SECURITY AND COMMUNICATION NETWORKS
Security Comm. Networks 2016; **9**:1226–1246
Published online 21 December 2015 in Wiley Online Library (wileyonlinelibrary.com). DOI: 10.1002/sec.1399

REVIEW ARTICLE

A survey of lightweight stream ciphers for embedded systems

Charalampos Manifavas[1], George Hatzivasilis[2]*, Konstantinos Fysarakis[2] and Yannis Papaefstathiou[2]

[1] Department of Electrical Engineering and Computing Sciences, Rochester Institute of Technology Dubai, Techno Point Building, Dubai Silicon Oasis, 341055, Dubai, UAE
[2] Department of Electronic and Computer Engineering, Technical University of Crete, Akrotiri Campus, 73100 Chania, Crete, Greece

ABSTRACT

Pervasive computing constitutes a growing trend, aiming to embed smart devices into everyday objects. The limited resources of these devices and the ever-present need for lower production costs, lead to the research and development of lightweight cryptographic mechanisms. Block ciphers, the main symmetric key cryptosystems, perform well in this field. Nevertheless, stream ciphers are also relevant in ubiquitous computing applications, as they can be used to secure the communication in applications where the plaintext length is either unknown or continuous, like network streams. This paper provides the latest survey of stream ciphers for embedded systems. Lightweight implementations of stream ciphers in embedded hardware and software are examined as well as relevant authenticated encryption schemes. Their speed and simplicity enable compact and low-power implementations, allow them to excel in applications pertaining to resource-constrained devices. The outcomes of the International Organization for Standardization/International Electrotechnical Commission 29192-3 standard and the cryptographic competitions eSTREAM and Competition for Authenticated Encryption: Security, Applicability, and Robustness are summarized along with the latest results in the field. However, cryptanalysis has proven many of these schemes are actually insecure. From the 31 designs that are examined, only six of them have been found to be secure by independent cryptanalysis. A constrained benchmark analysis is performed on low-cost embedded hardware and software platforms. The most appropriate and secure solutions are then mapped in different types of applications. Copyright © 2015 John Wiley & Sons, Ltd

KEYWORDS
authenticated encryption; embedded systems; lightweight cryptography; stream ciphers

***Correspondence**
George Hatzivasilis, Dept. of Electronic and Computer Engineering, Technical University of Crete, Akrotiri Campus, 73100 Chania, Crete, Greece.
E-mail: gchatzivasilis@isc.tuc.gr

1. INTRODUCTION

In pervasive computing environments, numerous embedded devices are interconnected in order to provide ambient intelligence applications and various types of enhanced services. The security components of these devices are of significant importance, as they have to continuously communicate information, in many cases sensitive and critical in nature, which must be protected from malicious entities [1,2]. The aforementioned is exacerbated by the fact that these devices may often feature direct interaction with the physical world, via corresponding actuators, potentially compromising users' safety in case of misuse.

Still, the deployed protective mechanisms must comply with the constrained resources of the target embedded systems. Lightweight cryptography (LWC) [3] focuses on cryptographic mechanisms suitable for such systems. Lightweight block and stream ciphers are expected to perform well in embedded devices and mainly provide confidentiality and data integrity [4–6]. Block ciphers are the most common choice in LWC as their design is often more straightforward and their security properties are well-studied. Moreover, there are communication protocols that cannot be implemented using stream ciphers. Stream ciphers' main drawback is the initialization process, necessary before any communication takes place. On the other hand, stream ciphers are suitable for applications where the plaintext length is unknown or continuous, like network streams, including military applications where the cipher stream is deployed under a secure setting and fed to devices that are expected to function in insecure and hostile environments. Also, stream ciphers are typically fast, compact, and consume low power, making them an attractive choice for resource-constrained devices, like low-power radio frequency identification (RFID) tags.

In recent years, stream ciphers have gained ground as several research efforts have investigated secure stream cipher designs. A milestone event in pertinent research efforts was the eSTREAM competition [7], held by European Network of Excellence for Cryptology from 2004 to 2008, which promoted the design of efficient and compact stream ciphers suitable for wide adoption. In 2012, the International Organization for Standardization and the International Electrotechnical Commission (ISO/IEC) standardized stream ciphers for LWC in the ISO/IEC 29192-3:2012 [8] standard. Since then, several new designs based on eSTREAM concepts and other novel structures have been proposed.

In addition to confidentiality and integrity, authentication is another important security property that must be considered. Thus, except from designing individual ciphers, efforts have been made to implement symmetric cryptosystems that provide authenticated encryption functionality. The interest in such schemes is expected to increase because of the ongoing Competition for Authenticated Encryption: Security, Applicability, and Robustness (CAESAR) [9]. CAESAR is organized by the international cryptographic research community and intends to select a portfolio of algorithms offering advantages over Advanced Encryption Standard Galois/Counter Mode (AES-GCM) [10] and being suitable for widespread adoption. The final selection will be announced in 2017.

Hardware implementations of stream ciphers are mostly suitable for ultra-constrained devices, as they typically implement the exact cryptographic functionality without redundant components (e.g., general purpose processing units). The complexity of a design is defined by the gate equivalent (GE) metric, with the authors considering an upper limit of 3000 GEs for LWC [11–13]. However, reported characteristics of each cipher are closely related to a specific setting and cannot be directly transferred to other applications. In contrast, software implementation of ciphers retains portability across devices with the same microprocessor [14]. On the other hand, they are mostly appropriate for more powerful devices, as they presuppose the existence of a microprocessor.

In this work, the authors present a survey of lightweight stream ciphers and relevant authenticated encryption schemes in hardware and software. The implementation details of the typical cipher versions are presented, as stated by their designers, and a benchmark analysis is performed on low-cost embedded devices. The remainder of this paper is structured as follows: Section 2 presents related work and relevant surveys; Section 3 presents the general design approaches for stream ciphers; Section 4 presents the stream ciphers that are typically examined in the context of LWC and their main features; Section 5 features performance evaluation and benchmark of the presented algorithms and identifies the most suitable and secure solutions for different types of applications; Section 6 refers future work of stream cipher design and application, with the concluding remarks following in Section 7.

2. RELATED WORK

Stream ciphers have been applied in several real-time applications (e.g., mobile phone telephony) because of high performance in hardware. In 2004, a performance analysis of these mainstream ciphers was presented in [15]. The main stream cipher designs and the basic hardware-efficiency and security trade-offs were identified.

In 2007, the development of the lightweight block cipher PRESENT signified a milestone in LWC with several lightweight designs being proposed afterwards. A first survey on LWC was held in the same year [16], reviewing several symmetric and asymmetric ciphers for embedded hardware and software.

The eStream project advanced the research efforts for compact stream cipher designs. A survey and software benchmark of stream ciphers for wireless sensor networks was presented in 2007 [17], focusing on eSTREAM candidates. Several ciphers were deployed and evaluated on the same micro-controller platform. Similarly, hardware results for selected eSREAM candidates in application-specific integrated circuit (ASIC) are presented in [18].

The ISO/IEC standardization effort in 2012 enhanced cryptanalysis results for secure stream cipher design. Cryptanalysis techniques for stream ciphers and relevant attacks are presented in [19] and [20], respectively.

In 2013, a comparative analysis of newer stream cipher implementations for embedded systems was conducted in [13]. The most efficient implementations in embedded hardware and software were evaluated based on a set of predefined metrics.

Table I summarizes the main features of these related studies. In this paper, we survey the main traditional and lightweight stream ciphers along with the latest design and cryptanalysis results in the field. We include authenticated encryption schemes and relevant CAESAR candidates that are based on stream ciphers. We indicate the safe ciphers and perform a benchmark analysis on embedded hardware and software platforms.

3. STREAM CIPHER DESIGN

3.1. General features

Stream ciphers are an alternative type of symmetric cryptosystem to block ciphers [21]. They are based on the idea of the one-time pad (OTP) cipher, known as Vernam cipher [22]. OTP utilizes a completely random keystream, and each digit of the keystream is combined with one digit from plaintext to produce a digit of ciphertext. OTP was proved to be unbreakable [23]; however, the keystream digits have to be completely random, and moreover, the keystream

Table I. Surveys of lightweight stream ciphers.

Authors	Year	Main focus	Evaluated stream ciphers
Batina *et al.* [15]	2004	Performance and security tradeoffs for traditional stream ciphers	RC4, A5/1, E0
Eisenbarth *et al.* [16]	2007	Lightweight cryptography implementations	Salsa20, LEX
Fournel *et al.* [17]	2007	Survey and software benchmark of stream ciphers for WSN	eStream candidates
Good *et al.* [18]	2007	Hardware benchmark for stream ciphers	eStream candidates
Bokhari *et al.* [19]	2012	Cryptanalysis techniques for stream ciphers	Traditional stream ciphers
Manifavas *et al.* [13]	2013	Lightweight cryptography implementations	Traditional and lightweight stream ciphers, eStream finalists
Banegas *et al.* [18]	2014	Attacks on stream ciphers	Traditional stream ciphers
This study	2015	Survey and hardware/software benchmark analysis of stream ciphers and authenticated encryption schemes	eStream finalists, ISO/IEC standardized stream ciphers, CASEAR candidates based on stream ciphers, latest lightweight stream cipher proposals

must have the same length as the plaintext. Thus, OTPs introduce significant practical management and maintenance problems and are considered impractical for wide use.

Stream ciphers address these issues by sacrificing a degree of security: they apply a secret key, which is used to generate a pseudorandom keystream. The secret key is the symmetric key, and the pseudorandom keystream generated is used in the same way as the keystream in OTPs. However, the fact that the keystream is not completely random introduces security concerns. The keystream must be random enough to ensure that if an attacker knows the keystream, he cannot recover the secret key or derive the cipher's internal state.

The keystream period is a significant security factor for stream ciphers. It stands for the number of encrypted bits or blocks that are processed before the keystream is repeated. If this is exhausted and the keystream is repeated, cryptanalysis could potentially decrypt the sequential ciphertext bits [21]. Thus, the keystream period should be larger than the size of the plaintext that will be encrypted. If the keystream period elapses, a different key or nonce must be used to re-initialize the cipher; a longer period implies fewer instances that this will have to take place.

After initialization, all stream ciphers require a few clock cycles to encrypt a bit. Yet, the initialization process is also essential in determining the applicability of a stream cipher. Stream ciphers with fast initialization phase are suitable for applications where many short messages have to be processed. On the other hand, when few and large messages must be encrypted, stream ciphers with fast encryption are appropriate. Thus, stream ciphers cannot be easily classified in terms of performance, as it depends on both the initialization and encryption/decryption operations.

3.2. Cryptanalysis and attacks

Attackers aim to exploit stream cipher designs in order to discover the key that is used for decryption/encryption. Then, they can recover part or the whole communication processed by this key. Passive attacks exploit the output or the initialization/resynchronization phases. They mainly analyze design flaws. Active attacks insert, delete or replay ciphertext bits (e.g., fault attack), and try to misuse the communication channel. They are countered by strong data integrity and authentication mechanisms.

Exhaustive key search is a brute force attack where an attacker tries out all possible combinations until it finds out the key. The computational complexity cannot exceed the key size in bits but can also be lower. All ciphers are vulnerable to these attack. The lower bound of the exhaustive key search complexity for a cipher denotes the intuitive limit that all the other attack types try to improve upon.

The re-initialization process of a cipher design is another target of attacks. If the input is related to the internal state without sufficient non-linearity when the new keys are produced, an attacker can determine the *related keys* and the initial one. *Chosen-IV* attacks exploit weaknesses in key scheduling and extract the initial state from the memory.

Side-channel attacks measure electromagnetic emission or power consumption during the processing of the data from the cipher. The goal is to derive useful informaiton about the algorithm's internal processes.

In *distinguishing* attacks, the attacker tries to distinguish the key stream from a purely random sequence. Such attacks can turn into complete key recovery.

Correlation attacks analyze the linear functions of the cipher and determine the produced keystream by observing the output bits. *Linear* attacks correlate the linear functions of selected keystream bits or linear functions of selected keystream and state bits. Similarly, *algebraic* attacks resolve systems of algebraic equations that are utilized for the production of the key or state bits. One popular stream cipher design approach utilizes non-linear components with masking of block ciphers. *Linear masking* distinguishes a non-linear characteristic that exhibits some bias. Missing linear combinations, derived from the linear functions of the cipher, are applied to the output and discover the traces of this distinguished characteristic.

Cube attacks are a relatively new attack type and can be applied to almost any cryptosystem. The attack takes advantage of the existence of a low-degree polynomial representation of a single output bit as a function of the key and plaintext bits. These bits are summed over all possible values of a subset of the plaintext bits to detect linear equations in the key bits, which can be efficiently solved.

