

An exhaustive review of the stream ciphers and their performance analysis

Raghavendra Ananth¹, Narayana Swamy Ramaiah²

¹Department of Electronics and Communications, JAIN Deemed to be University, Bangalore, India

²Department of Computer Science Engineering, JAIN Deemed to be University, Bangalore, India

Article Info

Article history:

Received Dec 14, 2022

Revised Sep 17, 2023

Accepted Sep 25, 2023

Keywords:

Field programmable gate array

Internet of things

Lightweight

Security

Stream ciphers

ABSTRACT

The number of internet of things (IoT) applications has increased, which has increased the demand for low-resource gadgets. The data produced by these devices must be protected to guarantee security. The devices operate in conditions with limited space, computational power, memory, and energy. High-security standards are difficult to achieve with limited resources. The detailed analysis of various stream ciphers and their performance metrics is reviewed in this manuscript. The functionality of the stream ciphers is categorized and thoroughly discussed based on both the hardware and software viewpoints. The security attacks and their countermeasure methods using stream ciphers are discussed. The performance metrics of most hardware-based stream ciphers, including the ECRYPT stream cipher project (eSTREAM) ciphers, are discussed. Each hardware stream cipher design highlights the hardware constraints such as chip area, frequency, throughput, and hardware efficiency. The work also highlights the various applications using these stream ciphers. The current trends using these stream ciphers are discussed with futuristic goals.

This is an open access article under the [CC BY-SA](https://creativecommons.org/licenses/by-sa/4.0/) license.



Corresponding Author:

Raghavendra Ananth

Department of Electronics and Communications, JAIN Deemed to be University

Bangalore 560069, Karnataka, India

Email: araghavendra.research@gmail.com

1. INTRODUCTION

Widely used applications like big data, cloud computing, and e-commerce have resulted in a growing demand for efficiency and security in data processing. The cryptography core and information security create lots of opportunities with real-time challenges. Providing high-level security with high-speed architecture at a low-cost implementation while considering low-resource constraints became a prominent demand for most applications. Wireless networks, device authentication, and radio-frequency identification (RFID) systems have low-resource constraints with low-cost implementation requirements. The lightweight block and stream ciphers protect attackers' information and provide data integrity and confidentiality [1], [2]. Block ciphers are the primary choice in lightweight cryptography (LWC) and are easily designed with functionality. However, block ciphers use further as communication protocols, and they can't be designed using stream ciphers. The necessity of the initialization phase before the communications happen has significant drawbacks for the stream ciphers. The stream ciphers are suited to most application requirements where the input text is continuous or unknown. Stream ciphers are compact, easy to design, fast, less-power utilization, and suitable for low-constrained devices [3], [4].

Stream ciphers have received more attention in recent years due to various research initiatives to develop secure stream ciphers. Research activities and competitions have been conducted in past decades to find novel architectures. As an effort, ECRYPT stream cipher project (eSTREAM) completion is among them

and was held by the european network of excellence for cryptology (ENEC) from 2004 to 2008. This competition promotes to finds of compact and novel stream ciphers for a wide range of usage. Later, the international organization for standardization and the international electrotechnical commission (ISO/IEC) standardized stream ciphers formed for LWC in the ISO/IEC 29192-3:2012 standard. Many stream ciphers proposals and concepts have been proposed [5], [6]. Authentication is one of the prime security features to be considered in most applications, apart from confidentiality, and data integrity. Competition for authenticated encryption (AE): security, applicability, and robustness (CAESAR) conducts the cryptographic research community competition to find suitable cipher algorithms and should be advantaged over advanced encryption standards (AES) [7]. The hardware-based stream ciphers are well-suited to low-resource-constrained devices and use direct cryptographic functions and basic operations without additional components [8]. The stream ciphers have constructed software and hardware acceleration using cryptographic functions, feedback shift registers, and basic operations [9], [10]. The cryptographic functions are categorized into either Boolean or vectorial functions with different cryptographic properties. The shift registers like divided into the linear feedback shift register (LFSR) and non-linear feedback shift register (NFSR) based on feedback mechanisms. In addition, XOR and rotation operations commonly used essential functions while constructing the stream ciphers.

The performance characteristics of various stream ciphers are examined in this paper, both from a hardware and software perspective. The approach of the stream cipher is described in section 2, as well as an overview of its design with tabulation. In section 3, we'll go over security attacks and countermeasures. In section 4, the performance realization and its application usage are listed. The future trends of current stream ciphers are highlighted in section 5. Finally, the overall work in section 6 concludes.

2. STREAM CIPHERS

The stream ciphers are an alternative branch of the symmetric cryptosystem, which provides better speed and scalability for hardware-based approaches. The stream ciphers are classified based on functionality, represented in Figure 1. The LFSR based Stream ciphers are bit-oriented types. The key generation units are designed using a more significant number of LFSR units. An example of a combiner generator with non-linear features in E0 is represented in Figure 2.

The E0 is Bluetooth encryption that supports point-to-point communications in wireless networks. The E0 mainly contains four LFSRs with 4-bit memory. The memory bits are updated using C functions. The E0 uses a 128-bit key with a 74-bit initialization vector (IV). The keystream receives the composite output with a feedback mechanism. The E0 is used mainly in Bluetooth combiner with alternative mapping correlation analysis [11], [12]. With an irregular clock control mechanism, the clock controller generator introduces non-linear properties. In Figure 3, the clock controller generator is depicted. Two LFSR sets and a feedback controller are the key components. An example of a clock-controlled generator is A5/1 based stream cipher. This encryption technique is used in most global systems for mobile communications (GSM) based phones for air transmission encryption.

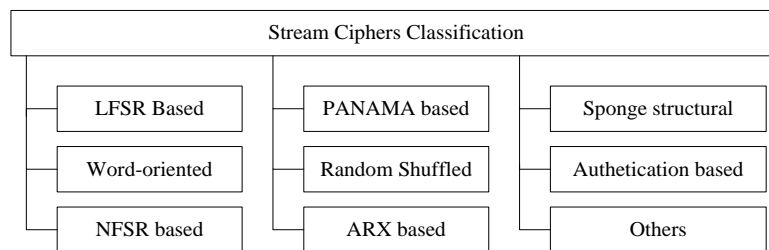


Figure 1. Classification of the stream ciphers

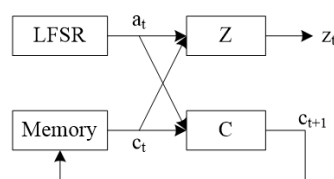


Figure 2. E0-based stream cipher

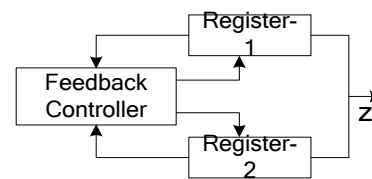


Figure 3. Clock controlled generator

The A1/5 cipher uses a 54-bit or 64-bit secret key for keystream generation and avoids the reduction of output efficiency [11]. Mutual irregular clocking keystream generator is also called MICKEY stream cipher, which provides low complexity and fewer resource constraints with high security on the hardware platform. The MICKEY cipher uses an irregular clocking mechanism of shift registers with an optimization mechanism against the attacks. The MICKEY cipher generally uses an 80-bit key, whereas the MICKEY 2.0 cipher uses a 128-bit secret key with an IV of 80/128-bit [13]–[15]. The MICKEY cipher uses two registers (R and S) with a feedback control mechanism to generate the keystream bit.

