IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received January 8, 2018, accepted March 1, 2018, date of publication March 13, 2018, date of current version April 18, 2018.

Digital Object Identifier 10.1109/ACCESS.2018.2815567

An Intra-Slice Security Solution for Emerging 5G Networks Based on Pseudo-Random Number Generators

BORJA BORDEL[1], AMALIA BEATRIZ ORÚE[2], RAMÓN ALCARRIA[1], AND DIEGO SÁNCHEZ-DE-RIVERA[1]

[1]Universidad Politécnica de Madrid, 28040 Madrid, Spain
[2]Instituto de Tecnologías Físicas y de la Información, Consejo Superior de Investigaciones Científicas, 28006 Madrid, Spain

Corresponding author: Borja Bordel (bbordel@dit.upm.es)

This work was supported in part by the Ministry of Economy and Competitiveness through SEMOLA project under Grant TEC2015-68284-R, in part by the Autonomous Region of Madrid through MOSI-AGIL-CM project (co-funded by EU Structural Funds FSE and FEDER) under Grant P2013/ICE-3019, in part by the Ministerio de Economía, Industria y Competitividad, in part by the Agencia Estatal de Investigación, and in part by the Fondo Europeo de Desarrollo Regional through the COPCIS project under Grant TIN2017-84844-C2-1-R. The work of B. Bordel was supported by the Ministry of Education through the FPU Program under Grant FPU15/03977.

ABSTRACT Future 5G networks must provide communication services to a great and heterogeneous collection of scenarios: from traditional mobile communications to emerging applications such as Industry 4.0 or the Internet of Things (IoT). In this context, the network slicing technique is defined, where network resources are packaged and assigned in an isolated manner to the sets of users according to their specific requirements. Two different domains are, thus, defined: the intra-slice domain (where dedicated and specific solutions have to be deployed) and the inter-slice domain (including transversal solutions). One of the key topics which should be redefined following this approach is security. Traditionally, some solutions (such as stream ciphers) were not considered in mobile networks. However, 5G systems will be extensively employed in other new and very distinct scenarios, where requirements are different. For example, the use of resource constrained devices with little mobility and real-time data streaming in certain IoT applications suggests the use of stream ciphers (and other similar techniques) as the main security solutions. Therefore, in this paper, we investigate and propose a new security solution for emerging 5G networks, to be applied in the intra-slice domain. The proposed solutions employ lightweight pseudo-random number generators in order to provide the keystream used in stream ciphers which protect the private information and hide the communication signals in the frequency spectrum using spread spectrum techniques. We also describe and evaluate a first implementation of the proposed solution, using both, a simulation scenario and a real deployment.

INDEX TERMS 5G mobile communication, cryptography, Internet of Things, network slicing, random number generation, security.

I. INTRODUCTION

Future 5G networks must provide communication services to a great and heterogeneous collection of scenarios. Traditional voice calls or mobile data connections are only two examples among a very wide collection of applications that includes some of the most popular technological paradigms nowadays, such as the Internet-of-Things (IoT) [1], Industry 4.0 [2] or Cyber-Physical Systems (CPS) [3].

A homogeneous portfolio of communication services [4] cannot fulfil the very different requirements needed by all these scenarios. Then, 5G networks must define several operation planes in order to address and meet the needs of all future systems and applications. Each plane should be isolated from the rest of them to, for example, avoid the spreading of malfunctions or cyber-attacks in a public free service (e.g. Internet access in airports).

This design approach is known as network slicing. Network slicing [5] is a technique where network resources are packaged and assigned in an isolated manner to groups of users according to their specific requirements. With this view, the entire network infrastructure is divided into dedicated vertical segments, focused on proving a certain set

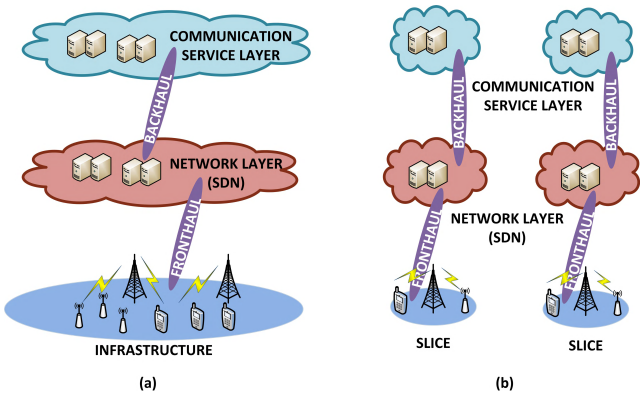

FIGURE 1. Schematic view of the overall architecture for (a) traditional mobile networks and (b) future 5G system with network slicing.

of communication services with common requirements, and isolated from the rest of existing slices. Fig. 1 represents in a schematic way the new network architecture in comparison to traditional mobile deployments.

In this new generation of mobile networks, thus, two different domains are defined [6]. The intra-slice domain contains all functions, components and solutions specifically designed to address and solve problems related to the application of mobile communications in specific scenarios (IoT systems, eHealth applications, etc.). On the other hand, inter-slice domain includes all transversal mechanisms that are common to all application scenarios and components focused on slice coordination, service orchestration, management, etc.

This new approach makes it mandatory to redefine most of the existing proposals for mobile networks, in order to ensure they fulfill the requirements of the domain where they are going to be employed. In that way, many intra-slice ad hoc solutions are nowadays needed in order to describe the future functional components that will make up the emerging 5G networks.

Obviously, this situation also affects one of the key topics in communication engineering nowadays: security.

Traditionally, mobile networks have implemented security solutions adapted to scenarios composed of few powerful devices, which, besides, may present a high level of mobility. Encryption solutions in 4G networks (for example) include complex algorithms to correct lags and signal drifts in the key sharing process [7]. These techniques are required in traditional scenarios, but they make security solutions very heavy and resource consuming.

On the contrary, new scenarios for 5G networks, such as IoT applications, present important differences with respect to traditional scenarios. First, the level of mobility is much lower, as usually the associated objects (furniture, appliances, city infrastructures, etc.) are static by default. Later, the use of communication services in these scenarios is not sporadic (as voice calls or connection to the Internet), and usually there is a continuous data flow. Moreover, IoT devices tend to be small and resource constrained, so complex algorithms are not a valid solution. Finally, new cyber-attacks must be considered, as IoT systems (as unattended pervasive wireless systems) are sensitive to interference, spectrum scanning attacks, etc., which are not frequent problems in traditional mobile networks.

Nowadays, most of these applications are supported by other communication technologies such as Bluetooth or ZigBee. Systems using these technologies, however, cannot integrate a great amount of devices or support pervasive deployments. Because of this fact, cryptographic and security solutions are usually based on traditional block ciphers whose computational cost is unaffordable for future embedded devices.

In conclusion, new security solutions for the IoT (and other similar paradigms, like CPS) intra-slice domain are needed. In particular, we argue that stream ciphers, with the appropriate configuration, may address this problem.

Therefore, the objective of this paper is to describe a new stream cipher, specifically designed to be applied in emerging 5G networks. The cipher will be deployed in order to protect communications between resource constrained devices and base stations in 5G systems. Our proposal is based on a simple lightweight Pseudo Random Number Generator (PRNG), which is the core of an stream cipher that protects the user information, the meta-information of the system (including data size, the encryption scheme, etc.) and hides the communication signals thanks to a spread spectrum technique. The entire solution may be easily implemented in any resource constrained microcontroller or System-on-Chip.

The remainder of this paper is organized as follows. Section II describes the state of the art on stream ciphers for IoT applications. Section III presents the employed PRNG, the proposed initial configuration process and the behavior of the designed cipher. Section IV describes the performed experimental validation. Finally, Section V presents the obtained results and Section VI concludes this work.

II. STATE OF THE ART

In the last ten years, many different and novel security solutions for IoT systems have been proposed (including block ciphers [8], hash functions [9] and non-traditional techniques [10]). However, in this section we focus on lightweight stream ciphers, as it is the objective of our work.

Probably, the most important attempt for developing a lightweight stream cipher for emerging technological systems is the ECRYPT II eSTREAM project [11].

In this project, a catalogue of seven totally new functional lightweight stream ciphers was created and released [12], [13]. It includes the HC-128 [14], Rabbit [15], Salsa20/12 [16], SOSEMANUK [17], Grain [18], MICKEY [19] and Trivium [20] ciphers. Some of these algorithms, such as MICKEY, have already been reported to be insecure (a differential fault attack has been reported against MICKEY 2.0 in 2013 [21]), although others are still considered safe (e.g. Trivium) and, they have even been specified as an international standard [22].

The only theoretically secure cipher is known as the One Time Pad. It is obtained by combining a truly random key sequence, which is as long as the message, and the message, by means of XOR addition [23], [24]. The problem here is that we need to secretly send that random key sequence to the intended recipient.

The practical alternative is to use a pseudorandom generator: First a pseudorandom keystream is generated from a seed, which plays the role of a real key that is much shorter than the full keystream combined to the message. This seed must be unpredictable and long enough to avoid exhaustive search attacks. In this way the final objective of any PRNG is to generate a cryptographically secure random number sequence, that is, the sequence must have a very long period in which the sequence must be indistinguishable from a perfect random sequence, and must be unpredictable.

In consequence, all stream ciphers (including previously cited ones) are focused on the generation of an adequate key stream, as XOR operation is very simple and supported by all programming languages. As truly random number streams are very difficult to generate nowadays, key streams are generated by means of Pseudo Random Number Generators (PRNGs) [25].

For example, Trivium cipher is based on a PRNG with three shift registers of different lengths. These registers are placed as in a circle, and while samples are moving around, some positions are modified using combinations of other samples in the registers [26]. These PRNGs, based on Linear Feedback Shift Registers (LFSRs), are very common [27], and were the initial phase of PRNGs based on Feedback with Carry Shift Registers (FCSRs) or on combinations of both types of registers [28].

These kinds of proposals, however, are not proved to be secure, as the Linear feedback shift register can be easily cryptanalyzed. Finding a cryptographically secure PRNG is an open problem in cryptography. Some standards (the RC4 algorithm, for example) tried to address this problem and failed [29], [30], and some PRNGs that are considered valid (such as the BBS generator [31]) are very slow to be operated at real-time. On the other hand, in order to improve the randomness of the generated key stream, some proposals consider a chaotic flow [32] as input, although problems in managing chaotic dynamics have reduced the utility of these techniques.

Instead of ad hoc combinations of LFSRs and/or FCSRs, some proposals employ existing pseudorandom generators as initial input. One of these reference proposals is Lagged Fibonacci Pseudorandom Generator (LFG) [33]. Its reasonably good behavior on standard statistical tests makes it perfect to be employed as input to create more complicated PRNGs [34].