Time/Memory/Data tradeoffs explore the general structure of a cipher and summarize the analysis results in large tables. At attack phase, these precomputed tables are utilized in order to retrieve the key of the actual data that is being processed.

Guess-and-determine attacks guess a part of the state and try to reconstruct the whole state based on the keystream that is actually observed.

Slide attacks correlate two copies of the same encryption process that are one round out of order. The two relevant plaintexts are lined up, trying to find a match. The attack reveals the key in the state update of the cipher.

In *divide and conquer* strategy, the cipher is disassembled into basic components. A few bits are determined in each state, and the most vulnerable components are attacked first. Designs that exhibit high-correlation immunity are less vulnerable to such attacks.

A survey of cryptanalysis techniques for stream ciphers is presented in [19], where the exhaustive key search, related key, side-channel analysis, distinguishing, correlation, algebraic, linear masking, time-memory tradeoff, and guess-and-determine attacks are explained. The fault, chosen-IV, cube, and slide attacks are additionally described in a later survey [20], along with the proposed mechanisms to mitigate them.

3.3. Synchronous and self-synchronous ciphers

Stream ciphers generate keystream digits based on their internal state. There are two categories of stream ciphers based on the update operation of the internal state.

Synchronous stream ciphers update the internal state independently from the plaintext or ciphertext data. A sender and a receiver must be synchronized before encryption/decryption takes place, and if part of the message is added or removed, this synchronization is lost. A resynchronization mechanism is, thus, required to maintain communication, introducing complexity and computational overhead. On the other hand, if a bit of the ciphertext message is altered, only a single plaintext bit is affected, that is, the error is not propagated to the rest message. This characteristic is very important in applications like wireless communications, where the error rate is high, but it also makes it difficult to detect errors. Figure 1 illustrates the operation of synchronous stream ciphers. Synchronous stream ciphers are vulnerable to active attacks, as an attacker could disrupt a communication and discover correlations between the altered ciphertext bits and the corresponding plaintext bits. Most of the cryptanalysis efforts focus on known-plaintext attacks.

Self-synchronizing stream ciphers update the internal state based on the *N* previous ciphertext bits; Figure 2 illustrates their functionality. With self-synchronizing stream ciphers, the receiver can automatically synchronize herself with the sender after receiving *N* ciphertext digits. This makes it easier to recover the message even if some bits are added or removed. If a bit is altered, *N* plaintext bits will be affected at most. Thus, it is more difficult to perform active attacks. However, in a cryptographic context, this is also a disadvantage, as up to *N* bits will be decrypted incorrectly for every altered bit. Moreover, self-synchronizing ciphers suffer from important security issues. As the key is updated based on previous bits, statistical regularities in plaintext are disclosed in the keystream. Self-synchronizing ciphers are mainly vulnerable to chosen ciphertext attacks, and once the attacker decrypts any part of the message, it can decipher more. These features make the design of secure self-synchronizing stream ciphers a difficult task, which also explains the rarity of new proposals. In eSTREAM project, only the ciphers SSS [24] and MOUSTIQUE [25] are self-synchronizing but both are broken. As stated in [26], there is little interest in self-synchronization as they are not supported by the industry today.

Both synchronous and self-synchronizing ciphers can be used for encryption/decryption, providing confidentiality, but also as pseudorandom number generators by encrypting a long sequence of zeros. Integrity and authenticity are additional features that are accomplished by more advanced primitives. Authenticated encryption (AE) schemes utilize the pure ciphers (block or stream) and achieve confidentiality, integrity, and authenticity. AE processes a message and produces the ciphertext, along with a message authentication code (MAC) for authenticity and integrity check. AE schemes for embedded system are detailed in sub-section 4.5.

3.4. Design taxonomy

Several stream cipher designs are proposed in the literature. In this sub-section, we introduce the most suitable of them for LWC. Figure 3 illustrates the stream cipher design taxonomy.

3.4.1. Main design types.

Stream ciphers are usually built using feedback shift registers (FSRs). At each cycle, an FSR gets one bit as input and produces one bit at the output. The input bit is a function of the previous state. There are two types of FSRs based on the feedback function: the linear feedback shift registers (LFSRs) and the feedback with carry shift registers (FCSRs).

Linear feedback shift registers are a common choice for stream ciphers (e.g., [27–29]). They are easy and fast to implement in hardware, and their properties can be studied and analyzed mathematically. However, their linear nature means they are not secure unless they are used

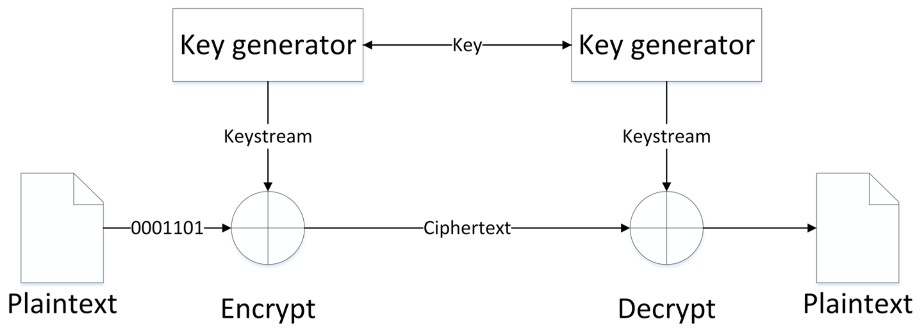

Figure 1. Synchronizing stream cipher.

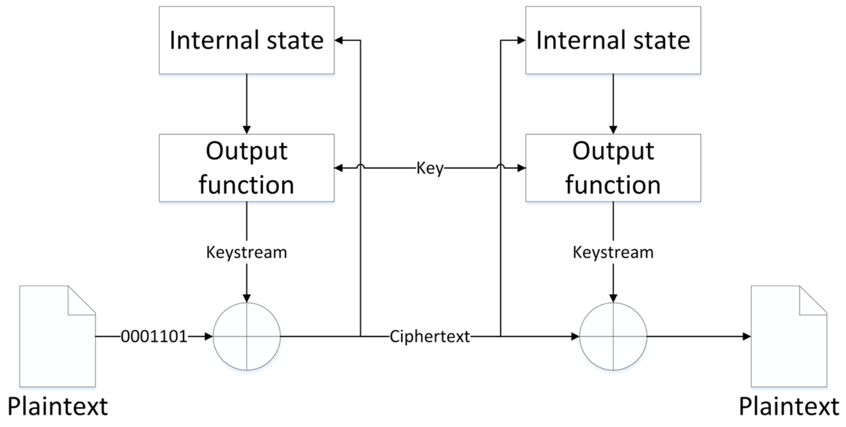

Figure 2. Self-synchronous stream cipher.

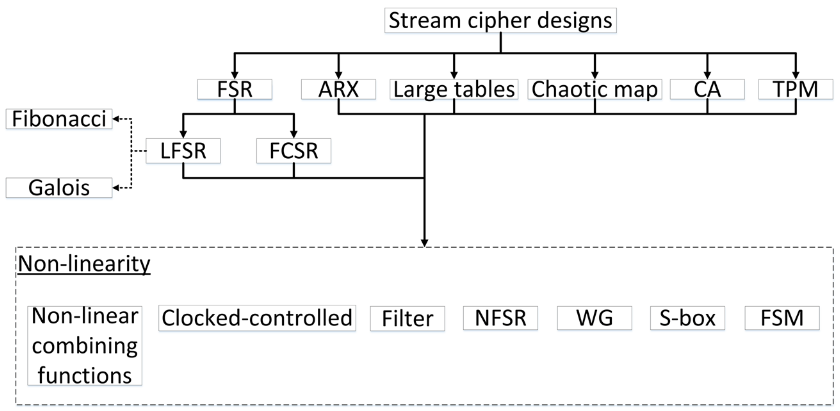

Figure 3. The taxonomy of the stream cipher designs.

along with additional components to introduce nonlinearity. There are two styles of LFSR according to the shift operation. Fibonacci LFSR is the typical style, where each bit is copied to its right neighbor and the bits are right shifted. The rightmost bit is the output. The new leftmost bit that is inserted is the parity of some special bits, called taps. Galois LFSR is the alternative style, where the bits are right shifted, except from the tap bits. The rightmost bit of the taps is XOR-ed with the previous output bit before the copy is done. The original rightmost bit from the taps is both the new leftmost bit that is inserted and the output. Galois LFSRs are considered more efficient in hardware than Fibonacci LFSRs as they introduce lower gate delay and, thus, lower clock cycle time.

Feedback with carry shift registers are arithmetic analogs of LFSRs [30]. They are similar to LFSRs but with additional memory for maintaining a carry from one stage to the next, forming a finite device. They can be efficiently implemented in a parallel architecture. However, FCSRs are periodic in nature and cannot be directly applied in cryptography [31].

Except from the stream ciphers that are based on FSRs, there are several alternative design types.

Ciphers that use modular **add, rotate, and XOR operations (ARX)** are popular because of their fast and cheap implementation (e.g., [32–34]). These operations run in constant time, reducing the feasibility of timing attacks. ARXs produce fast and compact implementations but their security properties are not well-studied.

In contrast to ARX ciphers that use simple operation, **large table** designs are proposed for high speed in software (e.g. [35,36]). The state is maintained in large tables and the content is updated at each round by non-linear functions. This approach exhibits among the fastest results at the cost of high memory consumption in both hardware and software.

A **cellular automata (CA)** [37] is a regular grid of cells, where each one has a finite number of states. The cells update their state according to a fixed rule. The rule determines the new state of each cell based on the current state of a cell and its neighbours. Given the rule, it is easy to determine the future states but very difficult to estimate the previous ones. In cryptography, CAs can act as an one-way function whose inverse is hard to find (e.g., [38,39]). Their hardware implementation is simple, while word-wise implementations can be efficient in software. Moreover, parallel transformations are also possible, allowing to increase throughput.

Chaotic maps [40] are maps that exhibit a chaotic behavior. They can be parameterized by a discrete-time parameter, like an iterated function. In cryptography, their ordinary setting is the mixture of the plaintext with a keystream using bitwise operations [41]. The optimum outcome is a pseudorandom keystream with good statistical properties. Their main advantage is the level of confusion and diffusion that they achieve. Moreover, they can be directly implemented in hardware.

A **Tree Party Machine (TPM)** [42] is a multi-layer feed-forward neural network. Although it is not a common choice in cryptography, it has been applied to stream cipher design [43] because of the level of randomness that it achieves. In the common topology, two TPMs update their weight vectors by mutual learning based on each other's output in order to achieve the weight space synchronization. The common inputs vary dynamically at each learning iteration, but kept identical throughout all mutual learning time. The TPM properties guarantee that given the mutually exchanged TPM output over an open channel, an eavesdropper cannot exactly determine which internal weight vectors has been updated each time. Thus, the eavesdropper cannot derive the final synchronized weight vectors. These final weight vectors are mapped to a shared key.

3.4.2. Non-linearity.

Several schemes have been proposed to increase the security of the main LFSR design and introduce the needed nonlinearity. Examples include the non-linear combining functions, the clock-controlled generators and the filter generators. With a **non-linear combining function**, the outputs of several parallel LFSRs are passed to a non-linear Boolean function. A **clock-controlled generator** uses two LFSRs. In the ordinary setting, an LFSR is stepped regularly. The generator implies that the LFSR must be clocked irregularly according to the output of the second LFSR. When **filter generators** are used, the entire state of an LFSR is fed into a non-linear filtering function. Combinations of all these schemes are also examined in the literature (e.g., [44,45]).

Non-linear feedback shift registers (NFSRs) are another core component, utilized by stream ciphers to provide non-linearity (e.g., [44–46]). Furthermore, they are more resistant to cryptanalytic attacks. Yet, only a few theoretical results exist for NFSRs. The security level of ciphers that make use of NFSRs depends on the difficulty of analyzing the design itself. In hardware, they are typically slightly more complex than LFSRs and FCSRs. However, it is difficult to design a good NFSR, as there are no simple and precise guidelines on building strong non-linear Boolean functions. These functions are used in NFSRs to determine the balance between the zeroes and ones in the output, the nonlinearity, the algebraic degree, and the correlation immunity.

The **Welch–Gong (WG)** functions are another family of structures which, in contrast to NFSRs, can be analyzed mathematically [47]. They are proven to guarantee a variety of randomness properties, like ideal two-level autocorrelation, balance, long period, ideal tuple distribution, and high and exact linear complexity. These properties satisfy security requirements for encryption and authentication [28].

The **Substitution-boxes (S-box)** are implemented in the form of lookup tables which map m input bits to n output bits. They are a common and well-studied choice in block ciphers, used to obscure the relationship between the key and the ciphertext. S-boxes are also utilized by many stream ciphers (e.g., [27,31,48–51]) in order to enhance their non-linearity features.

