LPsec: A Fast and Secure Cryptographic System for Optical Connections

M. IQBAL[1], L. VELASCO[1*], N. COSTA[2], A. NAPOLI[3], J. PEDRO[2,4], AND M. RUIZ[1]

[1]*Optical Communications Group, Universitat Politècnica de Catalunya, Barcelona, Spain.*
[2]*Infinera Unipessoal Lda, Lisbon, Portugal.*
[3]*Infinera Ltd., London, UK.*
[4]*Instituto de Telecomunicações, Instituto Superior Técnico, Portugal.*
**Corresponding author: luis.velasco@upc.edu*

The high capacity and low latency of optical connections are ideal for supporting the current and future communication services, including 5G and beyond. Although some of those services are already secured at the packet layer using standard stream ciphers, like Advanced Encryption Standard (AES) and ChaCha, secure transmission at the optical layer is still not implemented. To secure the optical layer, cryptographic methods need to be fast enough to support high-speed optical transmission and cannot introduce significant delay. Moreover, methods for key exchange, key generation and key expansion are required, which can be implemented on standard coherent transponders. In this paper, we propose *Light Path SECurity* (LPsec), a secure cryptographic solution for optical connections that involves fast data encryption using stream ciphers and key exchange using Diffie-Hellman (DH) protocol through the optical channel. To support encryption of high-speed data streams, a fast, general purpose Pseudo-Random Number Generator (PRNG) is used. Moreover, to make the scheme more secure against exhaustive search attacks, an additional substitution cipher is proposed. In contrast to the limited encryption speeds that standard stream ciphers can support, LPsec can support high-speed rates. Numerical simulation for 16-QAM, 32-QAM and 64-QAM show that LPsec provides sufficient security level while introducing negligible delay only.

1. INTRODUCTION

Optical networks are vulnerable to a variety of attacks such as eavesdropping, physical infrastructure attacks, interception, and jamming (refer to [1] for a survey on the topic). Most of the previous works focused on the upper layers, leaving the optical layer for pure transport. However, the security of the optical layer should not be overlooked, as building a secure platform on top of an unsecure one is a risky practice.

Designing a security solution involves combining multiple technologies for key distribution and data encryption and decryption, among others. For the former, although Quantum Key Distribution (QKD) provides secure key agreement and initial experimental deployments of QKD are being currently reported (see, e.g., [2]), it is not expected to be available in the short term. With regard to encryption, there are several solutions, from *stream ciphers* (each incoming plaintext digit is encrypted sequentially) to *block ciphers* (where plaintext digits are grouped into blocks and then encrypted together). Salsa20 and ChaCha [3] are examples of stream ciphers and Advanced Encryption Standard (AES) [4] is the most extended block cipher. However, one of the main challenges for securing the optical layer is that encryption and decryption need to operate at line speeds of 100s of Gb/s and should not introduce any meaningful delay to the optical transmission; otherwise, this would negatively impact the supported services.

In this regard, some works have already proposed encryption solutions for the optical layer. In the context of passive optical networks, the authors in [5] proposed the optical spectral phase and delay encoding technique, where the optical signal is split into multiple spectral slices that are multiplexed after applying a delay and phase shift to produce the encrypted signal. However, this method has a constraint of generating narrow spectral slices, putting stringent requirements on optical bandpass filters [6]. The authors in [7] reported a physical layer security method based on a piecewise chaotic permutation of symbols and subcarriers. The method relies on an initial key, thus requiring the implementation of a key exchange strategy.

Chaos-based optical transmission is another technique for encryption in the optical domain, where the signal to be

encrypted is converted into a noise-like signal by applying broadband chaos as optical carrier [8]. The resulting chaotic signal is sent to the receiver and decryption can only be performed by separating the chaotic carrier from the data signal at the receiver. Recently, authors in [9] have shown the chaos-based encryption at the physical layer for wavelength-division multiplexing (WDM) networks, where a gaussian modulated constant-amplitude random-phase light is used to generate the chaotic signal for encryption. The same signal is required for decryption and is sent over the transmission fiber, which reduces fiber capacity. The decryption also requires the synchronization between the sent and received signal. The authors in [10] proposed chaos-masking encryption techniques for long distance communication, where advanced digital signal processing is used to decrypt the signal. However, this scheme does not provide sufficient security if high confidentiality is required.

In access networks, Optical Code Division Multiple Access (OCDMA) provides intrinsic confidentiality through multiple access interface noise [11, 12]. Nonetheless, eavesdropping by coded waveform analysis was reported in [13] and data interception by differential detection was reported in [14]. Optical steganography has been proposed and demonstrated for WDM systems [15-17]. This technique hides the signal from the eavesdropper, e.g., through dispersion, to avoid interception. Another option is to carry the signal in Amplified Spontaneous Emission (ASE) noise, as proposed in [18].

Some vendors implement AES at the Optical Transport Network (OTN) layer [19], where key exchange can be carried out using header bytes of the Optical Data Unit (ODU) frame. Note that such a solution brings the additional requirement of implementing OTN, in addition to the intrinsic complexity of AES. In this case, AES is used as a block cipher that encrypts blocks of 128 bits.

In this paper, we propose LPsec, an approach that includes tailored solutions for both key exchange and encryption/decryption so they can be easily implemented at the optical layer. For the key exchange, we design a mechanism based on the Diffie-Hellman (DH) key exchange [20], where the initial public keys of the two end parties, i.e., the Transmitter (Tx) and the Receiver (Rx), are exchanged via the Software Defined Networking (SDN) controller and are periodically updated through the optical channel to enhance the security level. For encryption, we rely on two ciphers: *i*) a traditional stream cipher that uses a symmetrical key and *ii*) permutations of symbols. Each cipher has its drawbacks, but when combined they provide the required security level to encrypt data at 100s of Gb/s. Besides, LPsec exhibits negligible transmission delay.

The rest of the paper is organized as follows. Section 2 provides the necessary background on cryptography describing substitution ciphers, followed by the discussion of stream ciphers, block ciphers and key exchange mechanisms. Next, our proposal to secure optical connections (LPsec) is presented, where the proposed encryption scheme is described, and the key exchange is outlined. Section 3 presents the details of key exchange, including the initial one and the periodic key updates. Then, the symmetric key generation and its expansion are detailed. Security analysis based on the Pseudo-Random Number Generator (PRNG) is also discussed. Section 4 details the building blocks of LPsec, including optical encryption and key management. As the operations are governed by *Finite State Machines* (FSM), their construction at the Tx and Rx are also discussed. Illustrative results are presented in Section 5, including the introduced delay and security level against several attacks. Finally, Section 6 draws the main conclusions of this work.

2. SECURE OPTICAL LAYER

This section first introduces some basic cryptographic concepts, such as ciphers, key extension, and key exchange [21], that are used in the rest of the paper. Next, the cryptographic techniques that we are proposing to implement for securing the optical layer are described.

A. Background on Cryptography

In cryptography, a cipher is an algorithm that transforms a plaintext message (m) into a ciphertext (c) (encryption), and vice versa (decryption). There are several types of ciphers, e.g., a *substitution cipher* encrypts units (e.g., each letter in a text) of plaintext by replacing them with the ciphertext with the help of a key. The receiver performs the inverse process to recover the original plaintext. Although the number of substitution alphabets might be large, substitution ciphers can be broken by frequency analysis.