The word-orient stream ciphers work on 8-bit to 32-bit with LFSR with finite state machine (FSM) or non-linear filter generation combinations. SNOW series (1.0, 2.0, and 3.0). The SNOW 1.0 cipher uses a 128-bit secret key with a 32-bit word size. The SNOW-based stream cipher representation is shown in Figure 4. It contains two registers, finite field operation with a feedback mechanism, non-linear FSM with two memory units, and XOR operation as output to generate the running key [16], [17]. SNOW 2.0 cipher uses a 128/512-bit secret key with an IV of 128-bit. The ZUC is a stream cipher used as a 3rd generation partnership project (3GPP) encryption standard and developed for Chinese studies for inclusion in the 4th generation (4G) or long term evolution (LTE) project. The ZUC cipher uses a 128-bit secret key size with an IV of 128-bit, and it is built with LFSR-based architecture. The ZUC architecture mainly includes LFSR layers, a Bit recognition layer, and non-linear function and key loading. The ZUC cipher mainly focuses more on timing attacks [18]. The SOSEMANUK is one of the software-based eSTREAM projects, which uses a 128/256-bit secret key with an IV of 128-bit for a 32-bit word length. The stream cipher uses most of the features and working principles of SNOW 2.0 with SERPENT-based transformations. The efficiency and security analysis is improved than the SNOW 2.0 stream cipher [19].

The LFSR and NFSR combination is constructed using the GRAIN family to enhance the cryptographic properties. The GRAIN family targets hardware-based constrained environments to improve the gate count, memory, and power consumption features. The GRAIN family has three stream ciphers: GRAIN-v1, GRAIN-128, and GRAIN-128a. The GRAIN-v1 considers 80-key with 80-bit IV using NFSR and LFSR [20]. The GRAIN-128 considers a 128-bit key with IV of 96-bit [21]. The GRAIN stream cipher mainly has two shift registers, LFSR and NFSR, and output functions are represented in Figure 5. The key initialization mechanism is crucial for realizing the attack scenarios in the GRAIN cipher using IV and XOR operations. The small-state-based stream cipher is introduced with continuous key use to solve the hardware complexity, illustrated in Figure 6.

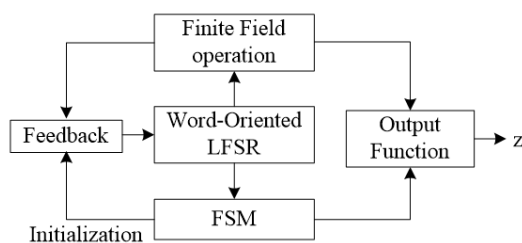


Figure 4. SNOW-based stream cipher

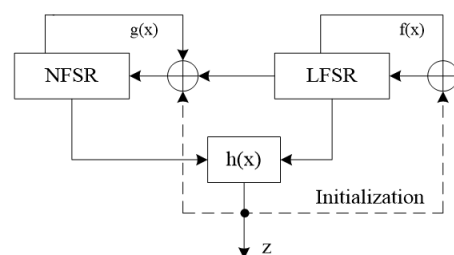


Figure 5. GRAIN-based stream cipher

The TRIVIUM series [22] stream ciphers are hardware featured with simple architecture and are interconnected with three NFSRs with low-degree feedback mechanisms, and quadratic filter functions are represented in Figure 7. The TriviA cipher [23] generates the keys for ciphertext and tags and provides independent hash pairs to calculate the tag. The "encode-hash combine" or ECH hash creates distinct hash pairs. The TriviA provides a 124-bit security key for authentication and a 128-bit key for privacy. The TRIVIUM is one of the eSTREAM finalist hardware stream ciphers and uses an 80-bit secret key size with an IV of 80-bit [24]. The TRIVIUM cipher can generate up to 264 keystream bits with a 288-bit internal state. The cipher can solve bit-oriented issues with strong security and performance efficiency. The hardware based fast and secured AE is introduced as TriviA, which uses a 128-bit secret key size with an IV of 80-bit. Fruit-2.0 is a stream cipher that is ultra-lightweight and has a more straightforward internal state system [25]. The Fruit 2.0 cipher has an 80-bit secret key and a 70-bit IV. Fruit 2.0 is used to strengthen against related-key attacks with a modified initialization process.

The Platelet stream cipher is well suited for lower-constraint devices and does not rely on non-volatile memory (NVM) [26]. The Platelet cipher improves the security weakness by storing the key in non-rewritable NVM and rewritable NVM. Platelet cipher uses a 128-bit secret key size with an IV of 40-bit. The Platelet uses

128-bit random output data in each iteration. The Rabbit examines the security for algebraic and correlation attacks by arranging the key/IV setup parameters. The MORUS is an authenticated stream cipher with 128/256 bits of secret keys and a 128-bit IV [37]. MORUS v1 uses the status update function to avoid collisions during the initialization and encryption/decryption stages.

The sponge structural-based stream ciphers are designed based on sponge structure with LFSR or permutations, and one of its internal state outputs is directly considered a keystream sequence. The KECCAK and ASCON are examples of sponge structural-based stream ciphers. The KECCAK is a sponge construction type cipher that uses more random permutations, allows multiple inputs, and provides any amount of data outputs [38]. The KECCAK cipher uses a 128-bit secret key without an IV process. The KECCAK cipher provides better authentication features without using any additional authentication module. The ASCON is one of the CAESAR finalists' ciphers and known AE modules [39]. The ASCON cipher uses a 128-bit secret key with an IV of 128-bit. The ASCON uses a substitution permutation network (SPN) structure with a fixed permutation of an iterative process. It performs both software and hardware implementations with better performance and cost. The ASCON is best known for cube and key recovery attacks.

