Message Security Level Integration with IoTES: A Design Dependent Encryption Selection Model for IoT Devices

Matasem Saleh[1†], NZ Jhanjhi[2†], Azween Abdullah[3†] and Raazia Saher[4††]
[1]matasemsaleh@gmail.com, [2]noorzaman.jhanjhi@taylors.edu.my, [3]azween.abdullah@taylors.edu.my
[†]School of Computer Science (SCE), Taylor's University, Subang, Malaysia
[4]raaziasaher@gmail.com
[††]College of Computer Science and Information Technology (CCSIT), King Faisal University, Al-Hassa. Saudi Arabia

Summary
The Internet of Things (IoT) is a technology that offers lucrative services in various industries to facilitate human communities. Important information on people and their surroundings has been gathered to ensure the availability of these services. This data is vulnerable to cybersecurity since it is sent over the internet and kept in third-party databases. Implementation of data encryption is an integral approach for IoT device designers to protect IoT data. For a variety of reasons, IoT device designers have been unable to discover appropriate encryption to use. The static support provided by research and concerned organizations to assist designers in picking appropriate encryption costs a significant amount of time and effort. IoTES is a web app that uses machine language to address a lack of support from researchers and organizations, as ML has been shown to improve data-driven human decision-making. IoTES still has some weaknesses, which are highlighted in this research. To improve the support, these shortcomings must be addressed. This study proposes the "IoTES with Security" model by adding support for the security level provided by the encryption algorithm to the traditional IoTES model. We evaluated our technique for encryption algorithms with available security levels and compared the accuracy of our model with traditional IoTES. Our model improves IoTES by helping users make security-oriented decisions while choosing the appropriate algorithm for their IoT data.

Keywords:
IoT, System Security, IoT Device Security, Cryptography, Machine Learning, System Design

1. Introduction

IoT is a rapidly-developing technology, with Statista projecting over 75.44 billion "things" to be Internet-connected by 2025 [1] [1]. It implies billions of physical devices worldwide with Internet connectivity for data-gathering and sharing. This advancement is induced by affordable computer chip production and the advent of communication technology [2]. The Internet proves instrumental in connecting actual contexts with digital ones and catalyzing technological evolution through smaller, more powerful, affordable embedded computers (IoT devices). The **I**nternet **o**f **T**hings (IoT) constitutes various heterogeneous devices following environmental disparities for deployment purposes. Even though IoT devices entail sensors that capture and assess information from the surrounding environment to be further analyzed in remote locations, the IoT devices share many similar attributes despite distinct variations and diverse elements. For example, all IoT devices aim toward data-capturing and sharing with restricted resource capacities (computational power, memory size, and energy) [3-5]. These devices are incorporated into various applications following their capacity to be connected to the internet and controlled remotely. Most IoT applications and devices have been deployed across multiple industries involving hospitals, travel, smart homes, and engineering.

Given that resource constraints inevitably limit their software, such installations should be lightweight to facilitate IoT devices and resolve the abovementioned restrictions. The IoT devices are susceptible to the risks potentially encountered by any Internet-connected equipment as they are primarily developed for online data management and transmission without human aid. For example, data collection in smart cities with IoT devices primarily entails health monitoring (blood pressure, heartbeat, or body activity) and home automation (temperature and moisture details and power consumption patterns).

The high demand for IoT technology and its significant impact on people's daily lives have regrettably garnered hackers' and attackers' attention to exploit and control devices, and the subsequent data derived from the IoT devices will have a significant impact on people's daily lives. Thus, IoT devices should be safeguarded against internal and external threats to mitigate adverse implications, given that 90% of IoT-derived data would be installed in third-party databases in the following years [6]. As such devices denote primary targets for attackers following the stored information type, it is vital to encrypt IoT device-generated messages by considering the essentiality of steadily-accumulating account data [7].

[1] https://www.statista.com/statistics/471264/iot-number-of-connected-devices-worldwide/

Manuscript received August 5, 2022
Manuscript revised August 20, 2022
https://doi.org/10.22937/IJCSNS.2022.22.8.41

Thus, encrypting data gathered by IoT devices is critical because it adds a degree of safety and instils a sense of trust in the IoT device's user. Unfortunately, this process is not straightforward, as the IoT device designer confronts several obstacles due to various factors. In summary, five factors are discussed with the reasoning in [8]. The first is the heterogeneous environment in which the IoT device is placed, and the second is the range of applications available within the same environment. Thirdly, the price of IoT components has decreased significantly in recent years, limiting the funds available for security research. At the same time, the fourth reason is that most IoT designers come from an embedded device background, where these devices operate in closed systems and lack internet access, and where the most effort and experience are focused on hardware development. The last factor is the considerable number of encryption algorithms, where each of them has its own pros and cons, which makes the selection of the encryption algorithm for a specific device a challenging task.

Researchers and concerned organizations performed significant development to mitigate the effects of the aforementioned factors. The researchers extensively analyzed multiple encryption techniques and quantified their effect on various IoT device resources. On the other side, concerned organizations provide workshops and publish periodic documentation to educate designers about the best practices for protecting IoT devices.

The support provided to designers by organizations requires a lot of effort and time. That is why there is a need to provide an automated solution where it can provide them with the necessary support with a substantially reduced requirement of time and effort. IoTES was developed to overcome these shortcomings. IoTES is a web app developed as part of an academic research project to evaluate the feasibility of applying machine learning to assist IoT device designers when they want to select an appropriate encryption algorithm with the least effect on their device resources. IoTES is an open-source project to encourage other academics to enhance its features.

This study aims to improve the security features of IoTES by identifying security level deficiency in IoTES and providing an improved solution with enhanced security features. The following section analyzes the IoTES model with motivation for its development, structure, advantages, and shortcomings. In the next section, we discuss the potential improvement in IoTES. The next section provides the implementation of security level enhancement in IoTES using machine learning models, algorithms, and their evaluation. The last section concludes the paper with a discussion of the security level enhancement we made to address one of IoTES's shortcomings

2. IoTES Model for Design Dependent Encryption Selection for IoT Devices

This section explains the Model for Design Dependent Encryption for IoT Devices (IoTES [18]) with its motivation, related work and implementation details. We also discuss the advantages and limitations of IoTES to determine the possible improvement.

2.1 Motivation

Data privacy and security have become challenging given the heterogeneously increasing number of internet-connected devices ("things"). Hence, IoT device-gathered data security is a pivotal and daunting study area. Specifically, IoT device application has facilitated the connection of daily things and communication among machines, people, and systems in the physical world [3]. The IoT platform, which encompasses individual-distributed devices to gather data on their daily routines, places, family members, and bank accounts [9], has highlighted the essentiality of data encryption for high data integrity and confidentiality. Encryption and hash functions are also utilized to protect the passwords and confidential data embedded in IoT devices apart from significantly influencing system connectivity and authentication. Based on the primary motive underlying this study, designers who establish device security are static, given the versatile nature of IoT devices, where changes in design and specifications could be made for multiple contexts. For example, designers' security design relies on past knowledge or skills limited to specific physical scenarios or settings. As a broader range of hardware IoT device designs is anticipated following global advancements towards complete IoT-based automation, security denotes a primary concern for IoT networks. In other words, optimal encryption methods substantially impact the security extension for this rapidly growing network. IoTES aimed to recommend and establish integration between appropriate encryption methods and design requirements for high security.

