future internet

MDPI

Article

Securing Critical User Information over the Internet of Medical Things Platforms Using a Hybrid Cryptography Scheme

Oluwakemi Christiana Abikoye [1], Esau Taiwo Oladipupo [2], Agbotiname Lucky Imoize [3,4,*], Joseph Bamidele Awotunde [1], Cheng-Chi Lee [5,6,*] and Chun-Ta Li [7,*]

[1] Department of Computer Science, Faculty of Information and Communication Sciences, University of Ilorin, Ilorin 240003, Nigeria
[2] Department of Computer Science, The Federal Polytechnic Bida, Bida 912211, Nigeria
[3] Department of Electrical and Electronics Engineering, Faculty of Engineering, University of Lagos, Akoka, Lagos 100213, Nigeria
[4] Department of Electrical Engineering and Information Technology, Institute of Digital Communication, Ruhr University, 44801 Bochum, Germany
[5] Research and Development Center for Physical Education, Health, and Information Technology, Department of Library and Information Science, Fu Jen Catholic University, New Taipei City 24206, Taiwan
[6] Department of Computer Science and Information Engineering, Asia University, Taichung City 41354, Taiwan
[7] Bachelor's Program of Artificial Intelligence and Information Security, Fu Jen Catholic University, No. 510, Zhongzheng Road, New Taipei City 24206, Taiwan
* Correspondence: aimoize@unilag.edu.ng (A.L.I.); cclee@mail.fju.edu.tw (C.-C.L.); 157278@mail.fju.edu.tw (C.-T.L.)

Abstract: The application of the Internet of Medical Things (IoMT) in medical systems has brought much ease in discharging healthcare services by medical practitioners. However, the security and privacy preservation of critical user data remain the reason the technology has not yet been fully maximized. Undoubtedly, a secure IoMT model that preserves individual users' privacy will enhance the wide acceptability of IoMT technology. However, existing works that have attempted to solve these privacy and insecurity problems are not space-conservative, computationally intensive, and also vulnerable to security attacks. In this paper, an IoMT-based model that conserves the privacy of the data, is less computationally intensive, and is resistant to various cryptanalysis attacks is proposed. Specifically, an efficient privacy-preserving technique where an efficient searching algorithm through encrypted data was used and a hybrid cryptography algorithm that combines the modification of the Caesar cipher with the Elliptic Curve Diffie Hellman (ECDH) and Digital Signature Algorithm (DSA) were projected to achieve user data security and privacy preservation of the patient. Furthermore, the modified algorithm can secure messages during transmission, perform key exchanges between clients and healthcare centres, and guarantee user authentication by authorized healthcare centres. The proposed IoMT model, leveraging the hybrid cryptography algorithm, was analysed and compared against different security attacks. The analysis results revealed that the model is secure, preserves the privacy of critical user information, and shows robust resistance against different cryptanalysis attacks.

Keywords: Internet of Medical Things; healthcare data; elliptic curve cryptography; privacy preservation; Caesar cipher; security

Citation: Abikoye, O.C.; Oladipupo, E.T.; Imoize, A.L.; Awotunde, J.B.; Lee, C.-C.; Li, C.-T. Securing Critical User Information over the Internet of Medical Things Platforms Using a Hybrid Cryptography Scheme. *Future Internet* **2023**, *15*, 99. https://doi.org/10.3390/fi15030099

Academic Editor: Guan Gui

Received: 3 February 2023
Revised: 25 February 2023
Accepted: 27 February 2023
Published: 28 February 2023

1. Introduction

The Internet of Things (IoT) offers a seamless way to connect people, devices, applications, and platforms [1] for communication, participation, and collaboration purposes [2–4]. The IoT is a recent paradigm in the wireless communication field, which comprises smart

appliances with a digital entity that can universally connect to a network and the Internet [5]. Smart devices can adapt intellect and the capability to sense, understand, and respond to their environment, leveraging evolving technologies [6] and Global Positioning Systems (GPS) with positioning algorithms [7]. The GPS is useful for obtaining location and positioning information. Embedded sensors can also be integrated to collect data for desired analyses [8] as well as enable machine-to-machine communication [9]. The intelligent systems in IoT systems most often need integration stages such as calibration, filtering, amplification, or compensation of signals from different sensors. The cloud usually reinforces IoT systems to solve the problems of data access, storage, scalability, and computing through services that provide various cloud computing architectures. The IoT has been incorporated into industrial manufacturing, Intelligent Transportation Systems (ITS), agriculture, healthcare systems, smart grid, home automation, food supply-chain, and mission-critical applications [10,11]. The amalgamation of IoT and medical systems birthed the Internet of Medical Things (IoMT) [12]. The IoMT has made activities such as real-time drug prescription, patient monitoring, real-time diagnosis of patients, and a host of other services in the healthcare system, which otherwise require the physical presence of healthcare workers and patients, possible remotely [12]. This amalgamation of the IoT and medical systems has both immediate and future benefits. The risk of getting infected during a pandemic such as COVID-19 can be greatly reduced since activities that involve people having contact with one another, such as the learning and evaluation of students, can be done using online platforms [13], and online business meetings which are possible through the Internet can replace physical ones [14]. The effective control of the spreading of disease can be achieved with the aid of IoT sensors, optical cameras, temperature detectors, and other IoT-enabled devices that can be used to detect and report the location of infected persons. Thus, the spread of disease can be controlled at the primary stage [15]. Other benefits that come with this amalgamation include connected hospitals (where data from different hospitals, such as treatment procedures, death rates, and the number of seats available for a new patient, can be readily accessible [16,17] via the IoMT platform), telehealth, and remote monitoring of patients [18,19].

All these great advantages and the promising future nature of the IoMT are very admirable. However, connecting an innumerable number of remotely controlled smart devices through the Internet increases users' security and privacy risk [9]. Applying protected communication in the IoMT and integrating security machinery into its devices involve many challenges [20] and have been the main obstructions to its progress. The characteristics of the security requirements of IoMT are capable of addressing almost all its security issues, such as confidentiality, among others [21]. Despite various solutions in the literature that address the security issues of IoMT, maintaining the confidentiality of patients' information in the IoMT still requires much attention [22], as people are still reluctant to provide critical user data or authorization to a server where such data could be accessed. Moreover, hackers can use a botnet [23] to hijack patients' information via the administration of IoT-based gadgets [24–27]. Therefore, the IoMT model that secures both IoMT gadgets and medical information is highly needed.

Just as the security of data in the cloud is important, the preservation of the privacy of the data such as medical information of people is also very important. However, the preservation of data privacy on the cloud has various challenges. Some of these challenges include:

i. Cloud service providers may not have privacy protection that is strong enough to protect the data that are put in their care [28];
ii. It is possible that cloud service providers change their technology without informing their customers; in such a case, issues related to performance and latency may arise [29,30];
iii. If cryptography algorithms alongside authentication techniques are used to preserve the privacy of data on the cloud, searching for information from the encrypted data becomes an issue, as encryption conceals the relationships among the data [31];

iv. If the secrete key of encryption is released to cloud service provider, the security of the data being protected may be breached [32–36];
v. Apart from breaching the security of the data, performing decryption of the data before searching for particular information in the data stored on the cloud is not efficient, especially when dealing with a huge amount of encrypted data or resource starve devices.

Researchers have identified these problems and have come up with different approach to address issues around the privacy preservation of cloud data. However, deficiencies have been found in many of these existing techniques. Searchable encryption (SE) appears a suitable approach to solve the data privacy problem in the cloud setting. An additional challenge is raised by SE in the multiuser setting, whereby each user may have access to a set of encrypted data segments stored by a number of different users. Multiuser searchable encryption schemes allow a user to search through several data segments based on some search rights granted by the owners of those segments. Privacy needs in this setting are many, and not only the confidentiality of the data segments but also the privacy of the queries should be ensured against impostors and potentially malevolent CSP. However, existing searchable encryption techniques such as Fully Homomorphic Encryption (FHE) implementations (e.g., [37–40]) require computationally intensive operations which make the schemes impractical. Practical FHE is still far from being realized, and improving the performance of FHE is still a very active research area.

In this research work, a secure IoMT model with an efficient cloud data privacy-preserving technique is projected for securing and preserving the privacy of sensitive medical information on the cloud. The scheme employed the use of a hybrid-based encryption/decryption scheme named Hybrid Modified Caesar Cryptosystem (HMCC). In HMCC, the existing stream Caesar cipher is modified into block cipher for improved data security, Elliptic Curve Diffie–Hellman (ECDH) is employed for secret key sharing, and an Elliptic Curve Digital Signature Algorithm (ECDSA) is applied to achieve digital signing and signature verification. The scheme proposes the application of B+ file organization for easy file storage and retrieval on the cloud.

This research has contributed to the body of knowledge as outlined below:

i. The existing Caesar cipher was modified to improve its security by extending the character set of the existing Caesar cipher from 26 to 256 (0–255), reshuffling the extended character set using an encryption key, and determining the ciphertext characters based on the location of each plaintext character in the plaintext and the location of the character on the character set using modulus 256 addition. The modulus 256 subtraction was used for recovering the plaintext characters from the ciphertext based on the position of each character on the ciphertext and on the character set. This was conducted to replace the mere shifting of characters by a constant value in the classical Caesar cipher;
ii. An efficient algorithm for searching through the encrypted files based on B+ file organization was proposed, and;
iii. The two innovations were combined to form a secure IoMT model which has the ability to secure and preserve the privacy of the patient medical records on the cloud.

The remaining parts of this paper are arranged as follows: Section 2 presents the related work. Section 3 describes the proposed IoMT model and the analysis of the model. Section 4 captures the results and useful discussions. Section 5 discusses limitations and future works. Finally, Section 6 provides the conclusion to the study.

2. Related Work

The Internet of Things has facilitated the proliferation of the Internet of Healthcare Things (IoHT) and the IoMT. With the IoHT and IoMT, the problems of healthcare systems can be reduced and monitored for almost all kinds of diseases. The authors in [41] combined the concept of the IoT and cloud computing with the healthcare system. IoT applications

are projected to address many challenges, and a cloud-based model was employed to offer services to recognize a patient from the home. The work in [42] discussed the design of a new method for the healthcare system in response to academic, industrial, and societal needs of the IoT. The authors discussed a system for monitoring heart rate using a smart health band, whose information is sent to the patient's friends and family.

Various applications, technologies and frameworks and the industrial importance of IoT were discussed in [43]. The paper discussed the security problems associated with the privacy of the IoT and model the necessity of avoiding classification attacks. An intelligent IoT model for enhanced medical care systems and e-health with reduced IoT attacks was proposed. A highlight of the significance of the IoT in the medical care system was also described.

A body sensor network capable of proficiently and continually communicating data to a public IoT-based system was proposed in [44]. The author explained the role of communication systems in the improvement in IoT practices in healthcare systems. Some essential security restrictions that could improve the security of the planned procedure were also suggested alongside the upcoming system on movable objects.

The increased exchange of audiovisual data through various cloud services necessitates an effective security approach. It is anticipated to have improved security resiliency, minimal quality compromise, and low computing complexity. Unfortunately, cryptography and steganography, two of the most important traditional security techniques, have a large computational cost, therefore limiting their ability to communicate over clouds, where each data input was previously very large, and a huge volume of multimedia data are transmitted over the network.

To address the aforementioned issues and make a potential solution more feasible, the authors in [45] used the Feistel structure and substitution permutation cryptography, which uses five rounds of substitution permutation to increase confusion and diffusion. The study only used a 64-bit block cipher with an identical key size to maintain superior security with minimal processing. The findings showed that even while keeping low entropy, strong correlation, and adequate computing time for multimedia data encryption, the model delivered higher attack resilience. The study, however, failed to address the issue of privacy of owners' data over the network, which was not considered.

The authors of [46] used an inventive cryptographic model with optimization techniques to study the security of medical images in the IoT. The majority of the time, patient data are kept on a cloud server in the hospital, therefore security is crucial. Therefore, a different framework was needed for the efficient storage and secure transmission of medical images combined with patient data. The most advantageous key was selected utilizing hybrid swarm optimization to increase the security level of the encryption and decryption process, specifically elliptic curve cryptography's grasshopper and particle swarm optimization techniques. According to this technique, the IoT framework secures medical images. The outcomes of this execution were contrasted and compared, while a variety of encryption algorithms and their optimization techniques are known to have the highest peak signal-to-noise ratio values, 59.45 dB, and a 1 structural similarity index, respectively. However, the privacy of the patient information was not taken into consideration.

An efficient and scalable Advanced Encryption Standard (AES) cryptosystem was presented by the authors in [47] to ensure enhanced defence against known assaults and the security of the medical record, successfully before transmission, especially medical imaging. The main contribution of the study was creating a scalable, safe hardware–software co-design system; a flexible and adaptable medical imaging processing system that incorporates authentication-based methods of the AES cryptosystem on ZedBoard with the least amount of overhead, including PUF and TRNG3. This system addressed security issues, while the issue of privacy was not considered.

Although various searchable SE techniques address the issue of privacy preservation of data in a cloud setting, various implementations of these techniques have been found to have one issue or the other. SEs based on symmetric key encryption techniques such

as [48] were found leak important information about the documents when analysed by statistical techniques. The approach was also found to be useful only for words of the same length. The SE technique employed by [49,50] are not efficient because they only support exact match queries. The drawback of the existing searchable SE techniques which support only exact keyword searches is that the system efficiency reduces as the number of distinct keywords in the document increases.

In order to solve the problem of searchable SE, researchers introduced Fuzzy Searchable Encryption-based (FSE) systems. FSE returns both exact matching or closest possible matching files based on keyword similarity semantics, as fuzzy keyword search can tolerate minor typos and formatting inconsistencies [51]. Adjedj [52] presented a technique solving the issue of preserving privacy in a biometric identification system based on a fuzzy search scheme. However, the technique is unsuitable for many applications when data are regularly updated or streaming. Other FSE-based techniques such as [53–55] are based on symmetric key encryption techniques which are vulnerable to Man-in-the-Middle Attack (MIMA).

As a measure to solve the problem of MIMA faced by SE and FSE that are based on symmetric encryption techniques, a searchable encryption technique that is based on public key encryption (PkSE) was introduced. PKSE addresses cases such as when the outsourced data (e.g., medical data, stock quotes, emails, etc.) are public and uploaded by different owners and the user is not aware of it; at the same time, the user wishes to retrieve certain files without revealing to the server which file they want. The first PKSE developed by [56] failed to protect keyword privacy in the public settings. The PKSE proposed by [57] attached a tag, which can be computed by the client to form a particular query y F(pk, x1), and by the server from a ciphertext that encrypts it G(pk, c) with a plaintext. Although the scheme was adjudged to be secure, it was left without solution to the problem of finding a standard model scheme approach, and the scheme only provided privacy to text drawn from a space of large min-entropy.