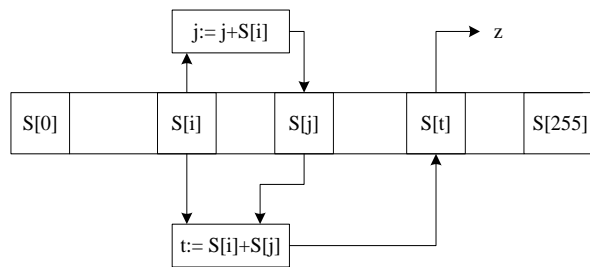


Figure 8. RC4-based keystream generation

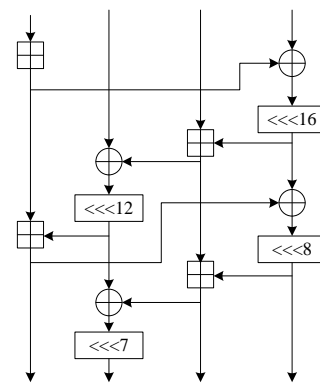


Figure 9. ARX-based round function for Chacha

The A2U2 is one of the AE ciphers commonly used in printed electronics-based RFID tags [40]. The A2U2 uses two NFSRs followed by a key-bit mixing mechanism with a shrinking filter to generate the ciphertext. A2U2 cipher uses a 56-bit secret key without an IV process. The Welch Gong (WG)-7 is a lightweight stream cipher used for RFID authentication and encryption [41]. WG-7 cipher uses an 80-bit secret key with an IV of 81-bit. The WG-7 consists of 23-stage LFSRs for keystream generation. The WG-7 is secure against time/data/memory trade-off attacks. The WG-8 is a lightweight stream cipher used for low resource constrained smart devices [42]. To generate the ciphertext, the WG-8 uses 20-stage LFSRs with feedback polynomial and transformation modules. The WG-8 cipher uses an 80-bit secret key with an IV of 80-bit. The WG-8 is capable of resisting the most common security attacks. The Hummingbird (HB) is an ultra-lightweight stream cipher commonly used in high-volume consumer devices like smart cards, RFID tags, and wireless devices [43]. HB cipher uses a 16-bit block size, 64/256-bit secret key with an IV of 64-bit. The HB encryption mainly contains four 16-bit block ciphers, followed by an internal state register update unit and a 16-bit LFSR module. The 16-bit block cipher is constructed using a typical substitution permutation (SP) network. The HB-2 is a lightweight authentication encryption module targeted at low-constrained devices [44]. HB 2.0 cipher uses a 128-bit secret key with an IV of 64-bit. Grain-128a is a new version of Grain-128 with authentication features [45]. Grain-128a cipher uses a 128-bit secret key with an IV of 96-bit. Grain-128a was used to strengthen all known attacks. The Rabbit-MAC is a lightweight AE module commonly used in wireless sensor networks (WSNs) [46]. The Rabbit-MAC cipher uses a 128-bit secret key without an IV process and generates the 128-bit random data at the output side for each iteration. The pseudo-random data is XOR'ed with plaintext/ciphertext to generate the encryption/decryption process in Rabbit-MAC. ACRON is a lightweight authenticated cipher and uses a 128-bit secret key with an IV of 128-bit [47]. The authentication tag length must be less than or equal to 128 bits. The six LFSRs are concatenated, followed by feedback bits in the ACRON structure. The ACRON is capable of resisting traditional and statistical attacks. The Sablier is one of the hardware-based stream ciphers built with authentication features [48]. The Sablier v1 cipher uses an 80-bit secret key with an IV of 80-bit. The Sablier performs the authentication mechanism using shift registers and accumulators in keystream generation.

The BEAN is a lightweight stream cipher module designed based on the GRAIN cipher [49]. The BEAN cipher uses two FCSRs followed by an S-Box and filtering. The BEAN cipher uses an 80-bit secret key with an IV of 64-bit. The BEAN cipher utilizes fewer hardware resources than the GRAIN cipher. The BEAN cipher can be resistant to most traditional attacks. The new scalable stream cipher with rule 30 is CAR30. The CAR30 cipher is constructed using the cellular automata (CA) rule 30 with maximum length CA followed by XOR operation to generate the ciphertext. The CAR30 is implemented both on software and hardware platform. In general, the CAR30 can scale up to any key size and IV. Most current works on CAR30 use a 128-bit secret key with an IV of 120-bit [50]. The CAR30 provides better throughput than other GRAIN and TRIVIUM ciphers. TinyStream is a new lightweight stream cipher algorithm for WSNs. TinyStream cipher uses a 128-bit secret key without an IV process [51]. The TinyStream cipher is constructed using tree parity machine (TPM) with a loop system mechanism. The summary of the stream cipher types and their algorithms is tabulated in Table 1. The list of the stream ciphers with functionality is tabulated in Table 2. The stream cipher type, secret key size, and IV size are mentioned in the ciphers tabulation.

Table 1. Types of the stream ciphers and their algorithms

Stream cipher type	Stream cipher algorithms
LFSR	A5/1 [11], E0 [11], [12], and MICKEY 2.0 [13]–[15]
Word-oriented	SNOW series [16], [17], ZUC Series [18], and Sosemanuk [19]
NFSR	GRAIN [20], [21], TRIVIUM [22], [24], Trivia [23], Fruit V2 [25], Platelet [26], Quivium [27], and Kreyvium [28]
PANAMA	Enocoro and MUGI [29]
Random shuffled	RC4 [30], RC4-A [32], HC-128 [32], and HC-256 [33]
ARX based	Salsa20 and Chacha [34], [35], Rabbit [34], [36], and MORUS [37]
Sponge Structural Authentication	KECCAK [38] and ASCON [39]
Other	A2U2 [40], WG7 [41], WG8 [42], HB-1 [43], HB-2 [44], GRAIN-128a [45], RabbitMAC [46], ACORN [47], and Sablier [48]
	Bean [49], CAR30 [50], and TinyStream [51]

Table 2. List of stream ciphers and their approaches

Stream ciphers	Key size	IV	Type	Stream ciphers	Key size	IV	Type
A5/1 [11]	54,64	0	LFSR	Chacha, Salsa-20 [35]	256	32	ARX
E0 [11], [12]	128	74	LFSR	Rabbit [36]	128	128	ARX
MICKEY [13], [15]	80/128	80/128	LFSR	MORUS V1. [37]	128/256	128	ARX
SNOW [16], [17]	128/512	128	LSFR+FSM	KECCAK [38]	128	NA	ARK+LFSR
ZUC [18]	128	128	LFSR+XOR.	ASCON [39]	128	128	ARK+SPN
SOSEMANUK [19]	128/256	128	LFSR+FSM	A2U2 [40]	56	NA	LFSR+2 NFSR
GRAIN [20], [21]	80/128	64/96	LFSR+NFSR	WG-7 [41]	80	81	LFSR+WG
TRIVIUM [22], [24]	80	80	Three SR	WG-8 [42]	80	80	LFSR+WG
Trivia [23]	128	80	Three SR	Hummingbird [43]	256	64	Hybrid
Fruit -V2 [25]	80	70	LFSR+NFSR	Hummingbird -2 [44]	128	64	Hybrid
Platelet [26]	128	40	LFSR+NFSR	GRAIN-128a [45]	128	96	LFSR+NFSR
QUAVIUM [27]	128	80	Three SR	Rabbit-MAC [46]	128	NA	Chaotic tables+XOR
Kreyvium [28]	128	80	Three SR	ACORN [47]	128	128	Six LFSR's
PANAMA [29]	256	NA	Hash+Stream	Sablier [48]	80	80	ARX
RC4 [30]	8 to 2048	NA	ARX	BEAN [49]	80	64	FCSR+S-Box
HC-128 [32]	128	128	Large tables	CAR30 [50]	128	120	CA
HC-256 [33]	256	256	Large tables	TinyStream [51]	128	NA	TPM
Rabbit [34]	128	64	Chaotic tables				

3. SECURITY ATTACKS AND COUNTERMEASURE METHODS

This section analyzes different types of security attacks and their countermeasure methods. The attacker's main aim is to use cipher designs to find the secret key used in the encryption or decryption process. Two attacks happen: passive attacks and active attacks. Passive attacks occur in the initialization or output phases. The attacker retrieves the information, copies them, and uses it for harmful or malicious purposes. Whereas active attacks, the attackers are trying to recreate the original data in the form of an insert, replay or delete. These two attacks will modify the key information, or system resources will be damaged.

