Secured Remote Healthcare System Using Lightweight Model

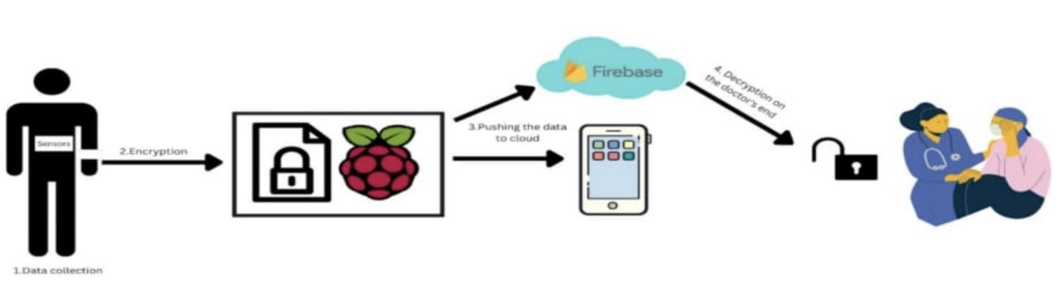
B Vignesh, Vishal A S, Devakumar V, V Nithin Krishna, A Baskar, Dr. Senthilkumar T, Kartik Srinivasan (Sr. Architect, IBM)

(A) Goal & Contributions

The necessity for primary healthcare visits to hospitals can be eliminated by integrating IoT and health wearables. Additionally, patients' medical costs are much lower due to this. Additionally, by tracking a patient's health statistics over time via an application, doctors can prescribe appropriate drugs. A thorough study of the data was collected concerning physical and environmental activity fluctuations to comprehend how the sensors operate. In the past, patient data was stored on paper and secured in file cabinets, making it relatively simple to safeguard and keep. However, today's patient records are electronically saved on computers, servers, and storage devices because of technological advancements and the advent of the digital era. Increased dangers of data breaches, malware, viruses, and other hostile assaults come along with electronic documents. Various algorithms such as Blowfish, AES, Acorn, and Homomorphic encryption have been implemented in the model to ensure maximum security and prevent compromising the data

Our goals are as follows :
1. Compare various lightweight encryption algorithms and find the best suited algorithm for the problem statement
2. After finding the best suited lightweight algorithm try to optimize it as much as possible to get the best results.

(B) Architecture Diagram



Here's a brief description of how our model typically works:

• The patient wears the wearable sensors on the body, and the sensors, when activated collect the data of the patient such as the Heartbeat, ECG, GSR, Airflow, and Body Temperature Values.

• The data that has been collected is then passed through multiple encryption techniques to protect the integrity of data and prevent compromising the sensitive Information.

• The encrypted data is then pushed to the cloud for storage purposes.

• The data is then decrypted on the doctor's end so that they view the patient's records.