Every IoT device entails novel architecture without IoT hardware standardization [10-13]. Multiple non-encrypted (constrained) devices have been developed as IoT devices, encountering limited resources[14-16]. Meanwhile, a considerable number of encryption methods have emerged [17] in line with recent research [13]. The justifications mentioned above render identifying appropriate encryption methods for multiple IoT devices challenging for IoT device designers and developers [13]. Many Empirical works have been published for IoT designers to take constructive measures within the IoT industry. The measures based on the "Security by Design Concept" by optimizing designers' knowledge and developing suitable IoT encryption selection methodologies have prompted us to structure and develop a

model named IoTES with all the necessary data and select the most appropriate encryption for every IoT device.

2.2 Structure of IoTES

This section explains the development phases of IoTES, such as experimental design, data analysis, machine learning model and web app development, as shown in Figure 1.

2.2.1 Experimental Design

The first step in IoTES development is building a dataset using the encryption algorithms and the IoT devices. IoTES dataset is created via a series of experiments including various encryption methods (28 in all) belonging to three distinct encryption classes: Symmetric, Asymmetric, and Lightweight. These encryption techniques are implemented in Python using standard libraries. Finally, a platform is developed to facilitate the testing process. Three single-board computers are used to evaluate these encryption algorithms: Raspberry Pi-3, Raspberry Pi-Zero, and Pocket Beagleboard. The developed platform is capable of encrypting the message and logging the effect of the encryption on device resources, such as processing power and time consumption.

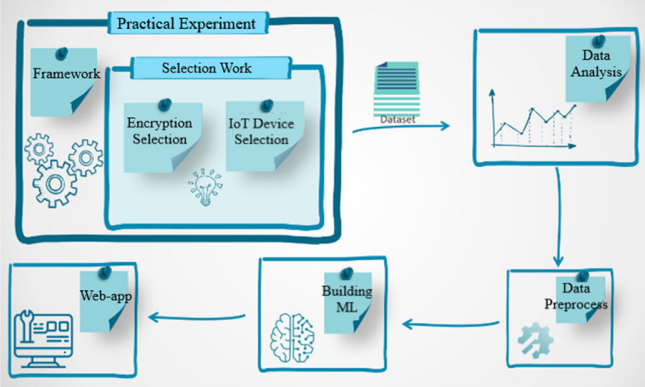

Fig. 1. IoTES Methodology

2.2.2 Data Analysis and Preprocessing

Data analysis is performed in the next phase to validate the data and get valuable insights. After data analysis, the dataset is processed through several preprocessing and feature engineering techniques for statistical significance in the machine learning model. Data preprocessing is an essential step in data management for our machine learning model and includes cleaning, normalization, and the change of features. The study's categorical features required much data manipulation, as predictive models perform better by converting the categorical data to numeric-categorical values. One hot encoding technique encodes the features, while the label encoding technique transforms the target column in IoTES. Data scaling is also performed for several features as the machine learning algorithms determine the distance between data. The numerical columns are scaled so that data ranges between [0,1]. Standard scaler is not being used because the best performing machine learning models are tree-based models, and standardization did not affect the results.

After preprocessing, the data is divided into two sets, a train set, and a test set. We have used an 80/20 data-splitting ratio to develop an accurate prediction model, i.e., 80 percent of the data is used to train the model, and just 20 percent is used to evaluate its performance. Data validation is performed using 25% of the training data to evaluate the model stability. For statistical purposes, the dataset included a large amount of data from many measurements made using a single device and encryption with the same settings, leading to many duplicate instances. To prevent data leaking from the training set to the testing set, we have grouped this kind of data and be exposed to the training set or testing set. The preprocessing steps are shown in Algorithm 1, and the same preprocessing steps are performed for the security level addition implemented in this paper due to the same rough dataset being used.

2.2.3 Machine Learning Model Development

The next step is to build a machine learning model capable of selecting an appropriate encryption algorithm based on various inputs. Two machine learning approaches are implemented for IoTES. The first approach is constructed using supervised learning multiclass classification models with the encryption's name column as the target. Five different supervised learning models are trained and tested for the target. The data is then used as training and testing sets to evaluate the model's performance. Following training and testing of the classification models, the results of all models are compared to select the most accurate model and carry it out to the implementation phase. The gradient boosting technique is selected for the final model of IoTES (the results of all models are published in [18]).

The second approach is to mimic the flow of the experiment, where the information about the encryption algorithm is already known, such as the key size, block size, cipher type, and the device information such as processor type and frequency, and memory size. The unknown values for use are the load of the encryption algorithm on device resources. Therefore, the measured parameters are specified as the target variable. These measured parameters are the four features, including 'DURATION', 'CPU%', 'PERFORMANCE', and 'MEM_USAGE%'. We use regression models for all four features because of

numerical feature values. We also use a classification model for 'DURATION' because it contains a broad date range and will improve the results if the user input is categorized into low or high duration. Four regression models and one classification model are also used for our proposed approach. Our proposed IoTES with security level is expected to have better predictions for other devices than those utilized in this experiment.

2.2.4 Web-App Development

The last stage is to make the model available to the community, and a web-based application is designed to serve the purpose. We construct a dashboard using the python module ipywidgets. Ipywidgets is a collection of interactive HTML widgets for use with Jupyter notebooks. It empowers the user with control over data and visualizes data changes on a graph. We utilize the voilà python package, which converts a standard HTML notebook into a standalone web application. The complete work, including codes, models, dashboards, and datasets, is uploaded to GitHub and linked to a platform as a service (Heroku) that enables application execution on the cloud. The next step is to link the domain (iotes.net) to the Heroku platform and serve all web requests to our model using the developed web app.

2.3 Advantages

Choosing an effective encryption method for an IoT device requires taking several aspects into account, including the device's design, the capacity of its resources, and the encryption's consumption of those resources. Human decision-making based on data will diminish as the number of variables increases, while machine learning is a well-known approach for addressing such multidimensional issues. Therefore, IoTES apply machine learning to choose the appropriate encryption approach based on the resources available to the IoT device. IoTES is designed as a web application so that designers can readily use it. It generates suggestions based on the device specification and the designer's requirements. By using IoTES, designers will save significant time and effort in picking a suitable encryption method for their products. The IoTES does not need any previous understanding of encryption.

2.4 Limitations

There are some limitations present in IoTES which are described in this section.

Platform Testing Accuracy: IoTES platform incorporates different types of profilers, yet the memory profiler provides inaccurate readings, and the deployment of a better profiler is required to have better readings**.**

IoT Device Variety: IoTES uses specific primary devices for generating the dataset due to the absence of an available dataset. Nevertheless, the proposed model could not be deployed on all IoT devices following the diversity of current IoT devices.

Limitation of Encryption Algorithms: Multiple encryption schemes are identified, each having unique implications on different resource availability given the broad device diversity. Furthermore, the number of available encryption algorithms continues to expand daily. IoTES does not include all of them in the training or testing model following various methods.