Furthermore, these attacks are extensively classified based on cryptography usage. Exhaustive key search is an attack (brute force) where attackers try to find all the possible core combinations to find the primary secret key. This type of attack's computational complexity remains lower and possesses more on plaintext and ciphertexts. The exhaustive key search is analyzed in detail using the TRIVIUM cipher [22], [24] with key recovery. Correlation attacks realize the cipher's linear function and calculate the keystream based on output observation. Algebraic attacks use the algebraic equations of the main cipher and are used further to generate

the key bits. Similarly, linear attacks are also correlated with the linear functions of the defined keystream bits and initialization bits.

Distinguishing attacks are a type of attack in which attackers try to differentiate the keystream information from a random sequence feature. These attacks may recover the complete key details in the future. The side-channel attack is a type in which the attacker retrieves the data information from the cipher while calculating the power consumption or electromagnetic emission process. In this attack, the attacker hacks the complete information from the internal operations of the cipher technique. The related-key attack is a type of target attack happening during the re-initialization process of the cipher design operation. The attacker will generate the related keys only if the cipher technique does not use the non-linearity feature and is directly related to plain text and new-key generation. Similarly, the chosen-plain text or IV attacks use the key scheduling weakness and retrieve the useful initial state information from the main memory. The basic structure of the cipher realizes the time, memory-data trade-off attacks, and summarization of the related results in a larger table.

Divide and conquer attacks are attacks in which the cipher operation is divided into essential components. The very few bits of information are calculated in each state operation, and the most harmed components are attacked first in this conquer attack. The new type of attack that is applied to any cryptosystem is cube-attacks. The output bit is a function of the plain text bits and key bits. So, the attacker used to sum up the possible combination of the plaintext bits and detect linear equations with the help of key bits. Collision attacks or internal state collisions happen when the two keystream bits are generated from the same colliding state. The collision state finding requires the number of keystream bits using plaintext or IV. Guess and determine attack is a type of attack. The attacker will guess the few unknown variables of the stream cipher and determine or deduce the remaining unknown variables using guessed ones. The fault attack allows the conflict to introduce flipping faults into one of the LFSRs. It is more difficult to find or trace the fault in NFSR than in LFSR-based ciphers. The summary of the different types of attacks and countermeasure methods using stream ciphers is tabulated in Table 3. These stream cipher approaches are used as a countermeasure method with detailed analysis to recover the data or key from attackers.

Table 3. Summary of the different types of attacks and countermeasure methods using stream ciphers

Analysis/attacks	Countermeasure method using stream ciphers
Classical cryptanalysis, differential cryptanalysis	MICKEY [13], [15], GRAIN [20], [21], HC-256 [33], Rabbit [34], Salsa 20 [34], [35], ASCON [39], WG-7 [41], WG-8 [42], Hummingbird [43], Hummingbird-2 [44], ACORN [47], Sablier [48]
Exhaustive key search	TRIVIUM [22], [24]
Time memory-data trade-off attacks, timing attacks	ZUC [18], SOSEMANUK [19], Fruit -V2 [25] Platelet [26], WG-7 [41], WG-8 [42], GRAIN-128a [45], CAR30 [50]
Collision analysis/attacks, internal-state collision	E0 [11], [12], SOSEMANUK [19], QUAVIUM [27], Kreyvium [28], ASCON [39]
Divide and conquer attack	E0 [11], [12], Rabbit [36]
Side channel attacks	MICKEY [13], [15], KECCAK [38], GRAIN-128a [45], CAR30 [50]
Key recovery attacks, related-key attacks	MICKEY [13]–[15], Platelet [26], Rabbit [36], Hummingbird [43], GRAIN-128a [45], BEAN [49]
Distinguishing attacks, Algebraic attacks	SOSEMANUK [19], Platelet [26], HC-128 [32], WG-8 [42], Sablier [48] SOSEMANUK [19], Fruit -V2 [25], Rabbit [34], WG-7 [41], WG-8 [42], Hummingbird [43], Hummingbird -2 [44], GRAIN-128a [45], CAR30 [50]
Fault attacks	GRAIN [20], [21], Fruit -V2 [25], GRAIN-128a [45], CAR30 [50]
Cube-attacks	TrivA [23], Fruit -V2 [25], WG-7 [41], Hummingbird [43], ACORN [47], Sablier [48]
Guess and determine attack	Fruit -V2 [25], Rabbit [36], ACORN [47], Sablier [48], CAR30 [50]
Correlation attacks	A5/1 [11], RC4 [30], Rabbit [34], [36], MORUS v1 [37], WG-7 [41], WG-8 [42], Rabbit-MAC [46]
Chosen-plain text attacks	A2U2 [40]
Side resynchronization attack	MICKEY [13]–[15], CAR30 [50]
Bitstream modification attack	SNOW [17]
Polynomial density analysis	TrivA [23]
Exhaustive key search	TRIVIUM [22,24], HC-128 [32], Rabbit [36]
Clock fault Injection	TRIVIUM [22], [24]
Discrete fourier transform attack	WG-7 [41], WG-8 [42]

4. PERFORMANCE ANALYSIS AND APPLICATIONS

This section discusses the hardware realization of the stream ciphers and their performance analysis. Most of the authors implemented the stream ciphers using the field programmable gate array (FPGA) platform. The stream ciphers are constructed with macroblocks using hardware description language (HDL) and later implemented on FPGA. The performance metrics include area in terms of slices, maximum operating frequency (Fmax) in terms of MHz, latency in terms of clock cycles (CC), throughput (Mbps), and efficiency

(Mbps/Slice). The design module uses program logic blocks, and programmable interconnects on FPGA. The FPGA contains configurable logic blocks (CLBs), input-output blocks (IOBs), dedicated multipliers, a digital clock manager (DCM), and block RAMs. The CLBs are constructed using slices and lookup tables (LUTs). The slice definition is varied based on FPGA device selection.

For example, one slice contains a minimum of two 4-input LUT, Flip-flops, adder tree, and multiplexors on Spartan-3 FPGA. The LUT holds the design information in the Boolean equations and Truth table. The maximum operating frequency is obtained after synthesis operation based on design architecture using the Xilinx tool. The latency is analyzed based on the execution of the design to generate the first output in the simulation process. The latency is calculated regarding CC in hardware realization. The throughput is measured based on input data width, frequency, and latency parameters. So, throughput = (input width * Fmax)/latency. The hardware efficiency is measured in terms of throughput per slice. The summary of the performance analysis of the other stream ciphers is listed in Table 4.

The performance analysis of MICKEY, GRAIN, and TRIVIUM-based stream ciphers is summarized in Table 5. Vendors like Xilinx (Spartan-3, Virtex-Series, and Artix-7) or Intel (Cyclone- 4) based FPGA devices implement these stream ciphers. Ciphers like Fruit-V2, Platelet, and RC4A are implemented on an ASIC-based platform using complementary metal–oxide–semiconductor (CMOS) technology. The stream ciphers like A5/1, E0, Fruit-V2, and Platelet offer low-latency designs than others.