Finite state machines (FSMs) are mathematical models for computing sequential logic. At each time point, the FSM is in one of a finite number of states. A transition to a new state is performed as a result of a triggering event or condition. In stream cipher design, FSMs perform well in software and enhance protection against guess-and-determine attacks [27].

3.4.3. Design guidelines.

The maintenance of the internal state is the most space-consuming part of a cipher. This burden is even heavier in stream ciphers, as, comparatively, a larger percentage of the hardware implementation is used for storing the internal values. For example, a hardware implementation of the Grain stream cipher, presented in [44], requires 1294 gates, with the bulk of them being used for this task. In more detail, the internal state must be at least twice the security level in bits [44]. The nonlinearity functionality, on the other hand, can be compact in hardware.

Thus, reducing the area factor of a stream cipher is a difficult task. Stream cipher designers typically aim to reduce the size of the registers and the complexity of the Boolean and filtering functions.

In hardware, an LFSR uses flip-flops to maintain its state and XOR gates to compute the feedback. In order to decrease the hardware area, designers can decrease the size of the internal state and the number of XORs. However, a cipher with short registers and a simple feedback function can be vulnerable to cryptanalysis.

An NFSR uses more complex Boolean operations or S-boxes to form the feedback function. The area occupied by these Boolean operations is larger than the one of the XORs of an LFSR, but the resulting number of required flip-flops is smaller.

Word size is another important factor. In this regard, bit-oriented and binary-oriented stream ciphers are efficiently implemented in hardware, while word-oriented ciphers are preferable in software.

In terms of hardware implementations, the most popular approach is to exploit LFSRs and NFSRs ([44–46]). For software implementations, using table-driven ciphers ([35,41]) or building blocks from block ciphers ([27,31,48–51]) are suggested when aiming to achieve high throughput rates. To increase the provided level of security, larger internal states and mixing of algebraic domains ([28,29]) are often adopted. In general, it is proposed to use simple operations that are implemented by the native processor's instruction set to increase speed.

4. STREAM CIPHER IN LWC

In this section, the authors survey stream ciphers that have been applied in embedded systems. The widely used ciphers and their vulnerabilities are highlighted, then focusing on recent research efforts and standards for LWC.

4.1. Traditional stream ciphers

Traditional ciphers for stream encryption and pertinent security issues are presented in [21]. As referred in said work, ciphers Rivest Cipher 4 (RC4) [32], A5/1 [52], and E0 [52], although not completely compromised, suffer from serious security vulnerabilities and should not be used in new applications.

Rivest Cipher 4 is the most widely used software stream cipher and is included in popular protocols and standards, like Wired Equivalent Privacy (WEP), Wi-Fi Protected Access (WPA), and Transport Layer Security (TLS). It has a simple design and is remarkably fast. It relies solely on byte manipulations, without an LFSR. RC4 is efficient in both hardware and software. However, security issues have been reported because of its dated design, including criticism for its tenuous key scheduling process. Specifically, RC4 does not take a separate nonce with the input key, leaving it up to the communication protocol to specify how to combine these two parameters and produce the keystream. One solution is a strong MAC that hashes the key with a nonce, but analysis revealed biased outputs of the keystream [53]. Thus, the keystream bytes can be correlated with the key ([54,55]), a fact that was exploited, along with related effects, to break WEP ([56,57]). The latest attack on RC4 requires 224 connections [58]. Although there are no practical attacks on the cipher itself, the latest results speculate that well-equipped agencies (e.g., state-funded entities) may have better attacks that render the cipher insecure.

A5/1 was designed to provide over-the-air communication privacy for the Global System for Mobile cellular telephone standard has been widely used in GSM telephony in Europe and the USA. In its design, three combined LFSRs with irregular clocking are used to encrypt bursts of traffic, as is required in GSM. A5/1 is nowadays considered insecure, as attacks both in the cipher itself and its implementations in GSM have been presented. Most of the cryptanalysis was based on time–memory tradeoff and other known-plaintext attacks, with many of them breaking the cipher in a few minutes or seconds ([59–62]). In GSM, active attacks and ciphertext-only analysis prove the protocol to be insecure even when in the presence of a secure cipher [63]. The latest attacks utilize parallel computing platforms, like field-programmable gate array (FPGAs) and general-purpose computation on graphics processing units, to launch practical attacks on A5/1 and GSM at runtime ([64,65]).

E0 is used in Bluetooth. It uses four shift registers (SHR) to produce a sequence of pseudorandom numbers, which are XOR-ed with the data. It is also vulnerable, and practical attacks have been reported both for the cipher and the Bluetooth protocol. Typically, E0 uses 128-bit key. However, analysis has decreased the security level to 65 bits even when longer keys are used ([66,67]). In Bluetooth, statistical and known-plaintext attacks have further decreased the security to 38 bits ([68,69]).

Block ciphers can operate as stream ciphers too. The NIST modes of operation [70] indicate three ways to implement this functionality: the cipher feedback (CFB) mode for implementing self-synchronized ciphers, and the output feedback (OFB) and counter (CTR) modes for implementing synchronized ciphers. The implementation involves the encryption functionality for the most part, as the decryption is almost identical under these modes. Security issues on these stream modes are discussed in [20]. AES is the most investigated block cipher in terms of its stream versions. The cipher is safe with biclique cryptanalysis achieving slightly better results than exhaustive search [71]. AES in CTR mode (AES-CTR) is currently the only secure and widespread solution for stream encryption [21]. Figure 4 illustrates the encryption process of AES-CTR. PRESENT, the standardized block cipher for LWC (ISO/IEC 29192), has also been studied but was found to be inferior to the dedicated stream ciphers.

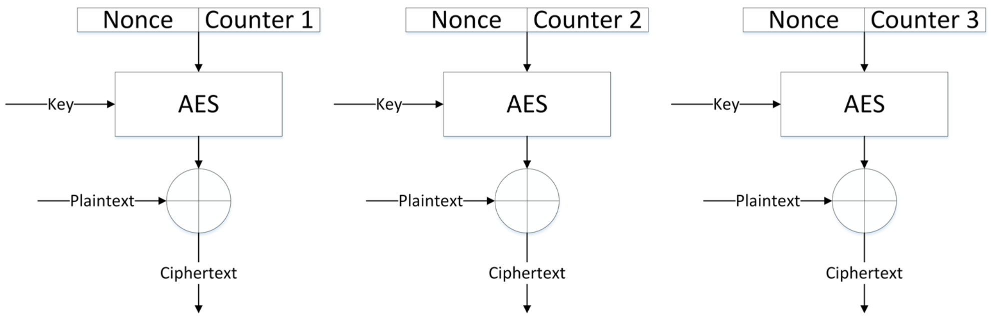

Figure 4. Advanced Encryption Standard (AES) in counter mode of operation.

4.2. The eSTREAM project

The eSTREAM [7] project was part of the European Network of Excellence for Cryptology II within the Information & Communication Technologies Programme of the European Commission's Seventh Framework Programme and delivered a small portfolio of promising stream ciphers.

Two profiles of ciphers for software and hardware implementations were defined. Profile 1 consists of ciphers with high throughput in software that are faster than the 128-bits AES-CTR. Finalist ciphers include Salsa20/12 [33], Rabbit [41], HC-128 [35], and SOSEMANUK [27]. Profile 2 consists of ciphers that are suitable for highly constrained environments and are more compact in hardware than the 80-bits AES. The finalists include Grain [44], Trivium [72] and MICKEY 2.0 [45].

Several ciphers were submitted and rejected because of security vulnerabilities or lower overall performance. From a total of 34 ciphers, only two self-synchronizing stream ciphers were submitted, and both were proven to be insecure. The finalists were also cryptanalyzed and were found to be secure against all attacks that are faster than the exhaustive key search attack. However, newer cryptanalysis attempts have reported some tangible results for some of these ciphers.

In LWC, the most notable ciphers are the two finalists of Profile 2, that is, Grain and Trivium. They are very attractive primitives and, thus, have been extensively investigated, as their compact and efficient hardware implementation can be applied to ultra-constrained devices. Nevertheless, the finalists of Profile 1 (i.e. Salsa and Rabbit) are also extensively examined for compact embedded software applications as they are fast and do not consume significant resources. Thus, the aforementioned four ciphers, along with AES-CTR, are used as a benchmark for newer stream cipher proposals.

4.2.1. Profile 1 software-oriented ciphers.

Salsa20/r, where 'r' stands for the iterations of the round function, is an eSTREAM finalist for software implementations. It uses 256-bit keys and 128-bit initialization vectors (IVs). Three variants have been proposed to cover the tradeoff between security and performance, catering for the different application needs. Salsa20/20 is designated for encryptions in typical cryptographic applications, while the variants Salsa20/12 and Salsa20/8 offer faster but less secure operation. Its design is based on simple operations of addition, modulo 232, bit rotation and bitwise XOR (ARX), which are efficiently implemented in software. The encryption and decryption operations are identical, allowing for a more compact implementation. Salsa20 is included in the Crypto++ cryptographic library [73] as a high-speed stream cipher.

The most lightweight software implementation [16] requires 1452 bytes of code and 18,400 cycles for encryption. The most compact hardware implementation [18] occupies 12 126 GE, which is far beyond the scope of LWC.

Attacks have been reported for the reduced versions of Salsa20 [74]. For the variants Salsa20/20 and Salsa20/12, no better attack than the exhaustive key search has been reported.

Rabbit is a synchronous stream cipher that was designed for high software performance, with very fast key setup and encryption functionality. It is applicable to internet protocols and other applications which process large amounts of data or a large number of packets; it is also one of the most efficient software proposals of eSTREAM. Rabbit was patented by its designers but the algorithm is free to use. It is included in the cryptographic library for embedded systems CyaSSL [75] and is standardized in the ISO/IEC 18 033-4:2011 [76] standard for information technology security techniques.

Rabbit uses simple and basic operations, taking advantage of the features of modern processors. It provides strong non-linearity that is not based on NFSR and S-boxes; Rabbit is based on the ideas of chaotic maps.

The key size is 128 bits, and the IV is 64 bits. Its software implementation on a 1.7 GHz Pentium 4 system consists of 1976 bytes of code and needs 486 cycles for key setup and 5.1 cycles for encrypting a byte [77]. Rabbit was also a candidate for the eSTREAM profile 2. Its most compact hardware implementation requires 3800 GE, achieving 88 Kbps of throughput [77].

Practical fault analysis attacks require around 128–256 faults and precomputed table of size 241.6 bytes to recover the complete internal state in about 238 steps [78].

HC is a cipher that has two main variants HC-128 and HC-256 [36] for 128-bit and 256-bit keys, respectively, with 128-bit IVs. It uses two large secret tables, each one with 512 32-bit elements and operates on 32-bit words. At each step, an element is updated using a non-linear feedback function, and a 32-bit output is generated by a non-linear output filtering function. HC is suitable for parallel computing and modern superscalar microprocessors as three consecutive steps can be computed in parallel. The feedback and output functions of each step can be performed simultaneously. It is now included in the CyaSSL crypto library.

As a table-driven cipher, it can yield impressive performance in software when large streams of data are encrypted. However, because of the initialization overhead, when small streams are processed, the performance is low (as frequent re-initialization is required). It is estimated that, in hardware, 52 400 GE would be occupied just to cover the memory requirements [18].

No significant cryptanalytic advances have been reported on HC [79,80], and the cipher is considered safe. Its authors estimate a keystream period larger than 2256 bits.

SOSEMANUK is a synchronous stream cipher. The key size varies between 128 and 256 bits, and the IV size is 128 bits. However, the provided security level is the same, at 128 bits, for all key sizes. It adopts some of the design principles of the stream cipher SNOW 2.0 [81] and the block cipher Serpent [82]. SOSEMANUK performs better than SNOW 2.0 as it has a faster IV setup phase and needs less amount of static data. It uses an LFSR and a finite state machine (FSM) and operates on 32-bit words. For the setup phase, a 24-round Serpent is used to fill the LFSR and the FSM. At the keystream generation phase, four output words of the FSM are fed to Serpent's third S-Box and then XOR-ed to the LFSR's output words.

In software, its implementation takes about 2000–5000 bytes of code depending on the platform and the compiler, and it achieves a throughput of 360 Kbps [27], while its hardware implementation occupies 18819 GE and achieves 3200 Kbps of throughput [18].

An improved differential fault analysis attack requires around 4608 faults, 235.16 SOSEMANUK iterations and 223.46 bytes storage to recover the inner state [83]. It takes about 11.35 hours on a PC to perform the attack.

