Future Generation Computer Systems 141 (2023) 154–172

Contents lists available at ScienceDirect

Future Generation Computer Systems

journal homepage: www.elsevier.com/locate/fgcs

An extended analytical framework for heterogeneous implementations of light cryptographic algorithms

Issam W. Damaj [a,*], Hadi Al-Mubasher [b], Mahmoud Saadeh [b]

[a] Department of Applied Computing and Engineering, Cardiff School of Technologies, Cardiff Metropolitan University, Cardiff, United Kingdom
[b] Department of Electrical and Computer Engineering, Faculty of Engineering, Beirut Arab University, Debbieh, Lebanon

ARTICLE INFO

Article history:
Received 27 September 2021
Received in revised form 1 November 2022
Accepted 5 November 2022
Available online 18 November 2022

Keywords:
Cryptography
Algorithms
Heterogeneous computing
Decision making
Performance evaluation
Classification

ABSTRACT

The increased need for data, combined with the emergence of powerful Internet of Things (IoT) devices, has resulted in major security concerns. The decision-making related to choosing an adequate cryptographic algorithm to use is, indeed, an example concern that affects the performance of an implementation. Lightweight or tiny ciphers are considered to be the go-to algorithms when talking about embedded systems and IoT devices. Such ciphers, when properly integrated, are expected to have a minimal effect on the overall device utilization and thus provide effective performance. In this paper, we propose a unified analytical framework for lightweight ciphers as implemented within heterogeneous computing environments. This framework considers a carefully identified set of metrics that can adequately enable the capturing, ranking, and classifying the attained performance. To that end, a designer can make effective evaluations and exact adjustments to an implementation. This framework uses three decision-making approaches, namely the Technique for Order of Preference by Similarity to Ideal Solution (TOPSIS), Preference Ranking Organization Method for Enrichment Evaluation (PROMETHEE) II, and Fuzzy TOPSIS. Such approaches take into account both hardware and software metrics when deciding on a suitable cryptographic algorithm to adopt. Validation entails a thorough examination and evaluation of several performance classification schemes. The results confirm that the framework is both valid and effective.

1. Introduction

Algorithms nowadays offer a large choice of efficient implementation possibilities due to the advances in high-performance computing and IoT. Multi-core processors, Graphical Processing Units (GPUs), and high-end programmable devices, like Field Programmable Gate Arrays (FPGAs), can be easily integrated in today's computers. A plethora of co-design tools that facilitate implementations in hardware and software [1,2], support the multiplicity of processing possibilities. Furthermore, according to [3], the number of IoT devices is expected to reach 75.44 billion by the end of 2025. Because of the vulnerabilities addressed by these devices in the operating environment, the IoT security has always been a difficult problem. However, due to their multi-application features, these devices are rapidly rising in number [4]. Nonetheless, there are various questions about which algorithm is appropriate for a particular implementation option or for installation on an embedded device, and vice versa. What effect might a hybrid processing system have on an algorithm, and how would a performance evaluation based on disparate performance measures be made? How can lightweight cryptography deal with the security and privacy concerns that may face devices because of their limited processing power and tight resource constraints [5]?

Measures, metrics, and indicators are at the heart of any performance evaluation. Indicators are variables, qualitative, or quantitative factors that could provide a suitable way to measure success. A qualitative performance indicator is a descriptive feature, a viewpoint, a value, or a characteristic. On the other hand, a numerical measurement based on counting, adding, averaging, or other calculations is referred to as a quantitative performance indicator. To create measuring frameworks and benchmarks, qualitative and quantitative measures can be integrated [6]. There is a quite big number of analytical frameworks in the literature [4,7,8], including those developed in [6,9–11]. However, only a small amount of research has been done on establishing analytical frameworks for classifying and ranking lightweight cryptographic algorithms based on their heterogeneous qualities. Furthermore, there are few, if any, studies that integrate qualitative and quantitative metrics in a holistic manner to aid multidimensional evaluation of lightweight cipher performance. Furthermore, in lightweight cryptography, no assessment has been published to

* Corresponding author.
E-mail address: IDamaj@cardiffmet.ac.uk (I.W. Damaj).

https://doi.org/10.1016/j.future.2022.11.007

categorize algorithms entirely on a basis that covers several degrees of performance, such as Ultralight, Light, Moderately Light, or Not Light.

For that purpose, we present the Extended Lightness Indicator System (ELIS), which is based on the Lightness Indicator System (LIS) in [6] and aims to capture the different characteristics of lightweight cryptographic algorithms and allow for basic performance evaluations. Throughput, Execution Time, Clock Cycles per Instruction (CPI), Algorithmic Complexity, Power Consumption, and other "atomic" properties are captured by the suggested analytical framework's qualitative and quantitative basic indicators. Furthermore, the suggested approach provides a collection of Combined Measurement Indicators (CMIs) which integrate a number of basic indicators; the suggested CMIs include the Lightness Indicator (LI) as the primary indicator that combines the most metrics. The proposed framework makes use of geometric mean of ratios (statistical) and Multiple-Criteria Decision Makings (MCDMs) methods at the same time while capturing the heterogeneous implementations of lightweight cryptographic algorithms, which is one of the contributions presented in the paper. Embedded system designers and developers, smart city planners, network administrators, in addition to database administrators, can make use of this framework while choosing the suitable algorithm that secures the data in their systems.

The following is a breakdown of the paper's structure. Section 2 discusses related work with an emphasis on the framework that was adopted. The rationale for establishing the framework, as well as the ensuing research aims, are summarized in Section 3. The analytical framework, which includes the performance indicators and their mathematical formulation, is described in Section 4. Section 5 contains comprehensive results that demonstrate the benefits of the suggested analytical framework, as well as a detailed explanation of the findings. Furthermore, the benefits of the created framework is explored in the perspective of broader cryptography. Section 6 brings the article to a close and suggests areas for further research.

2. Related work

This section shows how analytical frameworks in similar work were built and presents the methods used. Also, this section elaborates on the importance of benchmarks for the performance evaluation of computer systems and algorithms in general, and for cryptographic algorithms in specific. This section is concluded by pointing out how other work on lightweight ciphers carried out the evaluation process, and thus shows the advantage of this work over others.

2.1. Background

The analytical frameworks in [6,9–11] were developed according to the Generic Benchmark Model (GBM) proposed in [6]. The GBM is a six-element model that includes the Goal, Input, Activities, Output, Outcomes, and the desired Performance profile of the analytical framework. The model captured the relations between the resources, execution, mathematical formulation, and outcomes. As per the GBM, the Goal includes the objective of the analytical framework being developed. Also, the Input names the target algorithms, besides the performance metrics. Moreover, the Activities include the algorithms' implementations as well as the obtained results. The Output is the formation of the key indicators and the creation of any rubrics, if needed. The Outcomes are the statistical evaluation conceptualization formed by combining the Outputs. The application of the established framework is referred to as the Performance dimension, where the algorithms are ranked and classified according to the obtained results. Throughout the developed analytical frameworks, a CMI was calculated to aggregate the heterogeneous simple Key Indicators (KIs) inform a single quantity. To calculate the CMIs in [6,9–11], the geometric mean was used as it can use the ratios to derive the central tendency of data values. Because its ratio is the same as the geometric mean of performance ratios, the geometric mean is preferred, implying that the reference implementation's selection is unimportant when comparing the performance of two different implementations. The generic equation of CMIs can be found in Eq. (1):

$$CMI = \sqrt[n]{ratio_1 \times ratio_2 \times \cdots ratio_n} \tag{1}$$

Where $ratio_i = \dfrac{KI_i}{KI_i^{ref}}$

KI_i is the ith KI, and KI_i^{ref} is the reference indicator measurement KI_i

The uniqueness of the analytical framework developed in [9] was interpreted through enabling the qualitative classification of joint scheduling algorithms by defining performance levels and proposing evaluation schemes and charts. The performance levels were defined for all the key indicators, there were four levels on the scale: Low, Somewhat Satisfactory, Satisfactory, and High. As for the evaluation schemes, two evaluation schemes were proposed, the Indicator Status Scheme (ISS), and the Aggregate Numeric Measures (ANM). The ISS comprises the Percentage of Satisfaction (PoS) which is equal to the percentage of KIs that accomplish a rank of satisfactory or above. The ANM scheme was created to comprehend the mean (μ), standard deviation (σ), skewness (ζ), and kurtosis (κ) of the ratios. To enable the qualitative classification of the algorithms using the proposed schemes, a chart for each scheme was developed. The developed charts have a four-level scale: Ineffective, Somewhat Effective, Effective, and Highly Effective. Although the investigations in [6,9–11] adopted statistical formulations, they clearly encouraged exploring the adoption of MCDM and machine learning models. In [6], the authors highlighted the opportunity to further develop benchmarks with a focus on cryptographic algorithms. The authors in [4] presented an approach that uses TOPSIS and Criteria Importance Through Inter criteria (CRITIC) MCDM methods to choose a lightweight cryptographic algorithm for the Internet of Health Things, where choosing an algorithm in such field is challenging due to the nature of wearable devices, sensors, and nodes. The authors suggested capturing more indicators and including fuzzy MCDM methods in the future.

2.2. Benchmarks

According to Merriam-Webster dictionary, a benchmark is "a standardized problem or test that serves as a basis for evaluation or comparison (as of computer system performance)"[12]. A benchmark is done by carrying out a test series (measurements) used to assess performance level/speed of hardware components and/or software programs according to specified standards. It is important to identify exactly what the benchmark is intended to test when comparing benchmark results.

In the context of computer system performance analysis, benchmarking is defined as the process of the measurement and evaluation of computational performance, devices, networking protocols, and networks under specific reference conditions—in comparison to a reference evaluation. The benchmarking procedure seeks to ensure a reasonable comparison of a number of solutions or successive developments of a System Under Test [13]. There are system benchmarks that test the entire system in addition to those that only test specific components of the system, like memory, hard drives, CPU function, network connections, video

cards, and sound cards. The most popular benchmarks are Whetstone [14], LINPAC [15], Dhrystone [16], Standard Performance Evaluation Corporation (SPEC) [17], etc. SPEC benchmarks are designed to provide performance metrics for comparing the workload of computationally intensive tasks on different computer systems. SPEC benchmarks separately target Central Processing Units (CPUs), graphics and workstations, handheld devices, high-performance computing, Java client/server, mail servers, web servers, file systems, power measurements, networking systems, database systems, etc. [17,18]. In addition to system benchmarks, the algorithm benchmarks, software benchmarks, embedded system benchmarks, and cryptography benchmarks have also been developed.

The term *Algorithm Benchmark* usually refers to a system that enables the comparison of the performance of various algorithm implementations under specific environments, such as the used programming language like, C, C++, etc. To assess the strengths and weaknesses of different algorithms, different benchmarks are applied [19,20]. In practices related to computing, the term *Algorithm Benchmark* is interchangeably used with the term *Benchmark*. The performance of algorithms is analyzed by applying different tests that are ranked according to the running time of algorithms [21].

Software benchmarks are usually concerned with systems such as compilers, database management systems, operating systems, software products, etc. [22–25]. Müller et al. [22] proposed a benchmark that enables several optimization techniques and tests their existence in OpenMP compilers. In [23], the authors proposed TPC-W as an e-commerce benchmark; it evaluates servers in a controlled Internet e-commerce environment that mimics the operations of a business-oriented transactional web server. Demeyer et al. [24] proposed benchmarks for comparing various techniques dealing with software evolution. In [25], Daneva presented a holistic view of software products benchmarking.

Embedded system benchmarks have been widely used in the literature. The Embedded Microprocessor Benchmark Consortium (EEMBC) has led in the development of these benchmarks over the past years. EEMBC's advances are aimed at assisting system developers in choosing the best processors, networking appliances, and smartphones/tablets. EEMBC is primarily concerned with the hardware and software of embedded systems. While organizing its benchmark suites, EEMBC focuses on Automotive, Digital Media, Java, Multicore Processors, Networking, Office Automation, Signal Processing, Smartphones/Tablets, and Browsers. [26]. Among the benchmarks designed by EEMBC are AutoBench, BrowsingBench, and AndBench. AutoBench is a set of benchmarks that lets users estimate how well microprocessors and microcontrollers might perform in automotive, industrial, and general-purpose applications. The BrowsingBench benchmark is an industry-accepted and standardized method for evaluating web browser performance. BrowsingBench evaluates the browsing experience on portable gaming devices, smartphones, navigation devices, netbooks, and other systems that have internet access. AndBench is a benchmark that provides a standardized, industry-accepted method to evaluate the performance of the Android platform. MiBench is a free embedded benchmark suite that is commercially representative. MiBench's applications include networking, security, automotive and industrial control, workplace automation, consumer devices, and telecommunications [27].

