

Design, Validation and Characterization Grain-128 AEADv2 Lightweight Crypto IP Core

Fathimathu Ruksana K K¹Nandakumar R²

¹Defence Institute of Advanced Technology (DIAT) Pune, Maharashtra, India fathimathuruksana2@gmail.com

²NIELIT Calicut, Kerala, India nanda24x7@gmail.com

Abstract. The need for low resource devices has significantly increased as a result of the development of Internet-of-things applications. Various light weighted stream cipher algorithms have emerged as promising solutions for securing IoT communications. They are used on hardware because of its simplicity, speed, limited resources and processing power. Lightweight cryptography overcomes these constraints by optimum utilization of the resources available, while maintaining security. Because of their economical implementations, Grain128AEAD ciphers stand out among other stream ciphers. There are different variants of the Grain cipher, and Grain-128AEADv2 is one of the ciphers. Grain 128 AEADv2 cipher with 64 bits plaintext and 128 bits key 96 bit nonce(IV), 64 bits associated data forming a 64 bits ciphertext and 64 bits Tag. The Grain 128AEAD cipher is small and quite efficient. In this paper, Design validation and characterization of grain128AEADv2 cipher IP core. The hardware platform used the Xilinx Artix-7 FPGA board and Vivado is used for simulations, synthesis, and implementations utilizing the Verilog hardware description language. The performance parameters results were observed with the help of Xilinx Vivado.

Keywords: Stream cipher, FPGA, grain128aeadv2, grain, Lightweight cipher.

(Article history: Selected papers from NIELIT's International Conference on Communication, Electronics and Digital Technologies (NICE-DT'26), 29-30Jan, 2026, Gorakhpur, India)

1. Introduction

The rapid adoption of Internet of Things (IoT) devices has transformed several areas, including smart homes and industrial automation. However, the security of these interconnected devices remains a significant concern due to their resource constrained nature. The operating environments of the devices are those with extremely restricted energy, memory, computing power, and space resources [1]. It is difficult to achieve high levels of security with little resources. It is crucial to maximize throughput, reduce area consumption, and maintain security while implementing a cipher on hardware. In today's communication landscape, security, efficiency, cost, and power consumption form interrelated constraints. This sparked the development of a new cryptography paradigm which is Lightweight Cryptography. The goal of lightweight cryptography is to create algorithms with fewer memory, processing power, and energy consumption [2]. The problem is to maintain high-level security on low-resource devices (LRDs). As a result, maximizing throughput and area consumption needs specific attention when implementing cryptographic algorithms. wireless body networks (WBAN), Smart cards (SD), RFID and the Industrial Internet of Things (IIoT) are the most widely used low-resource technologies. With the use of the Industrial Internet of Things, businesses can collect, compile, and evaluate sensor data. Aiming to optimize machine efficiency and enhance overall operational throughput. According to [3], Xilinx offers versatile, standards-based FPGA solutions that include hardware optimization, software programmability, real-time processing, and universal connection. While also offering the safety and security elements that are essential for Industrial IoT systems. One notable feature of the Artix-7 FPGA is its great reliability and versatility. In many applications, preserving the secrecy and integrity of the data exchanged across the network presents a major difficulty. Cryptography algorithms fall into two groups. symmetric ciphers and asymmetric ciphers, symmetric crypto systems that rely on private keys, but public keys are used in asymmetric ciphers. Using the recipient's public key, the sender encrypts the message in an asymmetric algorithm, and the original message is recovered by the recipient using the secret key to decrypt the encrypted text. With a symmetric cipher, the same secret key is used for decryption and encryption on both ends of the transmission. Block ciphers and stream ciphers are two different kinds of symmetric ciphers. However, because of their speed and stream ciphers are typically chosen. While the message is bit-by-bit encrypted in stream ciphers [4] with a pseudo-random key bit-stream, Fixed-size blocks are encrypted using in block ciphers. The need for cryptography has been acknowledged by the community, and as a result, researchers and developers have concentrated on creating encryption that is suited for implementations in both hardware and software that have limited computational resources.