Table 1 compares the characteristics of some of the most important lightweight stream ciphers for IoT scenarios nowadays. It can be seen that around 50% of ciphers are no longer secure, although some of them employ the largest secret keys. On the other hand, usually, faster ciphers are those whose encryption scheme has been broken. There is, in this tendency, an important exception: the Trifork generator.

TABLE 1. Comparison of the stream ciphers for IoT scenarios.

Cipher	Creation date	Secure (Nov. 2017)	Key length	Speed
HC-128	2004	Yes	128	↑↑
Rabit	2003	Yes	128	↑↑
Salsa	2004	No	256	↑
SOSEMANUK	2004	Yes	128	X
Grain	2004	No	80	X
MICKEY	2003	No	80	X
Trivium	2004	Yes	80	↑↑
RC4	1987	No	8 - 2048	↑
BBS	1986	Yes	It depends on the target application	↓↓
LFG	1985	No		↑↑
Trifork	2010	Yes		↑↑

Current encryption schemes, moreover, not only protect the private user information but also meta-information about the employed cipher. This is very important, as new cyber-attacks based only on meta-information have been reported. For example, the length of encrypted samples is sometimes a very valuable knowledge to build a successful attack. Therefore, our proposal considers a solution to protect, also, meta-information.

PRNGs can be also employed in spread spectrum techniques, which can hide the communication signal in the frequency spectrum, making them more robust against interferences and electronic noise, and helping the entire system to get synchronized. In this work we use an LFG-based PRNG as a core of a stream cipher for emerging 5G network and a spread spectrum technique. The proposed solution also includes this technology.

III. A STREAM CIPHER FOR 5G NETWORKS

In this section, the proposed stream cipher is described. The first subsection presents the Trifork PRNG, which is employed in our stream cipher design. Second subsection describes the proposed process for the initial configuration; and in the third subsection the main contribution and its behavior are described in detail.

A. A PRNG FOR REDUCED RESOURCE DEVICES

One of the lightest PRNG is Tausworthe generator (1), where c_i are binary parameters, b_i are binary variables and $\oplus$ is an operator representing the exclusive-or addition [35].

$$b_n = c_{q-1}b_{n-1} \oplus c_{q-2}b_{n-1} \oplus \ldots \oplus c_0 b_{n-q} \tag{1}$$

It can, besides, generate long random sequences and many-bit random numbers (as required by cryptographic applications). In fact, Tausworthe generator produces $k-$bit random numbers, independently from the underlying hardware platform [36]. If $k = 1$, the LFSRs cited above are obtained. In its most common implementation, however, only two c_i parameters are non-zero (2).

$$b_n = b_{n-r} \oplus b_{n-s} \tag{2}$$

Tausworthe generators, when implemented as in (2), are understood as a particular case of a more generic collection of PRNGs named as Lagged Fibonacci Generators (LFGs). The general form of LFG [37] is as indicated in (3), where $r > s > 0$ are the lags, $\circ$ is an operator indicating a binary operation, m is the base and $\{x_t, t = 0, \ldots, r-1\}$ is the r-dimensional initialization vector (IV) or seed.

$$LF\,[r, s, m, \circ; \{x_t, t = 0, \ldots, r-1\}] \tag{3}$$

The logical mapping presented in (4) allows calculating the new elements in the sequence $\{x_n\}$ for $n \geq r$.

$$x_n = x_{n-r} \circ x_{n-s} \tag{4}$$

The operator $\circ$ usually represents a binary addition, subtraction, a $mod - m$ multiplication or a bitwise exclusive-or (XOR). In order to take full advantage of the data representation capacity of electronic devices and computers, usually $m = 2^N$, being N the word length of the microcontroller executing the LFG.

In order to maximize the repetition period p of this PRNG, a special configuration for the IV, r and s should be selected. In fact, some authors [38] have indicated that LFG need to be initialized at random, and the randomness of the IV is a key factor conditioning the behavior of the entire LFG.

Moreover, LFGs may be expressed as trinomials over the Galois Field of two elements, $GF(2)$, see (5).

$$x^r + x^s + 1 \tag{5}$$

In general, it is proved that LFGs defined by irreducible, primitive $mod - 2$ trinomials are a good design option [39], as they generate the maximal repetition period p if at least one seed is odd [40] (6).

$$p = 2^{N-1}\left(2^r - 1\right) \tag{6}$$

Despite all these advantages, LFGs are not secure, and several faults on their security have been reported [40], [41]. Consequently, Perturbed Lagged Fibonacci Generators (PLFGs) were defined [34].

It is well known that all bits in the random number generated by LFGs have not the same behavior [42]. While the most significant bit (MSB) presents (if the entire PRNG exhibits the maximal repetition period) a period $p_{MSB} = 2^{N-1}(2^r - 1)$, the least significant bit presents a repetition period $p_{LSB} = (2^r - 1)$. Moreover, in general, the $k - th$ presents a period $p_k = 2^{k-1}(2^r - 1)$. Most randomness problems, thus, are associated to the LSB. In order to address this challenge, it was found that the perturbation of the low and high bits in x_{n-r} and x_{n-s} could improve the performance of the entire PRNG. These perturbations could be introduced by means of any logical operation, but the generation of new additional samples to perform these operations seems to be very costly. Then perturbations are introduced through some of the existing samples generated by the same PLFG.

In its general form (7) a PLFG is defined by a one-dimensional map including the same parameters as a standard LFG and a new control parameter, d, to act over the added perturbation.

$$LF\,[r, s, m, d, \circ; \{x_t, t = 0, \ldots, r-1\}] \tag{7}$$

As said, the induced modifications in the PLFG are perturbations in the least and most significant bits of the samples. Then, the logical mapping that defines the new elements in the sequence $\{x_n\}$ for $n \geq r$ is (8).

$$\begin{aligned} x_n &= \left(\left(x_{n-r} \oplus x_{n-s}^{\gg}\right) + \left(x_{n-s} \oplus x_{n-r}^{\ll}\right)\right) mod\ m \\ x_{n-s}^{\gg} &= (x_{n-s} \gg d) \\ x_{n-r}^{\ll} &= (x_{n-r} \ll d) \end{aligned} \tag{8}$$

In (8) n represents time, and d is a constant integer usually selected to fulfill that $2 \leq d \leq 0.7N$. Operators $\ll$ and $\gg$ represent the left-shift and right-shift operations. In fact, this is a lightweight way of performing multiplications or divisions, when one of the factors is a power of two (9).

$$\begin{aligned} (x_{n-s} \gg d) &= \left\lfloor \frac{x_{n-s}}{2^d} \right\rfloor \\ (x_{n-r} \ll d) &= \left(x_{n-r} \cdot 2^d\right) mod\ m \end{aligned} \tag{9}$$

As novelty, PLFGs consider three different operation types which highly improve their performance: $m - mod$ additions, XOR additions and bit-shift (left-shift and right shift). These operations, together with the introduced perturbations, improve dramatically the randomness and repetition period of the PRNG.

A PLFG is already a good PRNG to be applied in IoT scenarios and future 5G networks. However, it was reported that statistical attacks could be performed against these systems as the output random numbers are the same employed to obtain the following samples.

A three-branch PLFG was proposed: the Trifork generator [34], which (in fact) has three branches; each one composed of a Perturbed Lagged Fibonacci Generator. In order to obtain the final output of the global PRNG, the outputs of two different PLFGs are added by means of the XOR operation. In this scheme, the third PLFG composing the Trifork will remain totally hidden. This innovative proposal guarantees that the global output sequence is useless to infer neither the internal system parameters, the current or past system state nor the secret keys. This characteristic is very important as it allows us to design a secure meta-information protection algorithm (see Section III.C).

The three PLFGs making up the Trifork generator are interconnected, so they get perturbations in a cyclic manner thanks to a set of three new internal XOR operations. In these operations, the output of a PLFG is combined with the left-shifted output of the precedent PLFG. The left-shift is recommended to be about $\frac{N}{2}$ bits, where N is the bit word size. Using this configuration, the resulting period is much longer than the one obtained from conventional Lagged Fibonacci generators. It has been proved that the global output is unpredictable, as generated sequences pass successfully the most stringent randomness test suites.

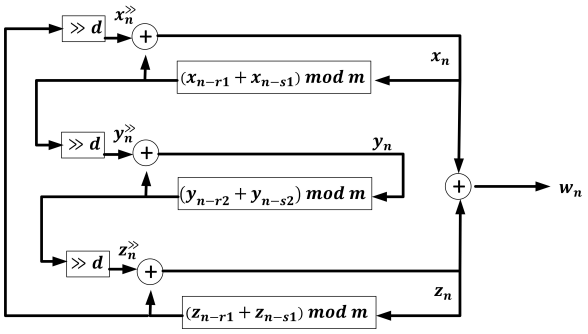

FIGURE 2. Block diagram of the Trifork generator.

The proposed architecture (see Fig. 2) hides the internal random numbers through the XOR combination of three different PLFGs. Moreover, it was found that the combination of three PLFGs allows reducing some operations, so Trifork generator requires less computational power than three separated PLFGs.

With these considerations, the mathematical expression employed to implement the Trifork generator is (10), where w_n is the output of the generator.

$$\begin{aligned} x_n &= ((x_{n-r1} + x_{n-s1})\ mod\ m) \oplus z_n^{\gg} \\ y_n &= ((y_{n-r2} + y_{n-s2})\ mod\ m) \oplus x_n^{\gg} \\ z_n &= ((z_{n-r3} + z_{n-s})\ mod\ m) \oplus y_n^{\gg} \\ x_n^{\gg} &= ((x_{n-r1} + x_{n-s1})\ mod\ m) \gg d \\ y_n^{\gg} &= ((y_{n-r2} + y_{n-s2})\ mod\ m) \gg d \\ z_n^{\gg} &= ((z_{n-r3} + z_{n-s3})\ mod\ m) \gg d \\ w_n &= x_n \oplus z_n \end{aligned} \tag{10}$$

B. INITIAL CONFIGURATION

As said, for each Trifork generator included in the stream cipher, a key must be shared between the base station and the remote IoT device. It is implicit that all stream ciphers are symmetric cryptographic schemes, so the problem of sharing keys is basic in these scenarios.

In previous works, we have already investigated some algorithms and protocols to share with safety symmetric keys in order to initiate stream ciphers based on PRNGs in 5G networks [43]. In fact, although future 5G devices to be integrated into the IoT slice (and other similar domains) will be resource constrained, they can also execute in a sporadic way some more complex processes (using, for example, auxiliary co-processors if needed).

In our application scenario, the deployed IoT system and 5G infrastructure may be represented by a set of two collections (11). B represents the set of 5G base stations and D the set of IoT devices which are connected to these base stations.

$$S_{iot} = \{B, D\} \tag{11}$$

Fig. 3 represents this scenario, where the number of base stations is fixed to one, without loss of generality.