Table 4. Performance analysis of stream ciphers

Authors	Year	Stream cipher	FPGA	Area (slices)	Fmax (MHz)	Latency (CC)	Throughput (Mbps)	Efficiency (Mbps/slices)
Galanis <i>et al.</i> [11]	2005	A5/1	Virtex-2	32	188.3	1	188.3	5.88
Gaj <i>et al.</i> [52]	2008	A5/1	Spartan-3	287	79	47	316	1.1
Galanis <i>et al.</i> [11]	2005	E0	Virtex-2	895	189	1	189	0.21
Kitsos <i>et al.</i> [53]	2012	E0	Spartan-3	140	187	1	187	1.335
Kitsos <i>et al.</i> [53]	2012	SNOW-3G	Spartan-3	3359	104	4	3328	0.99
Tsavos <i>et al.</i> [54]	2020	SNOW 2.0	Artix-7	3297	64	5	2040	0.618
Kitsos <i>et al.</i> [53]	2012	ZUC	Spartan-3	1147	38	4	1216	1.06
Wang <i>et al.</i> [55]	2020	ZUC-256	Cyclone-4	2300	115	4	3680	1.6
Ghafari <i>et al.</i> [25]	2016	Fruit -V2	90-nm	990	100	1	100	0.101
Mikhalev <i>et al.</i> [26]	2016	Platelet	18-nm	928	100	1	100	0.107
Pyrgas and Kitsos [56]	2020	Enocoro4	Artix-7	83	204	4	181	2.18
Pyrgas and Kitsos [56]	2020	Enocoro8	Artix-7	78	189	9	302	3.87
Galanis <i>et al.</i> [11]	2005	RC4	Virtex-2	140	60.8	2	120.8	0.86
Khalid <i>et al.</i> [31]	2016	RC4A	65-nm	37770	1300	512	10400	0.28
Bertoni <i>et al.</i> [38]	2012	KECCAK	Virtex-5	448	265	5160	52	0.12
Galanis <i>et al.</i> [11]	2005	W7	Virtex-2	608	96	8	768	1.26
Das <i>et al.</i> [50]	2013	CAR-30	Spartan-3	499	185.05	32	744	1.49

Table 5. Performance analysis of MICKEY, GRAIN, and TRIVIUM-based stream ciphers

Authors	Year	Stream cipher	FPGA	Area (Slices)	Fmax (MHz)	Latency (CC)	Throughput (Mbps)	Efficiency (Mbps/Slices)
Hwang <i>et al.</i> [57]	2008	MICKEY-2.0	Spartan-3	115	233	1	233	2.03
Li <i>et al.</i> [58]	2020	MICKEY 2.0	Spartan-7	78	250	1	250	3.21
Alharbi <i>et al.</i> [59]	2020	MICKEY 2.0	Spartan-6	225	370	1	370	1.64
Kitsos [14]	2006	MICKEY-128	Virtex-2	167	170	1	170	1.011
Hwang <i>et al.</i> [57]	2008	MICKEY-128	Spartan-3	176	223	1	223	1.27
Bulens <i>et al.</i> [60]	2007	MICKEY-128	Virtex-2	190	200	1	200	1.05
Kitsos <i>et al.</i> [53]	2012	MICKEY-128	Spartan-3	98	250	1	250	2.55
Alharbi <i>et al.</i> [59]	2020	MICKEY-128	Spartan-6	317	370	1	370	1.17
Hell <i>et al.</i> [20]	2007	GRAIN-80	Cyclone-2	1450	282	1	282	0.195
Hwang <i>et al.</i> [57]	2008	GRAIN-80	Spartan-3	348	130	16	2080	5.98
Gaj <i>et al.</i> [52]	2008	GRAIN-80	Spartan-3	356	155	19	2480	6.97
Kitsos <i>et al.</i> [53]	2012	GRAIN-80	Spartan-3	318	177	1	177	0.556
Li <i>et al.</i> [58]	2020	GRAIN-80	Spartan-7	62	333	1	333	5.37
Alharbi <i>et al.</i> [59]	2020	GRAIN-80	Virtex-7	133	693	161	693	5.24
Hwang <i>et al.</i> [57]	2008	GRAIN-128	Spartan-3	534	133	32	4256	7.97
Bulens <i>et al.</i> [60]	2007	GRAIN-128	Virtex-2	48	181	1	181	3.77
Alharbi <i>et al.</i> [59]	2020	GRAIN-128	Virtex-7	198	769	257	769	3.73
Chakraborti <i>et al.</i> [23]	2015	TRIVIA	Virtex-7	714	NA	NA	16000.89	23.65
Hwang <i>et al.</i> [57]	2008	TRIVIUM	Spartan-3	50	240	1	240	4.8
Gaj <i>et al.</i> [52]	2008	TRIVIUM	Spartan-3	388	190	21	12160	31.34
Bulens <i>et al.</i> [60]	2007	TRIVIUM	Virtex-2	41	207	1	207	5.05
Kitsos <i>et al.</i> [53]	2012	TRIVIUM	Spartan-3	149	326	1	326	2.18
Li <i>et al.</i> [58]	2020	TRIVIUM	Spartan-7	71	416	1	416	5.86
Alharbi <i>et al.</i> [59]	2020	TRIVIUM	Spartan-6	510	100	1	100	0.19

The A5/1 and Enocoro4/8 utilize less chip area, and SNOW-3G, ZUC-256, and RC4A utilize more chip area on FPGA. The ciphers like SNOW 2.0, SNOW-3G, ZUC, ZUC-256, and RC4A provide Gigabit speed more than other ciphers. The stream ciphers like MICKEY 2.0, MICKEY-128, GRAIN-80 [20], [53], [58], GRAIN-128 [60], and TRIVIUM [53], [57]–[60] offer low-latency (single clock cycle) designs than other designs. The ciphers like GRAIN-80 [52], [57], GRAIN-128 [57], TRIVIA [23], and TRIVIUM [52] provide Gigabit speed more than other ciphers. The throughput varies in each stream cipher based on Latency and Fmax.

Most stream ciphers are used in wireless applications, including wireless communication, Wi-Fi, wired equivalent privacy (WEP), and WSN. The stream cipher's applications are summarized and tabulated in Table 6. An A5/1 [11], MICKEY-128 [14], RC4 [30], WG-8 [42], Rabbit-MAC [46], and CAR30 [50] offer low-latency and lower power consumption and are suitable to use in wireless applications. The E0 cipher [11], [12] is specially designed to use in Bluetooth applications. The SNOW series ciphers are preferred in 3rd generation (3G) and 3GPP wireless standards. The SNOW-3G [17] provides better integrity and confidentiality in many 3G and 3GPP standards.