Cryptography has become essential in our digital life. It permits us to transmit safe transactions online, guarantee the accuracy of programs and services, maintain our data security, and much more. Thus, the speed with which cryptographic operations (encryption, decryption, hashing, and signing) can be done is critical. Cryptography benchmarks are developed to assess the performance of various cryptographic ciphers being implemented under various systems, such as *CPU/GPU* or other processors. These benchmarks measure the performance using some well-known algorithms such as the *Advanced Encryption Standard (AES)* for encryption/decryption and *Secure Hash Algorithm (SHA)* for signing [28]. Results from encryption, decryption, or hashing are measured in *MB/s*. The higher the indexes in the results means finer performance. In the literature, little work was done to develop benchmarks dedicated to cryptography. The benchmarks were mainly intended to analyze the performance of processors using computationally intensive cryptographic algorithms [29–31]. Rukhin et al. in [32] provide a test suite in statistics for generators of random and pseudorandom numbers used for applications in cryptography. The tests are useful in determining whether a generator is suitable for a certain cryptographic application or not. Yue et al. in [33] presented a suite of cryptographic benchmarks that target network processors (NPCryptBench). The benchmark is created to evaluate cryptographic performance on Network Processors; the benchmark focuses on the performance of the processor and memory subsystems.

2.3. Evaluation approaches

The problem of choosing an appropriate lightweight cipher for a specific application is multidimensional and difficult to be completely characterized by simple metrics like throughput, key size, latency, etc. Such approach fails to exhibit the existing interactions between indicators. In this investigation, a limitation in the illustration is addressed by the creation of novel composite indicators.

The majority of research on lightweight ciphers has concentrated on a simple classification metrics subset as shown in Table 1. The objectives, techniques, target device type, indicators, achievements, and open issues of each similar topic are mentioned in Table 1. As an example, in [7], performance analysis was performed in terms of the average processor and memory usage, power consumption, and response time. On the other hand, the approach in [34] aims to help in choosing a suitable lightweight cipher that can be used in Wireless Sensor Networks (WSNs) applications based on the outcomes of simple metrics acquired from a hardware implementation.

Furthermore, choosing the classification metrics can be related to the type of system or environment being addressed. In [36], ciphers were developed and classified to ease resource-limited lightweight ciphers for RCEs. As resource-limited devices have physical constraints in terms of area, memory, and power, the authors tried to ensure sufficient security for such devices by classifying and recommending the best lightweight ciphers which serve that purpose. However, lightweight ciphers were designed specifically for resource-constrained environments, so their solutions, guidance, prescriptions, investigations, and direction may be incompatible with traditional ciphers. Moreover, in [35], a comparison of a variety of ciphers was introduced; classifications and recommendations of the best ciphers were also provided to solve the security problem in IoT resource-constrained devices. Nevertheless, the presented ciphers are facing difficulties in lessening the key size without compromising the security, reducing memory and energy requirements, and improving execution speed—which may pose a threat to their security level from the evolving attacks.

The papers mentioned above, as well as those listed in Table 1, show the impediments in developing a unified analytical framework, which would give a comprehensive and accurate classification of lightweight ciphers for use in various types of

Table 1
A list of similar topics.

Ref.	Objective	Technique	Target Device Type	Indicators	Achievements	Open Issues
[4], 2020	Develop an evaluation framework to address the issues related to the decision making and evaluation of lightweight ciphers.	Statistics; MCDMs	Not mentioned	Chip area, code size, implementation, Random Access Memory (RAM) size, throughput, power consumption, energy, latency, key size	Ranking of ciphers based on MCDM methods results.	Fuzziness is needed for better and more accurate ranking
[35], 2020	Comparison of a variety of ciphers based on different metrics	Statistics	Not mentioned	Chip area, throughput, security, latency, power and energy, hardware/software efficiency, Figure of Merit (FoM)	Provide a variety of lightweight ciphers that can be used in the field of IoT.	New attacks may pose a threat to the level of security provided by the proposed ciphers
[36], 2019	Compare various modern cryptographic solutions employed to secure small-scale communication devices	Statistics	Not mentioned	Area, area/bit, performance efficiency, power and energy, energy/bit, RAM size, clock cycles, throughput, latency, code size, key size, block size	Facilitate resource-limited lightweight ciphers for Resource Constrained Environments (RCEs)	Lightweight ciphers were developed for resource-constrained environments, so their solutions, guidance, prescriptions, investigations, and direction might conflict with conventional ciphers
[8], 2019	Develop a framework for the benchmarking of lightweight block ciphers on a multitude of embedded platforms	Statistics	Micro-controllers	Block size, key size, round key size, number of rounds, RAM size, latency	Evaluation and ranking of lightweight block ciphers based on the FoM.	Using T-table implementation entails a large memory footprint; which worsens the FoM score
[6], 2018	Develop a framework that deals with the hybrid nature of modern computing systems	Statistics	FPGAs; High-end Multi-core Workstation	Algorithmic complexity, key size, number of rounds, block size, execution time, throughput, clock cycle per instruction, cache miss ratio, propagation delay, look-up table, logic register, power consumption	Classification of algorithms based on a combination of the heterogeneous characteristics of their hardware and software implementations	Ranking of the algorithms was not provided after their classification
[7], 2016	Present a performance evaluation analysis of cryptographic algorithms in embedded systems	Statistics	Computer-On-Modules (COMs)	Data size, key size, average time of encryption/decryption, total response time, CPU and memory consumption	Assess the run-time performance of cryptographic algorithms in embedded systems	Assessment was based on simple indicators; thus, the assessment may not be accurate or weak
[34], 2016	Comparison of different ciphers to understand the trade-off between security performance and operational cost	Statistics	Micro-controller; Low-power sensor mote	Key length, number of rounds, block length, RAM consumption, Read-Only Memory (ROM) consumption	Provide recommendations of ciphers on the security and cost levels.	The recommendation of ciphers was based on the results obtained from hardware implementation only; no combined measurement indicator was presented

systems, devices, or applications. Therefore, this paper adheres to an advanced methodology in evaluating lightweight ciphers. The assessment paradigm takes into account both simple metrics and high-order composite metrics at the same time, permitting for the characterization and evaluation of any cryptographic algorithm performance for any type of device or application. Furthermore, this characterization is performed using a unified analytical framework. The different issues shown in Table 1 are addressed in this paper with the help of MCDMs associated with a fuzziness flavor.

2.4. Similar analytical frameworks

The proposed ELIS was developed and inspired by the existing analytical frameworks that were developed according to the GBM proposed in [6]. A framework that assists in examining the joint scheduling algorithms and ranking their effectiveness was proposed in [9]. The mathematical framework presented in [11] enabled the effective analysis and evaluation of different deployments of routing protocols for low-power and lossy networks based on hybrid characteristics. The work presented in [37] introduced the concept of the vehicle as a computational resource and developed a system that allows the vehicles to share their computational resources within the context of smart cities, where such information can be analyzed to improve the Quality of Service (QoS) and Quality of Experience (QoE). The authors in [38] proposed a performance evaluation framework that aims to make it easier for Intelligent Transportation Systems (ITS) designers to choose the suitable Modern Hardware Devices (MHDs) and Machine Learning (ML) technique. Damaj et al. proposed an analytical framework for high-speed hardware Particle Swarm Optimization (PSO) in [10].

The rankings in the aforementioned frameworks were done according to combined heterogeneous indicators, where the geometric mean of indicators in each framework was determined. The uniqueness of the framework presented in [9] was presented in classifying the targeted algorithms in qualitative terms, where shifting from quantitative terms to qualitative terms was done using evaluation schemes and charts. The authors in [9] highlighted the opportunity of including MCDM methods in the evaluation process. The framework proposed in [10] confirms the development of a successful PSO processor that outperforms the existing implementations. The analyses and classifications in [37] were based on aggregate statistics and machine learning techniques.

3. Research objectives

Proposing a unified analytical framework for lightweight cryptographic algorithms is motivated by a number of factors. Cryptographic algorithms are usually quite complex and therefore, choosing a suitable lightweight algorithm for a specific application might not be an easy task. Therefore, the appropriate evaluation of cryptographic algorithms is crucial in determining their suitability for use in the desired work or environment. Moreover, the complexity and the size of the algorithm play the main role in determining its overall performance on a specific hardware or software device or both together (Hardware/Software co-design), or in a system as a whole. Furthermore, the impact of a cryptographic algorithm used in a specific system can be seen using various quantitative metrics like throughput, execution time, propagation delay, power consumption, and other QoS measures. However, it is difficult to see how different parameters or metrics are related to one another. As a result, the difficulty of assessing an algorithm's overall efficiency is totally associated to the absence of an all-encompassing measure of performance.

To the best of our knowledge, there is a limited attempt in the literature to develop a framework that enables objectively evaluating the lightness of cryptographic ciphers based on heterogeneous implementations, and qualitative and quantitative metrics. In [6], the authors laid the foundation of a similar framework, however, using basic statistics and without formulating a system that ranks or classifies implementation as per well-defined levels of lightness. In this sequel paper, we build on the work presented in [6] as the ELIS by introducing significant extensions to their mathematical models and using advanced MCDM techniques like PROMETHEE II, TOPSIS, and Fuzzy TOPSIS to enable accurate rankings, classifications, and accordingly evaluation and recommendation of use. The extended framework is equipped with advanced modern data analysis techniques, like performance levels, schemes, and charts that constitute a self-contained classification model. In summary, the following are our research objectives:

- Develop a unified analytical framework that allows for the classification of different cryptographic ciphers according to their lightness.
- Develop performance levels for a variety of indicators that influence the performance of heterogeneous implementations of cryptographic algorithms.
- Develop different performance schemes and charts, using different formulations, to enable the qualitative classification of cryptographic algorithms and mapping them to specific lightness levels.
- Validate the developed framework through classifying a set of lightweight cryptographic algorithms.
- Discuss the benefits and applicability of the developed framework in the broader scope of cryptography and information security.
- Present a discussion about the usefulness of the proposed framework.

4. The extended lightness indicator system

The GBM model of Damaj and Kasbah [6] is adopted for the development of the proposed unified analytical framework. The GBM is a six-element model which includes the Goal, Input, Activities, Output, Outcomes, and the desired Performance profile of the intended framework of analysis. The paradigm describes the relationships between the investigation's objectives, resource availability, target deployments, numerical interpretations, and the results. The ELIS, on the basis of the pre-mentioned model, is revealed in the following subsections.

4.1. Goal

The Goal of this work is to develop a unified analytical framework that evaluates the lightness of cryptographic ciphers that are believed to be of lightweight in the literature. This work allows for an effective classification, evaluation, and ranking.

4.2. Input

The algorithms under investigation, along with the performance indicators, are specified in the Input. The ELIS captures a collection of cryptographic algorithms designed for low-resource operations. The framework presented in [6] targets the ciphers that are believed to be small, tiny, lightweight, or ultra-lightweight ciphers. The ciphers that have been addressed include Skipjack, 3-Way, XTEA, KATAN, KTANTAN, and Hight.

The exploited supercomputers are the *Altera Stratix-IV* FP-GA, in addition to the *Dell Precision T7500* with its dual quad-core *Xeon* processor and 24 GB of RAM.

Table 2
The complexity analysis indicator rubric.

General	Scale				
Indicator	Logarithmic Low	Logarithmic High	Linear	Almost Quadratic	Higher than Quadratic
Complexity analysis	O(*logn*)	$\omega(logn)$ but better than linear	$\theta(n)$	$O(n^2)$ but worse than linear	$\omega(n^2)$

There are three classifications of identified performance metrics: Hardware Profile (HWP), Algorithmic Profile (AP), and Software Profile (SWP). The HWP captures the hardware specification in terms of throughput, resource utilization, propagation delay, power consumption, etc. The GAP contains algorithmic complexity and the security strength of the cryptographic algorithm. The SWP contains the execution time, throughput, cache analysis, and number of clocks per instruction.

4.3. Activities

The Activities includes software implementations of targeted cryptographic algorithms using *C* programming language and hardware implementations of these algorithms using *VHDL* hardware description language. The used software tools for software and hardware implementations and profiling include *Quartus*, *ModelSim*, and *Intel VTune Amplifier* under *Visual Studio*. The statistical analyses and validations were done under *Microsoft Excel*.

