Society for Science and Education
United Kingdom

TNC Transactions on Networks and Communications

Volume 7, No. 1
ISSN: 2054 -7420

# Merging Securely M2M Protocols, Internet of Things and Cloud Computing

**Dina Darwish**
*The International Academy for Engineering and Media Science, Egypt*
dina.g.darwish@gmail.com

**ABSTRACT**

The Internet of Things provides new ways for communication through the Web world using object-enabled networks. At the same time, M2M devices intercommunication and their communication through the web if they were connected to the Internet, presents new challenges, especially in security, that traditional communication models have not yet fully solved. Because of their inborn un-watched, minimal effort and mass-sent nature, M2M devices, and remote communication architectures and solutions for these devices, would encapsulate new dangers in security. These threats are not fully faced by use of security technologies and methods implemented in existing wireless devices, cellular networks or WLANs. The use of cloud computing gives a convenient, on demand and scalable network access to a shared pool of configurable computing resources and devices. This paper concentrates on a secure method to integrate the M2M protocols with the Internet of Things (IoT) and Cloud Computing under the name of Secure Machine-to-Internet Clouding (SM2IC) architecture. The secure design for integrating M2M protocols, along with IoT and cloud computing is proposed. To apply this design, an IoT enabled smart home scenario was examined to analyze secure communication between M2M devices and IoT applications. Also, the cloud computing is used to include different cloud applications, such as, IaaS, PaaS, and SaaS for monitoring the quality of service of M2M devices through IoT applications. Then, simulations were performed to test the proposed security technique, followed by conclusions and future work.

**Keywords**: Cloud computing; Internet of Things, M2M protocols, Secure Integration, Connected M2M Devices.

## 1 Introduction

Internet of Things (IoT) is considered as a technology aimed at providing customers with smarter services by linking different devices to the Internet and enabling these devices to exchange information with each other. IoT has been distinguished as a developing technology in numerous IT trend reports [1], and the number of IoT devices is proposed to increase [2,3]. It is suggested by some IT trend reports that the worldwide IoT market will be worth billions of dollars by 2022 [4]. The interconnection between the different kinds of IoT devices is a key issue for the achievement of IoT, in light of the fact that the numbers of IoT devices is developing ceaselessly. IoT standardization bodies have completed several endeavors to understand the interconnection issue. Numerous Web of Things (IoT) platforms were outlined and

**DOI:** 10.14738/tnc.71.5939
**Publication Date:** 02nd February 2019
**URL:** http://dx.doi.org/10.14738/tnc.71.5939

executed over the previous decade. In any case, most platforms were applied in light of particular solutions or created to address certain domain issues.

To interconnect different proprietary platforms and deliver common IoT services to customers, there is a strong need to create a standardized IoT platform. To face interoperability issues, seven standard development organizations (SDOs) started a global standards project named oneM2M [5], by providing scalable and interoperable IoT standards for communication of devices and services. The goal of oneM2M is to present a single horizontal service platform, that can be implemented in different industries to deliver smarter IoT services to users and to exchange and share data among IoT applications. Machine-to-machine (M2M) communication is one of the next frontiers in wireless communication. There exist a large number of possibilities, in terms of new use cases, services and applications, that is suggested to result from communication between M2M devices. M2M can present benefits for the production and market opportunities for various manufacturers of M2M devices and components, service providers, and network operators.

Due to the huge number of M2M devices expected to be used, in a highly distributed network, enforcement of traditional security methods will not be practical because of the high cost of implementation of these devices. Also, deploying the conventional centralized IT security network model, protected by a firewall, is challenged by the need for a dispersed model, so, de-centralized methods for realizing security must be accessed. The growing direction towards using de-centralized systems creates a lot of situations in which enforcement of security, is accompanied by a controlled risk. The principles of how to enforce security embraced by traditional concepts of access control are being changed by a shift to implement “trust.” An entity is considered “trusted” if it behaves correctly as expected to achieve its intended goal. By including pieces of the enforcement tasks to trusted elements distributed in a system, trust relationships can be created. This is the most important part in the organizational method of separate tasks within IT security. This security model, which is balanced between trust and enforcement, produces a useful, practical and scalable approach and, can be used for M2M communication.

Security is one of the main problems in any information system, including M2M systems. With a big market for M2M devices and networks, M2M systems require to be properly designed and implemented. Many applications envisioned for M2M can be done if security is properly considered from the beginning. There is a need for good security mechanisms and procedures, also, various characteristics of M2M systems and applications may constitute challenges to the design of useful security mechanisms.

A few troubles, for example, the support of heterogeneous communication advancements and protocols for communication between M2M devices, the restrictions on equipment of numerous M2M detecting and actuating platforms, and security desires from clients require to be recognized. Numerous lessons and specialized solutions have been achieved from research in many fields, for example, mobile ad hoc networks [6] [7] or remote sensor networks [8], but, M2M frameworks still need new strategies in security. The employment of different wired and wireless communication innovations guided by the utilization of a typical service platform decides the careful assessment of the applied cryptographic algorithms. The support of communication needs proper distinguishing strategies. The properties and asset restrictions of M2M systems form difficulties to the plan of suitable security innovations, that can deal with distinctive detecting and actuating M2M devices. Contingent upon security desires for the M2M systems and applications, clients will require systems that permit the control of how much individual data is anchored, while certain applications will require a specific level of individual data to be ensured [9].

Cloud computing is depicted by the National Institute of Standard and Technologies (NIST) [10] as follows: "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction." The fundamental thought behind Cloud computing began to pick up fame after that Google's Chief Eric Shmidt utilized it in 2006 [11], and along the most recent years, Cloud computing has influenced IT industry. The presence of virtually unlimited storage and processing capabilities at low cost has created a new computing model, inside which virtual resources can be used on-demand.

Cloud computing [12] provides also a new method for design, development, test, deployment, run and maintenance of applications on the Internet. It is required to take care of running operating systems, networks, load balancing, routers, firewalls, and storage by the application developer, and at the same time, integrating these things and, enabling them to interact with the system. Also, it is important to take care by the developer of scalability, because it determines how the application can fit many geographically distributed users. The Cloud user; developer or consumer, can reach the Cloud services over the Internet, and the Cloud users must pay for time and services they are in need. The Cloud can also be expanded to implement large numbers of service requests. Cloud computing considers the micro-lifecycle management of applications, and enables application managers to concentrate on application design and surveillance. The Cloud computing platform is composed of different services for developing, testing, running, deploying, and maintaining applications on the Cloud. This direction for delivering services over the Internet, has been widely used by large companies, such as, Amazon, Google and Facebook and so on to gain both economic and technical benefits. Cloud Computing is considered as a disruptive technology with huge impact on the delivery of Internet services as well as for the IT sector.

The Internet of Things and Cloud computing are both considered as emerging technologies and they possess their own features. Things are connected to their virtual representations on the Internet and are reached through the Internet (i.e. Things as services) [13]. Cloud computing implements the utility model, which allows end-users to use and consume services in an efficient and pay-per-use way.

However, various technical and business-related issues are still unsolved. Certain issues have been determined for each service model, these issues are related to security, privacy, and service-level agreements, which need to be addressed for users [14]. Moreover, the lack of standard APIs makes extracting code and data from a site difficult for customers. Also, public Cloud customers are exposed to price increases, reliability problems or even to providers going out of business.

In this paper, a secure approach for integrating M2M protocols with the internet of things through cloud computing, namely secure machine-to-internet clouding (SM2IC) is proposed, and tested using the appropriate software tools. In section 2, a background on work done in cloud computing, Internet of Things and M2M communication was presented as well as the need for their integration. In section 3, description of current M2M protocols adopted by ETSI. In section 4, a detailed description of the proposed secure machine-to-internet clouding architecture is given. In section 5, description of simulation environment was made. In section 6, description of parameters used during simulation was done. In section 7, experimental results were provided. In section 8, conclusions and future work were given. Finally, references were provided.

## 2 Merge of Cloud computing, Internet of Things, and M2M Communication

In the following sections, few important characteristics of Cloud are going to be described. The design of Cloud can be partitioned into four layers: datacenter (equipment), infrastructure, platform, and application [11]. Every one of them is considered as a service for the layer above and as a consumer for the layer beneath. By and by, Cloud services can be gathered in three fundamental categories: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS). SaaS is related to the monitoring of applications running on Cloud environments. Applications can typically be reached by a thin client or a web browser. PaaS is related to platform-layer resources (such as, operating system support, software development frameworks, etc.). IaaS is related to delivering processing, storage, and network resources, enabling the consumer to manage the operating system, storage and applications. It has attracted the biggest interest so far. There are different types of Clouds' deployment that have been determined [10, 11], as mentioned in the following: (i) Private Cloud designed for use by a single organization, typically possessed, directed, and operated by the organization itself; (ii) Community Cloud – designed for use by a specific community of consumers that have common concerns; (iii) Public Cloud – designed for open use by the public; (iv) Hybrid Cloud – is composed of two or more distinct Cloud infrastructures (private, community, or public); (v) Virtual Private Cloud – designed to address issues related to public and private Clouds, benefiting from advantage of virtual private network (VPN) technologies for enabling business owners to setup desired network settings (such as, security, topology, and so on).

Cloud computing model frees the business owner from the need to invest in the infrastructure, enabling him to rent resources according to his needs and only pay for the usage, then, it becomes attractive. Also, it enables decreasing operating costs, as service providers do not have to rent capacities according to peak load, and resources are left when service demand becomes low. In addition to these economic advantages, Cloud computing provides a number of technical benefits, such as, energy efficiency, optimization of hardware and software resource utilization, elasticity, performance isolation, and flexibility. The two terms of Cloud and IoT have evolved rapidly and independently. These terms are different from each other and, their characteristics are complementary, as Table [15] shows.

**Table 1. Complementary aspects of Cloud and IoT**

| | **IoT** | **Cloud** |
|---|---|---|
| **Displacement** | pervasive | centralized |
| **Reachability** | limited | ubiquitous |
| **Components** | Real world things | Virtual resources |
| **Computational capabilities** | Limited | Virtually unlimited |
| **Storage** | Limited or none | Virtually unlimited |
| **Role of the Internet** | Point of convergence | Means for delivering services |
| **Big data** | source | Means to manage |

For this reason, many researchers have suggested complementary characteristics of cloud and IoT, and have proposed integration, generally to obtain benefits in specific application scenarios [16, 17]. Generally, IoT can benefit from the virtually unlimited capabilities and resources of Cloud to compensate its technological limitations (such as, storage, processing, communication). Cloud can provide an efficient solution for IoT service management and composition as well as for applying applications and services that benefit from the things or the data generated by them. Also, cloud can exploit IoT by expanding its scope to manage real world things in a more distributed and dynamic manner, and for presenting new services in a large number of real life cases. Cloud can deliver the intermediate layer between the things and the applications, preventing all the complexity and functionalities necessary to use the latter. This has impact on future application design, because information collecting, executing, and transmission will create new challenges, especially in a multi-cloud environment. It is believed that Cloud fills some gaps of IoT (such as, the limited storage). And, some see IoT filling gaps of Cloud (such as, the limited scope). Most of these drivers pushing cloud and IoT integration fall in three categories that are communication, storage, and computation, while there exist other basic traversal drivers. IoT is characterized by a very high heterogeneity of devices, technologies, and protocols, but, it lacks different important characteristics such as scalability, interoperability, flexibility, reliability, efficiency, availability, and security. On the other hand, Cloud has proved to deliver them [18, 19], then, they can be identified as some of the main transversal drivers for cloud and IoT integration. There are two other transversal drivers, which are the ease of use and the reduced cost delivered by both users and providers of applications and services [19].

## 3 M2M communication and high level architecture

### 3.1 Background on M2M communication

M2M communication tries to reach the vision of connected things, or what is meant by Internet of Things (IoT) [20] [4], through a variety of possible uses in a world where intelligent applications provide a better and safer world. Also, the number of connected devices is rapidly increasing. International Data Corporation expects there will exist around 15 billion devices communicating over the network by the year 2015 [21], while Cisco Internet Business Solutions Group (IBSG) expects 25 billion devices connected to the Internet by 2015 and 50 billion by 2020 [22]. Machina Research white paper mentioned that by 2022, there will exist around 18 billion M2M connections in the world, up from approximately 2 billion today [23]. Ericsson claims that their vision of more than 50 billion connected devices by 2020 may appear realizable and within reach using the right approach [24]. Due to this rapid expansion, the concept of M2M communication is having more and more significance. Interoperability, between devices based on various access network technologies (e.g. mobile (2G/3G/4G), Wi-Fi, Bluetooth), with different platforms and data models is still very limited.