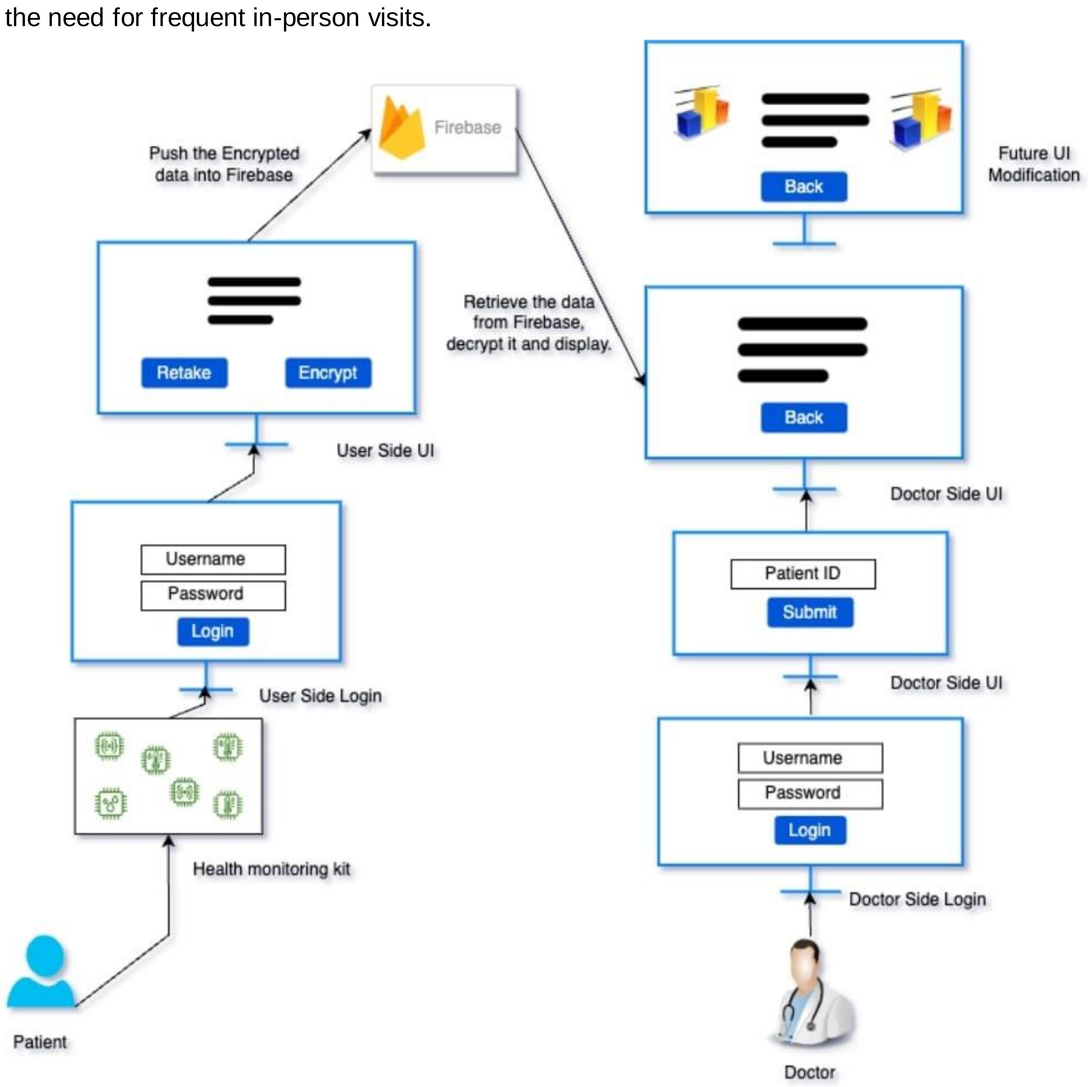
(C) Dataset

	ID	Temp	GSR	ECG	Heart_Beat	Air_flow
0	23176	32	338	99	96	3
1	45090	37	343	88	79	5
2	64779	36	347	86	90	4
3	55938	34	338	78	96	5
4	26006	39	325	93	98	2

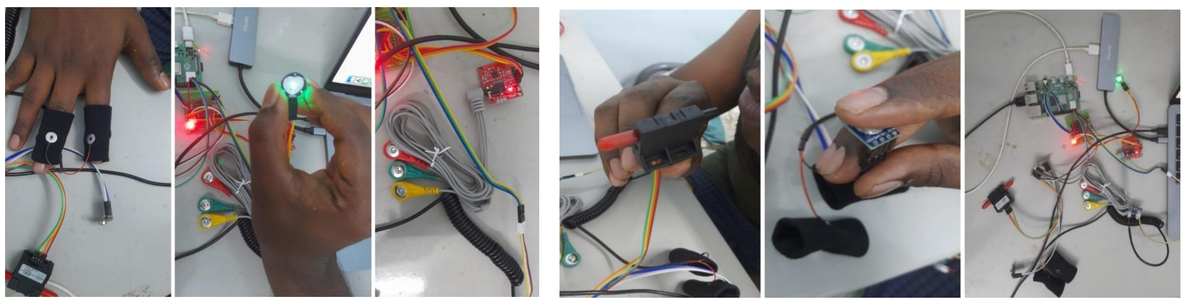
We have collected a real-time dataset with the help of the sensors. The various sensors used in this scenario are the ECG, Airflow Sensors, Heartbeat, Temperature, and GSR sensors. With the help of the UUID python library, a random patient ID is generated every time we try to gain insights into the patient. We have gathered real-time data from the sensors and have uploaded it to several cloud services, including Thingspeak and Firebase. Using a GSR sensor, a temperature sensor, a heartbeat sensor, an ECG sensor, and an airflow sensor, we collected data from the patients. We have used the UUID library to build the patient ID

