Intelligent Automation & Soft Computing
DOI:10.32604/iasc.2021.018560
Article

Security Empowered System-on-Chip Selection for Internet of Things

Ramesh Krishnamoorthy* **and Kalimuthu Krishnan**

Department of ECE, SRM Institute of Science and Technology, Chennai, 603203, India
*Corresponding Author: Ramesh Krishnamoorthy. Email: rameshk.tn@gmail.com
Received: 12 March 2021; Accepted: 25 April 2021

Abstract: Due to the rapid growth of embedded devices, the selection of System-on-Chip (SoC) has a stronger influence to enable hardware security in embedded system design. System-on-chip (SoC) devices consist of one or more CPUs through wide-ranging inbuilt peripherals for designing a system with less cost. The selection of SoC is more significant to determine the suitability for secured application development. The design space analysis of symmetric key approaches including rivest cipher (RC5), advanced encryption standard (AES), data encryption standard (DES), international data encryption algorithm (IDEA), elliptic curve cryptography (ECC), MX algorithm, and the secure hash algorithm (SHA-256) are compared to identify the suitable algorithm for implementation of on-chip security. The implementation of state-of-the-art crypto functions on FPGA for design space findings provide the power and area consumption requirement. The proposed work utilizes the Genetic Algorithm (GA) optimization technique to determine the fitness of the SoC with enhanced crypto algorithms to enable device security. To validate the device result attained through the GA model, the security benchmarks are implemented on GA resultant hardware devices and analyzed the execution time and performance. The accuracy of the algorithm is estimated using the confusion matrix method and attained 89.66% accuracy. Besides, several challenges that need to overcome for IoT system design from prototype model to the industrial application are discussed extensively.

Keywords: Crypto-cores; device security; internet-of-things (IoT); genetic optimization; system-on-chip (SoC)

1 Introduction

In this paper, we discuss the crypto algorithms that are incorporated in the state-of-the-art microcontrollers to provide secure design practices to IoT computing devices. The system-on-chip (SoC) consists of a core processor, random access memory (RAM), and read-only memory (ROM) on a single chip. The traditional MCUs do not consist of advanced central processing unit (CPU) cores. While the modern microcontroller units (MCUs) are designed with advanced 32-bit and 64-bit processors. The designers should focus on adopting high level of security to the IoT systems. The hackers work hard to intrude the device data during the communication. Therefore, the IoT applications require several layer of security to prevent hardware attacks [1].

The average prediction time and accuracy of various CPU architectures including digital signal processor (DSP) TMS320C6713, GPU GeForce 540m, CPU Atom Z530, and Core 2 Duo E7500 processor are analyzed for the proper selection of CPU [2]. The performance of the genetic algorithm was tested in different computation engines to estimate the processing time by executing the concurrent tasks. The graphics processing units (GPUs) that run in parallel have produced high performance and utilized for applications that involve fast computation [3]. The various security threats that are relevant to microcontroller-based designs and the corresponding impacts are demonstrated using multiple case studies [4]. To enable hardware security within the SoCs a physically downloadable function was created on a chip to perform encryption and device authentication procedures [5].

The vulnerabilities and challenges for designing the SoC-based IoT systems involve a trusted hardware environment to support hardware security. As the usage of microcontroller in IoT design increases, incorporating the intellectual property (IP) core within the SoC can bring more enhanced security features [6]. Upon transmitting raw data from the microcontroller device as plain text that may be susceptible towards attacks. Therefore, the MD5 algorithm is implemented on a low-cost Arduino microcontroller board to perform encryption operations before transmitting the data to the receiver [7]. The challenges of designing secured microcontrollers considering significant efforts in engineering, balancing performance, and the production cost were presented in [8]. The implementation of the AES algorithm on low-cost 8-bit microcontrollers to achieve data integrity between the devices upon exchanging the data through an unsecured channel [9]. Chao et al. have discussed the security issues concerning microcontroller-centered firmware development through developing a prototype model for the automatic secure upgrade on the ATmega128P microcontroller chip [10]. An intermittent architecture was proposed to enable security in off-the-shelf MCUs without any hardware changes [11]. The invalid code is inserted onto the code memory regions and the integrity is validated for authentication purposes. Similarly, cyclic redundancy code (CRC) is injected to recognize the invalid code region in the maximum flash ROM size of 64KB towards customized 8051 MCU [12]. The features of modern security in-built Azure Sphere MCU with connectivity support to access to the cloud were discussed in Kazemi et al. [13]. The software executing on the MCU will be secured from side-channel attacks through a proper evaluation performed in the design phase [14].

Related Works and Motivation: *T. Adegbija et al*. presented the microarchitectural characteristics to perform edge computing [15]. The idle energy and overall energy consumption impact of matrixTrans_128 and crc_large application benchmarks were analyzed using four different CPU cores. The in-depth survey of commercial IoT devices considering their key features such as memory, ports, processing abilities, connectivity interfaces, and security support are discussed comprehensively in Ojo et al. [16]. The execution of virtual peripherals using dynamically scalable threads and system hyper pipelining process that reduces the processor unpredicted runtime [17]. P. Kansakar et al. have proposed an efficient approach for selecting the architectural configuration of microprocessors. The microarchitecture configurations are validated by performing the design space exploration for determining low-power IoT processors [18]. The SoC selection for varying IoT applications was discussed extensively in Ramesh et al. [19]. Due to the lack of device scaling for general-purpose processors, the hardware acceleration towards various levels of computing was presented [20]. The existing works are primarily focused on building secure IoT systems using heterogeneous devices for sensing and communication applications. The devices utilized for development of IoT systems are chosen based on the factors considering power, temperature, and connectivity, etc. However, the existing literature are not reported with the feasible approach for optimum application-specific SoC selection. This research gap motivates the authors to carry out high-performance, and security-enabled SoC selection for application-specific design.

To address the security difficulties in resource-constrained devices a lightweight pseudorandom number generator was implemented in Atmel AVR 8-bit and Intel Curie 32-bit SoCs [21]. The IoT devices with the

absence of secure firmware updates may lead to a high possibility of attacks. An 8-bit AVR microcontroller-based implementation of NIST-compliant ECC for enabling secured data transfer in IoT sensor nodes was presented in Xu et al. [22]. The memory utilization and execution time in resource-constrained embedded devices are examined using Cortex-M series microcontroller device [23]. The contribution of the proposed work related to the preceding works is compared and shown in Tab. 1.