To show that a cryptosystem is secure, mathematical modeling and proofs are used to verify that it satisfies a set of security properties. In particular, the One-Time Pad (OTP) cipher shows *perfect security*, as proved by Shannon in [22]. OTP encrypts m using a key (k) with the same length of m (denoted as n), by just implementing a bitwise XOR operation. Two important properties of XOR are: *i*) if k is uniformly distributed on $\{0,1\}^n$ then, the ciphered message $m \oplus k$ is also uniformly distributed; and *ii*) the inverse operation (decryption) consists on applying the XOR function with the same (symmetric) key k, i.e., $m \oplus k \oplus k = m$. Although OTP shows perfect security, it does not fit well for stream ciphers, where the length of the messages tends to infinite. Semantic security provides a weaker notion of security that allows to build secure ciphers that use reasonably short keys. That entails splitting the data stream into chunks of data of predefined size. However, k cannot be reused from one data chuck to another, as that would reduce the security level. Nonetheless, k can be extended using a cryptographically secure PRNG to generate a sequence of stream keys (k_s).

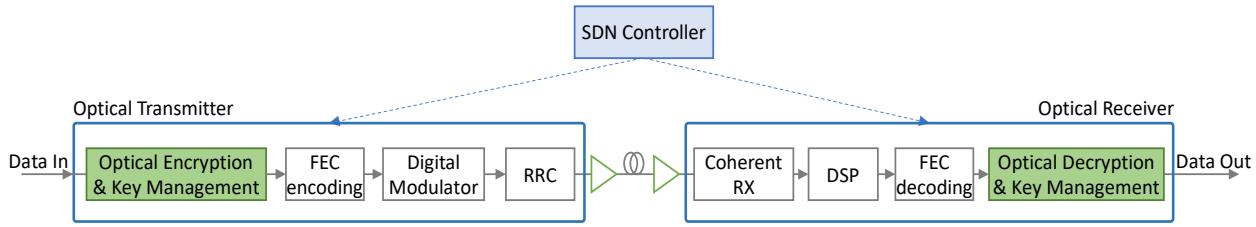

Fig. 1. Optical communication system considered for the implementation of LPsec.

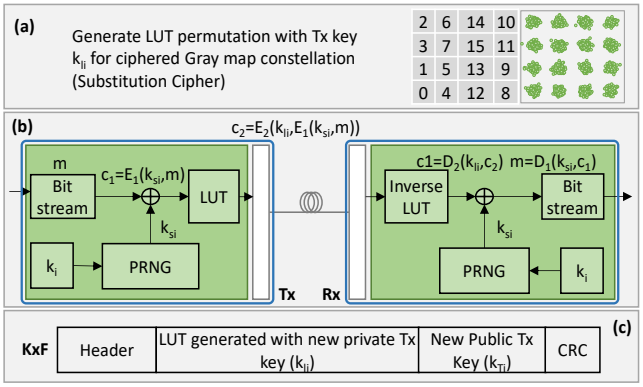

Fig. 2. Overview of LPsec. Example of 16-QAM LUT for encoding (a), encryption / decryption (b), and frame structure for periodical LUT and key synchronization (c).

Salsa and ChaCha are fast and secure stream ciphers that are appropriate for practical use and variants of them are being used in widely deployed protocols, such as Transport Layer Security (TLS) [23]. The PRNGs use a 256-bit seed, a 64-bit nonce, and a 64-bit counter to form a 512-bit block to create up to 2^{64} 512-bit pseudo random blocks. The design of these stream ciphers is highly parallelizable to speed-up encryption [3]. Block ciphers can be used as well to build a stream cipher; one popular block cipher is AES. In AES, an input block of 128 bits is processed as a 4×4-byte matrix. The AES algorithm performs 10, 12 or 14 rounds depending on the size of the cipher key (128, 192 or 256 bits). The process begins with the expansion of the initial key to produce a series of keys used in each round. At every round, the encryption begins by adding the round key as a XOR cipher followed by a non-linear byte substitution through a predetermined substitution table. Then, rows shifting followed by a mixing of columns and round key addition are performed. The procedure is repeated until completing the required number of rounds. Decryption is performed in the inverse manner.

The DH key exchange is a solution to exchange keys between two parties, Alice and Bob, that want to establish a secure communication channel. Both parties generate private (integer) keys k_p (i.e., k_a and k_b) and their related public keys k_P (i.e., k_A and k_B). The public keys are shared over the insecure channel and each party computes the symmetric key k that is used for data encryption using their own private key k_p and their counterpart's public key k_P.

B. Implementing LPsec in an Optical Coherent System

LPsec requires extending the standard coherent transponder with optical encryption and decryption blocks, as well as with some key management functionalities (see Fig. 1). In addition, cryptographic blocks need to operate at line speeds and should not introduce any significant delay to data transmission. To achieve such an objective, optical encryption should be based on simple operations performed on the input bit stream. The main design aspects of the cryptographic techniques proposed in this paper are analyzed hereafter.

As previously introduced, the encryption is based on two nested ciphers that provide a high security level. The outer cipher is a substitution cipher that relies on a Lookup Table (LUT) used for the substitution of bits before sending it to the modulator. This creates a ciphered gray map constellation through LUT permutations of incoming bits as suggested in Fig. 2a. Note that there are $M!$ permutations in an M-Quadrature Amplitude Modulation (QAM) system (e.g., there exist more than 2^{44} permutations in a 16-QAM system) and thus, we can use a random key (k_l) of the appropriate length (i.e., 44 bits in the example), to select the permutation of the LUT. The inner cipher is a stream cipher that encrypts data chunks of predefined size based on a cryptographically secure PRNG to generate a sequence of stream keys (k_s). The proposed encryption system is sketched in Fig. 2b, where output ciphertext c_2 is produced by the combination of the inner stream cipher E_1 and the outer substitution cipher E_2.

Note, however, that the sequence of stream keys $k_s = [k_{sj}]$ generated by the PRNG from a given key k cannot be infinite as this would reduce the security level of E_1. In addition, the LUT should be periodically regenerated to minimize the vulnerability of E_2. In consequence, we limit the lifetime of keys k and k_l, e.g., to 1 sec., which entails new keys being periodically generated at the Tx and exchanged with the Rx. Specifically, the DH key exchange method is used to generate the symmetric key k. The Tx and Rx generate a random private/public pair of keys ({<k_r, k_R>} and {<k_t, k_T>}) and exchange their public keys (k_T, k_R) with the other party. The initial key exchange can be facilitated by the SDN controller, which, once the optical connection is computed and established in the network [33], can collect the public keys and send them to the counterpart. However, symmetric keys should have a short lifetime and they need to be frequently updated, which makes the SDN not a suitable option.