Dataset Limitation: We could not produce a comprehensive and generalized dataset that entailed more devices and encryption methods towing to the two limitations mentioned earlier. Regardless, this work is implementable with improvements for model generalization.

Extended Features: Multiple features encompassing temperature, operating system type, power consumption, or message encryption frequency that could significantly influence IoT device resources are beyond the current study scope and could be addressed in the future version.

Encryption Algorithm Security Level: The dataset in IoTES does not include the security level provided by each encryption algorithm which is necessary for the designer to build a better decision on encryption selection.

3. IoTES with Security

The primary goal of IoTES is to strengthen security by encrypting messages inside IoT devices. This goal can be achieved by encouraging IoT device designers to adopt an encryption technique. The motivation is provided by simplifying the process of selecting an appropriate encryption algorithm while maintaining the load within the required limit. Since IoTES recommends several suitable encryption algorithms to the user based on their inputs and security is the main objective for selecting an appropriate encryption algorithm, the security level provided by the encryption algorithm has to be provided with it to strengthen the user decision and make it security-oriented. The current version of IoTES does not incorporate the security level information, which is one of the shortcomings of IoTES stated in the previous section. The following section explains the steps to add this important improvement to IoTES.

4. Methodology

The method used to achieve the goal of including the security level provided by each encryption technique in IoTES involves finding the availability of security information, adding security level information to the dataset, preprocessing the dataset, training, deploying and evaluating the model, which is discussed in the following section.

4.1 Collecting Encryption Security Information

The first step for including encryption algorithm security levels is to survey the literature and find the security level of the encryption algorithms included in IoTES. The security levels of the encryption algorithms are shown in Table 1. The security level for all the encryption algorithms included in IoTES is unavailable. Therefore, a new model named 'IoTES with Security' is developed along with the previously available version of IoTES.

Table 1: Security Level Provided By Different Encryption Algorithms used In IoTES

Cryptosystem	Operation	Key Size (bit)/Round	Strength Level	Description	Comparison to RSA	Status Through 2030
[19] AES	Encryption(sym)	128/10	128	Medium Security	3072	Acceptable
		192/12	192	Medium- High Security	7680	Acceptable
		256/14	256	High Security	15360	Acceptable
[20] DES	Encryption(sym)	56/16	64	Weak Security	$640-1024$	Avoid
[19, 21] 3DES (TDES)	Encryption(sym)	$112/3\times16$	80	Weak Security	1024	Legacy
		$168/3\times16$	112	Medium Security	2048	Legacy
[22, 23] ECDSA	Key exchange & Authentication	$160-223$	80	Weak Security	1024	Legacy
		$224-255$	112	Medium Security	2048	Acceptable
		$256-383$	128	Medium Security	3072	Acceptable
		$384-511$	192	Medium- High Security	7680	Acceptable
		$512+$	256	High Security	15360	Acceptable
[19] RSA	Encryption(asym) Size of modulus $N=p.q$	1024	≤ 80	Weak Security	–	Legacy
		2048	112	Medium Security	–	Acceptable
		3072	128	Medium Security	–	Acceptable
		7680	192	Medium- High	–	Acceptable
		15360	256	High Security	–	Acceptable
[24, 25] Rabin	Signature (asym)	3072	128	Medium Security	3072	Acceptable
		7680	192	Medium- High Security	7680	Acceptable
		15360	256	High Security	15360	Acceptable
[26] Rabbit		128	128	Medium Security	3072	Acceptable
[27-29] ChaCha No attack on chacha **[28]**	Encryption(sym)	128/6	107	Medium Security	$1024-2048$	Legacy
		256/6	139	Medium Security	$3072-7680$	Acceptable
		128/7	128	Medium Security	3072	Acceptable
		256/7	248	High Security	$768-15360$	Acceptable
		256/8	256	High Security	15360	Acceptable
		256/9	256	High Security	15360	Acceptable
[30] Fernet	Encryption & authentication (sym) AES 128 CBC	256	128	Medium Security	3072	Acceptable
[31, 32] ARC2	Encryption(sym)	*broken*		Weak Security	–	Avoid
[27-29, 33] Salsa No attack on	Encryption(sym)	128/7	111	Medium Security	$1024-2048$	Legacy
		256/7	151	Medium Security	$3072-7680$	Acceptable
		128/8	128	Medium Security	3072	Acceptable

Salsa **[28]**			256/8	251	High Security	768 − 15360	Acceptable
			128/9	128	Medium Security	3072	Acceptable
			256/9	256	High Security	15360	Acceptable
			128/10	128	Medium Security	3072	Acceptable
			256/10	256	High Security	15360	Acceptable
			128/11	128	Medium Security	3072	Acceptable
			256/11	256	High security	15360	Acceptable
			128/12	128	Medium Security	3072	Acceptable
			256/12	256	High Security	15360	Acceptable
			128/20	128	Medium Security	3072	Acceptable
			256/20	256	High Security	15360	Acceptable
[34-36] CAST	Encryption(sym)		64	48	Weak Security	480	Avoid
			128/5	31.4	Weak Security	≤ 480	Avoid
			128/16	44.2	Weak Security	≤ 480	Avoid
			256	256	High Security	15360	Acceptable
			1024	1024	High Security	≥ 15360	Acceptable
[37] Ed25519	Signature (asym)		256	128	Medium Security	3072	Acceptable
[38-40] Sphincs	Integrity		256	256	High Security	15360	Acceptable
[41] Schmidt-Samoa	Encryption(asym)		Like Rabin	–	–	–	–
[42] Lamport	Integrity		256	128	Medium Security	3072	Acceptable
[43] Present	Encryption(sym)		80/26	68,2	Weak Security	640 − 1024	Avoid
			80/27	72	Weak Security	640 − 1048	Avoid
			80/28	77,4	Weak Security	640 − 1048	Avoid
			128/28	122	Medium Security	2048 − 3072	Acceptable
[44] Clefia	Encryption(sym)		128	127.44	Medium Security	2048 − 3072	Acceptable
[31, 45-47] Simon	Encryption (sym)	**Block Size**					
		32	64/24	63	Weak Security	640 − 1024	Avoid
		48	72/24	56	Weak Security	640	Avoid
		48	96/25	80	Weak Security	1024	Legacy
		64	96/32	63	Weak Security	640 − 1024	Avoid
		96	96/37	88	Weak Security	1024 − 2048	Legacy
		64	128/31	120	Medium Security	2048 − 3072	Acceptable
		128	128/49	120	Medium Security	2048 − 3072	Acceptable
		96	144/38	136	Medium Security	3072 − 3072	Acceptable
		128	192/51	184	Medium Security	3072 − 3072	Acceptable
		128	256/24	248	High Security	307 − 15360	Acceptable
[31, 48] Skinny	Encryption(sym)		128/36	51.95	Weak Security	480 − 640	Avoid
			128/40	52	Weak Security	480 − 640	Avoid
			256	256	High Security	15360	Acceptable
			284	284	High Security	307 − 15360	Acceptable
[49] TEA	Encryption(sym)		128	121.5	Medium Security	2048 − 3072	Acceptable
[50] XTEA	Encryption(sym)		128/36	126,44	Medium Security	2048 − 3072	Acceptable
			128/64	128	Medium Security	3072	Acceptable
[51] XXTEA	Encryption(sym)		128/64	≤ 128	Medium Security	2048 − 3072	Acceptable

This message security level integration model will work for the security issues addressed in the [52-55], and also for the issues at the design level [56]. Further, this can help the IoT-based devices and software for different domain applications such as [56-58]. This can help further with any different IoT-based designs.