Table 1: Overview of proposed work and existing works

Research Contributions	This Paper	Tosiron *et al.* [15]	Mike *et al.* [16]	Tobias *et al.* [17]	Kansakar *et al.* [18]	Ramesh *et al.* [19]
Design Space Analysis	✓	✗	✗	✓	✓	✓
Security Benchmarks	✓	✓	✓	✓	✓	✗
Survey of Crypto-Cores	✓	✗	✗	✓	✗	✗
Performance Analysis	✓	✓	✗	✗	✓	✓
Execution Time Estimation	✓	✓	✗	✗	✗	✓
MCU Application	✓	✓	✓	✓	✓	✓
Optimization	✓	✗	✗	✗	✗	✓
SoC Selection	✓	✗	✗	✗	✓	✓
Security Challenges	✓	✓	✗	✗	✗	✗

✓ Topics Covered ✗ Topics Not Covered

The major contribution of this paper is discussed as follows:

- We presented the various challenges of unsecured IoT devices for the implementation of IoT applications.
- We performed the design space analysis for state-of-the-art crypto algorithms by implementing on an FPGA device. This work also aim to identify the suitable crypto algorithm to enable reliable data communication for the growth of future computing devices.
- We proposed the optimum selection of SoC with inbuilt security using genetic algorithm optimization.
- The final optimum solution is attained through evaluation of execution time and performance by implementing the crypto algorithms in GA resultant hardware.
- We determined the accuracy of securityenabled SoC selection algorithm using the confusion matrix method.

2 Challenges of Unsecured IoT Devices

The developer must enable the device security from the earlier stage to the implementation stage. Incorporating security will strengthen the drivers present within the IoT products. To enable IoT device security the following areas need to be focused on.

Authentication: Any input/output (IO) device attempts system interaction.The identity of IO device must be confirmed in advance to gaining access. The IoT systems are designed to perform end-user authentication involving traditional authentication and machine-to-machine communication (M2M) authentication [24]. Due to memory limitations, devices do not authenticate all types of services. Therefore, the authentication must be sheltered within the devices and must have the capability to update the credentials whenever required.

Encryption and Decryption: The IoT system performs data encryption by sharing the private key with the customers to unlock the data. The data transferred through the IoT device are vulnerable towards attack, which may happen either between the devices, or in the cloud [25]. Therefore, IoT devices need to combine industry-compliant standard encryption libraries to achieve safe and secured data transfer. Though there are various cryptographic functions available, it is better to incorporate complex crypto-core within the hardware device to secure the data.

***Enable secure devices*:** The device should be automatically updated to deliver a secure update mechanism [26]. The device have to receive a cryptography update from the dealer/vendor to empower the security information. As vulnerabilities are commonly found in IoT devices, that becomes a crucial challenge needs to be addressed. Essentially, in most cases, the IoT devices do not support and won't allow the user to execute the update.

Manage and update open-source software: The IoT ecosystem has built-in open-source software platforms, thus the developers should depend on the same software. During the development stage, the developers push towards a blank state while updating those software libraries. It is also essential to create an open-source update for the vulnerable modules. Therefore, during the software update test the combination of modules and then perform an update to avoid the vulnerability.

The IoT needs built-in security: Due to the growing demand for IoT device usage that significantly increase the amount of data they process. The IoT device consists of built-in security using crypto protocols and performs automatic open-source software updates [27]. The devices are loaded with appropriate crypto software interfaces considering the issues identified by the OWASP Top-Ten. The Top-Ten is an industry-standard incline of the utmost prevailing risks to web applications.

3 State-of-the-Art Crypto-cores for Securing IoT Devices

The crypto-core algorithm operates with symmetric and asymmetric algorithms to enable data encryption. The most commonly used block ciphers such as RC5, AES, DES, IDEA, ECC, MX, and SHA algorithms and their functional characteristics are shown in Tab. 2. The contemporary IoT devices are incorporated with SHA-512, ECC crypto-cores to enable on-chip security. The other crypto-cores can be ported within the chip.However, the area consumed by the algorithm is very high. The memory-constrained devices cannot uphold such high space algorithms due to the insufficiency of internal program storage space. All the crypto-cores are synthesized using Xilinx Vivado v.2016.4 software and implemented in Xilinx Zynq-7000 (xc7Zz20clg484-1) FPGA Device. The functional simulation and data path flow of each crypto engine are tested using Verilog scripts. The internal components are simulated using the targeted device supported clock frequency. During the FPGA implementation stage, the common clock frequency is chosen for functional testing with a range of 1 GHz to test the individual crypto-core. The algorithms are tested by sending the data from the master device to the slave device to ensure the proper functioning of encryption and decryption techniques. The PERL code scripts are written for analyzing the sub-modules such as register file, ALU, barrel shifter, address decoding, and control unit. The implementation of crypto-cores demonstrates the efficient resource utilization and memory consumption in a Xilinx Zynq-7000 FPGA Device. The leftover memory resources in the FPGA device can be used for the implementation of additional test bench. The systematic analysis of crypto-cores can enable the modern SoCs to achieve the secured IoT device communication.

From the results attained shown in Tab. 3, the area utilization of AES and SHA-256 crypto-cores were high when compared to the other crypto-cores. As many of the embedded devices have less programmable memory area, it becomes difficult to incorporate such crypto-cores in low-cost devices. As an alternate, designers suggest interfacing external memory for storing such algorithms to enable security. However, external memory interfacing may bring additional complexity, limited performance, and high cost.

The attained efficiency of the algorithms proves that ECC and SHA are highly recommendable for SoC device security when compared to the other algorithms. The logic blocks and slice registers usage for SHA-256 followed by DES and IDEA are higher than that of other crypto-cores that can support secured device-to-device communication. The results can prove that ECC, MX, and RC5 algorithms utilize fewer slice LUTs and registers compared to SHA-256 core. Through efficient FPGA implementation of crypto-cores, the complete resource utilization of crypto-cores in SoC design can be tested.