A modified lightweight algorithm based on Somewhat Homomorphic Encryption-Ring Learning with Error was proposed by [58]. The new technique was presented as an approach to securely encrypt IoT sensor signal value based on the frequency of transmitting the signal to the edge environment. The proposed work provides a secure key that the patient can only authorize to decrypt the original raw data from the sensor attached to the patient.

A summary of the related works is given in Table 1. From the review works, it is clear that the existing IoMT models are suffering from deficiency ranging from vulnerability to security attacks to a lack of privacy preservation of data and a high demand of memory and computation complexity. Evidently, an IoMT model that is less computationally and memory-intensive, which is capable of handling security and privacy issues when information is kept on the cloud, is a desirable project. This work presents an IoMT model that is capable of all of the afore-mentioned.

Table 1. Summary of the related works.

Reference	Title	Method	Strength	Weakness
[44]	Secure IoT-based healthcare system with body sensor networks	Proposed body sensor network capable of proficiently and continually communicating data to a public IoT-based system	A secure IoT system was developed	Privacy issue was not addressed

Table 1. *Cont.*

Reference	Title	Method	Strength	Weakness
[45]	Lightweight Feistel structure-based hybrid crypto model for multimedia data security over uncertain cloud environment	Feistel structure and substitution permutation cryptography	Higher attack resilience	The cipher has low entropy In addition, privacy issue was not addressed
[46]	Hybrid optimization with cryptography encryption for medical image security in the Internet of Things	Elliptic curve cryptography's grasshopper and particle swarm optimization techniques	High peak signal to noise ratio and structural similarity of 1	Only security of the data was addressed; the issue of privacy was not addressed
[47]	Obfuscated AES cryptosystem for secure medical imaging systems in IoMT edge devices	Adaptable medical imaging processing system that incorporates authentication-based methods of the AES cryptosystem on ZedBoard	A secured scalable hardware/software cryptosystem was built	The issue of privacy was not addressed
[48–50]	Practical techniques for searches on encrypted dataSecure indices for efficient searching on encrypted compressed dataPrivacy-preserving keyword searches on remote encrypted data	Symmetric cryptography algorithm and searchable encryption techniques	Both security and privacy issues were addressed	The system's efficiency reduces as the number of distinct keywords in the document increases
[51–55]	Fuzzy keyword search over encrypted data in cloud computingBiometric identification over encrypted data made feasible	Fuzzy Searchable Encryption-based (FSE) systems	Searchable encryption technique that is tolerant to some typo errors	Vulnerable to MIMA They are not suitable for systems that need constant updating
[56,57]	Public key encryption with keyword searchDeterministic and efficiently searchable encryption	Searchable encryption technique that is based on public key encryption (PkSE)	Addresses the problem of MIMA faced by SE and FSE	Cannot protect keyword privacy in public, and the scheme only provides privacy to text drawn from a space of large min-entropy
[59]	Towards Privacy-Preserving Medical Homomorphic Encryption	Fully Homomorphic Encryption	Addresses both security and privacy issues	The scheme is both computationally and memory-intensive

3. The Proposed IoMT Model

The aim of the IoMT is the ubiquitous deployment of home-based healthcare. Various healthcare centres and research institutes should also be able to communicate with one another and access permitted information from the cloud. With all this in place, the privacy of the patient information and query access of the information in an IoMT should be preserved. The IoMT model presented in this paper assumes that a patient can receive medical treatment in any healthcare centre that is connected to the cloud. In this scenario, the patient ID and other information about the identity of the patient have to be provided by the patient in order to provide healthcare access to medical records of the patient. Authorized doctors from the healthcare centre where the patient registered can access the medical record of the patient upon providing identification and the right key. Research

institutions accessing the healthcare cloud will only be able to access information based on the agreement during the registration with the healthcare centre.

The proposed IoMT model has three stages: data acquisition, data storage, and application stages. Three things are involved in the data acquisition stage: acquisition of the data by the sensor, encryption of the acquired data, and computation of the storage key value for proper storage of the encrypted data on the cloud. The second stage in the proposed IoMT model is the storage of the encrypted data on the cloud. The cloud here uses B+ file organization, where the data are positioned on the cloud based on the key attached to the data. The third stage in the proposed model is the application stage, where authorized doctors, patients who visit another healthcare centre, and registered research institutions are given permission to access information on the cloud. The proposed IoMT model is predicated on the following premises:

1. Each healthcare centre has a cloud where information about patients is stored, and various clouds of different healthcare centres are networked together;
2. Patients are embedded with different wireless sensors capable of communicating data to a central sensor connected to the healthcare cloud where the patient has registered;
3. The central sensor is loaded, among other things, with encryption and decryption applications;
4. Information from patients is stored in a central database through the help of a cloud network;
5. A timestamp named "T" that is included in every message sent and received ensures the accuracy of the data;
6. No two clients can communicate privately on the IoMT platform. The communication is between the HC and the clients;
7. The architectural design of the proposed IoMT model is depicted in Figure 1.

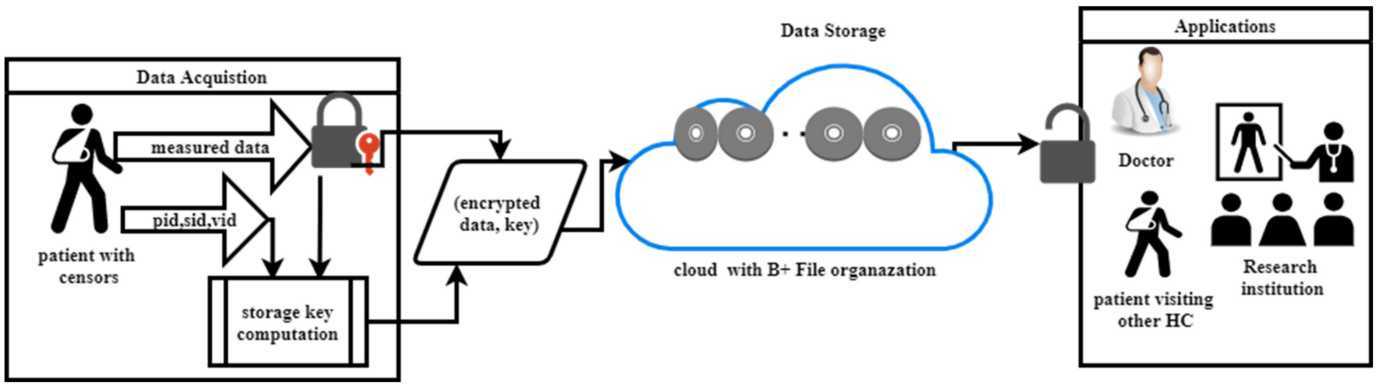


Figure 1. Proposed Secure IoMT Model.

The proposed IoMT model in this paper is discussed under the following subsections: (1) generation of initialization parameters, (2) creation of initial shared secret key, (3) registration of the recognized clouds of other healthcare and patients, (4) updating shared key, (5) Secure data transmission, and (6) privacy preservation of stored data on cloud.

3.1. Generation of Initialization Parameters

The description of the notations of the parameters used in the proposed scheme is given in Table 2.

Table 2. Description of the notations used in the proposed scheme.

Notation	Description
HC_{id}	Healthcare identification number
P_{id}	Patient identification number
GHC_{id}	Guest healthcare identification number
$y^2 \equiv x^3 + ax + b \ mod \ p$	EC equation
p	The large prime number obtained from chosen EC
a,b	Constants of EC such that $4a^2 + 27b^2 \ mod \ p \neq 0$
G	The base point which produces the subgroup of the elliptic curve that has a large prime as its order
n	Order (number of a point) of the subgroup. nG = 0, prime n is large
h	The cofactor of the subgroup, which is the ratio $\frac{\lvert E \rvert}{\lvert E_p \rvert} = \frac{order \ of \ elliptic \ curve \ E}{order \ of EC \text{ over a prime field } \ E_p}$. h should be small $h \leq 4$, preferably, h = 1
pr_{HC}	Healthcare private key
$pr_{patient}$	Patient private key
pr_{GHC}	Guest healthcare private key
pub_{HC}	Healthcare public key $(pr_{HC} \times G)$
$pub_{patient}$	Patient public key $\left(pr_{patient} \times G\right)$
pub_{GHC}	Guest healthcare public key $(pub_{GHC} \times G)$
H_{HC}	The hash function used by healthcare
nList	A list containing reference IDs $(HC_{id}, \ P_{id}, \ GHC_{id})$
HASH	Signing cipher message Cm hash function
ssk	Shared secret key
SNP	Share node point = (ssk × G)
PRK	Private random key (bit size is dependent on the type of EC)
K	A random integer is chosen from (1, p − 1)
CM	The ciphertext (all encrypted points)
$\oplus$	Exclusive OR operation
+	Addition operation used for the ECC encryption process
T	Timestamp

The following processes take place during the initialization procedure:

- The *HC* assigns an identification number HC_{id} to itself, P_{id} to each patient, and GHC_{id} to each guest's healthcare;
- HC, the central sensor in each patient, and each GHC are preloaded with encryption and decryption algorithms.

3.2. Creation of Initial Shared Secret Keys

The shared secret key is generated and kept by the HC. This makes it possible for patients and the GHC to successfully decrypt the cipher text. Therefore, utilizing its H_{HC}, HC_{id}, and PRK, the HC generates the initial ssk. Algorithm 1 illustrates the key generating procedure.

Algorithm 1: Initial shared secret key creation

Input: H_{HC}, HC_{id}, PRK
Output: ssk
ssk = $H_{HC}\ (HC_{id}) \oplus PRK$
initial shared secrete key ← *ssk*

3.3. Registration of Patients and Guest Healthcare Centres

To benefit from the facility of the proposed IoMT model, users have to be registered by the HC. Patients and guest healthcare centres that benefit will be given identification codes. Each time a patient or healthcare centre registers, the shared secret key is updated before adding the client to the network.

3.4. Updating the Shared Key in the Proposed IoMT Model

Updating the shared secret key is performed by the HC. Each time a patient, doctor, or GHC registers/leaves the network, the shared secret key has to change. The proposed scheme ensures that the ssk is updated before a new client is added to the network and after an existing client in the IoMT network leaves the network using Equation (1).

$$\text{ssk} = H_{HC}\left(P_{id}\right) \oplus \text{ssk} \tag{1}$$

If a client P_{id}/GHC_{id} joins the network, the HC adds its identification number to the nList and updates the ssk. Algorithm 2 outlines the steps necessary for a new client *Ni* to join the IoMT.

Algorithm 2: Procedure for updating ssk, SNP, and nList when a client joins the HC network

Input: id_{Ni}, H_{HC}, ssk
Output: Updated ssk, snp and nList

1. Start
2. HC generate new id_{Ni} for the new client
3. HC update key ssk = $H_s\left(id_{Ni}\right) \oplus$ ssk;
4. HC update shared point snp = ssk × G;
5. HC update its nList
6. Stop

HC removes the P_{id} or GHC_{id} of the client departing the network, updates the ssk using Equation (1), and removes the related hashed ID from its node list. Algorithm 3 provides instructions for carrying out a client's departure operation.

Algorithm 3: Procedure for updating ssk, snp, and nList when a node leaves WSN

Input: H_s, id_{Ni}, ssk
Output: Updated ssk, snp and nList

1. Start
2. Node sends leave $\left(id_{Ni}\right)$
3. HC removes the reference Id of the leaving client
4. HC update key ssk = $H_s\left(id_{Ni}\right) \oplus$ ssk
5. HC update shared point snp = ssk × G;
6. HC update its nList
7. Stop

3.5. Secure Data Transmission and Storage

Maintaining security requirements during data transmission is of paramount importance for reliable and dependable communication. For confidentiality in the communication, the message being transmitted must be encrypted by the client (sender) before transferring it to the recipient. Thus, illegal access to the message being transferred is prevented [60,61]. Integrity, authentication, authorization, and non-repudiation can be obtained through the process of signing and verification of the transmitted data. A client that wants to store information in the cloud has to create public and private keys using the agreed EC. The recipient (HC) verifies if the request is coming from a registered member. The request is granted by the HC if the HC generates its own private and public keys and sends the public key as a response to the request. Both the client and HC form a shared secrete key, ssk, for the encryption and decryption for the communication. Figure 2 depicts the scenario.

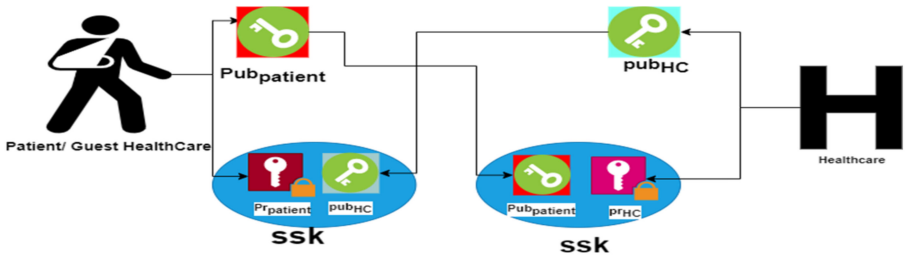


Figure 2. Formation of the shared secrete key using ECDH.