M2M (Machine-to-Machine) communication is initiated between two or more entities without any direct human intervention [25]. Actors in this environment are broad range of communication capable devices, such as, computers, mobile phones, tablets, a variety of sensors, actuators, pieces of industrial and medical equipment, and other everyday devices [26].

### 3.2 M2M high level architecture defined by ETSI

ETSI's work determined a high-level architecture view describing all constituents of M2M systems, their roles, and relationships. The high-level architecture of M2M system is composed of two main parts, which are Device and Gateway Domain, and a Network Domain, as shown in Figure 1 [27]. The device and gateway domain is consisting of the following elements:

- M2M Device: executes M2M Device Applications (DA) using M2M Device Service Capabilities Layer (DSCL).

- M2M Gateway: executes M2M Gateway Applications (GA) using M2M Gateway Service Capabilities Layer (GSCL).
- M2M Area Network: conveys connectivity based on Personal or Local Area Network technologies (e.g. ZigBee, Bluetooth) between M2M devices and M2M gateways.

The network domain is comprising of the following components:

- M2M Access Network: empowers M2M devices and M2M gateways to communicate with the Core Network. It can rely upon any of the following existing access network solutions: Digital Subscriber Line (DSL), satellite, GSM EDGE Radio Access Network (GERAN), Universal Terrestrial Radio Access Network (UTRAN), evolved UTRAN (eUTRAN), Wi-Fi (IEEE 802.11), and Worldwide Interoperability for Microwave Access (WiMAX), that can be used for M2M communication when needed.
- M2M Core Network: allows interconnection with other networks, delivers IP connectivity or other connectivity choices, service and control functions, and roaming. Similar to access network, it can depend on different existing core networking (CN) solutions (3GPP CN, ETSI Telecoms & Internet converged Services & Protocols for Advanced Networks (TISPAN) CN, and 3GPP2 CN) that can be changed to meet certain M2M communication requirements when needed.
- M2M Network Service Capabilities Layer (NSCL): delivers shared M2M functions by different M2M applications.
- M2M Applications: execute the service logic and implement M2M service capabilities available through open interfaces.
- M2M Network Management Functions: is composed of all the functions, such as, provisioning, supervision, and fault management needed to deal with access and core networks.
- M2M Management Functions: is composed of all the functions, such as, M2M Service Bootstrap Function (MSBF) implemented to simplify the bootstrapping of permanent M2M service layer security credentials needed to deal with M2M service capabilities in the network domain.

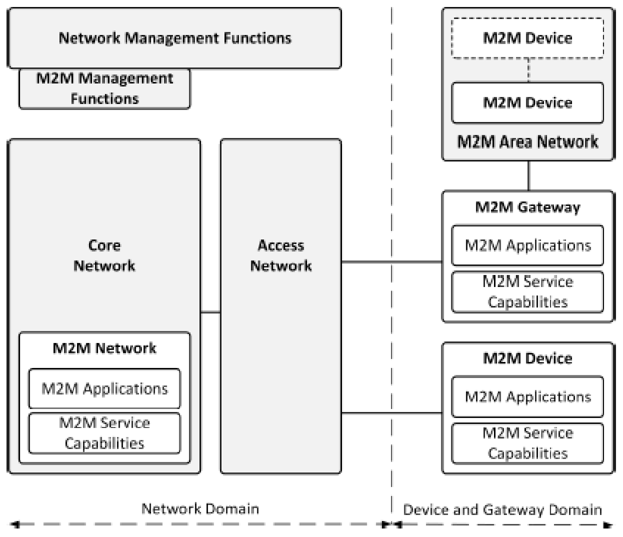


**Fig. 1. High-level architecture of M2M system defined by ETSI**

## 4 The proposed Secure Machine-to-Internet Clouding (SM2IC) security technique

### 4.1 Communication process in the proposed security technique

The idea proposed in this paper is based on presenting secure smart home connections between the user and his home devices. So, any user can open, shut down and monitor his home devices from outside his home using his smartphone or any similar device, besides, monitoring and controlling the quality of service delivered from his home devices from abroad.

The proposed technique provides a detailed description of how to initiate a secure connection from a user's device, such as, PC or tablet and so on to a central controller home device. This central controller device then sends signals to other desired home devices (e.g. microwave, TV, Lighting system and so on.) to control the status of these devices. The user's request communicates with the Internet using the hypertext transfer protocol, known as HTTP, then, the internet passes this request to the central device using the Https (or secure hypertext transfer protocol), also, the central controller device's hardware is Raspberry pi enabled, and this device can exchange data with the internet using Get and Post commands through Https. In its turn, the central controller device communicates in both directions with home devices previously mentioned using M2M network through one of the following technologies (Bluetooth, ZigBee, radio waves, Wi-fi ……. and so on).

The user's holding his device with cloud applications (such as, PaaS, IaaS and SaaS) receives a feedback from the home devices to the central device then to the Internet to his device about the completion of the secure connection and about the status of his home devices. The cloud applications residing in the user's device allow the user to monitor the quality of service (QoS) delivered through the whole process at the end home devices through giving precise measurements about the different parameters relating to the connection, internet, central device and controlled home devices, and this enables the user to control and modify the desired parameters according to what he expects.

The whole process starting from the connection initiation to the connection completion, must be secure and uncompromised. How security is achieved through this proposed security protocol, will be discussed in details through the following subsections. Fig.2 shows the whole connection process of the proposed security technique from the beginning to the end.

In Fig.3, the connection process of the proposed security technique is demonstrated focusing on the type of networks supporting M2M communication used and at the same time showing the different layers of the Internet of things. First, the user sends a request from his device containing cloud applications to change the status of his home devices. In this case, the user carrying his device represents the IoT application layer. The request traverses the Internet using Http protocol, then, the Internet forwards the user's request to M2M core network, which is responsible of connectivity with other networks, and includes both M2M applications and M2M service capabilities. Then, the request passes to the access network. The Internet, the M2M core network and the access network compose the IoT network layer. The request goes then to the M2M gateway, which contains both M2M applications and M2M service capabilities.

The M2M gateway represents the IoT service and application support layer. The user's request then reaches the M2M controller device, which contains M2M applications and M2M service capabilities. This

M2M controller device sends the user's request to the M2M Area Network, which works according to one of the following technologies (such as, Bluetooth, ZigBee, radio waves, Wi-fi …). In turn, the M2M Area Network forwards the user's request to the desired M2M devices, which include M2M applications and M2M service capabilities. The M2M controller device, M2M Area Network and M2M devices constitute the IoT device layer. Then, the feedback carrying the devices status is sent to the direction of the user, traversing all the preceding layers. The user can monitor and modify the home devices state using cloud applications according to his will. Then, the communication process in this technique occurs in both directions (indicated by lines with arrows in both directions, in Fig.3).

In the following subsections, the proposed security technique is going to be described in details. Fig.4 illustrates the proposed security technique for M2M communication through steps. In the first step, the user initiate a secure connection from his device, which contains cloud applications to monitor or modify the parameters of the whole process. The encryption and decryption processes are performed according to a proposed security technique in this paper, named Double Key Secure Internet (DKSI), and this new security technique is going to be explained in section 4.2.

The user's device must generate a connetion request number, composed of the user's device ID encrypted using the proposed Double Key Secure Internet (DKSI) tehnique and the key used to encrypt it, as well as, an authentication setting number, containing the user's password encrypted using the DKSI technique by the same key used to encrypt the user's device ID. The user's connection request traverses the Internet, the M2M core network, access network and M2M gateway to reach the M2M controller.

In the second step, the M2M controller checks the connection request number and the authentication setting number by decrypting the device's ID and the user's password using DSA or RSA technique with the first encryption key sent by the user. If the user's device ID and the user's password are verified, then, the controlller device can transport the requested tasks to the desired devices.

In the third step, the M2M controller device generates a temporary conection key, containing the controller device ID encrypted using the DKSI encryption technique by a new generated key from the M2M controller, then, the temporary connection key is encrypted for the second time using the user's first key generated at the beginning of the connection with the DKSI encryption technique. Then, the user's request containing connection keys passes from the M2M controller device to the M2M required devices through M2M area network.

In the fourth step, the required M2M devices have to check the temporary connection key containing the controller device ID and the second encryption key generated by the controller device, as well as the first encryption key sent from the user's device. First, the connection key is decrypted using the user's device generated key, then, it is decrypted for the second time using the second key generated by the controller device. Once, the controller device's ID is checked and verified, then, the desired devices status can be changed.

Also, each connected device has to create a new connection key, containing its ID encrypted with the DKSI technique using the second key, which was generated by the controller device, then, this connection key is encrypted for the second time using the user's device generated key. Then, the new connection key has to reach the controller device.

In the fifth step, the controller device checks the connected M2M devices IDs by decrypting the connected devices IDs using firstly the user's device generated key, then, secondly using the controller's device generated key with the DKSI technique. Once, their IDs are verified, then, the controller device encrypt its own ID and the connected M2M devices IDs separately using the user's device generated key with the DKSI technique forming two new connection keys, and then relay them to the user's device.

In the sixth step, the user decrypts the controller device's ID and the connected M2M devices IDs, using the key generated from his device with the DKSI technique to be verified. Then, the user can monitor the connected devices from his device and send a feedback through his device to change connected devices status. Apart from the different steps of the security technique, the desired devices status are sent from the user at the connection initiation, to the M2M controller device by passing through the Internet, the M2M core network, the access network, and the M2M gateway. Then, the M2M controller device retransmits the desired devices required status to the requested devices by passing through the M2M area network. Then, the requested devices resend back their encrypted IDs and their modified status to the M2M controller device again through the M2M area netwok. Finally, the M2M controller device resends the desired devices encrypted IDs and their modified status back to the user's device by traversing the M2M gateway, access network, M2M core network and the Internet. And, the user can modify the home devices status again as desired.

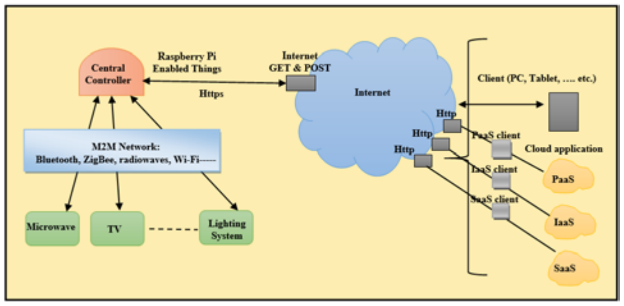


**Fig.2 Description of the whole connection process of the proposed security technique**

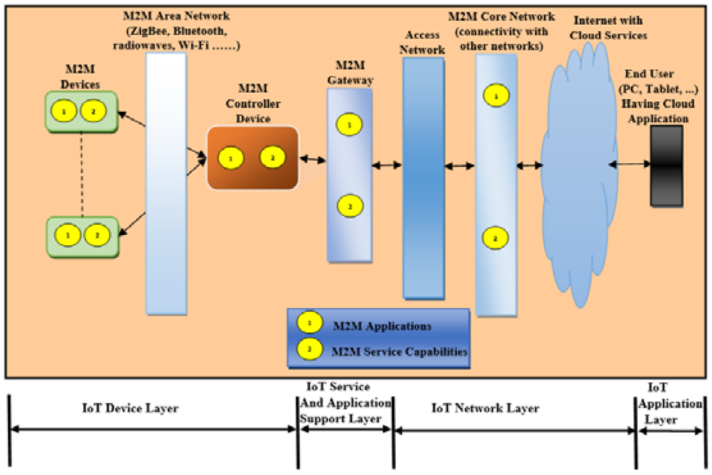
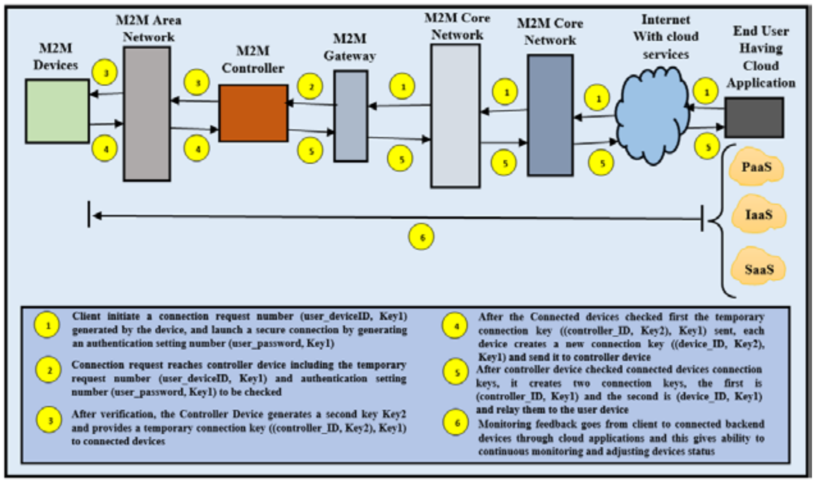