1.1 Grain Cipher

Large gate foot prints and memory are needed for the development of block ciphers in cryptography. They are not used on devices with limited resources as a result. However, because of their throughput and ease of usage, stream ciphers can be employed in contexts with limited resources. It is this nonlinear filter generator concept that forms the basis of the Grain family of

stream ciphers. A stream cipher family called Grain is built around a nonlinear filter generator. To give the keystream sequence non-linearity, a nonlinear function is applied using parts of the LFSR sequence as input. In order to fortify its resilience to algebraic assaults and augment the intricacy of the correlation between the state/key and keystream. Grain integrates the nonlinear combiner with an NFSR. An 80-bit key is used in Grain v1, while a 128-bit key bit version was suggested in [5]. Grain-128 was developed with greater vigor in light of prior Grain building accomplishments, producing a nonlinear degree 2 NFSR feedback function, but with notable durability and non-linearity. In [6], cryptanalysis of a significant number of initialization rounds was accomplished by taking advantage of the low degree functions. These findings showed that a larger security margin was required for the nonlinear functions [7]. When Grain 128a was presented in [8], a choice authentication mode was included along with an increase in the nonlinear feedback function's degree. Grain 128 was later improved, showing how complex Boolean operations become more and more necessary. The foundation of Grain-128AEADv2 is Grain 128AEAD [9]. In the initialization phase, a modification is made to safeguard the key reconstruction from a known state. Later improvements to Grain 128 were made, highlighting the requirement for increasingly intricate Boolean operations.

1.2 Grain-128 AEADV2

Authenticated encryption with support for related data is provided via the Grain-128AEADv2 cipher. Grain-128a is very similar to this specification. The key and IV have different sizes: 128 and 96 bits, respectively. Grain-128AEADv2 [10] is made up of two main components. Pre-output and authenticator generators. A pre-output function (y), a non-linear feedback shift register (NFSR), and a linear feedback shift register (LFSR) make up the pre-output generator. Accumulator and shift register make up the authenticator generator. A stream of pseudo-random bits is produced by the pre-output generator, which serve as the authentication tag and encryption.

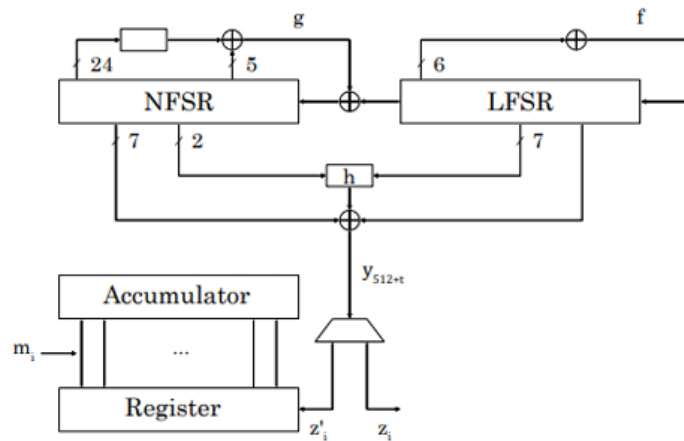


Fig. 1. Building blocks for Grain-128AEADv2

Building blocks for Grain-128AEADv2

$$S(t) = [s_0^t, s_1^t, \dots, s_{127}^t]$$

Whereas NFSR is represented by

$$B(t) = [b_0^t, b_1^t, \dots, b_{127}^t]$$

A pre-output generator's 256-bit state is represented by these two shift registers.

The LFSR's function is described as

$$L(S_t) = S_{127t+1} = S_{0t} + S_{7t} + S_{38t} + S_{70t} + S_{81t} + S_{96t}$$

The NFSR's function is described as

$$b_{127}^{t+1} = S_0^t + F(B_t)$$

$$b_{127}^{t+1} = s_0^t + b_0^t + b_{26}^t + b_{56}^t + b_{91}^t + b_{96}^t + b_3^t b_{67}^t + b_{11}^t b_{13}^t + b_{17}^t b_{18}^t + b_{27}^t b_{59}^t + b_{68}^t b_{84}^t + b_{22}^t b_{24}^t b_{25}^t + b_{70}^t b_{78}^t b_{82}^t + b_{88}^t b_{92}^t b_{93}^t b_{95}^t + b_{40}^t b_{48}^t + b_{61}^t b_{65}^t$$

The LFSR provides seven of the nine variables that make up the Boolean function $h(x)$, while the NFSR provides the remaining two.

$$h(x) = b_{12}^t s_8^t + s_{13}^t s_{20}^t + b_{95}^t s_{42}^t + s_{60}^t s_{79}^t + b_{12}^t b_{95}^t s_{94}^t$$

Next, the pre-output function provides the pre-output generator's output.

$$y_t = s_{93}^t + b_2^t + b_{15}^t + b_{36}^t + b_{45}^t + b_{64}^t + b_{73}^t + b_{89}^t + h(x)$$

There are 320 initialization clocks in Grain-128AEADv2. The cipher then reintroduces the key, subsequently the Accumulator and Register's startup. Before keystream generation starts, this adds up to $320+64+128 = 512$ initialization clocks. Thus, Grain-128AEADv2 delivers security in a manner that is comparable to that of Grain's earlier iterations.