4.4. Output

The Output of the analytical framework includes the measures and indicators. The general algorithmic, hardware, and software profiles are included in the measures. The indicators that belong to the general algorithmic profile capture the algorithm's time complexity as well as its ciphering strength; the key indicators include the following in bold:

- **Algorithm Complexity** (AC): Analysis of asymptotic complexity using Big-O, small-ω, and θ-notations.
- **Cipher Strength**: based on the **Key Size** *(KS)*, **Number of Rounds** *(NR)*, and the text **Block Size** *(BS)*.

The complexity of algorithms is determined by determining the number of resources required to be executed. In order to do that, the asymptotic behavior of these algorithms must be studied. The asymptotic behavior of algorithms classifies them based on their rate of growth as the size of the input increases. The rubric offered in [39] is on the basis of the following conventional complexity analysis classification:

- O($f(n)$): The algorithm demonstrates an asymptotic growth that is the same as the function f(n) but no worse than it.
- $\omega(f(n))$: The algorithm demonstrates an asymptotic growth that is not superior to the function f(n).
- $\theta(f(n))$: The algorithm demonstrates an asymptotic growth that is the same as the function f(n).

Where, n is the input size.

A rubric is defined to make the evaluation of the specified ciphers possible. The points on the scale of the defined rubric include: Logarithmic Low (LL), Logarithmic High (LH), Linear (L), Almost Quadratic (AQ), and Higher than Quadratic (HQ). For example, LL describes the situation in which the hardness of a specific algorithm is no worse asymptotically than log(n), but is possible to match it; such complication might be shown as O(log(n)). Table 2 provides a full overview of the established rubric. In the development of statistical formulations, these qualitative properties are mapped into quantities, where each point on the scale is assigned to a fixed value. As a result, each point on the scale is converted into numerical quantities of 20%, 40%, 60%, 80%, and 100%.

As mentioned in [6], the Cipher Strength is an evaluation of a cryptographic cipher according to many features that could involve Block Size, Key Size, and the Number of Rounds. The key size represents the size of the key used in cryptographic algorithms in bits. The security of the cryptographic algorithms is related to the length of its key. For the targeted cryptographic algorithms, the reliability and validity of the algorithm increases with the length of its key. However, in a broader sense, the relationship between security and key lengths may be handled more delicately.

In addition, block ciphers convert a plain-text block of several bits into a cipher-text block. To guarantee the cryptographic scheme's security, the block size cannot be too small. That is to say, the larger the block size, the stronger the cryptographic algorithm is.

Furthermore, rounds are critical to the cipher strength; the mixes of text being encrypted, permutations, substitutions, and shifts can be managed in a single round. More rounds, in general, result in more confusion and diffusion, and therefore more security. As a matter of fact, indicators such as the Block Size, Key Size, and the Number of Rounds must be thoughtfully identified after taking the involved cryptographic ciphers into consideration. The proposed indicators are unlikely to be applicable to all cryptographic ciphers.

The software profile comprises the following indicators:

- **Execution Time** (*ET*): the amount of time that passes between the start and finish of an activity.
- **Throughput** (*TH*): the quantity of tasks completed during a certain period of time.
- **Clock Cycle per Instruction** (*CPI*): the required amount of clock cycles to carry out every instruction.
- **Cache Miss Ratio** (*CMR*): the proportion of memory access cache misses.

The hardware profile comprises the following indicators:

- **Propagation Delay** (*PD*): the amount of time it takes for a signal to travel, utilizing combinational logic.
- **Look-Up Table** (*LUT*): the amount of customizable hardware mix needed to execute an algorithm. The size of the hardware in an Altera device is determined by the quantity of LUTs introduced. While in other devices, the area can be described in terms of the total number of logic elements, gates, slices, etc.
- **Logic Register** (*LR*): represents the number of logic registers in the system as a whole.
- **Power Consumption** (*PC*): the amount of energy used by the designed hardware in Watts.

4.5. Outcomes

The Outcomes element is the expression of CMIs in terms of the KIs. The *LI* is the primary CMI calculation in the proposed unified analytical framework. Besides the *LI*, other indicators were used in the development of this framework; the Multiple-Criteria Unified Measurement Indicators (MCUMIs), which were derived after applying different MCDMs.

4.5.1. The LI

The Geometric Mean of KI ratios is used in the mathematical formulation of CMIs to provide a comprehensive calculation that captures the performance of cryptographic algorithms in the algorithmic, hardware, and software profile. The equation of the *LI* and that of the aforementioned indicators can be found in [6] and Eqs. (2) through (5):

$$LI = \sqrt[14]{GAP.SWP.HWP} \tag{2}$$

Where *GAP*, *SWP*, and *HWP* are the indicators of the general algorithmic profile, software profile, and hardware profile, respectively.

$$GAP = \frac{AC_{ref}}{AC}.\frac{KS_{ref}}{KS}.\frac{NR_{ref}}{NR}.\frac{BS_{ref}}{BS} \tag{3}$$

$$SWP = \frac{ET_{SWref}}{ET_{SW}}.\frac{TH_{SW}}{TH_{SWref}}.\frac{CPI_{ref}}{CPI}.\frac{CMR_{ref}}{CMR} \tag{4}$$

$$HWP = \frac{ET_{HWref}}{ET_{HW}}.\frac{TH_{HW}}{TH_{HWref}}.\frac{PD_{ref}}{PD}.\frac{LUT_{ref}}{LUT}.\frac{LR_{ref}}{LR}.\frac{PC_{ref}}{PC} \tag{5}$$

As mentioned in [6], the LIS allows the cryptographic algorithm classification based on their lightness. Higher *LI* is influenced by more throughput, more efficient memory performance, smaller size, lower power consumption, lower complexity, and lower resource utilization. The normalization of key indicators was done based on the fact that the *LI* is proportional to the indicators, either directly or inversely. A new reference is used to calculate the *LI* for each cryptographic algorithm, the reference of each indicator is the midpoint of the Satisfactory performance level that corresponds to it.

4.5.2. The MCUMIs

In order to calculate the MCUMIs, three MCDM methods were applied: PROMETHEE II, TOPSIS, and Fuzzy TOPSIS. The lightweight cryptographic ciphers were ranked based on the *Net Outranking Flow*, *Performance Score*, and the *Closeness Coefficient*, which were obtained in the final steps of PROMETHEE II, TOPSIS, and Fuzzy TOPSIS, respectively. It is important to mention that equal weights are given to all indicators in all of the used MCDMs. In general, there are two types of criteria: benefit and cost. The benefit criterion states that a greater value is preferable; the cost criterion states the contrary. The adopted MCUMIs are presented in the following subsections.

4.5.2.1. Net outranking flow. The *Net Outranking Flow* can be defined as the equilibrium between the positive and the negative outranking flows. A high net flow results in a finer alternative. A positive outranking flow ϕ^+(a) indicates how one alternative outranks all others. The higher the ϕ^+(a), the better the alternative is. On the other hand, a negative outranking flow ϕ^-(a) indicates how an alternative is outranked by all the others. The lower ϕ^-(a), the finer the alternative is. The *Net Outranking Flow* is obtained in the final step of the PROMETHEE II MCDM technique. In our unified analytical framework, the *Net Outranking Flow* was calculated for each involved lightweight cryptographic algorithm, the followed steps are similar to those presented in [40] and are as follows:

Step 1: Create a decision matrix D with n-alternatives and m-criteria to ascertain the dimensionality of the problem. In our case, n equals the number of ciphers used in our work, and m equals the number of key indicators.

Because the data in the decision matrix D comes from different sources, normalization must be done before it can be transformed into a dimensionless matrix that allows the various criteria to be compared.

Step 2: Normalize the developed matrix using the maximum method presented in [41]. This method employs the greatest value in the considered set. The formulas in Eqs. (6) and (7) describe profit type and cost type criteria, respectively:

$$r_{ij} = \frac{x_{ij}}{max_j(x_{ij})} \tag{6}$$

$$r_{ij} = 1 - \frac{x_{ij}}{max_j(x_{ij})} \tag{7}$$

Step 3: Determine the deviation by pairwise comparisons, using Eq. (8):

$$Cj(a, b) = Aj(a) - Aj(b) \tag{8}$$

Cj (a, b) indicates the difference between the evaluations of a and b on each criterion

Step 4: Calculate the preference function, Pj (a, b), using Eq. (9):

$$P_j(a, b) = \begin{cases} 0 & if \;\; C_j(a, b) \le 0 \\ C_j(a, b) & if \;\; C_j(a, b) > 0 \end{cases} \tag{9}$$

Step 5: Determine the aggregated preference function π(a, b), also named the multi-criteria preference index, as presented in [40] and Eq. (10):

$$\pi(a, b) = \sum_{j=1}^{k} P(a, b)w_j \tag{10}$$

Where w_j >0 are the weights related to each criterion The symbol π (a, b) shows that the degree of a is preferred to b over all criteria.

π (a, b) $\approx$ 0 $\Rightarrow$ weak preference of a over b.

π (a, b) $\approx$ 1 $\Rightarrow$ strong preference of a over b.

Step 6: Calculate the leaving and entering outranking flows, using Eqs. (11) and (12):

$$\phi^+(a) = \frac{1}{n-1}\sum \pi(a, x) \tag{11}$$

Where the values of $\pi(a, x)$ are the ones per row

$$\phi^-(a) = \frac{1}{n-1}\sum \pi(a, x) \tag{12}$$

Where the values of $\pi(a, x)$ are the ones per column

Step 7: Calculate the *Net Outranking Flows* ϕ(a) using Eq. (13) and rank the alternatives accordingly. The alternatives undergo complete ranking using PROMETHEE II.

$$\phi(a) = \phi^+(a) - \phi^-(a) \tag{13}$$

Thus, all the alternatives are able to be compared according to the values of ϕ(a). The highest values of ϕ(a) denote the most preferred alternatives.

4.5.2.2. Performance score. The *Performance Score* is the relative proximity of an alternative with respect to the positive ideal solution; the higher the *Performance Score*, the better the alternative is. The ideal positive solution is the one that strengthens the benefit criteria while minimizing the cost criteria. The *Performance Score* is obtained in the final step of the TOPSIS MCDM technique. In our unified analytical framework, the *Performance Score* was calculated for each involved lightweight cryptographic algorithm, the followed steps are the same as those presented in [42] and are as follows:

Step 1: Form a decision matrix D of n-alternatives and m-criteria to identify the problem dimensionality. In our case, n equals the number of ciphers used in our work, and m equals the number of key indicators. Because the data in the decision matrix D come from different sources, it must be normalized before it can be transformed into a dimensionless matrix that allows various criteria comparison.

Step 2: Normalize the developed matrix using Eq. (14):

$$r_{ij} = \frac{x_{ij}}{\sqrt{\sum_{i=1}^{n} x_{ij}^2}} \tag{14}$$

With i=1, …,n and j=1, …The normalized decision matrix R obtained indicates the relative rating of the alternatives.

Step 3: Multiply the normalized decision matrix by its assigned weights to get the weighted normalized decision matrix P. The weighted normalized value p_{ij} is calculated using Eq. (15):

$$p_{ij} = w_i . r_{ij} \tag{15}$$

With i=1, …,n and j=1, …,m

Step 4: Determine the positive ideal solutions A^+ (benefits) and negative ideal solutions A^- (costs) using Eqs. (16) and (17):

$$A^+ = (P_1^+, P_2^+, \ldots, P_n^+) \tag{16}$$

$$A^- = (P_1^-, P_2^-, \ldots, P_n^-) \tag{17}$$

Where $P_j^+ = \max(P_{ij})$ and $P_j^- = \min(P_{ij})$

Step 5: Calculate the Euclidean distances from the positive ideal solution A^+ (benefits) and the negative ideal solution A^- (costs) for each alternative A_i using Eqs. (18) and (19):

$$d_i^+ = \sqrt{\sum_{i=1}^{n} (d_{ij}^+)^2} \tag{18}$$

$$d_i^- = \sqrt{\sum_{i=1}^{n} (d_{ij}^-)^2} \tag{19}$$

Where $d_{ij}^+ = p_j^+ - p_{ij}$ and $d_{ij}^- = p_j^- - p_{ij}$

Step 6: Determine the relative closeness ξ_i for each alternative A_i in relation to the positive ideal solution. ξ_i is given by Eq. (20):

$$\xi_i = \frac{d_i^-}{d_i^+ + d_i^-} \tag{20}$$

Step 7: Sort the options in order of preference based on the relative closeness. Alternatives with a greater ξ_i value are the strongest, and as a result, they ought to be chosen since they are closer to the ideal answer.