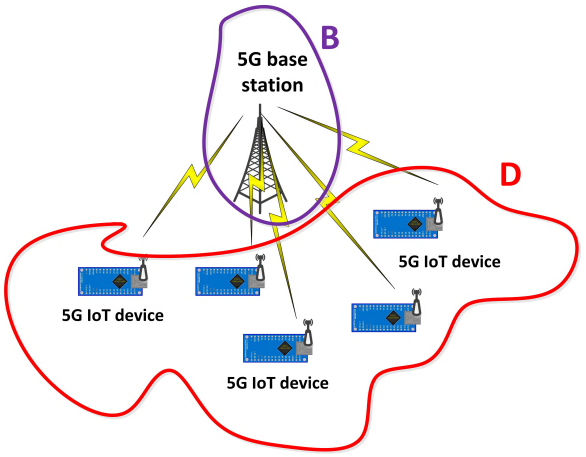

FIGURE 3. Study scenario for this work.

In this scenario, and in almost any other case of symmetric key sharing nowadays [44], a Public Key Infrastructure (PKI) is the most efficient solution to address this problem. In that way, in our system two different cryptographic functions will coexist: f_{sym} (i.e. the stream symmetric cipher based on a PRNG, see subsection C), and f_{asym} (i.e. the KPI).

As symmetric cryptographic methods need an asymmetric method to support the symmetric key sharing, in this work we are employing the RSA (Rivest, Shamir and Adleman) algorithm [45], as it can be employed to encrypt and to digitally sign documents. Besides, there are efficient implementations of this algorithm in almost any existing programming language [46].

The designed protocol to share the symmetric keys based on the RSA cryptographic algorithm is very simple. It consists of a triple handshaking procedure (the most recommended process to establish a communication link, secure or not), where mechanisms to address and solve the eventually problematic situations (packet loss, unexpected delays in the communications channel, etc.) have been considered.

Fig. 4 presents a message sequence chart describing the basic use case and behavior of the proposed protocol.

Basically, the base station, which has a powerful computational infrastructure, must calculate and broadcast its RSA public key (so, the costliest procedures are not performed by the IoT devices). This key is also signed, so the identity of the owner is certified.

Each 5G IoT device receives the public key and, when it wants to stablish a secure communication link, employs it to encrypt and send to the base station the private symmetric keys which are going to be used to initialize the stream cipher.

In order to guarantee the key sharing process has been performed successfully, the base station and the 5G IoT device confirm each other the transaction using the initialized stream cipher. If shared symmetric keys contain any error, the acknowledgment (ACK) message will not be recovered (either in the base station or in the IoT device), and the key sharing process could be repeated.

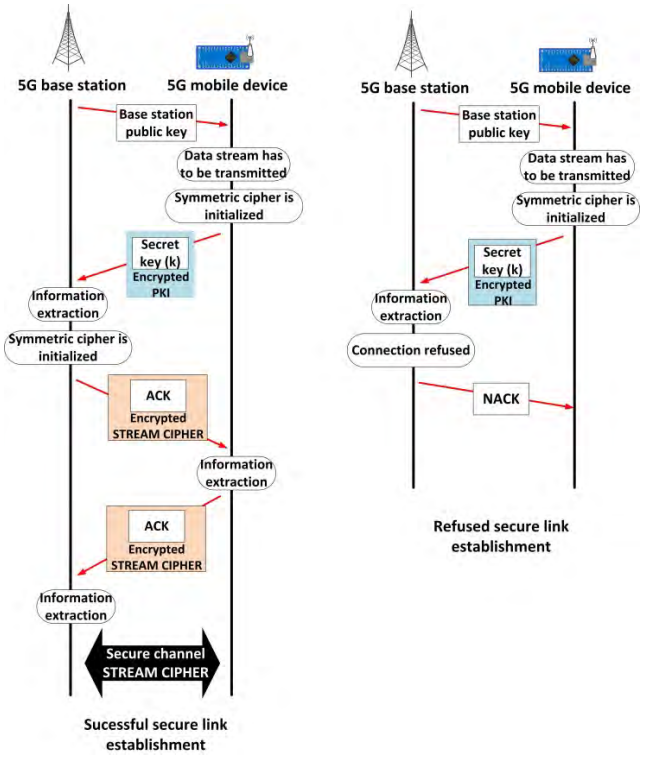

FIGURE 4. Message sequence chart of the proposed initialization protocol: successful secure link establishment and refused secure link establishment.

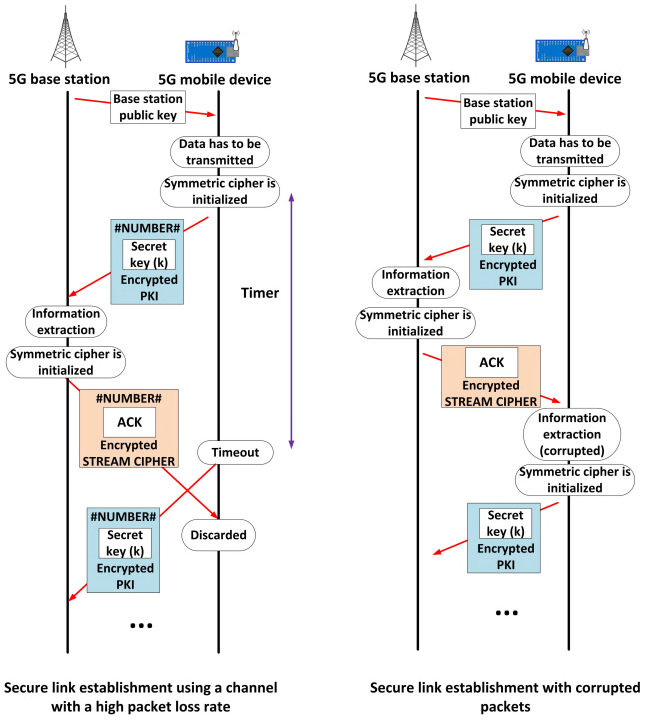

FIGURE 5. Message sequence chart of the proposed initialization protocol: secure link establishment in radio channels with delays and packet loss rate; and secure link establishment with corrupted packets.

This triple handshaking process allows checking that a bidirectional secure link is open and, besides (see Section III.C), enables the symmetric ciphers (both in the base station and the IoT device) to get synchronized.

On the other hand, if the base station is congested, the IoT device has not contracted any communication service, etc., the 5G network may refuse the link establishment and a non-encrypted NACK message is sent.

The proposed protocol also includes timers and forwarding techniques in order to get a secure link in aggressive environments; i.e. in presence of high delays, bursts of errors, interference, etc. A detailed analysis of all possible cases is not the objective of this section, as previous works have addressed this problem [43]. Only, as an example, Fig 5 presents two different situations (those which have been proved to be the most probable [43]).

In the first one (Fig. 5, on the left), it is represented the proposed solution for situations where packet sending suffers a great delay. As can be seen, basically, a timer is triggered every time a packet is sent during the initiation process. If no answer is received before this timer expires, then, the packet is sent again. Eventually, a response to the first request could be received. Thanks to a transaction number included in every packet, this response may be discarded, waiting for the answer to the last request.

In the second one (Fig. 5, on the right) it is represented the protocol reaction if an error occurs during the key sharing process and any of the sent messages (the keys or the acknowledgments) cannot be recovered adequately. In particular, once the IoT device detects an error has occurred, it reinitiates the process.

In order to avoid the system to enter in an infinite loop, if any problem (delays, communication errors, etc.) persists beyond a certain number of attempts, the procedures is cancelled, and user applications are informed that the communication link could not be established.

Considering the private keys have been correctly shared, both remote stream ciphers may be initialized, and the encryption process can start.

C. SYSTEM OPERATION

The proposed stream cipher consists of a transmitter and a receptor, which are slightly different. The stream cipher receives a constant sequence of samples which are obtained each T_s seconds (the sampling period) from an analog information source (with a maximum bandwidth f_B). Each sample has a length of N bits, according to the word length in the underlying hardware architecture. The stream cipher encrypts the information, randomizes the length of the original samples (so no meta-information about the core of the cipher is transmitted) and generates a pseudorandom code (synchronized with the encryption code) employed in a spread spectrum system which hides the communication signals in the frequency spectrum below the noise level.

Fig. 6 presents a block diagram of the transmitter. The scheme has three main parts: the encryption core, the meta-information protection module, and the synchronized

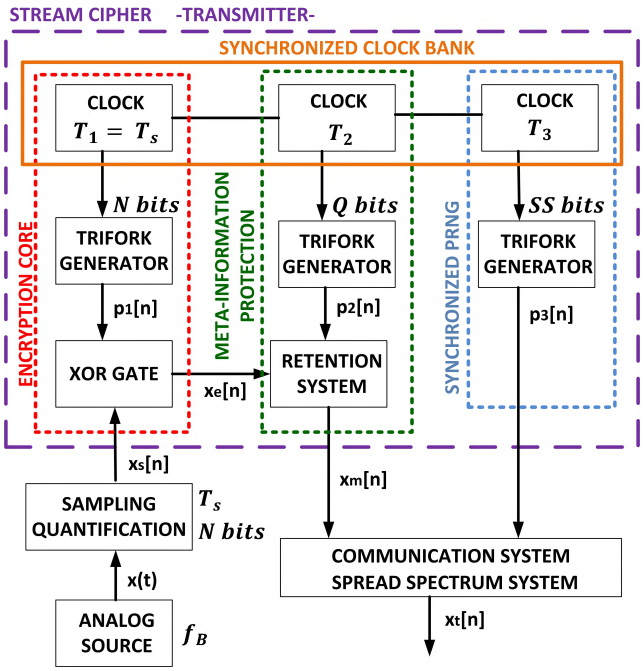

FIGURE 6. Block diagram of the transmitter of the proposed stream cipher.

PRNG to feed the spread spectrum system (not included in the cipher's design as it belongs to the radio communication system).

In the next paragraphs we are explaining each one of these three main parts in detail.

Before describing the encryption core, it is important to note that three different Trifork generators are considered in the proposed cipher. In consequence, the global private key of our system has a length around $9N$ bits, where N the length of the original samples being encrypted (the precise value depends on the specific configuration of the cipher). Nowadays, any hardware platform may support samples with, at least, 10 bits (including the simplest microcontrollers, such as the well-known Arduino project [47]). Thus, the proposed scheme employs key lengths similar to most standard stream ciphers (see Table 1). In this way, the proposed scheme, at least, has the same security level of existing solutions (or even higher).