Figure 3 describes how patients' information is stored in the cloud being used by the HC where the patient has registered and how the information can be retrieved anywhere, anytime by the patient or any authorized user. The message originating from a client is first encrypted. The hash value of the encrypted message is taken and converted to an integer. The encrypted message is then appended with time stamp T and signed using ssk before it is sent to the cloud. When the message is received in the cloud, the message is authenticated to be sure that the message is from a registered client of the cloud and has not been tampered with during transmission. If it is found to be registered, the cloud uses the integer generated from the hash value to determine the location in the cloud where the information is to be stored using B+ file organization and updates the patient record with the integer. The encryption and decryption processes in the proposed model are discussed as follows:

i. **Encryption of the message**

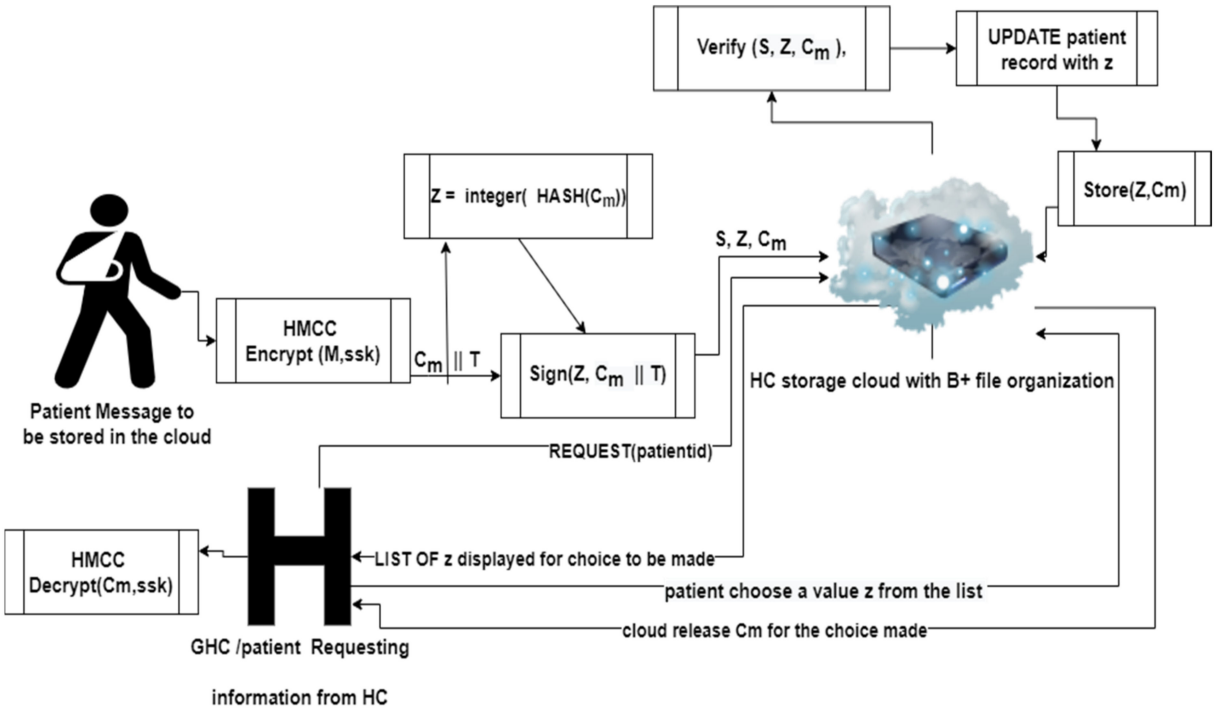


Figure 3. Secure information storage and retrieval from the proposed IoMT model.

The data to be encrypted are divided into blocks. Each block is converted to binary digits. Algorithm 4 describes how plaintext is divided into blocks of an equal number of characters.

Algorithm 4: Conversion of plaintext into blocks of an equal number of characters

Input: Ptext
Output: blocks // a set of strings with equal number of characters

1. LET M = number of characters in Ptext
2. COMPUTE the size of each block from the chosen elliptic curve, $N = FLOOR\left(\frac{p-8}{8}\right)$
3. COMPUTE the number of blocks in the Ptext $numblocks = CEIL\left(\frac{M}{N}\right)$
4. LET B = [ ]
5. t = 1
6. FOR i = 1 TO num blocks
7. B[i] = Ptext[t: t+ N]
8. t = t + N
9. ENDFOR
10. $blocks \longleftarrow B$
11. STOP

A modified version of the Caesar algorithm is used for encryption and decryption purposes. The modification to the existing Caesar algorithm became very necessary as a result of its vulnerability to different cryptanalysis attacks, low key space, and lack of diffusion and confusion properties which are required for modern-day cryptographic algorithms. The following discusses the modifications made to the existing Caesar cipher which make it unbreakable. Both encryption and decryption Caesar algorithms were modified. The following modifications were made to the existing Caesar cipher:

i. The generation of a key for rearranging ASCII characters from the ssk key;
ii. The introduction of confusion property into ciphertext by reshuffling of ASCII characters using Modified Columnar Transposition technique;
iii. The introduction of diffusion and confusion property through the modification of bits of the mixture of the plaintext and secrete key, and;
iv. The determination of ciphertext characters based on the location of the character in the plaintext and the location in the character set in the reshuffled ASCII characters. This makes the character set a lookup table for the modified Caesar cipher instead of the mere shifting of alphabets in the traditional Caesar cipher. The lookup table is based on modular addition and subtraction in modulus 256. During the encryption process, an ASCII character found at location i on the block of plaintext and at location j on the lookup table (the reshuffled ASCII characters table) will be represented by an ASCII character at location $t = (i + j) mod\ length(lookuptable)$ of the lookup table to form ciphertext character. During the decryption process, an ASCII character found at location j on the ciphertext block and location t on the lookup table will be represented by an ASCII character at location $i = (t - j) mod\ length(lookuptable)$.

This concept is based on the quotient remainder theorem.

$$\text{Let } t = (i + j) mod\ m = (i\ mod\ m + j\ mod\ m) mod\ m \tag{2}$$

From the quotient remainder theorem, i and j can be written as:

$$i = mq_1 + rnd_1, \quad i\ mod\ c = rmd_1$$

$$j = mq_2 + rmd_2, \ \ j\ mod\ m = rmd_2$$

$$where\ m\ is\ the\ divisor,\ q_1\ and\ q_2\ are\ quotients\ in\ each\ case$$

$$and\ rmd_1,\ rmd_2\ are\ remainder\ in\ each\ \ case$$

$$t\ can\ be\ defined\ in\ terms\ of\ i\ and\ j\ as\ follows$$

$$t = mq_1 + rmd_1 + mq_2 + rmd_2$$

$$t = m(q_1 + q_2) + rmd_1 + rmd_2$$
$$(t - i) = m(q_1 + q_2) + rmd_1 + rmd_2 - (mq_1 + rmd_1)$$
$$= m(q_1 + q_2 - q_1) + rmd_1 + rmd_2 - rmd_1$$
$$= mq_2 + rmd_2 = j$$
$$likewise\ (t - j) = m(q_1 + q_2) + rmd_1 + rmd_2 - (mq_2 + rmd_2)$$
$$= m(q_1 + q_2 - q_2) + rmd_1 + rmd_2 - rmd_2$$
$$= mq_1 + rmd_1 = i$$

Hence, encryption and decryption can be conducted as base on this theorem. A detailed description of the operations of the modified Caesar encryption module is given in Algorithm 5a.

Algorithm 5a: Modified Caesar Encryption Module (HMCC)

Input: set of Blocktext, Binkey
Output: a set of encryptedblocktext:
Variables: caesarchar: 1D array of ASCII characters as reshuffled

```
START
LET alphabet = 0:255
COUNT the number of zeros and ones in binkey, X
confusionkey = CALL Randgen(X,256)// Algorithm 5d
caesarchar = Reshuffleforencryption(confusionkey, alphabet)// Algorithm 5c
Blocktext = Blocktex XOR Binkey
Blocktext = Bitmodification(Blocktext,x1)// Algorithm 5b
INITIALIZE c = ' ' //a string with no character
FOR i = 1 TO number of characters in Blocktext
    ch = Charcaesarxor (i)
    FOR j = 1 TO LENGTH(alphabet)
        CONVERT ch to decimal number chno
        IF chno = caesarchar(j) THEN
            t = (i+j) MOD (LENGTH(alphabet))
            BREAK
        ENDIF
        END FOR
        c = CONCATENATE (c, caesarchar(t))
END FOR
LET encryptedblocktext = c
STOP
```

To ensure diffusion and confusion in the modified Caesar cipher, a bit modification function was introduced for encryption, and a reverse bit modification function which reverses encryption bit modification was introduced during the decryption process. In these modules, bit grouping, bit reshuffling, and exclusive OR operations were used to ensure proper mixing of key and plaintext bits, which ensures diffusion and confusion in HMCC. Algorithm 5b gives the procedure for bit modification, which introduces diffusion and confusion into the modified Caesar algorithm.

Algorithm 5b: Encryption bit manipulation process

```
Module encmanipulatedbits = Encbitmanipulation(binarystr, randomseed)
Output: encmanipulatedbits
1. START
2. randb = CHANGE randomseed to string of bits
3. rseeds = COUNT the number of bits in randb
4. Randkeys = ARRAY [1.. rseeds] of random seeds // geerated using randomseed
5. nibbleleft = ''
6. nibbleright = ''
7. sizeofbit = LENGTH (binarystr)
8. FOR i = 1 to sizeofbit, STEP 1byte
    i. onebyte = binarystr(i:i + 7)
    ii. nibbleleft = CONCATENATE (nibbleleft, onebyte(1:4))
    iii. nibbleright = CONCATENATE(nibbleright, onebyte(5:8))
        END FOR
9. FOR j = 1 to rseeds
    i. Seed1 = Randkeys(j)
    ii. nibblexor = nibbleleft XOR nibbleright
    iii. Fbyte = CONCATENATE(nibblexor, nibbleright)
    iv. confusionkey = GENERATE sizeofbit random integers from Seed1
    v. manipbit = CALL Reshuffleforencryption(Fbyte, confusionkey)
    vi. nibbleleft = manipbit(1: sizeofbit/2)
    vii. nibbleright = manipbit(sizeofbit/2 + 1: sizeofbit)
        END FOR
10. encmanipulatedbits = manipbit
11. STOP
```

During the encryption/decryption process, ASCII characters are reshuffled to form a lookup table for the determination of ciphertext/plaintext in each block. The algorithm to achieve the procedure is given in Algorithm 5c.

Algorithm 5c: Pseudocode for reshuffling ASCII characters

```
Module encryptionreshufflement (pblocktext, ckey)
Output
rearrangedtext [1 . . . C]: array of characters
Variables      t, col, a,    C message[1   C]: arrays of characters
1.    START
2.    C = COUN T the number of characters in pblocktext
3.    msg = ''
4.    t = 1
5.    FOR col = 1 TO C
6.        a = ckey[col]
7.        msg[ t] = pblocktext[a]
8.        t = t + 1
9.    ENDFOR
10.   rearrangedtext = message
11.   STOP
```

Algorithm 5d generates an array of random number. This array of random numbers designates the pattern that is used for scrambling ASCII characters or bits.

Algorithm 5d: Pseudocode for random number generation

Module Randgen (seed, n)
Output: R[1 . . . n]: Array of randomly generated integers

1. START
2. INITIALIZE random number generator with seed
3. GENERATE array R of n random integers from Random number generator
4. STOP

ii. **The signing of the Encrypted Message**

By encrypting the message to be sent, the confidentiality of the message is achieved. However, the integrity of the message is not achieved because a third party can modify the encrypted message without the knowledge of both the client and HC. Therefore, to be convinced of the integrity of the message, the client appends a signature to the encrypted message using its private key pr_{NiA}, which depends on the Elliptic Curve Digital Signature Algorithm (ECDSA). The description of how the sending node A appends its signature to the encrypted message is given in Algorithm 6.

Algorithm 6: Appending signature of the sending node to the encrypted message

INPUT: encrypted points
OUTPUT: Signed message

1. START
2. LET $C_m \leftarrow encrypted\ points$
3. LET $M_{sent} = (C_m, t_{o,})$ / / Append timestamp t_o to the encrypted message
4. COMPUTE e = HASH(M_{sent})
5. COMPUTE z = the first p bits of e starting from left
6. GENERATE an integer k randomly
7. COMPUTE (x, y) = k $\times$ G
8. COMPUTE c = x mod p where c $\neq 0$
9. If c = 0 go to 6
10. COMPUTE d = (z + $pr_{NiA} *$ c) k $^{-1}$
11. If d = 0 go to 6
12. (c,d) $\leftarrow$ ciphertext signature
13. STOP

iii. **Signature Verification of the signed and encrypted message**

The receiving HC B does not just start the decryption of any received message. It first verifies the authenticity of the received message to verify that the received message originated from the sending client A. HC B verifies the received message using the public key pub_{NiA} of node A. Algorithm 7 describes the procedure taken by receiving node B in verifying the received message $M_{received}$.

Algorithm 7: Verification of received signed message

Input: $M_{received}$, pub_{NiA}, ciphertextsignature (c,d)
Output: Verified ciphertexts

1. START
2. Verify that the integers (c,d) $\in$ 1, 2, 3, . . . ,p $-$ 1)
3. COMPUTE e = HASH($M_{received}$)
4. COMPUTE z = the first p bits of e starting the left hand side
5. COMPUTE $u1 = e * d^{-1}\ mod\ p$
6. COMPUTE $u2 = c *\ d^{-1}\ mod\ p$
7. COMPUTE (x, y) = u1 $*$ G + u2 $* pub_{NiA}$
8. Verified $\leftarrow$ c $\equiv$ x mod p
9. STOP.

iv. **The decryption of the verified received message** $M_{received}$

The same information obtained from the modified Caesar key for generating the seed used in generating the confusion key is also obtained to ensure that the random number generated during the encryption process is also generated during the decryption process. Again, the module Reshuffleforencryption used for scrambling ASCII characters during the encryption process is also used during the decryption process. This is also done to ensure that the same arrangement used for the ASCII characters during the encryption process is used during the decryption process. However, without knowing the key, it is extremely hard to know this arrangement. The description of how ciphertext is converted back to plaintext in modified Caesar is represented by Algorithm 8a, while Algorithm 8b describes the process of reversing the bit modification performed during the encryption process.

Algorithm 8a: Modified Caesar Decryption Module

```
Module DMCaesar (Blocktext, Binkey)
Output: decryptedblocktext: string of characters
Variables: caesarchar: 1D array of ASCII characters as reshuffled
1.  START
2.  LET alphabet = 0:255
3.  X = COUNT the number of o in binkey
4.  x1 = COUNT the number of zeros in binkey
5.  confusionkey = CALL Randgen(X,256)//Algorithm 5d
6.  caesarchar = CALL Reshuffleforencryption(confusionkey, alphabet) 5c
7.  ptext = ''
8.  length = COUNT the number of characters in Blocktext
9.  FOR i = 1 TO length
10.   ch = Charcaesarxor (i)
11.   FOR j = 1 TO LENGTH(caesarchar)
12.     IF ch = caesarchar(j)
13.         t = (j − i) MOD (LENGTH(alphabet))
14.         BREAK
15.     ENDIF
16.   END FOR
17.   pcharl = caesarchar(t)
18.   p = CONCATENATE (p, pchar)
1.  END FOR
19. p = CALL Reversebitmodification(p, ×1) // Algorithm 9
20. p = P XOR binkey
21. LET decryptedblocktext = p
22. STOP
```

3.6. Privacy Preservation in the proposed IoMT Model

The identity of the patient is not shared by the cloud. In addition, the health records of the patients are stored in encrypted form on the cloud so as to avoid unauthorized access. In order to make it easy to search through the encrypted data, the encrypted data are stored on the cloud using B+ tree file organization, in which the information is stored using (key, value) pair form. Algorithm 9 gives the description of how the key is generated.