4.2 Adding Security Level to Dataset

All the steps of data preprocessing are provided in Algorithm 1. We start with processing the data and split the dataset into four categories, i.e., Device Parameters, Algorithm Parameters, Measure Parameters and other parameters. Next, we remove any unnecessary data which is found in the 'other parameters' category. The Performance column has a wide range of values due to the usage of lightweight algorithms with very high performance. Any encryption algorithm with performance higher than 400 is considered as good. All other numerical columns are normalized. For the traditional approach, 'TYPE' and 'CIPHER TYPE' columns are removed for dimensionality reduction of data. The traditional IoTES model includes all the 28 encryption algorithms, and "IoTES with Security" include only those encryption algorithms for which their security level is known. Therefore, for adding security level information to the traditional IoTES model, we add the security level information in a new column, remove the algorithms not having their security level, and encode it with the label encoder technique. We remove all the digital signature algorithms as the security level of only two such algorithms is available.

For the "IoTES with Security" model, we do not need to add the security level in the dataset as we will use the security level from the user dashboard. We take measure variables as target variables and split data into independent and target variables for each target variable. The categorical data is coded using the one-hot-encoding technique. The final step in data preprocessing is to avoid data leakage from train to test set. Because our data consists of repeated experimental tests, it includes many repeated and similar data. If we do not separate this kind of data properly, we will end with similar data in the training and testing set where the test results will not be reliable, and our model will not perform as expected in production. Therefore, duplicate data is added to the same group to avoid duplication between the train and test data.

Algorithm 1 Data Preprocessing

```
Split Parameters into categories
Device Parameters ← [' DEVICE',' CPU MODEL NAME',' CPU ARCHITECTURE',' CPU COUNT',
                     ' TOTAL CPU MAX (MHZ)']
Algorithm Parameters ← [' TYPE',' ALGORITHM',' FUNCTIONALITY',' CIPHER TYPE',' KEY SIZE',
                        ' BLOCK SIZE',' MESSAGE SIZE']
Measure Parameters ← [' DURATION (SEC)',' CPU%',' PERFORMANCE',' MEM USAGE (%)']
Other Parameters ← ['RANGE_DURATION (SEC)'....]
for ALL Data do
    if !Data.equalTo(Device) OR !Data.equalTo(Algorithm) OR !Data.equalTo(Measure) then
        remove DATA
    end if
end for
PERFORMANCE ← 400
numericalcolumns scaled to [0,1] range
if Model.equalTo(Traditional Approach) then
    remove Model.'TYPE'
    remove Model.'CIPHER TYPE'
end if
if Model.equalTo(Traditional Approach) and Model.contains(SECURITY DATA) then
    remove Model.'TYPE'
    remove Model.'CIPHER TYPE'
    Add SECURITY INFORMATION
    LabelEncoder encodes SECURITY INFORMATION
    for ALL Algorithms do
        if !Algorithm.contains(SECURITY INFORMATION) then
            remove Algorithm
        end if
    end for
    for ALL Algorithms do
        if Algorithm.contains(DIGITAL SIGNATURE) then
```

```
            remove Algorithm
        end if
    end for
end if
if Model.equalTo(New Approach) then
    Split DATA to Independent variables and Target variables
    Target variables ← Measure Parameters
end if
OneHotEncoder encodes Categorical Variables
group MEASUREMENTS from DATA
Split groups to Training and Test
Save Traditional Approach Training Files
Save Traditional Approach Test Files
Save New Approach Training Files
Save New Approach Test Files
```

4.3 Training model

As we have seen in the previous step, the traditional model contains two datasets. The first dataset contains 28 encryption algorithms without including the security strength of these algorithms. The second dataset includes 15 encryption algorithms along with the security level of each of those algorithms. The dataset is split into two sets for the training and testing of both models. We split data into independent and target variables where the target column is the algorithm column. We train six different models to compare their results and carry the best-performed model. Therefore, we plot the confusion matrix and classification report for comparison purposes. All the training and testing steps of the traditional classification approach IoTES are described in Algorithm 2.

We created four regression models and one classification model for the "IoTES with Security" model. Four regression models use measure variables as target column i.e., 'DURATION (SEC)', 'CPU%', 'PERFORMANCE', 'MEM USAGE (%)'). We train and test six machine learning algorithms for each measurement target to compare their result and carry out the best-performed model to the production step. An extra classification model is trained and tested for 'DURATION (SEC)' to classify the encryption which has higher time than the ones having lower time, and this model is developed because the data of 'DURATION (SEC)' has long variations and the classification model improves the selection accuracy. The result of each model is recorded for comparison purposes. All the process of training and testing of the "IoTES with Security" model is described in Algorithm 3.

Algorithm 2 Traditional Model Training and Testing

```
Load Traditional Approach Training Files.LabelEncoder
Load Traditional Approach Test Files.LabelEncoder
Split DATA to Independent variables and Target variables
Target variables ← Algorithm
Classification Models ← [' Logistic Regression',' Gradient Boosting',' Decision Tree',Random Forest','
                         XGBoost']
for ALL Traditional Approach and Traditional Approach.SecurityData do
    for ALL Classification Models do
        Build Model
        Train Model
        Test Model
        Plot Confusion Matrix
        Extract Classification Report
        if CurrentModel > PreviousModel then
            Best Model ← CurrentModel
        end if
        Save Model
        Save Model.SecurityData
    end for
```

```
19: end for
```

Algorithm 3 New Approach Model Training

```
1: Load New Approach Training Files.LabelEncoder
2: Load New Approach Test Files.LabelEncoder
3: Models ← [' SVM',' Logistic Regression',' Decision Tree',' Random Forest',' XGBoost']
4: for ALL Target Variables do
5:     for ALL Models do
6:         Build Model
7:         Train Model
8:         Test Model
9:         Save Model
10:    end for
11:    Plot Confusion Matrix
12:    for ALL Models do
13:        if CurrentModel > PreviousModel then
14:            Save Mosel Result
15:            Best Model ← CurrentModel
16:        end if
17:    end for
18: end for
19: if Target Variables.equals('DURATION(SEC) then
20:    Build Classification Model
21:    Train Classification Model
22:    Test Classification Model
23:    Select Model ← BestModel
24:    Build Regression Model (0.5 threshold point)
25:    Train Regression Model
26:    Test Regression Model
27:    Save Model
28: end if
```

4.4 Models Deployment

The selection of all the created models is specified in the web app deployment. User input data widgets are created in three different categories. The first category is the Algorithm category, where the user specifies the algorithm type, cipher type, key size, block size, and message size. The second category is the Device category which allows the user to select all specified parameters based on the device. The device widgets are connected; if the user selects a known device, all other input widgets fields will be filled automatically. The third category contains the measured parameters, which must be specified by the user based on the requirement. The security level has been added to the new window in the web app named "IoTES with Security" and utilizes the machine learning models with the security level. For "IoTES with Security", the user selects all the options described above, along with an additional security level, as shown in Figure 3.