As can be seen, the encryption core of the proposed cipher is made of a Trifork generator and a (software) XOR gate. The XOR gate receives two data sequences to be operated. Signal $x_s[n]$ is composed of N bit samples, with a sample period of T_s seconds. In order to guarantee the digital signal $x_s[n]$ correctly represents the original analog signal $x(t)$, it must be guaranteed that the Nyquist theorem is fulfilled (12).

$$T_s \leq \frac{1}{2 \cdot f_B} \tag{12}$$

In that way, $x_s[n]$ takes values in the range $\{0, \ldots, 2^N - 1\}$, and it has a symbol rate as indicated in (14), and a binary rate as indicated in (14).

$$\lambda_s = \frac{1}{T_s} \tag{13}$$

$$\lambda_b = \frac{N}{T_s} \tag{14}$$

The second injected data flow in the XOR gate $p_1[n]$ is a pseudorandom number sequence generated by a Trifork PRNG. This generator is configured to produce a signal with the same characteristics (sample length, and symbol and binary rate) than the information signal. That requirement is necessary to apply the proposed encryption mechanism.

As said, as a main encryption mechanism, a XOR gate is included in our proposal (15).

$$x_e[n] = x_s[n] \oplus p_1[n] \tag{15}$$

XOR encryption may be highly strong or very weak, depending on the use of this technology. If signal $p_1[n]$ is a sequence of random numbers, then, all possible values have the same probability, and this characteristic is transferred to the encrypted signal (16).

$$P(p_1[n] = \xi_i) = P(x_e[n] = \xi_i) = \varphi_1 = \frac{1}{2^N} \forall \xi_i \tag{16}$$

On the other hand, because of the structure of XOR operation, given a sample of the encrypted signal $x_e[n]$, all possible values in the range of the original signal $x_s[n]$ have the same probability of having produced that sample (17).

$$P\left(x_s[n] = \xi_j | x_e[n] = \xi_i\right) = \varphi_2 = \frac{1}{2^N} \forall \xi_i, \xi_j \tag{17}$$

Then, considering the Shannon's information theory, the mutual information between the original and the encrypted signal $I(x_s; x_e)$ represents the residual information that remains in the encrypted signal about the original one (18). A simple calculation proves that this quantity is zero. Thus, the encrypted signal does not contain information coming from the original signal.

$$I(x_s; x_e) = \sum_{i=0}^{2^N-1} \sum_{j=0}^{2^N-1} P\left(x_s = \xi_j, x_e = \xi_i\right) \cdot log\left(\frac{P\left(x_s = \xi_j | x_e = \xi_i\right)}{P(x_e = \xi_i)}\right) = 0 \tag{18}$$

This statistical demonstration is not valid if signal $p_1[n]$ has a short period (i.e. it cannot be considered random during the entire data transmission and the PRNG is not secure) or if the same pseudorandom sequence is employed to encrypt various messages. In both cases a simple cryptanalysis may break the encryption. In order to address the first problem, the Trifork generator has been proved to be secure with a very long period and a good random behavior. In order to address the second problem, each time a transmission starts a new configuration is generated for the stream cipher (as described in the previous section).

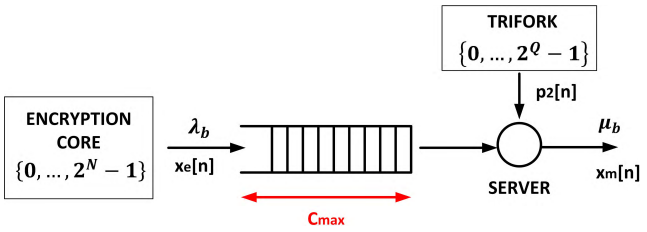

FIGURE 7. Block diagram of the retention system.

In that way, an encrypted signal $x_e[n]$ protecting the original private information is obtained. This signal, however, still contains some meta-information about the system.

In fact, recently, it has been proved that meta-information may be also successfully employed to break encryptions considered secure [48]. Trifork generator partially addresses this problem as the random signal $p_1[n]$ is not the direct output of the PLFGs, but a XOR combination of them. However, samples in $x_e[n]$ still contain an important meta-information about the system: the sample length.

Although attacks against the proposed encryption core are complicated to perform, a good quality input information for them is that the signals $p_1[n]$ and $x_s[n]$ take values in the range $\{0, \ldots, 2^N - 1\}$. This information could be perturbed if additional unused bits are included into each sample (for example), but then a percentage of the available bit rate is wasted. Instead of that, our proposal considers a meta-information protection phase, focused on hiding the information about the sample length.

The proposed meta-information protection phase consists of a Trifork generator and a retention system.

Trifork generator is configured to produce Q bit random numbers with a period of T_2 seconds. The clock that controls the operation of this Trifork generator is synchronized with the clock that controls the operation of the encryption core. This is important in order to enable the information recovering in the receptor. In conclusion, the generated random signal $p_2[n]$ has a symbol rate as indicated in (19), and a binary rate as indicated in (20).

$$\gamma_s = \frac{1}{T_2} \tag{19}$$

$$\gamma_b = \frac{Q}{T_2} \tag{20}$$

The retention system is showed on Fig. 7. It consists of a FIFO (First In First Out) binary queue, where output samples from the encryption core are stored as sequences of bit. This queue is served by a unique server which each T_2 seconds sends a new sample made of the first M bits in the queue. If there are not enough bits in the queue to create a M bit sample, the server waits until they are received. In this context, M is a random number generated by the Trifork PRNG which takes values in the range $\{0, \ldots, 2^Q - 1\}$.

With the proposed scheme, the sample length of the original system is hidden. Besides, the word length of the Trifork generator in the meta-information protection phase is also hidden. Although by collecting enough samples and analyzing their length it is probable to find the value of Q, there is no guarantee to obtain the real value.

On the other hand, with the proposed scheme, the random signal $p_2[n]$ may be easily recovered by a cyber attacker. However, that circumstance does not reduce the security level of the proposed system as, thanks to the XOR operation, it is impossible to find out the internal state of the Trifork generator using only the output sequence. Besides, although the original sample length was deducted, that does not break the encryption. Therefore, the proposed scheme is a valid meta-information protection solution.

The length of the output samples from the meta-information protection module follows a uniform distribution in the range $\{0, \ldots, 2^Q - 1\}$, as the Trifork generator that controls the server.

As the described retention system includes a queue, it is necessary to study this scheme using the queue theory, in order to configure all design parameters properly.

The first consideration we must do is to guarantee that the retention system is not congested. In this case, as only one server is considered, the congestion level ρ is equal to the traffic volume A supported by the retention system. In order to guarantee the system is not congested (so a percentage of the samples will be lost), the traffic volume cannot be greater than 1 *Erlang*. The Erlang's theory defines the traffic volume as the quotient of average input traffic rate divided by the average output traffic rate. As the proposed queue is a binary queue, this parameter is properly estimated using binary rates (21).

$$\rho = A = \frac{\lambda_b}{\mu_b} \leq 1 \tag{21}$$

The input binary rate in the retention system is constant (15), and equal to the binary rate of signal $x_e[n]$, λ_b. The output binary rate μ_b, however, depends on the output sample length, which follows a uniform distribution.

The average value of the output sample length may be easily obtained (22). And, as a sample is extracted from the queue each T_2 seconds, the average output binary rate is directly obtained (23)

$$\eta_t = \sum_{i=0}^{2^Q-1} \frac{i}{2^Q} = \frac{2^Q\left(0 + 2^Q - 1\right)}{2^Q \cdot 2} = \frac{2^Q - 1}{2} \tag{22}$$

$$\mu_b = \frac{2^Q - 1}{2 \cdot T_2} \tag{23}$$

Using this information, we obtain a relation between parameters in the encryption core and parameters in the meta-information protection step that is necessary to fulfill (24).

$$A = \frac{\frac{N}{T_s}}{\frac{2^Q-1}{2 \cdot T_2}} = \frac{2 \cdot N}{2^Q - 1} \cdot \frac{T_2}{T_s} \leq 1 \tag{24}$$

If it is guaranteed that $A \leq 1$, there are not structural sample losses in the retention system, but there is a certain

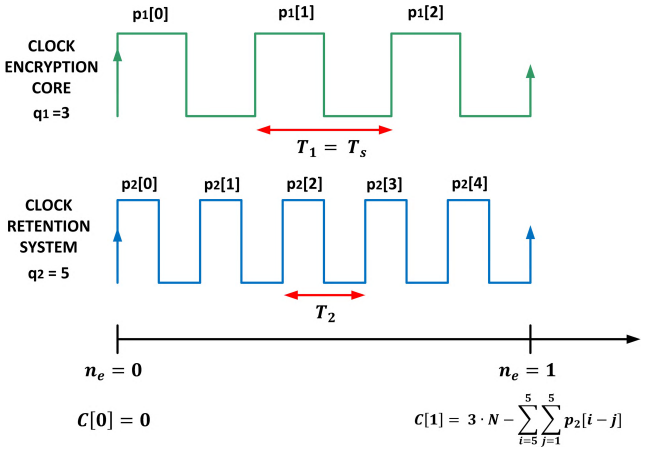

FIGURE 8. Temporal diagram of signals p_1 [n] and p_2 [n].

loss probability P_L because of the limited capacity of the queue.

In general, to operate correctly, the queue must be able to store at least one input sample (i.e. N bits). Nevertheless, if $Q > N$, then, the queue must be able to store (at least) as much samples as needed to accumulate $2^Q - 1$ bits (the maximum sample length in the retention system). Thus, there is a minimum capacity C_{min} the queue must have (25).

$$C_{min} = N \cdot \left\lceil \frac{2^Q - 1}{N} \right\rceil \tag{25}$$

The queue capacity C may be higher than C_{min} depending on the loss probability P_L the system can tolerate. Traditional mathematical analyses in queue theory are based on Markovian traffic, so they are not applicable in our case [49]. On the other hand, proposals for general traffic models (such as the Kingman's formula [50]) are designed for pure waiting systems (where queues are infinite), and they are not valid in this context either. Thus, in order to evaluate the loss probability in our scenario, we are analyzing the evolution of the queue capacity in time.