Algorithm 8b: Decryption Reverse Bit Modification Process

Module blockbits = Dec Reversebitmodification(binarystr, radseed)
Input: binarystr, randomseed**Output**: reversebinarystr

1. START
2. randb = binary value of randomseed
3. rseeds = number of bitsin randb
4. Randkeys = array of generated random seeds from radseed
5. bsize = number of bits in binarystr
6. Rbinrystr = binarystr
7. FOR i = rseeds to 1
   i. s1 = Randkeys[i]
   ii. confusionkey = GENERATE bsize random integers from s1
   iii. Rmanipbit = CALL Reshuffleforencryption(Rbinarystr, confusionkey) // algorithm 5c
   iv. nibbleleft = Rmanipbit(1: bsize/2)
   v. nibbleright = Rmanipbit(bsize/2 +1: bsize)
   vi. nibblexor = nibbleleft XOR nibbleright
   vii. Rmanipbit = CONCATENATE(nibblexor, nibbleright)
   END FOR
8. hx = bsize / 2
9. Rm1 = Rmanipbit(1:hx)
10. Rm2 = Rmanipbit(hx+1:bsize)
11. pmsg = ''
12. FOR s = 1 to hx
   i. Lm = Rm1(s:s+3)
   ii. Rm = Rm2(s:s+3)
   iii. pmsg = CONCATENATE(pmsg, Lm, Rm)
   END FOR
13. reversebinarystr = pmsg
14. STOP

Algorithm 9: Storing encrypted file on cloud storage using B+ file organisation

Input: HC private key (ssk, Cm)

1. START
2. COMPUTE p = number of bits in ssk
3. COMPUTE e = HASH(Cm)
4. ENCRYPT ence = HMCC(e) // encrypt using algorith 5a
5. COMPUTE z = the first p bits of ence starting from the left hand side
6. CONVERT z to an integer
7. COMPUTE z = z + sid + vid //append te sensor id and value that represent the interpretation of the measured value from te sensor
8. UPDATE patient record by adding z
9. STORE value on cloud using z as the key of B+ file organization
10. STOP.

Cm in Algorithm 9 represents the encrypted data. ssk is the shared secret key between the HC and patient. Integer z is obtained by appending sensor identification number sid and the value measured by the sensor represented by vid to the integer obtained from the hash value of encrypted data from a particular sensor. The data can be decrypted only by the authorized user using a private key provided by the HC. If the patient visits another healthcare centre GHC and there is the need to access the patient's health record from the cloud, the patient will be asked to supply the ssk so as to be able to decrypt the content of the health record. Knowing the hash value of the file location of the encrypted data cannot

help an intruder in obtaining the content of the file. Hence, the HC and patient are in full control of determining which information is to be shared and which is not. This implies that the privacy of the patient's information is preserved. To search for a particular piece of information from the encrypted data in the cloud, the user only needs to analyse the sid and vid components of the record keys. Only the records that meet up with the criteria searched for will be decrypted, and not the entire data. A user can only access information pointed to by the key on the cloud.

Retrieving a particular piece of information from the cloud uses Algorithm 10. The user chooses which information to retrieve from the cloud and the search is made, followed by decryption of the file.

Algorithm 10: Searching for a particular file on the cloud with B+ file organization

Input: ssk, patientid

```
START
INPUT sid, vid
LET found = FALSE
WHILE found = FALSE
        x = CONCATENATE sid and vid
        k = SELECT key from the list available to the user and extract sid and vid
            component of the key
        IF x = k THEN found = TRUE
        DECRYPT the file with the key using ssk
ENDWHILE
STOP.
```

The presented privacy model has the following advantages:

1. The space on the cloud is preserved as there is no need to store encrypted data redundantly, contrary to the approach in [59];
2. Access to the data on cloud is based on B+ file organization which can be done both sequentially and randomly. Hence, search time is greatly reduced;
3. There is no need for intensive computation on the encrypted data for searching purposes. This is an improvement to the existing FHE implementations;
4. B+ tree is a self-balancing data structure for executing accurate and faster searching, inserting, and deleting procedures on data;
5. B+ trees do not waste space;
6. It takes an equal number of disk accesses to fetch records;
7. B+ trees have redundant search keys, and storing search keys repeatedly is not possible;
8. Faster search queries as the data are stored only on the leaf nodes.

3.7. Analysis of the Proposed IoMT Model

The HMCC was examined on a Hewlett Packard laptop equipped with an AMD E1–1200 APU with Radeon(TM) HD Graphics 1.40 GHz, 4.00 GB (3.59 GB useable), a 64-bit Windows 10 operating system, and an x64-based processor. The Scientific Python Development Environment (Spyder), Copyright 2009–2020, was utilized during the development and analysis. The proposed IoMT system was subjected to security testing against a variety of attacks. The following subsections discuss various security analyses carried out on the proposed IoMT model.

i. **Some samples of plaintext were carefully used as input into HMCC**. The output ciphertexts obtained were analysed by examining how the cryptosystem handled these known plaintexts. Samples of plaintext used were plaintext with the same characters all through, plaintext with repetitive terms, and plaintext with non-

repetitive terms. Analysis of the output ciphertexts from HMCC were carried out using the following procedure:

1. Encrypt a block of plaintext P1 that contains the same characters throughout;
2. Encrypt a block of plaintext P2 that contains repetitive characters;
3. Encrypt block of plaintext P3 that contains distinct characters;
4. Examine the ciphertext of each block obtained from steps 1 and 3.

ii. **Spectral Frequency Analysis of Modified Caesar Cipher:** Information regarding plaintext, key, or both can be obtained by cryptanalysts by making use of frequency analysis of ciphertexts. HMCC resistance to frequency analysis attacks was carried out using spectral frequency analysis. Frequency analysis of some samples of plaintext and of the ciphertext produced when the plaintext passed through the encryption algorithm of HMCC were examined by plotting the histogram of the frequencies of the characters in the plaintext and ciphertext.

iii. **Differential Cryptanalysis of HMCC:** HMCC resistance to differential attacks was evaluated. The encryption results when two plaintexts that differ only by one bit were encrypted with the same key were analysed. Net Pixel Change Rate (NPCR) and Unified Average Changing Intensity (UACI) were used as metrics for measuring HMCC's resistance to cryptanalysis attack. The NPCR and UACI were determined using the formula in Equations (3) and (4), respectively, for two ciphertexts C1 and C2 of length l.

$$NPCR = \frac{\sum_{i=1}^{l} W(i)}{l} \times 100 \tag{3}$$

$$UACI = \frac{100}{l \times 256} \sum_{i=1}^{l} |C1(i) - C2(i)| \tag{4}$$

where $W(i) = \begin{cases} 0, \; if \; C1(i) = C2(i) \\ 1, \; if \; C1(i) \neq C2(i) \end{cases}$

The following procedures were followed for the differential cryptanalysis of MCC:

1. Encrypt a sample of plaintext P using MCC encryption module to obtain ciphertext C1;
2. Change a symbol of P to another character to obtain a new plaintext P1;
3. Encrypt the plaintext P1 using MCC encryption module to obtain ciphertext C2;
4. Determine $NPCR = \frac{\sum_{i=1}^{l} W(i)}{l} \times 100$ where $W(i) = \begin{cases} 0, \; if \; C1(i) = C2(i) \\ 1, \; if \; C1(i) \neq C2(i) \end{cases}$;
5. Determine $UACI = \frac{100}{l \times 256} \sum_{i=1}^{l} |C1(i) - C2(i)|$;

iv. **HMCC Information Entropy Analysis:** All of the 256 ASCII symbols are used by HMCC, so the expected maximum entropy is 8 when the formula in Equation (5) is used for entropy calculation.

$$H(m) = \sum_{i=0}^{2^N - 1} P(m_i) \log_2 \left(\frac{1}{P(m_i)} \right) \tag{5}$$

where N signifies the bit size of message m, 2^N represents the symbols' sample space, $P(m_i)$ denotes the probability of m_i, and $\log_2$ represents the logarithm in base 2. The entropy is expressed in bits. The entropy $H(m)$ of a message m encrypted with a 2^N symbol sample space is N if Equation (5) is applied. The following procedures were used to carry out the entropy analysis of the MCC cipher:

1. Encrypt a sample of plaintext P using HMCC encryption module to obtain ciphertext C;
2. Find distinct unique ASCII characters Ptext and Ctext from P and C, respectively;
3. Determine the frequency of each of the unique characters in Ptext and Ctext from P and C to respectively obtain pfrequency and cfrequency;

4. Determine the probability of each unique character in P and C, respectively, using the formula pcharprobability [i] = pfrequency[i]/LENGTH(P) for I = 1 to LENGTH(P) and ccharprobability [i] = cfrequency[i]/LENGTH(C) FOR i = 1 to LENGTH(C);
5. Determine the entropies of the plaintext and ciphertext by applying Equation (5):

   i. $\boldsymbol{H(P) = \sum_{i=0}^{2^N-1} pcharprobability\ [i] \times log_2\left(\frac{1}{pcharprobability\ [i]}\right)}$

   ii. $\boldsymbol{H(C) = \sum_{i=0}^{2^N-1} ccharprobability\ [i] \times log_2\left(\frac{1}{ccharprobability\ [i]}\right)}$

6. Compare H(P) and H(C) with the maximum entropy value.

v. **Autocorrelation Analysis of HMCC:** The autocorrelation function is used to define the resemblance of two sequences. Cryptanalysts use autocorrelation to calculate secret key length in classical ciphers. A cipher that is resistant to autocorrelation attacks will produce ciphertext that has more uniform and lower autocorrelation than plaintext. The following steps were taken in order to carry out autocorrelation analysis:

   1. Encrypt a sample plaintext P to obtain ciphertext C;
   2. Find distinct unique ASCII characters Ptext and Ctext from P and C, respectively;
   3. Determine the frequency of each of the unique characters in Ptext and Ctext from P and C to respectively obtain pfrequency and cfrequency;
   4. Plot the graph of the ASCII values of characters in Ptext against pfrequency and the ASCII values of Ctext against cfrequency;
   5. Compare the two graphs.

vi. **Strict Avalanche Criterion of HMCC**

Two variants of SAC, namely Strict Plaintext Avalanche Criterion (SPAC) and Strict Key Avalanche Criterion (SKAC), were used to measure the strength of HMCC. Here, SKAC was adopted and also adapted for SPAC. Data sets as input of SKAC and SPAC tests were produced by the recommendation of [62]. A total of 100 sequences of random 128-bit plaintext blocks and keys were used to prepare the SKAC and SPAC matrices. A total of 12 and 800 different keys and plaintext block inputs were respectively used to test the strict key avalanche criterion (SKAC) and strict plaintext avalanche criterion (SPAC) of HMCC.

vii. **Comparative Security Analysis of the HMCC with Existing Ciphers**

A comparative analysis of HMCC with the existing symmetric ciphers in the literature was carried out. Two of the most widely documented block ciphers, namely DES and AES, were experimentally compared with HMCC. Information entropy, autocorrelation analysis, strict avalanche criterion, and NPCR and UACI values of differential cryptanalysis were used as parameters for the comparative analysis.

4. Results and Discussion

In this section, experimental results on the security analysis of the proposed IoMT model are discussed. Security analysis, encryption/decryption time analysis, and comparative security analysis with the existing systems are discussed.

4.1. Results of Security Analysis

i. **Man-in-the-Middle, replay, and denial-of-service attacks**: An attacker in MIMA is capable of impersonating both the sender and receiver [63]. If this attack succeeds, the attacker can send information to the receiver and also respond to the sender. An attacker in MIMA is also capable of resending the original message sent by a legitimate node to deceive the receiver. The activities of a MIMA attacker can also lead to denial-of-service attacks if such an attacker decides to regularly transmit false signals to deny authorized network users access to resources or services to which they are entitled. The proposed scheme is resistant to MIMA attacks of any form. A MIMA attacker that intercepts messages being transmitted between a client and the healthcare centre does not have enough information to compute a shared secret key between the two because it depends on the private keys of both,

which were not shared. Because the delivered message contains a timestamp T that identifies the precise instant when the message was sent, a MIMA attack that resolves to engage in replay assaults will fail. This establishes the timing difference used to identify any attack during the replay phase. A MIMA attacker who resolves to launch denial-of-service attacks is incapacitated because the attacker needs a shared secret key to encrypt the data, which is only possible through stealing IDs or becoming a legitimate user.

ii. **Analysis of Encryption algorithm of HMCC:** Table 3 shows plaintext samples and the corresponding resulting ciphertexts when plaintexts are submitted to the HMCC encryption technique.

Table 3. Sample ciphertext results from HMCC.

Plaintext	Ciphertext
aaaaaaaaaaaaaaa	\x8b5§\x82j\x99h¿®JÞØ\x9f\x943\x01
aaaaaaaabbbbbbbbdddwwwww	\x97¡ÞFöWès}±eÎ×{\x9b9\xad×ó\x8b!ð\x8b1»\x15ÀÑyç\x8aÎ
God is good all the time. Great	1\x95\x01èk\x00ÜätR\x116\x1f-Ô\x7fNnZ¢Ö\x19\x86_jÉQó\x83é′¾

According to the results in Table 3, all repetitive terms are omitted from the ciphertext, and it is difficult to verify from ciphertext only the characters in the plaintext that are repeated. As a result, using repetitive phrases in plaintext provides no useful information to the cryptanalysts to attack HMCC.

iii. **Spectral Frequency Analysis of Modified Caesar Cipher:** The result of the frequency analysis of the plaintexts and their corresponding ciphertexts are given in Table 4 labelled serial number 1–3, and the frequency analysis of the sampled plaintexts and the corresponding obtained ciphertexts are respectively shown by the histograms in Figures 4–6.

Table 4. Sample plaintexts and corresponding ciphertexts from HMCC for spectral frequency analysis.