Table 6. Applications of the stream ciphers

Applications	Stream ciphers
Wireless communications, 802.11b, WEP, and WSN	A5/1 [11], MICKEY-128 [14], RC4 [30], WG-8 [42], Rabbit-MAC [46], and CAR30 [50]
Bluetooth	E0 [11], [12]
3G, 3GPP	SNOW [17]
LTE/4G	ZUC [18]
NVM	Platelet [26]
Homomorphic cryptography and compression	Kreyvium [28]
VLIW microprocessor and CPU	PANAMA [29], HC-128 [32], HC-256 [33], Rabbit [36], KECCAK [38], and ASCON [39]
Printed electronics, RFID tags, passive RFID system	A2U2 [40], WG-8 [42], Hummingbird -2 [44], and BEAN [49]
RFID authentications and low-power applications	WG-7 [41], Hummingbird [43], GRAIN-128a [45], Rabbit-MAC [46], ACORN [47], and Sablier [48]

In contrast, The ZUC series [18] ciphers are a modified version of the SNOW series and are preferred to use in 4G and LTE wireless standards. Platelet ciphers [26] are used in NVM for continuous key access. The Platelet cipher provides string security against attacks using double-layer LFSR and NLFSR. The Kreyvium cipher [28] is a new variant of Trivium that provides efficient real-time solutions to homomorphic-ciphertext-based compression applications. The general processor and VLIW based processors use PANAMA [29], HC-128 [32], HC-256 [33], Rabbit [36], KECCAK [38], and ASCON [39] stream ciphers. The encryption speed of these ciphers is commonly tested on Pentium series processors. The stream ciphers like A2U2 [40], WG-7 [41], WG-8 [42], Hummingbird [43], Hummingbird-2 [44], GRAIN-128a [45], Rabbit-MAC [46], ACORN [47], Sablier [48], and BEAN [49] are implemented to use for authentication of two or more devices. The RFID system, passive tags, internet of things (IoT), and low-power devices are commonly used in those ciphers to strengthen the data.

5. FUTURE TRENDS

The keystream generation is an essential part of the stream ciphers and the main functional requirement for most application domains. The stream ciphers' preamble remains the same, with their high performance and efficiency as the block ciphers. The recent trends towards IoT indicate that the millions of embedded devices are interconnected with resource constraints capabilities and interaction mechanisms with corresponding users. Social mobility and smart city applications need to include a distributed framework to transmit high amounts of cipher data securely. Most of the present industries, like 5th generation wireless networks, vehicular adhoc-networks, smart camera-based Urban-surveillance, and green networking, will focus more on security to secure their data from attackers.

Stream ciphers are the best option rather than block ciphers for streaming applications. However, research is still improving cipher usage in a well-organized manner. Currently, parallel computing systems are widely used in most embedded system applications. So, incorporating a lightweight stream cipher with high-degree parallelism is challenging in maintaining desired performance. Most current stream ciphers focus more on basic operations with cipher structures and can resist most of the existing attacks. However, these ciphers must incorporate most cryptographic properties for further security evaluation and performance analysis. Focus on internal state architecture resource utilization and power consumption while implementing the lightweight

ciphers. Implementing the AE methods using stream ciphers is still in demand because of the current trends in IoT usage. Security feature improvements using stream ciphers on cloud computing applications remain an open research spot.

6. CONCLUSION

As embedded or IoT gadgets increase in our daily lives, pervasive computing becomes a reality. Networked computers have undergone a significant change in their architecture, usage, and number to protect the security of those sources and the data kept on or transmitted to them. This manuscript presents an exhaustive review of the stream ciphers for low-constrained devices. The traditional and benchmarked stream cipher's design and authenticated ciphers are analyzed. The resistive streams ciphers for corresponding attacks are highlighted. The implemented results of these stream ciphers are examined in detail using the FPGA platform. From this, GRAIN-128, GRAIN-128A, TRIVIUM, and MICKEY stream ciphers provide better security and performance results than other ciphers. The most appropriate stream ciphers for corresponding application requirements are highlighted based on cryptographic functionalities. The requirements and systematic plans for future designs are highlighted.

REFERENCES




- [1] M. A. Philip and Vaithianathan, "A survey on lightweight ciphers for IoT devices," in *2017 International Conference on Technological Advancements in Power and Energy (TAP Energy)*, IEEE, Dec. 2017, pp. 1–4. doi: 10.1109/TAPENERGY.2017.8397271.
- [2] L. Jiao, Y. Hao, and D. Feng, "Stream cipher designs: a review," *Science China Information Sciences*, vol. 63, no. 3, pp. 1–25, Mar. 2020, doi: 10.1007/s11432-018-9929-x.
- [3] C. Manifavas, G. Hatzivasilis, K. Fysarakis, and Y. Papaefstathiou, "A survey of lightweight stream ciphers for embedded systems," *Security and Communication Networks*, vol. 9, no. 10, pp. 1226–1246, Jul. 2016, doi: 10.1002/sec.1399.
- [4] A. Perez-Resca, M. Garcia-Bosque, C. Sanchez-Azqueta, and S. Celma, "A new method for format preserving encryption in high-data rate communications," *IEEE Access*, vol. 8, pp. 21003–21016, 2020, doi: 10.1109/ACCESS.2020.2968816.
- [5] S. Gnatyuk, M. Iavich, V. Kinzyavyy, T. Okhrimenko, Y. Burmak, and I. Goncharenko, "Improved secure stream cipher for cloud computing," *CEUR Workshop Proceedings*, vol. 2732, pp. 183–197, 2020.
- [6] R. Chatterjee, R. Chakraborty, and J. K. Mandal, "Design of cryptographic model for end-to-end encryption in fpga based systems," in *2019 3rd International Conference on Computing Methodologies and Communication (ICCMC)*, IEEE, Mar. 2019, pp. 459–465. doi: 10.1109/ICCMC.2019.8819761.
- [7] A. Babaei and G. Schiele, "Physical unclonable functions in the internet of things: state of the art and open challenges," *Sensors*, vol. 19, no. 14, pp. 1–18, Jul. 2019, doi: 10.3390/s19143208.
- [8] A. Mars and W. Adi, "New family of stream ciphers as physically clone-resistant VLSI-structures," *Cryptography*, vol. 3, no. 2, pp. 1–22, Apr. 2019, doi: 10.3390/cryptography3020011.
- [9] T. Good, W. Chelton, and M. Benaissa, "Review of stream cipher candidates from a low resource hardware perspective," *SASC 2006 - Stream Ciphers Revisited*, pp. 125–148, 2006.
- [10] P. R. Hridya and J. Jose, "Cryptanalysis of the grain family of ciphers: a review," in *2019 International Conference on Communication and Signal Processing (ICCSP)*, IEEE, Apr. 2019, pp. 0892–0897. doi: 10.1109/ICCSP.2019.8697972.
- [11] M. D. Galanis, P. Kitsos, G. Kostopoulos, N. Sklavos, and C. E. Goutis, "Comparison of the hardware implementation of stream ciphers," *International Conference on Electronics, Circuits and Systems*, vol. 2, no. 4, pp. 267–274, 2005.
- [12] M. Hermelin and K. Nyberg, "Correlation properties of the bluetooth combiner," *ICISC 1999: Information Security and Cryptology - ICISC '99*, 2000, pp. 17–29. doi: 10.1007/10719994_2.
- [13] S. Babbage and M. Dodd, "The stream cipher MICKEY 2.0," *ECRYPT Stream Cipher Project, Report*, pp. 191–209, 2006.
- [14] P. Kitsos, "On the hardware implementation of the MICKEY-128 stream cipher," *Cryptology ePrint Archive*, 2005.
- [15] L. Ding and J. Guan, "Cryptanalysis of MICKEY family of stream ciphers," *Security and Communication Networks*, vol. 6, no. 8, pp. 936–941, Aug. 2013, doi: 10.1002/sec.637.
- [16] P. Ekdahl and T. Johansson, "A new version of the stream cipher SNOW," in *SAC 2002: Selected Areas in Cryptography*, 2003, pp. 47–61. doi: 10.1007/3-540-36492-7_5.
- [17] M. Moraitis and E. Dubrova, "Bitstream modification attack on SNOW 3G," in *2020 Design, Automation and Test in Europe Conference and Exhibition (DATE)*, IEEE, Mar. 2020, pp. 1275–1278. doi: 10.23919/DATE48585.2020.9116222.
- [18] G. Sekar, "The stream cipher core of the 3GPP encryption standard 128-EEA3: timing attacks and countermeasures," in *Information Security and Cryptology: 7th International Conference, Inscrypt 2011*, 2012, pp. 269–288. doi: 10.1007/978-3-642-34704-7_20.
- [19] C. Berbain *et al.*, "Sosemanuk, a fast software-oriented stream cipher," in *New stream cipher designs*, 2008, pp. 98–118. doi: 10.1007/978-3-540-68351-3_9.
- [20] M. Hell, T. Johansson, and W. Meier, "Grain: a stream cipher for constrained environments," *International Journal of Wireless and Mobile Computing*, vol. 2, no. 1, pp. 86–93, 2007, doi: 10.1504/IJWMC.2007.013798.
- [21] M. Hell, T. Johansson, A. Maximov, and W. Meier, "A stream cipher proposal: grain-128," in *2006 IEEE International Symposium on Information Theory*, IEEE, Jul. 2006, pp. 1614–1618. doi: 10.1109/ISIT.2006.261549.
- [22] C. D. Cannière, "Trivium: a stream cipher construction inspired by block cipher design principles," in *International Conference on Information Security*, Springer, Berlin, Heidelberg, 2006, pp. 171–186. doi: 10.1007/11836810_13.
- [23] A. Chakraborti, A. Chattopadhyay, M. Hassan, and M. Nandi, "Trivium: a fast and secure authenticated encryption scheme," in *International Workshop on Cryptographic Hardware and Embedded Systems*, Springer, Berlin, Heidelberg, 2015, pp. 330–353. doi: 10.1007/978-3-662-48324-4_17.
- [24] F. E. Potestad-Ordonez, C. J. Jimenez-Fernandez, and M. Valencia-Barrero, "Experimental and timing analysis comparison of FPGA trivium implementations and their vulnerability to clock fault injection," in *2016 Conference on Design of Circuits and Integrated Systems (DCIS)*, IEEE, Nov. 2016, pp. 1–6. doi: 10.1109/DCIS.2016.7845270.

