}_{F'}(A)$ and $E^{10}_{F'}(A)$, therefore $P\left[E^{00}_{F'}(A) = 1\right] = P\left[E^{10}_{F'}(A) = 1\right]$. Then it is possible to derive the following expression from (20) and (21):

$$\begin{aligned} &ADV^{PRF}_{F_K\_MOD}(A) - ADV^{PRF}_{Func\_MOD}(A) \\ &\quad = P\left[E^{11}_{F'}(A) = 1\right] - P\left[E^{01}_{F'}(A) = 1\right] \end{aligned} \tag{22}$$

As the inputs for adversary $A$ in games $E^{n1}_{F'}(A)$ and $E^{n}_{F'}(A)$ are the same then $P\left[E^{n}_{F'}(A) = 1\right] = P\left[E^{n1}_{F'}(A) = 1\right]$, which means that:

$$\begin{aligned} &ADV^{PRF}_{F_K\_MOD}(A) - ADV^{PRF}_{Func\_MOD}(A) \\ &\quad = P\left[E^{1}_{F'}(A) = 1\right] - P\left[E^{0}_{F'}(A) = 1\right] \end{aligned} \tag{23}$$

In Fig. 11a, a graphical diagram for game $E^n_{F'}(A)$ is shown. Adversary $A$ makes $q$ queries to the oracle that implements the function $F_K\_MOD$ or $Func\_MOD$ depending on the value of configuration bit $n$. The diagram in Fig. 11a can be represented also as in Fig. 11b. As both schemes are equivalent from the point of view of adversary $A$ we can conclude that it is possible to build an adversary $C$ from $A$ that produces thesame result bit as $A$. Moreover, as the role of adversary $C$ in Fig. 11b is the same as the role of adversary $A$ in Fig. 10 when performing game $EX^b_F(A)$, (23) can be rewritten as:

$$\begin{aligned} &ADV^{PRF}_{F_K\_MOD}(A) - ADV^{PRF}_{Func\_MOD}(A) \\ &= P\left[E^{1}_{F'}(A) = 1\right] - P\left[E^{0}_{F'}(A) = 1\right] \\ &= P\left[EX^{1}_{F}(C) = 1\right] - P\left[EX^{0}_{F}(C) = 1\right] \\ &= ADV^{PRF}_{F_K}(C) \end{aligned} \tag{24}$$

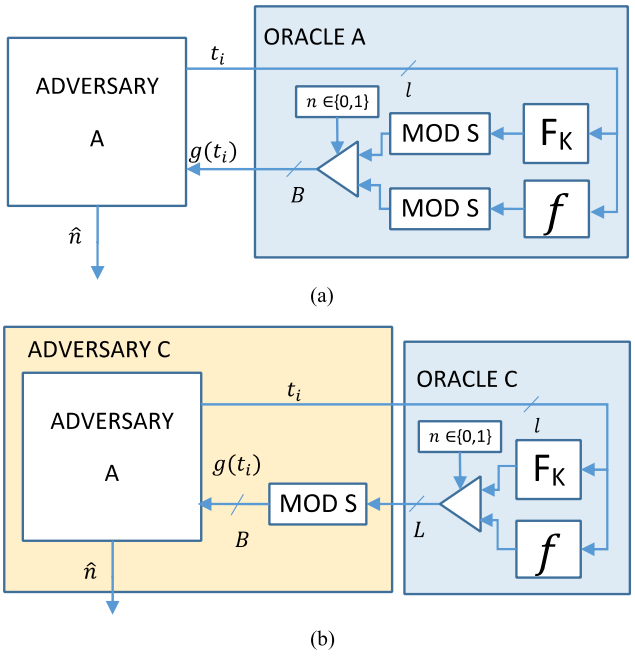



**FIGURE 11.** Diagram of (a) experiment $E^n_{F'}(A)$ and (b) its equivalent using the adversary C. In both cases adversary A generates the same amount of queries to the oracles. In each query A sends ti word represented with l bits and receives g(ti) symbol, represented with $B = \lceil \log_2 S \rceil$ bits. At the end, adversary A produces the result bit $\hat{n}$.

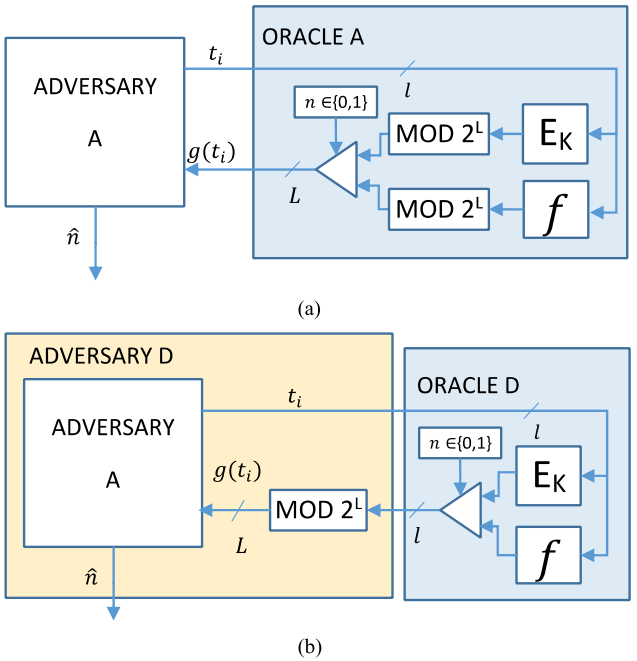



**FIGURE 12.** Diagram of (a) experiment $E^n_F(A)$ and (b) its equivalent using the adversary D. In both cases adversary A generates the same amount of queries to the oracles. In each query A sends $t_i$ word and receives $g(t_i)$ symbol. At the end, adversary A tries to guess the configuration bit $\hat{n}$.

As defined in Section II, $F_K(x) = E_K\_MOD(x)$, which means that:

$$ADV^{PRF}_{F_K\_MOD}(A) = ADV^{PRF}_{E_K\_MOD}(C) + ADV^{PRF}_{Func\_MOD}(A) \tag{25}$$

On the other hand, thanks to game C in Table 1, it is possible to define the following advantages:

$$ADV^{PRF}_{E_K\_MOD}(A) = P\left[E^{11}_F(A) = 1\right] - P\left[E^{10}_F(A) = 1\right] \tag{26}$$

$$ADV^{PRF}_{Func'\_MOD}(A) = P\left[E^{01}_F(A) = 1\right] - P\left[E^{00}_F(A) = 1\right] \tag{27}$$

where $P\left[E^{nb}_F(A) = 1\right]$ is the probability that the result of the game $E^{nb}_F(A)$ is $\hat{b} = 1$. In the case of (26) we are measuring advantage of $A$ over $E_K\_MOD$, as we are performing game $E^{1b}_F(A)$, and in this case adversary $A$ tries to distinguish between $E_K\_MOD$ and a random function taken from the set of function $Func(l, L)$. In the case of (27), as we are performing game $E^{0b}_F(A)$ the adversary $A$ tries to distinguish between $Func'\_MOD$ and a random function taken from $Func(l, L)$, which is equivalent to measure the advantage over $Func'\_MOD$.

According to Table 1, if $b = 0$ the input argument for adversary $A$ is always the same in games $E^{00}_F(A)$ and $E^{10}_F(A)$, therefore $P\left[E^{00}_F(A) = 1\right] = P\left[E^{10}_F(A) = 1\right]$. Then, as $E^{00}_F(A) = E^{10}_F(A)$ and $E^{n1}_F(A) = E^n_F(A)$ it is possible to derive (28) from (26) and (27).

$$\begin{aligned} &ADV^{PRF}_{E_K\_MOD}(A) - ADV^{PRF}_{Func'\_MOD}(A) \\ &\quad = P\left[E^{11}_F(A) = 1\right] - P\left[E^{01}_F(A) = 1\right] \\ &\quad = P\left[E^{1}_F(A) = 1\right] - P\left[E^{0}_F(A) = 1\right] \end{aligned} \tag{28}$$

In Fig. 12a, game $E^n_F(A)$ is shown graphically. As Fig. 12a and Fig. 12b are equivalent from the point of view of adversary $A$, it is possible to build an adversary $D$ from $A$ that produces the same return bit than $A$. In addition, the role of adversary $D$ in Fig. 12b is the same as the role of adversary $A$ in Fig. 10 when performing game $EX^b_F(A)$. Then equation (28) results as:

$$\begin{aligned} &ADV^{PRF}_{E_K\_MOD}(A) - ADV^{PRF}_{Func'\_MOD}(A) \\ &\quad = P\left[E^{1}_F(A) = 1\right] - P\left[E^{0}_F(A) = 1\right] \\ &\quad = P\left[EX^{1}_F(D) = 1\right] - P\left[EX^{0}_F(D) = 1\right] \\ &\quad = ADV^{PRF}_{E_K}(D) \end{aligned} \tag{29}$$

Therefore:

$$ADV^{PRF}_{E_K\_MOD}(A) = ADV^{PRF}_{E_K}(D) + ADV^{PRF}_{Func'\_MOD}(A) \tag{30}$$

According to (30), equation (25) can be written as:

$$\begin{aligned} ADV^{PRF}_{F_K\_MOD}(A) = ADV^{PRF}_{E_K}(D) + ADV^{PRF}_{Func'\_MOD}(A) \\ + ADV^{PRF}_{Func\_MOD}(A) \end{aligned} \tag{31}$$

which is the same expression as in equation (15) and then proves Lemma 1.

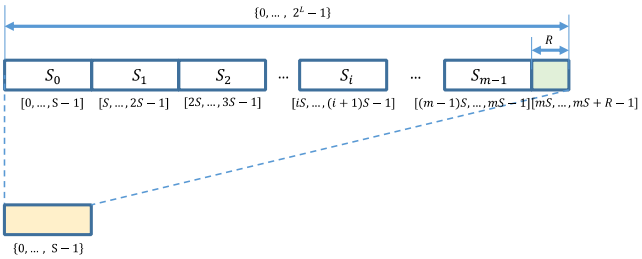



**FIGURE 13.** Domain $\{0, \ldots, 2^L - 1\}$ is mapped into interval $\{0, \ldots, S-1\}$. Each interval Si is mapped in the range $\{0, \ldots, S-1\}$. As $2^L$ is not a multiple of S the last green interval is called $S_{last}$ and it contains only R values that are mapped to the range $\{0, \ldots, R-1\}$.

## APPENDIX D
## PROOF OF LEMMA 2

To prove the Lemma 2 it is necessary to obtain the term $ADV^{PRF}_{Func\_MOD}(A)$, that depends on game $E^{01}_{F'}(A)$ and $E^{00}_{F'}(A)$ as shown in (21). This term measures the capability of $A$ for distinguishing the function $Func\_MOD(x) = f(x)\, mod S$, where $f$ is a random function taken from the set of functions $Func(l, L)$, from a random function from the set $Func(l, T)$, where $T = log_2 S$ and $S$ is not a power of two. The function $f$ maps values from the domain $\{0, \ldots, 2^l - 1\}$ to $\{0, \ldots, 2^L - 1\}$. By applying modulo-$S$ operation to the outputs of function $f$, values in the domain $\{0, \ldots, 2^L - 1\}$ are mapped to domain $\{0, \ldots, S-1\}$. Therefore function $Func\_MOD$ will map values from the domain $\{0, \ldots, 2^l - 1\}$ to $\{0, \ldots, S-1\}$, as random functions in the set $Func(l, T)$ do. In Fig. 13, the way of mapping values from $\{0, \ldots, 2^L - 1\}$ to $\{0, \ldots, S-1\}$ thanks to modulo-$S$ operation is shown. As $S$ is not a power of two, $2^L$ is not a multiple of $S$. Then the remainder $R$ of the division between $2^L$ and $S$ can be written as:

$$R = 2^L - m \cdot S \tag{32}$$

where $m$ is the quotient of the division.

According to (32) the range $\{0, \ldots, 2^L - 1\}$ can be divided in $m$ blocks with $S$ values each one and one last with only $R$ values, as shown in Fig. 13. The $m$ blocks are called $S_i$, with $0 \le i \le m-1$ and the last one with $R$ values is called $S_{last}$. After modulo-$S$ operation, values in each range $S_i$ will be equally distributed in destination range $\{0, \ldots, S-1\}$, however the last range $S_{last}$ includes the last $R$ values in the range $\{m \cdot S, \ldots, m \cdot S + R - 1\}$ and they only are mapped in the first $R$ values of the destination range, $\{0, \ldots, R-1\}$. It will make that each value in the destination range $\{0, \ldots, R-1\}$ can have one more occurrence than values in the range $\{R, \ldots, S-1\}$. Therefore, if each input of modulo-$S$ operation were uniformly distributed in the range $\{0, \ldots, 2^L - 1\}$ it will be possible to consider that its output will not be uniformly distributed, as $R$ is not zero. It means that the output of modulo-$S$ operation would have the following probability distribution:

$$P(x) = \begin{cases} \dfrac{m+1}{2^L} & for\ 0 \le x \le R-1 \\ \dfrac{m}{2^L} & for\ R \le x \le S-1 \end{cases} \tag{33}$$

where $P(x)$ is the probability of occurrence of value $x$ at the output of the modulo-$S$ operation.

During game $E^{0b}_{F'}(A)$, the adversary $A$ makes $q$ queries and they are not repeated. The queries are made to an oracle performing function $g$, that can be configured thanks to $b$ value as $g = Func\_MOD$ or taken randomly from the set $Func(l, T)$.

In $E^{0b}_{F'}(A)$, $f \in Func(l, L)$ and it is a random function. According to [24] if the inputs of a random function $f$ are not repeated we can consider that each output of $f$ is randomly and uniformly distributed in its output range, which in this case is $\{0, \ldots, 2^L - 1\}$, independently of anything else. The same assumption can be done with the case of a random function from the set $Func(l, T)$ we can consider that each of its output is random and uniformly distributed in the range $\{0, \ldots, S-1\}$.

In the case of game $E^{00}_{F'}(A)$, as $g$ is a random function taken from the set $Func(l, T)$, the final output from the oracle can be considered random and uniformly distributed in $\{0, \ldots, S-1\}$.

However, in game $E^{01}_{F'}(A)$, the oracle performs as function $g$ the function $Func\_MOD(x) = f(x)\, mod S$. As $f$ is a random function with a uniform distributed output in the range $\{0, \ldots, 2^L - 1\}$, the output of $Func\_MOD$ will have a probability distribution as in (33) owing to modulo-$S$ operation.

Let us consider two situations when performing game $E^{01}_{F'}(A)$. The first of them is when during the $q$ queries made by adversary $A$, at the output of $f \in Func(l, L)$ no value falls in the range $S_{last}$, but in any of the remainng $S_i$ intervals. In this case the output of modulo-$S$ could be considered randomly and uniformly distributed in the range $\{0, \ldots, S-1\}$, which is a situation indistinguishable from the game $E^{00}_{F'}(A)$, where adversary analyzes the outputs from $g \in Func(l, T)$. Let us name this situation 'Good interval' or *Gint*.

The second case is the opposite, when during the $q$ queries performed in $E^{01}_{F'}(A)$, some output of $f$ falls in the range $S_{last}$, then we can consider that this result could make us perceive the behavior of $Func\_MOD$ no to be like a random function taken from $Func(l, T)$. Due to that it is possible to consider that, at least, a good adversary could have a chance to know that we are running $E^{01}_{F'}(A)$ instead of $E^{00}_{F'}(A)$. Let's name this situation 'Bad interval' or *Bint*.

If we call $w1$ to the event $E^{01}_{F'}(A) = 1$ and $w0$ to the event $E^{00}_{F'}(A) = 1$ then:

$$\begin{aligned} \mathrm{P}[w1] = {} & \mathrm{P}[w1|Gint] \cdot \mathrm{P}[Gint] \\ & + \mathrm{P}[w1|Bint] \cdot \mathrm{P}[Bint] \end{aligned} \tag{34}$$

where $P[Gint]$ and $P[Bint]$ are the probabilities of *Gint* and *Bint* to occur during the $q$ queries made by the adversary in game $E^{01}_{F'}(A)$, respectively.

As the adversary cannot distinguish the situation *Gint* during game $E^{01}_{F'}(A)$ from game $E^{00}_{F'}(A)$ because the output from the oracle in both situations can be considered random and uniformly distributed, then:

$$P\left[E^{00}_{F'}(A)=1\right]=P\left[E^{01}_{F'}(A)=1|Gint\right]\rightarrow P[w0]=P[w1|Gint] \quad (35)$$

In addition, by taking into account (34) and (35) we can rewrite (21) as:

$$\begin{aligned}ADV^{PRF}_{Func\_MOD}(A)&=P\left[E^{01}_{F'}(A)=1\right]-P\left[E^{00}_{F'}(A)=1\right]\\&=P[w1]-P[w0]=P[w1]-P[w1|Gint]=\\&P[w1|Gint]\cdot(P[Gint]-1)+P[w1|Bint]\cdot P[Bint]\end{aligned} \quad (36)$$

Moreover, as $P[Gint]=1-P[Bint]$ then (36) can be rewritten as:

$$ADV^{PRF}_{Func\_MOD}(A)=(\mathrm{P}[w1|Bint]-\mathrm{P}[w1|Gint])\cdot \mathrm{P}[Bint] \quad (37)$$

To get an upper bound for $ADV^{PRF}_{Func\_MOD}(A)$ we assume the worst case where a perfect adversary is able to always detect the function *Func_MOD* when running game $E^{01}_{F'}(A)$ and situation *Bint* is produced, which means that it always gives as result $E^{01}_{F'}(A)=1$. It means that with this perfect adversary $P[w1|Bint]=1$. Therefore (37) can be rewritten as:

$$ADV^{PRF}_{Func\_MOD}(A)=(1-\mathrm{P}[w1|Gint])\cdot \mathrm{P}[Bint]\leq \mathrm{P}[Bint] \quad (38)$$

The probability for an output value of the random function $f\in Func(l,L)$ of falling into the range $S_{last}$, is equal to $R/2^L$, as this range has $R$ values among the $2^L$ possible. As $q$ queries are performed by the adversary $A$ during game $E^{01}_{F'}(A)$, the probability for any of the $q$ oracle outputs of falling in $S_{last}$ is $\mathrm{P}[Bint]=q\cdot R/2^L$. Therefore (38) can be expressed as:

$$ADV^{PRF}_{Func\_MOD}(A)\leq q\cdot R/2^L \quad (39)$$

By taking into account equation (32), as $I=L-T$, $T=log_2 S$ and $m=\lfloor 2^L/S\rfloor$, $R$ can be expressed as:

$$\begin{aligned}R&=2^L-m\cdot S=\\&2^L-\left\lfloor\tfrac{2^L}{2^T}\right\rfloor\cdot 2^T=2^L\cdot\left(1-\left\lfloor\tfrac{2^L}{2^T}\right\rfloor\cdot\tfrac{2^T}{2^L}\right)=\\&2^L\cdot\left(1-\tfrac{\lfloor 2^I\rfloor}{2^I}\right)\end{aligned} \quad (40)$$

Therefore (39) can be rewritten as:

$$\begin{aligned}ADV^{PRF}_{Func\_MOD}(A)&\leq q\cdot R/2^L=q\cdot\left(1-\frac{\lfloor 2^I\rfloor}{2^I}\right)\\&\leq q\cdot\left(1-\frac{2^I-1}{2^I}\right)=\frac{q}{2^I}\end{aligned} \quad (41)$$

which corresponds with the equation (16) and finishes the proof of Lemma 2.

## APPENDIX E
## PROOF OF LEMMA 3

Regarding the term $ADV^{PRF}_{Func'\_MOD}(A)$, it is possible to make a similar reasoning to that for $ADV^{PRF}_{Func\_MOD}(A)$. In *Func'_MOD* the output of the random function $f\in Func(l,l)$ is subjected to a modulo-$2^L$ operation. Thanks to this operation the output range of $f$, $\{0,\ldots,2^l-1\}$, is mapped to the space $\{0,\ldots,2^L-1\}$. This case differs from the situation of Lemma 2, when analyzing *Func_MOD*, where the space $\{0,\ldots,2^L-1\}$ is mapped to the range $\{0,\ldots,S-1\}$.

In the proof of Lemma 2, the output interval of $f\in Func(l,L)$ is $\{0,\ldots,2^L-1\}$. It is divided in $m$ subintervals $S_i$ with $S$ values each one, and one last subinterval $S_{last}$ with the $R$ remaining values. Each subinterval $S_i$ is mapped to $\{0,\ldots,S-1\}$ after modulo $S$ operation while $S_{last}$ is mapped to $\{0,\ldots,R-1\}$. This concludes in equation (39) obtaining $ADV^{PRF}_{Func\_MOD}(A)$ as a function of $R$, $ADV^{PRF}_{Func\_MOD}(A)\leq q\cdot R/2^L$.

In the case of *Func'_MOD* we could make the same reasoning as for Lemma 2, it is possible to divide the output range of $f\in Func(l,l)$, $\{0,\ldots,2^l-1\}$, into $m$ intervals $S_i$ with $2^L$ values each one and one last subinterval $S_{last}$ with the $R'$ remaining values, where $R'$ is the remainder of the division between $2^l$ and $2^L$. Then it is possible to derive an expression similar to (39) for $ADV^{PRF}_{Func'\_MOD}(A)$:

$$ADV^{PRF}_{Func'\_MOD}(A)\leq q\cdot R'/2^l \quad (42)$$

In Lemma 2, $S$ is not a power of two and $2^L$ is not a multiple of $S$, then it was possible to deduce that $R=2^L-m\cdot S$, however in this case as $2^l$ is a multiple of $2^L$ the remainder of its division is zero, that is $R'=0$.

It means that $ADV^{PRF}_{Func'\_MOD}(A)=0$, which is the same expression as (17) and therefore it proves Lemma 3.

## REFERENCES

[1] A. Lenk, P. Marcus, and I. Povoa, "GeoFPE: Format preserving encryption of geospatial data for the Internet of Things," in *Proc. IEEE Int. Congr. Internet Things (ICIOT)*, Jul. 2018, pp. 172–175.

[2] K. Kim and S.-S. Lee, "Encoding of Korean characters with less radix in format-preserving encryption," in *Proc. Int. Conf. Inf. Commun. Technol. Converg. (ICTC)*, Oct. 2015, pp. 1075–1077.

[3] S. Liang, Y. Zhang, J. Guo, C. Dong, Z. Liu, and C. Jia, "Efficient format-preserving encryption mode for integer," in *Proc. 22017 IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Computing (EUC)*, Jul. 2017, pp. 96–102.

[4] *Guidelines for implementing and using the NBS Data Encryption Standad*, National Bureau of Standards USA. Standard FIPS PUB 74, 1981.

[5] M. Brightwell and H. Smith, "Using datatype-preserving encryption to enhance data warehouse security," in *Proc. 20th Nat. Inf. Syst. Secur. Conf.(NISSC)*, 1997, pp. 141—149.

[6] P. Rogaway, "A synopsis of format-preserving encryption," Voltage Secur., Cupertino, CA, USA, Mar. 2010. [Online]. Available: https://web.cs.ucdavis.edu/~rogaway/papers/synopsis.pdf

[7] M. Dworkin, *Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption*. Gaithersburg, MD, USA: NIST, 2016.