Table 2: Functional characteristics of various crypto-core architectures

Algorithm	RC5	DES	AES	IDEA	ECC	MX	SHA-256
Key	Symmetric	Symmetric	Symmetric	Symmetric	Asymmetric	Symmetric	Symmetric
Key length	0 to 255 bits	256 bits	128 bits	128 bits	256 default	64 bits	64 bits
Rounds	0 to 255	16	10	8	1	F_i; i = 1 to N	80
Number of keys	2r+2	R	128, 192, 256	52	Variable	64	64 byte
Maximum block size (bits)	32, 64, 128	64	18	64	Variable	64	256
Word Size (bits)	16, 32, 64	16, 32, 64	128	128	256	16, 32, 64	32
Component Structure	Primitive operations	S-Box	Substitution Permutation	Substitution Permutation	Point multiplication	Primitive operations	Hash functions
Speed	Slow	Very slow	Fast	Fast	Fast	Average	Average
Security Strength	Good	Adequate	Excellent	Good	Excellent	Adequate	Good

Table 3: Design space analyzed for crypto-core algorithms

Algorithm	RC5	DES	AES	IDEA	ECC	MX	SHA-256
CLB slices	459	1282	949	1088	201	206	1848
Registers	409	848	331	0	318	0	1630
Exec. time (Sec.)	4	7	5	6	8	3	8
Area Utilized (KB)	18.1	23	73	6.68	30.9	12	42
Throughput (Mbps)	300	380.8	303	424	397	270	402
DFF	281	848	331	0	318	0	1630
Frequency (MHz)	85	47.7	101	98.71	104	25	64.45
Efficiency (Mbps/Slice)	0.63	0.73	0.49	0.56	1.625	0.91	0.84
Delay Time (ns)	21.08	30.60	25.33	21.34	13.65	16.39	22.46

4 SoC Selection Methodology Using Genetic Algorithm Optimization

The number of stages included for system-on-chip selection in the traditional approach involves four stages. The device requirement identification stage to features and cost-based device assignment stage for secured application deployment as shown in Fig. 1. This GA-based approach provides a fixed path for the research community to choose the secured IoT device according to the user requirements. Also within the database of devices several parameters can be added that enables the user to save the device selection time. The optimum algorithm for high-performance, security-enabled, and application-specific SoC selection is illustrated in Algorithm 1.

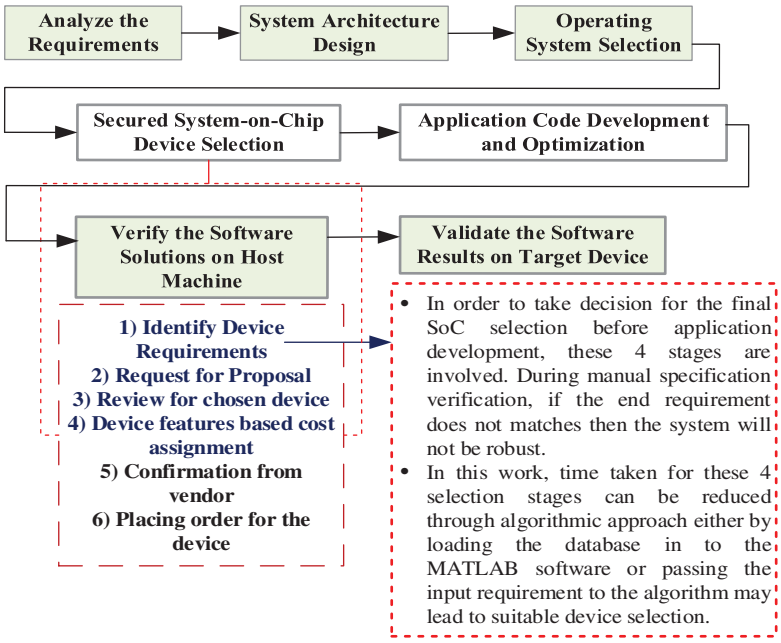

Figure 1: Several stages involved in embedded SoC selection

Algorithm 1: Optimum SoC Selection using Genetic Algorithm

Input: Level1 Parameters (Xi), where X_1 denotes Clock, X_2 Cache, X_3 RAM, X_4 On-Board Networks and X_5 Flash

Input: Level2 Parameter, comparison of security level between the GA resultant SoC devices

Input: Level3 Application-specific SoC selection for heterogeneous applications

Output: An SoC outcome with higher performance, security supported and application-specific device selection

1 Generate initial population

2 Perform selection operation to estimate available chromosome;

3 Initialize gen=0;

4 **while** (gen=0) **do**

5 Randomly choose the parent chromosome;

Algorithm 1 (continued).

```
Generate off-springs by applying uniform crossover;
if Fitness >MaxFitnessHist(gen+1) then
    [MaxFitnessHist(1,gen+1), MaxIndex]=Max(FitnessValues);
    AverageFitnessHist(1,gen+1)=mean(FitnessValues);
end
if Fitness=MaxFitnessHist(gen+1) then
    BestIndiv=pop(MaxIndex, :);
end
end
Select the Maximum Fitness Values as Final Optimum Solution
```

The objective function is written with three different levels. The level-1 is coded to identify the device that delivers maximum performance. The level-2 function is programmed to find the right device that offers an enhanced security solution. The level-3 function is developed to choose the IoT device suitable for heterogeneous applications. To achieve a better selection, the generation is restricted to 30. It is experimental that when generation reaches 30, the best selection is attained just one point more than the finest selection when the generation reaches 100. The collection of the optimum parameters for machine learning is more challenging due to improper selection of parameter values. At times machine learning algorithm may results with error values because of noisy data in the input database. To overcome this limitation this paper is proposed with genetic algorithm for optimum selection. The GA optimization operates by considering variables loaded with initial values for selection.

The algorithm consists of several parameters within the dataset, provided all the values assigned to individual parameters may not be the best one. In the case of smaller dataset, the user can directly choose the optimum value for the specific parameter without complex computations involved. While considering the larger dataset, the utilization of an optimization algorithm becomes the simplest way for optimal selection.The optimization is important because corrupted selection of parameter input values of the classifier may yield bad classification accuracy. The optimization problem can be solved using evolutionary algorithms (EA). The fitness-oriented EA is used when there are numerous solutions upon which one is better than another. In such a case, the fitness value related to the individual parameter is analyzed from the fitness function. This fitness value represents the optimal solution. The population-based EAs are used for optimization processes in which existing solutions are inappropriate to generate better results. The set of existing solutions from which the original solutions are to be generated is referred to as the population. The variation-driven EA is used when there is no solution is acceptable within the population conforming to the fitness function from every individual to produce better results. Accordingly, individual solutions will experience several variations to produce new solutions. While the genetic algorithm is the random-based evolutionary algorithm used to find an optimum solution. GA is the simplest algorithm compared to other traditional EAs. Fig. 2 shows the flow diagram of the genetic algorithm for secured SoC selection for IoT applications. In this work, the GA works on population involving computing devices. The population size (pop _size = 29) is the total number of devices considered as dataset solutions. Each solution is defined as an individual device. Every individual solution (device) has a chromosome which is represented as a set of attributes (features) that defines the individual.

