# Handbook of Research on Cloud Infrastructures for Big Data Analytics

Pethuru Raj
*IBM India Pvt Ltd, India*

Ganesh Chandra Deka
*Ministry of Labour and Employment, India*

A volume in the Advances in Data Mining and Database Management (ADMDM) Book Series

Information Science REFERENCE
An Imprint of IGI Global

Managing Director: Lindsay Johnston
Production Editor: Jennifer Yoder
Development Editor: Austin DeMarco
Acquisitions Editor: Kayla Wolfe
Typesetter: Michael Brehm
Cover Design: Jason Mull

Published in the United States of America by
Information Science Reference (an imprint of IGI Global)
701 E. Chocolate Avenue
Hershey PA 17033
Tel: 717-533-8845
Fax: 717-533-8661
E-mail: cust@igi-global.com
Web site: http://www.igi-global.com



Library of Congress Cataloging-in-Publication Data

CIP Data (Pending)

978-1-4666-5864-6 (ISBN Hardcover)
978-1-4666-5865-3 (ISBN eBook)
978-1-4666-5867-7 (ISBN Print and Perpetual)

This book is published in the IGI Global book series Advances in Data Mining and Database Management (ADMDM)
(ISSN: 2327-1981; eISSN: 2327-199X)

British Cataloguing in Publication Data
A Cataloguing in Publication record for this book is available from the British Library.

# Chapter 19
# Medical Data Analytics in the Cloud Using Homomorphic Encryption

**Övünç Kocabaş**
*University of Rochester, USA*

**Tolga Soyata**
*University of Rochester, USA*

## Abstract

*Transitioning US healthcare into the digital era is necessary to reduce operational costs at Healthcare Organizations (HCO) and provide better diagnostic tools for healthcare professionals by making digital patient data available in a timely fashion. Such a transition requires that the Personal Health Information (PHI) is protected in three different phases of the manipulation of digital patient data: 1) Acquisition, 2) Storage, and 3) Computation. While being able to perform analytics or using such PHI for long-term health monitoring can have significant positive impacts on the quality of healthcare, securing PHI in each one of these phases presents unique challenges in each phase. While established encryption techniques, such as Advanced Encryption Standard (AES), can secure PHI in Phases 1 (acquisition) and 2 (storage), they can only assure secure storage. Assuring the data privacy in Phase 3 (computation) is much more challenging, since there exists no method to perform computations, such as analytics and long-term health monitoring, on encrypted data efficiently. In this chapter, the authors study one emerging encryption technique, called Fully Homomorphic Encryption (FHE), as a candidate to perform secure analytics and monitoring on PHI in Phase 3. While FHE is in its developing stages and a mainstream application of it to general healthcare applications may take years to be established, the authors conduct a feasibility study of its application to long-term patient monitoring via cloud-based ECG data acquisition through existing ECG acquisition devices.*

DOI: 10.4018/978-1-4666-5864-6.ch019

## INTRODUCTION

Utilizing cloud computing resources such as Amazon EC2 (Amazon, n.d.), Microsoft Azure (Microsoft, n.d.), or Google (Google, n.d.) is commonplace for many corporations, due to its ability to prevent vast infrastructure investments. This concept dates back to the beginning of the Internet boom more than a decade ago with the emergence of the *Application Service Provider (ASP)* model: Rather than making an investment in costly server hardware, software licensing fees, and the personnel to manage this infrastructure, corporations can *rent* computation time, storage space, and licensing fees by running such applications as Salesforce.com (Salesforce, n.d.) over the Internet. The ASP model prevents upfront costs: a monthly subscription fee and a flexible licensing scheme allows smaller corporations to immediately start using such programs and expand with virtually no boundaries, since the computational and storage resources are provided by the application service provider (ASP) and the ASP can pool resources for many other clients. Additionally, this eliminates the need for corporations to have any expertise in setting up such sophisticated server infrastructure and the training on the application is done through online seminars.

Another dramatic example of such an ASP model is Paypal (Paypal, n.d.). The introduction of a merchant Application Programming Interface (API) by Paypal allowed any size corporation to start their business with near-zero investment, accept payments over the Internet by using Paypal as the intermediary, and grow with virtually no boundary. These examples show that, it is natural to shift the responsibility of computing (and storage) infrastructure investments to operators that can deliver their services by using the Internet as the delivery channel (i.e., Cloud Operators). By virtualizing their computational and storage resources, these cloud operators can provide these resources to their customers at a fraction of what the customers can build them for.