M2M Area Network (ZigBee, Bluetooth, radiowaves, Wi-Fi ......)
M2M Devices
M2M Controller Device
M2M Gateway
Access Network
M2M Core Network (connectivity with other networks)
Internet with Cloud Services
End User (PC, Tablet, ...) Having Cloud Application
M2M Applications
M2M Service Capabilities
IoT Device Layer
IoT Service And Application Support Layer
IoT Network Layer
IoT Application Layer

**Fig.3 The whole connection process of the proposed security technique focusing on the different networks types and the IoT layers**



1 Client initiate a connection request number (user_deviceID, Key1) generated by the device, and launch a secure connection by generating an authentication setting number (user_password, Key1)

2 Connection request reaches controller device including the temporary request number (user_deviceID, Key1) and authentication setting number (user_password, Key1) to be checked

3 After verification, the Controller Device generates a second key Key2 and provides a temporary connection key ((controller_ID, Key2), Key1) to connected devices

4 After the Connected devices checked first the temporary connection key ((controller_ID, Key2), Key1) sent, each device creates a new connection key ((device_ID, Key2), Key1) and send it to controller device

5 After controller device checked connected devices connection keys, it creates two connection keys, the first is (controller_ID, Key1) and the second is (device_ID, Key1) and relay them to the user device

6 Monitoring feedback goes from client to connected backend devices through cloud applications and this gives ability to continuous monitoring and adjusting devices status

**Fig.4 the proposed security technique steps for M2M communication**

## 4.2 Encryption and decryption techniques in the proposed security technique

### 4.2.1 Hash Function Used in the Proposed Security Technique

Most known cryptography techniques, such as RSA [28], DSA [29] or elliptic curve [30] encryption techniques utilize hash functions to ensure security of the user's data. Hash functions are created in one-way and can not be reversed or decrypted. In the proposed security technique (DKSI), the hash function SHA-2 was used. The SHA-2 is going to be described in the following section.

SHA-2 (Secure Hash Algorithm 2) [31] comprises of a set of cryptographic hash functions made by the United States National Security Agency (NSA). Cryptographic hash functions are considered as mathematical tasks that process digital data; by looking at the processed "hash" to a known and evaluated hash value, a person can estimate the data's integrity.

Also, evaluating the hash of a downloaded document and contrasting the outcome with a formerly created hash result can identify whether the download has been changed or altered. A key property of cryptographic hash functions is their resistance against collision; no one is capable of discovering two distinctive input values that deliver the same hash output. SHA-2 has huge changes from its antecedent, SHA-1.

The SHA-2 family is comprising of six hash functions with digests (hash values) that are 224, 256, 384 or 512 bits: SHA-224, SHA-256, SHA-384, SHA-512, SHA-512/224, SHA-512/256. SHA-2 was announced in 2001 by the National Institute of Standards and Technology (NIST) a U.S. federal standard (FIPS). Today, the best public attacks break preimage resistance for 52 rounds of SHA-256, and collision resistance for 46 rounds of SHA-256. The table 2 below show the most common properties for sha-256 hash function.

Table 2. the most common properties for sha-256 hash function

| Algorithm and variant | | Algorithm size (bits) | Internal state size (bits) | Block size (bits) | Max message size (bits) | Rounds | Operations | Security bits (Info) | Example Performance (MiB/s) | First published |
|---|---|---|---|---|---|---|---|---|---|---|
| SHA-2 | SHA-224<br>SHA-256 | 224<br>256 | 256<br>(8x 32) | 512 | $2^{64}$ - 1 | 64 | And, Xor, Rot, Add (mod $2^{32}$), Or, Shr | 112<br>128 | 139 | 2001 |

Based on its characteristics, Sha- 256 was chosen to be used as a hash function in the proposed technique, due to its relative proven strength against attacks and collisions.

#### 4.2.2 Proposed DKSI (named double key secure internet) encryption technique

In this paper, a new encryption technique to ensure security of data transactions is proposed. This technique implements SHA-2 (as a hash function) instead of SHA-1, because SHA-2 provides better security as explained in previous section 4.2.2. The proposed DKSI technique is performed on the user's ID, password and generated key. It depends on transforming the user's device ID into binary representation, then performing some transformations and logical operations on it, and retransforming it using a hash function along with some mathematical operations. The steps of the proposed DKSI encryption technique are described below as follows:

1. Take the user's device ID in combined representation using letters and numbers, for example, EAT56TYUI
2. Transform each letter or number to its ASCII code, then to its binary representation, composed of 8 bits; 0's and 1's
3. Generate the first user's key of a random 0's and 1's, but it must be the same length of the Device ID binary representation
   using Key = rand(1, len), where len is the length of the Device ID binary representation
4. ANDing the binary representation of the Device ID and the generated key
5. Shift to the left twice the output result and the generated key and put 2 0's at the left of both of them
6. The first two bits shifted from the left, can be: 00, 01, 10 and 11, and put them at the end of the right of both of the result and the key

Note: the output result represents the Device ID primary encryption. Steps 5 and 6 is performed for both the result from step 4 and the user generated key. The output length from step 6 is equal to the length of the binary representation of the Device ID plus two bits, and the length of the key in binary is increased by two bits.

7. Transform the primary encrypted device_id (Device ID') to numerical representation
8. Make hash function of the primary encrypted device_ID using SHA-2/256(Device ID')
9. Apply mathematical operations to the to the hashed primary encrypted device_ID after transforming it to hexadecimal format and using the logarithmic function, as follows, Final encrypted Device_id = [ (numerical (hashed Device_ID'))*log (len)] $^{K}$
10. Transform the user generated shifted key from its binary representation to numerical representation to become (key')
11. Apply some modifications on the user generated key before sending it, by New_Key = (K*key')/(M*P)

Assumptions for this encryption technique are described below:

- ☒ Where K, P and M are known numbers saved at the user's device, central controller, home devices,
- ☒ Also, the user_mobile_id and password are known to the central device
- ☒ The central device_id is known to the home devices
- ☒ The home devices ids are known to the the central device_id
- ☒ The central device_id and home devices ids are known to the user's mobile

Note: the user password is encrypted the same way as user's device ID following steps to 9 by using the same user generated key, and finally, the encrypted Device_ID, the encrypted password along with the transformed key are all sent to the central controller device for verification.

The proposed DKSI encryption technique would be applied later on central device ID, as well as home devices IDs by using the same 9 steps performed on the user's device_ID. At the central controller device, the received encrypted Device_ID and user's password are decrypted using the transformed user's key sent with them, once they are verified, the central controller device generates a second key using the same steps 3, 5, 6, 10 and 11 as performed for the user generated key. But, in this case, there exist double encryptions for the central device ID, by encrypting it using key 2 (central device generated key) then using key 1 (user's device generated key) implementing the same steps to 9 as done for the encryption of the user's Device_ID and password at the user's device first, and the central Device ID, along with key 1 and key 2 are sent to home devices for verification.

Then, at the home devices, once the central device _ID is verified after decrypting it, the home devices status are changed as required by the user, and the home devices IDs are encrypted using the same steps to 9 mentioned above using key2. Then, the encrypted home devices IDs are sent with key 2 back to the central controller device. Then, the central controller device checks the home devices IDs by decrypting them and comparing them to the IDs stored inside it, once, the home devices IDs are verified, the central controller device Id and the home devices IDs are encrypted by key 1 using the proposed DKSI technique steps to 9, and then, sent along with key 1 to the user's device to be verified. Finally, at the user's device, the received central controller device ID and home devices IDs are decrypted using key 1 to be checked and compared to the ones stored inside the user's device, once they are verified, a monitoring feedback can propagate easily from the user to the central controller device and the connected backend home devices to enable the user easily to monitor the status of these devices. The decryption process is going to be described in the following section.

### 4.2.3 Proposed DKSI (named double key secure internet) decryption technique

The inverse of the steps explained in the encryption is done in the decryption to recover the original user Device_ID and password, to check them against saved ones in the central device. The following decryption steps are reapplied every time there is a need for decryption in the proposed security DKSI technique. The decryption steps of the proposed DKSI security technique are described below:

Recover first the key by using these steps;

1. Apply reverse mathematical operations to that performed in encryption on the received key such as, Key'' = (M*P)*received_key /K;
2. Transform the received modified key (key'') from numerical representation to 8 bit binary representation.
3. Remove the leftmost two bits in the binary representation that were added during the encryption process.

4. Bring the rightmost two bits that were shifted during the encryption to the leftmost two bit locations, then, the original key was restored and the original Device_ID need to be found.

Recover second the original Device_ID by implementing these steps;

1. Apply the steps to 9 mentioned in the encryption above using the recovered key on the user's Device_ID, which is stored inside the central controller device to encrypt it, the result of the encryption is a vector.
2. Compare the newly encrypted user's Device_ID by the central controller device with the received encrypted user's Device_ID from the user's device, because the hash function SHA-2 cannot be reversed, so we have to repeat the DKSI encryption steps on the stored user's Device_ID. If the newly encrypted user's Device_ID matched the received encrypted user's Device_ID, then, the user's Device_ID is verified, and same steps of encryption to 9 are applied on the user's password stored inside the central device using the recovered key, then, the newly encrypted password is compared to the received encrypted user's password to be verified. Once both the user's Device_ID and password are verified, then, encrypt the central controller device ID using key2 generated by it, then, by using the recovered key 1, and send them to the home devices for verification, and so on as explained earlier in the DKSI technique.

Note: the decryption steps are repeated every time there is a need during the DKSI security technique. Fig. 5 illustrates encryption and decryption processes of the DKSI security technique.