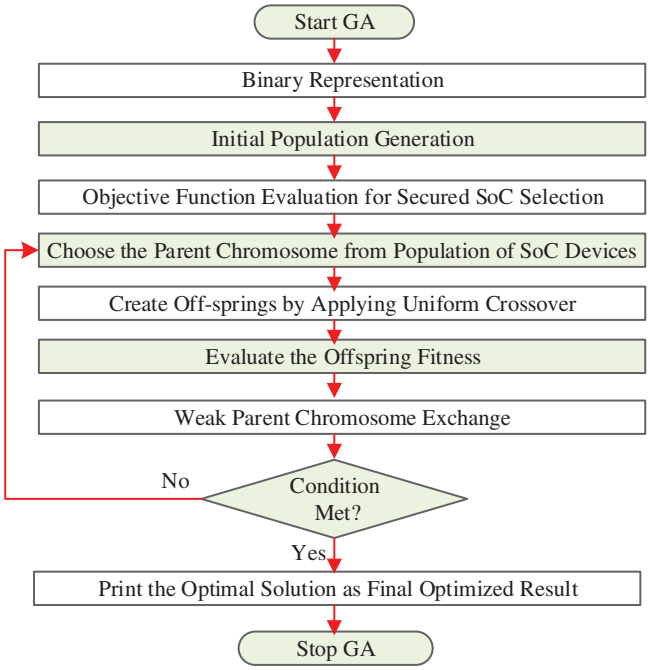

Figure 2: Genetic algorithm flowchart

Every chromosome has associated with a set of genes and it is symbolized as a string of 1's and 0's. Besides, each device has a fitness value. The fitness function is used to select the best device from the individuals. The higher the fitness value the quality of the outcome produced is a better solution. The choice of the best individual device is useful to generate a mating pool. The resultant superior individual device has a greater probability of being selected in the mating pool. The individuals in the mating pool are parents, where two parents from the available mating pool are selected and that will produce two off-springs. The process of selecting and mating with higher-quality individual devices will provide the off-springs with good characteristic properties. As a result, the algorithm end-up with the optimal solution. However, the offspring generated through the particular parents will have the same parent characteristics. In case of any new addition in the device database, the parent features will be present in the newly generated device offspring. To overcome such a problem, certain changes will be made to every added offspring to generate new individuals. The newly created individuals are the recent population that exchanges the formerly used old population. The population produced is named as a generation. The process of exchanging the old population with the new population is called replacement.

The representation and evaluation of chromosome:

The following steps are involved in the process of chromosome representation:

a) There are various chromosome representations available and the selection of appropriate representation depends on the specific problem. The better representation will make the search easier with reduced search space.

b) The chromosome representations include binary, permutation, and value. If the chromosomes are represented as a series of zeros and ones that is named as a binary chromosome. The permutation is used for specific problems such as salesman and order problems. While, in the value chromosome, the real value is encoded directly. Each bit position of the chromosome is called a gene, which has two distinct properties one is value and another one is location. Where the binary representation of the chromosome is called genotype and the chromosome representation of real value is known as phenotype.

c) After the chromosome is represented the appropriate method to work for the search space and then to determine the fitness value of each individual which is known as evaluation. After the attainment of each chromosome representation, the next step is to set the population through the proper selection of individuals from the mating pool.

d) The best individuals are selected based on the earlier estimated fitness value. Later, in the variation operators, the parents are chosen sequentially for mating.

e) The other method for parent selection is the random selection approach. For each set of selected parents, the crossover and mutation operators should be applied. The crossover operator in GA creates a new generation similar to mutation. The crossover process occurs through selecting the chromosome randomly and replacing the genes from one parent. At times, the off-springs come with half genes from one parent and the other half from another parent. However, in GA the genes carried from each parent are random.

f) The picking of chromosomes at a random position and exchanging of genes earlier and subsequent position from its parents takes place in the crossover. The resultant chromosomes are offspring and the operator used is named as a single-point crossover. The process of crossover is essential and without this process, the offspring will be alike its parent.

g) In the mutation operator, for every offspring some genes are chosen and value can be modified. Based on the representation of the chromosome, the mutation varies. However, the developer can decide the way to apply mutation.

h) In the binary encoding process, each gene in the value space will have the value 0 and 1, then the position of each bit value of one or more genes is exchanged. On the other hand, if the gene value originated from a space of real values then the binary mutation cannot be applied.

i) Without the mutation process, the offspring will have properties similar to its parents. The mutation took place to enhance new features to such offspring for the reason that mutation happens randomly. Therefore, it is not compulsory to raise the number of genes to be functional in mutation.

5 Experimentation Results and Security Benchmarks Validation

The increase in complexity of the algorithm may significantly raise the time taken to execute the function due to more number of computations. This might also bring performance issues as more input parameters are increased in computation. Hence, the complexity involved should be kept as low as possible. The shortcoming of this kind of execution is that the genetic algorithm depends on the number of attributes and order in the database. This increases a performance-wise overhead if the parameters are needed to be arranged before traversing.The description of three levels developed using genetic algorithm is discussed below.

1. The Level-1 chromosomes are consist of five attributes to provide the high-end SoC solution for high-performance applications. The level-1 optimization algorithm output shows the result in line with attributes such as clock, level-1 cache, level-2 cache, RAM, onboard networks, and Flash memory. Based on the attribute requirements passed on by the user, the algorithm have resulted with one specific device as a better solution. From the result shown in Fig. 3 the algorithm yields Intel Atom Broxton-M T5700 SoC device that provides high performance after implementation. If the user preferred to add more attributes including cost, power, and temperature range, that can be included easily in the dataset.
2. In the optimization level-2 function that is programmed to choose the device that supports high-security features. Although many devices have minimal security functionality enabled within a chip, comparatively the algorithm resultant device (Cryptocape – ATmega328P) has a trusted platform module with dedicated crypto engines ATSHA204 and ATECC108.

3. The level-3 routine is coded to select the optimum device that is appropriate for varying application necessities. In this level-3, we collected the dataset of various applications supported by each device and passed it as an input to the algorithm. This level-3 is tested to indicate the application-specific SoC in consideration with level-1 and level-2 functions. From the input passed, this level-3 yields with Broadcom BCM 2837B0 SoC as a result for particular application deployment. All three levels of objective functions programs are tested after loading the input database as a CSV file in MATLAB environment. The dataset consists of 12 basic devices, 7 medium range devices, and 7 high-end IoT devices. From this input device list, all three levels are programmed appropriately to obtain the optimum SoC selection.