In our approach, we perform a partial key exchange, where only the Tx generates a new pair of keys <k_{ti}, k_{Ti}>, as well as a new key k_{li} for the next period i. Next, both the public key k_{Ti} and the new permutation LUT(k_{li}) are sent to the Rx

through the optical channel. We propose to use a special frame (henceforth called Key exchange Frame, KxF) for the key exchange, (see Fig. 2c). A KxF is generated by the Tx and sent to the Rx periodically. The KxF includes a header of a fixed size that allows the Rx to detect its arrival. Because the Rx is not synchronized with the Tx for key exchange, any occurrence of the header pattern in the data stream must be prevented at the Tx side. Otherwise, the Rx would follow an erroneous key exchange procedure that will stop data transmission. The solution is to add escape bit sequences to break any KxF header pattern in the input data. To this end, two *Finite State Machines* (FSM) at the Tx and the Rx sides add and remove such escape bit sequences to/from the plain bit stream. Note that the occurrence of errors in the KxF is critical. In case errors still remain after the FEC decoding stage and impact the KxF, the decryption process should be restarted. For this very reason, the KxF includes a cyclic redundancy check (CRC).

The next sections detail the design of LPsec, how keys are generated and exchanged, and how the FSMs are defined and particularized.

3. KEYS AND SECURITY LEVEL

This section first details the key exchange process, including the initial exchange and the periodical updates. The generation and expansion of symmetric keys is detailed next, after which security level of the system is studied.

A. Key Exchange

As introduced in Section 2, an initial key exchange is performed through the SDN controller and then, the Tx updates the Rx with the keys to be used through the optical channel. Fig. 3 presents a sequence diagram detailing the computation performed by the Tx and Rx, as well as the messages exchanged through the control plane and the data and messages sent over the encrypted optical channel.

The initial key exchange is carried out at connection set-up through the SDN controller (messages 1-5 in Fig. 3), which collects the public key of the Rx (1) and sends it to the Tx (2). The Tx generates a pair of private and public keys and a random key k_{l0}, which is used to generate the initial LUT permutation. In addition, the Tx generates the symmetric key k_0 with its private key and Rx's public key. Key k_0 is used at this time to generate the particular KxF header pattern that will be used for key exchange on the optical channel. Both FSM_{Tx} and FSM_{Rx} must be generated for that specific pattern. Before sharing the LUT, it is encrypted with symmetric key k_0 and sent together with the Tx public key to the SDN controller (3), which shares them with the Rx (4). Upon the reception, the Rx generates the symmetric key k_0 with its private key and Tx public key, generates the FSM, and decrypts the LUT. The Rx replies to the SDN controller when it is ready to start the secure communication and the SDN controller notifies the Tx (5), which generates a new set of private and public keys, the LUT, and the symmetric key for

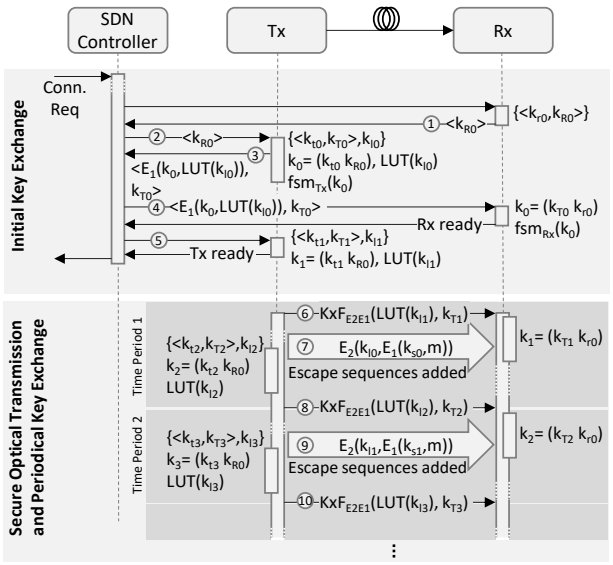

Fig. 3. Connection set-up and secure optical transmission

the next time interval. Then, the Tx replies to the controller that it is also ready, and the initialization phase concludes. At this time, the secure optical connection is established.

Once the initialization phase ends, the secure optical transmission phase begins and continues until the connection is torn down. At the starting time, the Tx updates the LUT and public key (6). Note that such exchange is encrypted using the nested encryption used for data transmission; in this case, keys k_{s0} (extended from symmetric key k_0) and k_{l0} are used for ciphers E_1 and E_2, respectively (6).

For the sake of clarity, we denote with subindex *i* the keys that participate in key exchange at the starting of time period *i*. This entails that during every time period *i-1*, the Tx generates the set of public and private keys ($<k_{ti}, k_{Ti}>$), the symmetric key k_i, and the random key k_{li} (and the permutation of the LUT). Then, at the start of time interval *i*, the Tx updates the Rx, which computes symmetric key k_i. The exchanged keys will be in place during time period *i*+1. In particular, key exchange *i* and all data transmitted during time period *i* are encrypted using keys *i*-1.

B. Symmetric Key Generation and Expansion

In the standard DH key exchange, two large prime numbers (p and g) are publicly selected. When Alice and Bob want to setup a secure communication channel, they generate private keys k_P, which are used to compute their public keys k_P, as:

$$k_P = g^{k_P} \bmod p \qquad (1)$$

After the public keys are exchanged, each party computes the symmetric key k for data encryption/decryption, as:

$$k = {k_B}^{k_a} \bmod p = {k_A}^{k_b} \bmod p \qquad (2)$$

Similarly, in LPsec, Tx and Rx exchange their public keys k_{T0} and k_{R0} through the SDN controller during the initial key exchange phase. Next, the Tx computes a new pair of keys every period and updates the Rx with the new public key k_{Ti}.

$$k_{R0} = g^{k_{r0}} \bmod p \qquad (3)$$

$$k_{Ti} = g^{k_{ti}} \bmod p \quad \forall i \tag{4}$$

Note that the Rx does not compute new public and private keys afterwards, and the initial pair $<k_{r0}, k_{R0}>$ is used along the lifetime of the optical connection. Therefore, the symmetric key k that is used for the stream cipher is updated periodically (e.g., every 1 sec.) as:

$$k_i = {k_{R0}}^{k_{ti}} \bmod p = {k_{Ti}}^{k_{r0}} \bmod p \tag{5}$$

Once the symmetric key is computed, it is expanded using a cryptographically secure PRNG to produce keys long enough for the stream cipher to encrypt / decrypt a chunk of data. Therefore, if the size of each chunk of data is U [b] and the transmission speed is B [b/s] then the number of chunks per second, J, can be computed as:

$$J = \frac{B}{U} \tag{6}$$

Hence, each symmetric key k_i generated for the time interval i, is expanded into J k_{sij} keys that the stream cipher will use for chunks j in [0 .. J-1]; keys k_{sij} are U bits long. For example, assuming that the size of data chunks is U=64 bits, the transmission speed is B=100 Gb/s, a new key is generated every 1 second, the PRNG needs to expand the symmetric key k into J=$2^{30.5}$ keys k_{sij} per second (i.e., one key every 0.64 ns), each U bits long. Therefore, the PRNG must be fast enough to work at 100s of Gb/s line speeds, in order not to introduce meaningful delay to data transmission.