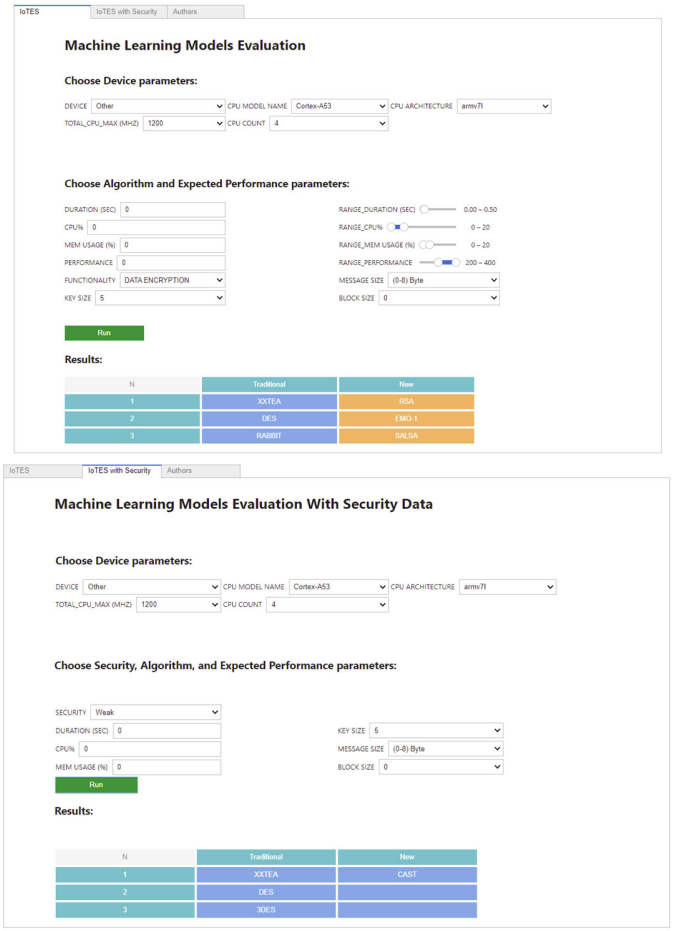

Fig. 3. IoTES Application Interface (A) without security information (B) after including security information

Measure parameters selected by the user are used as input to the traditional classification models. The model will return the top three algorithms matching the user input. The new approach models use device inputs as input parameters, and algorithm inputs are used for filtering the matching algorithms while the measure inputs and security level (if IoTES with Security is selected) input are used for sorting the algorithms. Euclidean distance is used to identify the suited algorithms by calculating scaled Euclidean distance between the measured + security level input values and model predictions. Algorithm filtering is removed if there is no algorithm satisfying the measure inputs. All these steps are explained in algorithm 4.

Algorithm 4 Models Deployment

```
for ALL Traditional Models, Traditional.Security Model and New Approach Model do
    Assess Data Path, LabelEncoder Path and Model Path
end for
Split InputData to Device, Algorithm and Measure
for ALL Traditional Models and Traditional.Security Models do
    Select Desired Algorithm, Measure values and Device type
    if Model.equalTo(Traditional.Security) then
        Select Security values
    end if
```

```
end for
if Model.equalTo(Traditional) then
    Input ← Measure Parameters
end if
 if Model.equalTo(New Approach) then
    Desired Value ← Measure Parameters
end if
for ALL Traditional Model and Traditional.Security Model do
    InputParameters ← Device, Algorithm and Measure Parameters
    for ALL Predictions do
        Sort Predictions.Probability then Print Prediction[0], Prediction[1], Prediction[2]
    end for
end for
for ALL New Approach Model do
    InputParameters, Filtering, SortingAlgorithms ← Device, Algorithm, Measure Params
    for ALL Algorithms do
        Use all available key and block sizes
        Calculate EuclideanDistance
        for ALL Target Variables do
            Assess IndividualCoefficients
        end for
        Calculate EuclideanDistance.MeasureTargetValues(Widgets, Predictions)
    end for
    Save DataSet
    for ALL Target Variables do
        Predict Value then Measure.Parameters ← Value
    end for
    if Algorithm < Requirements then
        Remove Algorithm
    end if
    if !(ALL Algorithms < Requirements) then
        Remove Filtering
    end if
    for ALL Models do
        Sort Models.EuclideanDistance then Print Model[0], Model[1], Model[2]
    end for
    Update Widget Values
end for
```

4.5 Models Results

As previously indicated, the traditional approach consists of two datasets, one containing all 28 encryption algorithms but not their security levels, while the other dataset contains just those algorithms to which security levels have been included. The first dataset was preprocessed before being trained and tested using six machine learning techniques to select the best model and carry it out for production. The results are shown in Table 2, showing that Linear Regression has the lowest performance following the Support Vector Classifier. The tree-based models are performing better than the previously mentioned models. It is visible from Figure 2(A) that XGBoost and Gradient Boosting are the highest-performing models. Comparing traditional approach (IoTES) models with the "IoTES with Security" in Figure 2, we find that the Linear Regression and SVC have a slightly improved result due to the target column with fewer variables, and it has only 15 encryption algorithms to classify as compared to 28

algorithms for IoTES. At the same time, the performance of most of the tree-based algorithms has somehow reduced because of an increased number of features. Nevertheless, XGBoost and GradientBoosting again have the highest performance.

As discussed earlier, we built a separate regression model for each measured value for the traditional IoTES model. We trained and tested six different models to select the best among them. For CPU%, MEM%, and PERFORMANCE, the tree-based models performed better than the other. Interestingly, the decision tree and GradientBoosting have very close results, and we have adopted the GradientBoosting model. We can see the model result of each measured column in Table 3. We can notice that the Mean Absolute Error (MAE) of the DURATION (SEC) parameter is a high value due to the reason that the Duration (SEC) column has a wide range of values. The higher values in this column come from the heavily tested algorithm such as Schmidt-Samoa, precisely when it is tested on a Pi-Zero device. That is why we decided to use a classification model to classify the algorithms with more encryption time than those requiring less time with 0.5 seconds as the threshold point. The training accuracy of the classification model is 99.8%, and the test accuracy is 100.0%. Then we used a separate regression model for the algorithms with a duration of more than 0.5 and lower than 0.5 seconds. The result of these two different regression models is shown in Table 4.