S/N	Plaintext	Encrypted Text from HMCC
1	**Plaintext:** God is good all the time. Great **ASCII Values:** [71, 111, 100, 32, 105, 115, 32, 103, 111, 111, 100, 32, 97, 108, 108, 32, 116, 104, 101, 32, 116, 105, 109, 101, 46, 32, 71, 114, 101, 97, 116]	**Ciphertext:** hýa95×ó¾öí›?r‴·&TZô÷ýBÉe©Pz» **ASCII Values:** [104, 253, 97, 57, 6, 53, 215, 243, 190, 246, 237, 155, 63, 4, 114, 34, 16, 180, 183, 38, 84, 90, 244, 247, 253, 66, 201, 101, 169, 80, 122, 187]
2	**Plaintext:** Aaabbbbccccddde **ASCII Values:** [97, 97, 97, 98, 98, 98, 98, 99, 99, 99, 99, 99, 100, 100, 100, 101]	**Ciphertext:** î×-E†}}Pð+GÌWÿ ASCII values: [238, 215, 30, 69, 134, 125, 125, 80, 240, 43, 71, 204, 3, 4, 87, 255]
3	**Plaintext:** ccccccccccccccc ASCII Values: [99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99, 99]	Ciphertext: âûKÕŠ±‰3]Â¼N*í **ASCII Values:** [226, 251, 75, 213, 7, 138, 177, 137, 15, 51, 93, 194, 188, 78, 42, 237]

Figure 4 shows the histograms of the plaintext and ciphertext in S/N 1 of Table 4. From Figure 4, it can be seen that the frequency of the characters of the plaintext is not uniform, while the frequency of the ciphertext characters is relatively uniform. This characteristic reveals that HMCC is resistant to frequency analysis attack.

Figure 5 shows the histograms of the plaintext and ciphertext in S/N 2 of Table 4. From Figure 5, it can be seen that the frequency of the characters of the plaintext is not uniform. However, the frequency of the ciphertext characters is uniform. This result confirms the fact that attackers cannot use frequency analysis of the ciphertext to obtain any useful information about the plaintext.

Figure 6 shows the histograms of the plaintext and ciphertext in S/N 3 of Table 4. From Figure 6, it can be seen that the frequency of the single character of plaintext has been spread uniformly over several characters in the ciphertext obtained from HMCC such that it cannot be determined from the ciphertext that the plaintext contains the same character. This result also confirms that HMCC is resistant to frequency analysis attacks.

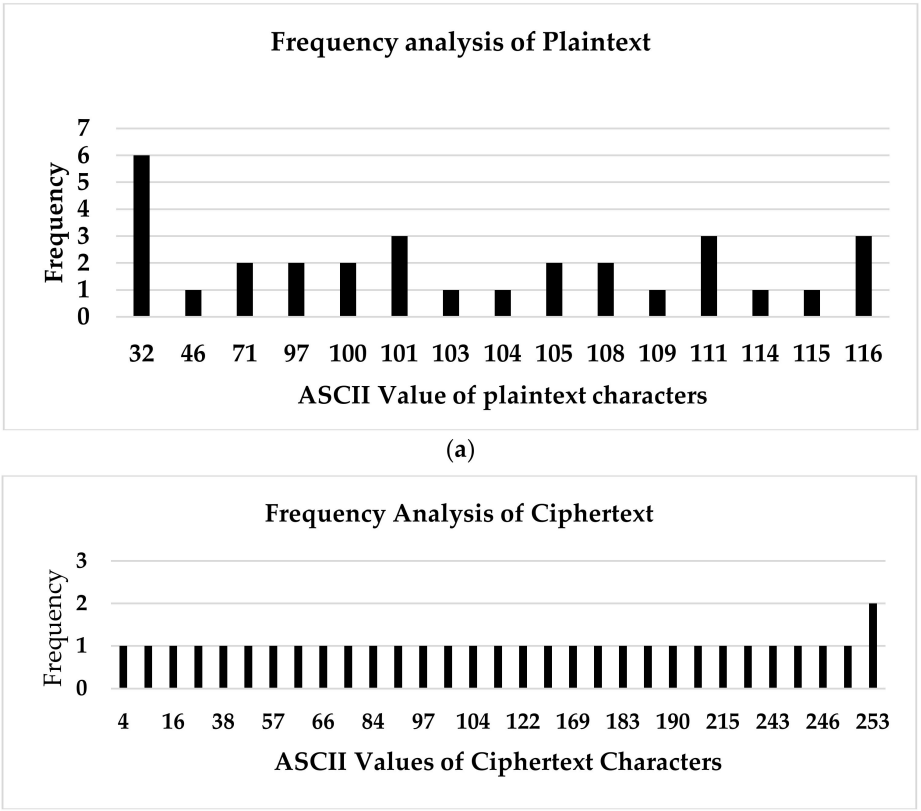


Figure 4. Histogram of the frequency of (**a**) sample plaintext and (**b**) ciphertext in Table 4 S/N 1.

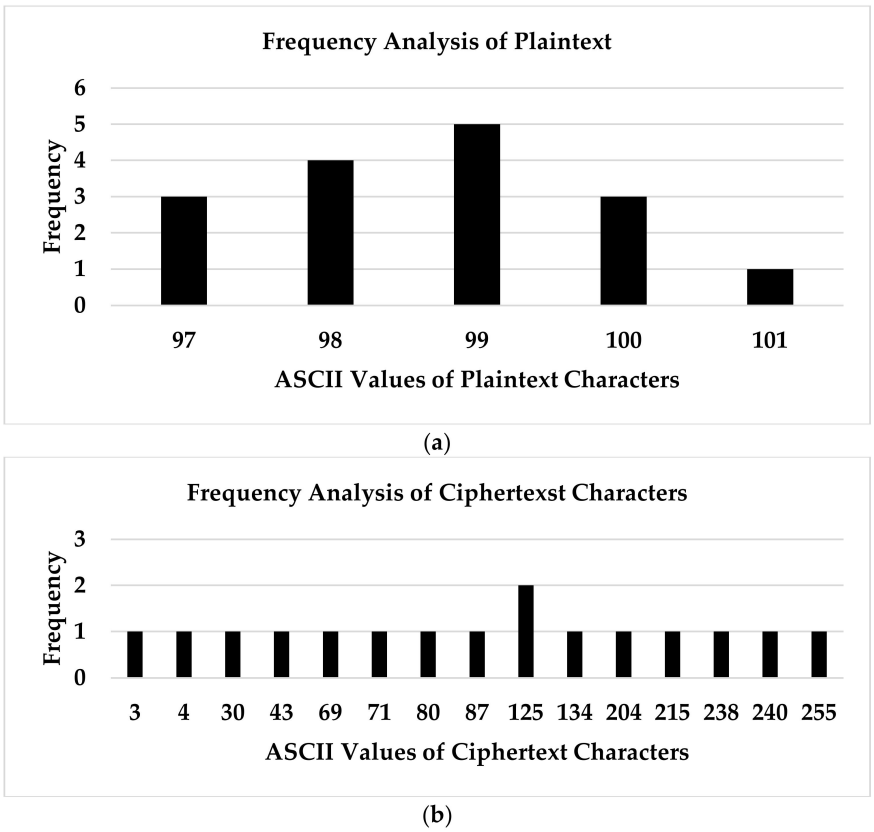
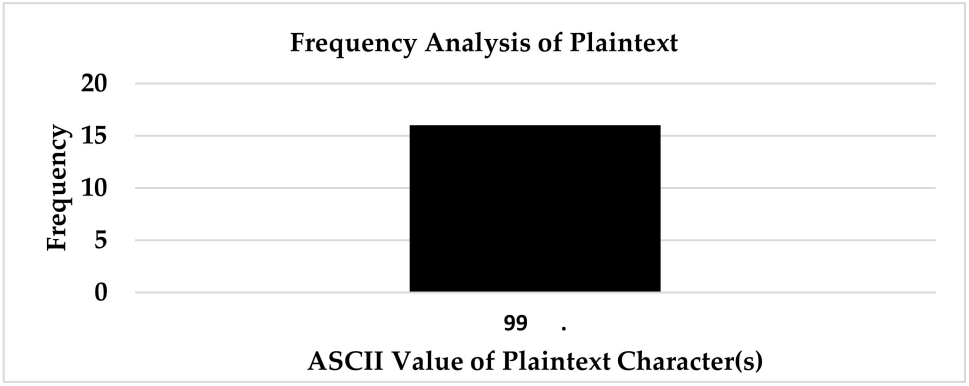
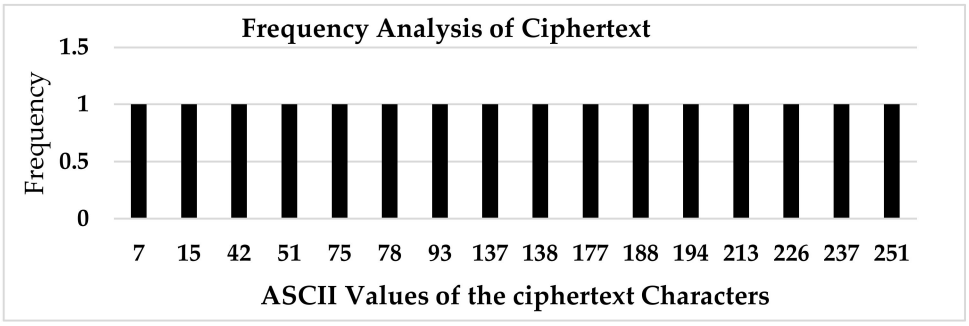


Figure 5. Histogram of the frequency of (**a**) sample plaintext and (**b**) ciphertext in Table 4 S/N 2.



(**a**)



(**b**)

Figure 6. Histogram of the frequency of (**a**) sample plaintext and (**b**) ciphertext in Table 4 S/N 3.

As can be seen from the sample outputs in Figures 4–6, the histograms show that the frequencies of the characters of the ciphertext are uniform. Where the plaintext contains the same characters, the ciphertext produced by HMCC spreads the frequency of the plaintext character over several characters uniformly. A cipher that is capable of producing ciphertext that has a relatively uniform frequency of characters is said to be resistant to frequency analysis attack. Hence, HMCC cannot be broken by applying a frequency analysis attack.

iv. **Differential Cryptanalysis of HMCC:** Table 5 displays some plaintext samples, the obtained ciphertexts from HMCC, and estimated NPCR and UACI.

Table 5. Sample Differential Cryptanalysis of HMCC.

Plaintext 1	Plaintext 2	C1	C2	NPCR (%)	UACI (%)
ccccccccccccccc	ccccccccãccccccc	BEÄÏpBr\n¬Ú$\r\x12\x89>‘“	§Ã\x83\x7f4A>\x11§öÅòCt\x80î	100	30.61
God is good all	God is gïod all	\x7fYös\x90\x82\x04äÙx¿íà¥\x08¤	es§ª\x16wú\x17¤ûÚÈO¤ma	100	32.64

A cipher that is resistant to differential assaults is one that causes a larger percentage of modification in ciphertext characters due to modest variations in plaintext characters [64]. The ideal values of NPCR and UACI are, respectively, 99.6093% and 33.4635% for a cipher that is resistant to differential cryptanalysis attack [65]. Whereas the NPCR and UACI fluctuate with secret keys, the values of NPCR and UACI from the samples provided in Table 5 reveal that HMCC is immune to differential cryptanalysis attack since in both samples, 100% of the ciphertext characters are altered due to a change in one character of plaintext. The values of UACI obtained in both cases are also very close to the standard 33.46%. This result shows that HMCC is not vulnerable to differential cryptanalysis attacks.

v. **HMCC Information Entropy Analysis**

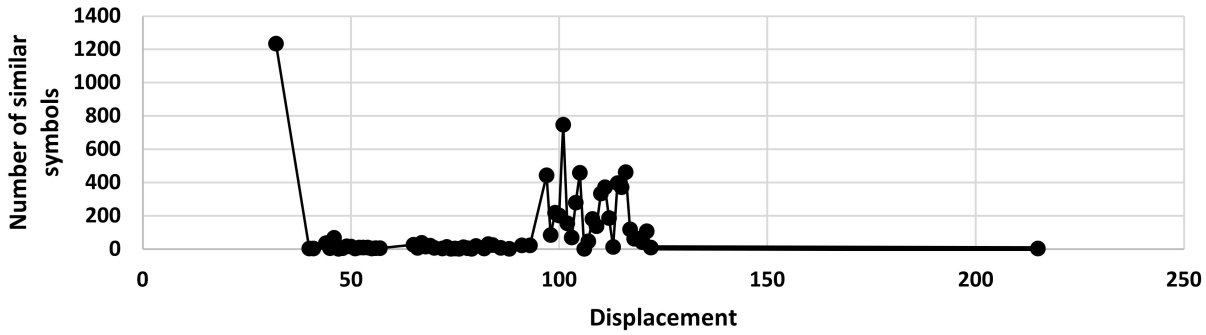
The entropies of a sample of plaintext and those of the ciphertext obtained when the plaintext is encrypted using an HMCC cipher were measured using the formula in Equation (5) and are given in Table 6.

Table 6. Sample entropy values obtained from entropy analysis of HMCC.

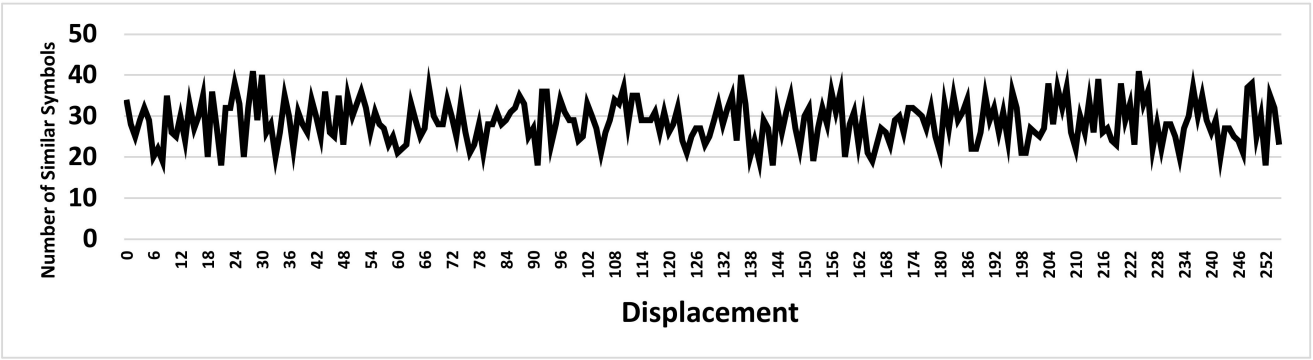
Text Type	Entropy
Plaintext	4.5009
Ciphertext	7.9773

The ciphertext entropy value (7.9773) is very closed to the expected entropy value (8). This result shows that the HMCC cipher's diffusion mechanism is strong. Any cipher with an effective diffusion property certifies that when a plaintext is encrypted, all of its symbols are transformed. Any cipher with an ineffective diffusion property will result in the ciphertext having numerous indistinguishable symbols, making the cipher vulnerable to entropy attack. As a result, HMCC is immune to information entropy analysis.

vi. **Autocorrelation Analysis of HMCC:** The autocorrelation function is used to define the resemblance of two sequences. Cryptanalysts use autocorrelation to calculate secret key length in classical ciphers. A cipher that is resistant to autocorrelation attacks produces ciphertext that has more uniform and lower autocorrelation than plaintext. Figure 7 depicts the graph of the autocorrelation of a sample of plaintext and the autocorrelation of the ciphertext generated after the sample plaintext was encrypted using the HMCC cipher.