As we have said, parameters T_1 and T_2 may be freely chosen, considering the previously mentioned limits (13) (25). However, in order to guarantee a good synchronization among the clocks in the clock bank, enabling an analysis of the queue capacity; we are adding a new requirement: T_1 and T_2 must have a rational factor of proportionality (26).

$$\frac{T_2}{T_s} = \frac{q_1}{q_2} q_1, q_2 \in \mathbb{N} \tag{26}$$

This requirement means that, every $q_2 T_2 = q_1 T_1$ seconds, signals $p_1[n]$ and $p_2[n]$ present a common edge (a rising edge or a falling edge); a characteristic that allows them to be synchronized. Temporal instants when both signals present a common edge are noted as n_e. See Fig. 8

In (27) it is described the queue capacity in these temporal instants. Besides, if we take as initial condition that $C\,[0] = 0$ (the queue is empty when the system starts operating), the queue capacity may be calculated as in (28).

$$C\,[n_e] = C\,[n_e - 1] + q_1 \cdot N - \sum_{j=1}^{q_2} p_2[q_2 \cdot n_e - j] \tag{27}$$

$$C\,[n_e] = n_e \cdot q_1 \cdot N - \sum_{i=q_2}^{n_e \cdot q_2} \sum_{j=1}^{q_2} p_2[i-j] \tag{28}$$

Considering the maximum queue capacity is C_{max}, samples are lost at n_e if this maximum capacity is overcome (29). Operating, and taking probabilistic values for random terms, an expression for the loss probability at $n = n_e$ is obtained (30).

$$C\,[n_e] = n_e \cdot q_1 \cdot N - \sum_{i=q_2}^{n_e \cdot q_2} \sum_{j=1}^{q_2} p_2\,[i-j] > C_{max} \tag{29}$$

$$P_L\,[n_e] = P\left(\sum_{i=q_2}^{n_e \cdot q_2} \sum_{j=1}^{q_2} p_2\,[i-j] < n_e \cdot q_1 \cdot N - C_{max} \right) \tag{30}$$

In order to obtain the global loss probability, it must be considered the aggregated result of the entire temporal series (31).

$$P_L = \sum_{n_e=1}^{\infty} P_L\,[n_e] \tag{31}$$

The proposed expression for the loss probability depends explicitly on four parameters: N, C_{max}, q_1 and q_2 . Besides, it depends implicitly on the parameter Q. However, before considering (31) as a valid expression to obtain the loss probability, it must be guaranteed that the series presents a convergent sum. In order to guarantee that, the distribution function of the summation in (31) is analyzed.

Each individual realization $p_2[n]$ in that summation is described by a uniform distribution $U[u]$ in the range $\{0, \ldots, 2^Q - 1\}$. This distribution is characteristic of the Trifork generator and of all samples it generates. As each realization (random number) is independent from the previous and the later ones, the probability density function of the summation of k realizations may be calculated as the convolution of the probability density function of the individual realizations (32). In this case, the convolution of $U[u]$ with itself k times (33) is represented by the operator $conv_k(\cdot)$.

$$pdf\,(X_1 + \ldots + X_k) = pdf\,(X_1) * \ldots * pdf\,(X_k) \tag{32}$$

$$pdf\left(\sum_{j=1}^{k} p_2\,[j] \right) = conv_k(U[u]) \tag{33}$$

The central limit theorem establishes that for large values of k (in practice for $k > 6$), the probability density function (pdf) described by $conv_k\,(U\,[u])$ is a Gaussian distribution. The mean and the variance of this distribution may be easily calculated from the mean and variance of the uniform

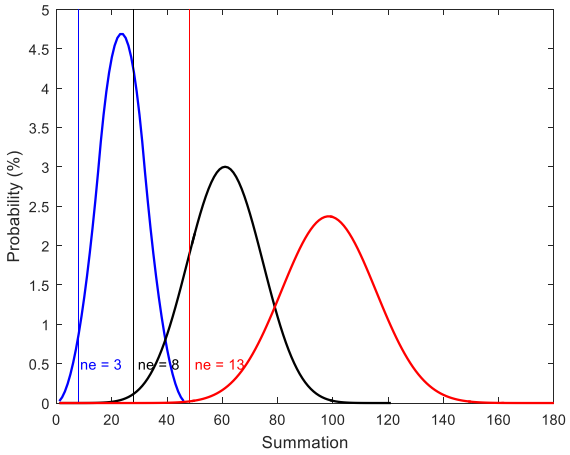

FIGURE 9. Representation of the evolution in the pdf as the number of samples in the summation goes up. The area below the limit (vertical lines) is smaller as n_e grows up. Configuration parameters: N = Q = 4. C_{max} = 4. $q_1 = q_2 = 1$. Blue graphic ($n_e = 3$). Black graphic ($n_e = 8$). Red graphic ($n_e = 13$).

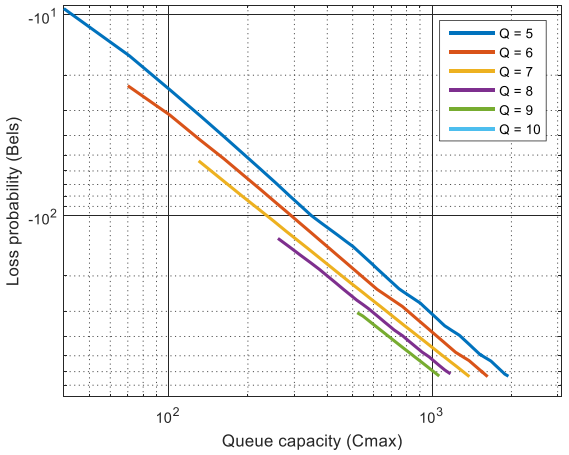

FIGURE 10. Loss probability depending on the queue capacity C_{max} for different values of Q parameter. Loss probability is represented in bels. The employed numerical algorithm has a precision of 0.1% Configuration parameters: N = 10. $q_1 = q_2 = 1$.

distribution (34).

$$conv_k\,(U\,[u]) = \Phi_{\eta_g,\sigma_g}\,[u]$$
$$\eta_g = k \cdot \eta_t$$
$$\sigma_g^2 = k \cdot \sigma_t^2 = k \cdot \frac{2^Q - 1}{12} \qquad (34)$$

Therefore, as k goes up, the mean value of the corresponding probability density function (*pdf*) moves to higher values in the abscissa axis. The increasing speed of the mean value is η_t units per convolution. The resulting Gaussian function also widens as k goes up; however, its widening speed is lower, only $\frac{\eta_t}{\sqrt{3}}$ units per convolution.

Finally, in (30), the limit value for which the probability $(n_e \cdot q_1 \cdot N - C_{max})$ is calculated also increases as n_e grows up. However, its increasing speed is much lower, $q_1 \cdot N$ units per convolution.

As a result of all this information, as n_e grows up, the area of the corresponding Gausian *pdf* that remains below the calculation limit is smaller. Then, its contribution to the sum of the series in (31) is also smaller and, thus, this sum is guaranteed to be convergent. Fig. 9 shows this situation.

Therefore, it is possible to numerically obtain graphics and tables that help people to choose the proper values for the design parameters Q, N, C_{max}, q_1 and q_2, given the maximum allowed loss probability P_L or vice versa. Fig. 10 and Fig. 11 are examples of these graphics. Fig. 10 represents the loss probability depending on the capacity of the queue C_{max} for different values of Q. Fig. 11 represents the loss probability depending on the capacity the parameter q_1 for different values of q_2. As can be seen, for current standard values and sample lengths, the loss probability is very low; as the traffic volume supported by the system is much lower than 1 *Erlang* (congestion situations only appear for small values such as $N = Q = 2$).

The output signal from the meta-information protection phase $x_m[n]$ protects both, the original private information

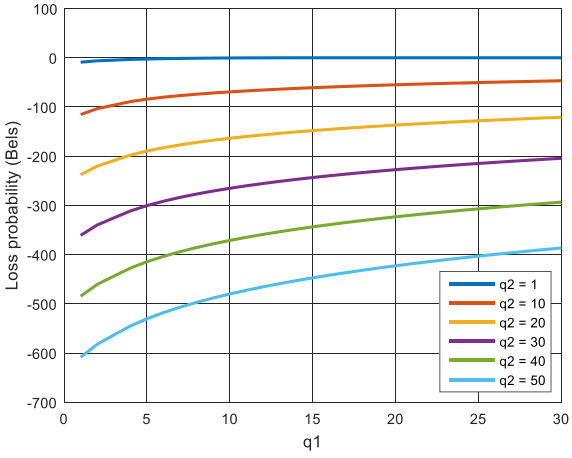

FIGURE 11. Loss probability depending on the q_1 parameter for different values of q_2 parameter. Loss probability is represented in bels. The employed numerical algorithm has a precision of 0.1% Configuration parameters: N = Q = 4. C_{max} = 4.

and the meta-information. However, it may be still detected in the frequency spectrum, so any attacker could easily capture the communication signals, although he cannot access to the protected information. Besides, attackers could try to generate interferences or deny the communication services by producing electronic noise or placing any conductive object to reflect the radio signals.

In order to address all these problems, a spread spectrum (SS) technique may be considered. The application of these techniques belongs to the communication system, and is outside the objective of this paper. Nevertheless, in the proposed system, the necessary pseudorandom number sequence to support these solutions can be generated apart from the communication module, so this third PRNG may be employed to synchronize the remote transmitter and receptor.

The proposed synchronized PNRG consists of a third Trifork generator which is controlled by a third clock,

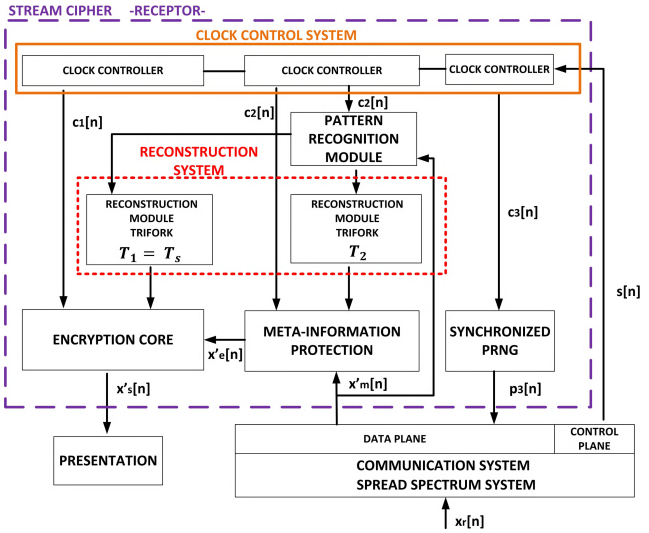

FIGURE 12. Block diagram of the receptor of the proposed stream cipher.

synchronized with the two previous ones. In particular it must be guaranteed that the period of this clock presents a rational factor of proportionality with T_1 and T_2 (35). It must be taken into account that, in order to feed a SS system, the period T_3 must be much shorter than T_1 and T_2.

$$\frac{T_3}{T_s} = \frac{q_3}{q_1}; \quad \frac{T_3}{T_2} = \frac{q_3}{q_2} \; q_1, q_2, q_3 \in \mathbb{N} \tag{35}$$

Using the secure signal $x_m[n]$ and the random signal generated by this synchronized PNRG, $p_3[n]$, the communication module sends the private information to the receptor.

In the receptor, an equivalent stream cipher to the one deployed in the transmitter is included. Fig. 12 presents the corresponding block diagram. Some details are not included, as components making up each one of the three main parts of the stream cipher (the encryption core, the meta-information protection module and the synchronized PRNG) are equal to the ones previously described for the transmitter. As can be seen, however, there are some important new modules. These new modules perform complex algorithms, so the receptor or the proposed stream cipher must be always placed in the base station. The proposed initiation protocol (see Section III.B) is designed by following this requirement.