2. Initialization

Grain-128AEADv2, the cipher undergoes 320 clock cycles, this involves returning the LFSR and NFSR inputs to the pre output function, which XORed them

$$s_{127}^{t+1} = y_t + L(\zeta(1))$$

$$0 \leq t \leq 319$$

$$b_{127}^{t+1} = s_0^t + F(B_t) + (2)$$

The cipher undergoes 64 clock cycles, during which the key is reintroduced and XORed with the input for both NFSR and LFSR

$$s_{127}^{t+1} = y_t + L(S_t) + K_{(t-256)}(3)$$

$$320 \leq t \leq 383$$

$$b_{127}^{t+1} = S_0^t + F(B_t) + y_t + K_{(t-320)}$$

The accumulator register is loaded with the pre-output key stream to initialize the authenticator generator

$$a_j^0 = y_{384}(5)$$

$$0 \leq t \leq 63$$

$$r_j^0 = y_{384}(6)$$

Then LFSR.NFSR updated as

$$s_{127}^{t+1} = L(\zeta(7))$$

$$384 \leq t \leq 511$$

$$b_{127}^{t+1} = S_0^t + F(B_t)(8)$$

The cipher has completed its initialization. S512 and B512, respectively, represent the LFSR and NFSR states. Keystream generation occurs as

$$Z_i = y_{(512+i)}(9)$$

The authentication bits are represented as

$$Z_i = y_{(512+2i)}(10)$$

The encrypted message represented as

$$C_i = Z_i + m_i$$

3. Implementation using Xilinx Vivado

In software platforms, cryptographic algorithms are implemented to assess their performance, aiming to achieve optimal utilization of device resources in dimensions such as size, speed, memory, power, and energy.

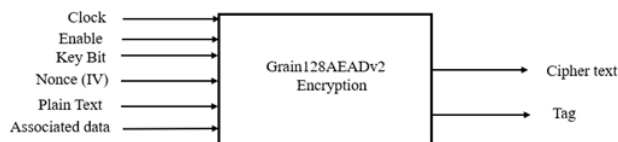


Fig. 2.Input-Output diagram of grain128aead cipher

The input-output (I/O) diagram gives an overall idea of a design. The design has 6 inputs: a positive edge triggered clock, an active high load, a 64-bit plain text, 96 bit nonce, and 128-bit key, 64-bit associated data and clock. Output produced is 64-bit cipher text and 64-bit Tag. Based on the I/O diagram (Fig. 2), the grain128aeadv2 cipher is simulated in Xilinx Vivado using Verilog Language. The function table below lists the test vectors that were employed along with the ciphertext that was generated as a result.

Table 1.Test vector For Grain 128aead cipher.

Plain Text	Key bit	Nonce	Ciphertext
000102030 4050607	00123456789abcdef 123456789abcdef	123456789abc def123456789	3c053c2810 7ce145
000102030 4050607	0fedcba9876543210 fedcba987645210	fedcba987654 3210 fedcba98	32b0bf7bcc 550afb
000102030 4050607	000000123456789000 00000123456789	0000001234567 890000000012	2385d5c766 0bed47

4. Simulation Result

Xilinx Vivado® is used for synthesis and analysis of Verilog HDL design. RTL schematic, power reports, timing analysis, utilization report, DRC checks, etc. are produced after completing synthesis and implementing the design. The Grain128aeadv2 cipher is simulated and implemented in Verilog using Xilinx Vivado® 2020.2 and synthesized on Xilinx Ar-tix-7 FPGA® to evaluate its hardware performance.