(**a**)



(**b**)

Figure 7. (**a**) Auto Correlation of Plaintext; (**b**) Autocorrelation of Ciphertext.

The graphs of autocorrelation of plaintext and ciphertext in Figure 4 indicate that ciphertext autocorrelation is uniform and lower than plaintext autocorrelation. A cipher that produces ciphertext whose autocorrelation is uniform and lower than the autocorrelation of the plaintext from where it is produced is said to be resistant to autocorrelation attacks. Going by the autocorrelation analysis result of the ciphertext produced, it can be said that HMCCC is immune to autocorrelation cryptanalysis attacks.

vii. **Resistance of HMCC against Brute-Force Attack:** To be immune to a brute-force attack, the secret key space of a cryptosystem that is relevant for modern-day information security should be larger than 2^{100} [66]. Although the size of the key in HMCC varies, the minimum key size is 128 bits; therefore, the key size is 2^{128}. The key size can be increased to any number of bits by specifying the block size to be used when encryption is to be done. More entropy implies a broader and more unpredictable key search space, with smaller and more difficult-to-identify repetitions and correlations. In HMCC, all the 256 ASCII characters can be randomly used in the formation of the encryption keys. Hence, the entropy of the keys can be calculated using the formula in Equation (6).

$$\mathrm{E} = \log_2 R^N \tag{6}$$

where R is the size of symbol space. Applying Equation (6), the entropy of a key with the smallest key size in HMCC is 128 bits. An attacker would need 2^{128} or 3.40×10^{38} attempts for a successful brute-force attack on this key. Taking the example of Summit, a supercomputer capable of performing 200 PFLOPS (10^{15} floating-point operations per second) [67], according to Equation (7), it would take this computer 1.7×10^{24} years to crack the HMCC cryptosystem in its least key space, assuming that the computer can perform 1000 FLOPS per checking.

$$Years = \frac{key\ combination \times 1000}{FLOPS} \times 31536000 \tag{7}$$

viii. **Resistance of HMCC Against Classical Cryptanalysis attack:** Kirchhoff's principle states that the attacker has access to the encryption algorithms but no access to the secret key. With the encryption algorithm in the hand of a cryptanalyst, differential attacks such as cipher text-only and known/chosen plaintext attacks can be carried out [68]. In HMCC, the confusion and diffusion process is realized by modifying the bits of the encrypted text; in addition, the encryption of each block of plaintext is done such that each block is encrypted with a unique key. Again, the character set from where the ciphertext characters are obtained is determined by the key so each block uses a different character set. If a plaintext spans more than one block and the blocks are of the same characters, each block will be encrypted differently as different keys, and different character sets are used to determine the ciphertext characters. A sample output where the block size is 16 and the number of characters to be encrypted is 32 is given in Table 7.

Table 7. Sample ciphertext produced by HMCC.

Block	Plaintext	Ciphertext
1	áaaaaaaaaaaaaaaa	cM9¶\x19ßq\x10àìi\x9aÜÿÕy
2	áaaaaaaaaaaaaaaa	\\\x1dßñ]Qê\x84±aweGÆZ

This discovery also suggests that a dictionary assault on the HMCC will be no more productive than an attack using brute force. Hence, HMCC is thought to be resistant against dictionary assaults.

ix. **Strict Avalanche Criterion of HMCC**

The results of SKAC and SPAC tests for each 128-bit position are represented by substitution box (sbox) in Tables 8 and 9, respectively. The sbox in Table 7 represents SKAC(i,j) and SPAC (i,j), which respectively satisfies Equations (6) and (7) above.

$$\mathrm{SKAC}_{(i,j)} = \frac{1}{2^n} W\left(a^{ei}{}_j\right) = \frac{1}{2}$$

$$\mathrm{SPAC}_{(i,j)} = \frac{1}{2^n} W\left(a^{ei}{}_j\right) = \frac{1}{2}$$

where $W\left(a^{ei}{}_j\right) = \underset{all\ X \in (0,1)^n}{\sum a^{ei}{}_j}$ and ei is the unit vector with bit i equal to 1 and all other bits equal to 0.

Table 8. Strict Key Avalanche Criterion (SKAC).

i/j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0.4954	0.5019	0.4977	0.4942	0.4959	0.4966	0.4924	0.4975	0.502	0.5056	0.5056	0.4945	0.4964	0.4977	0.4995	0.4977
1	0.5031	0.4944	0.507	0.4994	0.506	0.4987	0.5049	0.4986	0.4888	0.5004	0.5026	0.5049	0.4973	0.4983	0.499	0.5059
2	0.5049	0.494	0.5039	0.4997	0.4937	0.5027	0.4973	0.5049	0.5008	0.5023	0.5084	0.5005	0.5005	0.5027	0.5002	0.4957
3	0.5036	0.4985	0.502	0.4973	0.5006	0.5031	0.4984	0.5047	0.5027	0.5077	0.4952	0.4941	0.4929	0.4973	0.4998	0.5073
4	0.4986	0.4889	0.499	0.4952	0.5059	0.4952	0.508	0.5038	0.5005	0.4951	0.4951	0.4988	0.4942	0.4961	0.4923	0.4946
5	0.4999	0.4948	0.5021	0.497	0.4966	0.4941	0.5052	0.4942	0.4945	0.5072	0.4948	0.4963	0.4921	0.4957	0.498	0.5012
6	0.5047	0.4858	0.5002	0.4933	0.4981	0.5012	0.5061	0.4909	0.4967	0.4875	0.502	0.4973	0.5064	0.4905	0.5047	0.4994
7	0.5004	0.4992	0.5059	0.4984	0.4943	0.4991	0.4952	0.4919	0.501	0.4941	0.4996	0.4982	0.4991	0.5009	0.5102	0.4945

Table 9. Strict Plaintext Avalanche Criterion (SPAC).

i/j	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
0	0.4913	0.4879	0.4975	0.4977	0.4898	0.5041	0.4987	0.5055	0.4928	0.4938	0.4877	0.4938	0.4957	0.4946	0.4896	0.5059
1	0.4881	0.4927	0.5088	0.4891	0.4934	0.5011	0.4926	0.4971	0.4925	0.4891	0.486	0.4982	0.4891	0.4993	0.4969	0.4988
2	0.4887	0.4931	0.4873	0.4933	0.4969	0.4921	0.498	0.4929	0.4934	0.4859	0.487	0.4876	0.4993	0.4941	0.5038	0.4933
3	0.4926	0.4906	0.4963	0.4965	0.4941	0.4952	0.4916	0.4881	0.4922	0.4968	0.4895	0.4986	0.4878	0.4961	0.4933	0.5012
4	0.4946	0.4995	0.491	0.4975	0.4904	0.497	0.4828	0.4941	0.4839	0.4938	0.4852	0.4861	0.4982	0.4983	0.5012	0.4974
5	0.5007	0.4827	0.5023	0.4933	0.5081	0.4918	0.4931	0.4977	0.4895	0.492	0.4951	0.495	0.4896	0.4962	0.4802	0.491
6	0.492	0.4956	0.4943	0.4916	0.4864	0.4861	0.4943	0.4961	0.4966	0.4893	0.499	0.4908	0.488	0.4988	0.4849	0.4978
7	0.4984	0.4869	0.4891	0.4956	0.4911	0.4893	0.5012	0.4926	0.4958	0.4918	0.4986	0.4962	0.4888	0.4968	0.4867	0.4992

The sboxes have 8-bit input (i) and 16-bit output (j). The value of SKAC(i,j) corresponds to the differences in input bits (ie, i = 1..8). The expected value of each SKAC(i,j)/SPAC(i,j) is 0.5.

As can be seen from Tables 8 and 9, SKAC(i,j)/SPAC(i,j) values are not exactly 0.5 as expected, but the values are very close to the expected value. The absolute error of each SKAC(i,j) can be obtained using Equations (8) and (9), respectively.

$$\underset{0 \le i,j \le n}{\mathrm{er(i,j)}} = \left|2 * \left(\mathrm{SKAC}_{(i,j)}\right) - 1\right| \tag{8}$$

$$\underset{0 \le i,j \le n}{\mathrm{er(i,j)}} = \left|2 * \left(\mathrm{SPAC}_{(i,j)}\right) - 1\right| \tag{9}$$

The maximum absolute errors eSKAC and eSPAC are obtained by applying Equations (10) and (11), respectively:

$$\mathrm{eSKAC} = \max_{1 \le i,j \le n} \left|2 * \left(\mathrm{SKAC}_{(i,j)}\right) - 1\right| \tag{10}$$

$$\mathrm{eSPAC} = \max_{1 \le i,j \le n} \left|2 * \left(\mathrm{SPAC}_{(i,j)}\right) - 1\right| \tag{11}$$

The obtained values for eSKAC and eSPAC are 0.0284 and 0.0396 , respectively. Substituting the values of eSKAC and eSPAC in Equation (6), the interval of SKAC can be written as:

$$\frac{1}{2}(1 - 0.0284) \le SKAC_{(i,j)} \le \frac{1}{2}(1 + 0.0284) = 0.4858 \le SKAC_{(i,j)} \le 0.5142$$

$$\frac{1}{2}(1-0.0396) \leq SKAC_{(i,j)} \leq \frac{1}{2}(1+0.0396) = 0.4802 \leq SKAC_{(i,j)} \leq 0.5198$$

Figures 8 and 9 show the distribution of eSKAC and eSPAC of HMCC, respectively. It can be seen from these Figures that only an insignificant number of bits have error values that are very close to eSKAC and eSPAC, respectively. eSKAC is less than 3, while eSPAC is less than 4. HMCC can, therefore, be said to have satisfied SKAC and SPAC within the range of very small error values.

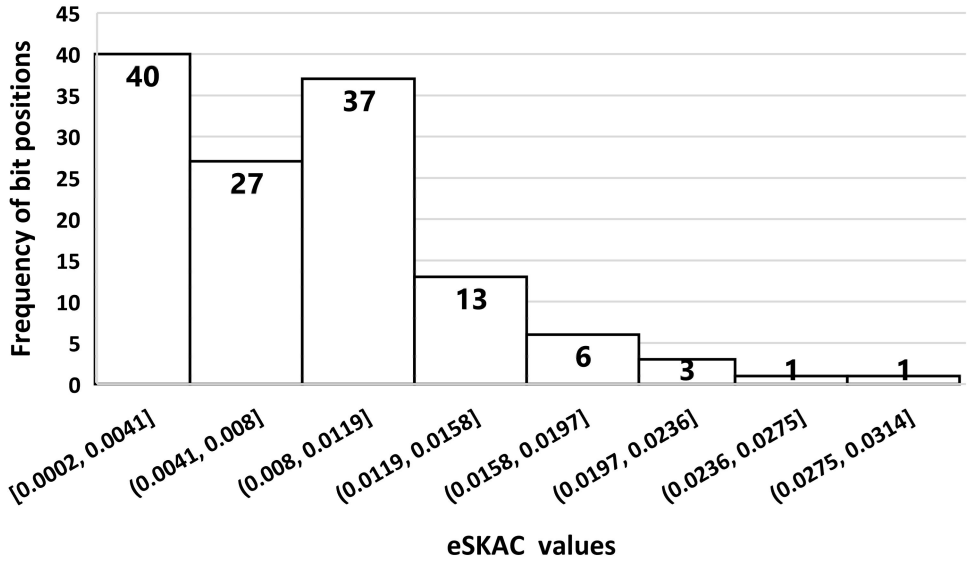


Figure 8. Histogram of SKAC error distribution of HMCC over bit positions.

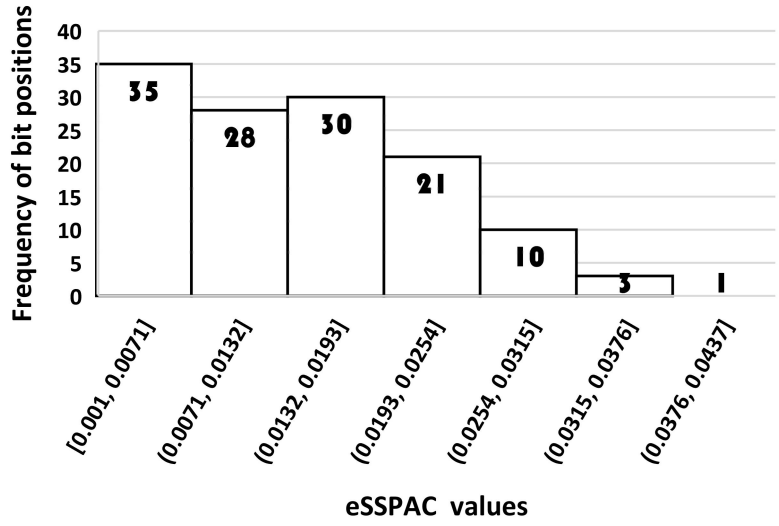


Figure 9. Histogram of SPAC error distribution of HMCC over bit positions.

4.2. Encryption and Decryption Time Analysis

A desirable encryption technique is resistant to cryptanalysis assaults while also being quick enough for real-time applications. 1171 symbols require only 0.593 s to encrypt and 0.565 s to decipher. As a result, HMCC is quick at both encrypting plaintext to produce ciphertext and decrypting ciphertext to produce plaintext. Hence, HMCC can be used in real-time applications.

4.3. Comparative Security Analysis of the HMCC with Existing System

The results of the comparative analysis of HMCC with AES and DES based on information entropy, autocorrelation analysis, strict avalanche criterion, and NPCR and UACI values of differential cryptanalysis are discussed in this section.

i. Comparative Analysis of Entropies of MCC, AES, and DES

Comparative studies of the entropies of the ciphertext produced from HMCC, AES, and DES were compared. The same samples of plaintext were used for all the ciphers. Table 10 compares the obtained entropies of the ciphers during the experiments. From Table 9, it can be seen that the three ciphers have very high entropies in all the samples considered, as the values are very close to the maximum value 8. Hence, HMCC entropy is comparable to the entropy of the standard AES algorithm.