Table 2: Traditional Classification Models Comparison with and Without Adding Security Level Feature

	Machine learning Model	***Train accuracy***	***Test accuracy***	***Train precision***	***Test precision***	***Train Recall***	***Test Recall***	***Train F1 Score***	***Test F1 Score***
Traditional Classification Approach	Linear Regression	0.77495	0.749397	0.800999	0.762203	0.745562	0.768547	0.749401	0.724557
	SVC	0.867203	0.855591	0.888322	0.867743	0.864948	0.877384	0.855289	0.850899
	XGBoost	1	0.958568	1	1	1	0.959633	0.959488	0.959282
	Decision Tree	0.997485	0.952534	0.997549	0.997535	0.997526	0.953678	0.953412	0.953357
	Random Forest	0.944366	0.925583	0.95474	0.944964	0.944172	0.939341	0.926726	0.924836
	Gradient Boosting	1	0.958166	1	1	1	0.959362	0.95912	0.958812
Traditional Classification Approach with Algorithm Security Level	Linear Regression	0.809873	0.664029	0.867482	0.795027	0.776298	0.715052	0.721937	0.66372
	SVC	0.896452	0.861871	0.899355	0.902837	0.898826	0.857903	0.856368	0.85311
	XGBoost	1	0.923741	1	1	1	0.909696	0.910214	0.909799
	Decision Tree	0.998072	0.920863	0.998284	0.998064	0.998161	0.907338	0.910541	0.908317
	Random Forest	0.979368	0.91295	0.980617	0.981506	0.980338	0.902539	0.893533	0.895189
	Gradient Boosting	1	0.930216	1	1	1	0.92148	0.917521	0.918981

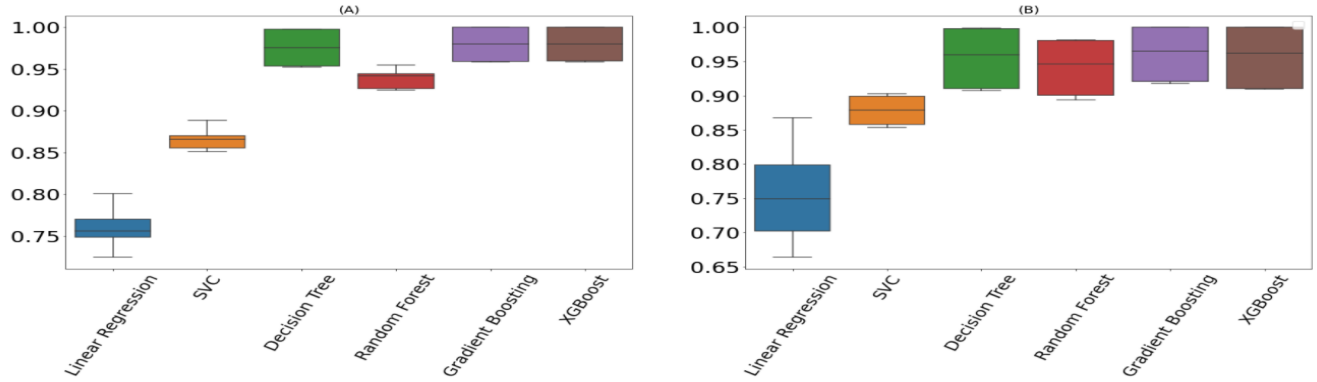

Fig. 2. Different ML models comparison (A) Traditional classification approach (B) Traditional classification approach with security level data.

Table 3: New Approach Ml Models Results Comparison For All Measure Target Columns

	DURATION (SEC)				CPU%			
	Train MAE	Test MAE	Train MSE	Test MSE	Train MAE	Test MAE	Train MSE	Test MSE
Lasso	1.60915	1.75337	26.03786	52.40168	0.2287	0.20975	0.08746	0.07304
SVR	0.63856	0.83064	22.00153	47.97857	0.05385	0.05115	0.00499	0.004
Tree	0.14429	0.3363	2.10946	12.4901	0.00599	0.00719	0.00014	0.00018
Forest	0.14584	0.33693	2.10959	12.49468	0.01116	0.01411	0.00049	0.00096
Boosting	0.14429	0.33635	2.10946	12.49025	0.00599	0.00716	0.00014	0.00018
XGBoost	0.14436	0.33647	2.10946	12.49056	0.00606	0.00697	0.00014	0.00017
	MEM USAGE (%)				PERFORMANCE			
	Train MAE	Test MAE	Train MSE	Test MSE	Train MAE	Test MAE	Train MSE	Test MSE
Lasso	0.04502	0.04506	0.00219	0.00219	0.30182	0.29002	0.126	0.12061
SVR	0.04933	0.04823	0.00261	0.0025	0.04933	0.07366	0.00851	0.01023
Tree	0.00494	0.00626	0.00015	0.00023	0.00494	0.01191	0.00095	0.0007
Forest	0.0053	0.00646	0.00016	0.00023	0.0053	0.01811	0.00146	0.00123
Boosting	0.00497	0.00646	0.00015	0.00023	0.00497	0.01221	0.00095	0.0007
XGBoost	0.00528	0.0065	0.00015	0.00022	0.00528	0.01277	0.00095	0.00071

Table 4: Duration (Sec) Ml Models Result For Classified Data

	Train MAE	Test MAE	Train MSE	Test MSE
<0.5 model	0.00399	0.00459	0.00004	0.00012
<0.5 model	0.14234	0.33308	2.10942	12.4898

5. Conclusion

IoTES provides IoT device designers with extensive dynamic support to make the process of identifying appropriate encryption for their devices as simple as possible. This assistance is far more advanced than other alternatives offered by other researchers and organizations. However, IoTES still has some limitations identified in this work. One of these limitations, namely the absence of the security level supplied by each encryption technique, has been addressed in this study. In this work, we have presented the "IoTES with Security" model with the added support of the security level feature to IoTES. Evaluation of our model improves the classification accuracy with the added security level support of encryption algorithms.

We still hope that all other IoTES drawbacks can be addressed as well, and we hope we can get the support of the research community to speed up the improvement process. For this purpose, all the work of building IoTES is available to all researchers interested in participating in the improvement process. Those who successfully solve one or more of the IoTES flaws will be listed as authors and participants on the website (www.iotes.net), along with their participation.

REFERENCES

[1] A. Rghioui and A. Oumnad, "Internet of Things: Visions, technologies, and areas of application," *technology,* vol. 6, no. 7, 2017.

[2] A. Ragab, G. Selim, A. Wahdan, and A. Madani, "Robust Hybrid Lightweight Cryptosystem for Protecting IoT Smart Devices," in *Security, Privacy, and Anonymity in Computation, Communication, and Storage*, (Lecture Notes in Computer Science, 2019, ch. Chapter 1, pp. 5-19.

[3] M. N. B. Anwar, M. Hasan, M. M. Hasan, J. Z. Loren, and S. T. Hossain, "Comparative Study of Cryptography Algorithms and Its Applications," *International Journal of Computer Networks and Communications Security,* vol. 7, no. 5, pp. 96-103, 2019.

[4] T. Poongodi, R. Krishnamurthi, R. Indrakumari, P. Suresh, and B. Balusamy, "Wearable Devices and IoT," in *A Handbook of Internet of Things in Biomedical and Cyber Physical System*: Springer, 2020, pp. 245-273.

[5] N. Maryanti, R. Rohana, and M. Kristiawan, "The Principal's Strategy In Preparing Students Ready To Face the Industrial Revolution 4.0," *International Journal of Educational Review,* vol. 2, no. 1, pp. 54-69, 2020.

[6] M. Hibti, K. Baïna, and B. Benatallah, "Towards Swarm Intelligence Architectural Patterns: an IoT-Big Data-AI-Blockchain convergence perspective," in *Proceedings of the 4th International Conference on Big Data and Internet of Things*, 2019, pp. 1-8.