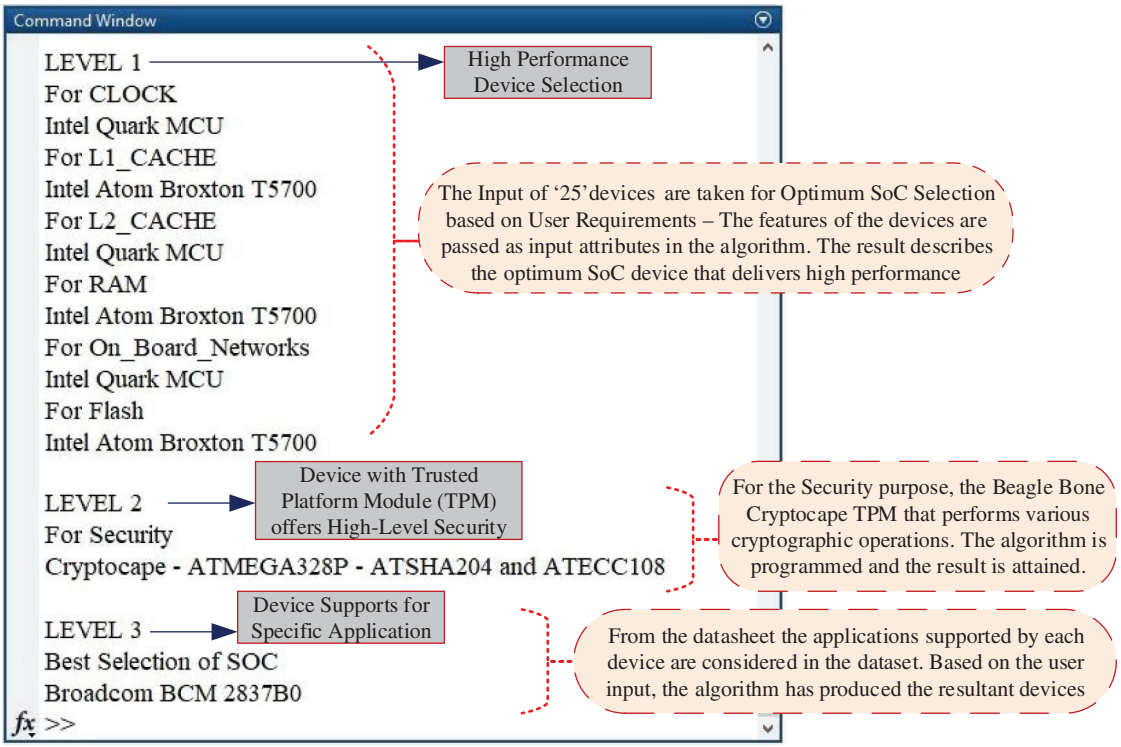

Figure 3: Objective function solution for Level-1, Level-2, and Level-3 parameters

The experimentation for the device dataset carry out 29 times and the resultant SoC is confirmed. The accuracy of the algorithm is calculated using the confusion matrix method (CMM). The CMM produces 10 true positive devices out of 14 medium and high-end devices. The other 12 true negative devices are attained as basic devices. In this work, we take five attributes for high-performance SoC device selection, $X = X_1, X_2, \ldots, X_n$. Where X_1 clock, X_2 level-1 cache, X_3 level-2 cache, X_4 RAM, X_5 on-board networks, and X6 flash memory. While level-2 is programmed directly to select the SoC which provides better security. And for the application-specific SoC selection, we have tabulated the application supported by each device from the datasheet and loaded it in the same database. The $Z = Z_1, Z_2, \ldots, Z_n$ are considered and we formed eleven applications as the input supported by each device in the dataset with the aim of delivering the right device for end-application deployment. The needed input requirements from level-1 such as clock, L1 Cache, L2 cache, RAM, onboard networks, and flash memory are collected. Similarly from level-2, the requirement of security feature is received from the user interms of low, medium, or high. While in level-3 the user adaptable application input is taken. Then

the algorithm is coded to analyze the exact match for the intended applications. The Level-1 result produces the SoC that is suitable for high-performance applications based on the application necessity and requirement of the user. Level-2 produces the SoC that offers a higherlevel of security. Level-3 is coded such that, the algorithm results with SoC that is appropriate for the application-based design. The parameters of the algorithm are selected by analyzing the internal features and characteristics of distinct SoC devices. The objective function of GA for three separate levels yield significant results.

Fig. 4 shows the plot result for the Level-1 function considering the input attributes such as clock, cache, RAM, onboard networks, and flash memory. After passing the input required to the algorithm, the Level-1 function produced the result as Intel Broxton M-T5700. For all three levels, histogram plots are plotted by considering the top five samples of available 29 devices in the database. In the future, the weight factor of the microcontroller chip can be further improved upon taking more input attributes such as thermal, and temperature, etc. Over the process of crossover and mutation, the weight factor has reached maximum if the generation increases. In the Level-1 histogram plot, for the first attribute 'clock' the black color represents the clock weightage of Intel Quark SoC that operates with a clock frequency of 2.13 GHz followed by dark grey color depicts Intel Atom E3845 that operates with a frequency of 1.91 GHz and so on. Correspondingly, for the other core attributes the associated weightage is labeled in the plot. Given Level-2, the parameter comparison is coded that to identify the devices that offer better security. Similar to Level-1, the weightage of secured devices for top samples are plotted in Fig. 5a. The black color represents the device that supports limited security and the white color represents the Cryptocape - ATmega328P with inbuilt (ATSHA204 and ATECC108) SoCs that provide a higher level of security.