C. Security Level and Encryption Speed

Let us assume that, under the DH protocol, an eavesdropper (i.e., Eve) knows the value of p, g, and the public keys of Alice and Bob. To compute the symmetric key, Eve still needs to know the private keys of either Alice or Bob, or to solve the discrete logarithm problem [24], which is considered computationally hard when p is large. However, the security level of a stream cipher depends on the randomness of the PRNG for key expansion [21]. Recall that OTP shows perfect security since ciphertexts do not reveal any information of the related plaintext, so an adversary cannot distinguish between two ciphertexts m_i and m_j encrypted with key k selected at random. However, stream ciphers cannot attain perfect security because PRNGs are utilized, and the length of the generated keys are shorter than those of the messages.

A PRNG is secure if an adversary cannot distinguish between a truly random sequence and the pseudo random sequence generated by the PRNG with a significant advantage. This is related to the computational feasibility of adversaries to perform predictions with a reasonable amount of time and memory. In practice, although standard stream ciphers based on Salsa20 and ChaCha produce high-quality PRNGs, they are not fast enough to be applied to optical transmission. In contrast, we use a general-purpose PRNG because of its high speed. To mitigate the impact of using a general-purpose PRNG only, an optical constellation-based substitution cipher is added. This approach is still not perfectly secure, as the distribution of the encrypted data is not uniform. Therefore, if the characteristics of the plain text are known, an adversary can apply frequency analysis and break the cipher. However, since the substitution cipher is fed with data encrypted using the XOR operation, frequency analysis will not provide useful information. As a consequence, this symbiotic relationship between the stream cipher and the substitution cipher results in a fast and secure cryptographic system.

4. DESIGN OF LPsec

In this section, we design the blocks for optical encryption / decryption and key management in terms of interconnected modules, which are governed by FSMs. We define generalized templates for the FSMs governing Tx and Rx, which enables the definition of random KxF header patterns both in contents and length. The generation of specific FSMs is detailed.

A. Optical Encryption / Decryption and Key Management

Fig. 4 presents a detailed design of the Optical Encryption and Key Management block at the Tx and that for Decryption and Key Management at the Rx. The Tx receives as input the data bit stream. Each individual plaintext digit is temporarily stored in a register while being checked by the FSM_{Tx} to prevent KxF header patterns. The encryption and decryption blocks perform operations over sets of bits, named *character* (char), where their size (b) coincides with the number of bits per symbol of the M-QAM modulation format used for the optical signal, e.g., b = 4 bits/symbol for 16-QAM (note that M=2^b). At every clock cycle during normal operation, the FSM_{Tx} reads one char from the input register (labeled *Da* in Fig. 4) and executes an internal state transition, which generates a tuple <*Sh*, *Se*, *Xa*> as output, where: *i*) *Sh* performs a char-size shift operation on the input register; *ii*) *Se* selects the input that is chosen as output in the selector; and *iii*) *Xa* is active during key exchange. In the case that a KxF header pattern is detected in the input data stream, an escape char is inserted, so the input *esc* in the selector will be chosen. At regular intervals, a new key exchange is initiated, so the Key Exchange module activates the *Kx* input on the FSM_{Tx}, and the KxF is transmitted instead of input data. Every char in the output of the selector is encrypted by stream cipher E_1 using a char from key $k_{s(i-1)j}$; the key register shifts one char every cycle and when it is empty a new key $k_{s(i-1)j}$ is expanded and loaded. Once a char is encrypted (c_1), it is used as input for the LUT and the substituted char (c_2) is generated.

The Key Generator module is responsible for generating new keys. At every time interval, the module generates a new pair of Tx public and private keys and uses the public key from the Rx to generate a new symmetric key. It also generates a new random key and selects the LUT permutation. The generated keys and LUT are sent to: *i*) the Key Exchange module that packs the LUT and the Tx public key in a KxF and activate the Kx signal to stop data transmission and start the key update; *ii*) the PRNG module that uses the symmetric

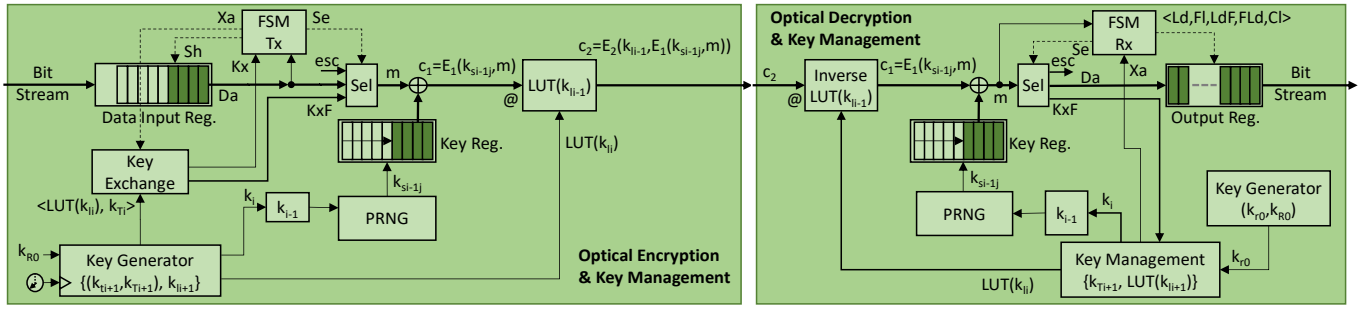

Fig. 4. Design of LPsec: Optical Encryption/Decryption and Key Management at the Tx/Rx.

key for expanding stream keys for the next chunk of data; and *iii*) the LUT module that uses the new LUT to update its contents.

At the Rx, the inverse process is performed. Every c_2 char received enters in the LUT and the original char c_1 is generated, which is then decrypted using a char from key $k_{s(i-1)j}$. The FSM_{Rx} inspects the plain chars (input *m*) in the search of escape chars being inserted by the Tx and generates a tuple *<Se*, *Ld*, *Fl*, *LdF*, *FLd*, *Cl>* as output. The plain chars are chosen at the output of the selector (signal *Se*) and can be temporarily stored in an output register (signal load, *Ld*) until a decision is made to send them as output bit stream (signal flush, *Fl*) or ignore them (signal clear, *Cl*). For convenience, the signals Load and Flush (*LdF*) and Flush and load (*FLd*) are defined. When the KxF header pattern is detected, the stored chars are ignored and the payload of the KxF is sent to the Key Management module. On the contrary, if an escape character is detected, it is ignored by choosing the *esc* output in the selector, and the output register is flushed.

When a KxF header pattern is detected, the Key Management module receives it and announces the FSM_{Rx} its end (signal *Xa*). The Key Management module will then distribute the received LUT to the LUT module and generate the new symmetric key to be distributed to the PRNG module.

B. Generalized FSM Templates

Specific FSMs need to be defined for the Tx and the Rx as a function of the generated KxF header pattern, which is generated at random during the initial key exchange phase. For the sake of the generalization of the FSMs, the header pattern is generated guaranteeing that any char does not appear more than once. In this way, the specific FSMs can be easily built from predefined parameterized templates by specifying the size of the header pattern and the specific chars.