[7] M. S. Mahdavinejad, M. Rezvan, M. Barekatain, P. Adibi, P. Barnaghi, and A. P. Sheth, "Machine learning for Internet of Things data analysis: A survey," *Digital Communications and Networks,* vol. 4, no. 3, pp. 161-175, 2018.

[8] S. Roy, U. Rawat, H. A. Sareen, and S. K. Nayak, "IECA: an efficient IoT friendly image encryption technique using programmable cellular automata," *Journal of Ambient Intelligence and Humanized Computing,* pp. 1-20, 2020.

[9] A. C. Chhoton, "Executing an Effective IoT Security Testing Methodology: A Complete Guideline for Device Developers," 2018.

[10] L. Marin, M. P. Pawlowski, and A. Jara, "Optimized ECC implementation for secure communication between heterogeneous IoT devices," *Sensors,* vol. 15, no. 9, pp. 21478-21499, 2015.

[11] M. A. U. Rehman, R. Ullah, C.-W. Park, and B. S. Kim, "Towards Network Lifetime Enhancement of Resource Constrained IoT Devices in Heterogeneous Wireless Sensor Networks," *Sensors,* vol. 20, no. 15, p. 4156, 2020.

[12] L. Wei, Y. Chen, Y. Zhang, L. Zhao, and L. Chen, "PSPL: A Generalized Model to Convert Existing Neighbor Discovery Algorithms to Highly-efficient Asymmetric Ones for Heterogeneous IoT Devices," *IEEE Internet of Things Journal,* 2020.

[13] S. Singh, P. K. Sharma, S. Y. Moon, and J. H. Park, "Advanced lightweight encryption algorithms for IoT devices: survey, challenges and solutions," *Journal of Ambient Intelligence and Humanized Computing,* pp. 1-18, 2017.

[14] J. Arshad, M. A. Azad, M. M. Abdeltaif, and K. Salah, "An intrusion detection framework for energy constrained IoT devices," *Mechanical Systems and Signal Processing,* vol. 136, p. 106436, 2020.

[15] C. Su, F. Ye, L.-C. Wang, L. Wang, Y. Tian, and Z. Han, "UAV-assisted wireless charging for energy-constrained IoT devices using dynamic matching," *IEEE Internet of Things Journal,* 2020.

[16] F. Samie, L. Bauer, and J. Henkel, "Hierarchical Classification for Constrained IoT Devices: A Case Study on Human Activity Recognition," *IEEE Internet of Things Journal,* 2020.

[17] T. Sharma, "Lightweight Encryption Algorithms, Technologies, and Architectures in Internet of Things: A Survey," in *Innovations in Computer Science and Engineering*: Springer, 2020, pp. 341-351.

[18] M. Saleh, N. Jhanjhi, A. Abdullah, and R. Saher, "IoTES (A Machine learning model) Design dependent encryption selection for IoT devices," in *2022 24th International Conference on Advanced Communication Technology (ICACT)*, 2022: IEEE, pp. 239-246.

[19] E. Barker, W. Barker, W. Burr, W. Polk, and M. Smid, "NIST special publication 800-57," *NIST Special publication,* vol. 800, no. 57, pp. 1-142, 2007.

[20] R. Liu, Z. Weng, S. Hao, D. Chang, C. Bao, and X. Li, "Addressless: enhancing IoT server security using IPv6," *IEEE Access,* vol. 8, pp. 90294-90315, 2020.

[21] E. Barker and N. Mouha, "Recommendation for the triple data encryption algorithm (TDEA) block cipher," National Institute of Standards and Technology, 2017.

[22] S. Vanstone, "Responses to NIST's proposal," *Communications of the ACM,* vol. 35, no. 7, pp. 50-52, 1992.

[23] M. Suárez-Albela, P. Fraga-Lamas, and T. M. Fernández-Caramés, "A practical evaluation on RSA and ECC-based cipher suites for IoT high-security energy-efficient fog and mist computing devices," *Sensors,* vol. 18, no. 11, p. 3868, 2018.

[24] M. A. Asbullah and M. R. Kamel, "Design and Analysis of Rabin-p Key Encapsulation Mechanism for CyberSecurity Malaysia MySEAL Initiative," *IJCR,* vol. 9, no. 1, pp. 19-51, 2019.

[25] M. O. Rabin, "Digitalized signatures and public-key functions as intractable as factorization," Massachusetts Inst of Tech Cambridge Lab for Computer Science, 1979.

[26] M. Boesgaard, M. Vesterager, T. Christensen, and E. Zenner, "The stream cipher rabbit," *ECRYPT Stream Cipher Project Report,* vol. 6, p. 28, 2005.

[27] J.-P. Aumasson, S. Fischer, S. Khazaei, W. Meier, and C. Rechberger, "New features of Latin dances: analysis of Salsa, ChaCha, and Rumba," in *International Workshop on Fast Software Encryption*, 2008: Springer, pp. 470-488.

[28] S. Dey and S. Sarkar, "Improved analysis for reduced round Salsa and Chacha," *Discrete Applied Mathematics,* vol. 227, pp. 58-69, 2017.

[29] A. R. Choudhuri and S. Maitra, "Differential Cryptanalysis of Salsa and ChaCha-An Evaluation with a Hybrid Model," *IACR Cryptol. ePrint Arch.,* vol. 2016, p. 377, 2016.

[30] "Fernet (symmetric encryption)." [Online]. Available: https://cryptography.io/en/latest/fernet/.

[31] H. AlKhzaimi and M. M. Lauridsen, "Cryptanalysis of the SIMON Family of Block Ciphers," *IACR Cryptol. ePrint Arch.,* vol. 2013, p. 543, 2013.

[32] "Analysis of RC2." [Online]. Available: https://www.cryptrec.go.jp/exreport/cryptrec-ex-1042-2001.pdf.

[33] S. Khazaei, "Neutrality-Based Symmetric Cryptanalysis," EPFL, 2010.

[34] X. Zhou, J. Li, X. Lai, and H. Yan, "Revisit and Cryptanalysis of a CAST Cipher," *DEStech Transactions on Computer Science and Engineering,* no. ICEITI, 2017.

[35] L. Elbaz and H. Bar-El, "Strength assessment of encryption algorithms," *White paper,* 2000.

[36] M. Ebrahim, S. Khan, and U. B. Khalid, "Symmetric algorithm survey: a comparative analysis," *arXiv preprint arXiv:1405.0398,* 2014.

[37] J. Yu, "Is there a case to prefer Ed25519 over ECDSA P-256 for DNSSEC?."

[38] J.-P. Aumasson *et al.*, "SPHINCS," 2019.

[39] D. Amiet, A. Curiger, and P. Zbinden, "FPGA-based accelerator for post-quantum signature scheme SPHINCS-256," *IACR Transactions on Cryptographic Hardware and Embedded Systems,* pp. 18-39, 2018.

[40] A. Hülsing, J. Rijneveld, and P. Schwabe, "Armed sphincs," in *Public-Key Cryptography–PKC 2016*: Springer, 2016, pp. 446-470.

[41] K. Schmidt-Samoa, "A new rabin-type trapdoor permutation equivalent to factoring," *Electronic Notes in Theoretical Computer Science,* vol. 157, no. 3, pp. 79-94, 2006.

[42] D. D. Berendsen, "A Comparative Study on Signature Schemes for IoT Devices," 2021.