[8] M. Bellare, P. Rogaway, and T. Spies. (Sep. 2010). *Addendum to' The FFX Mode of Operation for Format-Preserving Encryption*. [Online]. Available: https://csrc.nist.gov/projects/block-cipher-techniques/bcm

[9] E. Brier, T. Peyrin, and J. Stern. *BPS: A Format-Preserving Encryption Proposal*. Accessed: Jan. 1, 2020. [Online]. Available: http://csrc.nist.gov/groups/ST/toolkit/BCM/documents/proposedmodes/bps/bps-spec.pdf
[10] P. Chandrashekar, S. Dara, and V. N. Muralidhara, "Efficient format preserving encrypted databases," in *Proc. IEEE Int. Conf. Electron., Comput. Commun. Technol. (CONECCT)*, Jul. 2015, pp. 1–4.
[11] B. Cui, B. Zhang, and K. Wang, "A data masking scheme for sensitive big data based on format-preserving encryption," in *Proc. IEEE Int. Conf. Comput. Sci. Eng. (CSE) IEEE Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Jul. 2017, pp. 518–524.
[12] P. Wang, H. Luo, and J. Liu, "Format-preserving encryption for Excel," in *Proc. IEEE Int. Conf. Consum. Electron.-Taiwan (ICCE-TW)*, May 2016, pp. 1–2.
[13] R. Agbeyibor, J. Butts, M. Grimaila, and R. Mills, "Evaluation of format-preserving encryption algorithms for critical infrastructure protection," in *Critical Infrastructure Protection VIII*. Heidelberg, Germany: Springer, 2014, pp. 245–261.
[14] I. Oh, T. Kim, K. Yim, and S.-Y. Lee, "A novel message-preserving scheme with format-preserving encryption for connected cars in multi-access edge computing," *Sensors*, vol. 19, no. 18, p. 3869, Sep. 2019.
[15] H. Sun, H. Luo, and Y. Sun, "Data hiding for ensuring the quality of the host image and the security of the message," *IEEE Access*, vol. 7, pp. 64767–64777, 2019.
[16] K. Kim and K.-Y. Chang, "Performance analysis of format-preserving encryption based on unbalanced-feistel structure," in *Advances in Computer Science and Ubiquitous Computing* (Lecture Notes in Electrical Engineering), vol. 373. Springer: Singapore, 2015, pp. 425–430.
[17] K. Mallaiah, S. Ramachandram, and S. Gorantala, "Performance analysis of Format Preserving Encryption (FIPS PUBS 74-8) over block ciphers for numeric data," in *Proc. 4th Int. Conf. Comput. Commun. Technol.(ICCCT)*, Sep. 2013, pp. 193–198.
[18] M. Li, Z. Liu, J. Li, and C. Jia, "Format-preserving encryption for character data," *J. Netw.*, vol. 7, no. 8, pp. 1239–1244, 2012.
[19] T. W. Arnold, M. Check, E. A. Dames, J. Dayka, S. Dragone, D. Evans, W. S. Fernandez, M. D. Hocker, R. Kisley, T. E. Morris, J. Petreshock, and K. Werner, "The next generation of highly reliable and secure encryption for the IBM z13," *IBM J. Res. Dev.*, vol. 59, no. 4/5, pp. 6:1–6:13, Jul. 2015.
[20] A. Perez-Resa, M. Garcia-Bosque, C. Sanchez-Azqueta, and S. Celma, "Physical layer encryption for industrial ethernet in gigabit optical links," *IEEE Trans. Ind. Electron.*, vol. 66, no. 4, pp. 3287–3295, Apr. 2019.
[21] A. Perez-Resa, M. Garcia-Bosque, C. Sanchez-Azqueta, and S. Celma, "Chaotic encryption applied to optical ethernet in industrial control systems," *IEEE Trans. Instrum. Meas.*, vol. 68, no. 12, pp. 4876–4886, Dec. 2019.
[22] P. Rogaway, "Evaluation of some blockcipher modes of operation," Cryptogr. Res. Eval. Committees, Government Japan, Tokyo, Japan, 2011. [Online]. Available: https://crossbowerbt.github.io/docs/crypto/rogaway_modes.pdf
[23] M. Bellare, A. Desai, E. Jokipii, and P. Rogaway, "A concrete security treatment of symmetric encryption," in *Proc. 38th Annu. Symp. Found. Comput. Sci.*, Oct. 1997, pp. 394—403.
[24] M. Bellware and P. Rogaway. (May 2005). (Introduction to Modern Cryptography). [Online]. Available: http://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf
[25] M. Bellware and P. Rogaway, "Indistinguishably under chosen-plaintext attack," in *Introduction to Modern Cryptography*. May 2005, ch. 5–4. [Online]. Available: https://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf
[26] M. P. Fok, Z. Wang, Y. Deng, and P. R. Prucnal, "Optical layer security in fiber-optic networks," *IEEE Trans. Inf. Forensic Security*, vol. 6, no. 3, pp. 725–736, Sep. 2011.
[27] J. Ji, G. Zhang, W. Li, L. Sun, K. Wang, and M. Xu, "Performance analysis of physical-layer security in an OCDMA-based wiretap channel," *J. Opt. Commun. Netw.*, vol. 9, no. 10, p. 813, Oct. 2017.
[28] J. Hizanidis, S. Deligiannidis, A. Bogris, and D. Syvridis, "Enhancement of chaos encryption potential by combining all-optical and electrooptical chaos generators," *IEEE J. Quantum Electron.*, vol. 46, no. 11, pp. 1642–1649, Nov. 2010.
[29] A. Sultan, X. Yang, A. A. E. Hajomer, S. B. Hussain, and W. Hu, "Dynamic QAM mapping for physical-layer security using digital chaos," *IEEE Access*, vol. 6, pp. 47199–47205, 2018.
[30] Y. Gao, S. Hu, W. Tang, Y. Li, Y. Sun, D. Huang, S. Cheng, and X. Li, "Physical layer security in 5g based large scale social networks: Opportunities and challenges," *IEEE Access*, vol. 6, pp. 26350–26357, 2018.
[31] H. Rahbari and M. Krunz, "Full frame encryption and modulation obfuscation using channel-independent preamble identifier," *IEEE Trans.Inf. Forensic Security*, vol. 11, no. 12, pp. 2732–2747, Dec. 2016.
[32] K. Guan, J. Kakande, and J. Cho, "On deploying encryption solutions to provide secure transport-as-a-service (TaaS) in core and metro networks," in *Proc. 42nd Eur. Conf. Opt. Commun.*, Düsseldorf, Germany, Sep. 2016, pp. 1–3.
[33] J. Daemen and V. Rijmen, *The Design of Rijndael, AES—The Advanced Encryption Standard*. Berlin, Germany: Springer-Verlag, 2002.
[34] J. T. Butler and T. Sasao, "Fast hardware computation of x mod z," in *Proc. IEEE Int. Symp. Parallel Distrib. Process. Workshops Phd Forum*, Shanghai, China, May 2011, pp. 294–297.
[35] A. Klein, *Stream Ciphers*, London, U.K.: Springer-Verlag, 2013.

**ADRIÁN PÉREZ-RESA** was born in San Sebastián, Spain. He received the M.Sc. degree in telecommunications engineering from the University of Zaragoza, Zaragoza, Spain, in 2005. He is currently pursuing the Ph.D. degree with the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza.

He was an Research and Development Engineer with Telecommunications industry for more than ten years. He is a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include high-speed communications and cryptography applications.

**MIGUEL GARCIA-BOSQUE** was born in Zaragoza, Spain. He received the B.Sc. and M.Sc. degrees in physics from the University of Zaragoza, Zaragoza, in 2014 and 2015, respectively. He is currently pursuing the Ph.D. degree with the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza.

He is a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include chaos theory and cryptography algorithms.

**CARLOS SÁNCHEZ-AZQUETA** was born in Zaragoza, Spain. He received the B.Sc. degree in physics from the University of Zaragoza, Zaragoza, in 2006, the Dipl.-Ing. degree in electronic engineering from the Complutense University of Madrid, Madrid, Spain, in 2009, and the M.Sc. and Ph.D. degrees in physics from the University of Zaragoza, in 2010 and 2012, respectively.

He is currently a member of the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. His research interests include mixed-signal integrated circuits, high-frequency analog communications, and cryptography applications.

**SANTIAGO CELMA** was born in Zaragoza, Spain. He received the B.Sc., M.Sc., and Ph.D. degrees in physics from the University of Zaragoza, Zaragoza, in 1987, 1989, and 1993, respectively.

He is currently a Full Professor with the Group of Electronic Design, Aragón Institute of Engineering Research, University of Zaragoza. He has coauthored more than 100 technical articles and 300 international conference contributions. He is also a coauthor of four technical books. He holds four patents. He appears as the Principal Investigator in more than 30 national and international research projects. His research interests include circuit theory, mixed-signal integrated circuits, high-frequency communication circuits, wireless sensor networks, and cryptography for secure communications.

# Self-synchronized Encryption for Physical Layer in Gigabit Ethernet Optical Links

A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma

***Abstract*—In this work a new self-synchronized symmetric encryption solution for high speed communication systems necessary to preserve the format of the plaintext is proposed, developed and tested. This new encryption mechanism is based on the block cipher operation mode called PSCFB (Pipelined Statistical Cipher Feedback) and the modulo operation. The confidentiality of this mode is analyzed in terms of its IND-CPA (Indistinguishability under Chosen-Plaintext Attack) advantage, concluding that it can be considered secure in the same way as traditional modes are. The encryption system has been integrated in the physical layer of a 1000Base-X Gigabit Ethernet Interface, where the 8b/10b symbol flow is encrypted at line rate. Moreover, to achieve authenticity, integrity, data freshness and key refreshing at physical layer, also a hardware mechanism has been proposed and implemented next to the self-synchronized solution. The implementation have been carried out in an FPGA (Field Programmable Gate Array) device. Finally, an encrypted optical link has been tested with real Ethernet frames, getting maximum throughput and protecting the data traffic from passive and active eavesdroppers.**

***Index Terms*—Gigabit Ethernet, Physical Coding Sublayer, encryption, stream cipher, PSCFB**

## I. Introduction

In recent decades, we have witnessed the rise of broadband networks, mainly thanks to the advance of communication standards in physical media such as optical fiber. Thanks to them it is possible to provide the high data bandwidth demanded by the customers and the market [1]. To achieve information confidentiality maintaining the information throughput, high speed encryption systems must be used.

Encryption methods can be implemented at different levels of a communication system, for example MACsec [2] or IPsec [3], for layer 2 and layer 3, respectively. Regarding to physical level (layer 1), different solutions have been proposed depending on the transmission medium, such as [4], [5] or [6]. In the particular case of optical networks, physical layer it is considered critical to guarantee secure communications [7], [8], [9].

There are several encryption mechanisms for optical networks at physical layer. Some of them are focused on photonics technologies [9], [10], [11], while others are based on the protocols of the optical system such as in [12], where OTN (Optical Transport Network) frame payloads are encrypted at bit level. Other examples have been shown for 1 Gbps and 10 Gbps optical Ethernet standards, where several proposals have been developed for 64b/66b [13] and 8b/10b [14] physical encodings. For the particular case of Gigabit Ethernet, the encryption is performed in the PCS (Physical Coding Sublayer) layer before the 8b/10b encoder. In this way it is possible to preserve the coding properties such as the transition density and short run length, that are necessary to facilitate the operation of the remote CDR (Clock and Data Recovery) circuits [15] at the physical layer. In addition, to preserve the coding properties it is necessary to perform the encryption able to preserve the format of the plaintext, in this case the 8b/10b symbol flow.

This kind of encryption is usually called FPE (Format Preserving Encryption), and although many solutions have been proposed about it [16], the only ones approved by the NIST (National Institute of Standards and Technology) are the FF1 and FF3 modes of operation [17].

As far as the authors are concerned, there are no standardized solutions for FPE self-synchronized stream ciphers, but some proposals have been made. For example in [18], where the self-synchronized PSCFB operation mode was proposed to be used with the underlying FPE block cipher in [14]. An important issue regarded with symmetric encryption stream ciphers is that these must synchronize their keystreams before starting the encryption session. Furthermore, in case of missing the synchronization in the middle of a session an ad-hoc mechanism or protocol must be implemented to recover it. With the use of self-synchronized stream ciphers it is possible to dispense with these extra communication processes.

Unfortunately, self-synchronized solution in [18] uses internally the recommended FF3 scheme that is based on a non-binary Feistel structure formed by several processing stages where an AES block is required in each one of them. This increases the hardware complexity of the resulting FPE self-synchronized solution with respect to other non-FPE ciphers.

On the other hand, the PSCFB mode inherently has an encryption efficiency that can be less than 100% [19]. If $E_K$ is an underlying block cipher working in PSCFB mode, the encryption efficiency represents the number of ciphertext bits that can be produced in PSCFB mode relative to the number of output bits produced by $E_K$ working in a basic CTR (Counter) mode. As consequence of having an efficiency lower than 100% it is necessary to use input and output queues in the PSCFB structure, which increases the latency introduced by the encryption system [19].

This work was supported in party by MINECO-FEDER (TEC2014-52840) and FPU fellowship to M. García Bosque (FPU14/03523).

A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta and S. Celma are with the Electronic and Communications Engineering Department, Zaragoza University, Zaragoza, 50009 Spain (e-mail: {aprz, mgbosque, csanaz, scelma}@unizar.es).

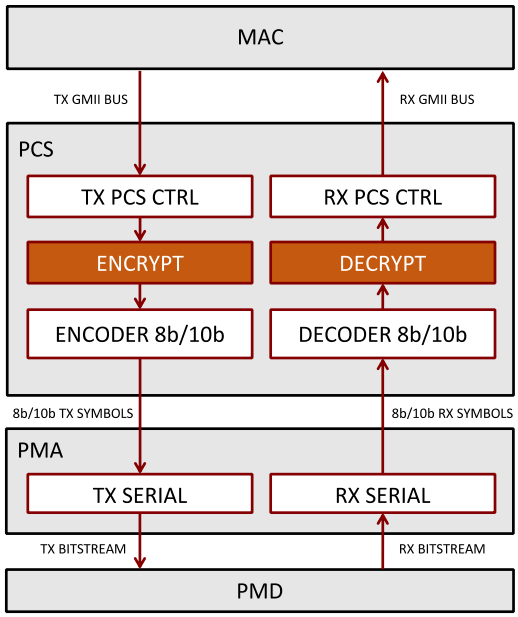



Fig. 1. Scheme of the Ethernet Interface formed by the PHY and MAC modules. MAC layer builds the Ethernet packets transmitted to the PHY. ENCRYPT and DECRYPT modules perform the format preserving encryption/decryption of 8b/10b symbols at the PCS sublayer. TX_SERIAL and RX_SERIAL are the serializer/deserializer modules that transmit and receive the bitstream from the optical link.

In order to reduce the hardware complexity of possible self-synchronous FPE stream ciphers, in this work a new structure able to preserve the format of the plaintext is proposed. It is based on a recommended block cipher working in an operation mode that is a synergic combination of PSCFB and CTR-MOD [20] modes. We have called it PSCFB-MOD.

Since this new proposed operation mode uses as underlying block cipher a recommended binary block cipher instead of an FPE one, as in [14] and [18], it is possible to reduce the hardware complexity introduced by the non-binary Feistel structure. In addition, with this new encryption scheme the input and output queues are not needed which reduces the latency introduced by the encryption system.

By using a recommended underlying block cipher, for instance AES, it is possible to develop a formal security proof, in the same way as in traditional confidentiality-only operation modes, such as CTR or CBC (Cipher Block Chaining). The formal security proof consists of the IND-CPA (Indistinguishability under Chosen Plaintext Attack) advantage expression of any adversary attacking this scheme.

The encryption system has been adapted to work in the 1000Base-X physical layer for Gigabit Ethernet standard, as mentioned before. In addition, other important security issues such as authenticity, integrity, data freshness and key refreshing at physical layer are addressed in this paper. These functionalities next to the self-synchronized solution PSCFB-MOD have been implemented in an FPGA (Field Programmable Gate Array).

The paper is divided in six sections. In Section II an introduction about Gigabit Ethernet standard, CTR-MOD mode and self-synchronized encryption is given. Section III presents the security analysis and the hardware implementation of the proposed PSCFB-MOD scheme and their comparison with the traditional CTR. Subsequently, Section IV describes the physical layer mechanism used to get authenticity, integrity, data freshness and key refreshing. Section V deals with the set-up and encryption results. Finally, in Section VI conclusions are given.

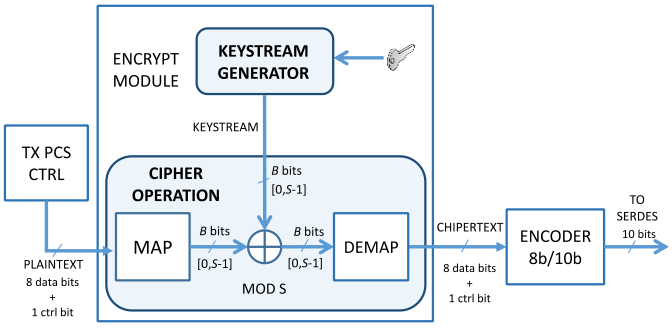



Fig. 2. Location and generic structure of a stream cipher in a physical layer with 8b/10b line encoding. 8b/10b symbols are formed by eight data bits and one control bit. The mapped symbols are represented with $B = \lceil log_2 S \rceil$ bits.

## II. Important Concepts

### A. Optical Gigabit Ethernet Encryption

Although there are no standardized solutions for FPE stream ciphers, its usage could be relevant in communications where a high encryption rate is necessary, as in physical layer 1000Base-X for optical Gigabit Ethernet systems [14].

In [14] encryption is carried out in the PCS sublayer where the 8b/10b encoding is performed. Since the 8b/10b encoding is used to provide important properties to the bitstream it is necessary to preserve the format of the 8b/10b symbols. Therefore, the encryption of a gigabit Ethernet symbol must give as result another valid symbol that must be within group of symbols supported by the standard, which means to perform an FPE encryption.

Encryption and decryption modules are located in the physical layer datapath as shown in Fig. 1, before the 8b/10b encoder/decoder. In this figure an Ethernet Interface is shown. It is composed of the MAC (Medium Access Control) and PHY layers. PHY module includes PCS, PMA (Physical Medium Attachment) and PMD (Physical Medium Dependant) sublayers.

Inside the ENCRYPT module, FPE encryption is performed as shown in Fig. 2. If $S$ is the possible number of different 8b/10b symbols, using the MAP block, each symbol is mapped to an integer value in the range $\{0, \ldots, S-1\}$ to get a plaintext in radix $S$. The plaintext is added modulo-$S$ to a keystream, which is also in radix $S$, to obtain the ciphertext. This ciphertext is then reverse mapped in the DEMAP module to get the final 8b/10b encrypted flow. The ciphered 8b/10b symbols will be encoded to 10-bit values and sent to the serializer.

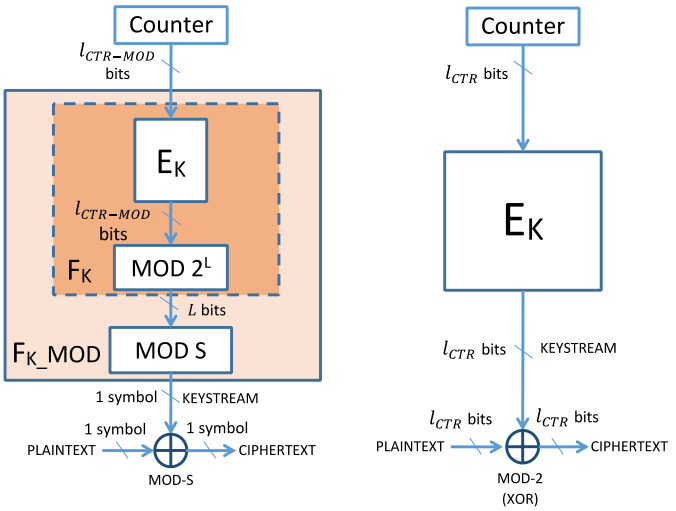



Fig. 3. Structure of (a) CTR-MOD mode and (b) CTR traditional mode.

## B. *CTR-MOD operation mode*

CTR-MOD mode can be considered a kind-of counter mode where the counter is in binary radix with a length of $l_{CTR-MOD}$ bits while the keystream output is a stream of symbols in radix $S$ and it is obtained thanks a function called $F_K\_MOD$. The ciphertext is obtained thanks to the module-$S$ addition between the keystream and the plaintext. A representation of this mode next to the traditional CTR are shown Fig. 3.

$F_K\_MOD$ can be considered a PRF (Pseudo Random Function) such that $F_K\_MOD: \mathcal{K} \times \{0,1\}^{l_{CTR-MOD}} \rightarrow \{0, \dots, S-1\}$. It is configured with a random $K$ taken from the keyspace $\mathcal{K}$ ($K \in \mathcal{K}$). $F_K\_MOD$ is built thanks to a block cipher $E_K$ as shown in Fig. 3a. In this figure the block cipher is modelled as a PRF such that $E: \mathcal{K} \times \{0,1\}^{l_{CTR-MOD}} \rightarrow \{0,1\}^{l_{CTR-MOD}}$, where $l_{CTR-MOD}$ (the counter length in bits) is also its block size. Its least significant $L$ output bits are taken as input of a modulo-$S$ operation, which is the same as performing the modulo-$2^L$ operation at the output of $E_K$. Finally we can represent $F_K\_MOD(x)$ in terms of $E_K$, such that $F_K\_MOD(x) = (E_K(x) mod\ 2^L)\ mod\ S$.

## C. *Self-Synchronized Encryption*

Several mechanisms have been proposed for self-synchronized encryption. On the one hand there are some proposals based on ad-hoc self-synchronized stream ciphers such those proposed in eSTREAM project, SSS and Mosquito [21], although they were dismissed due to their vulnerabilities [22], [23]. On the other hand self-synchronized stream ciphers can be built thanks to block ciphers working in certain operation modes such as CFB (Cipher Feedback), OCFB (Optimized Cipher Feedback) [24], SCFB (Statistical Cipher Feedback) [25] or PSCFB (Pipeline Statistical Cipher Feedback) [19]. Among them, PSCFB can be considered the best in terms of encryption throughput, as it can reach a value near to 100%.

The PSCFB mode of operation is shown in Fig. 4. It is

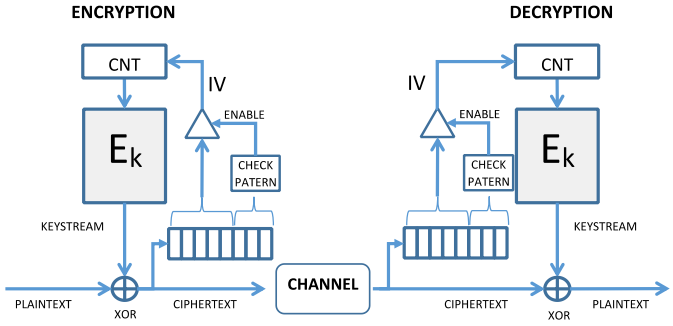



Fig. 4. Structure of PSCFB mode for encryption and decryption. The block cipher $E_K(\cdot)$ has a block size of $l$ bits, and is implemented using $P$ internal stages. CNT block represents the counter of this mode.

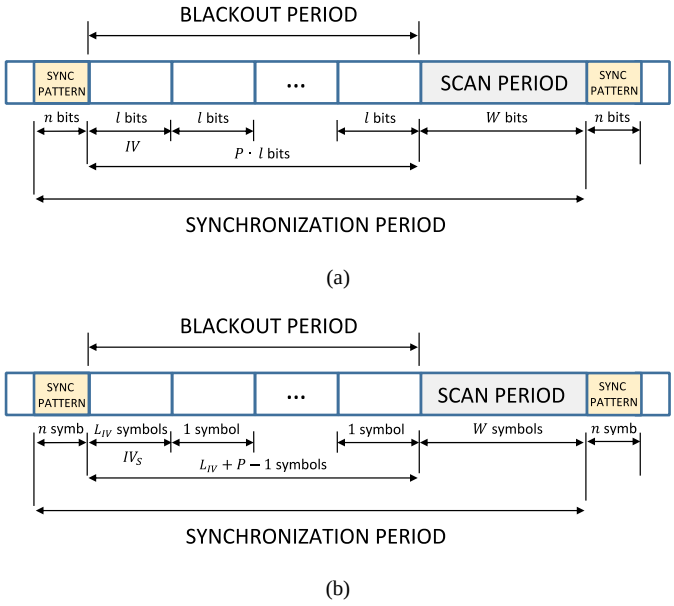



Fig. 5. Structure of the synchronization period in (a) PSCFB mode in binary radix and (b) PSCFB-MOD with radix $S$.

considered a combination of two NIST recommended operation modes CTR and CFB. The underlying block cipher $E_K$ has a block size of $l$ bits and its structure is a pipelined architecture with $P$ stages. It works initially in CTR mode, while the ciphertext is scanned to find a special $n$-bit sequence called synchronization pattern. As it is possible to assume that the output values of $E_K$ working in CTR mode are random and independent, the resulting ciphertext can be considered a random stream after performing the XOR operation between the keystream and the plaintext.

Due to the randomness of the ciphered values, sync pattern will be detected at a statistical random point in the ciphertext. When the pattern is detected, the underlying block cipher do not increment its counter, it captures the next $l$ bits and uses them as $IV$ (Initialization Vector) which feeds back the counter value at the block cipher input, which is the same operation made by a block cipher working in CFB mode. Moreover, as the block cipher has a pipeline architecture with $P$ stages, it is necessary to disable the sync pattern scanning since the $IV$ is captured

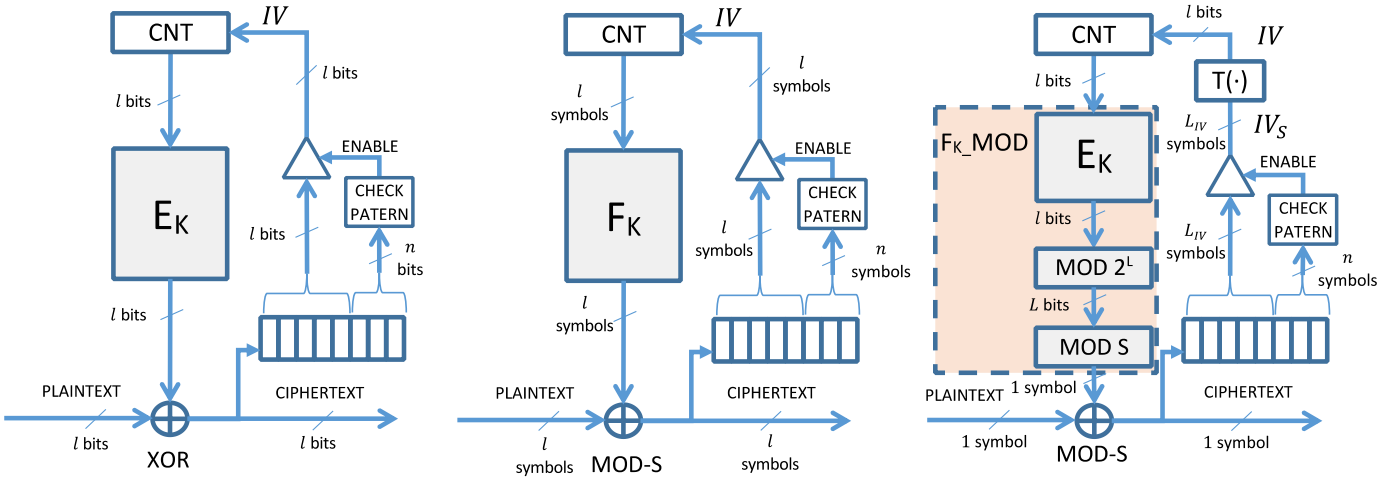



Fig. 6. Structure of (a) PSCFB mode in binary radix, (b) PSCFB mode in radix *S*, and (c) PSCFB-MOD mode. $E_K$ represents the block cipher in binary radix and a block size of *l* bits. $F_K$ represents the FPE block cipher in radix *S* and a block size of *l* symbols. Symbols in cases (b) and (c) are in radix *S*.

until $E_K(IV)$ is available as new keystream block, $P$ cycles later. This interval is called the blackout period and it is formed by the *IV* captured after sync pattern and the next *P*-1 ciphered blocks of *l* bits. In Fig. 5a complete synchronization cycle is shown.