4.5.2.3. Closeness coefficient. The *Closeness Coefficient* is the relative closeness of an alternative with respect to the ideal solution. It may be calculated using the notion of distance measurements, where a high *Closeness Coefficient* leads to a better alternative. The *Closeness Coefficient* is obtained in the final step of the Fuzzy TOPSIS MCDM technique. In our unified analytical framework, the *Closeness Coefficient* was calculated for each lightweight cryptographic algorithm. First, the numeric value of each KI for each target cipher was substituted by a qualitative term, according to the performance levels table presented in Table 4. Then, each qualitative term was substituted by a fuzzy value, according to the trapezoidal membership function presented in Table 3 and the curve in Fig. 1. The KI value for each cipher becomes in the form (a, b, c, d). Finally, a series of steps were applied as presented in [43], which are the following:

Step 1: Determine the normalized decision matrix for each fuzzy value using Eq. (21):

$$\tilde{r}_{ij} = \begin{cases} \left(\frac{a_{ij}}{d_j^*}, \frac{b_{ij}}{d_j^*}, \frac{c_{ij}}{d_j^*}, \frac{d_{ij}}{d_j^*}\right) & where \;\; d_j^* = max\{d_{ij}\} \qquad (benefit\ criteria) \\ \left(\frac{a_j^-}{d_{ij}}, \frac{a_j^-}{c_{ij}}, \frac{a_j^-}{b_{ij}}, \frac{a_j^-}{a_{ij}}\right) & where \;\; a_j^- = min\{a_{ij}\} \qquad (cost\ criteria) \end{cases} \tag{21}$$

Table 3
The trapezoidal membership function for the proposed performance levels.

Performance level	Membership function			
Low	0	0	15	30
Somewhat satisfactory	25	40	40	55
Satisfactory	45	60	60	75
High	70	85	100	100

Step 2: Calculate the Fuzzy Positive Ideal Solution (FPIS) (A^*) and Fuzzy Negative Ideal Solution (FNIS) (A^-), according to Eqs. (22) and (23):

$$A^* = \left(r_1^*, r_2^*, r_3^*, \ldots, r_n^*\right) \tag{22}$$

Where $r_j^* = \max\{\tilde{r_{ij4}}\}$

$$A^- = \left(r_1^-, r_2^-, r_3^-, \ldots, r_n^-\right) \tag{23}$$

Where $r_j^- = \min\{\tilde{r_{ij4}}\}$

Step 3: Determine the distance from each alternative to FPIS and FNIS using Eqs. (24) through (26):

$$d^* = \sum_{j=1}^{n} d_v(\tilde{r_{ij}}, r_j^*) \tag{24}$$

$$d^- = \sum_{j=1}^{n} d_v(\tilde{r_{ij}}, r_j^-) \tag{25}$$

$$d_v(\tilde{m}, \tilde{n}) = \sqrt{\frac{1}{4}(m_1 - n_1)^2 + (m_2 - n_2)^2 + (m_3 - n_3)^2 + (m_4 - n_4)^2} \tag{26}$$

Step 4: Calculate the *Closeness Coefficient*, according to Eq. (27):

$$CC_i = \frac{d_i^-}{d_i^- + d_i^*} \tag{27}$$

4.6. Performance

The performance analysis provides KIs measurements and allows the calculation of the *LI* and MCUMIs. The results allow sorting, ranking, and classifying the algorithms under study. Using the obtained indicator results, it is simple to sort executions based on the achieved performance.

4.6.1. Performance levels

To make it possible to rank and classify the target cryptographic algorithms, an analytical scheme was developed depending on the recommended levels in Table 4. The ranges that have been suggested are for a four-level performance scale that includes: Low, Somewhat Satisfactory, Satisfactory, and High. For example, the Block Size of a cryptographic algorithm is said to be Satisfactory if it is between 32 bits, inclusive, and 48 bits, inclusive, because the Block Size cannot be too small in order to secure the cryptographic scheme [6]. And According to [44], a block cipher is considered to be lightweight if the block size is 32, 48, or 64. Given that the indicators capture more than one profile, the midpoint of the Satisfactory level for each indicator was calculated and used as a reference in order to make proper comparisons based on the performed normalization.

The defined ranges in Table 4 are explained as the following:

- **Algorithmic Complexity**: An Almost Quadratic complexity, as if it were linked to the number 0.68, achieves a High Performance. As seen in Table 2, the other complexity levels are predicted to produce Somewhat Satisfactory and Low performance.

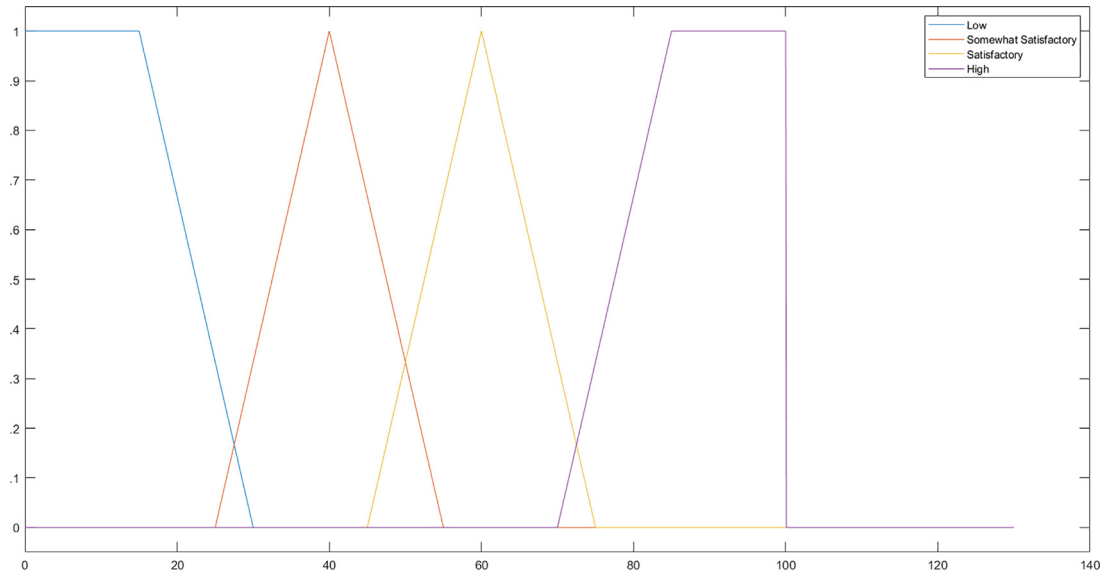

Fig. 1. The Membership Functions of the proposed performance levels.

- **Key Size**: It was shown before that increasing the key size deteriorates the performance of the cryptographic algorithm. And the evaluation in [45] showed that the key size of cryptographic algorithms is usually between 132 and 200 bits. Taking into consideration that we are dealing with cryptographic algorithms that are known as light, the key size of 96 bits is considered to attain a High Performance and a key size of 192 bits is considered to attain a Low Performance.
- **Number of Rounds**: In the world of cryptography, the greater the number of rounds in a cryptographic algorithm, the more complex the cryptographic algorithm is and the more difficult it is to perform cryptanalysis. According to [6], the number of rounds of lightweight cryptographic ciphers is between 32 rounds and 254 rounds. When it comes to lightness, the number of rounds of 32 is considered to attain a High Performance, while a number of rounds of 254 is considered to attain a Low Performance.
- **Block Size**: Having a high block size will decrease the speed of execution of a cryptographic algorithm, and thus decrease its performance. The block size of the target cryptographic algorithms is between 32 bits and 64 bits.
- **Software Execution Time**: To determine the speed of the software implementation of an algorithm, the execution time is measured, the execution time affects the performance, where the less the execution time the better the performance is. The range of the software execution time was determined by averaging the obtained execution times in [6,46]; the execution time in [46] was calculated using the following formula: ET = data bytes * cycles per byte * 1/F.
- **Software Throughput**: When it comes to evaluating the performance of a certain algorithm, the throughput must be taken into consideration. According to [47], the throughput of the lightweight cryptographic algorithms is between 4 and 27 Mbps in IoT applications, where high throughput is needed in such applications.
- **CPI**: According to [48], the number of clock cycles per instruction of the ARM which is a RISC processor is between 0.02 and 3.9 clock cycles, and that of the x86 which is a CISC processor is between 0.02 and 5.2 clock cycles. The processor used in our framework is a CISC processor, also, the performance of cryptographic algorithms in servers and mobile phones is out of our scope, so the included range is between 0.7 and 5.2.
- **Cache Miss Ratio**: The cache miss ratio affects the performance, where less miss ratio leads to better performance, the study done in [49] showed that an optimized cache memory has a cache miss ratio that is between 0.1 and 0.8.
- **Hardware Execution Time**: To determine the speed of the hardware implementation of an algorithm, the execution time is measured, the execution time affects the performance where the less the execution time the better the performance. The survey done in [50] showed that the execution time of cryptographic algorithms is between 0.8 and 3.5 μs.
- **Hardware Throughput**: The throughput affects the performance algorithm, where high throughput can lead to better performance. It was shown in [51] that the hardware throughput of cryptographic algorithms must be between 47 and 320 Mbps in IoT applications, where high throughput is required in such applications.
- **Propagation Delay**: The propagation delay affects the speed of execution of the hardware implementation of an algorithm, and thus affects the performance. It was shown in [52] that the propagation delay for cryptographic algorithms in IoT applications is between 2.18 and 200, where reliability is needed in such applications.
- **LUT**: Huge LUTs will lead to performance issues. According to [46], the number of look up tables of cryptographic algorithms is between 358 and 452 look up tables.
- **Logic Registers**: The number of logic registers may affect the lightness of an algorithm, where a large number of logic registers will make the algorithm more complex. The range of logic registers is the same as that in [6], which is between 167 and 750 logic registers.
- **Power Consumption**: It was shown that the power consumption affects the speed of the processor and thus affects its performance. According to [51], the power consumption of lightweight cryptographic algorithms must be between 225 and 700 mW in IoT applications, where in such applications, the processor speed must not be affected.

4.6.2. Evaluation schemes and charts

Although the ELIS ensures, and based on the different KIs provided, the classification of the algorithms based on their lightness, targeting an achieved performance level necessitates more

Table 4
The proposed key indicator performance levels and the values of the reference measurement.

Profile	Indicator	Low	Somewhat satisfactory	Satisfactory	High	Reference
Algorithmic profile	Algorithmic Complexity	1	0.85	0.68	≤0.51	0.68
	Key Size (bits)	>192	[128, 192]	[96, 128)	<96	112
	Number of Rounds	≥254	64	32	<32	32
	Block Size (bits)	>64	(48, 64]	[32, 48]	<32	40
Software profile	Execution Time (μs)	>750	(435, 750]	[120,435]	<120	277.5
	Throughput (Mbps)	<4	[4, 15.5]	(15.5, 27]	>27	21.25
	Clock Cycle Per Instr.	>5.2	(2.95,5.2]	[0.7,2.95]	<0.7	1.82
	Cache Miss Ratio	≥0.8	(0.45, 0.8)	[0.1, 0.45]	<0.1	0.27
Hardware profile	Execution Time (μs)	>3.5	(2.15, 3.5]	[0.8, 2.15]	<0.8	1.47
	Throughput (Mbps)	< 47	[47, 183.5]	[183.5, 320)	≥320	251.75
	Propagation Delay (ns)	>200	(101,200]	[2,101]	<2	51.5
	Look-Up Table (LUT)	>452	[405,452]	[358, 405)	<358	381.5
	Logic Register	>750	[458.5, 750]	[167, 458.5)	<167	312.75
	Power Consumption (mW)	>700	(462.5, 700]	[225,462.5]	<225	343.75

Table 5
The ANM scheme; the mean, standard deviation, kurtosis, skewness of ratios and LI qualitative interpretations.