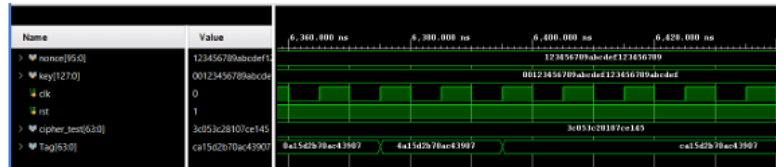


Fig. 3. Vivado simulation result of test vector 1



Fig. 4. Vivado simulation result of test vector 2

A. Throughput and Timing Analysis .

Timing analysis is carried out by taking into account the FPGA device's architecture, the logic that is used, and an examination of the default interconnecting route delays. Maximum operating frequency is calculated from the timing report. Given a constraint of 10000ns as the clock time period, the worst negative slack is obtained as 9995.674ns (Fig. 5). From this, the maximum operating frequency was calculated to be 231.11 MHz. Latency is the time taken for an instruction to be completed. This time, it refers to the quantity of cycles required to encrypt a message. Taking one clock cycle for each round, latency of this design is 512 cycles (512 rounds of cipher and 1 for input loading) at an operational frequency of 100MHz. The system's performance is gauged by its throughput, or the speed at which inputs are processed. It depends on the maximum frequency, block size and latency

$$\text{Throughput} = \text{maximum frequency} \times \text{total bits} / \text{latency}$$

$$\text{Throughput} = \text{Maximum frequency} * \text{total bits} / (\text{Latency}) = (231.11 * 64) / (512)$$

$$\text{Throughput} = 28.88 \text{ Mbps.}$$

Design Timing Summary		
Setup	Hold	Pulse Width
Worst Negative Slack (WNS): 9995.674 ns	Worst Hold Slack (WHS): 0.047 ns	Worst Pulse Width Slack (WPWS): 4999.441 ns
Total Negative Slack (TNS): 0.000 ns	Total Hold Slack (THS): 0.000 ns	Total Pulse Width Negative Slack (TPWS): 0.000 ns
Number of Failing Endpoints: 0	Number of Failing Endpoints: 0	Number of Failing Endpoints: 0
Total Number of Endpoints: 1275	Total Number of Endpoints: 1275	Total Number of Endpoints: 748
All user specified timing constraints are met.		

Fig. 5. Timing summary report

B. Resource Utilization Result

The resources used in the implementation phase are summarized in the table below. Gate equivalent is obtained from the resource utilization table. One LUT is generally considered to be made up of 4 ASIC gates and one flipflop is equivalent to 8 gates, giving a total of 5612 as the gate equivalent count.

Table 2. Resource Utilization Summary

Resource	Number utilized
LUT	453
FF	475

C. Power Utilization Summary

One of the reasons why FPGA is widely used in many applications is due to its low power consumption. Both static and dynamic power are present in the apparatus. Transistor leakage currents are the cause of static power. The junction temperature determines this switching activity brought on by capacitive load charging and draining is what generates dynamic power. The device's overall on-chip power consumption in this architecture is 86mW.

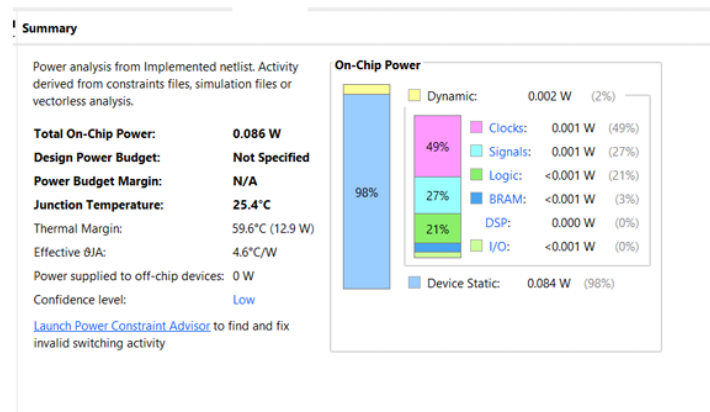


Fig. 6. Summary of power analysis report

D. On-chip debugging

The Verilog code was implemented on Xilinx Artix-7 FPGA Evaluation board XC7A100TCSG324-1. making use Virtual Input Output (VIO), Integrated Logic Analyzer (ILA) and IP cores of Xilinx Vivado2022.2. The on chip debugging results were compared with the results obtained after simulation and synthesis.