After the counter CNT of Fig. 4 is momentarily loaded with the new *IV*, it continues with its normal operation, which means a continuous increment (as in CTR mode) until a new sync pattern is found in the ciphertext. In that case the load operation of the *IV* is produced again and a new synchronization period starts.

Let us name *W* to the random bit-length of the scan period and $L_{IV}$ to the length of the *IV*, then, as mentioned in [19] the average size in bits of a complete sync period *u* is formed by the length of the sync pattern, $L_{sp} = n$, the blackout period, $L_{bp} = L_{IV} + l \cdot (P-1)$, and the average of the scan period, $L_{scan} = E\{W\}$ until next sync pattern:

$$u = L_{sp} + L_{bp} + L_{scan} = \\ = n + L_{IV} + l \cdot (P-1) + E\{W\} = n + l \cdot P + E\{W\} \quad (1)$$

since $L_{IV} = l$ bits.

## III. PSCFB-MOD OPERATION MODE

### A. PSCFB-MOD Description

To achieve self-synchronization for the encryption of a plaintext in radix *S*, the first approach could be applying PSCFB operation mode using an FPE block cipher instead of a traditional one in binary radix, such as AES. Both PSCFB structures, in binary and non-binary radix, are shown in Fig. 6a and Fig. 6b, respectively.

On the one hand the block cipher $E_K$ in Fig. 6a should be replaced by an FPE block cipher $F_K$, for example such as the FF3 structure built in [14]. On the other hand the XOR operation in Fig. 6a must be replaced by a modulo-*S* addition to generate the ciphertext in radix *S*. These modifications in Fig. 6a results in the structure of Fig. 6b, and it corresponds with the PSCFB mode in radix *S* that was firstly proposed in [18].

In both solutions, after the sync pattern is detected, the *IV* is captured and used to refresh the counter value. This *IV* has a length equals to the input block size *l* and the same radix as the ciphertext and plaintext.

In this work, to reduce the hardware complexity and latency of this first approach, PSCFB-MOD mode is proposed. Instead of using a block cipher, with the same width and radix for its input and output, a PRF $F_K\_MOD$ as described in Section II-B has been used, such that $F_K\_MOD: \mathcal{K} \times \{0,1\}^l \rightarrow \{0, \dots, S-1\}$. The resulting structure is shown in Fig. 6c.

The main difference between this mode and the previous of Fig. 6a and Fig. 6b is that the width and radix at the input of the underlying encryption function $F_K\_MOD$ is different to its output. While its input has a binary radix and an *l*-bit length the output is in radix *S* and it only has a length of one symbol. For this reason if the captured *IV* has a length of $L_{IV}$ symbols in radix *S*, it must be transformed to an *IV* with length of *l*-bits, which means with the same size and radix as the counter and the input of $F_K\_MOD$.

Let us name the captured Initialization Vector in radix *S* as $IV_S$, then the *IV* used to refresh the *l*-bit counter should be obtained from $IV_S$ thanks a function $T(\cdot)$, such that:

$$IV = T(IV_S) \quad (2)$$

The $IV_S$ can be represented by a vector of $L_{IV}$ symbols in radix *S* such that $IV_S = \{IV_{S_0}, IV_{S_1}, \dots, IV_{S_{L_{IV}-1}}\}$. The transformation function $T(\cdot)$ has consisted on two steps. Firstly, the numeral string $IV_S$ in radix *S* is converted to its binary radix representation with the function $NUM(\cdot)$, secondly the result

from this conversion is truncated to its least significant *l* bits to get the *IV* of length *l*. Therefore *T(·)* can be expressed as:

$$IV = T(IV_S) = NUM(IV_S)\ mod\ 2^l \quad (3)$$

where $NUM(IV_S) = \left(\sum_{i=0}^{i<L_{IV}} IV_{S_i} \cdot S^i\right)$.

Other issue that has to be taken into account is the length of the sync pattern. In the case of PSCFB in binary radix in Fig. 6a, this pattern is *n*-bit length. However in PSCFB schemes for Fig. 6b and Fig. 6c, as the radix is *S*, the sync pattern will have a length of *n* symbols. This length will influence in the desired SRD (Synchronization Recovery Delay) for the PSCFB encryption scheme [26], as we discuss in Subsection III-F.

The structure of a synchronization period of PSCFB-MOD is shown in Fig. 5b. As the output block size of $F_K\_MOD$ is one symbol the blackout period will be formed by the $L_{IV}$ symbols captured after sync pattern and the next *P*-1 ciphered symbols. According to this the average length in symbols of a complete sync period *u* is shown in (4) in a similar way as in (1).

$$u = L_{sp} + L_{bp} + L_{scan} = n + L_{IV} + P - 1 + E\{W\} \quad (4)$$

where *W* is the random length in symbols of the scan period.

### *B. IND-CPA Security in CTR and PSCFB modes*

Usually the security of the confidentiality-only operation modes for block ciphers is studied in the sense of IND-CPA (Indistinguishability under Chosen-Plaintext Attack) security [27]. A metric called adversary advantage is obtained thanks to a game between an active adversary *A* and an encryption oracle performing the target encryption scheme $S\mathcal{E}$. This encryption scheme $S\mathcal{E}$ is configured with a key and an experiment bit *b*, which the adversary tries to guess.

During the game the adversary sends to the oracle a sequence of *p* pair of messages $(M_1^0, M_1^1), \dots, (M_p^0, M_p^1)$. Both messages in each pair $(M_i^0, M_i^1)$ have the same length $m_i$. For each query from the adversary, the oracle responds with the ciphertext $C_i$ corresponding to the message $M_i^b$. Finally the adversary will try to guess if the oracle encrypted $(M_1^0, \dots, M_p^0)$ or $(M_1^1, \dots, M_p^1)$, which is the same as guessing the value of *b* after the *p* queries.

To measure the success of the adversary in breaking a symmetric encryption scheme $S\mathcal{E}$, the adversary advantage is defined in [28] as in the following equation:

$$ADV_{S\mathcal{E}}^{IND-CPA}(A) = 2 \cdot Pr\left(\hat{b} = b\right) - 1 \quad (5)$$

where the $ADV_{S\mathcal{E}}^{IND-CPA}(A)$ is the IND-CPA advantage of the adversary *A* over the encryption scheme $S\mathcal{E}$, and $Pr\left(\hat{b} = b\right)$ is the probability of the adversary *A* of guessing the correct value of configuration bit *b*. The advantage of *A* can be understood as the excess of this probability over 1/2. When the 'guess' probability is almost 1/2 and then the adversary advantage is negligible the encryption scheme $S\mathcal{E}$ can be considered secure.

It is demonstrated in [29] that an adversary *B* attacking the PRF security of the underlying block cipher $E_K$ of $S\mathcal{E}$ can be built thanks to the adversary *A*, their advantages are related as follows:

$$ADV_{S\mathcal{E}}^{IND-CPA}(A) = 2 \cdot ADV_{E_K}^{PRF}(B) + ADV_{S\mathcal{E}(Func)}^{IND-CPA}(A) \quad (6)$$

where $ADV_{SE}^{IND-CPA}(A)$ is the advantage of *A* attacking $S\mathcal{E}$ when its underlying encryption function is $E_K$, $ADV_{SE(Func)}^{IND-CPA}(A)$ is the advantage of *A* over $S\mathcal{E}$ when the underlying encryption function is a random function $Func(l,l)$ such that $Func(l,l)$: $\{0,1\}^l \rightarrow \{0,1\}^l$, and $ADV_{E_K}^{PRF}(B)$ is the prf-advantage of any adversary *B* over $E_K$ as defined in [27].

In the formal security proofs of CTR and PSCFB modes expression in (6) can be particularized changing $S\mathcal{E}$ term by CTR and PSCB, respectively:

$$ADV_{CTR}^{IND-CPA}(A) = 2 \cdot ADV_{E_K}^{PRF}(B) + ADV_{CTR(Func)}^{IND-CPA}(A) \quad (7)$$

$$ADV_{PSCFB}^{IND-CPA}(A) = 2 \cdot ADV_{E_K}^{PRF}(B) + ADV_{PSCFB(Func)}^{IND-CPA}(A) \quad (8)$$

where $ADV_{CTR}^{IND-CPA}$ and $ADV_{PSCFB}^{IND-CPA}$ are the IND-CPA advantages expressions of any adversary *A* against CTR and PSCFB encryption modes, respectively, and $ADV_{CTR(Func)}^{IND-CPA}$ and $ADV_{PSCFB(Func)}^{IND-CPA}$ are the advantages over each encryption scheme when the underlying encryption function is a random function $Func(l,l)$. In both security proofs the terms $ADV_{CTR(Func)}^{IND-CPA}(A)$ and $ADV_{PSCFB(Func)}^{IND-CPA}(A)$ are obtained [29], [13], then allowing to reach the final advantage expression of both operation modes.

In [29] it is proven that:

$$ADV_{S\mathcal{E}(Func)}^{IND-CPA}(A) \leq P_{S\mathcal{E}}(col) \quad (9)$$

where $P_{S\mathcal{E}}(col)$ is the probability of a collision among the counter values used during the game, it means the probability of a counter value repetition. According to this, (7) and (8) can be rewritten as:

$$ADV_{CTR}^{IND-CPA}(A) \leq 2 \cdot ADV_{E_K}^{PRF}(B) + P_{CTR}(col) \quad (10)$$

$$ADV_{PSCFB}^{IND-CPA}(A) \leq 2 \cdot ADV_{E_K}^{PRF}(B) + P_{PSCFB}(col) \quad (11)$$

where $P_{CTR}(col)$ and $P_{PSCFB}(col)$ are the collision probability for CTR and PSCFB, respectively.

Since in CTR mode the counter is incremented in each encryption step and it is never repeated, the probability of collision is zero: $P_{CTR}(col) = 0$. However during the IND-CPA game for PSCFB mode it is possible for the counter to be fed with a new *IV* value when the sync pattern is detected randomly at some point of the ciphertext. As this new counter value is a random one, this and the subsequent values of the counter during the next synchronization cycle could produce a collision with the values of the counter in previous cycles, then $P_{PSCFB}(col) \neq 0$. According to this, in [29] and [13] $P_{S\mathcal{E}}(col)$ is obtained, giving rise to the following expressions:

$$ADV_{CTR}^{IND-CPA}(A) \leq 2 \cdot ADV_{E_K}^{PRF}(B) \quad (12)$$

$$ADV_{PSCFB}^{IND-CPA}(A) \leq 2 \cdot ADV_{E_K}^{PRF}(B) + \frac{\mu^2}{l^2 2^l} \cdot \frac{P+1}{P^2} \quad (13)$$

For the specific case of PSCFB the second term $P_{PSCFB}(col)$ is not zero. This term depends on some parameters that define the architecture of the PSCFB scheme, such as the block size *l* and the number of pipeline stages *P* of the block cipher $E_K$, and also on the quantity of information bits $\mu$ encrypted by the oracle during the adversary game. From equations (12) and (13) it is possible to conclude that CTR mode, although it does not have the self-synchronous property, it is inherently more secure than PSCFB when using the same underlying block cipher $E_K$. As the first term $ADV_{E_K}^{PRF}$ in both expressions is the same, the second in CTR is zero which makes IND-CPA advantage in CTR lower than with PSCFB mode.

In the same way as in PSCFB, it is possible to obtain an IND-CPA advantage expression for PSCFB-MOD, in terms of some parameters of its encryption scheme. The idea in this work is to stablish the bounds for these parameters, under which the resulting structure of PSCFB-MOD is better in terms of IND-CPA advantage than an operation mode taken as reference. Particularly, the mode used as reference has been CTR, since, in general, it can be considered the best to achieve confidentiality-only encryption [30].

The expression of the IND-CPA security for PSCFB-MOD mode is given in Subsection III-C, while its proof is shown in Appendix A. The comparison analysis between PSCFB-MOD and CTR is in Subsection III-D.

### C. IND-CPA Security in PSCFB-MOD Mode

It is possible to express the IND-CPA security of PSCFB-MOD mode according to the following theorem:

*Theorem 1*: Let $F_K\_MOD: \mathcal{K} \times \{0,1\}^l \rightarrow \{0, \dots, S-1\}$ be the underlying function of the encryption scheme $S\mathcal{E}$ that corresponds with PSCFB-MOD symmetric encryption mode. Let *A* be an adversary attacking the IND-CPA security of $S\mathcal{E}$ that asks at most *p* queries formed each one for a pair of messages $(M_i^0, M_i^1)$ with a length of $m_i$ symbols of radix *S*. The *p* message queries will produce a total number of *q* encrypted symbols, which means that $q = \sum_{i=1}^{i=p} m_i$.

Then it is possible to express the IND-CPA advantage of *A* over the PSCFB-MOD scheme in terms of the prf-advantage of any adversary over the block cipher $E_K$ that is part of the $F_K\_MOD$ function, such that:

$$ADV_{PSCFB-MOD}^{IND-CPA}(A) \leq 2 \cdot ADV_{E_K}^{PRF}(B) + \\ + \frac{q}{2^{l-1}} + P_{max_{r_i}} \cdot \frac{q^2}{P-1+L_{IV}} \quad (14)$$

where $ADV_{E_K}^{PRF}(B)$ is the prf-advantage of any adversary *B* over $E_K$, $E_K$ corresponds to the block cipher that is part of the $F_K\_MOD$ function, *P* is the total number of pipeline stages in $F_K\_MOD$, $L_{IV}$ is the length of the $IV_S$ used in PSCFB-MOD scheme and *I* is the difference between *L* (the input bit length of modulo-*S* operation in $F_K\_MOD$) and *T*, with $T = log_2 S$.

As soon as $L \geq 128 + \log_2(1+2S)$ the term $P_{max_{r_i}}$ is expressed as:

$$P_{max_{r_i}} \leq 1/2^l + 1/S^{L_{IV}} \quad (15)$$

where *l* is the block size of $E_K$.

The proof of Theorem 1 is develop in Appendix A.

### D. Security Analysis: PSCFB-MOD vs CTR

Although usually block ciphers are analyzed as PRFs, PRPs (Pseudo Random Permutations) are what best models them. Thanks to the PRF-PRP switching lemma [31] it is possible to relate the PRF and PRP advantages of an adversary against a block cipher as shown in (19).

$$ADV_{E_K}^{PRF}(A) \leq ADV_{E_K}^{PRP}(A) + \frac{q^2}{2^{l+1}} \quad (16)$$

where $ADV_{E_K}^{PRF}(A)$ and $ADV_{E_K}^{PRP}(A)$ are the prf-advantage and prp-advantage of adversary *A* against block cipher $E_K$, respectively. The block size is *l* and the number of encryption queries performed by the adversary during the prf-advantage game is *q*.

According to (16), IND-CPA advantages for CTR and PSCFB-MOD in (12) and (14) can be rewritten as:

$$ADV_{CTR}^{IND-CPA}(A) \leq 2 \cdot ADV_{E_K}^{PRP}(B) + \frac{{q_{CTR}}^2}{2^{l_{CTR}}} \quad (17)$$

$$ADV_{PSCFB-MOD}^{IND-CPA}(A) \leq 2 \cdot ADV_{E_K}^{PRP}(B) + \\ + \frac{{q_{PSCFB-MOD}}^2}{2^{l_{PSCFB-MOD}}} + \frac{q_{PSCFB-MOD}}{2^{l-1}} + P_{max_{r_i}} \cdot \frac{{q_{PSCFB-MOD}}^2}{P-1+L_{IV}} \quad (18)$$

where $E_K$ is the underlying block cipher , $l_{CTR}$ and $l_{PSCFB\text{-}MOD}$ are the block sizes of $E_K$ in CTR and PSCFB-MOD, respectively, and $q_{CTR}$ and $q_{PSCFB\text{-}MOD}$ are the number of encrypted blocks and symbols during the IND-CPA games of each mode. By each encrypted block CTR mode encrypts $l_{CTR}$ information bits, while by each encrypted symbol PSCFB-MOD encrypts *T* information bits ($T = log_2 S$). Therefore the total number of encrypted bits in each mode during the IND-CPA game are $\mu_{CTR} = q_{CTR} \cdot l_{CTR}$ for CTR mode and $\mu_{PSCFB-MOD} = q_{PSCFB-MOD} \cdot T$ in PSCFB-MOD mode, respectively.

According to this, it is possible to express the advantages of equations (17) and (18) in terms of the encrypted bits and compare them with the IND-CPA advantages of other well-known operation modes, as shown in Table I.

We want to parametrize the structure of PSCFB-MOD to get a security at least better than the CTR mode when encrypting the same amount of information. It means to obtain the constraints needed to meet condition in (19).

$$ADV_{PSCFB-MOD}^{IND-CPA}(A) \leq ADV_{CTR}^{IND-CPA}(A) \quad (19)$$

TABLE I
IND-CPA ADVANTAGE COMPARISON OF DIFFERENT MODES

| Encryption Mode $\mathcal{SE}(E)$ | IND-CPA Advantage expression[1] $ADV^{IND-CPA}_{\mathcal{SE}(E)}(A)$ |
|---|---|
| PSCFB-MOD[3] | $2 \cdot ADV^{PRP}_{E}(B) + \frac{\mu^2}{T^2} \cdot f + \frac{\mu}{T \cdot 2^{l-1}}$ |
| CTR | $2 \cdot ADV^{PRP}_{E}(B) + \frac{\mu^2}{l^2 2^l}$ |
| CTR$ | $2 \cdot ADV^{PRP}_{E}(B) + \frac{2\mu^2}{l^2 2^l}$ |
| CBC | $2 \cdot ADV^{PRP}_{E}(B) + \frac{2\mu^2}{l^2 2^l}$ |
| CFB[2] | $2 \cdot ADV^{PRF}_{F}(B) + \frac{\mu^2}{m^2 2^{l+1}}$ |

[1]In each expression $l$ is the block size, $\mu$ the number of encrypted bits and $T$ the bits mapped per symbol.

[2]The term $ADV^{PRF}_{F}$ refers to the prf-advantage where $F_K$ is the function $select(E_K(\cdot)).E_K(\cdot)$ is the block cipher with block size $l$ and $select(\cdot)$ is a function that outputs $m$ fixed bits from its input.

[3]The term $f$ is : $f = 1/2^l + P_{max_{r_i}}/(P - 1 + L_{IV})$.

We assume that in both modes it is used a secure underlying block cipher $E_K$ that can be considered a good PRP, which means that the term $ADV^{PRP}_{E_K}$ is negligible in both expressions, (17) and (18). According to this, it is only necessary to compare the second term of both advantage expressions to meet (19), as shown in (20).

$$\frac{q_{PSCFB-MOD}}{2^{I-1}} + {q_{PSCFB-MOD}}^2 \left( \frac{1}{2^{l_{PSCFB-MOD}}} + \frac{P_{max_{r_i}}}{P - 1 + L_{IV}} \right) \leq \frac{{q_{CTR}}^2}{2^{l_{CTR}}} \quad (20)$$

Let us name $l_{PSCFB-MOD}$ as $l$. Since $q_{CTR} = \mu_{CTR}/l_{CTR}$, $q_{PSCFB-MOD} = \mu_{PSCFB-MOD}/T$ and $\mu_{CTR} = \mu_{PSCFB-MOD} = \mu$, expression in (20) can be rewritten as:

$$\frac{\mu}{T \cdot 2^{I-1}} + \frac{\mu^2}{T^2} \cdot \left( \frac{1}{2^l} + \frac{P_{max_{r_i}}}{P - 1 + L_{IV}} \right) \leq \frac{\mu^2}{{l_{CTR}}^2 \cdot 2^{l_{CTR}}} \quad (21)$$

where $I$=$L$-$T$.

Assuming what is mentioned in Theorem 1, as soon as $L \geq 128 + \log_2(1 + 2S)$, condition $P_{max_{r_i}} \leq 1/2^l + 1/S^{L_{IV}}$ is met. Then by substituting $P_{max_{r_i}}$ expression in (21) and doing some transformations we get (22) and (23).

$$\frac{1}{T^2} \cdot \left( \frac{1}{2^l} + \frac{1/2^l + 1/S^{L_{IV}}}{P - 1 + L_{IV}} \right) \leq \frac{1}{l_{CTR}^2 \cdot 2^{l_{CTR}}} - \frac{1/\mu}{T \cdot 2^{I-1}} \quad (22)$$

$$\frac{1/2^l + 1/S^{L_{IV}}}{P - 1 + L_{IV}} \leq \frac{T^2}{l_{CTR}^2 \cdot 2^{l_{CTR}}} - \frac{T/\mu}{2^{I-1}} - \frac{1}{2^l} \quad (23)$$

As $\mu \geq 1$ bit and $P \geq 1$, if (23) is met for $\mu = 1$ it is met for every $\mu$, and the same happens with $P$. Then (23) can be rewritten giving rise to the final condition:

$$\frac{1/2^l + 1/S^{L_{IV}}}{L_{IV}} \leq \frac{T^2}{l_{CTR}^2 \cdot 2^{l_{CTR}}} - \frac{T}{2^{I-1}} - \frac{1}{2^l} \quad (24)$$
$$L \geq 128 + log_2(1 + 2S)$$

Equation (24) is the final condition necessary to fix the values of the PSCFB-MOD parameters such as $l$, $L$ and $L_{IV}$. Note that also condition mentioned in Theorem 1 must be met, then it has also been included in (24).

According to this, we can conclude that if the underlying block ciphers used in CTR and PSCFB-MOD modes are good PRPs and have a negligible $ADV^{PRP}_{E_K}$ term, it is possible to obtain a PSCFB-MOD configuration to achieve the same or better IND-CPA security than CTR when encrypting the same amount of information. The parameters of PSCFB-MOD scheme $l$, $L$, and $L_{IV}$ will depend on the block size $l_{CTR}$ of the CTR mode taken as reference and the radix $S$ of the plaintext, as $T = log_2 S$. Expression in (24) will be the constraint necessary to achieve condition in (19).

### *E. PSCFB-MOD Hardware Implementation for 1000Base-X*

As we have mentioned in Section I the PSCFB-MOD encryption system has been adapted to work in the 1000Base-X physical layer for Gigabit Ethernet standard. In this standard only 267 possible symbols are valid in the 8b/10b encoding, which means that $S$ = 267.

On the other hand we assume that we want to achieve at least a better IND-CPA security than a standard and recommended block cipher, such AES (Advanced Encryption Standard), working in CTR mode with a standard block size of $l_{CTR}$ = 128 bits. According to this (24) can be rewritten as:

$$\frac{1/2^l + 1/S^{L_{IV}}}{L_{IV}} \leq \frac{T^2}{2^{142}} - \frac{T}{2^{I-1}} - \frac{1}{2^l} \quad (25)$$
$$L \geq 128 + \log_2(1 + 2 \cdot 267) \rightarrow L \geq 138$$

where $T = log_2 S \cong 8.06$ and $I = L - T$.

In this work, we have used the same structure for $F_K\_MOD$ than in [20], letting parameters $l$ and $L$ as fixed values such that $l = 192$ and $L = 149$. Therefore to accomplish with (25) it is also necessary that $L_{IV} \geq 17$.

If we substitute the keystream generator of Fig. 6c in the generic structure of Fig. 2, and set the PSCFB-MOD parameters as $l = 192$, $L = 149$ and $L_{IV} = 17$ then it is possible to get the final encryption scheme for PCS sublayer in 1000Base-X standard as shown in Fig.7.

In the same way as in [20] to get a block size $l = 192$ we have used as underlying block cipher $E_K$, a Rijndael structure configured with 192 bit block size and 128 bit key length. For modulo operation functions, the block MOD_$2^{149}$ of Fig. 7 is simply to take the 149 least significant bits of the Rijndael output while MOD_267 module, uses more resources as 267 is not a power of two. Its implementation has been based on [32], which presents a high-speed hardware structure for a generic operation '$x \bmod z$'.

The PSCFB-MOD structure shown in Fig. 7 has been synthetized in a Xilinx Virtex 7 FPGA (Field Programmable

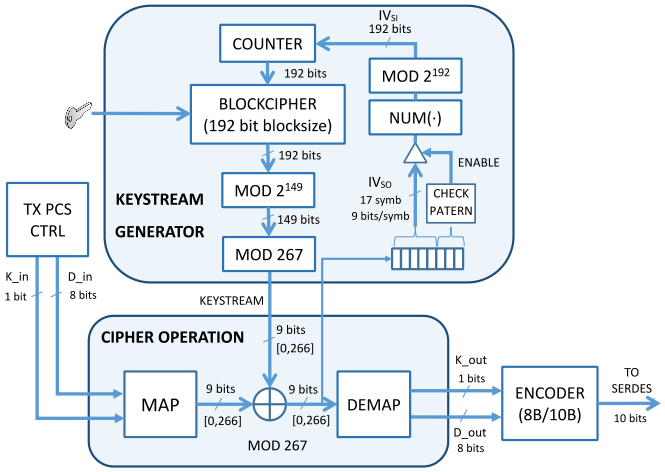



Fig. 7. Overall structure for PSCFB-MOD mode in a physical layer with 8b/10b encoding. Decryption will be as encryption but using a modulo-267 subtraction instead of an addition.