4.2.2. Profile 2 hardware-oriented ciphers.

Grain was designed with an easy and compact hardware implementation in mind. It is a synchronous stream cipher and utilizes both LFSR and a non-linear filtering function. The LFSR ensures a minimum keystream period and balances the output. The filtering function is considered as a type of NFSR and introduces nonlinearity to the cipher. The output of the LFSR is masked with the input of this NFSR to balance its state.

Grain adopts a bit-oriented architecture, which is appropriate for constrained hardware implementations. The simplest implementation produces 1 bit/cycle. However, it also offers the option to increase its speed by increasing the length of the processing word; one can increase the number of bits at the expense of more hardware (the rate can be increased up to 16 bits/cycle). It provides higher levels of security compared with the well-known A5/1 and E0 ciphers, while providing small hardware complexity.

Grain uses 80-bit keys and 64-bit IVs. It requires about 1294 GE for 1 bit/cycle and 3239 GE for 16 bits/cycle [44]. In [84], a software implementation is reported for the 1 bit/cycle rate that occupies 778 bytes of code and requires 107 366 cycles for initialization and 617 cycles to generate the output.

Grain-128 [85] is a version of the Grain cipher that supports 128-bit keys and 96-bit IVs, being designed for applications that require higher level of security. The minimum word length is 1 bit, and the maximum is 32 bits (the latter being a popular choice for software implementations). It requires 1857 GE for 1 bit/cycle and 4617 GE for 32 bits/cycle. Furthermore, Grain-128 is used as a building block for the state of the art hash function SQUASH [86].

Several cryptanalysis results have been reported because its selection in eSTREAM. The latest study [87] presents a differential fault attack on the Grain family of ciphers, under reasonable assumptions, which exploits certain properties of the Boolean functions and corresponding choices of the taps from the LFSR. An attacker injects a small number of faults and recovers the secret key.

Trivium is an eSTREAM Profile 2 finalist, and a standardized stream cipher for LWC (ISO/IEC 29192-3:2012). Its designers intended to explore how far a stream cipher can be simplified without sacrificing its security, speed, or flexibility. It is a synchronous, bit-oriented, stream cipher, which uses 80-bit keys and 80-bit IVs. It applies three SHR and forms a nonlinear internal state to avoid building nonlinearity mechanisms for the keystream output.

In hardware [88], its implementation in standard complementary metal-oxide semiconductor (CMOS) technology requires 2017 GE. An implementation in a custom design with dynamic logic and C2MOS flip-flops only occupies 749 GE.

Even though it was not designed for software applications, it performs reasonably well in software as well [68]. The initialization process requires 2050 cycles, the key setup 55 cycles, and the stream generation 12 cycles/byte.

Because of its simplicity, several attacks have been reported. An improved differential fault analysis attack can recover the inner state of the cipher by just injecting 2 faults and using 420 keystream bits [89].

MICKEY 2.0 (Mutual Irregular Clocking KEYstream generator) is the third finalist of eSTREAM Profile 2. It uses a Galois LFSR and a NFSR with irregular clocking to introduce nonlinearity as well as some novel techniques to guarantee period and pseudo-randomness. It is worth noting that there were 80 stages for the two registers in the first version of the cipher, but, because of security issues that were noticed during the early phases of eSTREAM, the states were later increased to 100. Its key size is 80 bits and the IV can vary from 0 to 80 bits. The

240 keystream bits can be generated from each pair of key/IV, and each key can be used with up to 240 different IVs of the same length. A hardware implementation occupies 3188 GE [90]. As with Grain and Trivium, a differential fault attack on MICKEY 2.0 was demonstrated in [91]. The attack requires around 214.7 faults, as MICKEY 2.0 is more complex than the other two finalists. Related key attacks with 65 key/IV pairs break MICKEY 2.0 with 0.9835 probability [92].

The **MICKEY-128 2.0** [93] version was designed to provide higher security. The key size is 128 bits and the IV varies from 0 to 128 bits. In this version, the stages of the two registers were increased from 128 to 160. At 170 MHz clock frequency the cipher achieves a maximum throughput of 170 Mbps. The hardware implementation [90] requires 5039 GE (higher than the 3000 GE limit for LWC). The cipher performs well on hardware. Yet, the irregular clocking mechanism indicates that the cipher cannot be directly parallelized. Similarly with MICKEY 2.0, MICKEY-128 2.0 is vulnerable to related key attacks [92]. The success probability for 113 related key/IV pairs is 0.9714.

4.3. The International Organization for Standardization/International Electrotechnical Commission standard for lightweight cryptography

International Organization for Standardization/International Electrotechnical Commission 29192-3:2012 [8] standardized stream ciphers for LWC. The standard includes two stream ciphers: Trivium and Enocoro [48]. Figure 5 illustrates the two designs.

The eSTREAM Profile 2 finalist Trivium, as described in the previous sub-section, is efficient in hardware and performs reasonable in software. **Enocoro** was designed by Hitachi in 2007 and is their fourth cryptographic algorithm to become a standard. Enocoro achieves the encryption process of AES with 1/10th the amount of power consumption. This is also the main advantage of the cipher, which makes it appropriate for providing basic security functionality in compact control equipment and sensors.

The Enocoro family consists of Enocoro-80 and Enocoro-128 [49], for 80- and 128-bit keys, respectively. Enocoro uses 64-bit IVs and adopts a byte-oriented design with an S-box, which performs well both in hardware and software. It produces 1 byte per round and up to 264 bytes for each pair of key and IV.

In hardware, Enocoro-80 requires 2700 logic gates [48], which are comparable with the relevant implementations of the other eSTREAM Profile 2 finalists. Enocoro-128 requires 4100 logic gates for 3520 Mbps of throughput at 440 MHz [49].

In software, Enocoro-128 requires 4869.5 cycles to initialize and achieves a 46.3 cycles/byte throughput [49]. The initialization phase is long, like Trivium, but the encryption is faster than AES-CTR and Grain.

No practical attacks have been reported, and the cipher is considered secure [94].

4.4. Other lightweight stream ciphers

4.4.1. Lightweight stream ciphers inspired by eSTREAM finalists.

BEAN [31] is a newer proposal that is based on Grain. Its key size is 80 bits, and it is more compact than Grain. BEAN uses two FCSRs and an S-box. The two FCSRs use different primitive polynomials to update themselves, while the S-box introduces better diffusion properties for keystream generation and amends cryptanalysis issues introduced by the FCSRs. It supports binary output production without additional hardware, which is another improvement over Grain. For software implementations, BEAN uses the same amount of memory as Grain but requires less time to generate the keystream bits. However, an efficient distinguisher attack and a state-recovery attack are demonstrated in [95], which are made possible because of the weak output function of BEAN.

Quavium [96] is proposed as a scalable extension of Trivium, and it supports the same key (80 bits), IV

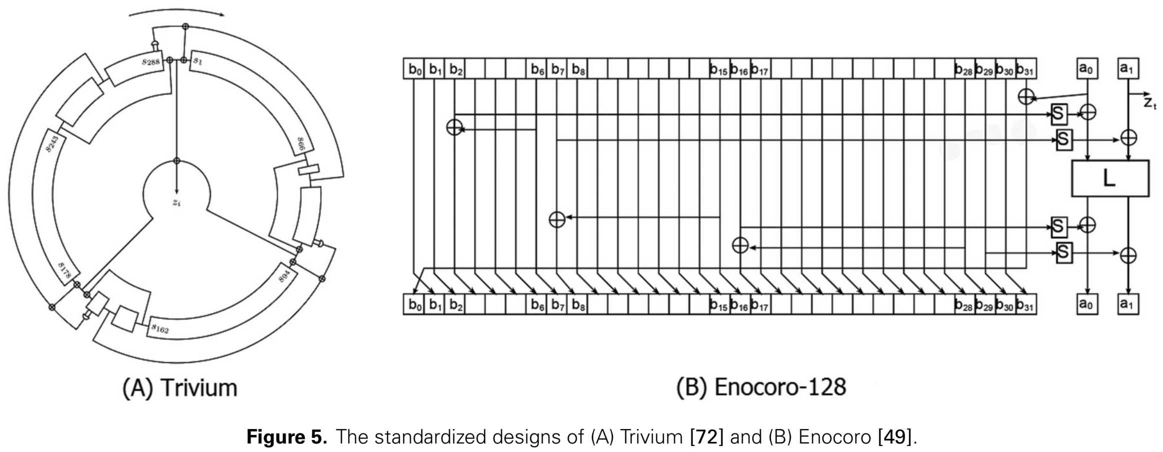

Figure 5. The standardized designs of (A) Trivium [72] and (B) Enocoro [49].

(80 bits), and internal state (288 bits) sizes as Trivium. It is based on four-round Trivium-like SHRs and *k*-order primitive polynomials. Instead of the three SHRs in series connection used in the original Trivium, Quavium uses four Trivium-like SHRs in coupling connection. It can also work with either two or three rounds, as the coupling connection retains the primitiveness of characteristic polynomials.

In hardware, Quavium requires 3496 GE and the three-round version requires 2372 GE, while in software, it is as fast as Trivium, with its three-round version achieving improved performance [96]. In more detail, a Trivium C++ implementation on an Intel Core 2 Duo 2.00~GHz achieves 16.8 cycles/byte, while the equivalent Quavium implementation achieves 17 cycles/byte, and the three-round Quavium 12 cycles/byte.

No cryptanalysis results are presented. However, we consider that the aforementioned improved differential fault analysis attack that is performed on Trivium [89] can affect the security of Quavium because of the Trivium-like SHRs.

CAvium [38] is another new proposal based on Trivium. It utilizes CA with both nonlinear and linear rules, producing a secure design that can reach the desired setup state of a cipher much faster than the equivalent LFSR and NFSR structures. Thus, CAvium significantly reduces the long key setup phase of Trivium (from 1152 to 144 rounds) and counters the cryptanalysis attacks on the reduced round versions, while retaining comparable complexity in hardware. No independent cryptanalysis results are presented.

4.4.2. Other proposals.

A2U2 [46] is a domain specific stream cipher. It was designed for the extremely resource limited environment of printed electronic RFID tags. In this application field, the area occupied for security has to be, at most, around 500 GE, while the power consumption is limited to a few tens of Ws. Throughput should also be reasonable to permit real time interactions with a large numbers of tags.

A2U2 is a synchronous stream cipher that uses 56-bit keys. Principles from both stream and block ciphers were taken into account during its design phase. In order to achieve a small hardware area, A2U2 uses short-length registers supported by compact functional blocks and reuse of hardware components. Its implementation is strongly based on efficient hardware design principles introduced by the KATAN [97] block cipher. More specifically, it adopts an LFSR-based counter and a combination of two NSFRs, as proposed by KATAN. The LFSR operates as a counter during the initialization process and then continues to operate as an LFSR. The feedback function of each NFSR provides the feedback to the other NFSR. Moreover, A2U2 uses irregular changes in the feedback and filter functions.

A2U2 is the most compact stream cipher of the ones examined in this study. It was estimated that the smaller version would require 284 GE and achieve 50 Kbps throughput.

However, an ultra-efficient chosen plaintext attack is presented in [98], which fully recovers the secret key of A2U2. It queries the encryption function on the victim tag twice and solves 32 spare systems of linear equations; it only takes 0.16 s to break the cipher on a laptop computer.

WG-7 [28] is an updated version of the WG cipher, which was a candidate in eSTREAM Profile 2 and which was faster than the other candidates and consumed less memory. WG-7 uses 80-bit keys and 81-bit IVs and is parameterized for RFID tags. It is a synchronous stream cipher and utilizes an LFSR of 23 stages over finite fields F27. The LFSR feeds a nonlinear filtering function which is based on a WG transformation.

However, a distinguishing attack is presented in [99]. The polynomial of the LFSR allows an attacker to distinguish the keystream that is generated by the cipher from a truly random keystream because of the small number of tap positions in the LFSR.

WG-8 [29] is an updated version of WG-7 that counters the abovementioned attack and improves its performance. It uses 80-bit keys and IVs, while the characteristic polynomial of the LFSR consists of eight tap positions (F28). WG-8 inherits the good randomness properties of the WG cipher family and is resistant to most common attacks against stream ciphers. The cipher performs well in embedded software and exhibits low energy consumption. In comparison with other lightweight stream ciphers, WG-8 is 2–15 times faster while consuming 2–220 times less energy. A key recovery attack requires 253 chosen IVs to recover the key with 253.32 complexity, for specifically selected key–IV pairs [100]. The provided security level is still adequate for wireless sensor networks (WSN) and RFID tags when these key–IV pairs are avoided.