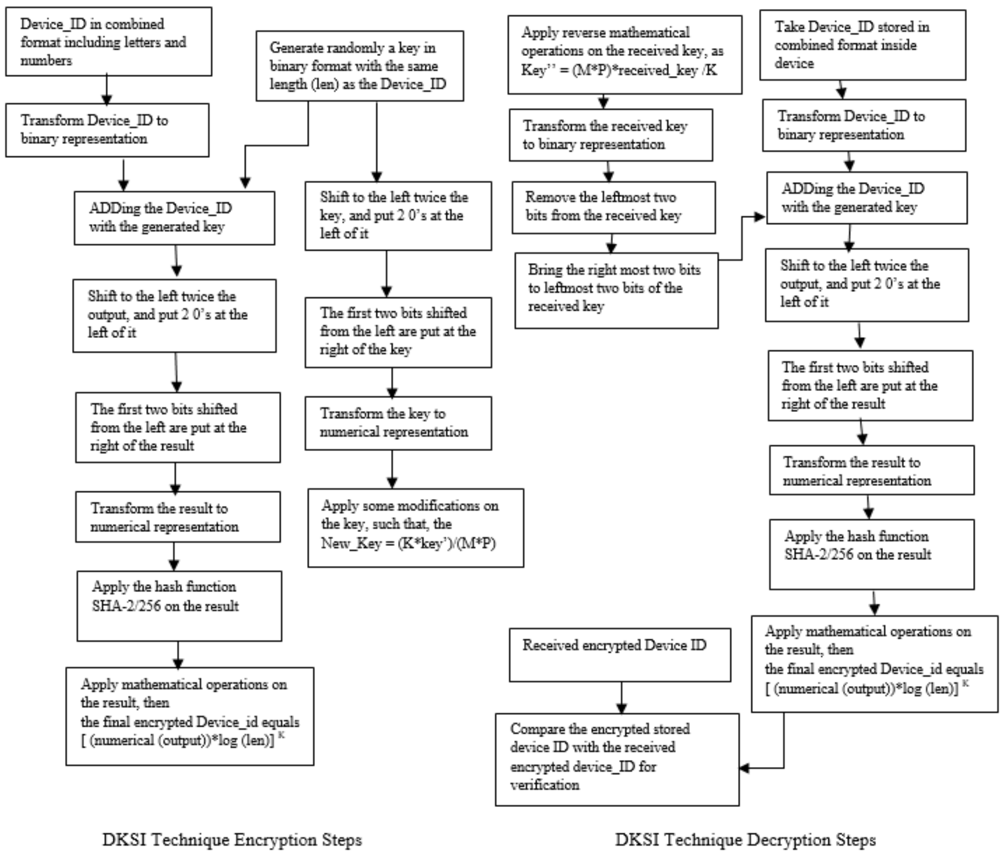

Device_ID in combined format including letters and numbers
Transform Device_ID to binary representation
Generate randomly a key in binary format with the same length (len) as the Device_ID
ADDing the Device_ID with the generated key
Shift to the left twice the key, and put 2 0's at the left of it
Shift to the left twice the output, and put 2 0's at the left of it
The first two bits shifted from the left are put at the right of the key
The first two bits shifted from the left are put at the right of the result
Transform the key to numerical representation
Transform the result to numerical representation
Apply some modifications on the key, such that, the New_Key = (K*key')/(M*P)
Apply the hash function SHA-2/256 on the result
Apply mathematical operations on the result, then the final encrypted Device_id equals [ (numerical (output))*log (len)] ^K
Apply reverse mathematical operations on the received key, as Key'' = (M*P)*received_key /K
Transform the received key to binary representation
Remove the leftmost two bits from the received key
Bring the right most two bits to leftmost two bits of the received key
Take Device_ID stored in combined format inside device
Transform Device_ID to binary representation
ADDing the Device_ID with the generated key
Shift to the left twice the output, and put 2 0's at the left of it
The first two bits shifted from the left are put at the right of the result
Transform the result to numerical representation
Apply the hash function SHA-2/256 on the result
Apply mathematical operations on the result, then the final encrypted Device_id equals [ (numerical (output))*log (len)] ^K
Received encrypted Device ID
Compare the encrypted stored device ID with the received encrypted device_ID for verification
DKSI Technique Encryption Steps
DKSI Technique Decryption Steps

**Fig.5 Encryption and decryption of the DKSI security technique**

## 5 Simulation Environment

### 5.1 MATLAB Simulink

Simulink [32] is made of a block diagram environment for multidomain simulation and Model-Based Design. It allows simulation, automatic code generation, and continuous test and verification of embedded systems. Simulink possesses a graphical editor, adaptable block libraries, and solvers for modeling and simulating dynamic systems. It is converged with MATLAB, enabling the user to integrate MATLAB algorithms into models and generate simulation results to MATLAB for investigation and assessment. Engineers everywhere use Simulink to realize their ideas off the ground, including reducing fuel emissions, developing safety-critical autopilot software, and designing wireless LTE systems.

Simulink delivers built-in support for prototyping, testing, and executing models on low-cost target hardware, such as Arduino, LEGO MINDSTORMS NXT, and Raspberry Pi. A client can create algorithms in Simulink for control systems, robotics, sound processing, and computer vision applications and see them working progressively.

Using Simulink Desktop Real-Time, a user can run Simulink models in real time on Microsoft Windows PCs and MacOS and link to a range of I/O boards to create and manage a real-time system. To process a model in real time on a target computer, Simulink Real-Time for Hardware-In-the-Loop (HIL) simulation, rapid control prototyping, and other real-time testing applications can be used.

With Simulink, the user can develop algorithms and models, and process them on low-cost embedded hardware including Arduino, LEGO MINDSTORMS NXT and EV3, and Raspberry Pi. Development for a range of embedded hardware applications such as control systems, robotics, audio processing, and computer vision can be performed. Simulink support for low-cost embedded hardware is existing in student and home-use versions.

### 5.2 Raspberry pi 3 general specifications

The Raspberry Pi [33] is composed of a series of small single-board computers developed in the United Kingdom by the Raspberry Pi Foundation to be used in teaching of basic computer science in schools and in developing countries. The original model became far more popular than expected, selling outside of its target market for uses such as robotics. Peripherals (including keyboards, mice and cases) are not included with the Raspberry Pi. A few accessories anyway have been incorporated into several official and informal bundles. A few generations of Raspberry Pis have been discharged. Raspberry Pi 3 Model B was discharged in February 2016 and was packaged with on-board WiFi, Bluetooth and USB boot capacities. As of January 2017, Raspberry Pi 3 Show B was the freshest mainline Raspberry Pi.

All Raspberry Pi models possess a Broadcom system on a chip (SoC), which includes an ARM compatible central processing unit (CPU) and an on-chip graphics processing unit (GPU, a VideoCore IV). CPU speed varies from 700 MHz to 1.2 GHz for the Pi 3, and on board memory varies from 256 MB to 1 GB RAM. Secure Digital(SD) cards are utilized to store the operating system and program memory in either the SDHC or MicroSDHC sizes. Most boards possess between one and four USB slots, HDMI and composite video output, and a 3.5 mm phono jack for audio. Lower level output is provided by a number of GPIO pins which support common protocols like $I^2C$. The B-models have an 8P8C Ethernet port, and the Pi 3 has on board Wi-Fi 802.11n and Bluetooth. The Raspberry Pi 3, has a quad-core Cortex-A53 processor. This model was expected to be highly dependent upon task threading and instruction set use. The Raspberry Pi 3 is equipped with 2.4 GHz WiFi 802.11n (150 Mbit/s) and Bluetooth 4.1 (24 Mbit/s) based on Broadcom BCM43438 FullMAC chip with no official support for Monitor mode but used through unofficial firmware patching and also has a 10/100 Ethernet port.

The Raspberry Pi Foundation recommends the use of Raspbian, a Debian-based Linux operating system. Other third party operating systems available via the official website are Ubuntu MATE, Snappy Ubuntu Core, Windows 10 IoT Core, RISC OS and specialised distributions for the Kodi media center and classroom management. It presents Python and Scratch as the main programming language, with support for many other languages. The default firmware is closed source, while an unofficial open source is available. Many other operating systems can also execute on the Raspberry Pi.

## 5.3 Simulink Support Package for Raspberry pi capabilities and features

Simulink Support Package for Raspberry Pi empowers you to create algorithms in Simulink, a block diagram environment for designing dynamic systems and creating algorithms, and execute them independently on your Raspberry Pi. The support package broadens Simulink with blocks for adjusting your Raspberry Pi, sending and accepting UDP packets, and reading and writing information from sensors. This includes writing information to the free ThingSpeak information aggregation service for Internet of Things applications.

In the wake of making your Simulink demonstrate, you can simulate it, tune algorithm parameters until you get it just right, and download the finished algorithm for independent execution on the device. With the MATLAB Function block, you can incorporate MATLAB code into your Simulink model. Using Simulink support package for Raspberry Pi, you compose the algorithm in Simulink and implement it to the Raspberry Pi utilizing automatic code generation. Execution is then performed on the Raspberry Pi. Utilizing Simulink for Raspberry Pi programming empowers you:

- Create and mimic your algorithms in Simulink and utilize automatic code generation to execute them on the device
- Incorporate signal processing, control configuration, state logic, and other advanced math and engineering schedules in your Raspberry Pi programming projects
- Intelligently tune and advance parameters as your algorithm runs on your Raspberry Pi

Notwithstanding utilizing Simulink Support Package for Raspberry Pi, you can deliver clear and convenient C code from MATLAB algorithms and actualize it on a Raspberry Pi utilizing Raspberry Pi support from MATLAB Coder. Simulink Support Package for Raspberry Pi influences you to create algorithms that execute independent on your Raspberry Pi. The support package broadens Simulink with blocks to guide Raspberry Pi digital I/O and read and write information from them. In the wake of building up your Simulink model, you can mimic it and download the finished algorithm for independent execution on the device. One particularly useful (and unique) capability provided by Simulink is the ability to tune parameters live from your Simulink model while the algorithm executes on the hardware.

## 5.4 Laptop specifications

Simulation of the smart home scenario was accomplished on a Dell laptop model Inspiron 15500 series, having the following specifications illustrated in table m below. A 64-bit operating system was used in simulation, because Matlab R2017a must be installed on 64-bit operating system machine. The Laptop used in simulation has 8 GB RAM and 2.40 GHz speed, since Matlab R2017a requires a computer with good speed and acceptable RAM. The rest of the Dell Laptop specifications is mentioned in the following table 3.

**Table 3. Operating System Specifications**

| | |
|---|---|
| Laptop Model | Dell model Inspiron 15 5000 series includes Nvidia Geoforce and Ubuntu |
| Laptop processor | Intel(R) Core(TM) i7-5500 CPU @ 2.40GHz 2.40 GHz |
| System Type | 64 bit operating system |
| Maximum possible array MATLAB can create | 12445 MB |
| Memory available for all arrays and data | 12445 MB |
| Memory used by MATLAB | 2402 MB |
| Computer Physical Memory (RAM) | 8102 MB |
| Physical memory and paging system | 14263 MB |

## 5.5 Smart home scenario to be implemented using MATLAB

The proposed security technique to be implemented in a smart home scenario can be built using MATLAB R207a Simulink Raspberry pi toolbox. The scenario begins when the user clicks from his android mobile phone a button to request the change of home devices status or switching them ON using android toolbox blocks; which provides blocks enabling user interaction inside his android enabled smartphone. Raspberry pi toolbox provides blocks to simulate a Wi-Fi UDP send and receive blocks for wireless communication. So, the proposed security technique is composed of three main parts; the first part represents the user clicking on his smartphone button to switch on/off home devices, then the request is sent wirelessly to the Raspberry pi enabled controller device. The second part represents the raspberry pi enabled controller device sending wirelessly the user request to the raspberry pi enabled home devices, after verifying the coming data. The third part represents the raspberry pi enabled home devices after verifying received data, change the status of home devices; here switching Raspberry pi LED ON/OFF. Figure 6 represents the main blocks of the smart home scenario to be implemented using MATLAB Simulink illustrating how they exchange data wirelessly. The figures of three parts constituting the proposed security technique are provided in the experimental results section.

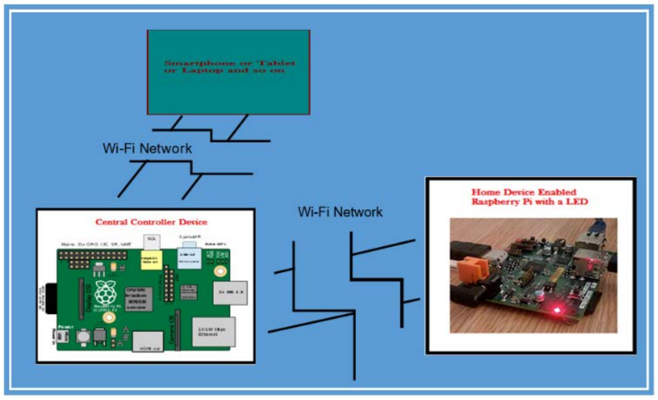

Smartphone or Tablet or Laptop and so on
Wi-Fi Network
Central Controller Device
Wi-Fi Network
Home Device Enabled Raspberry Pi with a LED

**Fig.6 Real life scenario to be implemented using MATLAB Simulink blocks and Raspberry pi toolbox**

## 6 Description of parameters used in simulation

The parameters evaluated during simulations are going to be described in the following subsections.