Gate Array) and the hardware resources used by this solution are shown in Table II in terms of LUTs (Look-Up Tables), registers and block RAMs. In this table PSCFB-MOD implementation is compared with other FPE implementations [33], [20], showing that this work entails a good *Encryption_Rate/resource* ratio. Although in this work the mode PSCFB-MOD entails more hardware resources than CTR-MOD [20], it is at expense of adding the self-synchronization property.

Regarding the comparison with the other PSCFB solution in radix *S* [18], another benefit of this work is that it does not need input and output queues, which makes it better in terms of encryption latency. While in [18] latency introduced in the PCS datapath is 648 ns, in this work it is only 48 ns.

As mentioned in the Introduction, in the original PSCFB specification [19] input and output queues are needed to store information temporarily during periods of resynchronization, where partial block cipher outputs are used to encrypt data due to the fact that the sync pattern is not aligned with the end of the block cipher output. It makes PSCFB mode inherently to have an encryption efficiency that can be less than 100% and increases its latency considerably as [18].

However, in PSCFB-MOD the sync pattern (formed by *n* symbols) is always aligned with the output block size of $F_K\_MOD$ as it is one symbol length. It means that the encryption throughput will be 100% and then no queue is needed in this encryption scheme which reduces its overall latency to the delay introduced only by the modulo-*S* addition.

The test set-up for the complete Gigabit Ethernet interface and the encryption results are shown in Section V.

### *F. Synchronization Recovery Delay Discussion*

The SRD (Synchronization Recovery Delay) is the metric used to examine the resynchronization properties of SCFB and PSCFB modes [26]. It is defined as the expected number of bits following a sync loss before synchronization is reestablished. According to [19], upper and lower bounds for SRD depend on the block size of the underlying block cipher, the number of pipelines *P* and the size of the synchronization pattern *n*.

TABLE II
COMPARISON WITH OTHER SOLUTIONS

| | FF1 [33] | FF3 [33] | CTR-MOD [20] | This Work |
|---|---|---|---|---|
| Slice Registers | 11285 | 5592 | 8807 | 10317 |
| Slice LUTs | 7426 | 3587 | 10974 | 12261 |
| 18K Block RAMs | 343 | 170 | 78 | 78 |
| Slices[1] | 3268 | 1596 | 3844 | 4355 |
| Encryption Rate (Mbps) | 41.1 | 109.6 | 1000 | 1000 |
| Encryption Rate/Slice (Kbps/Slice) | 12.57 | 68.7 | 260.1 | 229.6 |
| Self-synchronization | No | No | No | Yes |

[1]Slices are estimated from the number of register and LUTs, assuming they are not packed together.

As shown in Fig. 5, a sync cycle is composed of fixed length data such as the sync pattern and blackout period, and a variable length data formed by the scan period with a length of *W* bits or symbols. In [25] and [19] it is shown that *W* follows a probability distribution that can be approximated by a geometric distribution such as:

$$P(W) = (1-p)^W \cdot p \tag{26}$$

where *n* is the length of the sync pattern in bits and $p = 1/2^n$. When the radix of the symbols is *S* instead of 2, as in this work or in [18], it is possible to reach the same conclusion but using $p = 1/S^n$. According to this, the expressions for the first and second moment of the distribution of *W* will be as in [25] and [19], but using *S* as radix:

$$\begin{aligned} E\{W\} &= S^n - 1 \\ E\{W^2\} &= 2 \cdot S^{2n} - 3 \cdot S^n + 1 \end{aligned} \tag{27}$$

As in (1), we can express in a generic way the average sync cycle size in symbols instead of bits. It is the sum of the sync pattern, blackout period and scan period:

$$u = n + L_{IV} + l \cdot (P-1) + E\{W\} \tag{28}$$

where *u* is the average sync cycle size in symbols, *l* is the output block size of the underlying encryption function in symbols, $L_{IV}$ is the size of the *IV* after the sync pattern and *P* is the pipeline stages.

Since the same reasoning as in [19] can be made in terms of symbol slips or misalignment, the lower bound of SRD can be express in the same way:

$$\begin{aligned} SRD \geq{}& \frac{3}{2}\big(n + L_{IV} + l \cdot (P-1)\big) + \\ &+ \frac{1}{2u}\Big(\big(n + L_{IV} + l \cdot (P-1)\big)E\{W\} + E\{W^2\}\Big) \end{aligned} \tag{29}$$

| | H (Header) | | P (Payload) | |
|---|---|---|---|---|
| SCM | HEADER (2B) | STATE (1B) | COUNTER (5B) | MAC_F (8B) |
| $ICM_0$ | HEADER (2B) | COUNTER (6B) | | MAC_F (8B) |
| $ICM_1$ | HEADER (2B) | COUNTER (6B) | | MAC_I (8B) |
| $ICM_2$ | HEADER (2B) | COUNTER (6B) | | MAC_FI (8B) |
| SEM | HEADER (2B) | KEY_INDEX (2B) | COUNTER (4B) | MAC_F (8B) |
| SDM | HEADER (2B) | KEY_INDEX (2B) | COUNTER (4B) | MAC_F (8B) |
| EEM | HEADER (2B) | COUNTER (6B) | | MAC_F (8B) |
| EDM | HEADER (2B) | COUNTER (6B) | | MAC_F (8B) |

Fig. 8. Structure of the different messages used in the control encryption mechanism. The size in bytes of each message field is indicated in brackets. The 2-byte header *H* in each message contains the message type composed of an 8b/10b control symbol and a data one. The next 6 bytes are the payload *P*. Every message is formed by 16 bytes.

If we call $m = n + L_{IV} + l \cdot (P - 1)$, then, as in [19] but using radix *S* instead of binary radix it is possible to express the upper bound as:

$$SRD \leq \frac{3}{2}m + \frac{1}{2u}m \cdot E\{W\} + \frac{1}{2u}E\{W^2\} + \frac{n}{S^n}m \cdot \lambda \quad (30)$$

where $\lambda = (1 - 1/S^n)^{-m}$.

With both equations, (29) and (30), we can get the theoretical upper and lower bounds for SRD in PSCFB modes with radix *S*.

As in [25] and [19] with small values of *n*, better SRD are obtained. Indeed, with the parameters used in this work for *S*, *P*, *L* and $L_{IV}$ the values for SRD become huge when $n > 1$. Then, for a viable solution *n* has been set to the minimum $n = 1$.

In the case of PSCFB in radix *S* [18], *S* = 267, *P* = 41, *l* = 22, $L_{IV}$ = 22 and $n = 1$, which makes the maximum SRD bound be $SRD_{max}$ = 1618 symbols.

In this work, PSCFB-MOD, the parameters are *S* = 267, *P* = 84, *l* = 1, $L_{IV}$ = 17, $n = 1$, then the maximum SRD bound is $SRD_{max}$ = 382 symbols.

As the symbol cycle last 8 ns in 1000Base-X standard the upper bound for SRD will be approximately 12.94 µs and 3.56 µs for PSCFB in [18] and PSCFB-MOD in this work, respectively.

## IV. Control Encryption Mechanism

In any cryptographic system, confidentiality is not the only security characteristic that must be met. From a complete security perspective, some issues such as integrity, authenticity, data freshness and others such as key refreshing must be addressed. In our approach, a mechanism based in small messages formed by 8b/10b symbols has been implemented to provide the mentioned services.

Initial master key $K_M$ generation and set up is out of our scope of consideration. As in other encryption mechanisms these tasks can be carried out initially by upper layers, as for example in MACsec standard with protocols such as IEEE 802.1X. After the initial set up, different session keys can be derived from the master key and refreshed at the beginning of each encryption session thanks to a key index. In addition, session keys are composed by a set of three different keys: encryption keys ($K_{CIPH}$), frame MAC key ($K_{FRAME}$), and ICM MAC key ($K_{ICM}$). $K_{CIPH}$ is used as data encryption/decryption key while $K_{FRAME}$ and $K_{ICM}$ are used in the calculus of the MAC field for checking the integrity of the control messages and the encrypted 8b/10b symbol flow in the link, respectively.

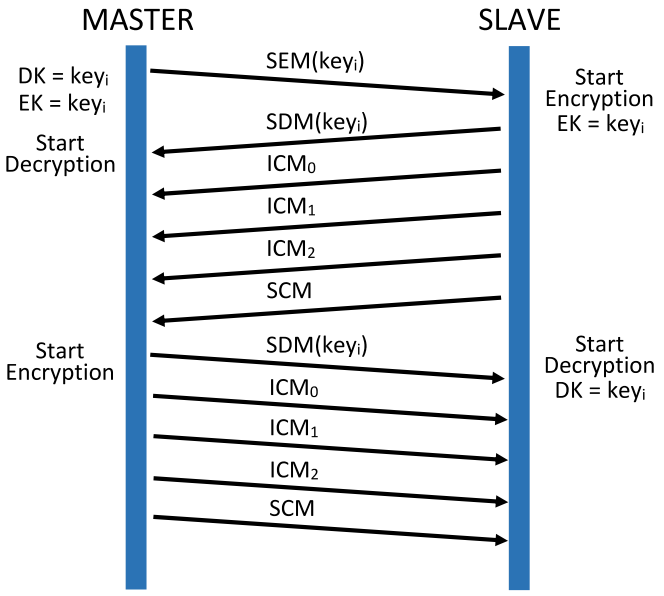



Fig. 9. Diagram of an encryption session establishment.

In this work we consider that one of the terminals is the master of the encryption session while the other is the slave. The master is responsible to start and finish the encryption session that is always stablished in a bidirectional way, which means that when the encryption session starts both directions in the transmission, master to slave and slave to master, are encrypted. Once the encryption session is stablished, data integrity and authenticity are checked periodically until the session is finalized. Also the master is responsible to perform the key refreshing while the session lasts. In Fig. 8 the structure of the messages used in these tasks is shown. Two groups of messages have been established. On the one hand the non-periodic messages, formed by SEM (Start Encryption Message), SDM (Start Decryption Message), EEM (End Encryption Message) and EDM (End Decryption Message). These are used for controlling the start and end of the encryption session. On the other hand the periodic messages, used for checking the integrity and status of the encryption link, and formed by SCM (State Check Message) and the set of ICM (Integrity Check Message) messages, formed by $ICM_0$, $ICM_1$ and $ICM_2$.

### A. Encryption Session Establishment

The establishment of an encryption session starts when the master sends a SEM to the slave. In this message the key index field indicates the initial key for the encryption session. When

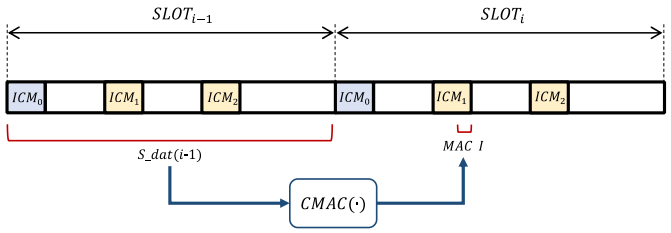



Fig. 10. Slot division for Integrity Check Messages.

the SEM is received at the slave, it responds with a SDM with the same key index and activate its cipher with the corresponding key just after SDM transmission. The slave also starts to transmit SCMs and ICMs periodically. With the SCMs slave updates its encryption state in the master, and with ICMs, master is able to check the integrity of the link in the direction slave to master. After the reception of the SDM the master starts to decrypt the link. Then the master checks the proper reception of SCMs and ICMs from the slave. After that it sends an SDM to the slave activating its encryption when it has been transmitted and sending also periodically SCMs and ICMs messages to the slave. The slave will perform the same actions as the master, activating its decryption after SDM reception and checking the periodic reception of SCMs and ICMs transmitted from the master. A similar process is produced when encryption session is finalized by the master, but using EEM and EDM messages. In Fig. 9 a diagram of an encryption session establishment is shown. According to this, the only control messages that are not encrypted are SEM and SDM, that are transmitted before starting the encryption session. The rest are transmitted while the encryption session is in process.

### B. *Message Authentication Control and Data Freshness*

In order to guarantee that no adversary is able to inject false control messages in the link a MAC (Message Authentication Code) field has been added at the end of each message. In addition, to avoid replying attacks with old transmitted messages and guarantee their freshness, a counter field has been used. The MAC field, is calculated in the following way:

$$MAC\_F = CMAC(K_{FRAME}, H|P, 64) \quad (31)$$

where *MAC_F* is the calculated MAC that will be used as the last field in the message, as shown in Fig. 8, and *CMAC(K, M, Tlen)* is a function specified in [34] that consists of an AES block cipher with 128 bit key size working in CMAC mode. It calculates the MAC field of *Tlen* bits for a message *M*, with a key *K*. In our case, MAC field is 64-bit length and $K_{FRAME}$ is the key used for the calculus of the MAC and is derived from the master key and the index key used at the beginning of the encryption session. The field *H|P* is the content of the message, which is the concatenation of the header *H* and the 6-byte payload *P*.

The MAC field *MAC_F* is also calculated at the receiver and compared with the MAC field received in the message. If they are different the message is discarded and a MAC field error is detected.

Regarding the counter values, two internal counters have been used, one for the non-periodic messages and other the periodic ones. At the receiver any message received with a counter lower that the expected is discarded.

The counter of the non-periodic messages at the transmitter is incremented each time messages SEM, SDM, EEM and EDM are transmitted. In the case of periodic messages, a set of ICMs formed by $ICM_0$, $ICM_1$ and $ICM_2$ is transmitted next to an SCM periodically. Each message of the set is sent with the same counter value, which is incremented in each new set transmission.

### C. *Integrity Check Messages*

The link integrity is evaluated periodically during the encryption session by calculating a MAC code over the 8b/10b symbol stream. The encryption session is divided in time slots of approximately 100 μs each one. In turn, each slot is split in four subslots. Slots are delimited by the transmission of $ICM_0$ message at the beginning of its first subslot as shown in Fig. 10. $ICM_1$ is transmitted at the beginning of the second subslot and it carries the MAC_I field which corresponds with the MAC value calculated over the 8b/10b symbols in the previous slot. This MAC value is calculated according to (32).

$$MAC\_I_i = CMAC(K_{ICM}, S\_dat_{i-1}, 64) \quad (32)$$

where $MAC\_I_i$ is the MAC value transmitted in the *i*-th slot, $S\_dat_{i-1}$ is the concatenation of whole encrypted 8b/10b symbols transmitted during the previous slot and $K_{ICM}$ is the key used for the calculus of the MAC_I field.

As $ICM_1$ does not carry any MAC_F field to check its own integrity, a last ICM message, $ICM_2$, is transmitted at the beginning of the third subslot. It carries a MAC field obtained from the content of the $ICM_1$ and $ICM_2$, which means that:

$$MAC\_FI = CMAC(K_{FRAME}, ICM\_dat_1|ICM\_dat_2, 64) \quad (33)$$

where $ICM\_dat_1 = H_{ICM1}|P_{ICM1}|MAC\_I$, $ICM\_dat_2 = H_{ICM2}|P_{ICM2}$, $H_{ICM1}$ and $H_{ICM2}$ are the headers of $ICM_1$ and $IMC_2$, respectively, and $P_{ICM1}$ and $P_{ICM2}$ are their payloads.

Three error issues can be detected regarding the ICM messages. A bad formed ICM set because some of the ICMs is lost in the set, a bad ICM set counter because the counter of the received $ICM_0$ is lower than the expected or because ICMs messages in the set do not have the same counter value, a bad MAC ICM set because the received MAC_F in $ICM_0$ or MAC_FI in $ICM_2$ are incorrect, and finally an ICM integrity error because the MAC_I field received is not equal than the MAC_I value calculated at the receiver in the previous slot. When complete ICM sets are received with good counter value and good MAC_F, MAC_FI and MAC_I fields during several slots the receiver consider that the link integrity is correct.

### D. *Key Refreshing*

Key refreshing is initiated by the master terminal by transmitting an SDM with an incremented key index value. On

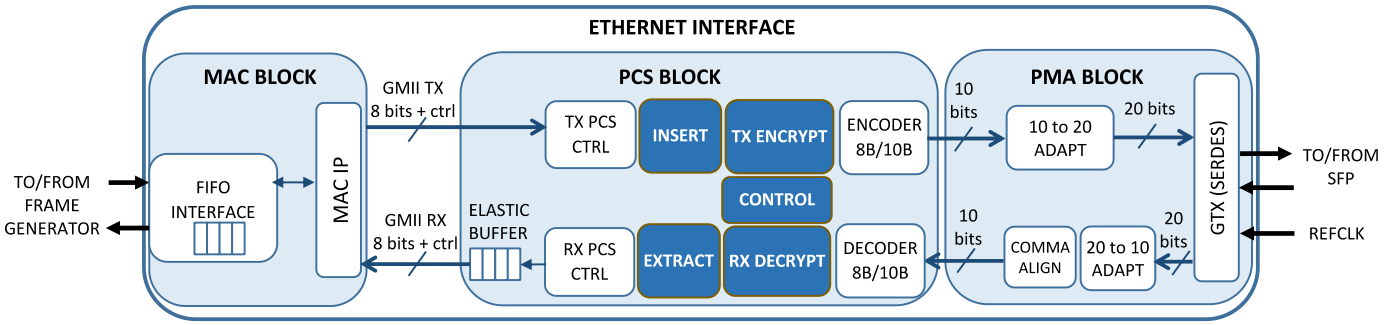



Fig. 11. Scheme of FPGA implementation for Ethernet interface with the encryption function. TX_ENCRYPT and RX_DECRYPT modules perform encryption/decription process with the PSCFB-MOD structure. CONTROL module carries out the transmissión/reception management of the control messages. INSERT/EXTRACT modules are able to insert and extract control messages according to what is mentioned in Section IV E.

the master side, encryption key $K_{CIPH}$ and MAC key $K_{FRAME}$ in TX direction will be updated thanks the new key index. The same will be done for $K_{ICM}$ used in the calculus of MAC_I, but this will be done in the next ICM time slot after SDM transmission. In the slave side, after receiving the SDM with the new index, the same key update process as in the master will be performed but with decryption keys $K_{CIPH}$, $K_{FRAME}$ for frames in RX and $K_{ICM}$ for ICM periods in RX. In both terminals is not necessary to stop the encryption and decryption process to update $K_{CIPH}$ as key update is performed transparently. After receiving the SDM from the master, the slave terminal sends a new SDM towards master using the new received key index and therefore performing the same mentioned process as in the master to slave direction.

The three kind of keys $K_{CIPH,}$ $K_{FRAME}$ and $K_{ICM}$, are obtained thanks to the master key configured before starting encryption session and the key index that has been updated at the beginning of the session. For this process KDF (Key Derivation Function) in counter mode as specified and analyzed in [35], [36] has been implemented. Particularly, KDF in counter mode is run each time a key refresh process is demanded to get a key material with length $L_{MAT}$ equal to 512 bits, which is equivalent to three keys with a length of 128 bit each one.

Assuming that $K_M$ is the master key and *i* the key index, each key is obtained according the expressions in (34), these equations are equivalent to run an iteration of the KDF algorithm to get the key material that is the concatenation of all required keys.

$$\begin{aligned} K_{CIPH} &= prf\_func(K_S(i), 0x01|VAL|L_{MAT}) \\ K_{FRAME} &= prf\_func(K_S(i), 0x02|VAL|L_{MAT}) \\ K_{ICM} &= prf\_func(K_S(i), 0x03|VAL|L_{MAT}) \\ K_S(i+1) &= prf\_func(K_S(i), 0x04|VAL|L_{MAT}) \end{aligned} \quad (34)$$

where $K_S(i)$ is a state key that is updated in each key refresh session and its initial value is the master key, such that $K_S(0) = K_M$. This is set when the first SDM, with key index equals to zero, is transmitted after the initial master key set up. $K_S(i)$ is used as key derivation key used to derive new key material in each KDF iteration at the beginning of the encryption session. In (34) $prf\_func$ is the PRF used in KDF and in our case an AES block cipher with 128 bit key size working in CMAC mode has been used. On the other hand the second argument of the PRF function has been defined as in KDF specification, the concatenation of a counter that takes values from 0x01 to 0x04, a constant value *VAL* formed by different tags that we have let to zero, and the length value of the key material $L_{MAT}$ which is equal to 512 bits.

### *E. Message Insertion*

The size of the control messages is constant and equal to 16 bytes. In order to insert these messages a mechanism similar to that presented in [37] has been used. In this mechanism, before performing the encryption, 8b/10b symbols are analyzed looking for the presence of idle sets. When at least 8 continuous idle sets are detected, the message is inserted in the 8b/10b symbol stream by replacing these idle sets by the symbols of the message itself. At the receiver, after decryption, the reverse process is performed, extracting the control message to be processed and replacing it by 8 idle sets.

Idle sets are formed by two 8b/10b symbols, a control one and a data one. They are sent continuously over the link when no frame is being transmitted. In a link where frames are transmitted continuously idle sets will be present only in the IFG (Inter Frame Gap) between frames. If the bandwidth occupied by the frame traffic is enough low and then IFGs are long enough, longer than 8 idle sets, then the insertion of the control messages will be possible. However if the occupied bandwidth is the maximum, the IFG is minimum, and as specified by the standard, it will last 96 µs or 6 idle sets. In this case it will be impossible to insert the control messages, unless we do not reduce the bandwidth used by the frames. In order to avoid this limitation and keep the maximum data bandwidth for the data frames, in this work the preamble of the transmitted frames has been reduced in 6 bytes, then widening the minimum IFG up to 9 idle sets, which is enough for our purposes. At the receiver, after decrypting and extracting control messages, preamble is restored to its original size before frames arrives to the MAC layer.

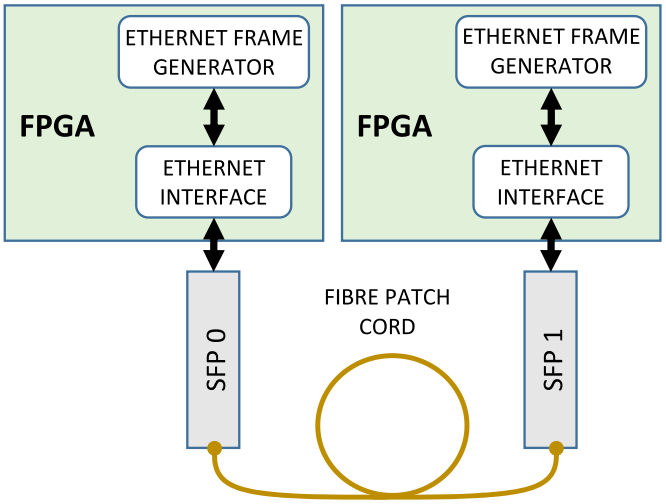



Fig. 12. Test set-up scheme.

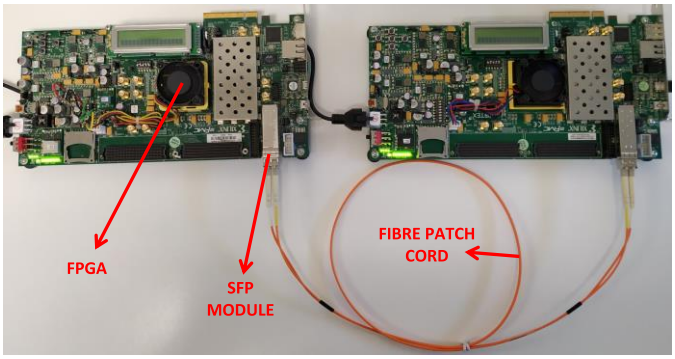



Fig. 13. Test set-up photo.

TABLE III
MEASURED SHANNON ENTROPY

| Patterns[1] | SE (t=1) | | SE (t=2) | | SE (t=3) | |
|---|---|---|---|---|---|---|
| | N-E | E | N-E | E | N-E | E |
| A | 1 | 8.06 | 0 | 8.059 | 0.33 | 8.059 |
| B | 2.19 | 8.06 | 1.05 | 8.058 | 1.26 | 8.059 |
| C | 5.48 | 8.06 | 4.46 | 8.058 | 4.44 | 8.058 |
| D | 7.79 | 8.06 | 7.45 | 8.059 | 7.42 | 8.058 |
| E | 7.98 | 8.06 | 7.82 | 8.058 | 7.81 | 8.059 |

[1]Pattern A corresponds with the case of no frame transmission, where only idle sets are transmitted over the link. Patterns B, C and D correspond to continuous frame transmission of 1024-bytes length at rates of 10.2%, 50% and 91% of the maximum Gigabit line rate. Pattern E corresponds to continuous frame transmision of random length and minimium IFG. SE is calculated for each pattern without encryption (N-E) and after encryption (E).

## V. TEST SET-UP AND ENCRYPTION RESULTS

### A. *Encryption Set-Up*

The system, formed by the encryption mechanism PSCFB-MOD and the encryption control module has been finally implemented in a Xilinx Virtex 7 FPGA. As shown in Fig. 11, these encryption modules have been integrated in a 1000Base-X Physical layer next to a MAC (Medium Access Control) module to give as result an Ethernet interface with encryption capabilities. As shown in Fig. 12, on the one hand, the Ethernet interface is connected to an SFP (Small Form-Factor Pluggable) module capable of transmitting at a rate of 1.25 Gbps. On the other hand the Ethernet interface is attached to an Ethernet Frame Generator able to generate and analyze real Ethernet traffic flows composed by data frames. The test set-up is shown in Fig. 12 and Fig. 13, where two FPGA eval boards are faced to test an encrypted optical link.