Calculation	Levels			
LI	Lightness is lower than the reference Lightness indicator (0, 0.4)	Lightness is somewhat lower than the reference Lightness indicator [0.4, 0.8)	Lightness is similar to the reference Lightness indicator [0.8, 1.4)	Lightness is higher than the reference Lightness indicator ≥1.4
Mean (μ)	Low: <1	Somewhat Satisfactory: [1, 4)	Satisfactory: [4, 7)	High: ≥7
Standard deviation (σ)	Uniform: *Almost no variation in the obtained results* (0,0.3)	Somewhat Disperse: *Small variation in the obtained results* [0.3, 0.6)	Disperse: *Significant variation in the obtained results* [0.6, 0.9)	Highly Disperse: *Large variation in the obtained results* ≥0.9
Kurtosis (κ)	Flat: *Almost no variation in the obtained results* <0	Normal: [0]	Turbulent: *Large variation in the obtained results* > 0	
Skewness (ζ)	Flat: Skewed towards low scores: <0	Normal with no skewness: [0]	Skewed towards high scores > 0	

details about the KIs. In addition to the LIS, an analytical approach that relies on ANM is acquired from [9]. The acquired scheme, in addition to its qualitative interpretations, can be found in Table 5. Further to a single LI score, the scheme includes statistical aggregations of the calculated KI ratios, namely, the mean (μ), standard deviation (σ), skewness (ζ), and kurtosis (κ), which allows the effective ranking and classification of the targeted ciphers. Furthermore, the ANM chart has a four-level scale, namely, Not Light, Somewhat Light, Light, and Ultralight. The description of the chart can be found in Table 6. Accordingly, the lightness of a cryptographic algorithm is reflected by its LI score and statistical aggregations of KI ratios. For example, an algorithm can be classified as Ultralight according to the ANM evaluation chart if the LI score of this algorithm is higher than that of the reference implementation, the mean is high, and for any standard deviation, Kurtosis, and Skewness.

In addition to the ANM scheme, the ISS scheme is acquired from [9]. The ISS comprises the PoS that is equal to the percentage of KIs that achieve a rank of Satisfactory or more. The KI performance levels proposed in Table 4 are used to provide interpretations in qualitative terms. Table 7 summarizes the ISS scheme with its qualitative interpretations. There are four levels to the scale on the ISS chart: Not Light, Somewhat Light, Light, and Ultralight. The chart can be found in Table 8. Accordingly, the lightness of a cryptographic algorithm is reflected by its LI and PoS scores. For example, a cryptographic algorithm can be classified as Light according the ISS evaluation chart if the PoS of this algorithm is high, and the LI score of the algorithm is at most similar to that of the reference implementation.

In addition to the acquired schemes, three MCDM schemes were developed. The schemes are developed based on the LI and the calculated MCUMIs. The schemes can be found in Tables 9–11. The breakpoints were set based on the PoS breakpoints of the ISS scheme, where test vectors were created and classified using the involved MCDM methods to calculate the different MCUMIs. In addition to the MCDM schemes, an MCDM chart is created to enable the qualitative ranking of the target algorithms using the proposed MCDM schemes. The MCDM chart has a scale that is made up of four levels, namely, Not Light, Somewhat Light, Light, and Ultralight. The description of the chart is displayed in Table 12. Accordingly, the lightness of a cryptographic algorithm is reflected by its LI and MCUMI scores. For example, if a certain cryptographic cipher has a high MCUMI like the *Net Outranking Flow*, and its LI score is higher than that of the reference execution, then it is classified as Ultralight as per the PROMETHEE II chart.

5. Analysis and evaluation

Three evaluation approaches are discussed in this study that rely on statistics in association with qualitative terms. The first is

Table 6
The ANM evaluation chart.

Level	Lightness level
Ultralight: The execution of the algorithm attains high lightness level	• The LI score is **higher** than the reference lightness level. • The mean is **high**. • For **any** standard deviation. • For **any** Kurtosis. • For **any** Skewness.
Light: The execution of the algorithm attains satisfactory lightness level	• The LI score is **higher** than the reference lightness level. • The mean is at least **somewhat satisfactory**. • For **any** standard deviation. • For **any** Kurtosis. • For **any** Skewness. OR • The LI score is **similar** to the reference lightness level. • The mean is at least **somewhat satisfactory**. • For **any** standard deviation. • For **any** Kurtosis. • For **any** Skewness.
Somewhat Light: The execution of the algorithm attains somewhat satisfactory lightness level	• The LI score is **somewhat lower** than the reference lightness level. • The mean is at least **somewhat satisfactory**. • For **any** standard deviation. • For **any** Kurtosis. • For **any** Skewness.
Not Light: The execution of the algorithm attains low lightness level	Otherwise.

Table 7
The ISS scheme; the PoS and LI qualitative interpretations.

Calculation	Levels			
LI	Lightness is lower than the reference Lightness indicator $(0, 0.4)$	Lightness is somewhat lower than the reference Lightness indicator $[0.4, 0.8)$	Lightness is similar to the reference Lightness indicator $[0.8, 1.4)$	Lightness is higher than the reference Lightness indicator ≥ 1.4
PoS	Low: $(0, 60\%)$	Somewhat Satisfactory: $[60\%, 70\%)$	Satisfactory: $[70\%, 90\%)$	High: $\geq 90\%$

Table 8
The ISS evaluation chart.

Level	Lightness level
Ultralight: The execution of the algorithm attains high lightness level	• The PoS is **high**. • The LI score is **higher** than the reference lightness level.
Light: The execution of the algorithm attains satisfactory lightness level	• The PoS is **high.** • The LI score is **at most similar** to the reference lightness level. OR • The PoS is **satisfactory.** • The LI score is **at least similar** to the reference lightness level.
Somewhat Light: The execution of the algorithm attains somewhat satisfactory lightness level	• The PoS is **satisfactory.** • The LI score is **at most somewhat lower** than the reference lightness level. OR • The PoS is **somewhat satisfactory.** • The LI score is **at least somewhat lower** than the reference lightness level.
Not Light: The execution of the algorithm attains low lightness level	Otherwise.

Table 9
The PROMETHEE II scheme; the LI and the *Net Outranking Flow* qualitative interpretations.

Calculation	Levels			
LI	Lightness is lower than the reference execution $(0, 0.4)$	Lightness is somewhat lower than the reference execution $[0.4, 0.8)$	Lightness is similar to the reference execution $[0.8, 1.4)$	Lightness is higher than the reference execution ≥ 1.4
Net Outranking Flow	Low: $(-0.36, -0.21)$	Somewhat Satisfactory: $[-0.21, 0.2)$	Satisfactory: $[0.2, 0.36)$	High: ≥ 0.36

Table 10
The TOPSIS scheme; the LI and *Performance Score* qualitative interpretations.

Calculation	Levels			
LI	Lightness is lower than the reference execution (0, 0.4)	Lightness is somewhat lower than the reference execution [0.4, 0.8)	Lightness is similar to the reference execution [0.8, 1.4)	Lightness is higher than the reference execution ≥1.4
Performance Score	Low: (0, 0.45)	Somewhat Satisfactory: [0.45, 0.61)	Satisfactory: [0.61, 1)	High: ≥1

Table 11
The Fuzzy TOPSIS scheme; the LI and *Closeness Coefficient* qualitative interpretations.

Calculation	Levels			
LI	Lightness is lower than the reference execution (0, 0.4)	Lightness is somewhat lower than the reference execution [0.4, 0.8)	Lightness is similar to the reference execution [0.8, 1.4)	Lightness is higher than the reference execution ≥1.4
Closeness Coefficient	Low: (0, 0.5)	Somewhat Satisfactory: 0.5	Satisfactory: (0.5, 1)	High: ≥1

Table 12
The MCDM evaluation chart.

Level	Lightness Level
Ultralight: The execution of the algorithm attains high lightness level	• The MCUMI is **high**. • The LI score is **higher** than the reference execution.
Light: The execution of the algorithm attains satisfactory lightness level	• The MCUMI is **high.** • The LI score is **at most similar** to the reference execution. OR • The MCUMI is **satisfactory.** • The LI score is **at least somewhat lower** than the reference execution. OR • The MCUMI is **somewhat satisfactory.** • The LI score is **at least similar** to the reference execution.
Somewhat Light: The execution of the algorithm attains somewhat satisfactory lightness level	• The MCUMI is **satisfactory.** • The LI score is **lower** than the reference execution. OR • The MCUMI is **somewhat satisfactory.** • The LI score is **at most somewhat lower** than the reference execution. OR • The MCUMI is **low.** • The LI score is **at least somewhat lower** than the reference execution.
Not Light: The execution of the algorithm attains low lightness level	Otherwise.

the ANM scheme, along with its chart, which relies mainly on the mean, standard deviation, kurtosis, and skewness, in addition to the LI in its classification. The second method is the ISS scheme, along with its chart, which relies on the PoS and the LI in its classification. The final method is the MCDMs, which relies on the MCUMIs and the LI. The results confirm that the MCDM method is the best as it enables the classification and ranking of the algorithms with an added accuracy and with identified quality ranks. The findings of this sequel paper's analysis are based on the same simulation parameters provided in [6].

5.1. Results

The ELIS is an application of the proposed unified analytical framework on a number of cryptographic ciphers that have been described in the literature as tiny, lightweight, little, or minute. The *Activities* are carried out in line with the GBM, and after determining the system's *Goal* and *Input*, they are carried out as follows:

1. Use *VHDL* under *Quartus* to do the hardware implementations of the target lightweight ciphers.
2. Use *Quartus* and *ModelSim* to analyze the hardware profile.
3. Use *C* programming language under *Visual Studio* to do the software implementations of the target lightweight ciphers.
4. Use *Intel VTune Amplifier* to analyze the software profile.
5. Calculate and examine the whole algorithmic profile.
6. Use a statistical software program to combine and evaluate the findings from all profiles.
7. Use *Microsoft Excel* to derive and investigate the MCUMIs. Based on the *Activities*, the following steps are then performed:
8. Create the *Output* KIs file.
9. Calculate the *Outcomes*' aggregate indicators.
10. Build the complete *Performance* report.

Tables 13 through 15 detail the development and implementation of the broad algorithmic, software, and hardware characteristics of the cryptographic algorithms under study. As a result, the *Skipjack* algorithm achieved the greatest throughput of software execution with 156.098 Mbps, while the *KTANTAN-48* algorithm attained the best throughput of hardware execution with 480 Mpbs. With 77 *ALUTs* and 167 *LRs*, the *3-Way* algorithm achieved the smallest hardware area (see Table 14).

Table 13
General algorithmic profile.

Algorithm name	AC	Mapped AC	RS	NR	BS
Skipjack	AQ	0.8	80	32	64
XTEA	AQ	0.8	96	64	64
3-Way	AQ	0.8	128	11	96
HIGHT	AQ	0.8	80	32	64
KATAN-32	AQ	0.8	80	254	32
KATAN-48	AQ	0.8	80	254	48
KATAN-64	AQ	0.8	80	254	64
KTANTAN-32	AQ	0.8	80	254	32
KTANTAN-48	AQ	0.8	80	254	48
KTANTAN-64	AQ	0.8	80	254	64
Reference	**AQ**	**0.68**	**112**	**32**	**40**

Table 14
Software profile.

Algorithm name	ET (μs)	TH (Mbps)	CPI	CMR
Skipjack	0.41	156.098	1.327	0.164
XTEA	2.57	24.903	0.729	0.033
3-Way	2.32	41.379	1.107	0.036
HIGHT	8.64	7.407	1.33	0.003
KATAN-32	27.46	1.165	0.634	0.006
KATAN-48	40.33	1.19	0.634	0.004
KATAN-64	52.83	1.211	0.627	0.003
KTANTAN-32	791.08	0.04	0.986	0.001
KTANTAN-48	803.83	0.06	0.975	0.001
KTANTAN-64	821.83	0.078	0.965	0.001
Reference	**277.5**	**21.25**	**1.82**	**0.27**

Table 15
Hardware profile.

Algorithm name	ET (μs)	TH (Mbps)	PD (ns)	ALUT	LR	PC (mW)
Skipjack	7.49	8.55	11.9	554	142	331.01
XTEA	6.18	10.35	11.1	2799	135	322.77
3-Way	0.8	120	3.82	77	167	331.01
HIGHT	1.85	34.59	127.78	2036	72	332.66
KATAN-32	1.47	21.77	43.57	2145	540	328.63
KATAN-48	1.89	25.4	79.94	3982	556	329.95
KATAN-64	2.38	26.89	78.31	4315	572	330.94
KTANTAN-32	0.09	372.09	40.03	1947	112	328.58
KTANTAN-48	0.1	480	72.78	3662	128	329.81
KTANTAN-64	0.15	438.36	79.3	4075	144	331
Reference	**1.47**	**251.75**	**51.5**	**381.5**	**312.75**	**343.75**

Table 16
Indicators.