(D) Proposed System & Design

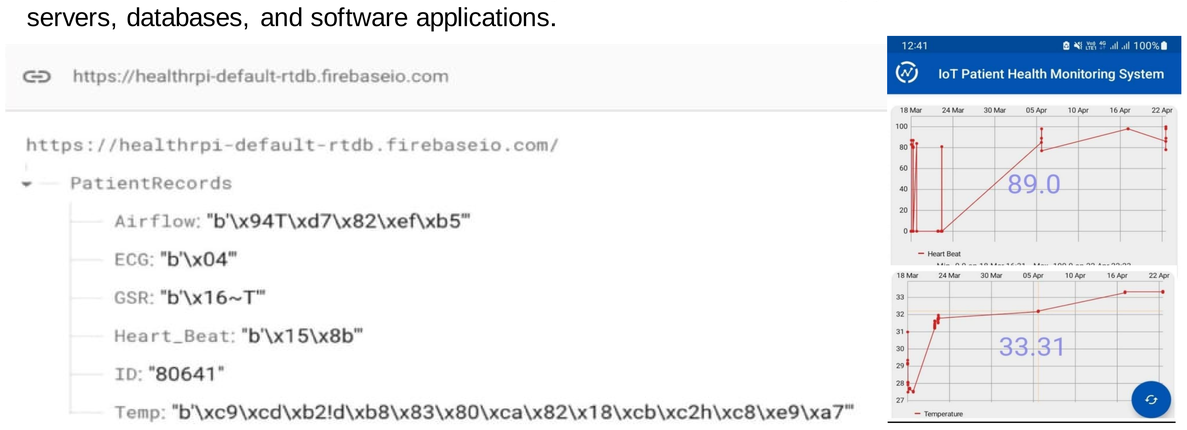
The data is collected from the patients with the help of various sensors. After collecting the data, it is passed through multiple encryption algorithms to encrypt the data to prevent any attacks and to protect the integrity of the data. After encrypting the data, it is pushed to Firebase and stored in the cloud. A user interface has been developed to retrieve the decrypted data and display it on the patient/doctor's side. They, in turn, would use their credentials to view their records stored. Overall, a remote healthcare monitoring system enables continuous monitoring and management of patient's health outside of traditional healthcare settings. It facilitates early detection of health issues, and personalized care interventions, and improves patient outcomes while reducing the need for frequent in-person visits to healthcare facilities. The remote healthcare monitoring system's ultimate goal is to improve patient outcomes, enhance access to care, and optimize healthcare resource utilization by enabling continuous monitoring and timely intervention without the need for frequent in-person visits.



(F) Implementation & Observation



We have successfully Implemented robust security measures to protect the confidentiality, integrity, and availability of the collected health data. Apply encryption techniques to data at rest and in transit. Implement access controls and authentication mechanisms to ensure that only authorized individuals can access the data. With the help of the Firebase library package, we were successfully able to push the encrypted data to the cloud and avoid any attacks that would compromise the data. We have Built or deployed the central monitoring system, which consists of servers, databases, and software applications.



(G) Results & Analysis

With the help of the Firebase library package, we were successfully able to push the encrypted data to the cloud and avoid any attacks that would compromise the data. We have Built or deployed the central monitoring system, which consists of servers, databases, and software applications.

Algorithm	Technique	Keytype	Size of Data	Avg CPU Usage	Time Taken
AES	Encryption	128	951	6.8	0.6
AES	Decryption	128	951	5.5	0.5
AES	Encryption	128	1	4.7	0.1
AES	Decryption	128	1	4.7	0.1
AES	Encryption	256	951	16.7	0.7
AES	Decryption	256	951	15.9	0.5
AES	Encryption	256	1	6.2	0.05
AES	Decryption	256	1	7.1	0.02

Algorithm	Technique	Keytype	Size of Data	Avg CPU Usage	Time Taken
Homomorphic	Encryption	Short	951	13.4	1.4
Homomorphic	Decryption	Short	951	6.9	0.6
Homomorphic	Encryption	Short	1	7.1	0.03
Homomorphic	Decryption	Short	1	9.5	0.01
Homomorphic	Encryption	Long	951	48.8	37.7
Homomorphic	Decryption	Long	951	54.7	12.4
Homomorphic	Encryption	Long	1	53.0	0.04
Homomorphic	Decryption	Long	1	16.6	0.01