### B. *Encryption Throughput and Latency*

Several conclusions can be drawn from simulation and hardware debugging. On the one hand, it is possible to conclude that encryption/decryption is performed correctly and synchronously. Thanks to the mechanism explained in Section IV E, the activation and deactivation of the encryption session can be carried out transparently without affecting negatively neither in the traffic nor in the link status. On the other hand, the periodic transmission of ICM and SCM messages is possible without the necessity of reducing the overall data bandwidth. In particular bursts of 1024-byte length frames were tested with a duration of $10^7$ frames transmitted using the maximum possible bandwidth of 98%, according the minimum standard IFG. Another advantage is that the proposed encryption system only introduces a total extra latency of 224 ns in the PCS datapath, which includes the latency of the INSERT and TX_ENCRYPT modules in Fig. 11 before 8b/10b encoding.

These advantages of performing encryption at physical layer constrast with encryption techniques at other layers [38] [39], which introduces overhead on data frames and reduce the overall data bandwidth. For example, in IPsec the encryption overhead can reduce the overall throughput between 20% and 90% of the maximum achievable [40]. In addition, regarding to the previously mentioned latency we can conclude that this work can perform encryption with comparable latency figures to those achieved from other physical layer techniques such as in OTN encryption [41], where latency is in the range of hundreds of nanoseconds [42].

### C. *Traffic Pattern Encryption*

One of the benefits of this kind of encryption, as mentioned in [14] is that it is possible to mask data patterns and hide the presence of transmitted frames, for example making unrecognizable their beginning and end, and also their complete headers, therefore hiding statistical traffic features, which could improve the overall security. As in [14] to prove this capability SE (Shannon Entropy) has been measured as shown in (35) for different encrypted and non-encrypted data traffic patterns. The 8b/10b symbol stream for each traffic pattern, mapped between 0 and *S*-1, has been grouped in tuples of $t$ symbols called $\beta_t$, and the probability for each tuple, $P(\beta_t)$, has been calculated. Particularly, SE has been measured for values of $t$ from 1 to 3.

$$SE = -\frac{1}{t} \cdot \sum_{0 \le \beta_t < S^t} P(\beta_t) \cdot \log_2 P(\beta_t) \quad (35)$$

SE result is given in bits/symbol. Ideally, if every $t$-tuple ($\beta_t$) is equally likely with probability $P(\beta_t) = p = S^{-t}$ the value of Shannon Entropy for every $t$ should be as in (36).

$$-\frac{1}{t} \cdot S^t \cdot p \cdot \log_2 p = \log_2 S = \log_2 267 \cong 8.0606 \quad (36)$$

Owing to the limited memory in FPGA hardware resources, measurements for each pattern have been calculated at simulation stage. The results for SE values has been shown in Table III. In this table it is possible to note that encrypted flows achieve the maximum entropy value while non encrypted flows achieve a value different for each traffic pattern and lower than the maximum in (36). This fact shows that the different traffic patterns are indistinguishable when encryption is active, which proves the masking property of the proposed scheme.

## VI. Conclusion

In this work the authors have presented a new self-synchronous symmetric encryption scheme able to cipher data preserving its format. The security analysis has been carried out concluding that it is possible to get at least the same or better security than classical CTR mode structures using 128 bit block size ciphers. In addition, as the underlying block cipher in the proposed mode can be a recommended one in binary radix, the hardware complexity is reduced in regards to the typical FPE modes as FF1 or FF3.

The proposed solution has been parametrized to work in an optical Ethernet Interface with encryption capabilities. It has been tested with real Ethernet traffic proving its masking property at physical layer. The implementation results also give a good *Encryption_Rate/Slice* ratio than other existing implementations for FPE solutions. Moreover the proposed self-synchronous encryption structure does not need input and output queues, which contrasts with typical PSCFB schemes, reducing the latency introduced in the encryption datapath. Finally, a control mechanism has been developed to provide data integrity, authenticity, freshness and key refreshing over the encrypted link.

## Appendix

### A. Proof of Theorem 1

By taking into account that the underlying encryption function of PSCFB-MOD is $F_K\_MOD$ instead of $E_K$, it is possible to express directly its IND-CPA advantage expression in the same way as with PSCFB in (11), such that:

$$ADV^{IND-CPA}_{PSCFB-MOD}(A) \le 2 \cdot ADV^{PRF}_{F_K\_MOD}(B) + P_{PSCFB-MOD}(col) \quad (37)$$

where $ADV^{IND-CPA}_{PSCFB-MOD}$ is the advantage of $A$ attacking the PSCFB-MOD scheme when its underlying function is the PRF $F_K\_MOD$, $ADV^{PRF}_{F_K\_MOD}$ is the prf-advantage over $F_K\_MOD$ and $P_{PSCFB-MOD}(col)$ is the collision probability of the counter during the IND-CPA game for PSCFB-MOD.

According to the following two lemmas it is possible to proof Theorem 1.

*Lemma 1*: Let $F_K\_MOD: \mathcal{K} \times \{0,1\}^l \to \{0, \dots, S-1\}$ be a PRF, such that $F_K\_MOD(x) = (E_K(x) \bmod 2^L) \bmod S$, as defined in Section II-B, where $S$ is not a power of two, $E_K$ is a block cipher with block size $l$ and $L$ is an integer number smaller than $l$. Then any adversary $A$ making $q$ oracle queries when attacking the prf-security of $F_K\_MOD$ will obtain an advantage bounded by the following expression according to [20]:

$$ADV^{PRF}_{F_K\_MOD}(A) \le ADV^{PRF}_{E_K}(B) + \frac{q}{2^I} \quad (38)$$

where $I = L - T$, $T = log_2 S$ and $ADV^{PRF}_{E_K}(B)$ is the prf-advantage of any adversary over $E_K$. Note that in the definition of the prf-advantage [27] $ADV^{PRF}_{F_K}$ of any adversary over a generic PRF $F_K$ the adversary sends $q$ blocks of data to an oracle performing that PRF. Then the oracle answer to the adversary with the $q$ encrypted blocks. In the case of $F_K\_MOD$ it has an input size of $l$-bits and an output size of one symbol in radix $S$ which means that the $q$ queries correspond with $q$ encrypted symbols.

*Lemma 2*: Let $F_K\_MOD: \mathcal{K} \times \{0,1\}^l \to \{0, \dots, S-1\}$ be a PRF, such that $F_K\_MOD(x) = (E_K(x) \bmod 2^L) \bmod S$, where $S$ is not a power of two, $E_K$ is a block cipher with block size $l$ and $L$ is an integer number smaller than $l$. Then the term $ADV^{PRF}_{F_K\_MOD}$ can be considered lower than the prf-advantage over a block cipher with block size $l_{REF}$ bits as soon as $L \ge l_{REF} + \log_2(1 + 2S)$.

*Lemma 3*: Let $S\mathcal{E}$ be a PSCFB-MOD scheme with an underlying encryption function $F_K\_MOD$ such that $F_K\_MOD: \mathcal{K} \times \{0,1\}^l \to \{0, \dots, S-1\}$, where $L_{IV}$ is the length of the $IV_S$ used in the PSCFB-MOD scheme, $P$ is the number of pipeline stages for $F_K\_MOD$ and $q$ is the number of encrypted symbols during the $p$ message queries performed in the IND-CPA adversary game against this encryption mode. Then the probability of collision $P_{S\mathcal{E}}(col)$ of any counter value during the $p$ queries is bounded by the following expression:

$$P_{PSCFB-MOD}(col) \le P_{max_{r_i}} \cdot q^2/(P - 1 + L_{IV}) \quad (39)$$

where $P_{max_{r_i}} \le 1/2^l + 1/S^{L_{IV}}$ as soon as the parametrization of $F_K\_MOD$ makes it be a good PRF when comparing it with a recommended block cipher with a 128 bit block size as specified in Lemma 2, which means that $L \ge 128 + \log_2(1 + 2S)$.

By substituting the terms $ADV^{PRF}_{F_K\_MOD}(A)$ and $P_{PSCFB-MOD}(col)$ of equations (38) and (39) in (37) it is

possible to get as result equation (14) which proves Theorem 1.

Expression of Lemma 1 is already proven in [20]. For clarity purposes proofs of Lemma 2 and Lemma 3 are included in Appendices B and C, respectively.

### *B. Proof of Lemma 2*

In the proof of Lemma 2 it is necessary to take into account that the underlying encryption function in PSCFB must be a good PRF. In this way it is possible to consider that its output values are random and independent. As $F_K\_MOD$ is the underlying function in PSCFB-MOD then it is necessary to know under what condition we can consider it as a good PRF.

According to [27] when a random function works as the underlying function in a counter scheme, successive values of the counter makes the output values of the function be random and independent. When the underlying function is not a random one, but a PRF, then if the prf-advantage of any adversary over this function is low enough we can consider that the PRF is undistinguishable from a random function and its output values also can be considered random and independent.

In general, recommended block ciphers can be considered good enough to generate random values as keystream in counter schemes such as CTR or PSCFB. The idea of this proof is to compare the prf-advantages of a recommended block cipher and the function $F_K\_MOD$. Let be $E_{K\_REF}$ the block cipher used as reference with a block size of $l_{REF}$ bits. Let be $F_K\_MOD$ a PRF such that $F_K\_MOD(x) = (E_K(x)\ mod\ 2^L)\ mod\ S$, where $E_K$ is a block cipher with block size *l* and *L* is an integer number smaller than *l*. Then their prf-advantages should be related as:

$$ADV^{PRF}_{F_{K\_MOD}}(A) \leq ADV^{PRF}_{E_{K\_REF}}(A) \tag{40}$$

According to Lemma 1 [20] prf-advantage of $F_K\_MOD$ is bouded by the following expression:

$$ADV^{PRF}_{F_{K\_MOD}}(A) \leq ADV^{PRF}_{E_K}(B) + \frac{q}{2^I} \tag{41}$$

where $q$ is the number of queries performed during the prf-advantage game and $I = L\text{-}T$.

On the other hand, thanks to PRF-PRP switching lemma expression (16) it is possible to rewrite prf-advantages of $F_K\_MOD$ and $E_{K\_REF}$ as:

$$\begin{aligned} ADV^{PRF}_{F_{K\_MOD}}(A) &\leq ADV^{PRP}_{E_K}(B) + \frac{q^2}{2^{l+1}} + \frac{q}{2^I} \\ ADV^{PRF}_{E_{K\_REF}}(A) &\leq ADV^{PRP}_{E_{K\_REF}}(A) + \frac{{q_{REF}}^2}{2^{l_{REF}+1}} \end{aligned} \tag{42}$$

where $q_{REF}$ are the number of queries performed during the prf-advantage game of $E_{K\_REF}$.

By taking into account (42) and assuming that we want to get a better bound for the prf-advantage with $F_K\_MOD$ than with $E_{K\_REF}$ when performing the same number of queries in the prf-game, which means $q_{REF} = q$, then (40) can be expressed as:

$$ADV^{PRP}_{E_K}(B) + \frac{q^2}{2^{l+1}} + \frac{q}{2^I} \leq ADV^{PRP}_{E_{K\_REF}}(A) + \frac{q^2}{2^{l_{REF}+1}} \tag{43}$$

We assume that in both cases secure underlying block cipher $E_K$ and $E_{K\_REF}$ are used. As block ciphers can be considered good PRPs it means that the term $ADV^{PRP}_{E_K}$ is negligible in both sides of the condition in (43). According to this, it is only necessary to compare the second term of both advantage expressions to meet (43).

$$\frac{1}{2^{l+1}} + \frac{1}{q \cdot 2^I} \leq \frac{1}{2^{l_{REF}+1}} \tag{44}$$

As $q \geq 1$ the left side of (44) is maximum with $q = 1$. Then if (44) meets for $q = 1$ it will met for every *q*. Therefore:

$$\frac{1}{2^l} + \frac{1}{2^{I-1}} \leq \frac{1}{2^{l_{REF}}} \tag{45}$$

As $L \leq l$, let be $P = l - L$. Since $I = L - T$, then:

$$\frac{1}{2^{P+L}} + \frac{1}{2^{L-T-1}} \leq \frac{1}{2^{l_{REF}}} \tag{46}$$

By doing some transformations in (46) we get:

$$2^L \geq 2^{l_{REF}}(2^{T+1} + 1/2^P) \tag{47}$$

As $P \geq 0$ the right side in (47) will be maximum with $P = 0$. In addition $T = log_2 S$. Therefore (47) can be rewritten as:

$$L \geq l_{REF} + log_2(1 + 2S) \tag{48}$$

Equation (48) will be the condition to meet (40) and consider $F_K\_MOD$ a PRF better than a block cipher with block size $l_{REF}$ which proves Lemma 2.

### *C. Proof of Lemma 3*

The generic expression for the collision probability $P_{S\mathcal{E}}(col)$ of the counter values during the IND-CPA game against counter based encryption schemes $S\mathcal{E}$ is given in [29]. Particularly, traditional CTR and CTR$ modes are analyzed. In CTR mode the counter values are never repeated while in CTR$ the counter is reset to a random value at the beginning of a message query, which means that some repetition can happen between the counter values used to encrypt different messages.

As mentioned in Section III-B each message query in the IND-CPA game is formed by a pair of messages $(M_i^0, M_i^1)$ with a length of $m_i$ blocks of data. The $p$ message queries will produce a total number of $q$ encrypted blocks, which means that $q = \sum_{i=1}^{i=p} m_i$. These blocks of data have the same length and radix as the output of the underlying encryption function. In the case of PSCFB-MOD these blocks are formed by one symbol in radix *S*, as $F_K\_MOD$ output.

For the case of CTR$, the underlying encryption function is the block cipher $E_K$, with an input and output that are *l*-bit length. The *l*-bit length counter values along the game are represented in the following table:

$$\begin{matrix} r_1+1,\ r_1+2,\ \dots, r_1+m_1 \\ r_2+1,\ r_2+2,\ \dots, r_2+m_2 \\ \dots \quad\quad \dots \quad\quad \dots \\ r_p+1,\ r_p+2,\ \dots, r_p+m_p \end{matrix} \tag{49}$$

In (49) each *i*-th row correspond with the counters used in the encryption of the pair $(M_i^0, M_i^1)$ with length $m_i$. At the beginning of each query a random counter value $r_i$ is generated. As the counter is incremented in each encryption step and the message has a length of $m_i$ data blocks the last value of the counter in each row will be $r_i + m_i$.

In [29] it is demonstrated that the collision probability between any counter value in the i-th row and the previous row counters is limited by:

$$\Pr(col_i | no\_col_{i-1}) \leq \frac{Nchoices(r_i)}{2^l} \tag{50}$$

where $col_i$ is the event where there is a collision in the first *i* rows of the table, and $no\_col_{i-1}$ the event where there is no collision in the first *i-1* rows. $Nchoices(r_i)$ are the number of choices of $r_i$ that can produce a collision and $1/2^l$ is the probability for $r_i$ to be equal to any of these choices. We call this probability $P_{r_i}(x) = P(r_i = x)$, with $x$ any *l*-bit length value in $Nchoices(r_i)$. As $r_i$ is generated randomly we can consider it has a uniform random distribution such that $P_{r_i}(y) = P_{r_i} = 1/2^l$ for any *i* and $y \in [0, 2^l - 1]$. It means that $P_{r_i}(x)$ is equal to $1/2^l$ for any $x \in Nchoices(r_i)$.

Finally, given (50), in [29] it is shown that $P_{S\mathcal{E}}(col)$ for CTR\$, the collision probability along the counter table in (49), can be expressed as:

$$\begin{aligned} P_{CTR\$}(col) &\leq \sum_{i=2}^{p} \Pr(col_i | no\_col_{i-1}) \\ &\leq \frac{\sum_{i=2}^{i=p} Nchoices(r_i)}{2^l} \\ &\leq \frac{(p-1)\cdot \sum_{i=1}^{i=p} m_i}{2^l} = P_{r_i}\cdot(p-1)\cdot \sum_{i=1}^{i=p} m_i \end{aligned} \tag{51}$$

For the particular case of PSCFB, in [13] it is shown that the encryption session can be divided into different sync cycles as the shown in Fig. 5. In each sync cycle the counter is reset to a random *IV* value as mentioned in Section II-C. Let us assume that *N* sync cycles happen during the IND-CPA game between the adversary and the oracle. Then, the counter values during the session can be represented as in the following table:

$$\begin{matrix} r_1,\ r_1+1, \dots, r_1+k_1-1 \\ r_2,\ r_2+1, \dots, r_2+k_2-1 \\ \dots \quad\quad \dots \quad\quad \dots \\ r_N,\ r_N+1, \dots, r_N+k_N-1 \end{matrix} \tag{52}$$

Where $k_i$ is the length in blocks of data of the *i*-th sync cycle. The subsequent counters from $r_i$ in advance will be incremented up to $r_i + k_i - 1$. According to [13], by using the same reasoning than in [29], $P_{PSCFB}(col)$ can be expressed as in (51), in terms of $P_{r_i}$:

$$P_{PSCFB}(col) \leq P_{r_i}\cdot(N-1)\cdot\sum_{i=1}^{i=N} k_i \tag{53}$$

In PSCFB mode, as in CTR\$, the counter is an *l*-bit length value. It is reset to the *IV* at the beginning of each sync cycle and it can be considered random and uniformly distributed. Therefore, in (51) $P_{r_i} = 1/2^l$ for any *i* and $x \in [0, 2^l - 1]$.

In the case of PSCFB-MOD mode, as the encryption game can be also divided in *N* sync cycles the same expression as (53) could be applied to obtain $P_{PSCFB-MOD}(col)$.

However, as explained in Section III-A, the *l*-bit length *IV* is not taken directly from the ciphertext, but its value is obtained from the $L_{IV}$-symbol length $IV_S$ value, such that $IV = T(IV_S)$. As the ciphertext, in radix *S*, can be considered random and uniformly distributed, the same happens with $IV_S$, as it is taken from the $L_{IV}$ symbols after the sync pattern detection. Therefore $IV_S$ can be considered a uniform random value in the range $[0, S^{L_{IV}} - 1]$. Particularly the transformation $T(\cdot)$ as been chosen such that $IV = T(IV_S) = NUM(IV_S)\ mod\ 2^l$ as shown in (3).

$IV_S$ and its binary representation, *NUM(*$IV_S$*)*, can be considered random and uniformly distributed as soon as $F_K\_MOD$ output is considered also random, which means that parameter *L* in $F_K\_MOD$ must meet condition in Lemma 2. However, *IV* is obtained by truncating *NUM(*$IV_S$*)* to its least *l* significant bits. Therefore *IV* cannot be considered random and uniformly distributed as a bias is introduced due to modulo-$2^l$ operation. Owing to this, $r_i$ counter values at the beginning of each sync cycle will not be uniformly distributed and special considerations must be taken into account to obtain $P_{r_i}(x)$ as it will not be constant and equal to $1/2^l$.

Let assume $P_{max_{r_i}}$ as the maximum probability in the probability density function $P_{r_i}(x)$ then if we particularize (53) to PSCFB-MOD mode we can get an upper bound for the following collision probability such that:

$$P_{PSCFB-MOD}(col) \leq P_{max_{r_i}}\cdot(N-1)\cdot\sum_{i=1}^{i=N} k_i \tag{54}$$

In addition, as the output block size in PSCFB-MOD is one symbol with radix *S*, then $k_i = q_i$, where $q_i$ is the number of symbols with radix *S* encrypted in the *i*-th sync cycle then:

$$\begin{aligned} &P_{PSCFB-MOD}(col) \leq P_{max_{r_i}}\cdot(N-1)\cdot\sum_{i=1}^{i=N} q_i = \\ &P_{max_{r_i}}\cdot(N-1)\cdot q \leq P_{max_{r_i}}\cdot N\cdot q \leq P_{max_{r_i}}\cdot N_{max}\cdot q \end{aligned} \tag{55}$$

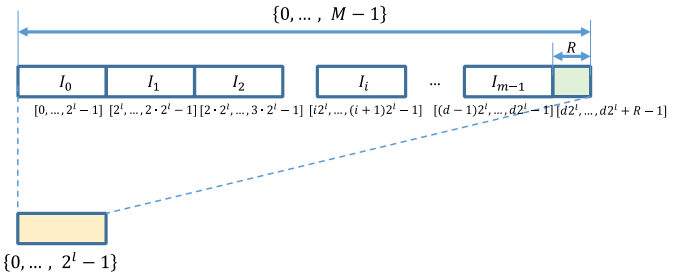



Fig. 14. Domain $\{0, \dots, M-1\}$ is mapped into interval $\{0, \dots, 2^l-1\}$. After modulo-$2^l$ operation, each range $I_i$ is equally distributed in the destination range $\{0, \dots, 2^l-1\}$, however the last range $\{d2^l, \dots, d2^l+R-1\}$, only has $R$ values and they are mapped only in the first $R$ values of $\{0, \dots, 2^l-1\}$. It makes each value in the destination range $\{0, \dots, R-1\}$ has one more occurrence than in $\{R, \dots, 2^l-1\}$.

where $q$ is the total number of symbols encrypted during the IND-CPA game and $N_{max}$ the maximum possible number of sync cycles produced. It is possible to express this value as $N_{max} = q/C_{min}$, where $C_{min}$ is the minimum size of any sync cycle in symbols. As shown in Fig. 5b the generic sync cycle length in PSCFB-MOD can be expressed as $u = n + L_{IV} + P - 1 + W$ symbols, where $n$, $L_{IV}$ and $P$ are constant, while $W$ is variable. The minimum value for this cycle length could be considered with $W$=0, then of $u_{min} = n + L_{IV} + P - 1$ , where $P$ is the pipeline size of the underlying block cipher and $n$ the sync pattern size. If we call $w = P - 1 + L_{IV}$ then $N_{max} = q/u_{min} \leq q/w$, then:

$$P_{PSCFB-MOD}(col) \leq P_{max_{r_i}} \cdot \frac{q^2}{w} = P_{max_{r_i}} \cdot \frac{q^2}{(P-1+L_{IV})} \quad (56)$$

which is the same as equation (39).

The last step to prove Lemma 3 is to demonstrate that $P_{max_{r_i}} \leq 1/2^l + 1/S^{L_{IV}}$, which is proved in the following paragraphs.

Let us name $NUM(IV_S)$ as $v_i$ and $IV$ as $r_i$. Then $v_i$ and $r_i$ will be two random variables such that $r_i = v_i \bmod 2^l$, where $v_i \in [0, M-1]$ , $r_i \in [0, 2^l-1]$ and $M$ is not a multiple of $2^l$. If $F_K\_MOD$ meets Lemma 2 its captured output $v_i$ will be random and uniformly distributed, then $P(v_i = x) = 1/M$, where $P(v_i = x)$ is the probability that $v_i$ is equal to $x$ and $x \in [0, M-1]$. In Fig. 14 the way of mapping values from the domain $\{0, \dots, M-1\}$ to $\{0, \dots, 2^l-1\}$ thanks to modulo-$2^l$ operation is shown. The remainder $R$ of the division between $M$ and $2^l$ can be written as:

$$R = M - d \cdot 2^l \quad (57)$$

Where $d$ is the quotient of the division, which means that $d = \lfloor M/2^l \rfloor$. After modulo-$2^l$ operation, as $R$ is different to zero, a bias is introduced in the resulting distribution of values of, $r_i$ in the range $\{0, \dots, 2^l-1\}$. $v_i$ values that are in the range $\{d2^l, \dots, d2^l+R-1\}$ will generate one more occurrence in the resulting range $\{0, \dots, R-1\}$ after modulo operation. Therefore the probability distribution after modulo-$2^l$ will be:

$$P(r_i = x) = \begin{cases} \frac{d+1}{M} & for\ 0 \leq x \leq R-1 \\ \frac{d}{M} & for\ R \leq x \leq 2^l-1 \end{cases} \quad (58)$$

The maximum value for the probability of $P(r_i = x)$ will be:

$$P_{max_{r_i}} = P(r_i = x | 0 \leq x \leq R-1) = \frac{d+1}{M} \quad (59)$$

As $d = \lfloor M/2^l \rfloor$ then:

$$P_{max_{r_i}} = \frac{\lfloor M/2^l \rfloor}{M} + \frac{1}{M} \leq \frac{M/2^l}{M} + \frac{1}{M} = \frac{1}{2^l} + \frac{1}{M} \quad (60)$$

As we have called $v_i = NUM(IV_S)$ then $v_i \in [0, M-1] \rightarrow v_i \in [0, S^{L_{IV}} - 1]$, which means that $M = S^{L_{IV}}$ and $P_{max_{r_i}}$ can be expressed as:

$$P_{max_{r_i}} \leq 1/2^l + 1/S^{L_{IV}} \quad (61)$$

The final expression for $P_{PSCFB-MOD}(col)$ in (56) next to the condition in (61) proves Lemma 3.

## REFERENCES


[1] Cisco Systems, "Cisco Global Cloud Index: Forecast and Methodology. 2015-2020," 2016. [Online]. Available: https://www.cisco.com/c/dam/en/us/solutions/collateral/service-provider/global-cloud-index-gci/white-paper-c11-738085.pdf. [Accessed 17 Aug. 2017].

[2] "IEEE Standard for Local and Metropolitan Area Networks: Media Access Control (MAC) Security, IEEE Std 802.1AE, 2006".