- [25] V. Ghafari, H. Amin, and Y. Hu, "Fruit-v2: ultra-lightweight stream cipher with shorter internal state," *Cryptology ePrint Archive*, 2016.
- [26] V. Mikhalev, F. Armknecht, and C. Müller, "On ciphers that continuously access the non-volatile key," *IACR Transactions on Symmetric Cryptology*, pp. 52–79, Feb. 2017, doi: 10.46586/tosc.v2016.i2.52-79.
- [27] Y. Tian, G. Chen, and J. Li, "Quavium - a new stream cipher inspired by trivium," *Journal of Computers*, vol. 7, no. 5, pp. 1278–1283, May 2012, doi: 10.4304/jcp.7.5.1278-1283.
- [28] A. Canteaut *et al.*, "Stream ciphers: a practical solution for efficient homomorphic-ciphertext compression," *Journal of Cryptology*, vol. 31, no. 3, pp. 885–916, Jul. 2018, doi: 10.1007/s00145-017-9273-9.
- [29] J. Daemen and C. Clapp, "Fast hashing and stream encryption with panama," In *International Workshop on Fast Software Encryption*, Springer, Berlin, Heidelberg, 1998, pp. 60–74. doi: 10.1007/3-540-69710-1_5.
- [30] R. L. Rivest, "The RC4 encryption algorithm," RSA data security Inc., Mar. 1992.
- [31] A. Khalid, G. Paul, and A. Chattopadhyay, "RC4-accsuite: a hardware acceleration suite for rc4-like stream ciphers," *IEEE Transactions on Very Large Scale Integration (VLSI) Systems*, vol. 25, no. 3, pp. 1072–1084, Mar. 2017, doi: 10.1109/TVLSI.2016.2606554.
- [32] H. Wu, "The stream cipher HC-128," in *New Stream Cipher Designs*, Berlin, Heidelberg: Springer Berlin Heidelberg, pp. 39–47. doi: 10.1007/978-3-540-68351-3_4.
- [33] H. Wu, "A new stream cipher HC-256," in *Fast Software Encryption: 11th International Workshop, FSE 2004*, Delhi, India: Springer, Berlin, Heidelberg, 2004, pp. 226–244. doi: 10.1007/978-3-540-25937-4_15.
- [34] M. Robshaw, O. Billet, and (Eds.), *New stream cipher designs - the eSTREAM finalists*, vol. 4986. in *Lecture Notes in Computer Science*, vol. 4986. Berlin, Heidelberg: Springer Berlin Heidelberg, 2008. doi: 10.1007/978-3-540-68351-3.
- [35] D. J. Bernstein, "ChaCha, a variant of salsa20," *Workshop Record of SASC*, pp. 1–6, 2008.
- [36] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, "Rabbit: a new high-performance stream cipher," In *International Workshop on Fast Software Encryption*: Springer, Berlin, Heidelberg, 2003, pp. 307–329. doi: 10.1007/978-3-540-39887-5_23.
- [37] A. Mileva, V. Dimitrova, and V. Velichkov, "Analysis of the authenticated cipher MORUS (v1)," in *In International Conference on Cryptography and Information Security in the Balkans*, Cham: Springer International Publishing, 2015, pp. 45–59. doi: 10.1007/978-3-319-29172-7_4.
- [38] G. Bertoni, J. Daemen, M. Peeters, G. Van Assche, and R. Van Keer, "1001 ways to implement keccak," *Third SHA-3 candidate conference*, Washington DC, pp. 4–8, 2012.
- [39] Dobraunig, Christop, M. Eichlseder, F. Mendel, and M. Schläffer, "Ascon lightweight authenticated encryption and hashing," *Submission to the CAESAR competition*, Accessed date: Jan. 15, 2023, [Online] Available at: <https://ascon.iaik.tugraz.at/>.
- [40] M. David, D. C. Ranasinghe, and T. Larsen, "A2U2: A stream cipher for printed electronics RFID tags," *2011 IEEE International Conference on RFID, RFID 2011*, pp. 176–183, 2011, doi: 10.1109/RFID.2011.5764619.
- [41] Y. Luo, Q. Chai, G. Gong, and X. Lai, "A lightweight stream cipher WG-7 for RFID encryption and authentication," in *2010 IEEE Global Telecommunications Conference GLOBECOM 2010*, Dec. 2010, pp. 1–6. doi: 10.1109/GLOCOM.2010.5684215.
- [42] X. Fan, K. Mandal, and G. Gong, "WG-8: a lightweight stream cipher for resource-constrained smart devices," *Greder Noida, India: Springer, Berlin, Heidelberg*, 2013, pp. 617–632. doi: 10.1007/978-3-642-37949-9_54.
- [43] D. Engels, X. Fan, G. Gong, H. Hu, and E. M. Smith, "Hummingbird: ultra-lightweight cryptography for resource-constrained devices," in *In International Conference on Heterogeneous Networking for Quality, Reliability, Security, and Robustness*, Springer, Berlin, Heidelberg, 2010, pp. 3–18. doi: 10.1007/978-3-642-14992-4_2.
- [44] D. Engels, M.-J. O. Saarinen, P. Schweitzer, and E. M. Smith, "The Hummingbird-2 lightweight authenticated encryption algorithm," in *International workshop on radio frequency identification: security and privacy issues*, Springer Berlin Heidelberg, 2012, pp. 19–31. doi: 10.1007/978-3-642-25286-0_2.
- [45] M. Ågren, M. Hell, T. Johansson, and W. Meier, "Grain-128a: a new version of Grain-128 with optional authentication," *International Journal of Wireless and Mobile Computing*, vol. 5, no. 1, pp. 48–59, 2011, doi: 10.1504/IJWMC.2011.044106.
- [46] R. Tahir, M. Y. Javed, and A. R. Cheema, "Rabbit-MAC: lightweight authenticated encryption in wireless sensor networks," in *2008 International Conference on Information and Automation*, Jun. 2008, pp. 573–577. doi: 10.1109/ICINFA.2008.4608065.
- [47] H. Wu, "ACORN: a lightweight authenticated cipher (v3)," *CAESAR: Competition for authenticated encryption: Security, applicability, and robustness*, 2016.
- [48] B. Zhang, Z. Shi, C. Xu, Y. Yao, and Z. Li, "Sablier v1," *Candidate for the CAESAR Competition*, 2014, pp. 1-38.
- [49] N. Kumar, S. Ojha, K. Jain, and S. Lal, "BEAN: a lightweight stream cipher," in *Proceedings of the 2nd international conference on Security of information and networks - SIN '09*, New York, USA: ACM Press, 2009, p. 168. doi: 10.1145/1626195.1626238.
- [50] S. Das and D. RoyChowdhury, "CAR30: A new scalable stream cipher with rule 30," *Cryptography and Communications*, vol. 5, no. 2, pp. 137–162, Jun. 2013, doi: 10.1007/s12095-012-0079-1.
- [51] T. Chen, L. Ge, X. Wang, and J. Cai, "TinyStream: a lightweight and novel stream cipher scheme for wireless sensor networks," in *2010 International Conference on Computational Intelligence and Security*, Dec. 2010, pp. 528–532. doi: 10.1109/CIS.2010.121.
- [52] K. Gaj, G. Southern, and R. Bachimanchi, "Comparison of hardware performance of selected phase II eSTREAM candidates," *State of the Art of Stream Ciphers Workshop (SASC)*, pp. 1–11, 2007.
- [53] P. Kitsos, N. Sklavos, G. Provelengios, and A. N. Skodras, "FPGA-based performance analysis of stream ciphers ZUC, Snow3g, Grain V1, Mickey V2, Trivium and E0," *Microprocessors and Microsystems*, vol. 37, no. 2, pp. 235–245, Mar. 2013, doi: 10.1016/j.micpro.2012.09.007.
- [54] M. Tsavos, N. Sklavos, and G. P. Alexiou, "Lightweight security data streaming, based on reconfigurable logic, for FPGA platform," in *2020 23rd Euromicro Conference on Digital System Design (DSD)*, IEEE, Aug. 2020, pp. 277–280. doi: 10.1109/DSD51259.2020.00052.
- [55] Y. Wang, L. Wu, X. Zhang, K. Xu, and W. Yang, "A hardware implementation of ZUC-256 stream cipher," in *Proceedings of the International Conference on Anti-Counterfeiting, Security and Identification, ASID*, IEEE, Oct. 2020, pp. 94–97. doi: 10.1109/ASID50160.2020.9271719.
- [56] L. Pyrgas and P. Kitsos, "Compact hardware architectures of enocoro-128v2 stream cipher for constrained embedded devices," *Electronics (Switzerland)*, vol. 9, no. 9, pp. 1–14, 2020, doi: 10.3390/electronics9091505.
- [57] D. Hwang, M. Chaney, S. Karanam, N. Ton, and K. Gaj, "Comparison of FPGA-targeted hardware implementations of (e)STREAM stream cipher candidates," *State of the Art of Stream Ciphers Workshop, (SASC) 2008, Lausanne, Switzerland*, pp. 151–162, 2008.
- [58] B. Li, M. Liu, and D. Lin, "FPGA implementations of grain v1, mickey 2.0, trivium, lizard and plantlet," *Microprocessors and Microsystems*, vol. 78, pp. 1–13, Oct. 2020, doi: 10.1016/j.micpro.2020.103210.