### 6.1 Response time

Real-time systems exist in the world around us. A modern car is considered as an example of a real-time system. Any person using the car will most likely want to have guarantees about the car's behavior. If the brakes need to be replaced in a nearby future, a lamp should indicate this, and not by the user who declares that there no longer exists any braking effect. A task is a program that performs some service or functionality in the system, like checking the brakes. A task's reaction time can be portrayed as the required time for checking the brakes. To be fit for giving ensures in continuous real-time systems, one must know the response times of tasks. If a message is sent from a source to a destination, the response time can be calculated as the time the message takes from the source to the destination. The factors are considered for evaluating the response time are the bandwidth of the network and the message size, and both are determined by symbols as follows:

$M$: the message size
$B_{wireless}$: the bandwidth of the network

$$\textit{Response time} = M / B_{wireless}$$

### 6.2 Memory consumption

To calculate the total memory consumed [34], it is necessary to calculate the number of concurrent users, the domain of the system, the amount of memory required per user, the buffer cache compensation, the number of virtual machines allocated and the system excess rate to estimate the memory size based on the data obtained through the investigation. Also, the system domain contains spaces for the OS, DMBS, engine, middleware engine, and other utilities. The result of the estimation of the memory amount can be expressed as follows.

$$\textit{Total Consumed Memory} = (T1 + M * q) * p * o$$

T1= The total memory for the system domain
M = The number of the virtual machines allocated
q = The amount of the required memory per user
p = Buffer cache compensation
o = system excess rate
By using the formula above, the result can be calculated as (384 + 959 * 2) * 1. 2 * 1. 3 = 3,591 MB. In consideration of the unit of memory expansion, the amount will be estimated as to b be 8,192MB.

### 6.3 Power consumption of transmitted signals

There exist cell phone base station tower networks across many nations globally, but there are still many areas within those nations that do not have good reception. Some provincial regions are probably not going to be successfully covered, in light of the fact that the cost of raising a cell tower is too high for just a couple of clients. In high reception zones, it is discovered that basements and the insides of vast buildings have poor reception. Weak signal strength can likewise be caused by damaging interference of the signals from nearby towers in urban territories, or by the construction materials used in few buildings, bringing about fast weakening of signal strength. Vast buildings for example warehouses, hospitals and manufacturing plants regularly have no usable signal more distant than a couple of meters from the outside walls. This is especially valid for the networks, which work at higher frequency, in light of the fact that these signals are lessened quickly by mediating obstacles, despite the fact that they can utilize reflection and diffraction to go around obstacles. The estimated received signal strength [35] in a mobile device can be calculated as follows:

$$dBm_e = -113.0 - 40.0\ \log_{10}\left(\frac{r}{R}\right)$$

More general, you can take the path loss exponent into consideration:

$$dBm_e = -113.0 - 10.0\ \gamma\ \log_{10}\left(\frac{r}{R}\right)$$

If the mobile device exists at *cell radius* distance from the cell tower, the *received power* is calculated as −113 dBm. The effective path loss is based on the frequency, the topography, and the environmental conditions. Actually, one could use any known *signal power* $dBm_0$ at any distance $r_0$ as a reference:

$$dBm_e = dBm_0 - 10.0\ \gamma\ \log_{10}\left(\frac{r}{r_0}\right)$$

Table 4 ilustrates the parameters of the received power signal.

**Table 4. The parameters of the received power signal**

| Parameter | Description |
|---|---|
| $dBm_e$ | Estimated received power in mobile device |
| −113 | Minimum received power |
| 40 | Average path loss per decade for mobile networks |
| *r* | Distance mobile device - cell tower |
| *R* | Mean radius of the cell tower |
| *γ* | Path loss exponent (average value of 4 for mobile networks) |

## 6.4 Bit error rate

In digital transmission, the number of **bit errors** [36] is considered as the number of received bits of an information flow over a communication channel that have been altered because of noise, interference, distortion or bit synchronization errors. The **bit error rate** (**BER**) is the number of bit errors per unit time. The **bit error ratio** (also **BER**) is the number of bit errors partitioned by the total number of exchanged bits amid an examined time interval. Bit error ratio is a unitless performance measure, frequently introduced as a percentage. The **bit error probability** $p_p$ is the normal estimation of the bit error ratio. The bit error ratio can be resolved as a rough estimate of the bit error probability. This estimate is precise for quite a long time interval and a high number of bit errors.

Estimating the bit error ratio empowers individuals to choose the convenient forward error rectification codes. Since most such codes rectify just bit-flips, but not bit-inclusions or bit-erasures, the Hamming distance metric is considered as the proper technique to gauge the number of bit errors. Numerous FEC coders additionally constantly measure the current BER. A more broad technique for estimating the number of bit errors is the Levenshtein distance. The Levenshtein distance measurement is more appropriate for estimating raw channel performance before frame synchronization, and when utilizing error correction codes created to amend bit-inclusions and bit-erasures, such as Marker Codes and Watermark Codes. The BER is depicted as the likelihood of a bit distortion due to electrical noise $w(t)$. On the account of a bipolar NRZ transmission, we have $x_1(t) = A + w(t)$ for a "1" and $x_0(t) = -A + w(t)$ for a "0". Every one of $x_1(t)$ and $x_0(t)$ has a period of $T$. Realizing that the noise has a bilateral spectral density $\frac{N_0}{2}$,

$x_1(t)$ is $\mathcal{N}\left(A, \frac{N_0}{2T}\right)$

and $x_0(t)$ is $\mathcal{N}\left(-A, \frac{N_0}{2T}\right)$.

Coming back to BER, we have the likelihood of a bit distortion $p_e = p(0|1)p_1 + p(1|0)p_0$.

$p(1|0) = 0.5 \operatorname{erfc}\left(\frac{A+\lambda}{\sqrt{N_o/T}}\right)$ and $p(0|1) = 0.5 \operatorname{erfc}\left(\frac{A-\lambda}{\sqrt{N_o/T}}\right)$

where $\lambda$ is considered as the threshold of choice, set to 0 when $p_1 = p_0 = 0.5$.

We can use the average energy of the signal $E = A^2T$ to suggest the last expression:

$p_e = 0.5 \operatorname{erfc}\left(\sqrt{\frac{E}{N_o}}\right)$.

## 6.4 Strength of the password

### 6.4.1 Password Cracking

In cryptanalysis and computer security, password cracking [37] is considered as the way toward recuperating passwords from information that have been stored or transferred by a computer system. A common technique (brute-force attack) is to attempt surmises over and again for the password and check them against an accessible cryptographic hash of the password.–The objective of password cracking can be to enable a client to recoup a forgotten password (introducing a totally new password is to a lesser degree a security hazard, but it needs System Administration privileges), to get unauthorized access to a system, or as a preventive measure by system executives to check for easily crackable passwords. On a file-by-file premise, password cracking is made to gain access to digital evidence, for which a judge has empowered access however the specific file's access is confined. The best cracking password techniques are; dictionary attack, brute force attack, rainbow table attack, blogs, phishing, social engineering, malware, offline cracking, shoulder surfing, spidering, guess, and port scan attack.

### 6.6.2 Password Strength

**Password strength [38] is depicted as the measure of a password's ability to resist password cracking attacks**. In its typical shape, it estimates how many attempts an assailant who does not have direct access to the password would need, overall, to get it accurately. The strength of a password is considered a function of length, complexity, and unpredictability. Utilizing strong passwords diminishes large danger of a security break, yet strong passwords do not eliminate the requirement for other viable security controls.The strength of a password is defined by;

- **Length**: the number of characters the password incorporates.
- **Complexity**: does it utilize a blend of letters, numbers, and symbols?
- **Unpredictability**: is it something that can be speculated effectively by an assailant?

Let's now look at a practical example. We will use three passwords namely

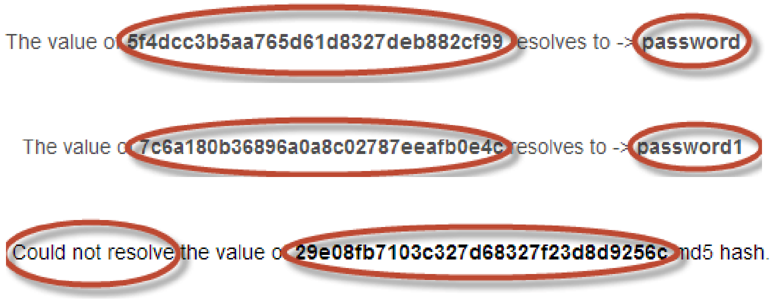
1. *password*
2. *password1*
3. *#password1$*

The higher the strength number, better the password. Let's suggest that we have to save our above passwords using md5 encryption. We will use an online md5convertor to convert our passwords into md5 hashes. The table 5 below shows the password hashes.

**Table 5. the passwords' hashes**

| Password | MD5 Hash |
|---|---|
| password | 5f4dcc3b5aa765d61d8327deb882cf99 |
| password1 | 7c6a180b36896a0a8c02787eeafb0e4c |
| #password1$ | 29e08fb7103c327d68327f23d8d9256c |

We will now use http://www.md5this.com/ to crack the above hashes. The images below illustrate the password cracking results for the above passwords. Fig.7 illustrates passwords' hashes.


The value of 5f4dcc3b5aa765d61d8327deb882cf99 resolves to -> password

The value of 7c6a180b36896a0a8c02787eeafb0e4c resolves to -> password1

Could not resolve the value of 29e08fb7103c327d68327f23d8d9256c md5 hash.

**Fig.7 passwords' hashes**

From the above outcomes, we figured out how to break the first and second passwords. We didn't figure out how to break the third password which is longer, perplexing and unexpected.

### 6.6.3 Types of Attacks against hash functions used in Passwords Encryption and their resistance

#### 1. Collision attack

In cryptography, a **collision attack** [39] on a cryptographic hash tries to find two inputs generating the same hash value, i.e. a hash collision. This is different than a preimage attack where a specific target hash value is determined. There are briefly two types of collision attacks:

#### Collision attack

Find two different messages *m1* and *m2* such that *hash(m1) = hash(m2)*.

#### Chosen-prefix collision attack

Given two different prefixes *p1, p2* find two appendages *m1* and *m2* such that *hash(p1 ∥ m1) = hash(p2 ∥ m2)* (where ∥ is the concatenation operation).

**2. Preimage Attack**

In cryptography, a **preimage attack** [40] on cryptographic hash functions attempts to discover a message, that has a specific hash esteem (value). A cryptographic hash function should resist attacks on its preimage. With regards to attack, there exist two kinds of preimage resistance:

- *preimage resistance*: for basically all pre-determined outputs, it is considered as computationally infeasible to discover any input that hashes to that output, i.e., given $y$, it is hard to discover an $x$ to such an extent that $h(x) = y$.
- *second-preimage resistance*: it is considered computationally infeasible to discover any second input which has an indistinguishable output as that of a predetermined input, i.e., given $x$, it is hard to discover a second preimage $x' \neq x$ with the end goal that $h(x) = h(x')$.

**3. Collision resistance**

**Collision resistance** [41] is considered as a property of cryptographic hash functions: a hash function $H$ is collision resistant, if it is hard to find two inputs that hash to the same output; that is, two inputs $a$ and $b$ such that $H(a) = H(b)$, and $a \neq b$.

Collision resistance is an even harder property, which we still need for most usages of hash functions: It is hard to find a pair of messages x1≠x2x1≠x2 with H(x1)=H(x2)H(x1)=H(x2).

Each hash function with bigger number of inputs than outputs will essentially cause collisions. Considering a hash function for example SHA-256, that produces 256 bits of output from an (discretionarily) arbitrarily extensive input. It must produce one of $2^{256}$ outputs for every member of a much bigger set of inputs. Collision resistance does not imply that no collisions exist; essentially that they are elusive.

**4. Preimage resistance**

Preimage resistance [42] is considered as the most fundamental characteristic of a hash function, which can be thought. It implies:

For a given h in the output space of the hash function, it is difficult to discover any message x with H(x)=h. Hard means takes additional time/costs than any (speculative aggressor) hypothetical attacker can contribute. In practice, uniqueness is not characterized by the (dynamic) abstract theoretical non-presence of collisions, but by the (pragmatic) practical non-presence of collisions. So as to discover a collision in SHA-256, you would (presumably) probably need to run the algorithm somewhere in the range of $2^{128}$ times. It is far-fetched that this will happen at any point in the near future, regardless of whether you check the total number of times SHA-256 will ever be processed by anybody in the whole universe combined. SHA-256 is thought to be practically difficult to "crack", that is, to recover the original plaintext from the hash.

## 7 Experimental Results

In the following subsections, the main parts of the security proposed technique are described. Then, each parameter of the five parameters described in the previous section 6 resulting from simulation is explained below.