Algorithm name	*LI*	*Net Outranking Flow*	*Performance Score*	*Closeness Coefficient*
Skipjack	1.53	0.04	0.54	0.37
XTEA	1.19	0.02	0.45	0.26
3-Way	2.46	0.108	0.56	0.66
HIGHT	1.3	0.01	0.43	0.24
KATAN-32	0.87	−0.003	0.42	0.12
KATAN-48	0.77	−0.07	0.37	0.12
KATAN-64	0.74	−0.103	0.35	0.12
KTANTAN-32	1.01	0.04	0.47	0.32
KTANTAN-48	0.92	−0.007	0.45	0.32
KTANTAN-64	0.87	−0.04	0.42	0.32

The LI was derived using Eq. (2) through (5). The *Net Outranking Flow*, *Performance Score*, and *Closeness Coefficient* were derived using the PROMETHEE II, TOPSIS, and Fuzzy TOPSIS MCDM techniques, respectively. The values of these indicators for each target algorithm can be found in Table 16. It can be noticed that the highest indicator values were scored by the *3-Way* algorithm. Figs. 2, 3, 4, and 5 present a per-indicator comparison.

The ANM, ISS, and MCDM schemes were applied, and the results are shown in Figs. 6 and 7, where 0 means that the algorithm is Not Light, 1 means that it is Somewhat Light, 2 means that it is Light, and 3 means that it is Ultralight. The value of each indicator for each targeted lightweight cipher is visualized on a Radar Chart. It is obvious that the *3-Way* algorithm was classified as Ultralight according to the ANM and ISS schemes, while no algorithms were classified as Ultralight according to the MCDM schemes.

Table 17 uses the Difference in Classification (δ_C) measure to quantify the difference between the scale levels of two classification methods. The metric δ_C measures the gap between two classifications, where the difference is classified into Nil, Small, Somewhat Significant, and Significant. The qualitative terms Not Light, Somewhat Light, Light, and Ultralight were replaced by the integer values 0, 1, 2, and 3, respectively and then the Difference in Classification δ_C was calculated. The difference was between the classifications done according to the ISS and those done according to the other schemes, the results are visualized in Fig. 8.

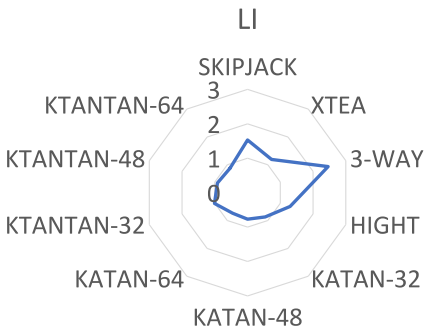

Fig. 2. The *LI* scores.

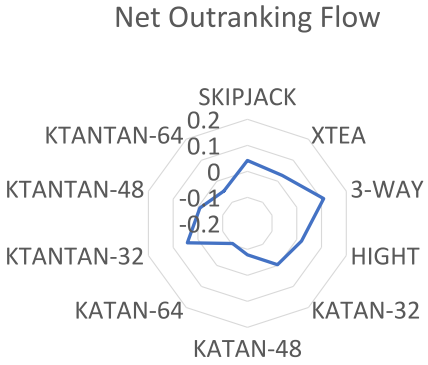

Fig. 3. The *Net Outranking Flow* scores.

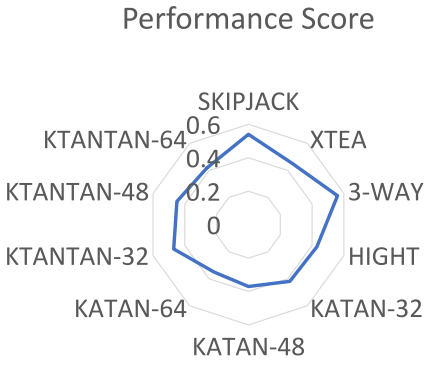

Fig. 4. The *Performance Score* results.

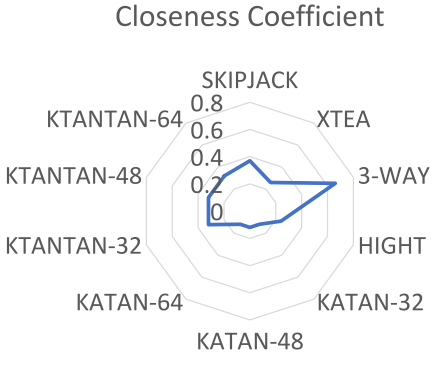

Fig. 5. The *Closeness Coefficient* results.

According to the results, the categorization difference varies from Nil to Small.

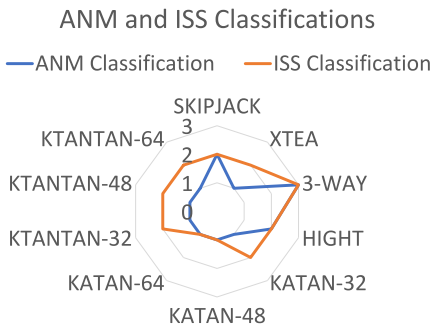

Fig. 6. The ANM and ISS classifications for each target algorithm. The Radar Chart depicts the scale: Not Light (0), Somewhat Light (1), Light (2), and Ultralight (3).

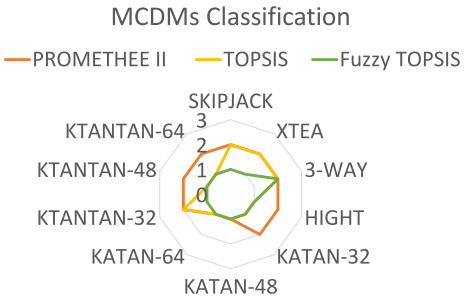

Fig. 7. The MCDMs classification for each target algorithm. The Radar Chart depicts the scale: Not Light (0), Somewhat Light (1), Light (2), and Ultralight (3).

Table 17
The scale and corresponding values of the difference in classification (δ_C) metric.

Rank	(δ_C)
Nil	0
Small	1
Somewhat significant	2
Significant	3

5.2. General evaluation

The current study may be assessed on framework development, application, and contextualization. The developed framework is unified as it combines algorithmic, hardware, and software features. The framework also uses MCDM and fuzzy MCDM techniques to give a set of standardized performance evaluation criteria in addition to a set of helpful performance indicators. The framework fills a void in performance analysis and in handling the heterogeneous aspects of today's computer systems. According to the findings, creating unified indexes/indicators is considered a useful approach since this could reflect certain characteristics in terms of a variety of different critical success factors; the factors or indicators used in this framework include the *LI*, *Net Outranking Flow*, *Performance Score*, and the *Closeness Coefficient*. Indeed, the established unified framework is expandable without requiring changes to the statistical computation or measurement structure, allowing for the addition of new indicators under various profiles.

The developed framework may be utilized at the application level to evaluate the quality of cryptographic algorithms based on their lightness, speed, complexity, and security strength. The developed framework can also help embedded system designers to choose the suitable algorithm because it ranks the cryptographic algorithms based on their hybrid characteristics. Other

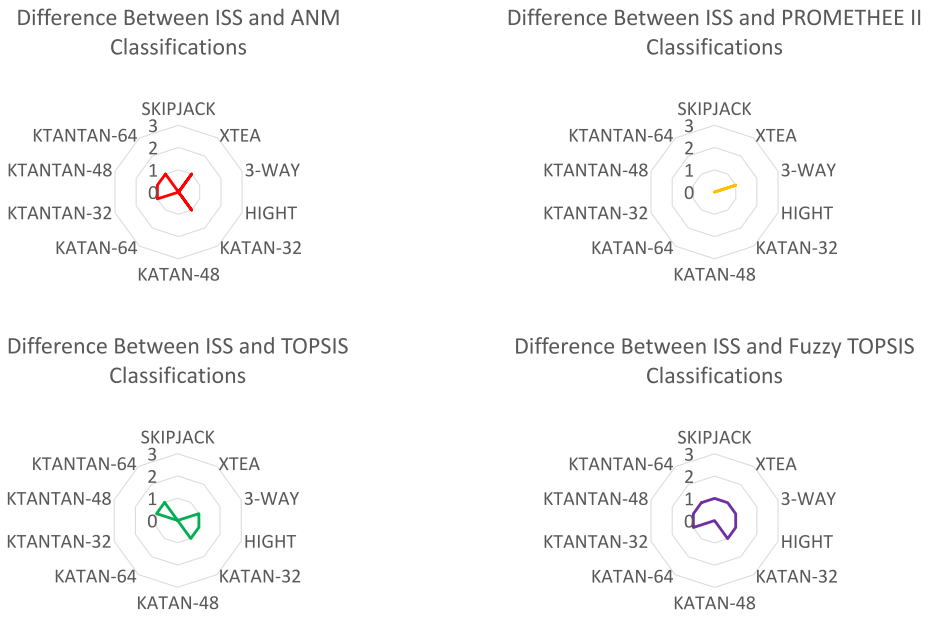

Fig. 8. The Difference in Classification (δ_C) between ISS and other schemes. The Radar Chart depicts the scale: Nil (0), Small (1), Somewhat Significant (2), and Significant (3).

applications, such as signal processing, image processing, and big data analytics, can benefit from the framework since the proposed algorithmic, hardware, and software profiles can be customized per application.

The framework is contextualized with respect to the intended application domain, bringing in a comprehensive and extensive collection of reference KIs. KIs, like the *ET*, *TH*, *CPI*, *CMR*, *PD*, *LUT*, *LR*, and *PC*, are context-free and hence versatile. Other KIs, like *KS*, *NR*, and *BS*, are specific for cryptographic algorithms.

The proposed framework targets the hybrid implementations of cryptographic algorithms, which is limited in the literature. A similar approach was done in [53], where the authors used a strategy to aggregate multiple hardware and software attributes to come up with a single estimate of execution time. With CMIs and MCUMIs, our suggested unified framework would undoubtedly enrich such investigations and support concrete evaluations at an added value.

5.3. Closely related work

The intended contribution in this paper has been proposed as a sequel to the investigation in [6]. Furthermore, a discussion about similar analytical frameworks is presented in Section 2.4. In [4], the authors used statistical aggregations, in addition to different MCDMs to allow for the classification and ranking of the proposed light-weight ciphers. In [8], the authors relied on single indicators and only one CMI in the classification of the lightweight ciphers they have presented. It is clear that the authors did not make any statistical aggregation of indicators or attempted to make ranking based on a qualitative scale. Most evaluations in the literature seek to categorize and rank a collection of suggested lightweight ciphers, as shown in the description of open issues in Table 1, but no classifications were presented.

In the proposed framework, the suggested CMIs, together with their accompanying ANM, ISS, and MCDM schemes, advance the topic towards its next level in terms of performance evaluation. The proposed framework is, indeed, a methodology that enables precise classifications per a concrete cryptographic algorithm lightness scale. Undoubtedly, the proposed framework can lead to automated benchmark tools that can be customized within the application's context. The created framework stresses providing an accurate classification and ranking of lightweight ciphers by proposing different methods of analysis and thus ensuring robust evaluations.

5.4. Limitations and future work

After all the classifications that were carried out, choosing an adequate algorithm to use is no longer confusing. Despite the fact that the obtained Difference in Classification (as shown in Fig. 8) is Small or Nil in most situations, the ANM, ISS, and MCDMs charts support distinct classes and enable different evaluations. When individual indicator scores are examined, it becomes clear that the source of disparities is the individual low scores achieved by multiple indicators. As a result of the low indicator scores, the ANM rankings are affected because of the use of averaging and produced lower scores than that of the ISS. Section 5.1 explains that the 3-Way algorithm achieved the highest score in terms of lightness in all of the classifications, however, KTANTAN-48 and KTANTAN-32 for instance, reflect a higher level of lightness (Light) as shown in the outcome of the ISS classification, while they show the opposite (Somewhat Light) in the outcome of the ANM classification. Moreover, KATAN-32, attains a higher lightness (Light) according to the classification of the PROMETHEE II MCDM, in comparison to the classification results of the TOPSIS MCDM which shows a lower lightness for this cipher(Somewhat Light). Therefore, prioritizing a classification over another based on the application that the lightweight cipher is needed for should have been present in this work.