Table 10. Comparison of entropies of HMCC, AES, and DES.

HMCC	AES	DES
7.9438	7.9302	7.94
7.9236	7.9345	7.9129
7.9662	7.966	7.962
7.8762	7.8831	7.8726
7.9667	7.9685	7.9613
7.952	7.9594	7.9533
7.9766	7.9748	7.971

ii. Comparative Analysis of MCC, AES, and DES based on NPCR and UACI Values of Differential Cryptanalysis

Differential cryptanalysis was carried out on HMCC, AES, and DES. Samples of the block texts used were randomly generated. A sample of the results when a 128-bit block of text and 128-bit key for HMCC and AES and a 64-bit block of text for DES were used in the experiments is shown in Table 11.

Table 11. The sampled result of comparative differential cryptanalysis of MCC, AES, and DES.

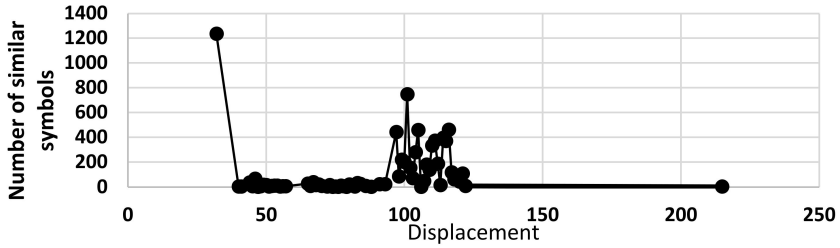
NPCR (%)			UACI (%)		
HMCC	**AES**	**DES**	**HMCC**	**AES**	**DES**
100	100	57.0312	27.549	27.6471	0.2237
100	100	46.875	32.549	43.6029	0.1838
100	100	51.5625	27.402	32.0588	0.2022
93.75	100	50	23.0147	44.951	0.1961
100	100	51.5625	30.4412	32.3284	0.2022
100	100	51.5625	23.7745	24.9755	0.2022
100	100	57.0312	38.7255	32.8431	0.2237
100	93.75	48.4375	34.5343	26.7647	0.19
100	100		27.1569	42.1814	
100	100		38.5784	31.9118	
100	100		36.4951	30.8088	
100	100		24.9755	40.3922	
100	93.75		26.8382	28.0637	
100	100		26.8627	32.7206	
100	100		26.1029	26.9608	
100	100		31.3725	34.5343	

The result of the differential cryptanalysis shown in Table 11 shows that HMCC and AES are strongly resistant to differential cryptanalysis attacks. This is affirmed by the value of NPCR which is 100% (the expected value), and the value of UACI which is very close to the expected value of 33.3% in most cases when a single character is altered in the input

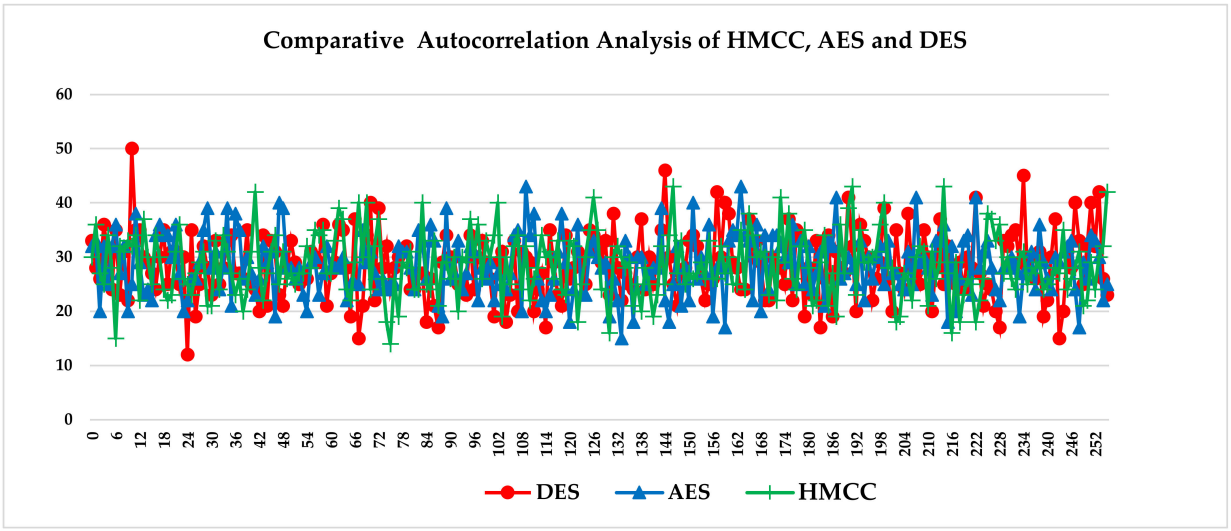
text. On the other hand, DES is less resistant to differential cryptanalysis attacks. This can be seen from the obtained values of NPCR and UACI, which are very far from the expected value of 100% and 33.3%, respectively.

iii. **Comparative analysis of MCC, AES, and DES based on Autocorrelation Analysis**

Comparative autocorrelation analyses of HMCC, AES, and DES were carried out by passing the same plaintext through the three ciphers. The autocorrelation analysis of the ciphertext produced from each cipher was carried out. The autocorrelation graphs shown in Figure 10 compare the autocorrelation of the input plaintext with the autocorrelation of the output ciphertexts from each of HMCC, AES, and DES ciphers.



(**a**)



(**b**)

Figure 10. (**a**): Autocorrelation of plaintext. (**b**): Comparative analysis of autocorrelation of ciphertexts produced by HMCC, AES, and DES.

From the graphs shown in Figure 10a, it can be seen that the autocorrelation of the plaintext has a non-uniform and large number of similar symbols, while the graphs of autocorrelation of ciphertext produced by HMCC, AES, and DES in Figure 10b have more uniform and reduced number of similar symbols when compared with that of the plaintext. It can also be observed from Figure 10b that the ciphertexts produced by HMCC and AES appear to show a slightly lower number of similar symbols than the ciphertext produced by DES. The graphs showed that HMCC and AES have similar symbols that are less than 50, while in some cases, the similar symbols in the ciphertext of DES reach 50. These results show that HMCC has similar behaviour to the standard AES algorithm in terms of autocorrelation analysis.

iv. **Comparative Analysis of HMCC, AES, and DES based on SKAC and SPAC**

SKAC and SPAC analyses were conducted for HMCC, AES, and DES. A summary of the results for the three ciphers for SKAC and SPAC results are respectively given in Tables 12 and 13, where the mean, standard deviation (STD), and coefficient of variance

(CV) of the SKAC and SPAC values are calculated. The coefficient of variance measures the amount of deviation of data points from the mean. It is calculated by dividing the standard deviation by the mean of the dataset. As the CV is the ratio of STD and means, it provides a means of comparing variations in different datasets even when the means are different.

Table 12. Comparative Analysis of SKAC for HMCC, AES, and DES.

CIPHER	MEAN	STD	CV
HMCC	0.498977	0.004795	0.009609
AES	0.499663	0.00397	0.007945
DES	0.438242	0.005367	0.012248

Table 13. Comparative Analysis of SPAC for HMCC, AES, and DES.

CIPHER	MEAN	STD	CV
MCC	0.493714	0.005331	0.010799
AES	0.500483	0.004301	0.008594
DES	0.499297	0.00602	0.012056

It can be seen in Tables 12 and 13 the AES has the lowest CV values, while DES has the highest values of CV. These results imply that AES provides better performance in terms of SKAC and SPAC than HMCC and DES. However, HMCC performs better than DES in terms of SKAC and SPAC. It can, therefore, be said that HMCC performance in terms of SKAC and SPAC is comparable to that of the standard AES.

5. Limitations and Areas of Future Research

The proposed IoMT in this paper promises to be efficient and secure. However, there is always rooms for improvement. The value that represents the interpretation of the measured value from the sensor is arbitrarily assigned. There should be a standardized method to achieve this feat. Security and storage aspects could be improved by employing a heterogenous and compact way of representing data, such as the technique proposed in [69]. Multiple IoT Environment (MIE) [70] and the Multiple Internets of Things (MIoT) [71–74] address data-driven and semantics-based aspects because they consider the contents exchanged by smart objects during their transactions; consequently, the extension of the concept in the proposed IoMT model to both MIE and MIoT will play a significant role. Future research should also consider the implementation of the proposed IoMT model in a real CSP platform.

6. Conclusions

In this contribution, a secure IoMT model that is resistant to various security attacks and preserves the privacy of patient data is proposed. In the design, an existing stream Caesar cipher was modified into a symmetric block cipher and an efficient technique that searches through encrypted files stored on the cloud without decrypting the file was developed. The Elliptic Curve Diffie–Hellman technique was employed to exchange shared secret keys to form the Hybrid Modified Caesar Cryptosystem (HMCC). A complete security analysis of the proposed HMCC was carried out based on python simulations. Generally, results show that the HMCC is resistant to brute-force attacks, sensitive to any change in secret key and plain text, can resist an entropy attack, generates a uniform histogram with low autocorrelation, is robust against differential, dictionary, and frequency analysis and classic attacks, and strict key avalanche criterion and strict plaintext avalanche criterion are satisfied within very minimal errors. Additionally, analysis of the IoMT-based model reveals its resistance to all forms of Man-in-the-Middle attacks. A comparative security analysis of the projected HMCC with existing ciphers also revealed that the HMCC security is comparable to that of standard AES and more secure than the DES, and the encryption time is very fast. HMCC also uses variable key length, which makes it

suitable for lightweight cryptography on resource-constrained devices. The searchable encryption-based technique employed is capable of direct access to information on the cloud irrespective of the size of data stored on the cloud, and also conserves space by avoiding data redundancy in the cloud in an attempt to use FHE. The scheme also reduces the complexity involved in computation in FHE implementation in [59]. It can, therefore, be said that the proposed IoMT-based model is provably secure and preserves the privacy of critical user data.

Author Contributions: The manuscript was written through the contributions of all authors. O.C.A., E.T.O. and J.B.A. were responsible for the conceptualization of the topic; article gathering and sorting were carried out by O.C.A., E.T.O. and J.B.A.; manuscript writing and original drafting and formal analysis were carried out by O.C.A., E.T.O., J.B.A., A.L.I., C.-C.L. and C.-T.L.; writing of reviews and editing were carried out by J.B.A., A.L.I., C.-C.L. and C.-T.L.; J.B.A. led the overall research activity. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Science and Technology Council, Taiwan, R.O.C., under contract no.: MOST 110-2410-H-165-001-MY2.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: The data that support the findings of this paper are available from the corresponding author upon reasonable request.

Acknowledgments: The work of Agbotiname Lucky Imoize is supported by the Nigerian Petroleum Technology Development Fund (PTDF) and the German Academic Exchange Service (DAAD) through the Nigerian–German Postgraduate Program under grant 57473408.

Conflicts of Interest: The authors declare no conflict of interest related to this work.

References

1. Dhillon, P.K.; Kalra, S. Secure and efficient ECC based SIP authentication scheme for VoIP communications in internet of things. *Multimed. Tools Appl.* **2019**, *78*, 22199–22222. [CrossRef]
2. Jagadeeswari, V.; Subramaniyaswamy, V.; Logesh, R.; Vijayakumar, V. A study on medical internet of things and big data in personalized healthcare system. *Health Inf. Sci. Syst.* **2018**, *6*, 14. [CrossRef] [PubMed]
3. Hoffman, D.; Novak, T. Consumer and object experience in the internet of things: An assemblage theory approach. *J. Consum. Resour.* **2018**, *44*, 1178–1204. [CrossRef]
4. Borgia, E. The internet of things vision: Key features, applications and open issues. *Comput. Commun.* **2014**, *54*, 1–31. [CrossRef]
5. Botta, A.; de Donato, W.; Persico, V.; Pescap, A. Integration of cloud computing and internet of things: A survey. *Future Gener. Comput. Syst.* **2016**, *56*, 684–700. [CrossRef]
6. Sharma, M.; Siddiqui, A. RFID based mobiles: Next generation Applications. In Proceedings of the 2nd IEEE International Conference on Information Management and Engineering (ICIME), Chengdu, China, 16–18 April 2010; pp. 523–526.
7. Matin, M.A.; Islam, M.M. Overview of Wireless Sensor Network. In *Wireless Sensor Networks–Technology and Protocols*; INTECH: Hong Kong, China, 2012; pp. 3–24.
8. Al-kahtani, M.S.; Khan, F.; Taekeun, W. Application of Internet of Things and Sensors in Healthcare. *Sensor* **2022**, *22*, 5738. [CrossRef]
9. Samaila, M.G.; Neto, M.; Fernandes, D.A.B.; Freire, M.M.; Inácio, P.R.M. Security Challenges of the Internet of Things. In *Beyond the Internet of Things, Internet of Things*; Batalla, J.M., Mastorakis, G., Mavromoustakis, C., Pallis, E., Eds.; Springer International Publishing AG: Berlin, Germany, 2017; pp. 53–82; ISBN 9783319507583.
10. Mosenia, A.; Jha, N.K. A comprehensive study of security of Internet-of-Things. *IEEE Trans. Emerg. Top. Comput.* **2017**, *5*, 586–602. [CrossRef]
11. Granjal, J.; Monteiro, E.; Silva, J.S. Security for the Internet of Things: A Survey of Existing Protocols and Open Research Issues. *IEEE Commun. Surv. Tutor.* **2015**, *17*, 1294–1312. [CrossRef]
12. Ogundokun, R.O.; Awotunde, J.B.; Adeniyi, E.A.; Ayo, F.E. Crypto-Stegno based model for securing medical information on IOMT platform. *Multimed. Tools Appl.* **2021**, *80*, 31705–31727. [CrossRef]
13. Darma, D.; Ilmi, Z.; Darma, S.; Syaharuddin, Y. COVID-19 and its impact on education: Challenges from industry 4.0. *AQUADEMIA* **2020**, *4*, ep20025.
14. Ilmi, Z.; Darma, D.C.; Azis, M. Independence in learning, education management, and industry 4.0: Habitat indonesia during COVID-19. *J. Anthr. Sport Phys. Educ.* **2020**, *4*, 63–66.
15. Kumar, K.; Kumar, N.; Shah, R. Role of IoT to avoid spreading of COVID-19. *Int. J. Intell. Netw.* **2020**, *1*, 32–35. [CrossRef]