### 7.1 Proposed security technique main parts

**Part 1:** android enabled device sending user's device id and password to controller device

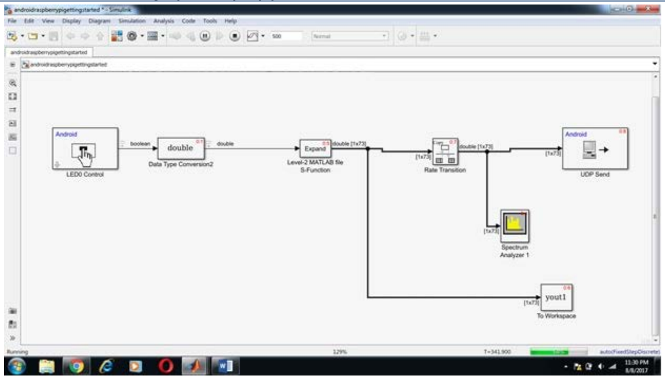


Fig.8 Process of sending user data from android device to controller device during simulation

In part 1, the user's sends the device id and password and key used for encryption to the controller device during simulation. Part 1 is represented in Fig.8. Main blocks of part 1 are: Android control button (from android toolbox and used for turning on/off remote devices through the controller device), double block (used to convert data to double), Level 2 MATLAB s-function (used for performing encryption), the To workspace block (carries yout1 3D array; which results after encryption data as mentioned in the proposed technique and represents the encrypted signal,and its role is to output the encrypted signal in the workspace), the rate transition block (to transform data in way that it can appear in the spectrum analyzer), the spectrum analyzer 1 block (to show signal), the android UDP send block (to send encrypted signal wirelessly to the controller device).

**Part 2:** controller device sending the controller device id and encryption keys to home devices

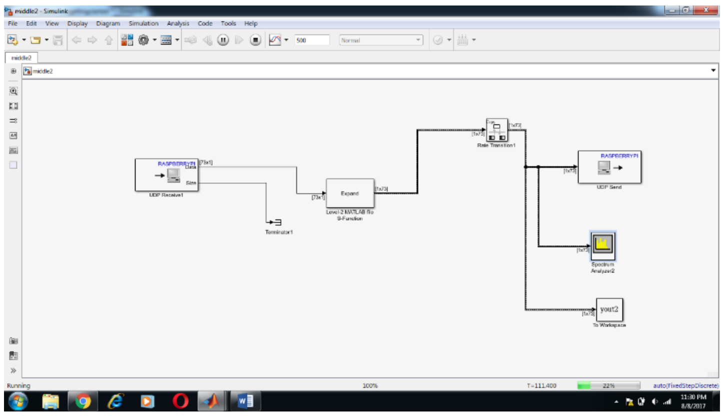


Fig.9. Process of controller device sending data to home devices during simulation

Part 2 represents the process of controller device sending data to home devices wirelessly during simulation. Part 2 is illustrated in Fig.9. Main blocks of part 2 are: the raspberry pi UDP receive block inside the controller device (from the raspberry pi toolbox (raspberry pi 3 model B); used to receive wirelessly the signal coming from the android device), the level-2 MATLAB function block (used for verifying the encrypted received signal and then sending encrypted device id and encryption keys to raspberry pi enabled home devices after verification), the rate transition block (used to enable encrypted signal that results from the level-2 MATLAB function to be displayed in the spectrum analyzer 2), the to workspace block (carries yout2 3D array; which results after encryption data and represents the second encrypted signal,and its role is to output the second encrypted signal in the workspace to be displayed), the raspberry pi UDP send block (from the raspberry pi toolbox (raspberry pi 3 model B), used to send data wirelessly to the raspberry pi enabled home device/s).

**Part 3**: the home devices verify the coming data and turn on/off connected device/s

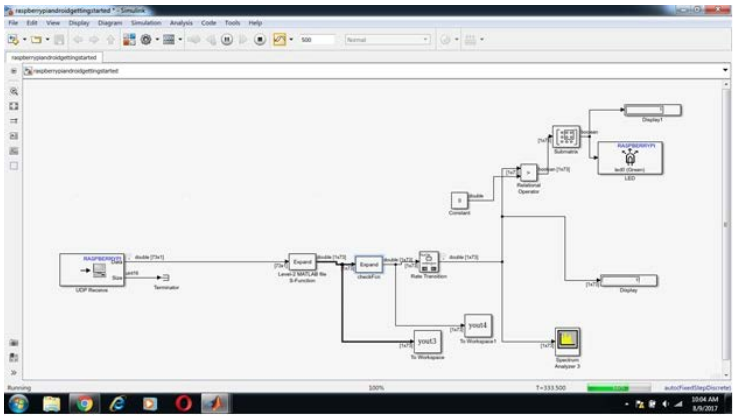


**Fig.10 Process of homes devices verifying data and turning device LED ON/OFF during simulation**

Part 3 represents the process of home devices verifying data and turning device LED ON/OFF during simulation. Fig.10. illustrates part 3. The main blocks of part 3 are: raspberry pi UDP receive block (from the raspberry pi toolbox (raspberry pi 3 model B); and shows the raspberry pi home  enabled device receiving data wirelessly from the raspberry pi controller device), the Level-2 MATLAB s-function block (verify the received encrypted controller device id, after verification, send signals to next checkFCn block), the checkFCn MATLAB block (used to change the home device status if received data are true), the to workspace block yout3 (outputs 3D array resulting from the Level-2 s-function to the workspace), the to workspace block yout4 (outputs 3D array resulting from thr checkFcn block to the workspace), the rate transition block (allows received signal to be displayed in the spectrum analyzer 3), the spectrum analyzer 3 (displays received signals), the display block (display the received signal as a numerical array), the relational operator block (checks if the signal is greater than 0, its outputs 1 and then turns on the LED conncted to the device; else if the the signal is smaller than 0; it outputs 0 then turn off the LED device), the submatrix block (changes the size of the array to fit the LED input), the raspberry pi LED block (from the raspberry pi toolbox (raspberry pi 3 model B), represents the LED of the raspberry pi connected enabled home device).

Let’s look at signals as they appeared in the spectrum analyzer. Signals appear as a line means devices receiving 0’s or no data sent or received. The below two figures show the signal carrying the encrypted data as well as the keys when sent and received. Fig.11 shows  spectrum analyzer 1 & 2 when receiving signal carrying encrypted data in parts 1 & 2. Fig.12 illustrates spectrum analyzer 3 upon receiption of signal carrying encrypted data in part 3.

The received signal in part 3 named yout 3 as appearing in the MATLAB interface in numerical representation is shown in Fig.13 below. The sent signals gives 1’s after sending the encrypted data and keys during simulation.

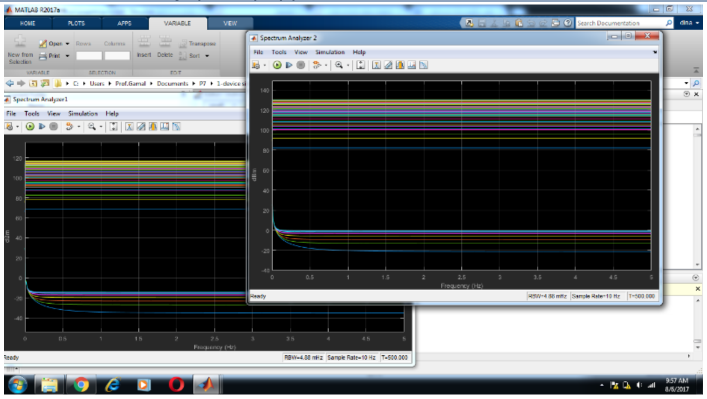


**Fig.11. Spectrum analyzer 1 & 2 showing encrypted signal receiption**

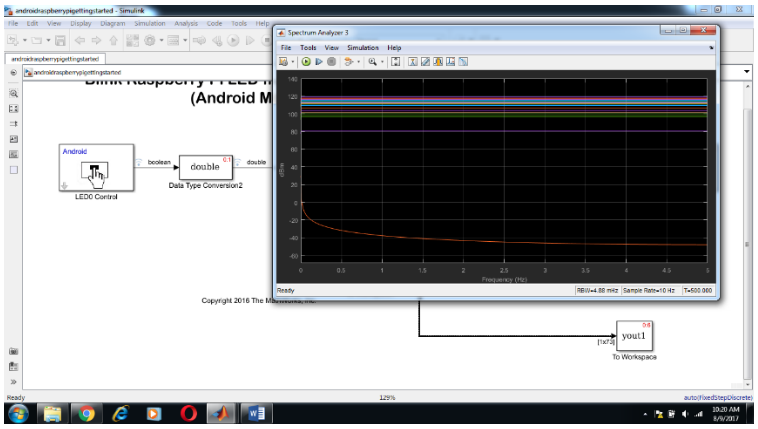
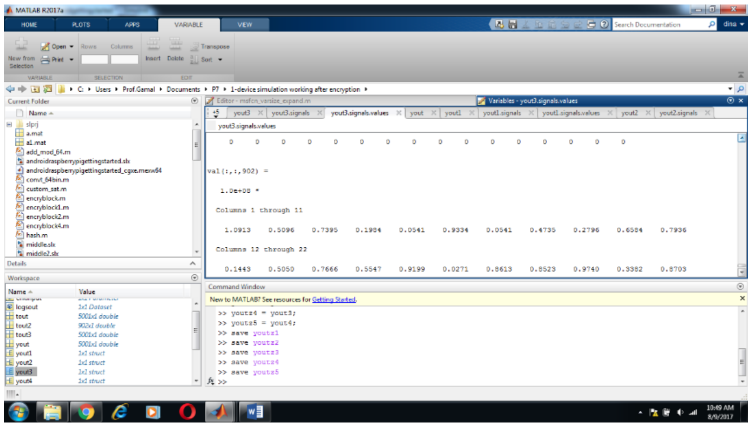


**Fig.12. Spectrum anlyzer 3 showing encrypted signal receiption**



```
val(:,:,902) =

   1.0e+08 *

  Columns 1 through 11

    1.0913    0.5096    0.7395    0.1984    0.0541    0.9334    0.0541    0.4735    0.2796    0.6584    0.7936

  Columns 12 through 22

    0.1443    0.5050    0.7666    0.5547    0.9199    0.0271    0.8613    0.8523    0.9740    0.3382    0.8703
```

```
>> youtz4 = yout3;
>> youtz5 = yout4;
>> save youtz1
>> save youtz2
>> save youtz3
>> save youtz4
>> save youtz5
>>
```

**Fig.13. Encrypted received signal in part 3 named yout3 in the MATLAB interface**

## 7.2 Response time

To know simulation time from the beginning of part 1, when the user presses the button to change connected home devices passing through the controller device to the end of part 3 when the home device LED is turned ON in case the received data carrying user device's id and user's password, and controller device id are all correct, a stop block is added in part 3; which represents the connected enabled raspberry pi home device with a LED. The role of the stop block is to stop simulation when a signal carrying data greater than 0's is entered, it means when the encrypted verified signal arrives to turn ON (in this case) the home device’s LED. There are two display blocks; one display to show the signal after transforming it to fit the LED size to 1’s, the other display to show the final encrypted signal received in numerical representation; which is connected to the stop block. After stopping the simulation, the response time is given, with analysis on the time partitioning across different tasks; in other words, how each task takes time during simulation obtained from report analyzer tool inside MATLAB. A stop block is added in part 3 to stop simulation when the received encrypted data are verified and the LED is turned ON to measure the response time. The response time here from the beginning to the end of the simulation is 24.24 s, but it is organized across many tasks, it means that each task takes an amount of time to be executed during simulation. So that, the compile and link task takes approximately more than 80% of the total response time, but some other tasks; such as display.outputs.major task takes a very small percentage of the response time not increasing than 1% of it. Also, Fig.14 shows the response time; as explained in section 6, organized across different tasks as obtained from the report analyzer tool in MATLAB. Table 6 illlustrates the tasks composing the total simulation time, and the percentage of each task from simulation time.