The receptor of the proposed stream cipher includes three new functionalities: the clock control system, the pattern recognition module and the reconstruction system.

When the communication and the SS systems receive an information signal $x'_r[n]$ with a transmission power below the noise level, they employ the random signal $p'_3[n]$ to recover a signal $x'_m[n]$ with an adequate Signal to Noise Ratio (SNR). In order to do that, it is necessary to adequately synchronize the pseudorandom code in the transmitter and the receptor.

Every SS system has the ability of getting automatically synchronized through the calculation of the autocorrelation (36). Moving in time the pseudorandom sequence, the synchronization between the transmitter and the receptor will be reached when the autocorrelation is maximal.

$$\theta(n_0) = \sum_{i=n_i}^{n_i+k} p'_3[i]\, p_3[i] = \sum_{i=n_i}^{n_i+k} p_3[i+n_0]\, p_3[i]$$
$$= \begin{cases} k+1 & if \; n_0 = 0 \\ \downarrow\downarrow\downarrow & if \; n_0 \neq 0 \end{cases} \tag{36}$$

These movements in time of the pseudorandom sequence $p'_3[n]$ are obtained through a synchronization signal $s[n]$, being able to stop and run the master clock by means of a clock controller. This clock controller is connected to two additional controllers, which stop and run the other two master clocks in the cipher, so the entire clock bank is synchronized and controlled by a clock control system.

Nevertheless, getting the master clocks synchronized does not guarantee the PRNGs in the receptor are totally synchronized too. In fact, SS systems have the possibility of storing or removing some samples of the pseudorandom sequences. In practice, this means that when clocks are synchronized, PRNGs (i.e. Trifork generators in the encryption core and the meta-information protection module) are ''almost-synchronized'', there are no guarantees that they are completely synchronized.

In order to guarantee a total synchronization of the PRNG, a pattern recognition module is included in the cipher. This module receives the signal $x'_m[n]$ whose sample length represents the sequence of generated pseudorandom numbers by the Trifork generator in the meta-information protection phase in the transmitter. This sequence is extracted and analyzed in order to infer the particular time instant $n = n_0$ for which the PRNG must be configured so it generates the same numerical sequence as described by the sample length in $x'_m[n]$.

As the receptor has the symmetric keys, and the clock control system has placed the PRNG in an ''almost-synchronized'' state, it is not complicated to deduct this specific instant using a simple program and a Trifork generator configured as the PNRG in the meta-information protection module.

The information about this instant, $n = n_0$, is transmitted to a reconstruction module, consisting of a fast evolution Trifork generator which is able to recover the internal state of the PNRG for the indicated time instant. This module, finally, places the PRNG of the meta-information protection system in a total synchronization state with the corresponding PRNG in the transmitter.

It is important to note that, in the receptor, the meta-information protection module is not necessary, as the original length may be directly reconstructed knowing that samples have a length of N bits. However, this module in receptor allows the cipher to detect problems, cyber-attacks, false transmissions and fails in the synchronization. As in the transmitter, if no enough bits are received to create a sample of Nbits, then, the encryption core (in this case) waits until more bits are received. Thus, a flow of N-bit samples $x'_e[n]$

is obtained as output of this module. This signal contains the encrypted private information to be recovered.

At this point, it must be considered that clocks in the clock bank have a rational factor of proportionality. Thus, every so often, both clocks have a common edge, and both associated PRNG produce a new sample at the same time. In particular (see Fig. 8) the first common edge corresponds to the transmission beginning. Therefore, once deduced the time instant $n = n_0$, the Trifork generator in the encryption core may be also synchronized with the corresponding one in the transmitter by means of a second reconstruction module. As a result, a new pseudorandom number sequence $p'_1[n]$ is generated.

Using the generated sequence of random numbers $p'_1[n]$ and the signal $x'_e[n]$, the private information may be deciphered and recovered using the same XOR operation employed in the transmitter (37). The obtained signal $x'_s[n]$ will be equal to the original private information $x_s[n]$ if both PRNG (in the transmitter and in the receptor) are totally synchronized.

$$\begin{aligned} x'_s[n] &= x'_e[n] \oplus p'_1[n] \\ &= x_s[n] \oplus p_1[n] \oplus p_1[n] \\ &= x_s[n] \end{aligned} \tag{37}$$

The synchronization process may be complex and time consuming, so (in our proposal) it is performed by the base stations. In traditional mobile networks, where devices present a high mobility level, these techniques are useless as the synchronization state is easily lost due to mobility. However, as we said, in the studied intra-slice domain, devices have a very low and limited mobility level, so there is a very low probability of the synchronization state to get affected due to this fact.

IV. EXPERIMENTAL VALIDATION

In relation to the proposed stream cipher, evaluating the security level of the proposed encryption scheme has no sense, as it is directly equivalent to the randomness of the Trifork generator, which has been deeply evaluated in the literature [34]. Besides, important characteristics such as the meta-information protection are not correctly considered in these analyses. Therefore, in this case, we have designed an evaluation method based on key performance indicators (KPI).

Four different experiments were designed in order to evaluate (i) the synchronization time depending on the number of active communication links, (ii) the synchronization time depending on the communication links being synchronized at the same time, (iii) the binary loss probability depending on the number of active communication links and (iv) the amount of resources the proposed scheme requires from a resource constrained microcontroller. The first three experiments were based on a simulation scenario. The final experiment was performed using a first practical implementation of the proposed solution.

The designed simulation scenario was based on the NS3 simulator, a network simulator very flexible and configurable. It is an open source tool where scenarios are represented as C++ programs. This tool was deployed in a Linux (Ubuntu 16.04) machine with 8GB of RAM memory and an Intel i7 processor.

The simulation scenario consisted of a unique base station and a variable population of IoT devices (see Fig. 3). Models for radio channels, interferences, network protocols, etc. were taken from the NS3 libraries for mobile networks. The simulation model for the base station was configured according to the characteristics of current real base stations [51]. In particular, if the Digital Signal Processors (which belong to the communication system) are not considered, most modern base station nowadays [51] are composed of a relatively small amount of hardware: a general purpose embedded processor, 15 Kilobits of SRAM and 32 Mbyte of DRAM.

In order to pack and assign these resources to the base station, a virtual machine was deployed. It was configured using container technologies; specifically, a Kernel-based Virtual Machine (KVM) was created. It was deployed by means of the libVirt Application Programming Interface (API), so the machine can be managed directly from the C++ program describing the simulation scenario.

With the objective of connecting the virtual machine and the simulated scenario represented inside the NS3 simulator, the base station in the simulation was provided with a TAP bridge. These bridges forward the input traffic in the simulated base station to the external virtual machine, so real algorithms, solutions and protocols may be easily tested.

As in this first three experiments, the impact of the proposed solution in the IoT devices is not evaluated (we are considering they can easily support the stream cipher), these IoT components are represented only as simulated nodes (which are also programmable in the NS3 simulator).

In the first experiment, a set of simulations was planned. With each new simulation, the number of IoT devices is incremented in one unit. These devices try to stablish a secure communication link and get synchronized with the base station sequentially. For each simulation, the synchronization time required by the last device to be synchronized is measured. New simulations were performed until the last IoT device could not get synchronized.

During the second experiment, the same scenario of first experiment was designed. However, in this case, IoT devices try to get synchronized at the same time, instead of sequentially. The total required time to synchronize all IoT devices is measured. New simulations were performed until IoT devices cannot get synchronized.

In these two first experiments, simulations were configured for different values of Q parameter. Other relevant configuration parameters were fixed to represent a real situation nowadays: $N = 10$

Finally, in the third experiment, a slightly different scenario was created. In this new simulation, IoT devices were supposed to be already synchronized. All IoT devices in the

scenario are using an active communication link to stream a binary data flow. With each new simulation, the number of IoT devices increments in one unit. The binary loss probability is evaluated in each simulation, until it is close to 100%. In order to evaluate the binary loss probability only from to the proposed stream cipher, we reduce the impact of the communication channel considering it ideal (without packet losses, interferences, the binary rate may be as high as desired, etc.). Simulations in this experiment were configured for different values of Q parameter. The other configuration parameters were fixed to represent a real situation nowadays: $N = 10$, $q_1 = q_2 = 1$

The last, and fourth, experiment was very different from the previous ones.

In order to evaluate the resource consumption of the stream cipher when implemented in resource constrained microcontrollers, we developed an Arduino program describing the cipher proposed for transmission. We evaluate the additional use in RAM memory, program space (Flash memory) and computational time due to the use of our proposal. As a practical consequence we are analyzing the limits of the proposal to be used in the application scenario.

The experiment was repeated for different values of Q and C parameters. The other configuration parameters were fixed to represent a real situation nowadays: $N = 10$, $q_1 = q_2 = 1$

As a hardware platform, we employed an Arduino Nano board. It includes an AVR microcontroller, the ATmega328 microcontroller. It also has 32 KB of Flash memory, 2 KB of SRAM memory and 1 KB of EEPROM (which is rarely employed).

V. RESULTS

In this section results of the described experiments in the previous section are presented and discussed.

In order to remove from the results of the first experiment (as much as possible) variations in the execution process of the simulations due to exogenous variables (e.g. delays operations performed by the Operating Systems), for each case twelve different simulations were performed. The average of all these measures was obtained in order to calculate the final results. Fig. 13 shows the obtained data.

As can be seen, the maximum synchronization time for the proposed stream cipher is around $T_{syn} = 30$ seconds. This maximum time may vary around 15% depending on exogenous variables, which do not directly depend on the proposed solution.

Evolution curves present two different areas. In the first zone, the synchronization time remains constant, around $T_{syn} = 0.9$ seconds, showing small fluctuations, which are not related to the proposed cipher. This value is equal for all values of Q parameters, as operation to be performed (such as memory allocation) are almost independent of these parameters. Then (in the second zone), once a certain number of active connections is overcome, the synchronization time starts growing exponentially, until the maximum synchronization is reached.

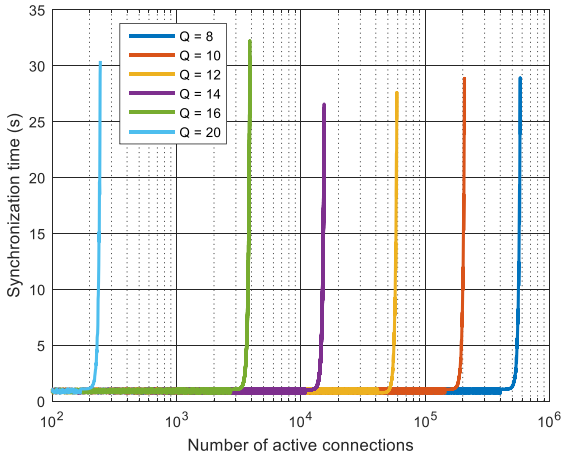

FIGURE 13. Results of the first experiment: synchronization time depending on the number of active communication links.