CAR30 [39] is a novel stream cipher that, like CAvium, makes use of CA. It utilizes the classical Rule 30 of CA along with a maximum length linear hybrid CA. This combination of a linear and a non-linear CA, and their operation over a number of rounds, reduces the linearity with the adjacent sequence at the production of the keystream, alleviating the relevant security issues of CA-based ciphers. Moreover, the proposed solution can use different key and IV sizes without changing the design and the structure of the cipher. CAR30 provides configurable security and extensibility to any key and IV length. The proposed setting uses 128-bit keys and 120-bit IVs for producing 128-bit blocks of keystream in each iteration. Its statistical randomnes properties were successfully evaluated against the National Institute of Standards and Technology (NIST) statistical test suite for random and pseudorandom number generators for cryptographic applications [101]. No independent cryptanalysis results are presented.

The design can be implemented efficiently in both hardware and software. In hardware, CAR30 achieves higher throughput than Grain and Trivium. In software, it is faster than Rabbit. Moreover, the initialization process is fast (160 cycles), which makes CAR30 suitable for encrypting small messages.

TinyStream [43] is a novel 128-bit stream cipher for WSN, based on TinySec [102]. It employs a Trusted Platform Module (TPM) to achieve keystream randomness, which is the main feature of the cipher. The 128-bit keystream is generated for each state rotation, and this approach passes the full ENT randomness test [103]. In software, it requires 65 024 bytes of ROM and 1 659 184 bytes of RAM. TinyStream is more efficient than TinySec in terms of computation and power consumption. No independent cryptanalysis results are presented.

4.5. Lightweight authenticated encryption

Authenticated encryption is a cryptographic operation that simultaneously provides confidentiality, integrity, and authenticity over processed data. The encryption produces the ciphertext along with an authentication tag. The decryption is combined in a single step with integrity validation. It retrieves the plaintext and produces an error if the authentication tag does not match the ciphertext. Figure 6, illustrates the generic AE scheme. AE is imperative in communication protocols, especially in on-line applications, in order to prevent an attacker from intercepting, tampering, or submitting ciphertexts to the receiver. If the attacker lunches such attacks (e.g., chosen ciphertext attacks), he can decrypt messages and completely reveal the communication data. AES-GCM is currently the most accepted solution for authenticated encryption in mainstream applications. The evolution of pervasive and ubiquitous computing leads to the development of relevant lightweight schemes for resource constrained devices. The CEASAR competition is expected to advance our knowledge in deploying secure and efficient solutions both for mainstream and lightweight schemes.

WG-7 was presented along with a mutual authentication protocol [28] between a reader and several RFID tags, which was based on the cipher. The protocol aims to provide un-traceability, resistance to tag impersonation, reader impersonation, and denial-of-service attacks. However, as mentioned previously, the cipher itself is vulnerable to attacks.

Hummingbird [50] is an ultra-lightweight cipher with a hybrid structure of block and stream cipher, with 256-bit key and 16-bit block sizes. It features better performance than PRESENT on 4-bit microcontrollers. However, it was found to be vulnerable to some types of attacks [104]. Its successor, **Hummingbird-2** [51], was developed with both software and hardware lightweight implementations in mind. It can optionally produce a MAC for each message processed, which can be used to form an authentication protocol fulfilling the requirements of the ISO 18 000-6C protocol [105]. Hummingbird-2 has a 128-bit key and a 64-bit IV and, as its predecessor, is targeted to low-end microcontrollers and hardware implementations in resource-constrained devices, such as RFID tags and wireless sensors. However, its security can be compromised by a related-key attack on the full cipher [106].

Grain-128a [107] is an updated version of the Grain-128 cipher, enhancing the security of the original proposal and having built-in support for authentication. It uses slightly different non-linear functions, in order to counter the attacks reported on Grain-128. Grain-128a utilizes 128-bit keys and 96 bit IVs, and supports variable tag sizes of up to 32 bits. It consists of an LFSR, an NFSR, and a pre-out function. The shift registers are clocked regularly, and the cipher outputs one bit per cycle or one bit per 2 cycles when authentication is used. In hardware, the smaller, 32-bit tag, version requires about 2769.5 GE. Without authentication, the cipher requires 2145.5 GE, while the original Grain-128 requires 2133 GE under the same conditions.

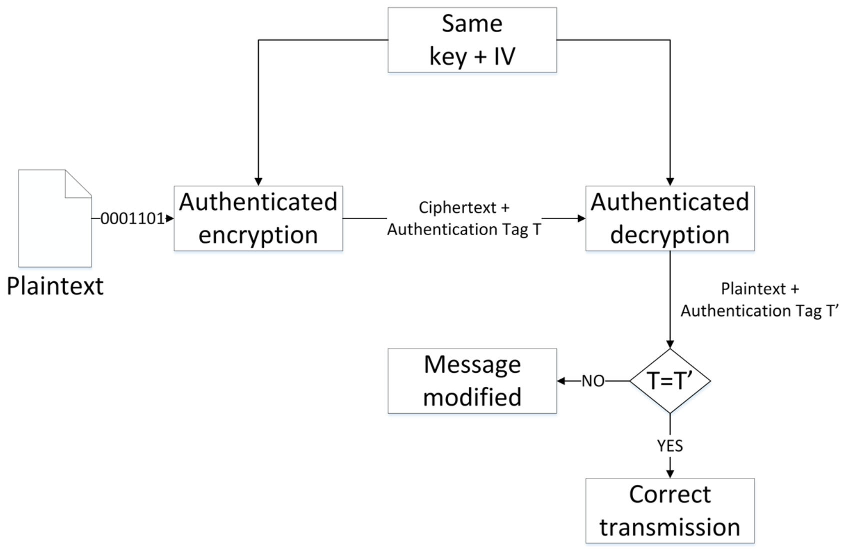

Figure 6. Authenticated encryption scheme.

Other than the aforementioned attack to the Grain family ciphers [87], a differential fault attack is also feasible, as presented in [108]. Said attack recovers the key of Grain-128a by observing the correct and faulty MACs of specific chosen messages. The attack exploits certain properties of the Boolean functions and the relevant choices of taps from the LFSR. The attack requires less than 211 fault injections and invocations of less than 212 MAC generation routines.

Rabbit-MAC [109] is a newly proposed scheme for authenticated encryption based on the stream cipher Rabbit. It is designed for WSNs and provides authenticity, integrity, and confidentiality at the link layer. Thus, except from end-to-end security between two end nodes of the WSN, Rabbit-MAC also achieves peer-to-peer security among the intermediate nodes of a communication path. The scheme assumes that the symmetric keys have been pre-distributed by a public-key cryptosystem. The IVs are included in the packet header to reduce the communication overhead introduced from exchanging secure IVs; moreover, these are unique for each different message to enhance security. The MAC component utilizes the next-state function of the cipher to preserve the implementation requirements; it follows the Encrypt-then-MAC approach [110] and its value is computed over the encrypted data and the packet header. The scheme fulfils the security requirements and is energy efficient. No independent cryptanalysis results are presented. However, we consider that the scheme is affected by the aforementioned practical fault analysis attacks that is performed in the Rabbit cipher [78].

ACORN [111] is a newly proposed stream cipher for lightweight authenticated encryption, and one of the ciphers submitted in CAESAR. ACORN uses a 128-bit key and IV and produces an authentication tag of a maximum of 128 bits. The cipher takes 1792 steps to initialize, while the state's size is 293 bits and consists of six concatenated LFSRs. There are three functions for generating the keystream bit from the state, computing the overall feedback bit, and updating the state, respectively. The tag is generated after processing all plaintext bits. The decryption follows the same approach. In order to counter known-plaintext or chosen plaintext attacks, the ciphertext and the tag are not given as output when the verification fails.

ACORN is bit-wise and is efficient in both hardware and software. The hardware cost is slightly higher than Trivium. The software implementation is fast, and moreover, 32 steps can be computed simultaneously, and the cipher can be parallelized. In comparison with AES-GCM, ACORN is more hardware efficient. In software, it produces smaller code size and is more efficient in general computing devices without AES-NI [112] capability.

Automated analysis on CAESAR candidates indicates that ACORN represents significantly elevated adaptive chosen plaintext attack risks [113], but no attack is presented so far.

Sablier [34] is another CAESAR candidate. It is a hardware-oriented stream cipher with built-in authentication. The latter does not decelerate encryption because of a carefully designed leak extraction strategy that is applied on the internal state. The cipher uses 80-bit keys and IVs. It takes 64 rounds to initialize, and produced tag is 32 bits. A new internal structure is proposed to produce the keystream, consisting of only bitwise XOR, bitwise logical AND, and bitwise intra-word rotations. Sablier's constrained device hardware implementation is 16 times faster than Trivium and requires 1925 GE. A practical state recovery attack for Sablier v1 is presented in [114].

ASC-1 [115] is an authenticated encryption stream cipher that utilizes AES with 128-bit key as a building block. In more detail, it uses 128-bit keys to encrypt three 128-bit blocks of plaintext, using a 56-bit IV for each block. Thus, the plaintext is processed in a Cipher Feedback (CFB)-like mode. However, this encryption strategy is also its main drawback as it only encrypts messages that consist of three 128-bit blocks. Also, its designers consider that the cipher is safe when the security problem is bounded to the case of distinguishing if the round keys are uniformly random or are generated by a key scheduling algorithm.

ALE [116] is another AES-based lightweight authenticated encryption scheme. It is also inspired by ASC-1 and the stream cipher LEX [117] (an eStream candidate based on AES). ALE uses 128-bit keys and IVs and encrypts plaintext of up to 245 bytes. It utilizes AES in order to benefit from the high security of the cipher and the performance of the AES-NI assembly instruction set. However, its security has already been compromised by cryptanalysts [118,119].

5. MAIN RESULTS

5.1. Discussion

Table II summarizes the features of each cipher and the best cryptanalysis results. The ciphers are referred in chronological order.

RC4, A5/1, E0, Trivium, Rabbit, SOSEMANUK, MICKEY 2.0, BEAN, WG-7, A2U2, ALE, Sablier, and the Grain and Hummingbird cipher families are vulnerable to attacks or not recommended for use in new applications and, thus, should be avoided. ASC-1 security is bounded as it requires a mechanism to ensure that round keys are uniformly random. Concerning the newer cipher proposals, that is, Quavium, CAvium, CAR30, and TinyStream, further independent cryptanalysis must be conducted to verify their security properties before they can be endorsed.

Advanced Encryption Standard counter is the typical and widely-accepted choice for stream encryption. It is efficient on many embedded devices, and where applicable, it should be preferred. It is used as a benchmark for stream encryption and its performance is the aim of new proposals.

However, AES-CTR is not always an option, especially in constrained devices (e.g. low-power embedded systems and RFIDs) and in applications where high throughput is required. Enocoro can be applied in such cases. The cipher is designed and supported by a large industrial entity

Table II. The general features of the examined ciphers.

Cipher	Year	Key size (bits)	Block size (bits)	IV	Type	Analysis / Attacks
RC4 [32]	1987	8-2048	1	—	ARX	Correlated-key attacks [54,55], attacks on WEP [56–58], biased outputs of keystream [53]
A5/1 [52]	1989	54, 64	—	0	LFSR	Time-memory tradeoff and known-plaintext attacks [59–62], attacks on GSM [63–65]
E0 [52]	1998	128	—	0	SHR	Cryptanalysis [66,67], attacks on Bluetooth [68,69]
AES [21]	1998	128, 192, 256	128	0	SPN	Biclique cryptanalysis [71]
Rabbit [41]	2003	128	128	—	Chaotic tables + simple arithmetic	Practical fault analysis [78]
Grain [44,85]	2004	80, 128	1, 16	64, 96	LFSR + NFSR	Differential fault attack [87]
Trivium [72]	2004	80	1, 8, 16	80	3 SHR	Improved differential fault attack [89]
Salsa [33]	2004	128, 256	32, 512	64, 128	ARX	Attacks on reduced versions [74]
HC [35,36]	2004	128, 256	—	128, 256	2 Large tables	Cryptanalysis [79,80]
SOSEMANUK [27]	2004	128,256	640, 32	64	LFSR + FSM	Improved differential fault analysis [83]
MICKEY 2.0 [45,93]	2006	80, 128	1	0-80, 0-128	Galois LFSR + NFSR	Improved differential fault attack [91], related key attacks [92]
Enocoro [48,49]	2008	80, 128	1	64	PRNG	Cryptanalysis [94]
Rabbit-MAC [109]	2008	128	128	—	Chaotic tables + simple arithmetic	Attacks on Rabbit
BEAN [31]	2009	80	2	64	FCSR+S-box	Distinguisher and state-recovery attacks [95]
Hummingbird [50]	2010	256	16	64	Hybrid	Several attacks [104]
WG-7 [28]	2010	80	1	81	LFSR + WG	Distinguishing attack [99]
TinyStream [43]	2010	128	—	—	TPM	-
Hummingbird-2 [51]	2011	128	16	64	Hybrid	Related-key attack [106]
Grain-128a [107]	2011	128	1	96	LFSR + NFSR	Differential fault attacks [87,108]
A2U2 [46]	2011	56	1	—	LFSR+ 2 NFSR	Ultra-efficient chosen-plaintext attack [98]
Quavium [96]	2012	80	1	80	4 Trivium-like SHR	-
CAvium [38]	2012	80	1	80	CA	-
ASC-1 [115]	2012	128	128	56	SPN (CFB-like mode)	Bounded security [115]
WG-8 [29]	2013	80	1, 11	80	LFSR + WG	Key recovery attack [100]
CAR30 [39]	2013	128	128	120	CA	-
ALE [116]	2013	128	128	128	SPN	Compromised by cryptanalysis [118,119]
ACORN [111]	2014	128	—	128	6 LFSR	Adaptive chosen-plaintext attack risk [113]
Sablier [34]	2014	80	—	80	ARX	Practical state recovery attack [114]

in embedded electronics and is a standardized cipher for LWC. It is efficient in hardware and performs reasonably well in software. Enocoro is optimized for low power consumption and can be used in compact control equipment and low-cost sensors. The encryption process consumes 1/10th the amount of power of AES and is faster than AES-CTR. For RFID tags, WG-8 is another attractive solution, as it is fast and consumes low energy.