**Table 6. The percentage of each task from simulation time**

| Task | Time(%) |
|---|---:|
| total simulation time from start to end | 100 |
| simulation Phase | 77.3 |
| compile and link Phase | 10.2 |
| initialization phase | 8.6 |
| simulation.outputs.major | 7.9 |
| simulation.setupruntimeresources | 6.3 |
| spectrumanalyzer.setupruntimeresources | 6.1 |
| M-S-function.outputs.major | 4 |
| termination phase | 3.9 |
| simulation.cleanupruntimeresources | 3 |
| M-S-function2.outputs.major | 2.8 |
| simulation.update | 0.6 |
| spectrumanalyzer.update | 0.6 |
| toworkspace.outputs.major | 0.3 |
| spectrumanalyzer.cleanupruntimeresources | 0.2 |
| M-S-function3.outputs.major | 0.2 |
| display.setupruntimeresources | 0.1 |
| display.cleanupruntimeresources | 0.1 |
| stop.outputs.major | 0.1 |
| S-function4.outputs.major | 0.1 |
| display.outputs.major | 0.1 |
| s-function5.outputs.major | 0.1 |
| toasyncqueueblock.setupruntimeresources | 0.1 |

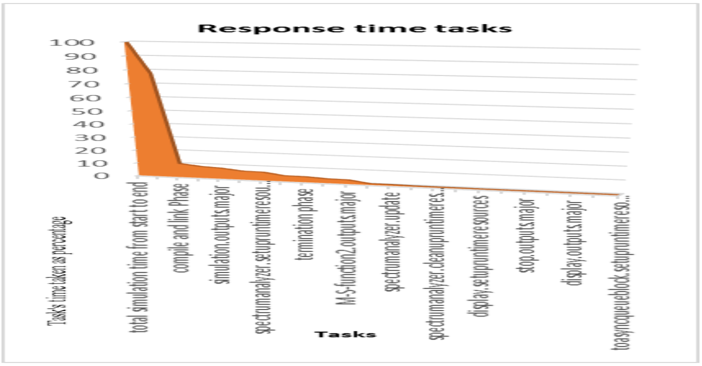


**Fig.14 shows the response time organized across different tasks during simulation**

## 7.3 Memory consumption

During the simulation, when the simulation time increases from 100 s to 1000 s, memory consumed increases gradually from below 5 MB to near 35 MB as shown in Fig.15. The experimental results proved that increasing the simulation time, increases the amount of memory consumed in Megabytes. Fig.15 shows memory consumption in Megabytes when the simulation time increases from 100, 200, 300, 400, 500, 600, 700, 800, 900 to 1000 seconds. But the amount of memory consumed expressed in Megabytes in general is good, and not very large. Table 7 shows memory consumption in (MB) versus simulation time.

**Table 7. Memory consumption in (MB) versus simulation time**

| time | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| Memory consumption during simulation (MB) | 3.4438 | 6.2819 | 10.3102 | 13.7435 | 17.1767 | 20.7017 | 24.1501 | 27.5986 | 31.0471 | 34.4956 |

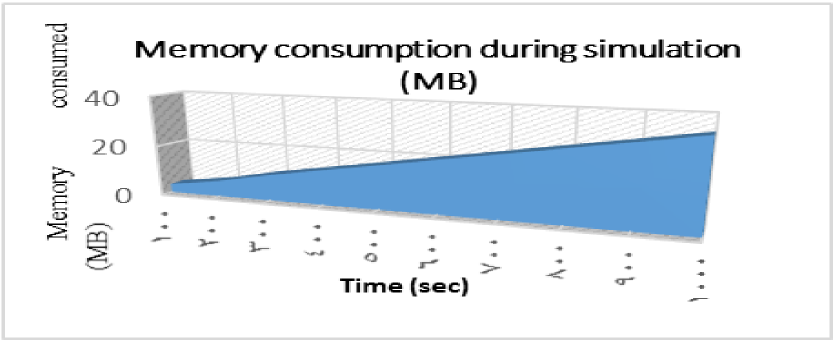


**Fig.15 Memory consumption (in Megabytes) versus simulation time (in seconds)**

## 7.4 Power of signals consumed

The experimental results showed that increasing the simulation time gradually from 100, 200, 300, 400, 500, 600, 700, 800, 900 to 1000 s, increases slightly by small portions the amount of power consumed in decibels of signals sent; either at the android device, the raspberry pi enabled controller device or the the raspberry pi enabled home device, and in some cases, the consumed power can decrease a little and increase again; for example at a simulation time of 600 seconds in the android device. In general, the power of signals consumed during simulation proved to be good and reasonable at the android device, the controller device and the home device, and ranges from 152 dB to 156 dB. Fig.16, Fig.17, Fig.18 and Fig.19 illustrates the power of signals consumed expressed in Megabytes in the android device, controller device, the home device and all the mentioned three graphs are added in the last graph respectively. Table 8 shows power consumed in (dB) versus simulation time.

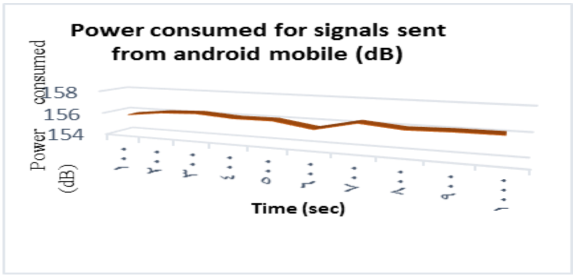


**Fig.16 Power consumed (in Megabytes) versus the simulation time (in seconds) at the android device**

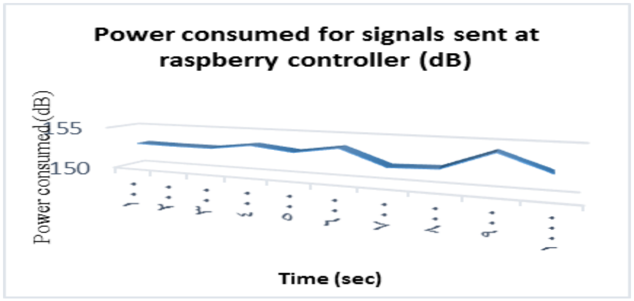


**Fig.17 Power consumed (in Megabytes) versus the simulation time (in seconds) at the raspberry pi** controller device

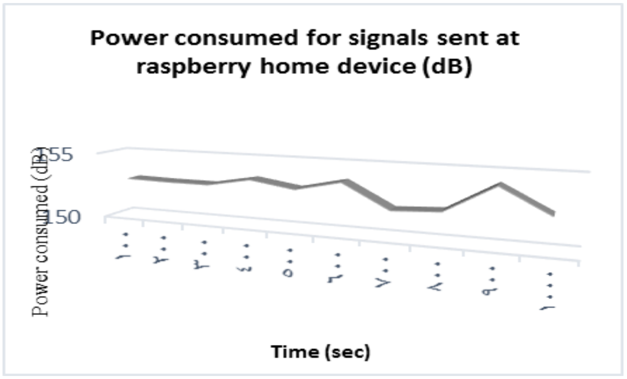


**Fig.18 Power consumed (in Megabytes) versus the simulation time (in seconds) at raspberry pi home device**

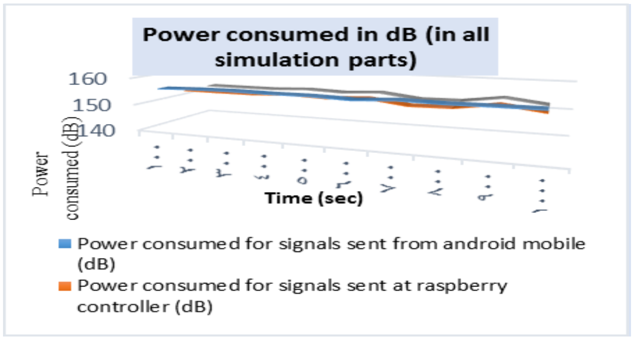


**Fig.19 Power consumed (in Megabytes) versus the simulation time (in seconds) at the android device, the controller device and the home device**

**Table 8. Power consumed in (dB) versus simulation time**

| Time | 100 | 200 | 300 | 400 | 500 | 600 | 700 | 800 | 900 | 1000 |
|---|---|---|---|---|---|---|---|---|---|---|
| Power consumed for signals sent from android mobile (dB) | 155.768 | 156.1378 | 156.2714 | 156.0421 | 156.0652 | 155.5702 | 156.3233 | 156.0253 | 156.1118 | 156.1221 |
| Power consumed for signals sent at raspberry controller (dB) | 152.9724 | 152.9094 | 152.9071 | 153.5063 | 153.0104 | 153.7477 | 152.0055 | 152.1923 | 154.1718 | 152.4356 |
| Power consumed for signals sent at raspberry home device (dB) | 152.9724 | 152.9094 | 152.9071 | 153.5063 | 153.0104 | 153.7477 | 152.0055 | 152.1923 | 154.1718 | 152.4356 |

## 7.5 Bit error rate during simulation

To measure bit error rate, it is required to make some adjustments on the three main parts of the proposed security technique blocks. First, a packet Output block is added at part 1 before the android UDP send block ( from the android toolbox), the block has three outputs; number of ticks, data_ready and data_error; which represents the third output of the packet output block and is used to measure errors that occurred during sending data wirelessly from the android device to the raspberry pi enabled controller device. Also in part 1, a packet input block is added; which has four outputs, captured data at the controller device, the data_ready, the data_error and the number of ticks. Also, the data_error in packet input block is used to measure errors in data received wirelessly at the IP address of the controller device. The spectrum analyzer shows signal at the android device, and the scope shows signal data at the controller device. In part 2, a packet input block is added, with four outputs; which are captured data at the controller device, data_ready, data_error and number of ticks. The data error output of the packet input block represents errors in data sent wirelessly from the controller device. Also in part 2, a packet output block is added, which has three outputs, number of ticks, data_ready and data_error. The data error represents errors in data received wirelessly at the IP address of the home device. The error of data sent from the android device and errors of data received at the controller device are measured. The error of data sent from the controller and errors of data received at the home device are measured. Fig.20, Fig.21, Fig.22, Fig.23, Fig.24 illustrate the percentage of data errors that occurred versus a simulation time of 100, 200, 300, 400, 500, 600, 700, 800, 900 and 1000 s, at the android device (sender), the controller device (receiver), the controller device (sender), the home device (receiver), and all the preceding graphs grouped in one graph respectively. The experimental results showed that the errors that occurred during transmission wirelessly is in general good, since it ranges from 0.5% to 20%. Table 9 shows data error percentage versus simulation time.

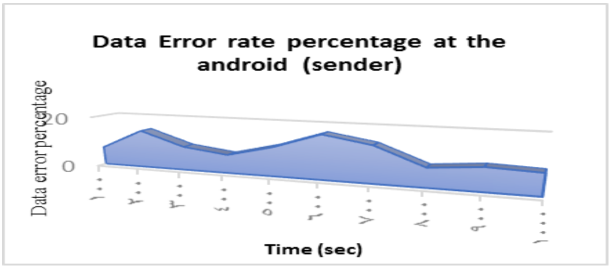


Fig.20 Percentage of data errors that occurred versus the simulation time at the android device as a sender

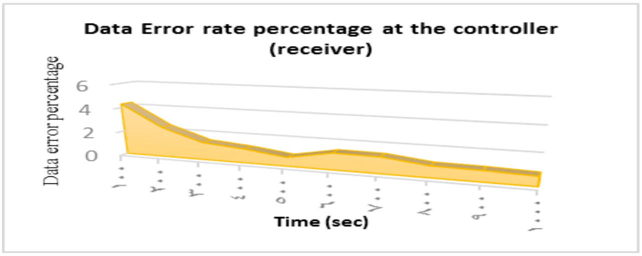


Fig.21 Percentage of data errors that occurred versus the simulation time at the controller device as a receiver

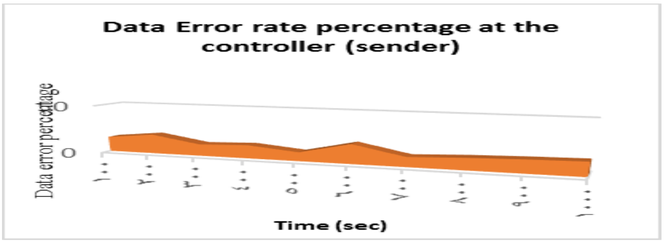


Fig.22 Percentage of data errors that occurred versus the simulation time at the controller device as a sender

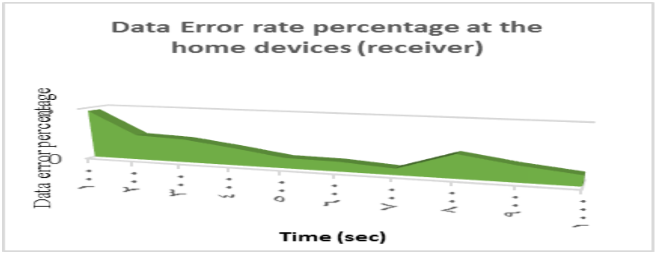


Fig.23 Percentage of data errors that occurred versus the simulation time at the home device as a receiver