[3] S. Kent and K. Seo, "RFC 4301: Security Architecture for the Internet Protocol," 2005. [Online]. Available: https://tools.ietf.org/html/rfc4301.

[4] H. Rahbari and M. Krunz, "Full Frame Encryption and Modulation Obfuscation Using Channel-Independent Preamble Identifier," *IEEE Trans. Inf. Forensics Security*, vol. 11, no. 12, pp. 2732-2747, 2016.

[5] Y. Zhao, X. Zou, Z. Lu and Z. Liu, "Chaotic Encrypted Polar Coding Scheme for General Wiretap Channel," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 12, pp. 3331-3340, 2017.

[6] W. Li, D. Mclernon, J. Lei, M. Ghogho, S. Zaidi and H. Hui, "Cryptographic Primitives and Design Frameworks of Physical Layer Encryption for Wireless Communications," *IEEE Access*, vol. 7, pp. 63660-63673, 2019.

[7] N. Skorin-Kapov, M. Furdek, S. Zsigmond and L. Wosinska, "Physical-Layer security in evolving optical networks," *IEEE Commun Mag.*, vol. 54, no. 8, pp. 110-117, Aug. 2016.

[8] M. Furdek, N. Skorin-Kapov, S. Zsigmond and L. Wosinska, "Vulnerabilities and Security Issues in Optical Networks," in *International Conference on Transparent Optical Networks (ITCON)*, Graz, Austria, 2014.

[9] M. P. Fok, Z. Wang, Y. Deng and P. R. Prucnal, "Optical Layer Security in Fiber-Optic Networks," *IEEE Trans. Inf. Forensics Security*, vol. 6, no. 3, pp. 725-736, 2011.

[10] J. Ji, G. Zhang, W. Li, L. Sun, K. Wang and M. Xu, "Performance Analysis of Physical-Layer Security in an OCDMA-Based Wiretap Channel," *J. Opt. Commun. Netw.*, vol. 9, no. 10, pp. 813-818, 2017.

[11] K. Cui, J. Wang, H. Zhang, C. Luo, G. Jin and T. Chen, "A Real-Time Design Based on FPGA for Expeditious Error Reconciliation in QKD System," *IEEE Trans. Inf. Forensics Security*, vol. 8, no. 1, pp. 184-190, 2013.

[12] K. Guan, J. Kakande and J. Cho, "On Deploying Encryption Solutions to Provide Secure Transport-as-a-Service (TaaS) in Core and Metro Networks," in *European Conference and Exhibition on Optical Communications*, Düsseldorf, September 18 – 22, 2016.

[13] A. Pérez, M. Garcia-Bosque, C. Sanchez-Azqueta and S. Celma, "Self-synchronized Encryption for Physical Layer in 10Gbps Optical Links," *IEEE Transactions on Computers*, vol. 68, no. 6, pp. 899-911, 2019.

[14] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta and S. Celma, "Physical Layer Encryption for Industrial Ethernet in Gigabit Optical Links," *IEEE Transactions on Industrial Electronics*, vol. 66, no. 4, pp. 3287-3295, April 2019.

[15] J. Han, H. Won and H.-M. Bae, "0.6–2.7-Gb/s Referenceless Parallel CDR With a Stochastic Dispersion-Tolerant Frequency Acquisition Technique," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 22, no. 6, pp. 1219-1225, June 2014.

[16] P. Rogaway, "A Synopsis of Format-Preserving Encryption," in *Voltage Security*, Cupertino, California, 2013.

[17] M. Dworkin, Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption, Gaithersburg, Maryland: NIST Special Publication 800-38G, 2016.

[18] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta and S. Celma, "Self-Synchronized Encryption Using an FPE Block Cipher for Gigabit Ethernet," in *15th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, Lausanne, Switzerland, 2019.

[19] H. M. Heys and L. Zhang, "Pipelined Statistical Cipher Feedback: A New Mode for High-Speed Self-Synchronizing Stream Encryption," *IEEE Transactions on Computers*, vol. 60, no. 11, pp. 1581-1595, 2011.

[20] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta and S. Celma, "A New Method for Format Preserving Encryption in High Data Rate Communications," *IEEE Access*, vol. x, no. x, p. xx, 2019.

[21] M. Robshaw and O. Billet, New Stream Cipher Designs: The eSTREAM Finalists, Springer, 2008.

[22] J. L. J. P. B. Daemen, "Chosen ciphertext attack on SSS," 2005. [Online]. Available: http://www.ecrypt.eu.org/stream/papersdir /044.pdf (p. 235).

[23] A. Joux and F. Muller, "Chosen-ciphertext attacks against MOSQUITO," in *Robshaw, M. (ed.) Fast Software Encryption*, Berlin, Springer, 2006, pp. 390-404.

[24] A. Alkassar, A. Geraldy, B. Pfitzmann and A.-R. Sadeghi, "Optimized Self-Synchronizing Mode of Operation," in *Proc. Conf. Fast Software Encryption (FSE '01)*, Apr. 2001.

[25] O. Jung and C. Ruland, "Encryption with Statistical Self-Synchronization in Synchronous Broadband Networks," in *Proc. Conf. Cryptographic Hardware and Embedded Systems (CHES '99)*, 1999.

[26] H. M. Heys, "Analysis of the Statistical Cipher Feedback Mode of Block Ciphers," *IEEE Transactions on Computers*, vol. 52, no. 1, pp. 77-92, 2003.

[27] M. Bellware and P. Rogaway, "Introduction to modern cryptography," May 2005. [Online]. Available: http://web.cs.ucdavis.edu/~rogaway /classes/227/spring05/book/main.pdf.

[28] M. Bellware and P. Rogaway, "Chapter 5.4 Indistinguishably under chosen-plaintext attack," in *Introduction to modern cryptography*, May 2005.

[29] M. Bellware and P. Rogaway, "Chapter 5.7 Security of CTR Modes," in *Introduction to modern cryptography*, May 2005.

[30] P. Rogaway, "Evaluation of some blockcipher modes of operation.," in *Technical report, Cryptography Research and Evaluation Committees (CRYPTREC) for the Goverment of Japan*, Feb. 2011.

[31] M. Bellare, A. Desai, E. Jokipii and P. Rogaway, "A concrete security treatement of symmetric encryption: Analysis of the DES modes of operation," in *Symposium on Foundations of Computer Science (FOCS)*, 1997.

[32] J. Buttler and T. Sasao, "Fast Hardware Computation of x Mod z," in *Parallel and Distributed Processing Workshops and Phd Forum (IPDPSW)*, Shanghai, 2011, pp. 294-297.

[33] R. Agbeyibor, J. Butts, M. Grimaila and R. Mills, "Evaluation of format preserving encryption algorithms for critical infrastructure protection," in *Critical Infrastructure Protection VIII*, Springer, 2014, pp. 245-261.

[34] M. Dworkin, Recommendation for Block Cipher Modes of Operation: The CMAC Mode for Authentication, Gaithersburg, Maryland: NIST Special Publication 800-38B, 2005.

[35] L. Chen, Recommendation for Key Derivation Using Pseudorandom Functions, Gaithersburg, Maryland: NIST Special Publication 800-108, 2009.

[36] M. Abdalla and M. Bellare, "Increasing the lifetime of a key: a comparative analysis of the security of re-keying techniques," in *Advances in Cryptology, ASIACRYPT 2000, LNCS 1976*, vol. 1976, Berlin Heidelberg, Springer-Verlag, 2000.

[37] A. Pérez, M. Garcia-Bosque, C. Sánchez-Azqueta and S. Celma, "Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems," *IEEE Transactions on Instrumentation and Measurement*, vol. 68, no. 12, pp. 4876-488, 2019.

[38] S. Salehi, Y. Rico Cao and H. Chen, "Bandwidth-IPSec security trade-off in IPv4 and IPv6 in Windows 7 environment," in *International Conference on Future Generation Communication Technology (FGCT)*, London, UK, Nov. 2013.

[39] C. Xenakis, N. Laoutaris, L. Merakos and I. Stavrakakis, "A generic characterization of the overheads imposed by IPsec and associated cryptographic algorithms," *Computer Networks*, no. 50, pp. 3225-3241, 2006.

[40] L. Troell, J. Burns, K. Chapman, D. Goddard, M. Soderlund and C. Ward, "Converged vs. Dedicated IPSec Encryption Testing in Gigabit Ethernet Networks, Technical Report," 2005. [Online]. Available: http://scholarworks.rit.edu/article/1743.

[41] K. Guan, J. Kakande and J. Cho, "On Deploying Encryption Solutions to Provide Secure Transport-as-a-Service (TaaS) in Core and Metro Networks," in *European Conference and Exhibition on Optical Communications*, Düsseldorf, September 18 – 22, 2016.

[42] MicroSemi, "In-flight Encryption in Service Provider Networks, No: PMC-2150716, Issue 2," 2016. [Online]. Available: https:// pmcs.com/cgi-bin/document.pl?docnum=2150716. [Accessed 17 Aug. 2017].

# Self-Synchronized Encryption Using an FPE Block Cipher for Gigabit Ethernet

A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, S. Celma
Group of Electronic Design
*Zaragoza University*
Zaragoza, Spain
{aprz, mgbosque, csanaz, scelma}@unizar.es

***Abstract*—In this work, a new solution for self-synchronized encryption in physical layer at Gigabit Ethernet optical links is proposed. The solution is based in the block cipher operating mode called PSCFB (Pipelined Statistical Cipher Feedback) using as underlying PRF (Pseudo Random Function) an FPE (Format Preserving Encryption) block cipher. Thanks to this structure is possible to encrypt 8b/10b Ethernet symbols preserving its coding properties at Physical layer in an optical Gigabit Ethernet interface. The IND-CPA (Indistinguishability under Chosen-Plaintext Attack) advantage is analysed for the first time concluding that this mode can be considered secure in the same way as traditional encryption modes are. In addition it provides self-synchronization while keeping an encryption throughput near 100%. Finally, the proposed mechanism has been simulated and synthetized in an FPGA (Field Programmable Gate Array) electronic device.**

***Keywords—Self-synchronized encryption, PSCFB, FPGA, Gigabit Ethernet.***

## I. INTRODUCTION

Nowadays thanks to the advance of optical communication standards it is possible to face the bandwidth demand of modern applications. On the other hand, information security is an important issue that must be taken into account owing the increase of threat events over the last years [1]. In order to provide security in such kind of networks high-speed encryption techniques must be applied.

In a communication model such as OSI (Open System Interconnect), encryption can be performed at different communication levels. Well-known mechanisms are MACsec and IPsec for layer 2 and 3, respectively. For layer 1 or physical layer, different techniques have been developed. Some of them are related directly with the optical technologies such as OFDM or QKD [2], while others deal with the optical protocols at physical layer such as the encryption of OTN (Optical Transport Network) frames at bit level [3] or encoded symbols in 8b/10b and 64b/66b data flows of optical Ethernet standards [4], [5], [6]. The mentioned solutions for OTN and optical Ethernet can be considered in-flight encryption schemes suitable for high-speed applications [4], [7].

In the case of optical Ethernet, the mentioned encryption schemes are usually composed of a stream cipher that encrypts directly the encoded symbols at PCS (Physical Coding Sublayer) preserving its coding properties. The solutions proposed for Ethernet using 8b/10b encoding [4] , [8], only work in a synchronous symmetric encryption scheme, while in Ethernet using 64b/66b encoding there are solutions of both types, synchronous [5] and asynchronous [6].

This work has been supported by MINECO-FEDER (TEC2014-52840-R and TEC2017-85867-R) and FPU fellowship to M. García Bosque FPU14/03523.

978-1-7281-3549-6/19/$31.00 ©2019 IEEE

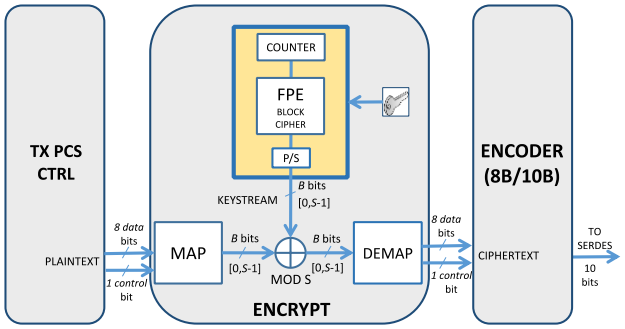



Fig. 1. Scheme of the PCS sublayer with encryption capabilities. The plaintext is formed by 8b/10b symbols composed of 8 data bits and 1 control bit. TX_PCS_CTRL is the PCS controller that generates the 8b/10b symbols. These are mapped in MAP block to *B*-bit values between 0 and *S*-1, with *S*=267. After encryption they are reverse mapped in DEMAP block and then encoded in ENCODER_8B10B module. In this scheme keystream generator is formed by an FPE block cipher working in counter mode.

The advantage of asynchronous encryption schemes over synchronous ones is that the firsts do not need any particular protocol to synchronize the stream ciphers between the transmitter and the receiver. It means that if the keystream sequences of both terminals lose their alignment, automatically the encryption synchronization can be recovered without the necessity of sending extra messages between the terminals.

The paper is organized as follows: Section II explains the previous Gigabit Ethernet encryption works and the basis of PSCFB (Pipeline Statistical Cipher Feedback) self-synchronized encryption mode, Section III proposes the new encryption mechanism based in PSFB mode and gives a new IND-CPA (Indistinguishability under Chosen Plaintext Attack) security expression for it, in Section IV implementation over FPGA is explained. Finally in Section V conclusions are given.

## II. PREVIOUS WORKS

### A. FPE Gigabit Encryption Scheme

In order to encrypt the 8b/10b symbol flow it is necessary to preserve the coding properties of the transmitted symbols as its main purpose is facilitate the work of the clock and data recovery circuits in the receiver side. Therefore, to perform the encryption in [4] and [8] 8b/10b symbols are mapped to integers between 0 and *S*-1 and a modulo-*S* addition is performed between them and the keystream values, also between 0 and *S*-1. Finally the resulting ciphertext is reverse

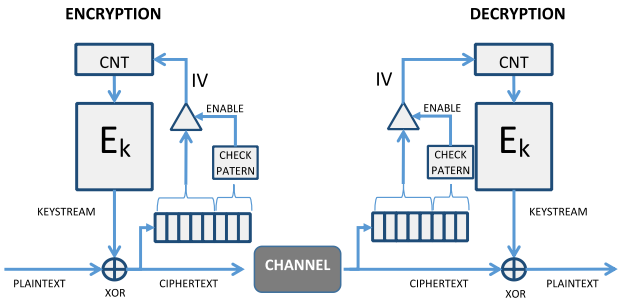



Fig. 2. Structure of PSCFB mode for encryption and decryption. The block cipher $E_k(\cdot)$ has a block size of $L$ bits, and is implemented using $P$ internal stages. The difference between the encryption and decryption is that the sync pattern scan is performed after the XOR operation in the transmitter and before the XOR in the receiver.

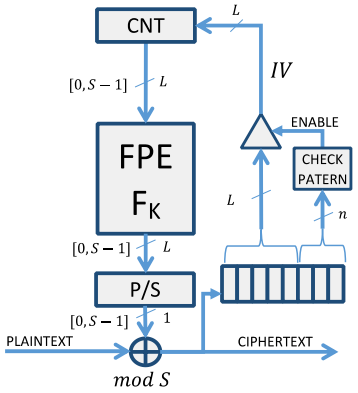



Fig. 4. Structure of PSCFB mode using an FPE block cipher as underlying function. The block cipher $F_k(\cdot)$ has a block size of $L$ symbols with values between 0 and $S$-1 and it is implemented using $P$ internal stages. Moreover, the $L$-symbol output of the block cipher is serialized to a stream of symbols thanks to the P/S (Parallel to Serial) module. The difference between the encryption and decryption is that the sync pattern scan is performed after the modulo-$S$ addition in the transmitter and before it in the receiver.

mapped to valid 8b/10b symbols. After that, the resulting symbols are encoded and transmitted to the serializer and to the optical link. The complete encryption process is shown in Fig. 1. In [8], the keystream generator is based in a classic ad-hoc stream cipher structure, particularly a chaotic one. However it is well-known that the cryptanalysis of ad-hoc stream ciphers are usually less understood and they are more difficult to use than block ciphers [9]. In [4] the keystream generator is based in an FPE (Format Preserving Encryption) block cipher working in CTR (Counter) mode. Particularly the structure used is FF3, one of the recommended by NIST (National Institute of Standards and Technology) [10]. This is the keystream generator structure shown in Fig. 1.

### B. Self-Synchronized Encryption

To achieve self-synchronous encryption, several proposals can be taken into account. On the one hand, ad-hoc self-synchronous stream ciphers such those in eSTREAM project, [11]. In this particular case the proposed solutions were dismissed owing to their vulnerabilities [12]. On the other hand, certain operation modes for block ciphers can be used, such as CFB (Cipher Feedback), SCFB (Statistical Cipher Feedback) [13] and PSCFB [14]. In these cases, as soon as the underlying block cipher is considered secure, and taking into account their analysis in the sense of their IND-CPA security, it is possible to consider these modes secure enough. Among

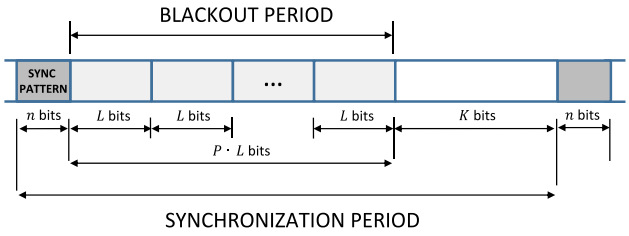



Fig. 3. Structure of the synchronization period in PSCFB mode. A complete synchronization period is formed by the sync pattern, blackout period and scan period. The *IV* (Initialization Vector) is in the first block of the blackout period. $E_k(IV)$, the encrypted value of the *IV*, is used as a keystream value $P$ blocks later of its capture because the cipher has $P$ stages.

the mentioned modes PSCFB is the only one that is able to achieve an encryption throughput near to 100%.

PSCFB mode is considered a combination of two NIST modes, CTR and CFB, its structure is shown in Fig. 2. Let's assume that the block cipher under this mode has a block size of $L$ bits, it is configured with a key $k$, and it has an encryption function $E_k(\cdot)$ that is implemented with a pipeline of $P$ stages. This block cipher usually works in CTR mode while the resulting ciphertext is analysed to detect a particular $n$-bit sequence called sync pattern. When this sequence is detected, the cipher starts to work in CFB mode and feeds back the next $L$ bits after the sync pattern using them as *IV* (Initial Value), then setting its counter to this value. The encrypted value of *IV*, $E_k(IV)$ will appear at the output of the block cipher $P$ cycles later as it has been implemented with a pipelined architecture of $P$ stages. As the encryption function $E_k(\cdot)$ of any block cipher can be usually modelled as a PRP (Pseudo Random Permutation) the resulting keystream and then the ciphertext can be considered a stream of random and independent values. Therefore the sync pattern will be detected at a random point in the ciphertext bitstream. Moreover, after *IV* capture, sync pattern detection is disabled during $P$ cycles until $E_k(IV)$ is ready as keystream value. This period is called blackout period. The sync pattern, blackout period, and the scanning period until next sync pattern is called the synchronization period and it is shown in Fig. 3.

## III. Self-Synchronized Proposed Mode

### A. Structure of the Proposed Mode

As mentioned before it is well-known that the cryptanalysis of ad-hoc stream ciphers are usually less understood and more difficult to use than block ciphers. Therefore in this work we have selected the proposal in [4] as a starting point to create a self-synchronized solution. This solution consists to apply the PSCFB mode of operation to the proposed cipher structure in [4]. It means to use directly an FPE block cipher working in PSCFB mode instead of a classical block cipher with binary radix. The FPE block cipher can be considered a PRF (Pseudo Random Function) $F_k$ such that $F_k: K \times \{0, S-1\}^l \rightarrow \{0, S-1\}^L$, where $L$ is the block size, $K$ the key space and $S$ the radix of the block cipher. The structure of this solution is shown in Fig. 4, where the XOR operation has been substituted by a modulo-$S$ addition and the block cipher $E_k(\cdot)$ has been substituted by the FPE block cipher $F_k(\cdot)$.

### B. Security Analysis

Traditional modes are analysed according to their IND-CPA security. According to [6], if the encryption

scheme is the PSCFB mode it is possible to write the mentioned IND-CPA advantage of an adversary *A* over this mode as:

$$ADV^{IND-CPA}_{PSCFB(F)}(A) = 2 \cdot ADV^{PRF}_{F}(B) + ADV^{IND-CPA}_{PSCFB(Func)}(A) \quad (1)$$

where $ADV^{IND-CPA}_{PSCFB(F)}(A)$ is the advantage of *A* attacking the PSCFB scheme when its underlying function is the PRF $F_k$, $ADV^{IND-CPA}_{PSCFB(Func)}(A)$ is the advantage of *A* over PSCFB scheme when its underlying function is a random function $Func(l, L)$ and $ADV^{PRF}_{F}(B)$ is the prf-advantage of *B* as defined in [15], where *B* is an adversary attacking the PRF security of $F_k$.

If $F_k$ in PSCFB mode is a function with binary radix, according to [6] and [15] it is possible to write the term $ADV^{IND-CPA}_{PSCFB(Func)}(A)$ as:

$$ADV^{IND-CPA}_{PSCFB(Func)}(A) \leq \Pr(col) \leq \frac{(N-1) \cdot \sum_{i=1}^{i=N} k_i}{2^l} \quad (2)$$

where $\Pr(col)$ is the probability of a collision among the counter values used as inputs of the block cipher during the game between the adversary *A* and the encryption scheme. *N* is the number of sync cycles performed by the PSCFB encryption scheme during the game, *l* is the input block size in bits, $2^l$ is the number of possible counter values in binary radix and an *l*-bit size, and finally $k_i$ is the length in data blocks for the i-th sync cycle.

In this work the underlying block cipher is an FPE block cipher with radix *S,* then equation (2) can be written as:

$$ADV^{IND-CPA}_{PSCFB(Func)}(A) \leq \Pr(col) \leq \frac{(N-1) \cdot \sum_{i=1}^{i=N} k_i}{S^l} \quad (3)$$

As $k_i$ is the length in data blocks for the i-th sync cycle it is possible to write $k_i$ as $k_i = \lceil \mu_i / L \rceil \leq \mu_i / L + 1$ where $\mu_i$ is the number of symbols with radix *S* encrypted in the i-th sync cycle and *L* is the block size in symbols.

By using the same reasoning than [6] it is possible to write $\Pr(col)$ as:

$$\Pr(col) \leq \frac{{\mu_s}^2}{S^l} \cdot \frac{C + L}{C^2 \cdot L} \quad (4)$$

where $\mu_s$ is the total number of symbols encrypted during the game and $C = (P-1) \cdot L + l$. Moreover, if we consider that $L = l$, as the blockcipher has the same input and output size, then $C = P \cdot L$. According to this, equation (1) for PSCFB mode can be written as:

$$ADV^{IND-CPA}_{PSCFB(F)}(A) \leq 2 \cdot ADV^{PRF}_{F}(B) + \frac{{\mu_s}^2}{L^2 S^L} \cdot \frac{P+1}{P^2} \quad (5)$$

Block ciphers are usually analysed as PRFs and their prf-security is measured thanks to the $ADV^{PRF}_{F}(B)$ term. However, PRPs (Pseudo Random Permutations) are what best model them and it is possible to relate their prp-security and prf-security thanks to the PRF-PRP switching lemma [16]. According to this, the PRP security is degraded a term ${\mu_s}^2 / L^2 S^{L+1}$ in respect to PRF one. Therefore, equation (5) can also be expressed as:

$$ADV^{IND-CPA}_{PSCFB(F)}(A) \leq 2 \cdot ADV^{PRP}_{F}(B) + \frac{{\mu_s}^2}{L^2 S^L} + \\ + \frac{{\mu_s}^2}{L^2 S^L} \cdot \left(\frac{1}{P} + \frac{1}{P^2}\right) \quad (6)$$

where $ADV^{PRP}_{F}(B)$ is the prp-advantage of an adversary *B* over $F_k$. As the number of bits per symbol is $T = \log_2 S$, the total number of bits transmitted during the game will be $\mu = \mu_s \cdot T$. Finally the IND-CPA advantage equation for PSCFB can be rewritten as:

$$ADV^{IND-CPA}_{PSCFB(F)}(A) \leq 2 \cdot ADV^{PRP}_{F}(B) + \\ + \frac{\mu^2}{L^2 \cdot S^L \cdot T^2} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right) \quad (7)$$

In the scheme proposed in Fig. 4 we know that the underlying FPE block cipher has radix *S*=267, however, to get a complete parametrization it is also necessary to fix the block size *L* and the pipelining value *P*. For doing that we have compared the IND-CPA advantage of a traditional CTR scheme with the proposed mode. IND-CPA advantage for CTR mode can be expressed as:

$$ADV^{IND-CPA}_{CTRC(F)}(A) \leq 2 \cdot ADV^{PRP}_{F}(B) + \frac{\mu^2}{L^2 2^L} \quad (8)$$

Assuming that the PRP advantage $ADV^{PRP}_{F}(B)$ is negligible in CTR and PSCFB, and that in both modes the same amount of information is encrypted, we can compare the second term of equations (7) and (8) as follows:

$$\frac{\mu^2}{L^2_{PSCFB} \cdot S^{L_{PSCFB}} \cdot T^2} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right) \leq \frac{\mu^2}{L^2_{CTR} \cdot 2^{L_{CTR}}} \quad (9)$$

where $L_{PSCFB}$ and $L_{CTR}$ are the block sizes of the underlying block ciphers in PSCFB and CTR modes, respectively. Assuming that $L_{CTR}$ = 128 bits, as in AES (Advanced Encryption Standard) we can stablish the value of $L_{PSCFB}$ provided that:

$$L^2_{PSCFB} \cdot S^{L_{PSCFB}} \geq \frac{L^2_{CTR} \cdot 2^{L_{CTR}}}{T^2} \left(1 + \frac{1}{P} + \frac{1}{P^2}\right) \\ = \frac{2^{142}}{T^2} \cdot \left(1 + \frac{1}{P} + \frac{1}{P^2}\right) \quad (10)$$

As $P \geq 1$ the right term will be maximum with $P = 1$, then:

$$L^2_{PSCFB} \cdot S^{L_{PSCFB}} \geq \frac{2^{142} \cdot 3}{2 \log_2 S} \quad (11)$$

In the particular case of Gigabit Ethernet, $S$ = 267, therefore to meet the condition of equation (11) $L_{PSCFB} \geq 17$ symbols with radix *S*. By accomplishing the mentioned

TABLE I
FPGA HARDWARE RESOURCES

| Slice Registers | Slice LUTs | 18K Block RAMs | Slices[1] | Encryption Rate (Kbps) | Encryption Rate/Slice (Mbps/Slice) | Latency (ns) |
|---|---|---|---|---|---|---|
| 24979 | 26455 | 77 | 9737 | 1000 | 102.7 | 648 |

[1]Slices are estimated from the number of register and LUTs, assuming they are not packed together.