While endless examples exist for such generic cloud computing offerings, one area that can benefit significantly from it deserves specific attention: Medical cloud computing. When the data storage is outsourced to a cloud operator over the Internet, an important issue arises: data privacy. Although different applications have different sensitivity levels to this issue, the highest level of sensitivity is clearly in the medical arena (Kocabas et al, 2013). Personal Health Information (PHI) is one of the most scrutinized concepts, protected by laws and regulations of the U.S.A. The Health Insurance Portability and Accountability Act (HIPAA, n.d.) dictates a strict set of rules and regulations to prevent the PHI from being misused. Therefore, to expand the cloud computing into the medical arena, one must clearly formulate the entire concept around these restrictions.

Cloud computing is an active research area for medical applications, partly due to the push by the US government to modernize the US Health system (Lobodzinski & Laks, 2012). The motivations behind this move are: 1) improving the quality of healthcare by using additional cloud-based long-term patient monitoring data that are otherwise unavailable to the healthcare professionals, and 2) reducing the operational costs at healthcare organizations (HCO) by eliminating the datacenters operated by HCOs. Long-term patient monitoring data (e.g., patient vitals such as ECG and blood pressure), obtained by sensors that transmit their patient information over the cloud can be used as an auxiliary diagnostic tool to improve diagnostic accuracy. This expands the boundaries of an HCO to outside the HCO by allowing the patients to use long-term monitoring devices, such as ECG patches.

In this chapter, we study the feasibility of such a cloud-based long-term monitoring system while preserving PHI. Preserving PHI requires ensuring data privacy at three distinct phases: Phase I. Acquisition, is where the medical data is acquired from a patient, whether it is within the HCO, or outside the HCO via disposable devices such as

ECG patches (Leaf, n.d.), Phase II. Storage, where the data is stored in the cloud for future access, and, Phase III. Computation, is where the data is processed, whether during a real-time application execution by a doctor, or by the long-term patient monitoring software.

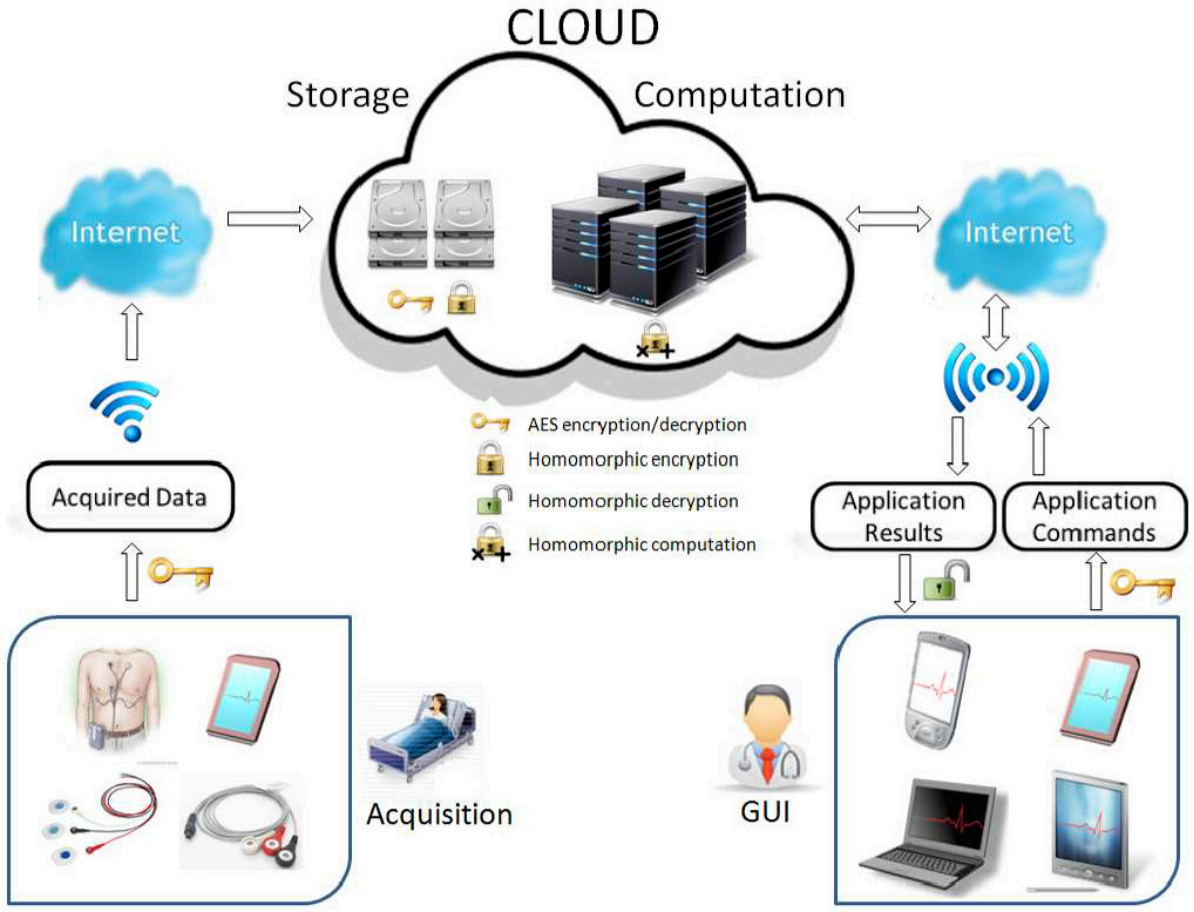
Existing AES-based encryption techniques (NIST, 2001) can ensure data privacy in phases I and II. However, ensuring data privacy during the application execution (i.e., Phase III) is only possible by transferring the data back and forth between the cloud and the mobile device. During this transfer, data must be in encrypted format while in the cloud, and must be decrypted when it reaches the mobile device. In contrast to this conventional methodology, we investigate an emerging new technique called Fully Homomorphic Encryption (FHE) (Gentry, 2009; Brakerski, Gentry, & Vaikuntanathan, 2012) and the possibility of its utilization in medical data analytics. We specifically investigate the application of remote health monitoring by using existing commodity ECG patches (Leaf, n.d.) and cloud computing. In our conceptual system, the entire application runs in the cloud, and the data acquisition (Phase I) and the visualization of analytics (Phase III) are achieved by thin devices (i.e., devices with significantly lower computational and storage capability as compared to the cloud resources). Therefore, these end nodes are disposable and the entire functionality of the application execution is outsourced to the cloud.