Fig. 5a illustrates a graph representing the parameterized template of FSM_{Tx}, where the *n*-char header pattern is specified by the char sequence $H = <h_0, h_1, .., h_k, .., h_{n-1}, h_n>$. In the graph, states S_{Tx} define the outputs (O_{Tx}) for *Sh*, *Se*, and *Xa* signals, while transitions among states are performed based on the char in the input *Da* and *Kx* (I_{Tx}). Normal operation is represented by states S_i, whereas key periodical key exchange is represented by states S_i'. The graph consists of $n+1$ S_i and n S_i' states, where S_i represents the state where *i* chars in the header sequence and in the right order have been detected. A value '-' in one of the inputs means whatever other value, different than those specified for the rest of transitions leaving from that state. State S_0, the initial state, and state S_1 are the most frequently transitioned, and many of the rest of the states have a direct transition to them. The remaining normal operation states account for partial header patterns found in the input data. State S_n is the one responsible for adding an escape character, and it has priority even in the case of a key exchange request (note that whatever the input from S_{n-1} a transition to S_n is always made).

Note that FSM_{Tx} adds an extra char when *n-1* chars in the data stream coincide with those defined for the header. Hence, the probability of adding a new char can be defined as:

$$P_{char\_added} = \frac{1}{2^{b\cdot(n-1)}} \qquad (7)$$

Similarly, Fig. 5b presents a graph representing the parameterized template of FSM_{Rx}. States S_{Rx} define the outputs for *Se*, *Ld*, *LdF*, *FLd*, and *Cl* signals (O_{Rx}) (outputs are specified in the inner table in Fig. 5b), while transitions among states are performed based on the decrypted char *m* and the *Xa* inputs (I_{Rx}). S_o is the initial state and every transition to that state loads the received char and produces a flush on the output register. State S_{n-1} is responsible for distinguishing between an escape char inserted by the Tx and the complete KxF header pattern. In the first case, the transition is to S_n, which discards that char and flushes the register. In the second case, the transition is to S_n', which sends the payload of the KxF to the Key Management module and clears the output register. During the reception of the KxF payload, the Key Management module keeps the *Xa* signal active until all the chars have been received. In the meanwhile, transitions are to the state S_n'. When the complete KxF payload has been received, the FSM transitions to either state S_o or S_1, and incoming chars are stored again in the output register.

Both graphs can be particularized for any given length of the KxF header greater or equal to 3 by just adding as many S_k intermediate states as needed.

C. FSM Particularization

We now illustrate the easiness to particularize the FSMs given the n-char KxF header pattern $H = <h_0, \ldots, h_{n-1}>$ and the bits per symbol of the modulation format (*b*). The specific FSMs for Tx / Rx are defined by: *i*) the state-transition matrix (STM) of dimensions $|S_{(\cdot)}|\times 2^{|I(\cdot)|}$; and *ii*) the output matrix

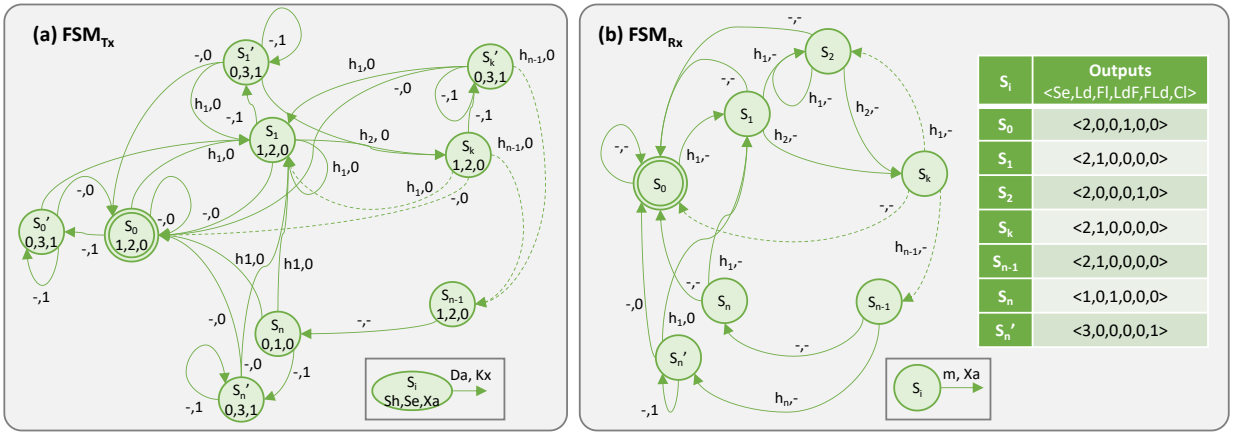

Fig. 5. Graphs representing parameterized templates for FSM$_{Tx}$ (a) and FSM$_{Rx}$ (b).

Algorithm 1. FSM$_{Tx}$ Generation

Input: *H, b*
Output: *STM, OM*

1: $Ch \leftarrow 2^b$; $n \leftarrow |H|$; $|S| \leftarrow 2n+1$
2: $S \leftarrow [i \text{ for } i: 0..|S|-1)]$
3: $STM \leftarrow [|S|]\ [2Ch]$
4: **for** i: 0..n-1 **do**
5: **for** c: 0..Ch-1 **do**
6: **if** $i = n$-1 **then**
7: $STM[i][c] \leftarrow S[n]$
8: $STM[i][Ch+c] \leftarrow S[n]$
9: **continue**
10: **if** $i = n$ **OR** $i = 2n$ **then** $STM[i][Ch+c] \leftarrow S[2n]$
11: **else if** $i <$ n **then** $STM[i][Ch+c] \leftarrow S[n+1+i]$
12: **else** $STM[i][Ch+c] \leftarrow S[n+i\%n]$
13: **if** $c = H[0]$ **then** $STM[i][c] \leftarrow S[1]$
14: **else if** $c = H[i\%n]$ **then** $STM[i][c] \leftarrow S[i\%n+1]$
15: $OM \leftarrow [|S|]$
16: **for** i: 0..$|S|$-1 **do**
17: **if** $i < n$-1 **then** $OM[i] \leftarrow [1, 2, 0]$
18: **else if** $i = n$ **then** $OM[n] \leftarrow [0, 1, 0]$
19: **else** $OM[i] \leftarrow [0, 3, 1]$
20: **return** *STM, OM*

(OM) of dimensions $|S_{(\cdot)}|\times|O_{(\cdot)}|$. Specifically, $|S_{Tx}| = 2\cdot|H|+1$, $|S_{Rx}| = |H|+2$, $|I_{Tx}| = |I_{Rx}| = b+1$, $|O_{Tx}| = 3$, and $|O_{Rx}| = 7$.

Algorithm 1 presents the pseudocode to generate FSM$_{Tx}$. The algorithm first computes the number of different chars as a function b (for input *Da*) and the number of states, initializes vector S with the states, and the state-transition matrix *STM* (lines 1-3); note that the *STM* is initialized with all transitions to state S_0. Next, the transitions for states are computed as follows (lines 4-14): 1) transitions from state S_{n-1} are to S_n disregarding the value of *Kx* (lines 6-9); 2) whenever input *Kx* is active, transitions from state S_i are to S_i' (lines 10-12); 3) whenever input *Kx* is not active, transition is to S_1 if $Da = h_0$ or to state S_{i+1} when $Da = h_i$ (lines 13-14). The output matrix *OM* is filled (lines 15-19) and the generated FSM$_{Tx}$ is eventually returned (line 20).