As mentioned in Section 3, this work is an extension to the existing work proposed in [6], however, significant future work can be carried out. Future work can include experimenting with other statistical formulations or machine learning techniques to produce more accurate classifiers with less computing complexity. Deep learning can be introduced to enable adaptive implementations. Beyond cryptography, the framework described in this study is considered to be highly reusable. Future work can include the development of classifiers for digital signature algorithms. The speed in which digital signature algorithms process data, as well as their ability to validate data integrity, are generally

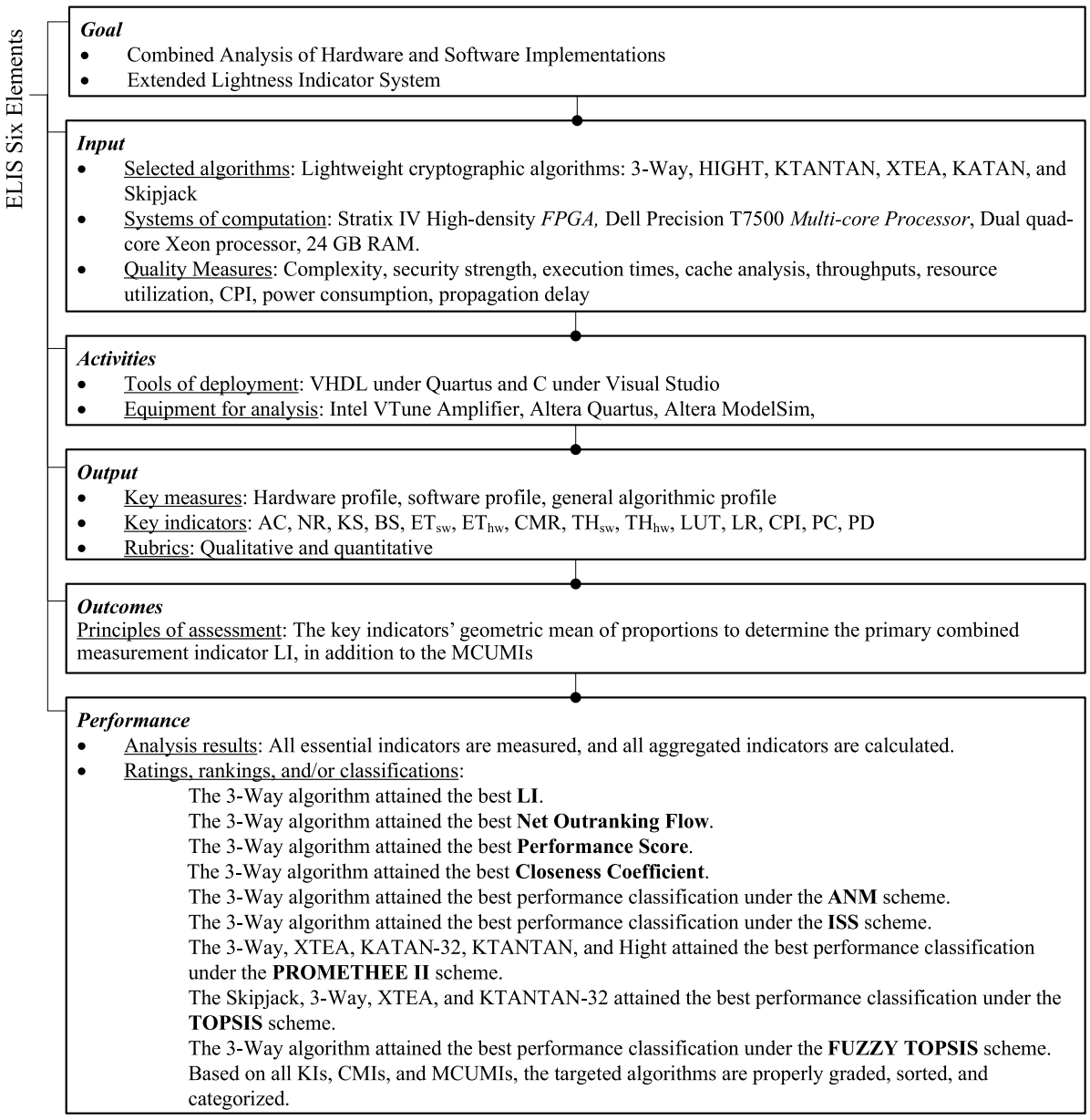

Fig. 9. The six elements diagram of the ELIS analytical framework.

measured in terms of their performance [54]. Future work may include the creation of an Effective Signatures Indicator (ESI) that combines context-specific indicators to help rank and categorize digital signature algorithms based on their effectiveness. The framework can also be expanded to include more indicators like the resistance against side-channel attacks, relevant attack models, and key scheduling. Also, more evaluation charts like the cross-matching evaluation chart and the selection model can be adopted from [38]. Finally, the framework can be integrated into a redistributable software package. A friendly User Interface (UI) can be added so that beneficiaries can easily use it.

6. Conclusion

Modern high-performance computers are hybrids of multi-core processors, *FPGAs*, and *GPUs*. The proposed unified framework seeks to characterize hybrid implementations of lightweight cryptographic algorithms and qualitatively assess their lightness. Processing subsystems are divided into profiles in the framework, each of which may be modified for a given application. The suggested framework was used to generate several essential measures, including lightness, security strength, complexity, speed indicators, net outranking flow, performance score, and closeness coefficient. Moreover, the employed computing devices with high processing power are the multi-core processors and the high-end *FPGAs* for software and hardware implementations, respectively. The framework enjoys being unique where there is no other framework that makes use of geometric mean of ratios, MCDM, fuzzy MCDM, quantitative ranking, and qualitative classification at the same time. With an *LI* of 2.46, *Net Outranking Flow* of 0.108, *Performance Score* of 0.56, and *Closeness Coefficient* of 0.66, the created framework ranks the 3-Way to be the lightest of all the algorithms. The lowest values of *LI* and *Performance Score* were recorded by *KATAN-64* algorithm, having values of 0.74, and 0.35, respectively. The lowest *Net Outranking Flow* was recorded by *KATAN-48* algorithm, having a value of −0.07. The *KATAN-32*, *KATAN-48*, and *KATAN-64* algorithms had

the lowest *Closeness Coefficient*, with a value of 0.12. The overall ELIS framework and its six elements are summarized in Fig. 9. Indeed, the proposed framework is scalable, upgradeable, and can be customized per application. The developed framework can be utilized in environments like Big Data processing and Cloud Computing. The framework aids deciding on a suitable lightweight cipher, where it guarantees that the system is secure without compromising performance. Moreover, the framework can be adopted for lightweight environments, such as IoT devices, where securing communication and data processing among edge devices and the corresponding database, or the server they are connected to, is of critical importance. Future work includes targeting additional processing systems, involving different application areas, integrating the framework within a co-design Integrated Development Environment (IDE), including additional indicators and evaluation charts, integrating the framework into a re-distributable software package, capturing partitioned implementations, and introducing machine learning into the process of classification.

Acronyms

AES	Advanced Encryption Standard
ANM	Aggregate Numeric Measures
CMI	Combined Measurement Indicator
COM	Computer-On-Module
CPI	Clock Cycles per Instruction
CPU	Central Processing Unit
CRITIC	Criteria Importance Through Inter criteria
EEMBC	Embedded Microprocessor Benchmark Consortium
ELIS	Extended Lightness Indicator System
ESI	Effective Signatures Indicator
FNIS	Fuzzy Negative Ideal Solution
FoM	Figure of Merit
FPGA	Field Programmable Gate Array
FPIS	Fuzzy Positive Ideal Solution
GBM	Generic Benchmark Model
GPU	Graphical Processing Unit
IDE	Integrated Development Environment
IoT	Internet of Things
ISS	Indicator Status Scheme
ITS	Intelligent Transportation Systems
KI	Key Indicator
LI	Lightness Indicator
LIS	Lightness Indicator System
MCDM	Multiple-Criteria Decision Making
MCUMI	Multiple-Criteria Unified Measurement Indicator
MHD	Modern Hardware Device
ML	Machine Learning
PoS	Percentage of Satisfaction
PROMETHEE	Preference Ranking Organization Method for Enrichment Evaluation
PSO	Particle Swarm Optimization
QoE	Quality of Experience
QoS	Quality of Service
RAM	Random Access Memory
RCE	Resource Constrained Environment
ROM	Read-Only Memory
SHA	Secure Hash Algorithm
SPEC	Standard Performance Evaluation Corporation
TOPSIS	Technique for Order of Preference by Similarity to Ideal Solution
UI	User Interface
WSN	Wireless Sensor Network

CRediT authorship contribution statement

Issam W. Damaj: Conceptualization, Data curation, Formal analysis, Investigation, Methodology, Project administration, Resources, Supervision, Validation, Writing – original draft, Writing – review & editing. **Hadi Al-Mubasher:** Data curation, Formal analysis, Investigation, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing. **Mahmoud Saadeh:** Data curation, Formal analysis, Investigation, Resources, Software, Validation, Visualization, Writing – original draft, Writing – review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

References

[1] I. Damaj, Parallel algorithms development for programmable logic devices, Adv. Eng. Softw. 37 (2006) 561–582, http://dx.doi.org/10.1016/j.advengsoft.2006.01.009.
[2] S.J. Kasbah, I.W. Damaj, R.A. Haraty, Multi-grid solvers in re-configurable hardware, J. Comput. Appl. Math. 213 (1) (2008) 79–94, http://dx.doi.org/10.1016/j.cam.2006.12.031.
[3] T. Alam, A reliable communication framework and its use in Internet of Things (IoT), 2020, http://dx.doi.org/10.36227/techrxiv.12657158.v1.
[4] L. Ning, Y. Ali, H. Ke, S. Nazir, Z. Huanli, A hybrid MCDM approach of selecting lightweight cryptographic cipher based on ISO and NIST lightweight cryptography security requirements for Internet of Health Things, IEEE Access 8 (2020) 220165–220187, http://dx.doi.org/10.1109/ACCESS.2020.3041327.
[5] R. Almadhoun, M. Kadadha, M. Alhemeiri, M. Alshehhi, K. Salah, A user authentication scheme of IoT devices using blockchain-enabled fog nodes, in: 2018 IEEE/ACS 15th International Conference on Computer Systems and Applications, AICCSA, 2018, pp. 1–8, http://dx.doi.org/10.1109/AICCSA.2018.8612856.
[6] I. Damaj, S. Kasbah, An analysis framework for hardware and software implementations with applications from cryptography, Comput. Electr. Eng. (2018) http://dx.doi.org/10.1016/j.compeleceng.2017.06.008.
[7] N.B. Silva, D.F. Pigatto, P.S. Martins, K.R. Branco, Case studies of performance evaluation of cryptographic algorithms for an embedded system and a general purpose computer, J. Netw. Comput. Appl. 60 (2016) 130–143, http://dx.doi.org/10.1016/j.jnca.2015.10.007, URL https://www.sciencedirect.com/science/article/pii/S1084804515002283.
[8] D. Dinu, Y. Corre, D. Khovratovich, L. Perrin, J. Großschädl, A. Biryukov, Triathlon of lightweight block ciphers for the Internet of Things, J. Cryptogr. Eng. 9 (2019) http://dx.doi.org/10.1007/s13389-018-0193-x.
[9] I.W. Damaj, A.M. El Hajj, H.T. Mouftah, An analytical framework for effective joint scheduling over TDD-based mobile networks, IEEE Access 7 (2019) 144214–144229, http://dx.doi.org/10.1109/ACCESS.2019.2945849.
[10] I. Damaj, M. Elshafei, M. El-Abd, M.E. Aydin, An analytical framework for high-speed hardware particle swarm optimization, Microprocess. Microsyst. 72 (2020) 102949, http://dx.doi.org/10.1016/j.micpro.2019.102949, URL https://www.sciencedirect.com/science/article/pii/S0141933119300407.
[11] I.W. Damaj, W.E. Mardini, H.T. Mouftah, A mathematical framework for effective routing over low-power and lossy networks, Int. J. Commun. Syst. 33 (11) (2020) e4416, http://dx.doi.org/10.1002/dac.4416, URL https://onlinelibrary.wiley.com/doi/abs/10.1002/dac.4416, e4416 dac.4416.
[12] Merriam-Webster, The Merriam-Webster Dictionary, revised ed., Merriam-Webster Mass Market, 2017.
[13] S. Bouckaert, S.C. Phillips, J. Wilander, S. Rehman, W. Dabbous, T. Turletti, Benchmarking Computers and Computer Networks, Whitepaper, IST Fire projects, 2011, URL http://www-sop.inria.fr/members/Thierry.Turletti/WP11.pdf.
[14] H. Curnow, B. Wichman, A synthetic benchmark, Comput. J. 19 (1) (1976) 43–49.
[15] J. Dongarra, P. Luszczek, Encyclopedia of Parallel Computing, Springer US, 2011, pp. 1033–1036.