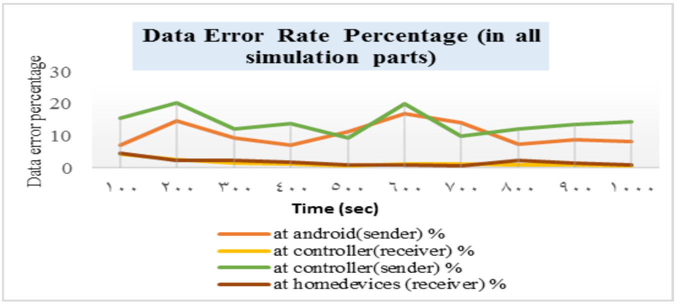


Fig.24 Percentage of data errors that occurred versus the simulation time at the android device (sender), the controller device (receiver), the controller device (sender), the home device (receiver)

**Table 9. Data error percentage versus simulation time**

| | data error percentage | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| **Time** | **100** | **200** | **300** | **400** | **500** | **600** | **700** | **800** | **900** | **1000** |
| **at android (sender) %** | 7.0929 | 14.7426 | 9.2636 | 7.1482 | 11.2777 | 16.9805 | 14.0409 | 7.2491 | 8.7657 | 8.0992 |
| **at controller (receiver) %** | 4.2957 | 2.4988 | 1.4329 | 1.1497 | 0.7199 | 1.3498 | 1.3141 | 0.9124 | 0.8666 | 0.7499 |
| **at controller (sender) %** | 15.3846 | 20.09 | 12.1293 | 13.7716 | 9.1782 | 20.0133 | 9.9843 | 12.0985 | 13.4207 | 14.2986 |
| **at home devices (receiver) %** | 4.6953 | 2.4488 | 2.2659 | 1.6746 | 0.9998 | 0.9332 | 0.5999 | 2.2122 | 1.4998 | 0.9799 |

## 7.6 Password strength checking

In the final parameter, it is required to test the strength of the device id for example, using mathematics once, and using the simulation inside the MATLAB R2017a. To find the possible combinations of finding a device id (for example). The basic formula used for finding a given combination is given by:

*C(n,k) = n!/(k!(n–k)!)*

Here, n is the total number of items and k is the number of members or items chosen from total number of given n. This can also be written as the binomial coefficient (n k) as below:

*(n(n–1)(n–2)…(n–k+2)(n–k+1))(k(k–1)(k–2)…2.1)(n(n–1)(n–2)…(n–k+2)(n–k+1))(k(k–1)(k–2)…2.1)*

So, in our case the total number of possible combinations is calculated as follows, we have 127 (n) characters at the computer keyboard, and the device id is composed of 10 (k) items (numbers and letters; capital or small), we have:

*C(127,10) = 127!/10!(127-10)! =*
*127.126.125.124.123.122.121.120.119.118.117!/1.2.3.4.5.6.7.8.9.10.117!*
*= 7.588684395810302e+20/3628800 = 209123798385425 possible combinations.*

After 209 trillion trials 209123798385425/5100290 = 20501167.42238432 hours of trials/24= 854215.3092660131 days of trials/365 = 2340.315915797296 years of trials. On an ordinary computer it means it is very difficult to regenerate the device_id in an ordinary computer using trials due to the huge number of possible combinations. Generating a code inside MATLAB R2017a to try to find the device id, and giving the number of trials or iterations during the simulation, gives us the below figures. Fig.25 shows the total number of trials to find the device id (in this case) versus the total number of hours of simulation. After all these trials, the device id was not found. Table 10 shows number of trials versus number of simulation hours.

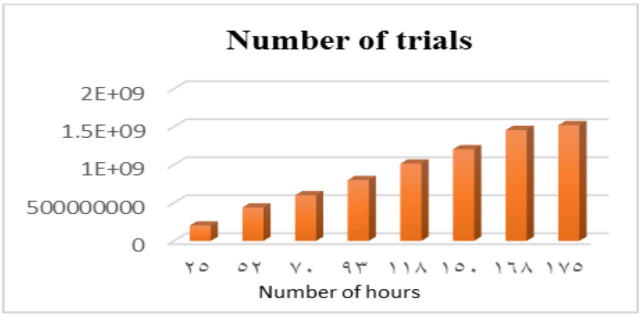


**Fig.25 Number of trials to find the device id versus the number of hours of simulation taken**

Table 10. Number of trials versus number of simulation hours

| Number of hours | 25 | 52 | 70 | 93 | 118 | 150 | 168 | 175 |
|---|---|---|---|---|---|---|---|---|
| Number of trials | 2.03E+08 | 4.39E+08 | 5.99E+08 | 7.99E+08 | 1.02E+09 | 1.20E+09 | 1.46E+09 | 1.52E+09 |

## 8 Conclusion and Future Work

From the results obtained from experimental simulations, it is concluded that the proposed security technique (SMI2C) provides good response time, reasonable amount of memory consumed, and power consumed during simulation, good bit error rate and strong technique for protecting passwords without any additional overheads on the proposed system. So, the proposed technique is useful in encryption as it protects user data during transmission between different devices and has many benefits.

In the Future work, a secure technique for internet could be developed taking into consideration reducing energy consumed during transmission; also, reducing memory consumed during signals transmissions could be studied. There are a lot of other parameters that can be considered as areas of research while designing a secure technique for online data transmissions in the future; such as speed of transmission, bit error rate and so on.

## REFERENCES

[1] Gartner's hype cycle special report for 2015, Gartner Inc., 2015. [Online]. Available: http://www.gartner.com/technology/research/hype-cycles/.

[2] Gubbi J., et al., *Internet of Things (IoT): A vision, architectural elements, and future directions*, Future Gener. Comput. Syst., 2013. 29 (7): p. 1645–1660.

[3] Miorandi D., et al., *Internet of things: Vision, applications and research challenges*, Ad Hoc Network, 2012. 10 (7):p. 1497–1516.

[4] Yasumoto K., Yamaguchi H., and Shigeno H., *Survey of real-time processing technologies of IoT data streams*, J. Inf. Process, 2016. 24 (2):p. 195–202.

[5] Husain S., et al., *Recent trends in standards related to the internet of things and machine-to-machine commun*., 2014, 4 (6).

[6] Djenouri D., Khelladi L., and Badache N., *A Survey of Security Issues in Mobile Ad-hoc Networks and Sensor Networks*, IEEE Communications Surveys, 2005. 7 (4):p. 2-28.

[7] Cho J.-H., Swami A., and Chen R., *A Survey on Trust Management for Mobile Ad-hoc Networks*, IEEE Communications Surveys & Tutorials, 2011. 13 (4):p. 562-583.

[8] Wang Y., Attebury G., and Ramamurthy B., *A Survey of Security Issues in Wire-less Sensor Networks*, IEEE Communications Surveys Tutorials, 2006. 8 ( 2 ):p: 2-23.

[9] Cha I., et al., *Trust in M2M Communication*, IEEE Vehicular Technology Magazine, 2009. 4 ( 3 ): p. 69-75.

[10] Mell P. and Grance T., *The nist definition of cloud computing*, National Institute of Standards and Technology, 2009. 53 ( 6 ) article 50.

[11] Zhang, Q., Cheng, L., and Boutaba, R., *Cloud computing: state-of-the-art and research challenges*. Journal of internet services and applications, 2010. 1 (1):p. 7-18.

[12] Zhou J., et al., *Cloud Architecture for Dynamic Service Composition*, International Journal of Grid and High Performance Computing, 2012. 4 (2):p. 17-31.

[13] Christophe, B., et al., *The web of things vision: Things as a service and interaction patterns*. Bell Labs Technical Journal, 2011. 16 (1):p. 55-61.

[14] Subashini, S., and Kavitha, V., *A survey on security issues in service delivery models of cloud computing*. Journal of Network and Computer Applications, 2011. 34 (1):p. 1-11.

[15] Botta A., et al., *Integration of Cloud Computing and Internet of Things: a Survey*, Journal of Future Generation Computer Systems, September 18, 2015.

[16] Gomes, M. M., Righi, R. d. R., and da Costa, C. A., *Future directions for providing better iot infrastructure*. In: *Proceedings of the 2014 ACM International Joint Conference on Pervasive and Ubiquitous Computing: Adjunct Publication (UbiComp '14 Adjunct.)*, 2014, p. 51-54.

[17] Alhakbani, N., ei al., *A framework of adaptive interaction support in cloud-based internet of things (iot) environment*. In: Internet and Distributed Computing Systems. Springer, 2014, p. 136-146.

[18] Fox, G. C., Kamburugamuve, S., and Hartman, R. D., *Architecture and measured characteristics of a cloud based internet of things*. In: Collaboration Technologies and Systems (CTS), 2012 International Conference on. IEEE, 2012, p. 6-12.

[19] Dash, S. K., Mohapatra, S., and Pattnaik, P. K*., A Survey on Application of Wireless Sensor Network Using Cloud Computing*. International Journal of Computer science & Engineering Technologies, 2010. 1 (4):p. 50-55.

[20] Atzoria L. and Giacomo Morabito A.I., *The Internet of Things: A Survey*, Computer Networks, 2010. 54 (15):p. 2787-2805.

[21] Gantz J., *The Embedded Internet: Methodology and Findings*, 2009. [Online]. Available: https://www.bryankorourke.com/blog/2010/3/11/the-embedded-internet-15-billion-devices-by-2015.html

[22] Evans D., *The Internet of Things: How the Next Evolution of the Internet Is Changing Everything*, 2011. [Online]. Available: http://www.cisco.com/web/about/ac79/docs/innov/IoT_IBSG_0411FINAL.pdf

[23] Hatton M., *The Global M2M Market in 2013*, Machina research whitepaper, 2013.

[24] Emmerson B., *M2M: The Internet of 50 Billion Devices*, Win-Win, 2010, pp. 19-22.

[25] M2M. [Online]. Available: SingTel M2M, http://info.singtel.com/large-enterprise/about-m2m.

[26] Watson D.S., et al., *Machine-to-Machine (M2M) Technology in Demand Responsive Commercial Buildings*, in Proceedings of the ACEEE Summer Study on Energy Efficiency in Buildings, 2004, pp.1-14.

[27] ETSI, TS 102 690 M2M Functional Architecture, 2011.

[28] RSA. [Online]. Available: https://en.wikipedia.org/wiki/RSA

[29] Digital signature algorithm. [Online]. Available: https://en.wikipedia.org/wiki/Digital_Signature_Algorithm

[30] Elliptic curve cryptography. [Online]. Available: https://en.wikipedia.org/wiki/Elliptic-curve_cryptography

[31] SHA-2. [Online]. Available: https://en.wikipedia.org/wiki/SHA-2

[32] MATLAB Simulink. [Online]. Available: https://www.mathworks.com/products/simulink

[33] Raspberry Pi. [Online]. Available: https://en.wikipedia.org/wiki/Raspberry_Pi

[34] Choil K.-H., et al., *Method of Calculating the Server Capacity for Cloud Computing for SaaS*, International Journal of Software Engineering and Its Applications, 2015. 9 (11):p. 117-126 .

[35] Signal strength. [Online]. Available:

https://en.wikipedia.org/wiki/Signal_strength_in_telecommunications

[36] Bit error rate. [Online]. Available: https://en.wikipedia.org/wiki/Bit_error_rate

[37] Password cracking. [Online]. Available: https://en.wikipedia.org/wiki/Password_cracking

[38] Password cracking of an application. [Online]. Available: ]https://www.guru99.com/how-to-crack-password-of-an-application.html

[39] Collision attack. [Online]. Available: https://en.wikipedia.org/wiki/Collision_attack

[40] Preimage attack. [Online]. Available:

[41] Collision resistance. [Online]. Available: https://en.wikipedia.org/wiki/Collision_resistance

[42] Preimage resistance and collision resistance. [Online].Available: https://crypto.stackexchange.com/questions/1173/what-are-preimage-resistance-and-collision-resistance-and-how-can-the-lack-ther