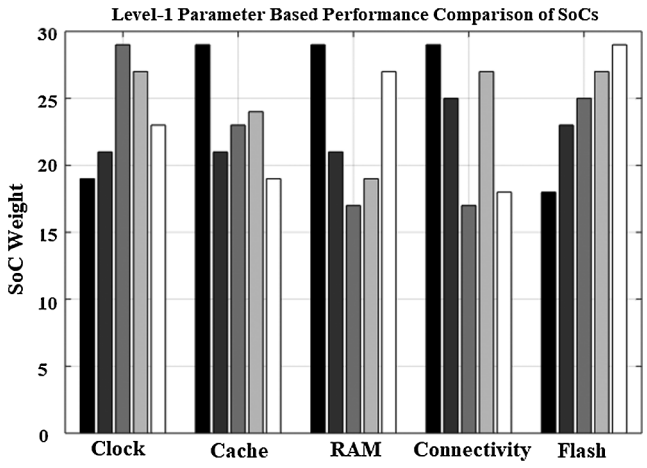

Figure 4: Histogram plot for level-1 high-performance SoC parameters

In the Level-3 function, the dataset of devices that supports for various IoT application deployments are analyzed and passed as an input to the algorithm. Based on the application development need of the user, the input is collected accordingly and the algorithm delivers the resultant device for the appropriate application. Fig. 5b shows the comparison of better application-specific SoC samples from the available dataset. The various security benchmarks are implemented on three hardware devices including Intel Broxton M-T5700, Intel Quark SoC, and ATmega328P hardware devices, and their performance and execution time are examined in Tab. 4. The benchmarks use different sizes of inputs to design real-world applications and experimented using Intel, and Atmel instruction set architectures (ISA).The proposed algorithm results with three separate devices namely Intel Atom Broxton M-T5700 (Device1 – High Performance), Cryptocape — ATmega328P incorporated with ATSHA204, and ATECC108 (Device2 –

Security) and ARM Cortex A53 — Broadcom BCM 2837B0 (Device3 – Varying Applications). All these three devices are considered as base configurations and the security benchmarks are ported on these devices to evaluate the execution time and performance. The device has functioned with its internal clock frequency for the analysis of execution time.

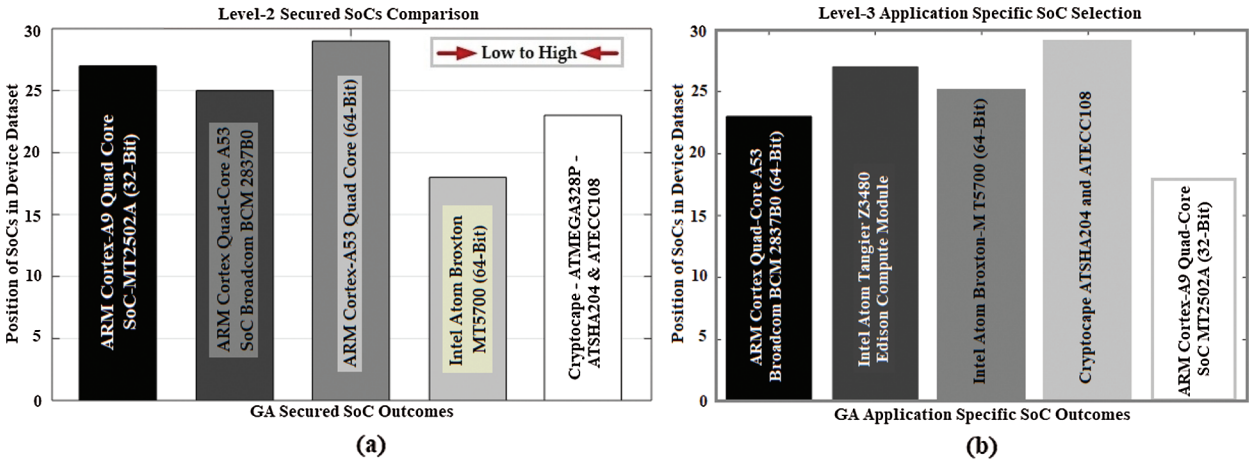

Figure 5: (a) Histogram plot for level-2 secured SoC devices and (b) Histogram plot for heterogeneous application supported SoCs

Table 4: Execution time and performance examined for security benchmarks

Benchmarks	Device 1 (Broadcom – BCM 2837B0)		Device 2 (Intel Atom Broxton M-T5700)		Device 3 (Crypto cape – ATMega328P)		Throughput (Mbps)	Packet Loss Rate (Data Rate = 30 Kbps)
	Execution Time (S)	Performance	Execution Time (S)	Performance	Execution Time (S)	Performance		
AES	0.97	0.46	0.99	0.46	0.86	0.44	59.31	0.37
DES	0.83	0.43	0.87	0.56	0.81	0.39	36.44	0.25
RC5	0.75	0.67	0.79	0.92	0.76	0.63	48.12	0.42
ECC	1.04	0.59	1.05	0.62	1.01	0.57	13.45	0.19
SHA-256	0.75	0.69	0.78	0.91	0.76	0.63	117.85	0.52

The algorithm resulting devices are appropriate for time-critical applications that contain high storage space and have the ability to process the benchmark programs. The benchmarks implementation for examining the runtime and performance are tested using Keil ARM CC compiler for ARM Cortex A53 SoC, and Atmel Studio IDE for ATmega328P SoC. The benchmarks are programmed on the selective resultant SoCs due to their support towards cross compiler and to save MIPS. The benchmarks are coded using a high-level language for hardware testing to analyze the execution time characteristics. The execution time and performance evaluation of devices was shown in Fig. 6. The Device2 executes the benchmarks with the least execution time and leads the performance comparatively than Device1 and Device3. The SoCs utilized for software execution are high-end devices that have many internal features. These devices also can communicate the data securely. From the experiment done, we observed that the security benchmark performances that are relying upon the memory and clock parameters. Also, the CPU performance is estimated by considering instruction count, and clocks per instruction (CPI).