Both the number of active connection for which the maximum synchronization time is reached, and the growth rate in the exponential zone strongly depend on the Q parameter.

As can be seen, as Q goes up, the growth rate in the exponential zone and the number of active connection for which the maximum synchronization time is reached increase. In particular, for $Q = 20$, the maximum number of active connections is $N_{conne} = 244$, whereas for $Q = 8$, the maximum number of active connections is $N_{conne} = 5.8 \cdot 10^5$. In fact, as Q parameter grows, the length of keys, the maximum capacity of the queue, etc., must be higher, and a lower number of connections can be maintained at the same time. Any case, if we consider that nowadays a base station can only maintain around one hundred simultaneous connections (due to limits in the use of the spectrum, data rate, etc.), the proposed solution does not limit the performance of the base station in any aspect.

In scenarios composed of pervasive infrastructures (such as IoT scenarios), when the system is powered up, many devices will try to get synchronized at the same time. In order to evaluate the response of the base station in this situation the second experiment was carried out. Fig 14 shows the obtained results.

As can be seen, once more, results strongly depend on Q parameter. Curves in Fig. 14 presents, also, two areas. In the first area, the global evolution of the synchronization time is linear with the number of devices that are trying to get synchronized. The slope of this line is higher as Q goes up (it must be remarked that axes are logarithmic). This situation may be explained by the small-time differences in the execution of the synchronization operations (such as memory allocation). Although considering only one realization these differences are not appreciable (see Fig. 13), they have an important effect when analyzing the accumulation of several realizations.

Fast variations in this slope may also appear when the base station starts being congested. The maximum number of devices that may be synchronized at the same time is

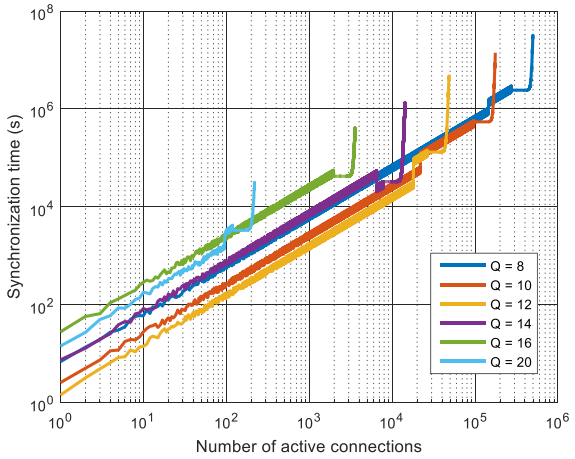

FIGURE 14. Results of the second experiment: synchronization time depending on the communication links being synchronized at the same time.

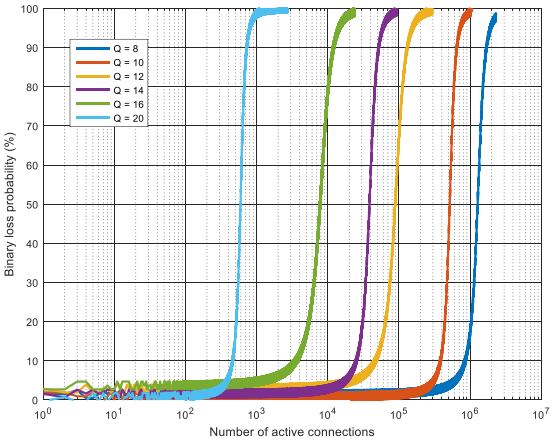

FIGURE 15. Results of the third experiment: binary loss probability depending on the number of active communication links.

similar to the maximum number of active connections which still allow new synchronizations. A difference around 10% between both values may be seen (the maximum number of devices that may be synchronized at the same time is lower, as it could be expected).

The obtained curves also present a second area. In this area the synchronization time goes up exponentially, indicating that the base station is in a congested state. The growth rate in this case also depends on the value of Q parameter. As the number of devices that can be synchronized is lower as Q parameter grows (see Fig. 13), the maximum accumulated synchronization time is also lower if Q is higher.

In conclusion, we may guarantee that the proposed cipher does not affect the synchronization of the IoT deployments in 5G mobile networks, as the imposed limits are above the real operation limits of base stations.

With previous experiments, the maximum number of devices that may be connected at the same time to a unique base station is evaluated. However, we have no information about the quality of the communication links between the base station and these devices. Experiment three aims to obtain that information. Fig. 15 shows the obtained results.

As can be seen, obtained curves are sigmoid-like functions. The binary loss probability remains between 0% and 5% if the number of active connections is below the maximum number of simultaneous connections a unique base station can handle (see Fig. 13). In particular for $Q = 20$ the binary loss probability is below 2% for $N_{conne} \lesssim 250$, whereas for $Q = 8$ this situation is maintained for $N_{conne} \lesssim 6 \cdot 10^5$.

Once the maximum number of connections is overcome, the binary loss probability starts growing, as no memory is available to handle the extra input flows. This probability has an asymptote for $p = 100\%$, since, as the quantity of data that are rejected goes up, the probability gets closer to 100%. Nevertheless, there is always a certain percentage of data that are accepted, so probability must be lower than 100% (although as close to this value as desired).

Small fluctuations may appear in the curves because of standard problems, such as the probability of queue to be overflowed.

The three described experiments are focused on base stations. However, IoT devices are (at least) as important as these components. In particular, we must guarantee that the proposed cipher is light enough to be implemented in resource constrained microcontrollers. Table 2 presents the results of the last experiment, where an implementation of the proposed cipher in a small microcontroller is evaluted.

As can be seen, the implementation of the proposed cipher is very lightweight, and it does not almost consume resources in the microcontroller. The use of the program space remains equal, and the use of RAM has grown only between 1% (for $C = C_{min}$) and 2% (for $C = 2 \cdot C_{min}$). As we are using an Arduino board, which is not prepared to work with bits at low level, the designed program must employ standard data types and the obtained program is almost independent from the value of Q parameter (in practice, for small values of Q we are allocating extra space for bits that are not going to be used).

As a consequence of a program that is independent from the value of Q, the sample processing time is also independent from this parameter. In this case, since microcontrollers only admit sequential programming (not parallel programming), the required time to update the three modules in the cipher is not negligible. In particular, the required time to process one sample has been multiplied by 10. In practice this implies that the maximum admissible bandwidth for input signals has been reduced in the same factor. Traditionally, Arduino board may consider analog signal with a maximum bandwidth up to 4.5MHz. When using our cipher, this quantity is reduced to 450KHz.

Despite this fact, standard analog signals in IoT scenarios have a much smaller bandwidth; usually no more than 100 KHz (almost three times the required throughput for voice transmissions). Therefore, the proposed cipher will not affect the wide majority of IoT applications, although they are based on resource constrained devices.

TABLE 2. Resource consumption of the proposed cipher in microcontrollers.

	Situation	Use of RAM	Use of program space	Processing time of one sample
$C = C_{min}$	Without cipher	9%	8%	112 μs
	Cipher Q = 8	10%	8%	1019 μs
	Cipher Q = 16	10%	8%	1020 μs
	Cipher Q = 32	10%	8%	1020 μs
$C = 2 \cdot C_{min}$	Without cipher	9%	8%	112 μs
	Cipher Q = 8	11%	8%	1019 μs
	Cipher Q = 16	11%	8%	1020 μs
	Cipher Q = 32	11%	8%	1020 μs

VI. CONCLUSIONS AND FUTURE WORKS

In this paper we have investigated and proposed a new security solution for emerging 5G networks, to be applied in the intra-slice domain.

The proposed solution consists of a stream cipher based on lightweight pseudo-random number generators which protect the private information, the meta-information and hide the communication signals in the frequency spectrum using spread spectrum techniques.

We have also described and evaluated a first implementation of the proposed solution, using both, a simulation scenario and a real deployment. Results showed that the proposed cipher may be implemented in both, resource constrained devices and future 5G base stations.

Although the sample processing time in resource constrained IoT devices is multiplied by ten if using the proposed cipher, it has been proved that the performance of none of 5G network components (both IoT devices and base stations) will be penalized if they include the proposed solution.

Future works will evaluate the performance of the proposed solution in microelectronic devices, such as System-on-chip, as these hardware components are envisioned to be the basis of future engineered solutions. On the other hand, an efficient resource-constraint hardware implementation of the proposed technology would be an interesting topic to be investigated nowadays.

Considering more technical aspects, problems and possible solutions when using the presented proposal in non-continuous transmissions should be evaluated (effects of bursts of information, synchronization problems, etc.).

Besides, and finally, the combination of the proposed cryptographic solution with other security methods should be investigated. Frameworks including cryptographic techniques, trust management technologies and other similar solutions should be integrated to design a totally secure future Internet-of-Things.

REFERENCES

[1] J. Gubbi, R. Buyya, S. Marusic, and M. Palaniswami, "Internet of Things (IoT): A vision, architectural elements, and future directions," *Future Generat. Comput. Syst.*, vol. 29, no. 7, pp. 1645–1660, 2013.

[2] B. B. Sánchez, R. Alcarria, D. S. de Rivera, and Á. S. Picot, "Enhancing process control in industry 4.0 scenarios using cyber-physical systems," *J. Wireless Mobile Netw.*, vol. 7, no. 4, pp. 41–64, 2016.

[3] B. Bordel, R. Alcarria, T. Robles, and D. Martín, "Cyber–physical systems: Extending pervasive sensing from control theory to the Internet of Things," *Pervasive Mobile Comput.*, vol. 40, pp. 156–184, Sep. 2017.

[4] Y.-B. Lin, Y.-W. Lin, J.-C. Wu, and Y.-C. Wang, "An investigation of telecom mobile data billing plans," *J. Internet Serv. Inf. Secur.*, vol. 6, no. 3, pp. 1–26, 2016.

[5] X. Foukas, G. Patounas, A. Elmokashfi, and M. K. Marina, "Network slicing in 5G: Survey and challenges," *IEEE Commun. Mag.*, vol. 55, no. 5, pp. 94–100, May 2017.

[6] *View on 5G Architecture*, 5G PPP Architecture Working Group, White Paper Version 2.0, Dec. 2017. [Online]. Available: https://5g-ppp.eu/wp-content/uploads/2018/01/5G-PPP-5G-Architecture-White-Paper-Jan-2018-v2.0.pdf

[7] G. Orhanou and S. El-Hajji, "The new LTE cryptographic algorithms EEA3 and EIA3," *Appl. Math*, vol. 7, no. 6, pp. 2385–2390, 2013.

[8] T. Shirai, K. Shibutani, T. Akishita, S. Moriai, and T. Iwata, "The 128-bit blockcipher CLEFIA," in *Fast Software Encryption*, vol. 4593. Berlin, Germany: Springer, 2007, pp. 181–195.