5. ILLUSTRATIVE RESULTS

We have implemented LPsec as a MATLAB-based simulation. In particular, we have integrated the encryption and decryption blocks in the Tx and Rx, as depicted in Fig. 1, where a single polarization 64 GBd optical signal was considered. Three different modulation formats are assumed: 16-QAM, 32-QAM and 64-QAM. In this section, we present the obtained results to validate LPsec.

A. Optical System Performance Analysis

Let us first analyze the performance from the optical perspective. In the simulator, the signal was sampled and passed through a root-raised-cosine pulse shaper with roll off factor of 0.06. The signal was launched into a fiber channel with N spans, each being 80 km long. After every span, an optical amplifier with a noise figure of 4.5 dB compensates for fiber losses. Additive white Gaussian noise is added after each span to model ASE noise. For the simulation of the fiber channel, standard single mode fiber with the following parameters was considered: fiber loss α = 0.21 dB/km, dispersion D = 16.8 ps/(km-nm) and nonlinear coefficient γ = 1.14 W^{-1}km^{-1}. A 2^{16} pseudo-random sequence was used to generate the payload. The signal was propagated then using the symmetric split-step Fourier method, solving the non-linear Schrödinger equation [25]. The signal was coherently received; it was down-sampled to 2 samples per symbol, and an ideal chromatic dispersion filter was used.

We first analyzed the performance of the system with and without encryption to verify that encryption does not degrade the performance of the system. Fig. 6a shows the obtained results for 25 spans, where we observe that the BER remains the same with and without encryption for the selected modulation formats for different input powers.

Once the optical performance was verified, we implemented convolutional forward error correction (FEC) encoding and decoding; FEC code rate of 2/3 was used for data encoding, whereas the Viterbi decoding algorithm was used at the receiver [26]. Fig. 6b presents the BER as a function of the number of spans and the inner table summarizes the maximum number of spans where the FEC corrected any transmission error.

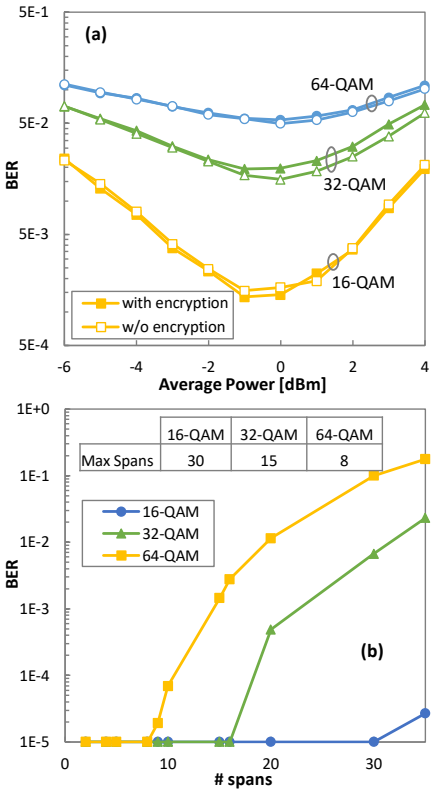

Fig. 6: BER w/ and wo/ encryption (a) and BER vs number of spans with FEC (b)

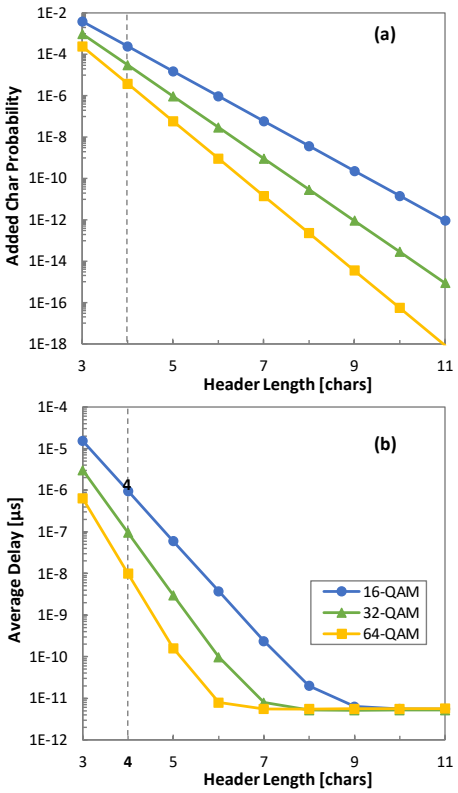

Fig. 7: Added Char Probability (a) and average delay (b)

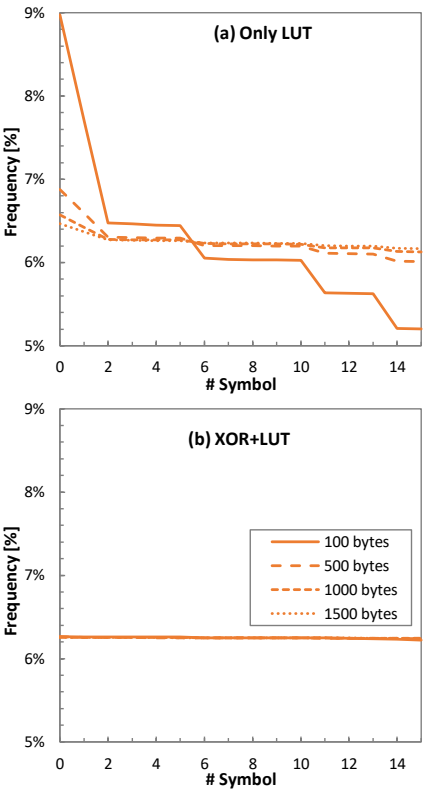

Fig. 8: Frequency of symbols w/o (a) and w/ encryption (b)

B. Delay Introduced by the KxF and Escape Characters

Next, we evaluate the delay introduced by the proposed KxF that allow exchanging the new LUT and Tx public key (but requires to stop the normal data transmission), as well as the additional escape chars added to avoid collisions of the transmitted data with the KxF; we assume 256-bit keys and 8-bit CRC. Recall that the operations related to the pure data encryption involve XOR operations (performed in blocks of b bits according to the used modulation format) and LUT access; we assume that those do not introduce significant delay.

Let us first analyze the probability of adding a new char as a function of the length of the header n for the considered modulation formats. Fig. 7a shows the results from plotting eq. (7), where the probability is below 0.4% and 0.024% even for lengths as short as 3 chars for 16-QAM and 64-QAM, respectively. Such probability is related to the average delay, so a longer header would further reduce the delay; however, it would also increase the size of the FSMs to be implemented in the Tx and the Rx; thus, a reasonable trade-off needs to be found.

Fig. 7b shows the average delay introduced vs. header's length (n), assuming time periods of 1 s., i.e., key exchange is performed every 1 s. First, we observe that the introduced delay is negligible, just a few ps even for n=3 and 16-QAM. Interestingly, when the header length is small, the probability of adding escape chars is higher and the average delay mainly depends on the number of chars added. However, as the header length increases, the average delay decreases to a point when the delay is mainly influenced by the *KxF* itself. In view of these results, n=4 char length is selected as it can provide a good balance between delay and simplicity.