Our conceptual system, shown in Figure 1, depicts phase I (Acquisition) of the long-term health monitoring through the use of remote sensors, incorporating AES encryption and transmission capability. While we specifically focus on the ECG-based applications in this chapter, expansion of it to other medical applications is straightforward. The System in Figure 1 can be applied to any system containing sensors that have similar capabilities with a backend application that has similar characteristics. Phase II (storage) and III (computation) are strictly in the cloud in this system.

This system is conceptualized to use the end nodes as thin devices, where the loss of a thin device does not necessarily imply compromised PHI, since the device contains almost no information. This is due to the real-time transmission of the PHI right after its acquisition. Since no data are kept in the acquisition devices in the long term, the privacy management responsibility of the data is only relevant in the cloud. A similar argument is true for the display devices (e.g., tablets). Since Phase III is primarily performed in the cloud, and no data is stored in the GUI device, the loss of a GUI device (see Figure 1) presents no privacy issues. The system in Figure 1 pushes the entire workload into the cloud, making the end nodes mere acquisition and display devices. The compromise of acquisition and GUI devices implying the potential compromise of PHI has become an important consideration by the FDA recently (FDA, 2013) and shows the importance of designing a system that doesn't depend on strict security standards on the end nodes to ensure overall system security.

In this chapter, we investigate the feasibility of running medical applications in the cloud by formulating Full Homomorphic Encryption (FHE) as the core of this idea. We identify the challenges in making this possible for the specific remote-ECG monitoring applications, without loss of generality. We provide pointers to the potential of FHE acceleration while it is being widely researched (PROCEED, n.d.) to arrive at conclusions for its practical use in more widespread medical applications. This chapter is organized as follows: We provide background information on Fully Homomorphic Encryption (FHE) and Electrocardiogram (ECG), followed by the introduction of a cloud-based medical application in detail. The challenges related to different parts of this application are determined and the results based on existing ECG-based patient data derived from the THEW database (Couderc, 2010) are presented. We conclude our chapter with discussions on future research challenges.

*Figure 1. Proposed cloud-based long term health monitoring system*




## BACKGROUND INFORMATION

We will use Electrocardiogram (ECG) data to gain an insight into the challenges in applying FHE into medical applications. In this section, first we will provide background information on Fully Homomorphic Encryption (FHE) and focus on two important FHE schemes. Next, we will provide background information on ECG by using sample data acquired from the THEW worldwide ECG database (Courderc, 2010) and identify operations that are necessary to provide insight for a doctor during the diagnosis of cardiovascular diseases.

### Emergence of Fully Homomorphic Encryption (FHE)

Conventional symmetric-key and public-key cryptosystems encrypt the data such that only authorized parties can access the data. In order to perform operations on the data, one needs to decrypt the encrypted data first and then perform the operations. On the other hand, FHE schemes enable computing meaningful operations on the encrypted data without observing the actual data. In other words, an example computation, $c = a + b$, becomes possible using FHE without actually knowing $a$ and $b$.