[16] R.P. Weicker, Dhrystone: A synthetic systems programming benchmark, Commun. ACM 27 (10) (1984) 1013–1030.
[17] J.L. Henning, SPEC CPU2000: Measuring CPU performance in the new millennium, Computer 33 (7) (2000) 28–35.
[18] SPEC, SPEC standard performance evaluation corporation, 2017, Available from http://www.spec.org/index.html.
[19] M.-Y. Chen, J.-D. Wei, J.-H. Huang, D. Lee, Design and applications of an algorithm benchmark system in a computational problem-solving environment, in: ACM SIGCSE Bulletin, Vol. 38, ACM, 2006, pp. 123–127.
[20] O. Mersmann, M. Preuss, H. Trautmann, Benchmarking Evolutionary Algorithms: Towards Exploratory Landscape Analysis, in: PPSN (1), 2010, pp. 73–82.
[21] G.T. Heineman, G. Pollice, S.M. Selkow, Algorithms in a Nutshell - A Desktop Quick Reference.
[22] M.S. Müller, An OpenMP compiler benchmark, Sci. Program. 11 (2) (2003) 125–131.
[23] D.F. García, J. García, TPC-W E-commerce benchmark evaluation, Computer 36 (2) (2003) 42–48.
[24] S. Demeyer, T. Mens, M. Wermelinger, Towards a software evolution benchmark, in: Proceedings of the 4th International Workshop on Principles of Software Evolution, ACM, 2001, pp. 174–177.
[25] M. Daneva, Software benchmark design and use, in: Re-Engineering the Enterprise, Springer, 1995, pp. 20–29.
[26] EEMBC, Website, 2017, URL http://www.eembc.org/.
[27] M.R. Guthaus, J.S. Ringenberg, D. Ernst, T.M. Austin, T. Mudge, R.B. Brown, MiBench: A free, commercially representative embedded benchmark suite, in: Proceedings of the Workload Characterization, 2001. WWC-4. 2001 IEEE International Workshop, WWC '01, IEEE Computer Society, Washington, DC, USA, 2001, pp. 3–14, http://dx.doi.org/10.1109/WWC.2001.15.
[28] A.J. Menezes, P.C. Van Oorschot, S.A. Vanstone, Handbook of Applied Cryptography, CRC Press, 2010.
[29] D.W. Chang, C.D. Jenkins, P.C. Garcia, S.Z. Gilani, P. Aguilera, A. Nagarajan, M.J. Anderson, M.A. Kenny, S.M. Bauer, M.J. Schulte, et al., ERCBench: An open source benchmark suite for embedded and re-configurable computing, in: Field Programmable Logic and Applications (FPL), 2010 International Conference on, IEEE, 2010, pp. 408–413.
[30] J. Babb, M. Frank, V. Lee, E. Waingold, R. Barua, M. Taylor, J. Kim, S. Devabhaktuni, A. Agarwal, The raw benchmark suite: Computation structures for general purpose computing, in: Field-Programmable Custom Computing Machines, 1997. Proceedings., The 5th Annual IEEE Symposium on, IEEE, 1997, pp. 134–143.
[31] Z. Tan, C. Lin, Y. Li, Y. Jiang, Optimization and benchmark of cryptographic algorithms on network processors, in: Systems, Man and Cybernetics, 2003. IEEE International Conference on, Vol. 3, IEEE, 2003, pp. 2296–2301.
[32] A. Rukhin, J. Soto, J. Nechvatal, M. Smid, E. Barker, A Statistical Test Suite for Random and Pseudo-random Number Generators for Cryptographic Applications, Tech. Rep., DTIC Document, 2001.
[33] Y. Yue, C. Lin, Z. Tan, NpCryptBench: A cryptographic benchmark suite for network processors, ACM SIGARCH Comput. Archit. News 34 (1) (2006) 49–56.
[34] K. Biswas, V. Muthukkumarasamy, X.-W. Wu, K. Singh, "Performance evaluation of block ciphers for wireless sensor networks", in: R.K. Choudhary, J.K. Mandal, N. Auluck, H.A. Nagarajaram (Eds.), Advanced Computing and Communication Technologies, Springer Singapore, Singapore, 2016, pp. 443–452.
[35] D. Sumit, B. Singh, P. Jindal, Lightweight cryptography: A solution to secure IoT, Wirel. Pers. Commun. 112 (2020) 1–34, http://dx.doi.org/10.1007/s11277-020-07134-3.
[36] P. Singh, B. Acharya, R. Chaurasiya, A comparative survey on lightweight block ciphers for resource-constrained applications, Int. J. High Perform. Syst. Archit. 8 (2019) 250, http://dx.doi.org/10.1504/IJHPSA.2019.104953.
[37] A.H. Salem, I.W. Damaj, H.T. Mouftah, Vehicle as a Computational Resource: Optimizing Quality of Experience for Connected Vehicles in a Smart City, Veh. Commun. 33 (2022) 100432, http://dx.doi.org/10.1016/j.vehcom.2021.100432, URL https://www.sciencedirect.com/science/article/pii/S2214209621001017.
[38] I. Damaj, S.K. Al Khatib, T. Naous, W. Lawand, Z.Z. Abdelrazzak, H.T. Mouftah, Intelligent transportation systems: A survey on modern hardware devices for the era of machine learning, J. King Saud Univ. - Comput. Inform. Sci. (2021) http://dx.doi.org/10.1016/j.jksuci.2021.07.020, URL https://www.sciencedirect.com/science/article/pii/S1319157821001877.
[39] T.H. Cormen, C.E. Leiserson, R.L. Rivest, C. Stein, Introduction to Algorithms, The MIT Press, 2001.
[40] L. Abdullah, W. Chan, Application of PROMETHEE method for green supplier selection: A comparative result based on preference functions, J. Ind. Eng. Int. 15 (2018) http://dx.doi.org/10.1007/s40092-018-0289-z.
[41] K. Palczewski, W. Sałabun, Influence of various normalization methods in PROMETHEE II: An empirical study on the selection of the airport location, Procedia Comput. Sci. 159 (2019) 2051–2060, http://dx.doi.org/10.1016/j.procs.2019.09.378.
[42] R. Krohling, A. Pacheco, Information technology and quantitative management (ITQM 2015) A-TOPSIS – An approach based on TOPSIS for ranking evolutionary algorithms, in: ITQM, Vol. 55, 2015, pp. 308–317, http://dx.doi.org/10.1016/j.procs.2015.07.054,
[43] B. Sahin, T.L. Yip, P.-H. Tseng, M. Kabak, A. Soylu, An application of a fuzzy TOPSIS multi-criteria decision analysis algorithm for dry bulk carrier selection, Information 11 (5) (2020) http://dx.doi.org/10.3390/info11050251, URL https://www.mdpi.com/2078-2489/11/5/251.
[44] B.J. Mohd, T. Hayajneh, K.M. Ahmad Yousef, Z.A. Khalaf, M.Z.A. Bhuiyan, Hardware design and modeling of lightweight block ciphers for secure communications, Future Gener. Comput. Syst. 83 (2018) 510–521, http://dx.doi.org/10.1016/j.future.2017.03.025, URL https://www.sciencedirect.com/science/article/pii/S0167739X17304661.
[45] j. Awotunde Joseph Bamidele, A. Oloduowo, I. Oladipo, R. Tomori, M. AbdulRaheem, Evaluation of four encryption algorithms for viability, reliability and performance estimation, Niger. J. Technol. Dev. 13 (2017) 74, http://dx.doi.org/10.4314/njtd.v13i2.5.
[46] W. Diehl, F. Farahmand, P. Yalla, J.-P. Kaps, K. Gaj, Comparison of hardware and software implementations of selected lightweight block ciphers, in: 2017 27th International Conference on Field Programmable Logic and Applications, FPL, 2017, pp. 1–4, http://dx.doi.org/10.23919/FPL.2017.8056808.
[47] M. Engineer, A. Shah, Performance analysis of lightweight cryptographic algorithms simulated on arduino UNO and MATLAB using the voice recognition application, in: 2018 International Conference on Circuits and Systems in Digital Enterprise Technology, ICCSDET, 2018.
[48] E. Blem, J. Menon, K. Sankaralingam, A Detailed Analysis of Contemporary ARM and x86 Architectures, University of Wisconsin - Madison, 2013.
[49] C. Waldspurger, T. Saemundsson, I. Ahmad, N. Park, Cache modeling and optimization using miniature simulations, in: USENIX Annual Technical Conference, 2017, pp. 487–498.
[50] M. Mushtaq, S. Jamel, A. Disina, Z. Pindar, N. Shakir, M. Mat Deris, A survey on the cryptographic encryption algorithms, Int. J. Adv. Comput. Sci. Appl. 8 (2017) 333–344, http://dx.doi.org/10.14569/IJACSA.2017.081141.
[51] A. Zaky, E. Elmitwalli, M. Hemeda, Y. Ismail, K. Salah, Ultra low-power encryption/decryption core for lightweight IoT applications, in: 2019 15th International Computer Engineering Conference (ICENCO), 2019, pp. 39–43, http://dx.doi.org/10.1109/ICENCO48310.2019.9027471.
[52] M. Usman, I. Ahmed, I. Aslam, U. Rabab, FPGA implementation of secure Internet of Things (SIT) algorithm for High Throughput Area ratio, Int. J. Future Gener. Commun. Netw. 11 (2018) http://dx.doi.org/10.14257/ijfgcn.2018.11.5.06.
[53] S. Spacey, W. Luk, P. Kelly, D. Kuhn, Rapid design space visualisation through hardware/software partitioning, in: Proceedings - 2009 5th Southern Conference on Programmable Logic, SPL 2009, 2009, pp. 159–164, http://dx.doi.org/10.1109/SPL.2009.4914913.
[54] A. Al Imem, Comparison and evaluation of digital signature schemes employed in NDN network, Int. J. Embedd. Syst. Appl. 5 (2015) http://dx.doi.org/10.5121/ijesa.2015.5202.

Issam Damaj is a Senior Lecturer in Computer Science with the Department of Applied Computing and Engineering, Cardiff School of Technologies, Cardiff Metropolitan University (Cardiff Met), Cardiff, United Kingdom (UK). Before joining Cardiff Met in 2022, he spent 16 years in professorial ranks in higher education institutions in Lebanon (Beirut Arab University, BAU, three years, Kuwait (American University of Kuwait, AUK, ten years) and Oman (Dhofar University, DU, three years). During his tenure, he published more than 100 technical papers and book chapters—in addition to various editorials, short papers, and technical reports. His research interests include hardware design, smart cities, and technical education. During his career, he was assigned a variety of leadership positions in university administration, quality assurance, and accreditation. In 2004, he was awarded a Doctor of Philosophy Degree in Computer Science from London South Bank University, London, UK. He received a master's degree in Computer and Communications Engineering from the American University of Beirut in 2001. In addition, he received a bachelor's degree in Computer Engineering from BAU in 1999. He is an associate editor and a reviewer with publishers that include IEEE, Elsevier, Wiley, and Springer. In addition, he is the recipient of various awards in mentoring, service, research, and academic high distinction. Dr. Damaj is a senior member of the IEEE. He maintains an academic website at www.idamaj.net.

Hadi Al Mubasher is a member of the Intelligent Hardware (iHW) Research Group, Electrical and Computer Engineering (ECE) Department, Beirut Arab University (BAU), Beirut, Lebanon. He received his bachelor's degree in Computer Engineering from BAU in 2021. His research interests include smart cities, performance evaluation, and information security. He is a student member of the IEEE.

Mahmoud Saadeh is a member of the Intelligent Hardware (iHW) Research Group, Electrical and Computer Engineering (ECE) Department, Beirut Arab University (BAU), Beirut, Lebanon. He received his bachelor's degree in Computer Engineering from BAU in 2021. His research interests include smart cities, performance evaluation, and information security. He is a student member of the IEEE.