hw_vio_1					
Name	Value	Activity	Direction	VIO	
> Tag	[H] AD10_05F0_2250_F606		Input	hw_vio_1	
> Cipher	[H] D9F4_5FEB_0FC1_6E93		Input	hw_vio_1	
> Key	[H] 0001_0203_0405_0607_0809_0A0B_0C0D_0E0F		Output	hw_vio_1	
> Nonce	[H] 0001_0203_0405_0607_0809_0A0B		Output	hw_vio_1	
> Reset	1		Output	hw_vio_1	

Fig. 7. On-chip debugging result of test vector

5. Conclusion

In this work, we implemented optimized version of Grain 128AEADv2 stream cipher in hardware using 28nm library. This project was aimed at implementation of a Grain 128AEADv2 cipher algorithm in order to improve throughput without compromising power. From the results obtained it can be observed that, the objectives of the project were met. Higher throughput was achieved while utilizing less number of FPGA resources and keeping power and energy consumption low. we have achieved a28.88Mbps in throughput without affecting the area and power consumption. Throughput of 28.88Mbps was obtained with total on-chip power consumption of 86mW and energy per bits consumption of 0.068nJ. These results were obtained on Xilinx Vivado and debugged using the Xilinx Artix-7 FPGA Evaluation board XC7A100TCSG324-1.

Acknowledgment

I would like to express gratitude to everyone who contributed to the project's success, whether directly or indirectly. Thanks for your support and motivation. I express my gratitude to my guide for helping me throughout the period of research and reaching completion of the project

References

1. M. A. Philip and Vaithiyanathan, "A survey on lightweight ciphers for iot devices," in 2017 International Conference on Technological Advancements in Power and Energy (TAP Energy), pp. 1–4, 2017.
2. A. Putnam, A. M. Caulfield, E. S. Chung, D. Chiou, K. Constantinides, J. Demme, H. Esmaeilzadeh, J. Fowers, G. P. Gopal, J. Gray, M. Hasel man, S. Hauck, S. Heil, A. Hormati, J.-Y. Kim, S. Lanka, J. Larus, E. Peterson, S. Pope, A. Smith, J. Thong, P. Y. Xiao, and D. Burger, "A reconfigurable fabric for accelerating large-scale datacenter services," in 2014 ACM/IEEE 41st International Symposium on Computer Architec ture (ISCA), pp. 13–24, 2014.
3. K. Mohajerani, M. Tempelmeier, F. Farahmand, E. Homsirikamol, W. Diehl, J.-P. Kaps, and K. Gaj, "Implementer's guide to hardware implementations compliant with the hardware api for lightweight cryp tography version 1.2. 0," 2022.
4. L. Jiao, Y. Hao, and D. Feng, "Stream cipher designs: a review," *Science China Information Sciences*, vol. 63, no. 3, p. 131101, 2020.
5. M. Hell, T. Johansson, A. Maximov, and W. Meier, "A stream cipher proposal: Grain-128," in 2006 IEEE International Symposium on Information Theory, pp. 1614–1618, 2006.
6. L. Karlsson, M. Hell, and P. Stankovski, "Not so greedy: Enhanced subset exploration for non-randomness detectors," in *International Conference on Information Systems Security and Privacy*, vol. 867, pp. 273–294, Springer, 2018.
7. P. Stankovski, "Greedy distinguishers and nonrandomness detectors," in *Progress in Cryptology- INDOCRYPT 2010* (G. Gong and K. C. Gupta, eds.), (Berlin, Heidelberg), pp. 210–226, Springer Berlin Heidelberg, 2010.
8. M. ° Agren, M. Hell, T. Johansson, and W. Meier, "Grain-128a: a new version of grain-128 with optional authentication," *Int. J. Wire. Mob. Comput.*, vol. 5, p. 48–59, dec 2011.
9. M. Hell, T. Johansson, W. Meier, J. S"onnerup, and H. Yoshida, "Grain 128 aead-a lightweight aead stream cipher; 2019," *Lightweight Cryptography (LWC) Standardization*.
10. M. Hell, T. Johansson, A. Maximov, W. Meier, and H. Yoshida, "Grain 128aeadv2: Strengthening the initialization against key reconstruction," in *International Conference on Cryptology and Network Security*, pp. 24–41, Springer.
11. Verilog Beginner, VB (2019, August 8), Determining the maximum operating frequency in Xilinx Vivado [Online]. Available: <https://verilogbeginner.wordpress.com/>