Cryptographic libraries suitable for embedded systems, like Crypto++ and CyaSSL, adopt the eSTREAM Profile 1 finalists Salsa20 and HC. Both of them were designed to outclass the AES-CTR in software. The main advantage of these ciphers is the fast encryption stage. Salsa20 is fast and has a short initialization phase, which makes it suitable for Internet protocols and applications where a large number of packets are processed. On the other hand, HC can be parallelized, and is thus appropriate for parallel computing applications and superscalar microprocessors [120]. The cipher excels in domains where large streams must be processed.

For authenticated encryption, ACORN is an attractive choice, as it surpasses AES-GCM in embedded hardware and software. It can be parallelized, complying with the latest norm of parallel computing, and, as AES, supports the AES-NI for higher efficiency. As a candidate in the CEASAR competition, it has undergone detailed cryptanalysis, enhancing its acceptability status. For WSNs and RFID tags, WG-8 is the best choice as it fortifies security and improves energy efficiency. It is specifically designed with wireless sensor and RFID applications in mind. It has short initialization phase and fast decryption/encryption process making it suitable for encrypting many short messages, as is usually the case in these domains.

Self-synchronizing ciphers seems to be of no interest to the industry, and there is limited effort in designing such ciphers. Secure synchronous ciphers constitute the main candidates for embedded systems. The designing of the non-linearity part is an important factor along with efficient sophisticated functions for computing the algebraic immunity of the cipher for a large number of inputs.

In authenticated encryption schemes the production of a robust authentication tag is also a core target. Other than the discrete attacks on the cipher or the tag themselves, attention must also be paid in types of attacks that exploit vulnerabilities in both primitives to disclose data (e.g., [108]). Several schemes that encrypt the message and produce the MAC simultaneously are vulnerable to attacks (e.g., [50,107,116]). To this end, the encrypt-then-MAC approach (where the message is encrypted first and then the authentication tag is produced) is widely adopted (e.g., [109,111]).

Nevertheless, it should be pointed out that choosing a secure cipher does not guarantee the security of the underlying application. As practice has shown, the design of the protocol itself is equally important. Potential oversights in protocol design can compromise the security of the implementation, a characteristic case of this being the WEP protocol.

5.2. Performance evaluation

Providing a fair comparative analysis of the presented ciphers is not a trivial task, as they are implemented in different platforms and the measured parameters are not correlated in a straightforward manner. However, the authors held a constrained benchmark analysis in hardware and software, focusing on the secure ciphers identified above (AES, Enocoro, WG-8, Salsa20, and HC). The comparison contributes to the classification and the conclusions mentioned previously.

In general, the new ciphers are designed with AES-CTR performance in mind and are more efficient. However, the confidence on AES and its tested robustness is an obstacle new proposals have to overcome in order to be widely adopted.

5.2.1. Hardware.

The potential hardware solutions are limited, as presented in the previous sections. The authors implement the five ciphers in Verilog. The cipher designs are evaluated on a low-cost Xilinx Spartan3 XC3S1000 FPGA (7680 slices, 630 MHz, 55 KB RAM).

The metrics of throughput, latency, maximum frequency, power, and occupied flip-flops and slices are used to characterize each cipher. Throughput counts the megabytes per second that the cipher's encryption operation achieves. Latency is the number of clock cycles that are required to process a single block. Maximum frequency in megahertz is the frequency limit that is accomplished by an implementation. The Xilinx Power Estimator is utilized for the power consumption estimation in watts. Flip-flops constitute the amount of memory elements that are used. Slices represent the hardware area of the implementation, consisting of all the memory and computational components. The overall efficiency and performance/size tradeoff is estimated by the throughput/slices metric (the higher the value, the better).

Table III, summarizes the best FPGA implementations. The implementations were based on the latest embedded hardware designs for each cipher (AES [121], Enocoro-128 [49], WG-8 [29], Salsa20 [18], and HC-256 [18]). The authors implement the most efficient variants of each cipher, based on the throughput/slice metric.

The relevant features in ASIC are also summarized, based on the original implementation details. The metrics of throughput, GE, and figure of merit (FOM) are used to characterize each cipher in ASIC. The throughput metric is the same as in FPGA. GE corresponds to the number of logic gates that are required to implement the cipher and represents the complexity and occupied area (similar to the slices metric in FPGAs). FOM is calculated as throughtput/GE^2. It is aggregate metric for estimating the size and performance tradeoff (as the throughput/slices in FPGAs. The most efficient implementations in ASIC are presented by the FOM metric (the higher the value, the better). Table IV, summarizes the best ASIC implementations.

Table III. Hardware implementations of stream ciphers on FPGA ordered by the slices/throughput metric (the higher the value, the better).

Cipher	Key size (bits)	Block size (bits)	IV (bits)	Latency (cycles / block)	Throughput (MBps)	Max frequency (MHz)	Power (W)	FFs	Slices	Throughput / Slices
Better is				Lower	Higher		Lower	Lower	Lower	Higher
WG-8	80	1	80	1	2112	192	0.016	207	398	0.66
Enocoro-128	128	1	64	1	900	118	0.008	239	292	0.38
Enocoro-128	128	1	64	1	1200	149	0.008	362	442	0.33
AES	128	128	—	226	8754	77	0.078	781	5948	0.18
WG-8	80	11	80	1	190	190	0.005	85	137	0.17
Salsa20	256	32	64	2	990	19.4	0.012	1286	2036	0.06
HC-128	128	512	128	—	—	—	—	—	>> 262000	—

Table IV. Hardware implementations of stream ciphers on application-specific integrated circuit ordered by the figure of merit metric (the higher the value, the better).

Cipher	Key size (bits)	Block size (bits)	IV (bits)	Latency (cycles / block)	Throughput at 100 KHz (MBps)	Tech (μm)	Area (GE)	FOM
Better is				Lower	Higher		Lower	Higher
WG-8 [29]	80	11	80	1	1100	0.65	3942	0.00884
Enocoro-128 [49]	128	1	64	1	800	0.09	4100	0.00594
WG-8 [29]	80	1	80	1	100	0.65	1786	0.00391
Enocoro-80 [49]	80	1	80	1	—	0.09	2700	—
AES [121]	128	128	—	226	56.64	0.35	2400	0.00122
Salsa20 [18]	256	32	64	2	99	0.13	12126	0.00008
HC-256 [18]	256	—	256	—	—	0.13	>> 524000	—

Table V. Software implementations of stream ciphers on BeagleBone ordered by the combine metric (the lower, the better).

Cipher	Key size (bits)	Block size (bits)	IV (bits)	Initialization (cycles)	Encryption (cycles)	ROM (KB)	RAM (KB)	Throughput at 720 MHz (MBps)	CM
Better is				Lower	Lower	Lower	Lower	Higher	Lower
Salsa20	128	512	64	460	29491	4.8	8.27	12.5	276
Salsa20	256	512	64	460	29729	4.8	8.35	12.4	278
Enocoro-128	128	1	64	4870	138	3.8	7.56	5.2	526
WG-8	80	1	80	1379	69	6.6	0.59	10.4	456
HC-128	128	512	128	2082876	17388	77.2	16.58	21.2	2621
AES	128	128	—	6953	20480	22.2	88	4.5	3552
AES-CTR	128	128	128	469.6	21942	22.3	88.5	4.2	3822

WG-8, Enocoro and AES achieve lightweight implementations that are suitable for embedded systems (less than 3000 GE). WG-8 is the most compact and more efficient. Enocoro also performs well and is the only secure standardized stream cipher for LWC. AES has moderate performance but high power consumption. The secure eStream ciphers, Salsa20 and HC, produce large implementations and aren't appropriate for embedded hardware. The internal memory of HC128 alone requires around 524KB, resulting in about 262000 slices in FPGA or 524000 GE in ASIC.

5.2.2. Software.

Similarly, in software, the ciphers are implemented in C and evaluated on a low-cost credit-card-sized BeagleBone embedded device (AM3359 ARM Cortex A8 single core CPU, 720 MHz, 256 MB RAM, Ubuntu OS). All implementations are assessed over a common benchmark suite.

The metrics of throughput, ROM, RAM, and combine metric (CM) were used. Throughput is the same as in hardware. ROM and RAM measures the code and runtime memory requirements in KB. CM is calculated as $(ROM \times encryption \quad cyles)/block \quad size$.

The software implementations are based on the latest embedded software designs for each cipher (AES [122]/AES-CTR [49], Enocoro-128 [49] WG-8 [29], Salsa20 [16], HC-128 [122]). The best reported variants of each cipher are implemented, based on the CM metric. Table V, aggregates the most attractive software implementations.

Salsa20 achieves the best performance, being fast and with a short initialization phase. Enocoro and WG-8 also perform well with low code and memory requirements. HC is the fastest cipher, while AES is the slowest cipher.

6. FUTURE WORK

Key drivers in stream cipher evolution come from the competition with block ciphers and the intrinsic requirements of different application domains. Stream ciphers will remain in the foreground because of their high efficiency and ability to destroy statistical properties in natural language, in contrast to block ciphers. The evolution of the Internet-of-things implies the interconnection of a large number of embedded devices, usually with constrained computational capabilities, and their interaction with users [123,124]. Smart cites and social mobility applications are expected to include distributed structures to process and transmit high amounts of streams. Examples of industrial focus include but are not limited to telephony, urban surveillance with smart cameras, vehicular ad hoc networks, green networking, and ambient environment monitoring. Although stream ciphers seem the natural choice for streaming applications, ongoing research should enhance their usage over well-analyzed and reputable block ciphers. In hardware, LFSR and NFSR-based designs are more efficient. In software, table-driven or ciphers with chaotic and/or LFSR structures are appropriate.

Parallel computing architectures and models are widely used today and are often appropriate for embedded applications. Thus, secure stream ciphers with a higher degree of parallelism constitute a desirable goal.

7. CONCLUSION

Embedded devices permeate our lives, bringing us closer to the vision of ubiquitous computing. This significant change in the form, number and usage of computing devices, introduces new challenges in terms of the security of their resources, as well as the associated data that are stored or transmitted. This paper presented a survey of lightweight stream ciphers for embedded devices. Standardized and novel stream cipher designs were analyzed, also including authenticated encryption schemes. Moreover, the latest cryptanalysis results were presented, identifying vulnerable algorithms. At present, AES-CTR, Enocoro, Salsa20, HC, Acorn, and WG-8 are the only safe solutions. A benchmark analysis of these ciphers is performed on embedded hardware and software platforms. The above help highlight the most appropriate and secure stream-based cryptographic primitives for the different types of devices that may be found in the context of ubiquitous computing.

ACKNOWLEDGEMENTS

This work was funded by the General Secretarial Research and Technology (G.S.R.T.), Hellas under the Artemis JU research program nSHIELD (new embedded Systems architecture for multi-Layer Dependable solutions) project. Call: ARTEMIS-2010-1, Grand Agreement No: 269317.