16. Celesti, A.; Fazio, M.; Galan, F.; Arquez, M.; Glikson, A.; Mauwa, H.; Bagula, A. How to develop IoT 'cloud e-health systems based on fiware: A lesson learnt. *J. Sens. Actuator Netw.* **2019**, *8*, 7. [CrossRef]
17. Debdas, S.; Panigrahi, C.K.; Kundu, P.; Kundu, S.; Jha, R. IoT application in interconnected hospitals. *Mach. Learn. Healthc. Appl.* **2021**, 225–247. [CrossRef]
18. Swaroop, K.N.; Chandu, K.; Gorrepotu, R.; Deb, S. A health monitoring system for vital signs using IoT. *Internet Things* **2019**, *5*, 116–129. [CrossRef]
19. Zamanifar, A. Remote patient monitoring: Health status detection and prediction in IoT-based health care. In *iIoT in Healthcare and Ambient Assisted Living*; Spinger: Berlin/Heidelberg, Germany, 2021; pp. 89–102.
20. Rose, K.; Eldridge, S.; Chapin, L. The Internet of Things: An Overview-Understanding the Issues and Challenges of a More Connected World. *Internet Soc. Pages* **2015**, *80*, 1–50.
21. Gamundani, A.M. An Impact Review on Internet of Things Attacks. In Proceedings of the 2015 International Conference on Emerging Trends in Networks and Computer Communications (ETNCC), Windhoek, Namibia, 17–20 May 2015.
22. Rahman, A.; Rahman, M.; Kundu, D.; Karim, R.; Shahab, S. Study on IoT for SARS-CoV-2 with healthcare: Present and future perspective. *Math. Biosci. Eng.* **2021**, *18*, 9697–9726. [CrossRef]
23. Yin, M.; Chen, X.; Wang, Q.; Wang, W.; Wang, Y. Dynamics on hybrid complex network: Botnet modeling and analysis ofmedical IoT. *Secur. Commun. Netw.* **2019**, *2019*, 6803801. [CrossRef]
24. Din, I.U.; Guizani, M.; Kim, B.S.; Hassan, S.; Khan, M.K. Trust management techniques for the internet of things: A survey. *IEEE Access* **2018**, *7*, 29763–29787. [CrossRef]
25. Din, I.U.; Almogren, A.; Guizani, M.; Zuair, M. A decade of internet of things: Analysis in the light of healthcare applications. *IEEE Access* **2019**, *7*, 89967–89979. [CrossRef]
26. Slam, S.U.; Khattak, H.A.; Pierson, J.M.; Din, I.U.; Almogren, A.; Guizani, M.; Zuair, M. Leveraging utilization as performance metric for CDN enabled energy efficient internet of things. *Measurement* **2019**, *147*, 106814.
27. Khan, S.; Sikandar, M.; Almogren, A.; Din, I.U.; Fortino, G.; Guerrieri, A. IoMT-based computational approach for detecting brain tumor. *Future Gener. Comput. Syst.* **2020**, *109*, 360–367. [CrossRef]
28. Manasrah, A.M.; Shannaq, M.A.; Nasir, M.A. An Investigation Study of Privacy Preserving in Cloud Computing Environment. In *Handbook of Computer Networks and Cyber Security*; Springer Nature AG: Cham, Switzerland, 2020; ISBN 978-3-030-22276-5.
29. Nabeel, M.; Bertino, E. Privacy preserving delegated access control in public clouds. *IEEE Trans. Knowl. Data Eng.* **2013**, *26*, 2268–2280. [CrossRef]
30. Dong, X.; Yu, J.; Luo, Y.; Chen, Y.; Xue, G.; Li, M. Achieving an effective, scalable and privacy-preserving data sharingservice in cloud computing. *Comput. Secur.* **2014**, *42*, 151–164. [CrossRef]
31. Joseph, N.M.; Daniel, E.; Vasanthi, N. Survey on privacy-preserving methods for Computing storage in cloud computing. In Proceedings of the IAmrita International Conference of Women in Computing, Coimbatore, India, 9–11 January 2013.
32. Jogade, S.; Sharma, R.; Kadam, R. Partitioning data and domain integrity checking for storage-improving cloud storage security using data partitioning technique. *Int. J. Emerg. Res. Manag. Technol.* **2014**, *3*, 133–137.
33. Chen, F.; Liu, A.X. Privacy and integrity preserving multi dimensional range queries for cloud computing. In Proceedings of the Networking Conference, Trondheim, Norway 2–4 June 2014; IEEE: Piscataway, NJ, USA, 2014.
34. Ku, W.S.; Hu, L.; Shahabi, C.; Wang, H. A query integrity assurance scheme for accessing outsourced spatial databases. *Geoinformatica* **2013**, *17*, 97–124. [CrossRef]
35. Hu, L.; Ku, W.S.; Bakiras, S.; Shahabi, C. Spatial query integrity with Voronoi neighbors. *Knowl. Data Eng.* **2013**, *26*, 863–876. [CrossRef]
36. Naruchitparames, J.; Güneş, M.H. Enhancing data privacy and integrity in the cloud. In Proceedings of the International Conference on High Performance Computing and Simulation (HPCS), Istanbul, Turkey, 4–8 July 2011; IEEE: Piscataway, NJ, USA, 2011.
37. Gentry, C.; Halevi, S. Implementing Gentry's fully-homomorphic encryption scheme. In Proceedings of the Advances in Cryptology–EUROCRYPT 2011: 30th Annual International Conference on the Theory and Applications of Cryptographic Techniques, Tallinn, Estonia, 15–19 May 2011; pp. 129–148.
38. Gentry, C.; Halevi, S.; Smart, N. Better bootstrapping in fully homomorphic encryption. In Proceedings of the Public Key Cryptography–PKC 2012: 15th International Conference on Practice and Theory in Public Key Cryptography, Darmstadt, Germany, 21–23 May 2012; pp. 1–16.
39. Gentry, C.; Halevi, S.; Smart, N. Fully homomorphic encryption with polylog overhead. *Eurocrypt* **2012**, *7237*, 465–482.
40. Gentry, C.; Halevi, S.; Smart, N.P. Homomorphic evaluation of the AES circuit. In Proceedings of the Advances in Cryptology–CRYPTO 2012: 32nd Annual Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2012; pp. 850–867.
41. Chavan, P.; More, P.; Thorat, N.; Yewale, S.; Dhade, P. ECG-Remote patient monitoring using cloud computing. *Imp. J. Interdiscip. Res.* **2016**, *2*, 368–372.
42. Arbat, H.; Choudhary, S.; Bala, K. IoT smart health band. *Imp. J. Interdiscip. Res.* **2016**, *2*, 300–311.
43. Islam, S.R.; Kwak, D.; Kabir, M.H.; Hossain, M.; Kwak, K.S. The internet of things for health care: A comprehensive survey. *IEEE Access* **2015**, *3*, 678–708. [CrossRef]
44. Huiyeh, K.A. secure IoT-based healthcare system with body sensor networks. *IEEE Access* **2016**, *4*, 10288–10299.

45. Rayappan, D.; Pandiyan, M. Lightweight Feistel structure based hybrid-crypto model for multimedia data security over uncertain cloud environment. *Wirel. Netw.* **2021**, *27*, 981–999. [CrossRef]
46. Elhoseny, M.; Shankar, K.; Lakshmanaprabu, S.K.; Maseleno, A.; Arunkumar, N. Hybrid optimization with cryptography encryption for medical image security in Internet of Things. *Neural Comput. Appl.* **2020**, *32*, 10979–10993. [CrossRef]
47. Chhabra, S.; Lata, K. Obfuscated AES cryptosystem for secure medical imaging systems in IoMT edge devices. *Health Technol.* **2022**, *12*, 971–986. [CrossRef]
48. Song, D.X.; Wagner, D.; Perrig, A. Practical techniques for searches on encrypted data. In Proceedings of the Security and Privacy, 2000. S&P 2000, Berkeley, CA, USA, 14–17 May 2000; IEEE: Piscataway, NJ, USA, 2000.
49. Goh, E.-J. Secure Indexes for Efficient Searching on Encrypted Compressed Data 2003. Available online: https://eprint.iacr.org/2003/216.pdf (accessed on 3 January 2023).
50. Chang, Y.-C.; Mitzenmacher, M. Privacy preserving keyword searches on remote encrypted data. In *Applied Cryptography and Network Security*; Springer: Berlin/Heidelberg, Germany, 2005.
51. Li, J.; Wang, Q.; Wang, C.; Cao, N.; Ren, K.; Lou, W. Fuzzy keyword search over encrypte d data in cloud computing. In Proceedings of the INFOCOM, San Diego, CA, USA, 14–19 March 2010; IEEE: Piscataway, NJ, USA, 2010.
52. Adjedj, M.; Bringer, J.; Chabanne, H.; Kindarji, B. Biometric identification over encrypted data made feasible. In *Information Systems Security*; Springer: Berlin/Heidelberg, Germany, 2009; pp. 86–100.
53. Kuzu, M.; Islam, M.S.; Kantarcioglu, M. Efficient similarity search over encrypted data. In Proceedings of the 28th International Conference on Data Engineering (ICDE), Arlington, VA, USA, 1–5 April 2012; IEEE: Piscataway, NJ, USA, 2012.
54. Lu, Y.; Privacy-preserving logarithmic-time search on encrypted data in cloud. NDSS 2012. Available online: https://www.ndss-symposium.org/wp-content/uploads/2017/09/04_1.pdf (accessed on 3 January 2023).
55. Wang, J.; Ma, H.; Tang, Q.; Li, J.; Zhu, H.; Ma, S.; Chen, X. A new efficient verifiable fuzzy keyword search scheme. *JoWUA* **2012**, *3*, 61–71.
56. Boneh, D.; Di Crescenzo, G.; Ostrovsky, R.; Persiano, G. Public key encryption with keyword search. In Proceedings of the Advances in Cryptology-EUROCRYPT 2004: International Conference on the Theory and Applications of Cryptographic Techniques, Interlaken, Switzerland, 2–6 May 2004; Springer: Berlin/Heidelberg, Germany, 2004.
57. Bellare, M.; Boldyreva, A.; O'Neill, A. Deterministic and efficiently searchable encryption. In Proceedings of the Advances in Cryptology-CRYPTO 2007: 27th Annual International Cryptology Conference, Santa Barbara, CA, USA, 19–23 August 2007; Springer: Berlin/Heidelberg, Germany, 2007; pp. 535–552.
58. Subramaniyaswamy, V.; Jagadeeswari, V.; Indragandhi, V.; Jhaveri, R.H.; Vijayakumar, V.; Kotecha, K.; Ravi, L. Somewhat Homomorphic Encryption: Ring Learning with Error Algorithm for Faster Encryption of IoT Sensor Signal-Based Edge Devices. *Secur. Commun. Netw.* **2022**, *2022*, 2793998. [CrossRef]
59. Kocabaş, Ö.; Soyata, T. Towards Privacy-Preserving Medical Homomorphic Encryption. In *Virtual and Mobile Healthcare: Breakthroughs in Research and Practice*; IGI Global: Hershey, PA, USA, 2015.
60. Fahrnberger, G. Editing Encrypted Messages without Decrypting or Understanding Them. Ph.D Thesis, University of Hagen, Hagen, Germany, 2019.
61. Goyal, T.K.; Sahula, V. Lightweight Security Algorithm for Low Power IoT Devices. In Proceedings of the Intl. Conference on Advances in Computing, Communications and Informatics (ICACCI), Jaipur, India, 21–24 September 2016; pp. 1725–1729.
62. Sulak, F. Statistical Analysis of Block Cipher and Hash Function. *Middle East Tech. Univ.* **2011**.
63. Gupta, S.; Verma, H.K.; Sangal, A.L. Security attacks & prerequisite for wireless sensor networks. *Int. J. Eng. Adv. Technol.* **2013**, *2*, 558–566.
64. Patidar, V.; Pareek, N.; Sud, K. A New Substitution-Diffusion Based Image Cipher Using Chaotic Standard and Logistic Maps. *Commun. Nonlinear Sci. Number Simul.* **2009**, *14*, 3056–3075. [CrossRef]
65. Liu, H.; Zhao, B.; Huang, L. Quantum Image Encryption Scheme Using Arnold Transform and S-box Scrambling. *Entropy* **2019**, *21*, 343. [CrossRef] [PubMed]
66. Álvarez, G.; Li, S. Some basic cryptographic requirements for chaos-based cryptosystems. *Int. J. Bifurc. Chaos* **2006**, 2129–2151. [CrossRef]
67. Murillo-Escobar, M.A.; Meranza-Castillón, M.O.; López-Gutiérrez, R.M.; Cruz-Hernández, C. Suggested Integral Analysis for Chaos-Based Image Cryptosystems. *Entropy* **2019**, *21*, 815. [CrossRef] [PubMed]
68. Chen, T.; Wu, C. Compression-unimpared batch-image Encryption Combining Vector Quantization and Index Compression. *Inf. Sci.* **2010**, *18*, 1690–1701. [CrossRef]
69. Matter, W.; Cauteruccio, F.; Stamile, C.; Ursino, D.; Sappey, D.; Cauteruccio, F.; Stamile, C.; Terracina, G.; Ursino, D.; Sappey-marinier, D. An automated string-based approach to extracting and characterizing White Matter fiber-bundle. *Comput. Biol. Med.* **2016**, *77*, 64–75. [CrossRef]
70. Baldassarre, G.; Giudice, P.L.; Musarella, L.; Ursino, D. A paradigm for the cooperation of objects belonging to different IoTs. In Proceedings of the International Database Engineering & Applications Symposium, IDEAS 2018, Villa San Giovanni, Italy, 18–20 June 2018; pp. 157–164.
71. Ursino, D.; Virgili, L. Humanizing IoT: Defining the profile and the reliability of a thing in a Multi-IoT scenario. In *iTowards Social Internet of Things: Enabling Technologies, Architectures and Applications*; Springer Nature: Berlin/Heidelberg, Germany, 2020.

72. Baldassarre, G.; Giudice, P.L.; Musarella, L.; Ursino, D. The MIoT paradigm: Main features and an '"ad-hoc"' crawler. *Future Gener. Comput. Syst.* **2019**, *92*, 29–42. [CrossRef]
73. Giudice, P.L.; Nocera, A.; Ursino, D.L. Virgili Building topic-driven virtual IoTs in a multiple IoTs scenario. *Sensors* **2019**, *19*, 2956. [CrossRef]
74. Cauteruccio, F.; Cinelli, L.; Fortino, G.; Savaglio, C.; Terracina, G.; Ursino, D.; Virgili, L. An approach to compute the scope of a social object in a Multi-IoT scenario. *Pervasive Mob. Comput.* **2020**, *67*, 101223. [CrossRef]