[43] A. F. Gutierrez and M. Naya-Plasencia, "Improving key-recovery in linear attacks: Application to 28-round PRESENT," in *EUROCRYPT 2020-Annual International Conference on the Theory and Applications of Cryptographic Techniques*, 2021, no. 12105: Springer, pp. 221-249.

[44] S. Ahmadi, M. Delavar, J. Mohajeri, and M. R. Aref, "Security analysis of CLEFIA-128," in *2014 11th International ISC Conference on Information Security and Cryptology*, 2014: IEEE, pp. 84-88.

[45] H. Chen and X. Wang, "Improved linear hull attack on round-reduced Simon with dynamic key-guessing techniques," in *International Conference on Fast Software Encryption*, 2016: Springer, pp. 428-449.

[46] R. Beaulieu, D. Shors, J. Smith, S. Treatman-Clark, B. Weeks, and L. Wingers, "SIMON and SPECK: Block Ciphers for the Internet of Things," *IACR Cryptol. ePrint Arch.,* vol. 2015, p. 585, 2015.

[47] Z. Chu, H. Chen, X. Wang, X. Dong, and L. Li, "Improved integral attacks on SIMON32 and SIMON48 with dynamic key-guessing techniques," *Security and Communication Networks,* vol. 2018, 2018.

[48] P. Derbez, V. Lallemand, and A. Udovenko, "Cryptanalysis of SKINNY in the Framework of the SKINNY 2018–2019 Cryptanalysis Competition," in *International Conference on Selected Areas in Cryptography*, 2019: Springer, pp. 124-145.

[49] A. Bogdanov and M. Wang, "Zero correlation linear cryptanalysis with reduced data complexity," in *International*

Workshop on Fast Software Encryption, 2012: Springer, pp. 29-48.
[50] J. Lu, "Related-key rectangle attack on 36 rounds of the XTEA block cipher," *International Journal of Information Security,* vol. 8, no. 1, pp. 1-11, 2009.
[51] E. Yarrkov, "Cryptanalysis of XXTEA," *IACR Cryptol. ePrint Arch.,* vol. 2010, p. 254, 2010.
[52] S. M. Muzammal, R. K. Murugesan and N. Z. Jhanjhi, "A Comprehensive Review on Secure Routing in Internet of Things: Mitigation Methods and Trust-Based Approaches," in IEEE Internet of Things Journal, vol. 8, no. 6, pp. 4186-4210, 15 March15, 2021, doi: 10.1109/JIOT.2020.3031162.
[53] S. Ali et al., "Towards Pattern-Based Change Verification Framework for Cloud-Enabled Healthcare Component-Based," in IEEE Access, vol. 8, pp. 148007-148020, 2020, doi: 10.1109/ACCESS.2020.3014671.
[54] Fatima-tuz-Zahra, N. Jhanjhi, S. N. Brohi and N. A. Malik, "Proposing a Rank and Wormhole Attack Detection Framework using Machine Learning," 2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS), 2019, pp. 1-9, doi: 10.1109/MACS48846.2019.9024821.
[55] B. Hamid, N. Jhanjhi, M. Humayun, A. Khan and A. Alsayat, "Cyber Security Issues and Challenges for Smart Cities: A survey," 2019 13th International Conference on Mathematics, Actuarial Science, Computer Science and Statistics (MACS), 2019, pp. 1-7, doi: 10.1109/MACS48846.2019.9024768.
[56] Kumar, T., Pandey, B., Mussavi, S.H.A. et al. CTHS Based Energy Efficient Thermal Aware Image ALU Design on FPGA. Wireless Pers Commun 85, 671–696 (2015). https://doi.org/10.1007/s11277-015-2801-8
[57] Saeed, Soobia, N. Z. Jhanjhi, Mehmood Naqvi, and Mamoona Humayun. "Analysis of software development methodologies." International Journal of Computing and Digital Systems 8, no. 5 (2019): 446-460.
[58] M. Humayun, N. Jhanjhi, M. Alruwaili, S. S. Amalathas, V. Balasubramanian and B. Selvaraj, "Privacy Protection and Energy Optimization for 5G-Aided Industrial Internet of Things," in IEEE Access, vol. 8, pp. 183665-183677, 2020, doi: 10.1109/ACCESS.2020.3028764.

Matasem Saleh is a Ph.D. scholar at Taylor's University, Malaysia. There, he is working as a researcher in the area of Drone Detection Systems, Privacy Protection and IoT security. He has worked in the industry for a decade as Telecom Project Manager in EmaarAltelal, Saudi Arabia. He obtained his MSc in Computer Engineering from the University of Engineering and Technology, Pakistan, in 2008, where he developed FlocARe, an open-source network management software.

Dr. Noor Zaman Jhanjhi (NZ Jhanjhi) is currently working as Associate Professor, Director Center for Smart society 5.0 [CSS5], and Cluster Head for Cybersecurity cluster, at the School of Computer Science and Engineering, Taylor's University, Malaysia. The cybersecurity research cluster has extensive research collaboration globally with several institutions and professionals. Dr Jhanjhi serves as Associate Editor and Editorial Assistant Board for several reputable journals, such as PeerJ Computer Science, Frontier in Communication and Networks. He received Outstanding Associate Editor for IEEE ACCESS for 2020, PC member for several IEEE conferences worldwide, and guest editor for several reputed indexed journals. Active reviewer for a series of top-tier journals has been awarded globally as a top 1% reviewer by Publons (Web of Science). He has high indexed publications in WoS/ISI/SCI/Scopus, and his collective research Impact factor is more than 500 points. He has several international Patents on his account, including Australian, German, Japan, etc. edited/authored more than 40 research books published by world-class publishers, including Springer, Taylors and Frances, Willeys, Intech Open, IGI Global USA, etc. He has great experience supervising and co-supervising postgraduate students, and more than 20 PG scholars are graduated under his supervision, and an ample number of current PG students are under his supervision. He is an external Ph.D./Master thesis examiner/evaluator for several universities globally. He has completed more than 30 internationally funded research grants successfully. He has served as a Keynote/Invited speaker for more than 30 international conferences globally, presented several Webinars worldwide, chaired international conference sessions, and provided Consultancy on several projects internationally. He has vast experience of academic qualifications including ABET, NCAAA, and NCEAC for 10 years. His research areas include Cybersecurity, IoT security, Wireless security, Data Science, Software Engineering, and UAVs.

Dr. Azween Abdullah is currently working with Taylor's University. He is currently a Professional Development Alumnus of Stanford University and MIT. His work experience includes 30 years as an academic in institutions of higher learning and as the Director of research and academic affairs at two institutions of higher learning, the Vice-President for educational consultancy services, 15 years in commercial companies as a Software Engineer, a Systems Analyst, and as a Computer Software Developer and an IT/MIS consultancy and training.

Raazia Saher is a lecturer at King Faisal University in the Department of Computer Science and Information Technology. She has over ten years of academic experience in this prestigious institution. She has a master's degree in electrical engineering and is a registered professional engineer in Pakistan Engineering Council. She acquired specialized skills in Next Generations Networks & Soft Switches while working as an operational engineer in Pakistan Telecommunications Company Limited.