C. Frequency Analysis Attack

In this section, we analyze how a frequency analysis can be used by an attacker in case a LUT substitution only is implemented (similar approach as in [7]). This will highlight why symmetric encryption is an important part of the optical encryption block. Suppose that an attacker can send a significant amount of data over the optical system, e.g., in the form of IP packets, and she/he can also eavesdrop on the transmitted signal. Then, the attacker could send packets filled with selected payloads, e.g., all 0's and observe the data sent over the fiber. Because LUT just substitutes blocks of b bits of data, the attacker can then map the pattern in the payload of the injected packets with the symbols sent over the fiber. Fig. 8 shows an example of frequency analysis, where the symbols are sorted by its frequency of appearance to facilitate its representation. For this test, we have generated Ethernet frames of fixed size starting in 100 bytes until 1500 bytes. In the frame, the only part that was not randomly generated was the MAC and IP addresses, which represent a small proportion of bits that are predictable. Even with long frames, differences in the frequency of the symbols can be observed in Fig. 8a, which enables this attack

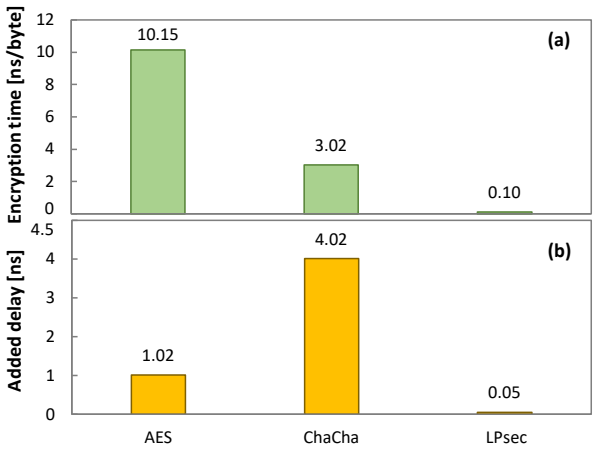

Fig. 9. Encryption time (a) and added delay (b) for a 16-QAM @32Gbaud optical system.

regardless of how many different LUT permutations exist. Fig. 8b shows that the attack by frequency analysis will not succeed when data is encrypted with the symmetric key, due to the properties of the XOR operation.

D. PRNGs Analysis for Stream Cipher

When applying stream ciphers, the main consideration is the selection of PRNG, as discussed in Section 3.C. Standard stream ciphers (e.g., ChaCha or AES in counter mode) can be used to produce high-quality PRNG. The quality of a PRNG can be examined using some of the available empirical statistical tests; see, e.g., [27], [28]. However, standard stream ciphers can hardly be used for the speeds that are targeted at the optical layer and, in consequence, other options should be analyzed. Specifically, the 64-bit all-purpose PRNGs in [29] exhibits enough speed for the specific requirements of the optical layer. Such PRNGs pass many of the statistical tests and can partially fulfil the requirements to be used in cryptographic applications. From the set of PRNGs proposed in [29], we selected Xoshiro256+ as stream cipher.

Let us first compare the speed of AES, ChaCha, and LPsec for a 32Gbaud 16-QAM system. For AES and ChaCha, the OpenSSL library was used and blocks of 128 bits for AES and 512 bits for ChaCha are considered for encryption. In contrast, Xoshiro256+ was integrated in the design of the optical encryption block in LPsec (see Fig. 4) and used as PRNG. Encryption times were computed on a stream of 1GB. The tests were performed on an Intel® Core™ i7-4790 CPU @ 3.60GHz using gcc version 9.3.0. O3 optimization was used in all the cases.

Fig. 9a presents the obtained encryption times for AES, ChaCha and LPsec, where we observe that the latter reduces the encryption time of AES by 2 orders of magnitude, whereas ChaCha reduces the encryption time by a factor of about 30. Note that encryption time must be smaller than the time required to transmit a block. With the results from Fig. 9a, AES can support up to 0.8 Gb/s, ChaCha up to 2.7 Gb/s, while LPsec can support up to 80 Gb/s, all using Intel® Core™ i7-4790 CPU. Of course, when those methods are implemented in specialized hardware, transmission times can scale several times, but these results show clearly the potential of LPsec.

Let us now analyze the average delay introduced by each encryption method. Recall that AES works on blocks of 128 bits, whereas ChaCha works on blocks of 512 bits and the encryption can only start when sufficient bits have arrived. Symmetrically, once a block has been encrypted, bits have to wait until they are actually transmitted to the destination. In contrast, LPsec works in groups matching the bits per symbol of the modulation format, so bits in the input bit stream have to wait half of the inverse of the baud rate, on average. Fig. 9b presents the average delay introduced for each of the encryption methods, where we can observe the very low delay of LPsec.

E. Security Level Against Exhaustive Search Attack

Finally, let us explore the security level of LPsec in terms of exhaustive search or brute force attacks, where all possible keys are tested until the correct one is identified. For this attack, we assume the *known-plain-text* attack model, where a plain text along with the cipher text are known to the attacker; if the plain text has some repeated properties, like headers or identifiers in the communication, that is applicable on cipher text only attacks as well.

The key length used in the encryption method will determine the strength of encryption with the longer keys being more difficult to crack. For instance, with a 256-bit key, the brute-force attack has a complexity of 2^{256}. However, techniques like precomputation attacks are short-cuts to the exhaustive search attacks and can greatly reduce such complexity [30]. Precomputation attacks exploit the birthday paradox (i.e., the probability that in a set of randomly chosen people, some pair of them will have the same birthday). The birthday attack is based on the fact that duplicate values or collisions appear much faster than expected. In general, if a system takes N different values, the first collision can occur after $\sqrt{N}$ random values [31]. In practice, a precomputed table can be generated by the attacker in offline mode. E.g., for a 256-bit key system, the attacker can precompute a table with precomputed cipher texts by using only 2^{128} random key entries. Then, she/he eavesdrops on each message and checks whether the ciphertext appears in the table. If there is a collision, then the key is used in the encryption and arbitrary information can be added by the attacker till the key is valid. In this case, the workload for the attacker becomes 2^{128}, which is much smaller than the default expected 2^{256}.

We have evaluated the key sizes in terms of precomputation attacks for LPsec. On the one hand, Xoshiro256+ generates 256-bit keys. Besides, the substitution cipher adds more complexity to the brute-force attack, e.g., key spaces of 44, 117, and 295 bits are produced

using 16, 32, and 64 -QAM, respectively, which results in total key sizes for LPsec of 300, 373, and 551. This means that the effective computation required by the birthday paradox is over 2^{150}, which can be considered safe enough for the foreseeable future [32]. In conclusion, LPsec can be considered secure against exhaustive search attacks.

6. CONCLUDING REMARKS

A complete solution to add encryption at the optical connection level has been presented in this paper. The solution includes a mechanism for key distribution from the Tx to the Rx and two ciphers that, when combined, can provide the required security level and are able to work on 100s of Gb/s data flows. The key distribution is performed using a Key exchange Frame that is sent periodically from the Tx to the Rx. Note that the proposed key exchange enables implementing security at the optical layer, although can be substituted as soon as other key exchange mechanisms, like QKD, become available. The Xoshiro256+ PRNG is used for symmetric key expansion, so as to provide keys that are used by the first XOR-based cipher. A second cipher based on LUT substitution improves the security level. The design of LPsec has been presented and is easily implementable on current coherent optical systems.