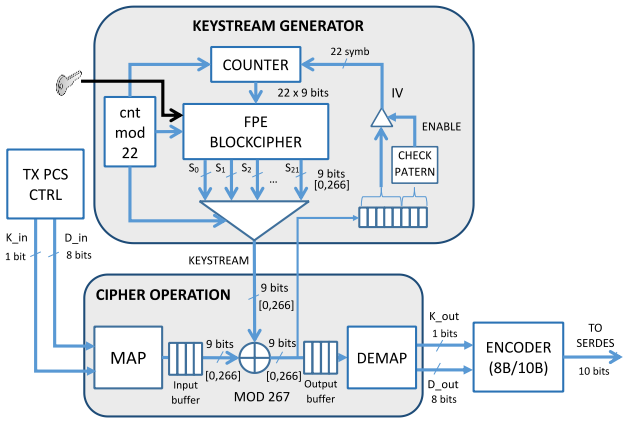



Fig. 5. Overall structure for PSCFB mode in a physical layer with 8b/10b encoding. Decryption will be as encryption but using a modulo-267 subtraction instead of an addition. In this structure, for each clock cycle a modulo-22 counter (called 'cnt_mod_22') selects one of the 22 symbols from the output block of the FPE block cipher, represented by the set $\{S_0, S_1, \ldots, S_{21}\}$. This counter and the multiplexer performs the function P/S in Fig. 4. The selected output symbol will be added to the incoming plaintext thanks to the modulo-267 addition. The FPE output block is refreshed every 22 cycles. The same happens with its input block that comes from CTR counter (called COUNTER). This COUNTER will also increment its value every 22 cycles.

condition it is possible to conclude that we can achieve the same or better IND-CPA security than a typical 128-bit block cipher working in CTR mode. In this work we have used the same structure for the FPE block cipher than in [4], it means that the block size is 22 symbols which meets the condition in (11).

## IV. IMPLEMENTATION

The final encryption structure has consisted of the PCS scheme shown in Fig. 1, where the FPE block cipher working in CTR mode has been substituted by the PSCFB operation mode shown in Fig. 4. The solution has been synthetized targeting a Virtex 7 FPGA platform, giving as result the hardware resources shown in Table I. Its final scheme is shown in Fig. 5. Furthermore, the encryption structure has been integrated in an Ethernet Interface as part of a PHY (Physical layer) and attached to a MAC (Medium Access Control) module as shown in Fig. 6. Simulations of Ethernet traffic while encrypting the link have been carried on using two Ethernet Interfaces facing each other.

## V. CONCLUSION

In this work, a new solution for self-synchronous symmetric encryption has been proposed for optical Gigabit Ethernet standard using 8b/10b encoding. An IND-CPA expression has been derived for the proposed encryption mode concluding that it can be considered secure as other traditional modes while keeping an encryption throughput near 100%. In

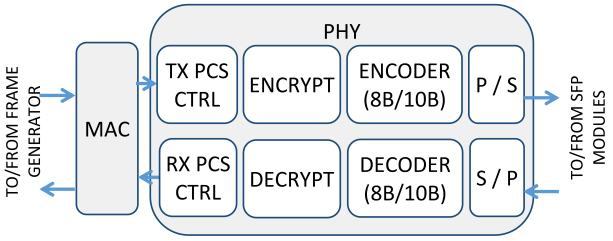



Fig. 6. Scheme of the Ethernet Interface formed by the PHY and MAC modules. ENCRYPT/DECRYPT modules contain the KEYSTREAM and CIPHER_OPERATION modules of Fig. 5. P/S and S/P modules are Parallel to Serial and Serial to Parallel modules, that transmit and receive the bitstream from the optical link.

addition, the solution has been synthetized targeting an FPGA platform, and simulated with Ethernet data frames.

## REFERENCES

[1] Cisco Systems, "Cisco 2018 Annual Cybersecurity Report," Feb. 2018.

[2] N. Skorin-Kapov, M. Furdek, S. Zsigmond and L. Wosinska, "Physical-Layer security in evolving optical networks," *IEEE Commun Mag.,* vol. 54, no. 8, pp. 110-117, Aug. 2016.

[3] K. Guan, J. Kakande and J. Cho, "On Deploying Encryption Solutions to Provide Secure Transport-as-a-Service (TaaS) in Core and Metro Networks," in *European Conference and Exhibition on Optical Communications*, Düsseldorf, September 18 – 22, 2016.

[4] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta and S. Celma, "Physical Layer Encryption for Industrial Ethernet in Gigabit Optical Links," *IEEE Transactions on Industrial Electronics,* June 2018.

[5] A. Pérez-Resa, M. Garcia-Bosque, C. Sanchez-Azqueta and S. Celma, "Chaotic Encryption for 10-Gb Ethernet Optical Links," *IEEE Tansactions On Circuits and Systems–I: Regular Papers,* p. doi: 10.1109/TCSI.2018.2867918, 2018.

[6] A. Pérez, M. Garcia-Bosque, C. Sanchez-Azqueta and S. Celma, "Self-synchronized Encryption for Physical Layer in 10Gbps Optical Links," *IEEE Transactions on Computers,* p. doi: 10.1109/TC.2018.2890259, 2019.

[7] MicroSemi, "In-flight Encryption in Service Provider Networks, No: PMC-2150716, Issue 2," 2016. [Online]. Available: https://pmcs.com/cgi-bin/document.pl?docnum=2150716.

[8] A. Pérez, M. Garcia-Bosque, C. Sánchez-Azqueta and S. Celma, "Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems," *IEEE Transactions on Instrumentation and Measurement,* p. doi:10.1109/TIM.2019.2896550, 2019.

[9] A. Klein, Stream Ciphers, London: Springer-Verlag, 2013.

[10] M. Dworkin, Recommendation for Block Cipher Modes of Operation: Methods for Format-Preserving Encryption, Gaithersburg, Maryland: NIST Special Publication 800-38G, 2016.

[11] M. Robshaw and O. Billet, New Stream Cipher Designs: The eSTREAM Finalists, Springer, 2008.

[12] J. L. J. P. B. Daemen, "Chosen ciphertext attack on SSS," 2005. [Online]. http://www.ecrypt.eu.org/stream/papersdir/044.pdf

[13] O. Jung and C. Ruland, "Encryption with Statistical Self-Synchronization in Synchronous Broadband Networks," in *Proc. Conf. Cryptographic Hardware and Embedded Systems (CHES '99)*, 1999.

[14] H. M. Heys and L. Zhang, "Pipelined Statistical Cipher Feedback: A New Mode for High-Speed Self-Synchronizing Stream Encryption," *IEEE Transactions on Computers,* vol. 60, no. 11, pp. 1581-1595, 2011.

[15] M. Bellware and P. Rogaway, "Introduction to modern cryptography," May 2005. [Online]. Available: http://web.cs.ucdavis.edu/~rogaway/classes/227/spring05/book/main.pdf.

[16] M. Bellware, A. Desai, E. Jokipii and P. Rogaway, "A concrete security treatement of symmetric encryption: Analysis of the DES modes of operation," in *Symposium on Foundations of Computer Science (FOCS)*, 1997.

# C

# Factores de impacto, áreas temáticas y justificación de la contribución por coautoría

Todos los coautores de los trabajos presentados en esta tesis doctoral, salvo el doctorando, son doctores. En la lista mostrada más abajo se recogen los factores de impacto de las revistas donde fueron publicados cada uno de los trabajos.

| Publicaciones | Factor de Impacto |
|---|---|
| A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaotic Encryption for 10-Gb Ethernet Optical Links". *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(2):859–868, Feb. 2019. | 3.934 |
| A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Physical Layer Encryption for Industrial Ethernet in Gigabit Optical Links". *IEEE Transactions on Industrial Electronics*, 66(4):3287–3295, April 2019. | 7.503 |
| A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-Synchronized Encryption for Physical Layer in 10Gbps Optical Links". *IEEE Transactions on Computers*, 68(6):899–911, June 2019. | 3.131 |
| A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems". *IEEE Transactions on Instrumentation and Measurement*, 68(12):4876–4886, Dec 2019. | 3.067 |
| A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "A New Method for Format Preserving Encryption in High-Data Rate Communications". *IEEE Access*, 8:21003–21016, 2020. | 4.098 |
| A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-synchronized Encryption for Physical Layer in 1Gbps Ethernet Optical Links". *IEEE Access*, Pending Acceptance. | N/A |
| A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-Synchronized Encryption Using an FPE Block Cipher for Gigabit Ethernet". In *2019 15th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, pages 81–84, Lausanne, Switzerland, July 2019. | N/A |

# 2018 Journal Performance Data for: IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I-REGULAR PAPERS

ISSN: 1549-8328
eISSN: 1558-0806
IEEE-INST ELECTRICAL ELECTRONICS ENGINEERS INC
445 HOES LANE, PISCATAWAY, NJ 08855-4141
USA

**TITLES**
ISO: IEEE Trans. Circuits Syst. I-Regul. Pap.
JCR Abbrev: IEEE T CIRCUITS-I

**LANGUAGES**
English

**CATEGORIES**
ENGINEERING, ELECTRICAL & ELECTRONIC - SCIE

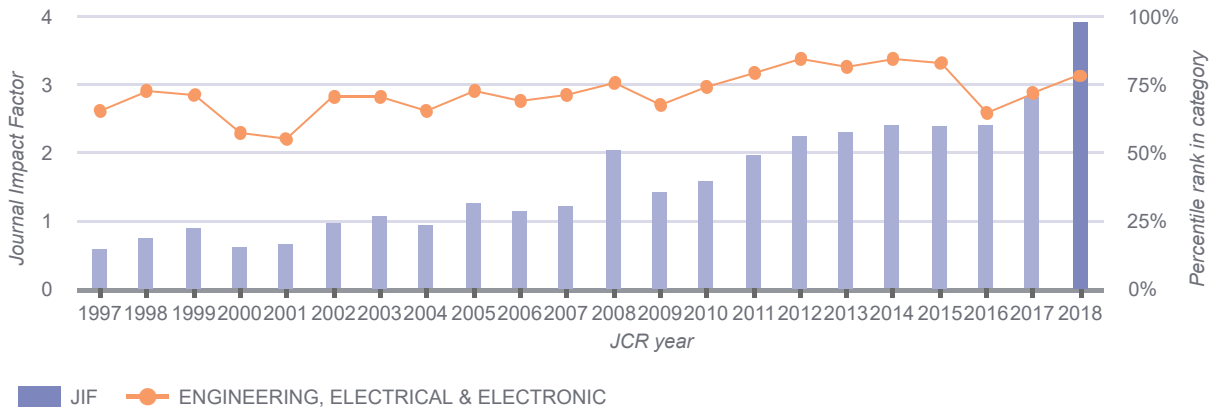
**PUBLICATION FREQUENCY**
**12 issues/year**

The data in the two graphs below and in the Journal Impact Factor calculation panels represent citation activity in 2018 to items published in the journal in the prior two years. They detail the components of the Journal Impact Factor. Use the "All Years" tab to access key metrics and additional data for the current year and all prior years for this journal.

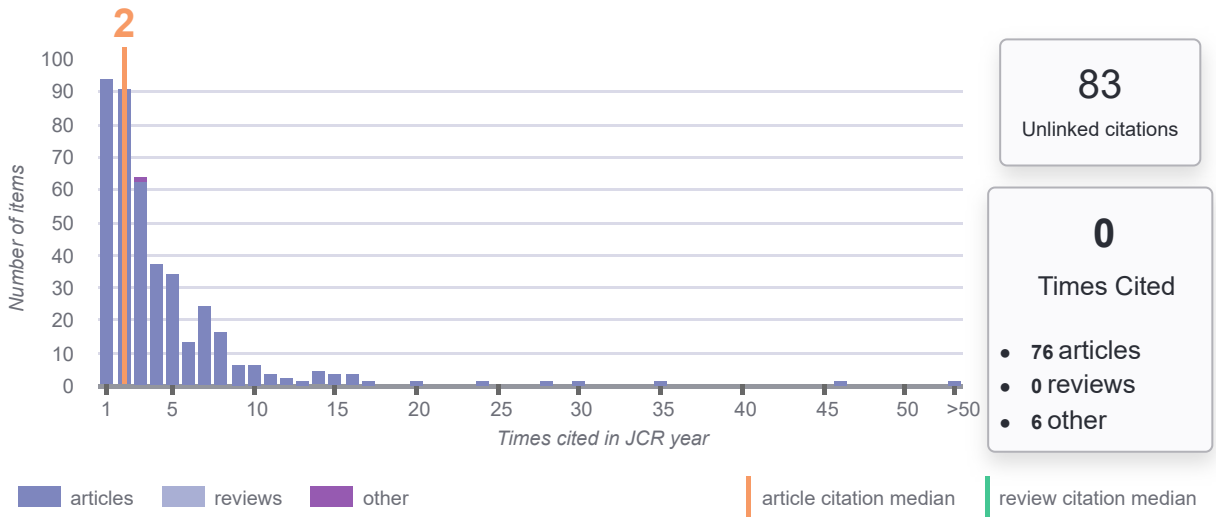
## 2018 Journal Impact Factor & percentile rank in category for: IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I-REGULAR PAPERS

**3.934**

2018 Journal Impact Factor




## 2018 JIF Citation Distribution for: IEEE TRANSACTIONS ON CIRCUITS AND SYSTEMS I-REGULAR PAPERS




**83**
Unlinked citations

**0**
Times Cited

- **76** articles
- **0** reviews
- **6** other

Rank

## Rank

### JCR Impact Factor

| JCR Year | ENGINEERING, ELECTRICAL & ELECTRONIC | | |
|---|---|---|---|
| | Rank | Quartile | JIF Percentile |
| 2018 | 57/265 | Q1 | 78.679 |
| 2017 | 74/260 | Q2 | 71.731 |
| 2016 | 93/262 | Q2 | 64.695 |
| 2015 | 44/257 | Q1 | 83.074 |
| 2014 | 39/249 | Q1 | 84.538 |
| 2013 | 46/248 | Q1 | 81.653 |
| 2012 | 39/243 | Q1 | 84.156 |
| 2011 | 52/245 | Q1 | 78.980 |
| 2010 | 64/247 | Q2 | 74.291 |
| 2009 | 80/246 | Q2 | 67.683 |
| 2008 | 56/229 | Q1 | 75.764 |
| 2007 | 66/227 | Q2 | 71.145 |
| 2006 | 64/206 | Q2 | 69.175 |
| 2005 | 57/208 | Q2 | 72.837 |
| 2004 | 73/209 | Q2 | 65.311 |
| 2003 | 61/205 | Q2 | 70.488 |
| 2002 | 61/203 | Q2 | 70.197 |
| 2001 | 91/200 | Q2 | 54.750 |
| 2000 | 88/204 | Q2 | 57.108 |
| 1999 | 59/205 | Q2 | 71.463 |
| 1998 | 58/208 | Q2 | 72.356 |
| 1997 | 67/193 | Q2 | 65.544 |

# 2018 Journal Performance Data for: IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS

ISSN: 0278-0046
eISSN: 1557-9948
IEEE-INST ELECTRICAL ELECTRONICS ENGINEERS INC
445 HOES LANE, PISCATAWAY, NJ 08855-4141
USA

**TITLES**
ISO: IEEE Trans. Ind. Electron.
JCR Abbrev: IEEE T IND ELECTRON

**LANGUAGES**
English

**CATEGORIES**
AUTOMATION & CONTROL SYSTEMS - SCIE

ENGINEERING, ELECTRICAL & ELECTRONIC - SCIE

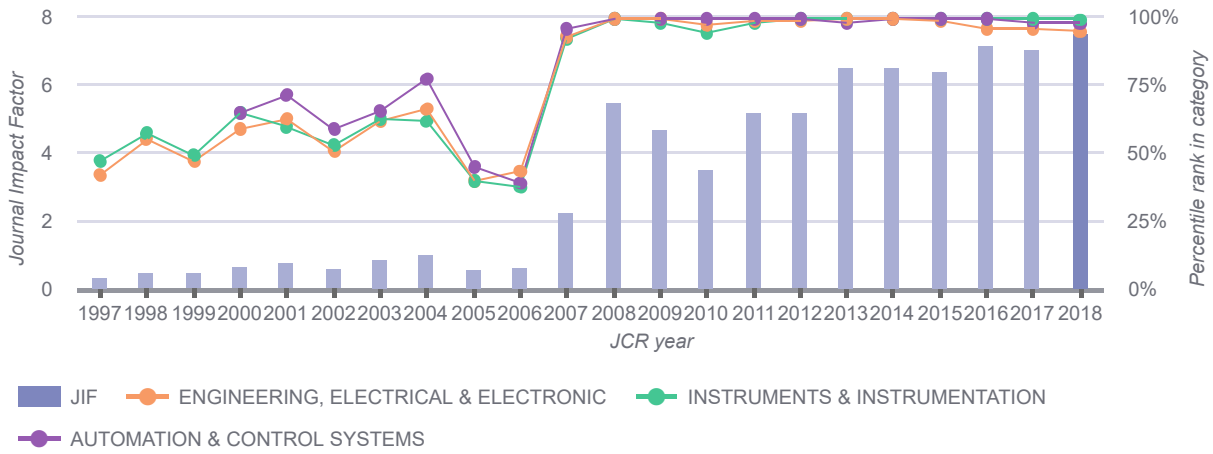
INSTRUMENTS & INSTRUMENTATION - SCIE

**PUBLICATION FREQUENCY**
**12 issues/year**

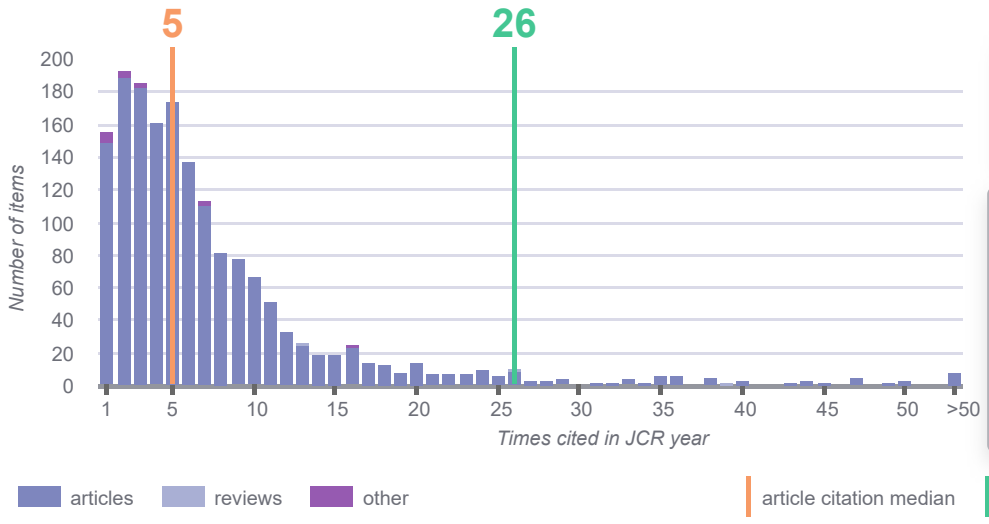
The data in the two graphs below and in the Journal Impact Factor calculation panels represent citation activity in 2018 to items published in the journal in the prior two years. They detail the components of the Journal Impact Factor. Use the "All Years" tab to access key metrics and additional data for the current year and all prior years for this journal.

## 2018 Journal Impact Factor & percentile rank in category for: IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS

**7.503**
2018 Journal Impact Factor




## 2018 JIF Citation Distribution for: IEEE TRANSACTIONS ON INDUSTRIAL ELECTRONICS




606
Unlinked citations

**0**
Times Cited

- **96** articles
- **0** reviews
- **21** other

Rank

## Rank

### JCR Impact Factor

| JCR Year | INSTRUMENTS & INSTRUMENTATION | | | AUTOMATION & CONTROL SYSTEMS | | | ENGINEERING, ELECTRICAL & ELECTRONIC | | |
|---|---|---|---|---|---|---|---|---|---|
| | Rank | Quartile | JIF Percentile | Rank | Quartile | JIF Percentile | Rank | Quartile | JIF Percentile |
| 2018 | 1/61 | Q1 | 99.180 | 2/62 | Q1 | 97.581 | 14/265 | Q1 | 94.906 |
| 2017 | 1/61 | Q1 | 99.180 | 2/61 | Q1 | 97.541 | 13/260 | Q1 | 95.192 |
| 2016 | 1/58 | Q1 | 99.138 | 1/60 | Q1 | 99.167 | 12/262 | Q1 | 95.611 |
| 2015 | 1/56 | Q1 | 99.107 | 1/59 | Q1 | 99.153 | 4/257 | Q1 | 98.638 |
| 2014 | 1/56 | Q1 | 99.107 | 1/58 | Q1 | 99.138 | 2/249 | Q1 | 99.398 |
| 2013 | 1/57 | Q1 | 99.123 | 2/59 | Q1 | 97.458 | 2/248 | Q1 | 99.395 |
| 2012 | 1/57 | Q1 | 99.123 | 1/59 | Q1 | 99.153 | 4/243 | Q1 | 98.560 |
| 2011 | 2/58 | Q1 | 97.414 | 1/58 | Q1 | 99.138 | 4/245 | Q1 | 98.571 |
| 2010 | 4/61 | Q1 | 94.262 | 1/60 | Q1 | 99.167 | 8/247 | Q1 | 96.964 |
| 2009 | 2/58 | Q1 | 97.414 | 1/59 | Q1 | 99.153 | 3/246 | Q1 | 98.984 |
| 2008 | 1/56 | Q1 | 99.107 | 1/53 | Q1 | 99.057 | 2/229 | Q1 | 99.345 |
| 2007 | 5/55 | Q1 | 91.818 | 3/52 | Q1 | 95.192 | 18/227 | Q1 | 92.291 |
| 2006 | 34/53 | Q3 | 36.792 | 32/51 | Q3 | 38.235 | 118/206 | Q3 | 42.961 |
| 2005 | 32/52 | Q3 | 39.423 | 26/46 | Q3 | 44.565 | 127/208 | Q3 | 39.183 |
| 2004 | 19/48 | Q2 | 61.458 | 11/46 | Q1 | 77.174 | 71/209 | Q2 | 66.268 |
| 2003 | 19/49 | Q2 | 62.245 | 17/47 | Q2 | 64.894 | 79/205 | Q2 | 61.707 |
| 2002 | 25/52 | Q2 | 52.885 | 21/49 | Q2 | 58.163 | 102/203 | Q3 | 50.000 |
| 2001 | 20/48 | Q2 | 59.375 | 14/47 | Q2 | 71.277 | 76/200 | Q2 | 62.250 |
| 2000 | 19/52 | Q2 | 64.423 | 16/44 | Q2 | 64.773 | 85/204 | Q2 | 58.578 |
| 1999 | 27/52 | Q3 | 49.038 | N/A | N/A | N/A | 110/205 | Q3 | 46.585 |
| 1998 | 23/52 | Q2 | 56.731 | N/A | N/A | N/A | 95/208 | Q2 | 54.567 |
| 1997 | 25/46 | Q3 | 46.739 | N/A | N/A | N/A | 113/193 | Q3 | 41.710 |

# 2018 Journal Performance Data for: IEEE TRANSACTIONS ON COMPUTERS

ISSN: 0018-9340
eISSN: 1557-9956
IEEE COMPUTER SOC
10662 LOS VAQUEROS CIRCLE, PO BOX 3014, LOS ALAMITOS, CA 90720-1314
USA

**TITLES**
ISO: IEEE Trans. Comput.
JCR Abbrev: IEEE T COMPUT

**LANGUAGES**
English

**CATEGORIES**
COMPUTER SCIENCE, HARDWARE & ARCHITECTURE - SCIE

ENGINEERING, ELECTRICAL & ELECTRONIC - SCIE

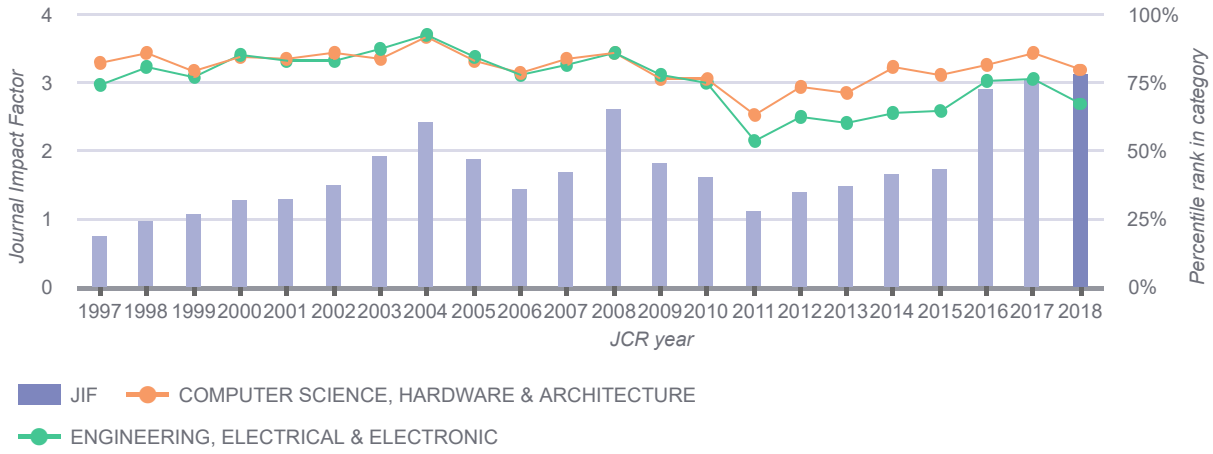
**PUBLICATION FREQUENCY**
**12 issues/year**

The data in the two graphs below and in the Journal Impact Factor calculation panels represent citation activity in 2018 to items published in the journal in the prior two years. They detail the components of the Journal Impact Factor. Use the "All Years" tab to access key metrics and additional data for the current year and all prior years for this journal.