REFERENCES

1. Manifavas C, Fysarakis K, Papanikolaou A, Papaefstathiou I. Embedded systems security: a survey of EU research efforts. *Security and Communication Networks* 2015; **8**(11): 2016–2036.
2. Fysarakis K, Hatzivasilis G, Rantos K, Papanikolaou A, Manifavas C. Embedded systems security challenges, *MeSeCCS 2014*, Lisbon, Portugal, 2014; 1–10.
3. Poschmann A. Lightweight cryptography cryptographic engineering for a pervasive world. *Doctoral Thesis*, IT-SECURITY series, No. 8, Faculty of Electrical Engineering and Information Technology, Ruhr-University, Bochum, Germany, 2009.
4. Zhang X, Heys HM, Li C. Energy efficiency of encryption schemes applied to wireless sensor networks. *Security and Communication Networks* 2015; **5**(7): 789–808.
5. Xiao Y, Chen HH, Du X, Guizani M. Stream-based cipher feedback mode in wireless error channel. *IEEE Transactions on Wireless Communications* 2009; **8**(2): 622–626.
6. Hatzivasilis G, Papaefstathiou I, Manifavas C, Askoxylakis I. Lightweight password hashing scheme for embedded systems, *9th IFIP WG 11.2 WISTP,* LNCS, vol. 8311, Springer, Heraklion, Crete, Greece, 2015; 260–270.
7. ECRYPT. *Stream Cipher Project, ECRYPT (2004–2008)*. (Available from: http://www.ecrypt.eu.org/stream) [Accessed 1 July 2015].
8. ISO/IEC 29192-3:2012. *International standard for lightweight cryptographic methods, ISO/IEC, 2012*. (Available from: http://www.iso.org/iso/home/store/catalogue_tc/catalogue_detail.htm?csnumber=56426) [Accessed 1 July 2015].
9. CAESAR. *CAESAR: Competition for Authentication Encryption: Security, Applicability, and Robustness, CAESAR, 2013*. (Available from: http://competitions.cr.yp.to/caesar.html) [Accessed 1 July 2015].
10. NIST. *Special Publication 800-38D, Recommendation for block cipher modes of operation Calois/Counter Modes (GCM) and GMAC, NIST, 2007*. (Available from: http://csrc.nist.gov/publications/nistpubs/800-38D/SP-800-38D.pdf) [Accessed 1 July 2015].
11. Sarma SE. *Towards the five-cent tag, MIT-AUTOID-WH-006*, Faculty of the Massachusetts Institute of Technology (M.I.T.), 2001.
12. Weis S. *Security and privacy in radio-frequency identification devices*, Faculty of the Massachusetts Institute of Technology (M.I.T.), 2003.
13. Manifavas C, Hatzivasilis G, Fysarakis K, Rantos K. Lightweight cryptography for embedded systems – a comparative analysis, *SETOP (2013),* LNCS, vol. 8247, Springer, Egham, UK, 2013; 333–349.
14. Hatzivasilis G, Gasparis E, Theodoridis A, Manifavas C. ULCL: An ultra-lightweight cryptographic library for embedded systems, *MeSeCCS 2014*, Lisbon, Portugal, 2014; 11–18.
15. Batina L, Lano J, Mentens N, Ors SB, Preneel B, Verbauwhede I. Energy, performance, area versus security trade-offs for stream ciphers, *The State of the Art of Stream Ciphers: Workshop Record*, Brugge, Belgium, 2004; 302–310.
16. Eisenbarth T, Paar C, Poschmann A, Kumar S, Uhsadel L. A survey of lightweight cryptography – implementations. *IEEE Design and Test of Computers* 2007; **24**(6): 522–533.
17. Fournel N, Minier M, Ubeda S. Survey and benchmark of stream ciphers for wireless sensor networks, *IFIP WISTP 2007,* LNCS, vol. 4462, Springer, Heraklion, Crete, Greece, 2007; 202–214.
18. Good T, Benaissa M. Hardware results for selected stream cipher candidates, *SASC 2007*, Bochum, Germany, 2007; 191–204.
19. Bokhari MU, Alam S, Massodi FS. Cryptanalysis techniques for stream cipher: a survey. *International Journal of Computer Applications* 2012; **60**(9): 29–33.

20. Banegas G. Attacks in stream ciphers: a survey. *International Association for Cryptologic Research* 2014, eprint, article 677.
21. Klein A. *Introduction to stream ciphers*, Stream Ciphers. Springer, 2013; pp. 1–13.
22. Bellovin SM, Miller F. Inventor of the one-time pad. *Cryptologia* 2011; **35**(3): 203–222.
23. Shannon CE. Communication theory of secrecy systems. *Bell System Technical Journal* 1949; **28**(4): 656–715.
24. Daemen J, Lano J, Preneel B. Chosen ciphertext attack on SSS. *eStream Report, Article 2005/044*, 2005.
25. KAasper E, Rijmen V, Bjorstad TE, Rechberger C, Robshaw M, Sekar G. Correlated keystreams in moustique, *AFRICACRYPT 2008,* LNCS, vol. 5023, Springer, Casablanca, Morocco, 2008; 246–257.
26. Canteaut A, Augot D, Biryukov, A, *et al*. Ongoing research areas in symmetric cryptography. *Technical Report D.STVL.4*, ECRYPT Information Society Technologies, 2006.
27. Berbain C, Billet O, Canteaut, A, *et al*. SOSEMANUK, a fast software-oriented stream cipher, *New Stream Cipher Designs,* LNCS, vol. 4986, Springer, 2008; 98–118.
28. Luo Y, Chai Q, Gong G, Lai X. A lightweight stream cipher WG-7 for RFID encryption and authentication, *IEEE Global Telecommunications Conference (GLOBECOM 2010)*, Miami, Florida, USA, 2010; 1–6.
29. Fan X, Mandal K, Gong G. WG-8 a lightweight stream cipher for resource-constrained smart devices, *9th International Conference QShine 2013,* LNCS, vol. 115, Springer, Greader Noida, India, January 11–12, 2013; 617–632.
30. Klapper A. A survey of feedback with carry shift registers, *SETA 2004,* LNCS, vol. 3486, Springer, Seoul, Korea, 2005; 56–71.
31. Kumar N, Ojha S, Jain K, Lal S. BEAN: a lightweight stream cipher, *2nd International Conference on Security of Information and Networks (SIN '09)*, Gazimagusa, North Cyprus, 2009; 168–171.
32. Paul G, Maitra S. *RC4 stream cipher and its variants*, Discrete Mathematics and Its Applications. CRC press, 2011.
33. Bernstein DJ. *The Salsa20 family of stream ciphers, CR.YP.TO*, 2007. (Available from: http://cr.yp.to/snuffle/salsafamily-20071225.pdf) [Accessed 1 July 2015].
34. Zhang B, Shi Z, Xu C, Yao Y, Li Z. *Sablier v1, CAESAR Competition*, 2014. (Available from: http://competitions.cr.yp.to/round1/sablierv1.pdf) [Accessed 1 July 2015].
35. Wu H. The stream cipher HC-128, *New Stream Cipher Designs The eSTREAM Finalists,* LNCS, vol. 4986, Springer, 2008; 39–47.
36. Wu H. A new stream cipher HC-256, *FSE 2004,* LNCS, vol. 3017, Springer, Delhi, India, 2004; 226–244.
37. Wolfram S. Cryptography with cellular automata, *Crypto-85,* LNCS, vol. 218, Springer, 1986; 429–432.
38. Sandip K, Debdeep M, Roy CD. Cavium strengthening trivium stream cipher using cellular automata. *Journal of Cellular Automata* 2012; **7**(2): 179–197.
39. Das S, Chowdhury RD. CAR30: a new scalable stream cipher with rule 30. *Cryptography and Communications* 2013; **5**(2): 137–162.
40. Pisarchik AN, Zanin M. Chaotic maps cryptography and security. In *Encryption: Methods, Software and Security*. Nova Science Publishers, 2010; 1–28.
41. Boesgaard M, Vesterager M, Pedersen T, Christiansen J, Scavenius O. Rabbit: a new high-performance stream cipher, *FSE 2003,* LNCS, vol. 2887, Springer, Lund, Sweden, 2003; 307–329.
42. Rosen-Zvi M, Ido Kanter EK, Kinzel W. Mutual learning in a tree parity machine and its application to cryptography. *Physical Review E* 2002; **66**(6): 066135–066148.
43. Chen T, Ge L, Wang X, Jiamei C. TinyStream: a lightweight and novel stream cipher scheme for wireless sensor networks, *CIS 2010*, Nanning, China, 2010; 528–532.
44. Hell M, Johansson T, Meier W. Grain a stream cipher for constrained environments. *International Journal of Wireless and Mobile Computing* 2007; **2**(1/2007): 86–93.
45. Babbage S, Dodd M. *The Stream Cipher MICKEY 2.0, eStream Project*, 2006. (Available from: http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey_p3.pdf) [Accessed 1 July 2015].
46. David M, Ranasinghe DC, Larsen T. A2U2: a stream cipher for printed electronics RFID tags, *IEEE International Conference on RFID*, Orlando, Florida, USA, 2011; 176–183.
47. Gong G, Youssef A. Cryptographic properties of the Welch-Gong transformation sequence generators. *IEEE Transactions on Information Theory* 2002; **48**(11): 2837–2846.
48. Watanabe D, Ideguchi K, Kitahara J, Muto K, Furuichi H. Enocoro-80: a hardware oriented stream cipher. *Third International Conference on Availability, Reliability and Security (ARES 08)* 2008; **1294**(1300): 4–7.
49. Systems Development Laboratory, Hitachi. *Enocoro-128v2: A Hardware Oriented Stream Cipher, Hitachi Ltd.*, 2009. (Available from: http://www.hitachi.com/rd/yrl/crypto/enocoro/enocoro_spec_20100222.zip) [Accessed 1 July 2015].
50. Engels D, Fan X, Gong G, Hu H, Smith EM. Hummingbird: ultra-lightweight cryptography for resource-constrained devices, *Financial Cryptography and Data*

Security - FC 2010, LNCS, vol. 6054, Springer, Tenerife, Canary Islands, Spain, 2010; 3–18.

51. Engels D, Saarinen MJO, Schweitzer P, Smith EM. The hummingbird-2 lightweight authenticated encryption algorithm, *RFID Sec 2011*, Amherst, Massachusetts, USA, 2011; 19–31.
52. Galanis MD, Kitsos P, Kostopoulos G, Sklavos N, Koufopavlou O, Goutis CE. Comparison of the hardware architecture and FPGA implementations of stream ciphers, *11th IEEE International Conference on Electronis, Circuitsand Systems (ICECS 2004)*, IEEE, Tel Aviv, Israel, 2004; 571–574.
53. Paul G, Rathi S, Maitra S. On non-negligible bias of the first output byte of RC4 towards the first three bytes of the secret key. *Designs, Codes and Cryptography* 2008; **49**(1–3): 123–134.
54. Paul G, Maitra S. Permutation after RC4 key scheduling reveals the secret key, *Selected Areas in Cryptography (SAC),* LNCS, vol. 4876, Springer, Ottawa, Canada, 2007; 360–377.
55. Akgun M, Kavak P, Demirci H. New results on the key scheduling algorithm of RC4, *INDOCRYPT,* LNCS, vol. 63625, Springer, Kharagpug, India, 2008; 40–52.
56. Klein A. Attacks on the RC4 stream cipher. *Designs, Codes and Cryptography* 2008; **48**(3): 269–286.
57. Erik T, Weinmann RP, Pyshkin A. Breaking 104 bit WEP in less than 60 seconds, *Information Security Applications,* LNCS, vol. 4867, Springer, Jeru Island, Korea, 2007; 188–202.
58. AlFardan NJ, Bernstein DJ, Paterson KG, Poettering B, Schuldt JC. On the security of RC4 in TLS, *USENIX Security*, Washington DC, USA, 2013; 305–320.
59. Biryukov A, Shamir A, Wagner D. Real time cryptanalysis of A5/1 on a PC, *Fast Software Encryption (FSE),* LNCS, vol. 1978, Springer, New York, USA, 2001; 1–18.
60. Ekdahl P, Johansson T. Another attack on A5/1. *IEEE Transactions on Information Theory* 2003; **49**(1): 284–289.
61. Barkan E, Biham E. Conditional estimators: an effective attack on A5/1, *Selected Areas in Cryptography,* LNCS, vol. 3897, Springer, Kingston, Canada, 2006; 1–19.
62. Shah J, Mahalanobis A. A new guess-and-determine attack on the A5/1 stream cipher. *Report 2012/208*, Cryptology ePrint Archive, IACR, 2012.
63. Barkan E, Biham E, Keller N. Instant ciphertext-only cryptanalysis of GSM encrypted communication, *Advances in Cryptology - CRYPTO 2003,* LNCS, vol. 2729, Springer, Santa Barbara, California, USA, 2003; 600–616.
64. Guneysu T, Kasper T, Novotny M, Paar C, Rupp A. Cryptanalysis with COPACOBANA. *IEEE Transactions on Computers* 2008; **57**(11): 1498–1513.
65. Nohl K, Paget C. GSM: SRSLY, *26th Chaos Communication Congress (26C3)*, Berlin, Germany, 2009; 21–49.
66. Hermelin M, Nyberg K. Correlation properties of the Bluetooth combiner, *Information Security and Cryptology ICISC'99,* LNCS, vol. 1787, Springer, Seoul, Korea, 2000; 17–29.
67. Fluhrer SR. *Improved key recovery of level 1 of the Bluetooth encryption system*, Foundations of Cryptography Basic Tools. Cambridge University Press: Cambridge, UK, 2002.
68. Lu Y, Vaudenay S. Faster correlation attack on Bluetooth keystream generator E0, *Advances in Cryptology CRYPTO 2004,* LNCS, vol. 3152, Springer, Santa Barbara, California, USA, 2004; 407–425.
69. Lu Y, Meier W, Vaudenay S. The conditional correlation attack: a practical attack on bluetooth encryption, *Advances in Cryptology CRYPTO 2005,* LNCS, vol. 3621, Springer, Santa Barbara, California, USA, 2005; 97117.
70. NIST Special Publication 800-38A. *Recommendation for block cipher modes of operation methods and techniques, NIST*, 2001. (Available from: http://csrc.nist.gov/publications/nistpubs/800-38a/sp800-38a.pdf) [Accessed 1 July 2015].
71. Bogdanov A, Khovratovich D, Rechbergerm C. Biclique Cryptanalysis of the full AES, *ASIACRYPT 2011,* LNCS, vol. 7073, Springer, Seoul, Korea, 2011; 344–371.
72. De Canniere C, Prenel B. *Trivium Specifications, eStream Project*, 2008. (Available from: http://www.ecrypt.eu.org/stream/p3ciphers/trivium/trivium_p3.pdf) [Accessed 1 July 2015].
73. Dai W. *Crypto++ Library 5.6.2, Crypto++*, 2013. (Available from: http://www.cryptopp.com/) [Accessed 1 July 2015].
74. Subhamoy M, Goutam P, Willi M. Salsa20 cryptanalysis: new moves and revisiting old styles, *9th International Workshop on Coding and Cryptography (WCC) 2015*, Paris, France, 2015; 1–10.
75. wolfSSL. *CyaSSL Embedded SSL Library, wolfSSL Inc.*, 2014. (Available from: http://www.yassl.com/yaSSL/Products-cyassl.html) [Accessed 1 July 2015].
76. ISO/IEC 18033-4:2011. *International standard for IT Security techniques, ISO/IEC*, 2011. (Available from: http://webstore.iec.ch/preview/info_isoiec18033-4%7Bed2.0%7Den.pdf) [Accessed 1 July 2015].
77. Boesgaard M, Vesterager M, Christiansen J, Zenner E. *The Stream Cipher Rabbit 1, eStream Project*, 2007. (Available from: http://www.ecrypt.eu.org/stream/p3ciphers/rabbit/rabbit_p3.pdf) [Accessed 1 July 2015].