Simulation results carried out for 16, 32 and 64-QAM signals show that LPsec has negligible impact on the performance of the optical transmission system. Moreover, the required periodical key exchange does not add any significant delays. The security against two well-known attacks, frequency analysis and exhaustive search, has been analyzed and it was shown that LPsec provides a high level of security against them.

As a follow-up of this work, we are starting the implementation of some of the LPsec modules described in Section 4 in a hardware prototype aiming at experimentally demonstrating 100s Gb/s encryption speed.

Funding. The research leading to these results has received funding from the European Community's through the MSCA REAL-NET project (G.A. 813144), the H2020 B5G-OPEN (G.A. 101016663), the MICINN IBON (PID2020-114135RB-I00) project, and from the ICREA Institution.

REFERENCES

1. M. Fok, Z. Wang, Y. Deng, and P. Prucnal, "Optical Layer Security in Fiber-Optic Networks," IEEE Transactions on Information Forensics and Security, vol. 6, pp. 725-736, 2011.
2. A. Aguado, D. Lopez, A. Pastor, V. Lopez, J. Brito, M. Peev, A. Poppe, V. Martin, "Quantum cryptography networks in support of path verification in service function chains," IEEE/OSA J. of Optical Communications and Networking, vol. 12, pp. B9-B19, 2020.
3. D. Bernstein, "ChaCha, a variant of Salsa20," in Workshop Record of SASC, 2008.
4. "Specification for the Advanced Encryption Standard (AES)," FIPS-197, National Institute of Standards and Technology (NIST), 2001.
5. M. Abbade, M. Cvijetic, C. Messani, C. Alves, and S. Tenenbaum, "All-optical cryptography of M-QAM formats by using two-dimensional spectrally sliced keys," Applied Optics, vol. 54. pp. 4359-4365, 2015.
6. M. Abbade, L. Lessa, M. Santos, A. Prado and I. Aldaya, "A New DSP-Based Physical Layer Encryption Technique Applied to Passive Optical Networks," in Proc. ICTON, 2018.
7. B. Liu, L. Zhang, X. Xin and N. Liu, "Piecewise Chaotic Permutation Method for Physical Layer Security in OFDM-PON," IEEE Photonics Technology Letters, vol. 28, pp. 2359-2362, 2016.
8. M. Sciamanna and K. A. Shore, "Physics and applications of laser diode chaos," Nat. Photon., vol. 9, pp. 151-162, 2015.
9. A. Zhao, N. Jiang, S. Liu, Y. Zhang, K. Qiu, "Physical Layer Encryption for WDM Optical Communication Systems Using Private Chaotic Phase Scrambling," IEEE/OSA Journal of Lightwave Technology, vol. 39, pp. 2288-2295, 2021.
10. L. Jiang, Y. Pan, A. Yi, J. Feng, W. Pan, L. Yi, W. Hu, A. Wang, Y. Wang, Y. Qin, L. Yan, "Trading off security and practicability to explore high-speed and long-haul chaotic optical communication," OSA Optics Express, vol. 29, pp. 12750-12762, 2021.
11. T. Shake, "Security performance of optical CDMA against eavesdropping," IEEE J. Lightwave Technol., vol. 23. pp. 655-670, 2005.
12. Z. Jiang, D. Leaird, and A. Weiner, "Experimental investigation of security issues in O-CDMA," IEEE J. Lightwave Technol., vol. 24. pp. 4228-4334, 2006.
13. Z. Si, F. Yin, M. Xin, H. Chen, M. Chen, and S. Xie, "Code extraction from encoded signal in time-spreading optical code division multiple access," OSA Optics Letters, vol. 35, pp. 229-231, 2010.
14. B. Dai, Z. Gao, X. Wang, N. Kataoka and N. Wada, "Demonstration of differential detection on attacking code-shift-keying OCDMA system," Electronics Letters, vol. 46, pp. 1680-1682, 2010.
15. B. Wu, A. Agrawal, I. Glesk, E. Narimanov, S. Etemad and P. Prucnal, "Steganographic fiber-optic transmission using coherent spectral-phase-encoded optical CDMA," in Proc. CLEO, 2008.
16. K. Kravtsov, B. Wu, I. Glesk; P. Prucnal, E. Narimanov, "Stealth transmission over a WDM network with detection based on an all optical thresholder," in Proc. IEEE/LEOS, 2007.
17. Z. Wang and P. Prucnal, "Optical steganography over a public DPSK channel with asynchronous detection," IEEE Photonics Technology Letters, vol 23, pp. 48-50, 2011.
18. B. Wu, Z. Wang, Y. Tian, M. Fok, B. Shastri, D. Kanoff, P. Prucnal, "Optical steganography based on amplified spontaneous emission noise," OSA Optics Letters, vol. 35, pp. 2065-2071 2013.
19. ADVA Layer 1 security. https://www.adva.com/en/innovation/network-security/layer-1-security. [Accessed: Sept. 2021].
20. D. Neuenschwander, "Diffie-Hellman Key Exchange," Probabilistic and Statistical Methods in Cryptology, 2004.
21. D. Boneh and V. Shoup, A Graduate Course in Applied Cryptography, http://toc.cryptobook.us [Accessed: Sept. 2021], 2020.
22. C. Shannon, "Communication theory of secrecy systems," Bell Labs Tech. J., vol. 28, pp. 656-715, 1949.
23. Y. Nir and A. Langley, "ChaCha20 and Poly1305 for IETF Protocols," IRTF RFC-8439, 2018.
24. H. Corrigan-Gibbs and D. Kogan "The Discrete-Logarithm Problem with Preprocessing," in Proc. EUROCRYPT 2018.
25. G. Agrawal, Nonlinear fiber optics, Academic Press, 5th edition, 2013.
26. A. Tychopoulos, O. Koufopavlou and I. Tomkos, "FEC in optical communications - A tutorial overview on the evolution of architectures and the future prospects of outband and inband FEC

for optical communications," IEEE Circuits and Devices Magazine, vol. 22, pp. 79-86, 2006.
27. TestU01, Empirical Testing of Random Number Generators [Online]. http://simul.iro.umontreal.ca/testu01/tu01.html.
28. NIST Empirical Testing of Random Number Generators [Online]. https://www.nist.gov/publications/statistical-test-suite-random-and-pseudorandom-number-generators-cryptographic.
29. D. Blackman and S. Vigna, "Scrambled Linear Pseudorandom Number Generators," [Online] arXiv:1805.01407, 2019.
30. D. Mcgrew, "Counter Mode Security: Analysis and Recommendations," vol. 2, Cisco Systems, 2002.
31. E. Biham, "How to Forge DES-encrypted messages in 2^{28} steps," Technion, Technical Report CS0884, 1996.
32. N. Ferguson, B. Schneier, T. Kohno, *Cryptography Engineering: Design Principles and Practical Applications*, Wiley Publishing, 2010.
33. M. Dallaglio, A. Giorgetti, N. Sambo, L. Velasco, and P. Castoldi, "Routing, Spectrum, and Transponder Assignment (RSTA) in Elastic Optical Networks," IEEE/OSA Journal of Lightwave Technology (JLT), vol. 33, pp. 4648-4658, 2015.