The above table depicts Avg. CPU usage and the time taken (in s) for encrypting/decrypting the dataset using AES Algorithm with short and long key lengths for the dataset. While AES is widely regarded as a robust and secure encryption algorithm, it does have certain limitations. Here are some notable limitations of AES and reasons why we have chosen to move forward with homomorphic and Acorn encryption techniques:

• Key Management: AES requires proper key management to maintain its security. The strength of AES encryption depends on the secrecy and randomness of the encryption key. If the key is weak or compromised, the encrypted data's security is also compromised.

• Lack of Homomorphic Properties: AES is a symmetric encryption algorithm requiring decryption to perform data computations. It does not possess homomorphic properties.

• Limited Scalability: AES operates on fixed-size blocks (128 bits) of data.

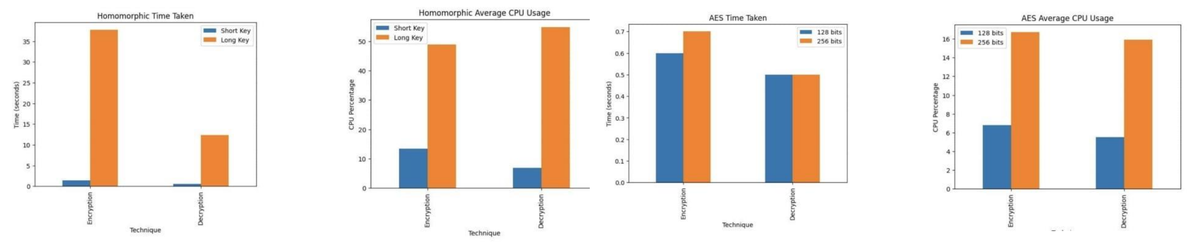
• Side-Channel Attacks: AES implementations can be vulnerable to side-channel attacks. These attacks exploit information leaked during encryption.

Algorithm	Encryption Time	Decryption Time
Homomorphic	26.8s	3.75s
Acorn	5.14s	7.03s
Acorn V2	0.09s	0.1s

Algorithm	CPU Usage Encryption Time	CPU Usage Decryption
Homomorphic	12.8	13.1
Acorn	18.2	20.4
Acorn V2	25.2	25.8

Analysis of Homomorphic vs Acorn Records in Time Taken

Analysis of Homomorphic vs Acorn Records in CPU Usage



From the above tables we can come to a conclusion that though Acorn algorithm is less time to encrypt and decrypt the patient's data is important to note that the Average CPU Usage is significantly higher than homomorphic encryption. It's important to note that the choice of cryptographic algorithms depends on the specific use case, security requirements, and constraints of the system. Both homomorphic encryption and Acorn algorithms serve different purposes and have their own strengths and limitations.
Therefore, the selection should be based on the specific needs and trade-offs of the application at hand

References

• Nan, Guofang et al. Distributed resource allocation in cloud-based wireless multimedia social networks. IEEE Netw. 280.–:74(2014) 28.4 2.
• Abiodun MK, Awotunde JB, Ogundokun RO, Misra S, Adeniyi EA, Arowolo MO, Jaglan V. Cloud and big data: a mutual benefit for organisation development.
J Phys Conf Ser. 2021;1767(1):012020 (IOP Publishing).
• Yu R, Zhang Y, Gjessing S, Xia W, Yang K. Toward cloud based vehicular networks with efficient resource management. IEEE Netw Mag. 2013;27(5):48–55.
• Adebiyi MO, Arowolo MO, Olugbara O. A genetic algorithm for prediction of RNA-seq malaria vector gene expression data classification using SVM kernels.
Bull Elect Eng Informat. 2021;10(2):1071–9.
• Suryandari Y. Survei IoT healthcare device. J Sistem Cerdas. 2020;3:153–64.
• Mutlag A, et al. Enabling technologies for fog computing in healthcare IoT systems. Fut Generat Comput Syst. 2019;90:62–78.
• Chrystinne OF, Carlos de JPL. An internet of things application with an accessible interface for remote monitoring patients. Springer International Publishing
Switzerland; 2015.