78. Kircanski A, Youssef AM. Differential fault analysis of Rabbit, *Selected Areas in Cryptography,* LNCS, vol. 5867, Springer, Calgary, Alberta, Canada, 2009; 197–214.
79. Kircanski A, Youssef A M. Differential fault analysis of HC-128, *AFRICACRYPT 2010,* LNCS, vol. 6055, Springer, Stellenbosch, South Africa, 2010; 261–278.
80. Stankovski P, Ruj S, Hell M, Johansson T. Improved distinguishers for HC-128. *Design, Codes and Cryptography* 2012; **63**(2): 225–240, Springer.
81. Ekdahl P, Johansson T. A new version of the stream cipher snow, *SAC 2002,* LNCS, vol. 2595, Springer, Newfoundland, Canada, 2003; 47–61.
82. Anderson R, Biham E, Knudsen L. *Serpent: A Proposal for the Advanced Encryption Standard, AES contest*, 1998. (Available from: http://www.cl.cam.ac.uk/~rja14/Papers/serpent.pdf) [Accessed 1 July 2015].
83. Ma Z, Gu D. Improved differential fault analysis of SOSEMANUK, *8th International Conference on Computational Intelligence and Security (CIS)*, IEEE, Guangzhou, China, 2012; 487–491.
84. Otte D. *AVR-Crypto-Lib, AVR-Crypto-Lib*, 2009. (Available from: http://www.das-labor.org/wiki/AVR-Crypto-Lib/en) [Accessed 1 July 2015].
85. Hell M, Johansson T, Maximov A. A stream cipher proposal, Grain-128, *IEEE International Symposium on Information Theory*, Seattle, WA, 2006; 1614–1618.
86. Shamir A. SQUASH a new mac with provable security properties for highly constrained devices such as RFID tags, *FSE 2008,* LNCS, vol. 5086, Springer, Lausanne, Switzerland, 2008; 144–157.
87. Sarkar S, Banik S, Maitra S. Differential Fault Attack against Grain family with very few faults and minimal assumptions. *Report 2013/494*, Cryptology ePrint Archive, IACR, 2013.
88. Mentens N, Genoe J, Preneel B, Verbauwhede I. A low-cost implementation of trivium, *SASC 2008*, Lausanne, Switzerland, 2008; 197–204.
89. Mohamed MSE, Bulygin S, Buchmann J. Improved differential fault analysis of trivium, *COSADE 2011*, Darmstadt, Germany, 2011; 147–158.
90. Good T, Benaissa M. Hardware performance of estream phase-III stream cipher candidates, *SASC 2008*, Lausanne, Switzerland, 2008; 163–174.
91. Banik S, Maitra S, Sarkar S. Improved differential fault attack on MICKEY 2.0. *Report 2013/029*, Cryptology ePrint Archive, IACR, 2013.
92. Ding L, Guan J. Cryptanalysis of MICKEY family of stream ciphers. *Security and Communication Networks* 2013; **6**(8): 936–941.
93. Babbage S, Dodd M. *The Stream Cipher MICKEY-128 2.0, eStream Project*, 2006. (Available from: http://www.ecrypt.eu.org/stream/p2ciphers/mickey128/mickey128_p2.pdf) [Accessed 1 July 2015].
94. Hell M, Johansson T. Security evaluation of stream cipher Enocoro-128v2. *CRYPTEC Technical Report, No. 2008*, 2010.
95. Agren M. On some symmetric lightweight cryptographic designs. *Doctoral Thesis*, Department of Electrical and Information Technology, Faculty of Engineering, LTH, Lund University, 2012.
96. Tian Y, Chen G, Li J. Quavium - a new stream cipher inspired by trivium. *Journal of Computers* 2012; **7**(5): 1278–1283.
97. De Canniere C, Dunkelman O, Knezevic M. KATAN & KTANTAN a family of small and efficient hardware-oriented block ciphers, *CHES 2009,* LNCS, vol. 5747, Springer, Lausanne, Switzerland, 2009; 272–288.
98. Chai Q, Fan X, Gong G. An ultra-efficient key recovery attack on the lightweight stream cipher A2U2, Cryptology ePrint Archive, IACR, 2011/247, 2011.
99. Orumiehchiha MA, Pieprzyk J, Steinfeld R. Cryptanalysis of WG-7: a lightweight stream cipher. *Cryptography and Communications* 2012; **4**(3–4): 277–285.
100. Ding L, Jin C, Guan J, Wang Q. Cryptanalysis of lightweight WG-8 stream cipher. *IEEE Transactions on Information Forensics and Security* 2014; **9**(4): 645–652.
101. NIST Special Publication 800-22 A. *Statistical test suit for random and pseudorandom number generators for cryptographic applications, NIST*, 2010. (Available from: http://csrc.nist.gov/publications/nistpubs/800-22-rev1a/SP800-22rev1a.pdf) [Accessed 1 July 2015].
102. Karlof C, Sastry N, Wagner D. TinySec: a link layer security architecture for wireless sensor networks, *SenSys '04*, Baltimore, Maryland, USA, 2004; 162–175.
103. Walker J. *ENT - A Pseudorandom Number Sequence Test Program, ENT*, 2008. (Available from: http://www.fourmilab.ch/random/) [Accessed 1 July 2015].
104. Saarinen MJO. Cryptanalysis of Hummingbird-1, *FSE 2011,* LNCS, vol. 6733, Springer, Lyngby, Denmark, 2011; 328–341.
105. ISO/IEC 18000-6. *International standard for parameters for air interface communications at 860 MHz to 960 MHz, ISO/IEC*, 2013. (Available from: http://www.iso.org/iso/home/store/catalogue_ics/catalogue_detail_ics.htm?csnumber=59644) [Accessed 1 July 2015].
106. Saarinen MJO. Related-key attacks against full Hummingbird-2, Cryptology ePrint Archive, IACR, 2013. 2013/070.

107. Agren M, Hell M, Johanson T, Meier W. A new version of grain-128 with optional authentication. *International Journal of Wireless and Mobile Computing* 2011; **5**(1): 48–59.
108. Banik S, Maitra S, Sarkar S. A Differential fault attack on grain-128a using MACs, *2nd International Conference SPACE 2012,* LNCS, Springer, Chennai, India, 2012; 111–125.
109. Tahir R, Javed Y, Cheema AR. Rabbit-MAC: lightweight authenticated encryption in wireless sensor networks, *IEEE International Conference on Information and Automation*, Zhangjiajie, China, 2008; 573–577.
110. Bellare M, Namprempre C. Authenticated encryption: relations among notions and analysis of the generic composition paradigm, *ASIACRYPT 2000,* LNCS, vol. 1976, Springer, Kyoto, Japan, 2000; 531–545.
111. Wu H. *ACORN: A Lightweight Authenticated Cipher, CAESAR competition*, 2014. (Available from: http://competitions.cr.yp.to/round1/acornv1.pdf) [Accessed 1 July 2015].
112. Intel Software Network Rev 3.01. *AES Instructions Set, Intel*, 2008. (Available from: https://software.intel.com/sites/default/files/article/165683/aes-wp-2012-09-22-v01.pdf) [Accessed 1 July 2015].
113. Saarinen MJO. BRUTUS: identifying cryptanalytic weaknesses in CAESAR first round candidates. *Report 2014/850*, Cryptology ePrint Archive, IACR, 2014.
114. Feng X, Zhang F. A practical state recovery attack on the stream cipher Sablier, *8th International Conference on Network and System Security (NSS 2014),* LNCS, Springer, Xi'an, China, 2014; 198–208.
115. Jakimoski G, Khajuria S. ASC-1: an authenticated encryption stream cipher, *Selected Areas in Cryptography 2012,* LNCS, vol. 7118, Springer, Toronto, Canada, 2012; 356–372.
116. Bogdanov A, Mendel F, Regazzoni F, Rijmen V, Tischhauser E. ALE: AES-based lightweight authenticated encryption, *FSE'13,* LNCS, Springer, Singapore, 2013.
117. Birykov A. *A New 128-bit Key Stream Cipher LEX, eStream Project*, 2005. (Available from: http://www.ecrypt.eu.org/stream/ciphers/lex/lex.pdf) [Accessed 1 July 2015].
118. Khovratovich D, Rechberger C. The LOCAL attack: cryptanalysis of the authenticated encryption scheme ALE. *Report 2013/357*, Cryptology ePrint Archive, IACR, 2013.
119. Wu S, Wu H, Huang T, Wang M, Wu W. Leaked-state-forgery attack against the authenticated encryption algorithm ALE, *ASIACRYPT 2013,* LNCS, vol. 8269, Springer, Bengaluru, India, 2013; 377–404.
120. Ayesha K, Deblin B, Goutam P, Anupam C. Optimized GPU implementation and performance analysis of HC series of stream ciphers, *Information Security and Cryptology (ICISC 2012),* LNCS, vol. 7839, Springer, Seoul, Korea, 2013; 293–308.
121. Moradi A, Poschmann A, Ling S, Paar C, Wang H. Pushing the limits: a very compact and a threshold implementation of AES, *Advances in Cryptology - EUROCRYPT 2011,* LNCS, vol. 6632, Springer, Tallinn, Estonia, 2011; 69–88.
122. Meiser G, Eisenbarth T, Lemke-Rust K, Paar C. Software implementation of eSTREAM profile I ciphers on embedded 8-bit AVR microcontrollers, *Workshop Record State of the Art of Stream Ciphers (SASC 07)*, 2007.
123. Fysarakis K, Hatzivasilis G, Papaefstathiou I, Manifavas C. RtVMF – a secure real-time vehicle management framework with critical incident response. *IEEE Pervasive Computing Magazine (PVC) – Special Issue on Smart Vehicle Spaces* 2016.
124. Fysarakis K, Hatzivasilis G, Askoxylakis IG, Manifavas C. RT-SPDM: real-time security, *HCI International 2015,* LNCS, vol. 9190, Springer, Los Angeles, CA, USA, 2015; 1–12.