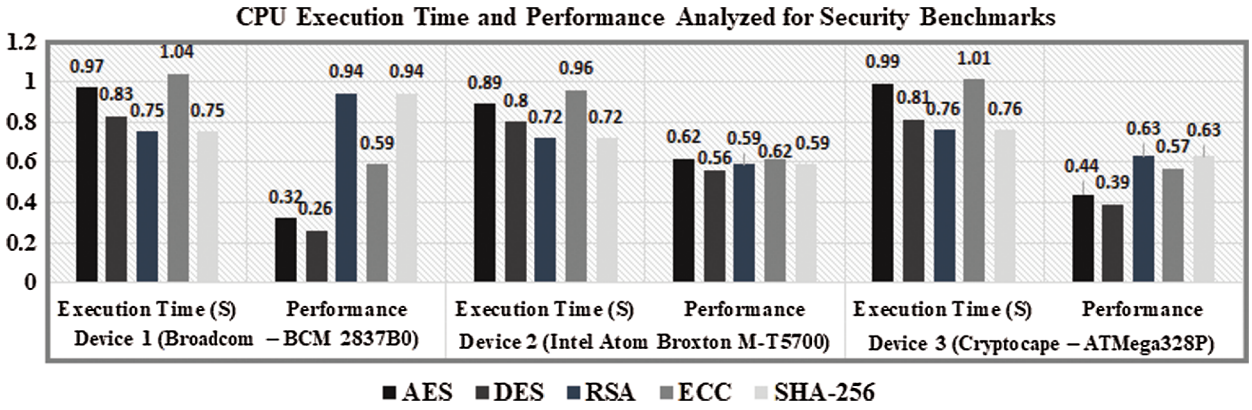

Figure 6: Execut ion time and performance captured of crypto-core benchmarks

The encryption and decryption operations are performed between the devices after loading the benchmark codes. The algorithms are tested according to their characteristics to convert from plain text to cipher text and vice versa. These benchmarks cannot be tested on low-end MCUs because they have very few internal memory resources. In addition, they require code optimization to decrease the application runtime. In this work, the algorithm resultant SoCs used for implementation has a large MIPS range and supports internal code optimization features. These device compilers can perform both inline and multi-file compilation as part of compiler optimization. The devices reduce the external memory access frequency by storing the frequently used data in the cache memory. Besides, without collecting the data and manipulating using registers, performing the calculations on the memory directly can bring high performance. And so efficient usage of cache memory provides an acceptable optimization for SoC devices.

The execution time taken of security benchmarks by providing a common clock source through which, we analyze Device1 and Device2 are taking slightly more execution time compared to Device3. However, the performance of Device3 is lower when matched to Device1 and Device2 this is because of inbuilt memory limitations for processing larger size benchmarks. Device1 used in the validation consists of 16 KB Level-1 cache and 128 KB of Level-2 cache, holds a RAM of size 1 GB and 4 GB external flash space. Device2 contains 4 MB of Level-1 cache, 4 GB of DDR4 memory, and 16 GB of flash memory. The Device3 Cryptocape hardware has 256 KB of EEPROM with a trusted platform module. It supports RSA, AES-128 bit encrypted EEPROM. The ATSHA204 chip present in the hardware module that performs SHA-256 and ECC encryption prototyping on the hardware device. Due to dedicated crypto chips present in the Cryptocape hardware, the execution time of the benchmarks is less when compared to Device1 and Device2. The external memory interfacing required for downloading the benchmarks becomes significant if the internal flash memory is less, but increases the system complexity and execution time. The time taken for fetching the operational code from the external memory in low flash memory devices will take much time for execution. The modern IoT devices are designed with more inbuilt flash ROM memory thereby additional complexity involved in external memory interfacing for downloading the code will be avoided.

Due to buffer overflow, the performance of the device may reduce. The double buffer is used to overcome the buffer size issues and increase the buffer size from 4 KB to 8 KB. The execution time of the CPU also depends upon the effective usage of cache memory with the least access time. The cache memory limitation is one of the primary concerns in the low-end devices as it contains very less cache space. The modern CPUs are designed with very high cache memory. The increase in CPU Level-1 and Level-2 cache memory size limit may also lead to packet loss during memory access. The ineffective usage of CPU registers may push towards increased conversion time from high-level code to machine code due to the level of code optimization in the compiler. The proper usage of code optimization such as

instruction scheduling, register allocation will significantly improve the performance of the embedded system. The internal wiring and flip-flop delays will impact the data transmission speed on the device. The access time of static random access memory (SRAM) and dynamic random access memory (DRAM) would decrease the device performance. Moreover, direct memory access will save a huge amount of time for providing physical memory access to peripheral devices. The processor can process several interrupts. The devices that use the vector interrupts are given with an interrupt vector. The response time for processing the interrupt will reduce the overall system performance.

The estimation of accuracy for the proposed algorithm is carried out using IBM Watson Studio in Python scripts. The device's input database is loaded in CSV format, in which the algorithm fetches the input data to perform multi-class classification. After the classification, the input required from the user is passed thereby the algorithm produces the accuracy. The output of classified elements is gathered and correlated to determine the algorithm accuracy. The algorithm accuracy is estimated using the confusion matrix method to predict the true positive device that offers highlevel of security. The true negative results predicted by the algorithm cannot offer security features. If the algorithm predicts the positive device outcome incorrectly that is defined as false positive. And, when the model falsely predicts the negative device which is named as false-negative devices. The classification metrics have a predictive class (secured positive devices) and a non-predictive class of devices (unsecured) following the dataset. The dataset contains 29 devices for accuracy estimation, in which, the algorithm predicts 17 devices as true positive (highly secured devices) and 9 devices as true negative non-predicted (unsecured devices). In the remaining devices, three devices are predicted as false positive and three devices are predicted as false-negative devices. The accuracy prediction is defined as the ratio of the total number of true predictions. The algorithm produces 89.66% of devices that are accurately categorized by the classifier.

$$\text{Accuracy} = (\text{TP} + \text{TN}) / (\text{TP} + \text{TN} + \text{FP} + \text{FN}) = (17 + 9) / (17 + 9 + 2 + 1) = 89.66\% \quad (1)$$

This analysis will provide the support to identify the secured SoC configuration for the implementation of the IoT system. The objective function is developed such that to find the solution with a higher fitness level. In case, if the objective function is coded without noticing the actual data, then the algorithm results with wrong solution. Besides the population size, crossover, and mutation the attributes must the selected precisely considering the actual requirements. The GA can support for finding the optimum solution for many such problems, the greedy search methods are even used to find the same. However, the GA is appropriate to produce more accurate results for a large dataset. The greedy search techniques are easier and find the optimal solution with the least computational time. Another major difficulty is that, if the number of attributes for comparison of the population exceeds, then the algorithm search space relate to the database will be increased, which may bring low performance. Before enabling the device connectivity, the system designer must follow industry standards and regulations to provide reliability and security. In some applications, the device selection decision is made too early due to practical constraints such as cost and time to market, etc. The decision must be left to the designer for the selection of secured hardware that may overcome certain trade-offs in security, robustness, performance, and cost.

6 Conclusion

This paper discusses the security challenges of traditional MCUs that are vulnerable to attacks. We provided with design space analysis of existing security protocols through FPGA implementation methodology. We also developed the algorithm for secured SoC selection using a genetic optimization tool kit. Finally, the accuracy of the algorithm is evaluated using the confusion matrix method. The major limitation of GA is that upon finding the solution for multi-objective problems it becomes more complex. For some specific problems, GA produces multiple outcomes when the dataset for the fitness function remains similar. In that case, it becomes necessary to validate the results to find the optimum solution

from the resulted multiple outcomes. In the future, more devices can be included in the dataset to perform decision-making comparisons and optimal solution findings using machine learning algorithms.

Acknowledgement: This work was supported by Visvesvaraya PhD Scheme, Ministry of Electronics and Information Technology (MeitY), Government of India. MEITY-PHD-2822.