[9] J. P. Aumasson, L. Henzen, W. Meier, and M. Naya-Plasencia, "Quark: A lightweight hash," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst. (CHES)*, vol. 6225, Aug. 2010, pp. 1–15.

[10] M. S. Rahman, A. Basu, and S. Kiyomoto, "Decentralized ciphertext-policy attribute-based encryption: A post-quantum construction," *J. Internet Serv. Inf. Secur.*, vol. 7, no. 3, pp. 1–16, 2017.

[11] M. Katagi and S. Moriai, "Lightweight cryptography for the Internet of Things," Sony Corp., Tokyo, Japan, Tech. Rep. 03-2011, 2008, pp. 7–10.

[12] *eSTREAM: The ECRYPT Stream Cipher Project*. Accessed: Mar. 17, 2018. [Online]. Available: http://www.ecrypt.eu.org/stream/

[13] M. Robshaw and O. Billet, *New Stream Cipher Designs: The eSTREAM Finalists*, vol. 4986. Berlin, Germany: Springer-Verlag, 2008.

[14] H. Wu, *The Stream Cipher HC-128* (Lecture Notes in Computer Science), vol. 4986. Berlin, Germany: Springer-Verlag, 2008, pp. 39–47.

[15] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, "Rabbit: A new high-performance stream cipher," in *Fast Software Encryption*. Berlin, Germany: Springer-Verlag, 2003, pp. 307–329.

[16] D. J. Bernstein, "The Salsa20 family of stream ciphers," in *New Stream Cipher Designs* (Lecture Notes in Computer Science), vol. 4986. Berlin, Germany: Springer, 2008, pp. 84–97.

[17] C. Berbain *et al.*, "SOSEMANUK, a fast software-oriented stream cipher," in *New Stream Cipher Designs* (Lecture Notes in Computer Science), vol. 4986. 2008, pp. 98–118.

[18] M. Hell, T. Johansson, and W. Meier, "Grain: A stream cipher for constrained environments," *Int. J. Wireless Mobile Comput.*, vol. 2, no. 1, pp. 86–93, 2007.

[19] S. Babbage and M. Dodd, "The MICKEY stream ciphers," in *New Stream Cipher Designs*. Berlin, Germany: Springer, 2008, pp. 191–209.

[20] C. de Canniere and B. Preneel, "Trivium," in *New Stream Cipher Designs*. Berlin, Germany: Springer-Verlag, 2008, pp. 244–266.

[21] S. Banik and S. Maitra, "A differential fault attack on MICKEY 2.0," in *Proc. Int. Workshop Cryptograph. Hardw. Embedded Syst.*, Aug. 2013, pp. 215–232.

[22] *Information Technology—Security Techniques—Lightweight Cryptography—Part 3: Stream Ciphers*, document ISO/IEC 29192-3:2012, 2012. [Online]. Available: https://www.iso.org/standard/56426.html

[23] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "SIMON and SPECK: Block ciphers for the Internet of Things," Nat. Secur. Agency, Fort Meade, MD, USA, Tech. Rep., 2015, p. 585.

[24] E. Biham and O. Dunkelman, "Cryptanalysis of the A5/1 GSM stream cipher," in *Progress in Cryptology—INDOCRYPT*. Berlin, Germany: Springer, 2000, pp. 43–51.

[25] A. B. Orúe, L. H. Encinas, V. Fernández, and F. Montoya, "A review of cryptographically secure PRNGs in constrained devices for the IoT," in *Proc. Int. Joint Conf. (SOCO)*, Sep. 2017, pp. 672–682.

[26] Y. Tian, G. Chen, and J. Li, "On the design of trivium," School Inf. Secur. Eng., Shanghai Jiaotong Univ., Shanghai, China Tech. Rep. 431-2009, 2009, p. 431.
[27] Y. Zhang, D. Xiao, W. Wen, H. Nan, and M. Su, "Secure binary arithmetic coding based on digitalized modified logistic map and linear feedback shift register," *Commun. Nonlinear Sci. Numer. Simul.*, vol. 27, nos. 1–3, pp. 22–29, 2015.
[28] M. B. Shemaili, C. Y. Yeun, K. Mubarak, and M. J. Zemerly, "A new lightweight hybrid cryptographic algorithm for the Internet of Things," in *Proc. Int. Conf. Internet Technol. Secur. Trans.*, Dec. 2012, pp. 87–92.
[29] A. Klein, "Attacks on the RC4 stream cipher," *Des., Codes Cryptogr.*, vol. 48, no. 3, pp. 269–286, 2008.
[30] L. Dorrendorf, Z. Gutterman, and B. Pinkas, "Cryptanalysis of the random number generator of the Windows Operating System," *ACM Trans. Inf. Syst. Secur.*, vol. 13, no. 1, 2009, Art. no. 10.
[31] L. Blum, M. Blum, and M. Shub, "A simple unpredictable pseudo-random number generator," *SIAM J. Comput.*, vol. 15, no. 2, pp. 364–383, 1986.
[32] A. B. O. López, G. Marañon, A. G. Estévez, G. P. Dégano, M. R. García, and F. M. Vitini, "Trident, a new pseudo random number generator based on coupled chaotic maps," in *Proc. 3rd Int. Workshop Comput. Intell. Secur. Inf. Syst. (CISIS)*, Oct. 2010, pp. 183–190.
[33] G. Marsaglia and L.-H. Tsay, "Matrices and the structure of random number sequences," *Linear Algebra Appl.*, vol. 67, pp. 147–156, Jun. 1985.
[34] A. B. Orue, F. Montoya, and L. H. Encinas, "Trifork, a new pseudorandom number generator based on lagged fibonacci maps," *J. Comput. Sci. Eng.*, vol. 2, no. 2, pp. 1–6, Aug. 2010.
[35] W. H. Payne, "FORTRAN Tausworthe pseudorandom number generator," *Commun. ACM*, vol. 13, no. 1, 1970, Art. no. 57.
[36] S. Tezuka and P. L'Ecuyer, "Efficient and portable combined Tausworthe random number generators," *ACM Trans. Model. Comput. Simul.*, vol. 1, no. 2, pp. 99–112, 1991.
[37] Y. Bi, "A reconfigurable supercomputing library for accelerated parallel lagged-Fibonacci pseudorandom number generation," M.S. thesis, University of Tennessee, Knoxville, TN, USA, 2006.
[38] N. S. Altman, "Bit-wise behavior of random number generators," *SIAM J. Sci. Statist. Comput.*, vol. 9, no. 5, pp. 941–949, 1988.
[39] D. E. Knuth, *The Art of Computer Programming, Seminumerical Algorithms*, vol. 2, 3rd ed. London, U.K.: Pearson Education, 1997.
[40] R. Anderson, "On Fibonacci keystream generators," in *Proc. Int. Workshop Fast Softw. Encryption*, Dec. 1994, pp. 346–352.
[41] W. W. Tsang *et al.*, "Development of cryptographic random number generators," Dept. Comput. Sci. Inf. Syst., Univ. Hong Kong, Hong Kong, HKU CSIS Tech. Rep. TR-2003-07, 2003.
[42] M. K. Chetry and W. V. Kandaswamy, "A note on self-shrinking lagged fibonacci generator," *IJ Netw. Secur.*, vol. 10, no. 3, pp. 185–187, 2010.
[43] B. B. Bordel and R. Alcarria, "Secure sensor data transmission in 5G networks using pseudorandom number generators," in *Proc. Res. Briefs Inf. Commun. Technol. Evol. (ReBICTE)*, vol. 3. 2017, pp. 1–11.
[44] T. Heer, O. Garcia-Morchon, R. Hummen, S. L. Keoh, S. S. Kumar, and K. Wehrle, "Security challenges in the IP-based Internet of Things," *Wireless Pers. Commun.*, vol. 61, no. 3, pp. 527–542, 2011.
[45] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.
[46] H. Seo, Z. Liu, J. Großschädl, and H. Kim, "Efficient arithmetic on ARM-NEON and its application for high-speed RSA implementation," *Secur. Commun. Netw.*, vol. 9, no. 18, pp. 5401–5411, 2016.
[47] S. F. Barrett, "Arduino microcontroller processing for everyone!" *Synthesis Lectures Digit. Circuits Syst.*, vol. 8, no. 4, pp. 1–513, 2013.
[48] T. Jager and J. Somorovsky, "How to break XML encryption," in *Proc. 18th ACM Conf. Comput. Commun. Secur.*, Oct. 2011, pp. 413–422.
[49] A. Braverman, J. G. Dai, and J. Feng, "Stein's method for steady-state diffusion approximations: An introduction through the Erlang-A and Erlang-C models," *Stochastic Syst.*, vol. 6, no. 2, pp. 301–366, 2017.
[50] J. F. C. Kingman and M. F. Atiyah, "The single server queue in heavy traffic," *Oper. Manage., Critical Perspect. Bus. Manage.*, vol. 57. 2003, p. 40.
[51] A. Hemani *et al.*, "Network on chip: An architecture for billion transistor era," in *Proc. IEEE NorChip Conf.*, vol. 31, Nov. 2002, p. 11.

BORJA BORDEL received the B.S. and M.S. degrees in telecommunication engineering from the Technical University of Madrid, in 2012 and 2014, respectively. He is currently pursuing the Ph.D. degree in telematics engineering with the Telecommunication Engineering School, UPM. His current research interests include cyber-physical systems, wireless sensor networks, radio access technologies, communication protocols, and complex systems.

AMALIA BEATRIZ ORÚE received the B.E. and M.S. degrees in telecommunications systems from the University of Oriente, Cuba, in 1984 and 1998, respectively, and the Ph.D. degree from the Polytechnic University of Madrid, Spain, in 2013. She is currently a Researcher with the Department of Information and Communications Technologies, Institute of Physical and Information Technologies, Spanish National Research Council, Madrid, Spain. She has served as a referee for different SCI journals and international conferences. She has published several research papers in international scientific journals, peer-reviewed workshops, and conferences. Her current research interests include cryptography, Internet of Things security, information security, teaching methods of cryptography, and educational innovation. She is a member of the IEEE Education Society and the IEEE Council on RFID.

RAMÓN ALCARRIA received the M.S. and Ph.D. degrees in telecommunication engineering from the Technical University of Madrid, in 2008 and 2013, respectively. He is currently an Assistant Professor with the E.T.S.I Topography, Technical University of Madrid. He has been involved in several research and development European and National projects related to future internet, Internet of Things, and service composition. His current research interests include service architectures, sensor networks, human–computer interaction, and prosumer environments.

DIEGO SÁNCHEZ-DE-RIVERA received the B.Sc. and M.S. degrees in telecommunication engineering from the Technical University of Madrid. He is currently pursuing the Ph.D. degree with the E.T.S.I Telecommunications, Technical University of Madrid. His current research interests include Internet of Things, sensor networks, and Web development.