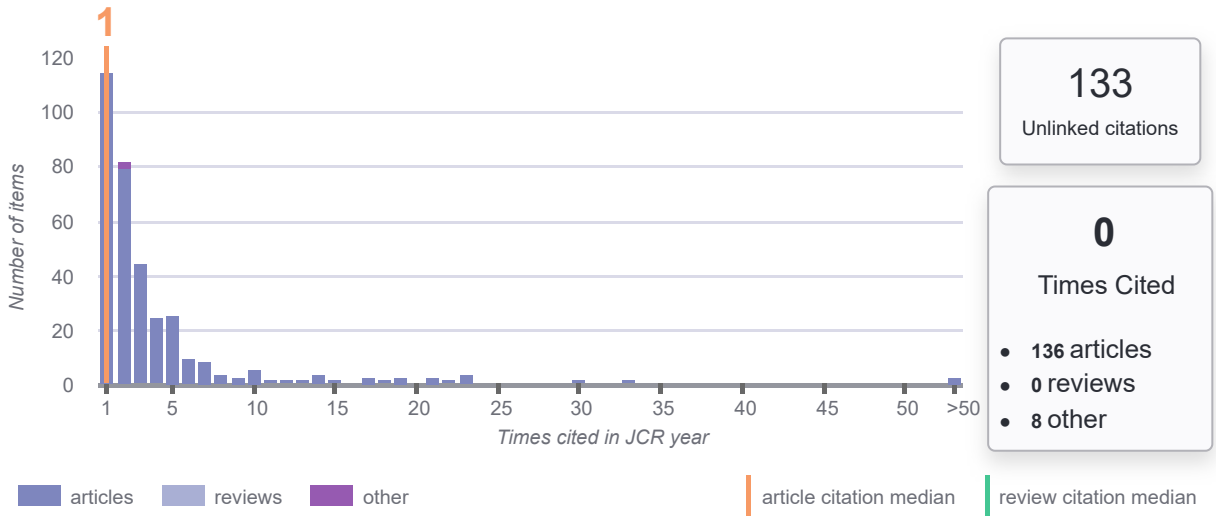
## 2018 Journal Impact Factor & percentile rank in category for: IEEE TRANSACTIONS ON COMPUTERS

**3.131**

2018 Journal Impact Factor




## 2018 JIF Citation Distribution for: IEEE TRANSACTIONS ON COMPUTERS




133
Unlinked citations

**0**
Times Cited

- **136** articles
- **0** reviews
- **8** other

Rank

## Rank

### JCR Impact Factor

| JCR Year | COMPUTER SCIENCE, HARDWARE & ARCHITECTURE | | | ENGINEERING, ELECTRICAL & ELECTRONIC | | |
|---|---|---|---|---|---|---|
| | Rank | Quartile | JIF Percentile | Rank | Quartile | JIF Percentile |
| 2018 | 11/52 | Q1 | 79.808 | 86/265 | Q2 | 67.736 |
| 2017 | 8/52 | Q1 | 85.577 | 63/260 | Q1 | 75.962 |
| 2016 | 10/52 | Q1 | 81.731 | 65/262 | Q1 | 75.382 |
| 2015 | 12/51 | Q1 | 77.451 | 92/257 | Q2 | 64.397 |
| 2014 | 10/50 | Q1 | 81.000 | 91/249 | Q2 | 63.655 |
| 2013 | 15/50 | Q2 | 71.000 | 99/248 | Q2 | 60.282 |
| 2012 | 14/50 | Q2 | 73.000 | 92/243 | Q2 | 62.346 |
| 2011 | 19/50 | Q2 | 63.000 | 114/245 | Q2 | 53.673 |
| 2010 | 12/48 | Q1 | 76.042 | 63/247 | Q2 | 74.696 |
| 2009 | 12/49 | Q1 | 76.531 | 55/246 | Q1 | 77.846 |
| 2008 | 7/45 | Q1 | 85.556 | 32/229 | Q1 | 86.245 |
| 2007 | 8/45 | Q1 | 83.333 | 42/227 | Q1 | 81.718 |
| 2006 | 10/44 | Q1 | 78.409 | 46/206 | Q1 | 77.913 |
| 2005 | 8/44 | Q1 | 82.955 | 33/208 | Q1 | 84.375 |
| 2004 | 4/44 | Q1 | 92.045 | 16/209 | Q1 | 92.584 |
| 2003 | 8/47 | Q1 | 84.043 | 27/205 | Q1 | 87.073 |
| 2002 | 7/46 | Q1 | 85.870 | 35/203 | Q1 | 83.005 |
| 2001 | 8/45 | Q1 | 83.333 | 35/200 | Q1 | 82.750 |
| 2000 | 8/49 | Q1 | 84.694 | 30/204 | Q1 | 85.539 |
| 1999 | 10/45 | Q1 | 78.889 | 47/205 | Q1 | 77.317 |
| 1998 | 7/47 | Q1 | 86.170 | 40/208 | Q1 | 81.010 |
| 1997 | 8/43 | Q1 | 82.558 | 51/193 | Q2 | 73.834 |

# 2018 Journal Performance Data for: IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT

ISSN: 0018-9456
eISSN: 1557-9662
IEEE-INST ELECTRICAL ELECTRONICS ENGINEERS INC
445 HOES LANE, PISCATAWAY, NJ 08855-4141
USA

**TITLES**
ISO: IEEE Trans. Instrum. Meas.
JCR Abbrev: IEEE T INSTRUM MEAS

**LANGUAGES**
English

**CATEGORIES**
ENGINEERING, ELECTRICAL & ELECTRONIC - SCIE

INSTRUMENTS & INSTRUMENTATION - SCIE

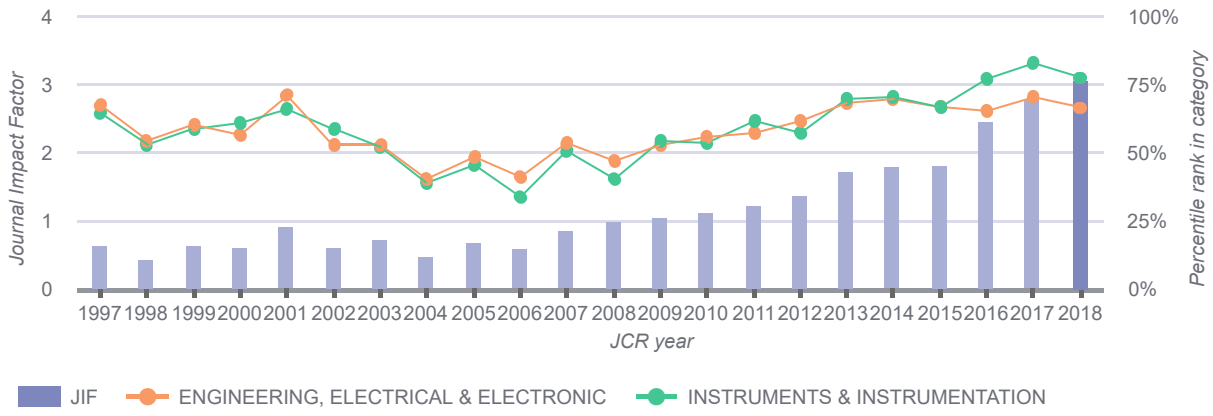
**PUBLICATION FREQUENCY**
**12 issues/year**

The data in the two graphs below and in the Journal Impact Factor calculation panels represent citation activity in 2018 to items published in the journal in the prior two years. They detail the components of the Journal Impact Factor. Use the "All Years" tab to access key metrics and additional data for the current year and all prior years for this journal.

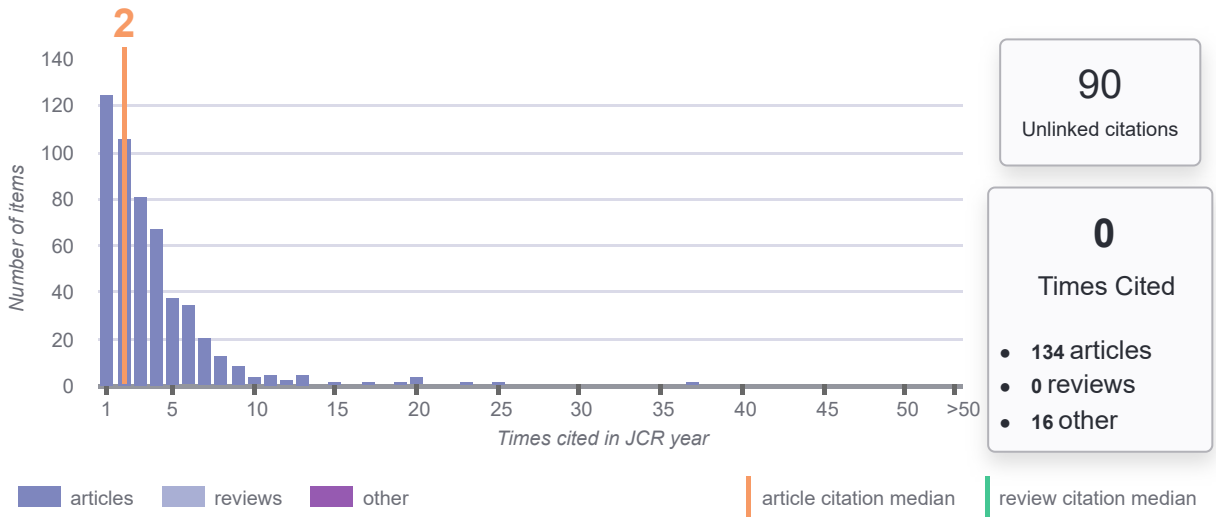
## 2018 Journal Impact Factor & percentile rank in category for: IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT

**3.067**

2018 Journal Impact Factor




## 2018 JIF Citation Distribution for: IEEE TRANSACTIONS ON INSTRUMENTATION AND MEASUREMENT




**90**
Unlinked citations

**0**
Times Cited

- **134** articles
- **0** reviews
- **16** other

Rank

## Rank

### JCR Impact Factor

| JCR Year | INSTRUMENTS & INSTRUMENTATION | | | ENGINEERING, ELECTRICAL & ELECTRONIC | | |
|---|---|---|---|---|---|---|
| | Rank | Quartile | JIF Percentile | Rank | Quartile | JIF Percentile |
| 2018 | 14/61 | Q1 | 77.869 | 88/265 | Q2 | 66.981 |
| 2017 | 11/61 | Q1 | 82.787 | 77/260 | Q2 | 70.577 |
| 2016 | 14/58 | Q1 | 76.724 | 91/262 | Q2 | 65.458 |
| 2015 | 19/56 | Q2 | 66.964 | 86/257 | Q2 | 66.732 |
| 2014 | 17/56 | Q2 | 70.536 | 76/249 | Q2 | 69.679 |
| 2013 | 18/57 | Q2 | 69.298 | 80/248 | Q2 | 67.944 |
| 2012 | 25/57 | Q2 | 57.018 | 94/243 | Q2 | 61.523 |
| 2011 | 23/58 | Q2 | 61.207 | 106/245 | Q2 | 56.939 |
| 2010 | 29/61 | Q2 | 53.279 | 111/247 | Q2 | 55.263 |
| 2009 | 27/58 | Q2 | 54.310 | 117/246 | Q2 | 52.642 |
| 2008 | 34/56 | Q3 | 40.179 | 122/229 | Q3 | 46.943 |
| 2007 | 28/55 | Q3 | 50.000 | 106/227 | Q2 | 53.524 |
| 2006 | 36/53 | Q3 | 33.019 | 122/206 | Q3 | 41.019 |
| 2005 | 29/52 | Q3 | 45.192 | 109/208 | Q3 | 47.837 |
| 2004 | 30/48 | Q3 | 38.542 | 126/209 | Q3 | 39.952 |
| 2003 | 24/49 | Q2 | 52.041 | 97/205 | Q2 | 52.927 |
| 2002 | 22/52 | Q2 | 58.654 | 97/203 | Q2 | 52.463 |
| 2001 | 17/48 | Q2 | 65.625 | 59/200 | Q2 | 70.750 |
| 2000 | 21/52 | Q2 | 60.577 | 89/204 | Q2 | 56.618 |
| 1999 | 22/52 | Q2 | 58.654 | 83/205 | Q2 | 59.756 |
| 1998 | 25/52 | Q2 | 52.885 | 96/208 | Q2 | 54.087 |
| 1997 | 17/46 | Q2 | 64.130 | 63/193 | Q2 | 67.617 |

# 2018 Journal Performance Data for: IEEE Access

ISSN: 2169-3536
eISSN: 2169-3536
IEEE-INST ELECTRICAL ELECTRONICS ENGINEERS INC
445 HOES LANE, PISCATAWAY, NJ 08855-4141
USA

**TITLES**
ISO: IEEE Access
JCR Abbrev: IEEE ACCESS

**LANGUAGES**
English

**CATEGORIES**

COMPUTER SCIENCE, INFORMATION SYSTEMS - SCIE

ENGINEERING, ELECTRICAL & ELECTRONIC - SCIE

TELECOMMUNICATIONS - SCIE

**PUBLICATION FREQUENCY**
**1 issue/year**

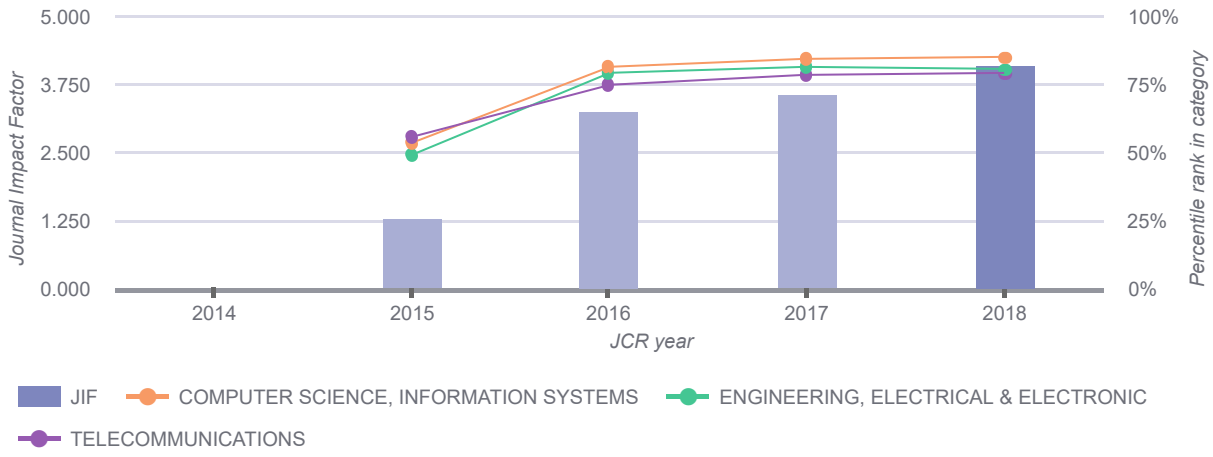
Open Access from 2013

The data in the two graphs below and in the Journal Impact Factor calculation panels represent citation activity in 2018 to items published in the journal in the prior two years. They detail the components of the Journal Impact Factor. Use the "All Years" tab to access key metrics and additional data for the current year and all prior years for this journal.

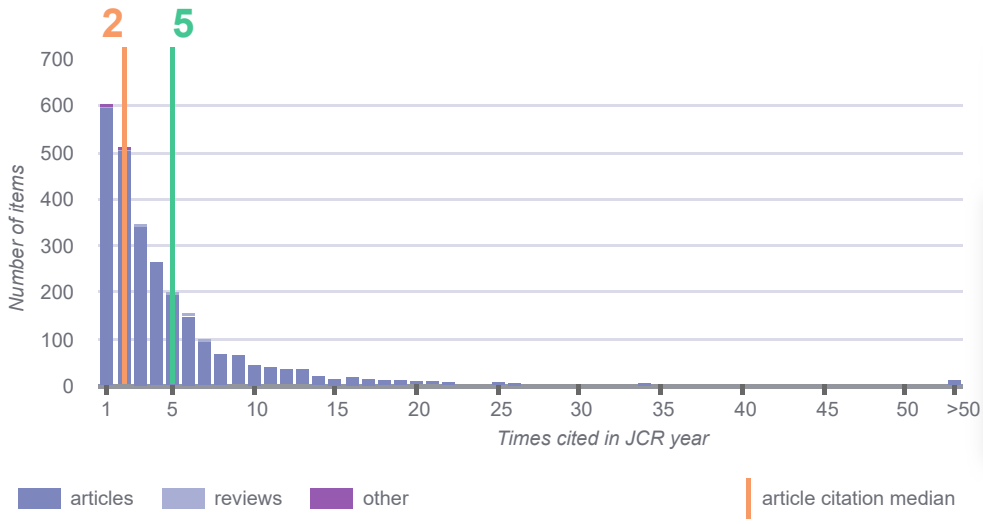
## 2018 Journal Impact Factor & percentile rank in category for: IEEE Access

**4.098**

2018 Journal Impact Factor




## 2018 JIF Citation Distribution for: IEEE Access




556
Unlinked citations

**0**
Times Cited

- **547** articles
- **5** reviews
- **17** other

Rank

## Rank

### JCR Impact Factor

| JCR Year | TELECOMMUNICATIONS | | | COMPUTER SCIENCE, INFORMATION SYSTEMS | | | ENGINEERING, ELECTRICAL & ELECTRONIC | | |
|---|---|---|---|---|---|---|---|---|---|
| | Rank | Quartile | JIF Percentile | Rank | Quartile | JIF Percentile | Rank | Quartile | JIF Percentile |
| 2018 | 19/88 | Q1 | 78.977 | 23/155 | Q1 | 85.484 | 52/266 | Q1 | 80.639 |
| 2017 | 19/87 | Q1 | 78.736 | 24/148 | Q1 | 84.122 | 48/260 | Q1 | 81.731 |
| 2016 | 23/89 | Q2 | 74.719 | 27/146 | Q1 | 81.849 | 54/262 | Q1 | 79.580 |
| 2015 | 37/82 | Q2 | 55.488 | 68/144 | Q2 | 53.125 | 131/257 | Q3 | 49.222 |

# D

# Lista de Publicaciones del Autor

## Publicaciones en Revistas


[PER19a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaotic Encryption Applied to Optical Ethernet in Industrial Control Systems". *IEEE Transactions on Instrumentation and Measurement*, 68(12):4876–4886, Dec 2019.

[PER19b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Physical Layer Encryption for Industrial Ethernet in Gigabit Optical Links". *IEEE Transactions on Industrial Electronics*, 66(4):3287–3295, April 2019.

[PER19c] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaotic Encryption for 10-Gb Ethernet Optical Links". *IEEE Transactions on Circuits and Systems I: Regular Papers*, 66(2):859–868, Feb. 2019.

[PER19d] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-Synchronized Encryption for Physical Layer in 10Gbps Optical Links". *IEEE Transactions on Computers*, 68(6):899–911, June 2019.

[PER20a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "A New Method for Format Preserving Encryption in High-Data Rate Communications". *IEEE Access*, 8:21003–21016, 2020.

[PER20b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-synchronized Encryption for Physical Layer in 1Gbps Ethernet Optical Links". *IEEE Access*, Pending Acceptance.

[GAR17b] M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, and S. Celma. "Application of a MEMS-Based TRNG in a Chaotic Stream Cipher". *Sensors*, 17(3):646, Mar. 2017.

[GAR18a] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, and S. Celma. A new simple technique for improving the random properties of chaos-based cryptosystems. *AIP Advances*, 8(3):035004, 2018.

[GAR19a] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, and S. Celma. Chaos-based bitwise dynamical pseudorandom number generator on fpga. *IEEE Transactions on Instrumentation and Measurement*, 68(1):291–293, Jan 2019.

[GAR19b] Miguel Garcia-Bosque, Guillermo Díez-Señorans, Adrián Pérez-Resa, Carlos Sánchez-Azqueta, Concepción Aldea, and Santiago Celma. A 1 gbps chaos-based stream cipher implemented in 0.18 µm cmos technology. *MDPI Electronics*, 8(6), 2019.

## Publicaciones en Congresos Internacionales

[PER18a] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaos-Based Stream Cipher for Gigabit Ethernet". In *2018 IEEE 9th Latin American Symposium on Circuits Systems (LASCAS)*, pages 1–4, Feb 2018.

[PER18b] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Using a Chaotic Cipher to Encrypt Ethernet Traffic". In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2018.

[PER18c] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Chaotic Encryption Applied to 10Gbps Ethernet Optical Links". In *Proceedigs of the 2018 International Symposium on Nonlinear Theory and its Applications (NOLTA 2018)*, pages 639–642, Tarragona, Spain, Sept. 2018.

[PER19e] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Self-Synchronized Encryption Using an FPE Block Cipher for Gigabit Ethernet". In *2019 15th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, pages 81–84, Lausanne, Switzerland, July 2019.

[PER20c] A. Pérez-Resa, M. Garcia-Bosque, C. Sánchez-Azqueta, and S. Celma. "Security Analysis of a New FPE Stream Cipher". In *2020 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2020.

[GAR17a] M. Garcia-Bosque, C. Sánchez-Azqueta, A. Pérez, A. D. Martínez, and S. Celma. "Fast and Secure Chaotic Stream Cipher with a MEMS-based Seed Generator". In *2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6, May 2017.

[GAR17c] M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, G. Royo, and S. Celma. "MEMS-based seed generator applied to a chaotic stream cipher". In Luis Fonseca, Mika Prunnila, and Erwin Peiner, editors, *Smart Sensors, Actuators, and MEMS VIII*, volume 10246, pages 303 – 308. International Society for Optics and Photonics, SPIE, 2017.

[GAR17d] M. Garcia-Bosque, C. Sánchez-Azqueta, A. Pérez, A. D. Martínez, and S. Celma. "Fast and secure chaotic stream cipher with a MEMS-based seed generator". In *2017 IEEE International Instrumentation and Measurement Technology Conference (I2MTC)*, pages 1–6, May 2017.

[GAR17e] M. Garcia-Bosque, A. Pérez, C. Sánchez-Azqueta, and S. Celma. "A new multiple ciphering scheme for improving randomness". In *2017 European Conference on Circuit Theory and Design (ECCTD)*, pages 1–4, Sep. 2017.

[GAR18b] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, and S. Celma. "A new randomness-enhancement method for chaos-based cryptosystem". In *2018 IEEE 9th Latin American Symposium on Circuits Systems (LASCAS)*, pages 1–4, Feb 2018.

[GAR18c] M. Garcia-Bosque, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, and S. Celma. "A New Technique For Improving the Security of Chaos Based Cryptosystems". In *2018 IEEE International Symposium on Circuits and Systems (ISCAS)*, pages 1–5, May 2018.

[GAR18d] M. Garcia-Bosque, G. Díez-Señorans, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, and S. Celma. "A Multi-Encryption Scheme and its Application to Improve the Random Properties of a Chaos-Based Stream Cipher". In *Proceedigs of the 2018 International Symposium on Nonlinear Theory and its Applications (NOLTA 2018)*, Tarragona, Spain, Sept. 2018.

[GAR19c] M. Garcia-Bosque, G. Díez-Señorans, A. Pérez-Resa, C. Sánchez-Azqueta, C. Aldea, and S. Celma. "A New Lightweight CSPRNG Implemented in a 0.18µm CMOS Technology". In *2019 15th Conference on Ph.D Research in Microelectronics and Electronics (PRIME)*, pages 221–224, July 2019.