Funding Statement: The authors received no specific funding for this study.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding this study.

References

[1] S. Chelloug and M. El-Zawawy, "Middleware for internet of things: Survey and challenges," *Intelligent Automation & Soft Computing*, vol. 24, no. 2, pp. 309–318, 2018.

[2] Y. Lisun, C. Choi, V. Sharma, I. Woungang and B. Bhargava, "Guest editorial: Advances in security and privacy technologies for forthcoming smart systems, services, computing, and networks," *Intelligent Automation & Soft Computing*, vol. 25, no. 1, pp. 117–119, 2019.

[3] P. Anghelescu, "Parallel optimization of program instructions using genetic algorithms," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3293–3310, 2021.

[4] B. Bordel, R. Alcarria, D. Martin and A. Sanchez, "Trust provision in the internet of things using transversal blockchain networks," *Intelligent Automation & Soft Computing*, vol. 25, no. 1, pp. 155–170, 2018.

[5] A. Ramadan, R. W. Aboshosha, B. Yadav, K. M. Alseadoon, I. J. Kashout *et al.,* "LBC-IoT: Lightweight block cipher for IoT constraint devices," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3563–3579, 2021.

[6] M. Abu Zaid, O. A. Tawfeek and S. Alanazi, "Applying and comparison of chaotic-based permutation algorithms for audio encryption," *Computers, Materials & Continua*, vol. 67, no. 3, pp. 3161–3176, 2021.

[7] M. Jalil Piran, S. G. Verma, V. Menon and D. Young Suh, "Energy-efficient transmission range optimization model for WSN-based internet of things," *Computers Materials & Continua*, vol. 67, no. 3, pp. 2989–3007, 2021.

[8] I. Qadeer and M. Khurram Ehsan, "Improved channel reciprocity for secure communication in next generation wireless systems," *Computers, Materials & Continua*, vol. 67, no. 2, pp. 2619–2630, 2021.

[9] C. Gao, L. Luo, Y. Zhang, B. Pearson and X. Fu, "Microcontroller based IoT system firmware security: Case studies," in *IEEE Int.Conf. on Industrial Internet (ICII)*, Orlando, FL, USA, pp. 200–209, 2019.

[10] Z. Du, "Personal data security and supervision in the age of large data," *Intelligent Automation & Soft Computing*, vol. 25, no. 4, pp. 847–853, 2019.

[11] D. Park, M. D. Yin and J. Cho, "Secure microcontroller with on-chip hierarchical code validator for firmware authentication," in *Int. Conf. on IT Convergence and Security (ICITCS)*, Beijing, pp. 1–3, 2014.

[12] D. Stiles, "The hardware security behind azure sphere," *IEEE Micro*, vol. 39, no. 2, pp. 20–28, April 2019.

[13] Z. Kazemi, A. Papadimitriou, D. Hely, M. Fazcli and V. Beroulle, "Hardware security evaluation platform for MCU-based connected devices: Application to healthcare IoT, " in *IEEE 3rd International Verification and Security Workshop (IVSW)*. Costa Brava, pp. 87–92, 2018.

[14] J. Dudak, G. Gaspar, S. Sedivy, P. Fabo, L. Pepucha *et al.,* "Serial communication protocol with enhanced properties-securing communication layer for smart sensors applications," *IEEE Sensors Journal*, vol. 19, no. 2, pp. 378–390, 2019.

[15] A. Tosiron, A. Rogacs, C. Patel and A. Gordon, "Microprocessor optimizations for the internet of things: A survey," *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol. 37, no. 1, pp. 7–20, 2018.

[16] M. O. Ojo, S. Giordano, G. Procissi and I. N. Seitanidis, "A review of low-end, middle-end, and high-end IoT devices," *IEEE Access*, vol. 6, pp. 70528–70554, 2018.

[17] T. Strauch, "Connecting things to the IoT by using virtual peripherals on a dynamically multithreaded cortex m3," *IEEE Trans. Circuits Syst. I, Reg. Papers*, vol. 64, no. 9, pp. 2462–2469, Sep. 2017.

[18] P. Kansakar and A. Munir, "Design space exploration methodology for parameter optimization in multicore processors," *IEEE Trans. Parallel Distrib. Syst.Syst*, vol. 29, no. 1, pp. 2–15, Jan. 2018.

[19] K. Ramesh, K. Krishnan, B. Chokkalingam, S. Padmanaban, Z. Leonowicz *et al.,* "Systematic approach for state-of-the-art architectures and system-on-chip selection for heterogeneous IoT applications," *IEEE Access*, vol. 9, pp. 25594–25622, 2021.

[20] M. Kim and Y. S. Shao, "Hardware acceleration," *IEEE Micro*, vol. 38, no. 6, pp. 6–7, Dec. 2018.

[21] B. Le Nguyen, E. Laxmi Lydia, M. Elhoseny, I. A. Pustokhina, D. Pustokhin *et al.,* "Privacy preserving blockchain technique to achieve secure and reliable sharing of IoT data," *Computers, Materials & Continua*, vol. 65, no. 1, pp. 87–107, 2020.

[22] L. Xu, C. Xu, Z. Liu, Y. Wang and J. Wang, "Enabling comparable search over encrypted data for IoT with privacy-preserving," *Computers, Materials & Continua*, vol. 60, no. 2, pp. 675–690, 2019.

[23] N. Ahmed, M. Abu Talib and Q. Nasir, "Secure attestation of program execution and program memory for IoT applications," *Computers, Materials & Continua*, vol. 67, no. 1, pp. 23–49, 2021.

[24] S. Hsiao and W. Sung, "Realization of IoT integration system of LED based on particle swarm optimization," *Intelligent Automation & Soft Computing*, vol. 27, no. 2, pp. 499–517, 2021.

[25] J. Sheu, I. Chen and Y. Liao, "Realization of internet of things smart appliances," *Intelligent Automation & Soft Computing*, vol. 25, no. 2, pp. 395–404, 2019.

[26] H. Rosilah, P. Selver, S. Muzafer, A. Khaleel and T. Milan, "A novel approach to data encryption based on matrix computations," *Computers, Materials & Continua*, vol. 66, no. 2, pp. 1139–1153, August 2020.

[27] Chung Le Van and D. Srinath, "Memetic optimization with cryptographic encryption for secure medical data transmission in IoT-based distributed systems," *Computers, Materials & Continua*, vol. 66, no. 2, pp. 1577–1594, September 2020.