- [59] F. Alharbi, M. K. Hameed, A. Chowdhury, A. Khalid, A. Chattopadhyay, and I. T. Javed, "Analysis of area-efficiency vs. unrolling for eSTREAM hardware portfolio stream ciphers," *Electronics*, vol. 9, no. 11, pp. 1–17, Nov. 2020, doi: 10.3390/electronics9111935.
- [60] P. Bulens, K. Kalach, F.-X. Standaert, and J.-J. Quisquater, "FPGA implementations of eSTREAM phase-2 focus candidates with hardware profile," *In-State of the Art of Stream Ciphers Workshop (SASC 2007)*, Jan. 2007.

BIOGRAPHIES OF AUTHORS



Raghavendra Ananth    holds a bachelor's degree in electronics and communication engineering from HMSIT college, Tumkur, and a master's degree in VLSI design and embedded system from REVA ITM College, Bangalore, Karnataka. Presently he is a research scholar in the Department of Electronics and Communications, JAIN Deemed to be University, Bangalore. His research areas include IoT security, VLSI Frontend design, and FPGA. He can be contacted at email: araghavendra.research@gmail.com.



Narayana Swamy Ramaiah    received Ph.D. from PRIST University, Tamil Nadu, in the year 2016, M.Tech. in the year 2004 from Visvesvaraya Technological University, Karnataka and B.E. in the year 2002 from Bangalore University, Karnataka. He has over 16 years of experience in teaching, research, and industry. He has published over 45 papers in peer-reviewed journals. His research areas include IoT (agriculture), blockchain, AI and machine learning, and cloud computing. He is currently working as a professor at Jain, deemed University Bangalore. He can be contacted at email: r.narayanaswamy@jainuniversity.ac.in.