To compute arbitrary functions on encrypted data, an FHE scheme should be capable of performing homomorphic additions and homomorphic multiplications over the encrypted text (termed *ciphertext*), which corresponds to addition and multiplication operations on the unencrypted message (termed *plaintext*) respectively when the resulting ciphertext is decrypted. Since any function can be represented as a combination of additions and multiplications, FHE scheme can compute arbitrary functions.

The FHE scheme is very useful in scenarios, where computation is outsourced to a third party and privacy of the data must be preserved at all times. With this scheme one can encrypt the data and store it in a database/cloud, and later ask a third party to perform some operations on the encrypted data. The third party never sees the original data but performs operations on the ciphertexts only, returning the result in encrypted form, which can only be decrypted by the secret key owner.

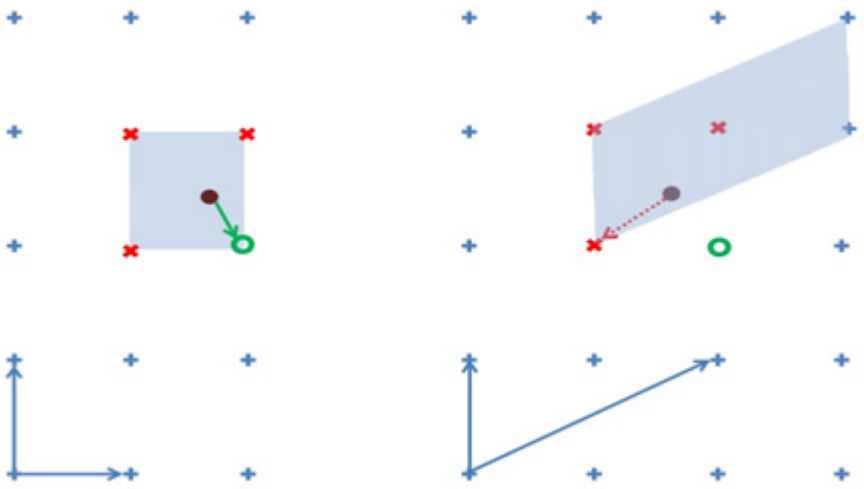
The idea of the homomorphic encryption was first proposed by Rivest et al. in 1978 (Rivest, Adleman, & Dertouzos, 1978). Since then, many schemes have been proposed (Goldwasser & Micali, 1982; El Gamal, 1985; Cohen & Fischer, 1985; Paillier, 1999; Damgård & Jurik, 2001), but these schemes support the only homomorphic addition or homomorphic multiplication, not both simultaneously within a single scheme. The closest cryptosystem to achieve the FHE scheme was proposed in (Boneh, Goh, & Nissim, 2005), which could perform many additions but only one multiplication. With his breakthrough work in 2009, Gentry (2009) proposed the first mechanism for an FHE scheme which could perform an arbitrary number of additions and multiplications homomorphically.

## Gentry's FHE Scheme

Gentry's (2009) proposal for the first FHE scheme is based on ideal lattices. An ideal lattice is a discrete additive and a multiplicative subgroup in n-dimensional space which can be represented by its basis vector. The fact that a lattice can have an infinite number of bases plays a key role for creating a public-key cryptosystem. Similar to other public key cryptosytems (Diffie & Hellman, 1976; Rivest, Shamir, & Adleman, 1978), security of the lattice based cryptosystems is based on an intractable problem which is very hard to solve unless a secret key is known. The hard problem in Gentry (2009) is the Closest Vector Problem (CVP) which states that given a point in n-dimensional space, it is hard to find the closest lattice point. If a good basis is known for the lattice, one can use Babai's nearest-vector approximation algorithm (Babai, L., 1985) to solve the CVP problem efficiently. The good basis of a lattice consists of almost orthogonal base vectors having a large decryption radius and it is used as the secret key. Figure *2* demonstrates the difference of decrypting a ciphertext with a good (on the left) and a bad (on the right) basis vector, where the result is mapped to an incorrect point on the lattice when a bad basis vector is used.

In Gentry's FHE scheme, encryption is performed by first mapping a message to a lattice point and then adding a small random noise to create the final ciphertext. The decryption can be done only by using a good basis which is only known by the secret-key holder. Homomorphic addition and homomorphic multiplication operations are performed by adding and multiplying lattice points respectively. During the homomorphic operations the noise inside the ciphertext grows with each operation. Specifically, homomorphic addition roughly doubles the noise, while homomorphic multiplication squares the noise. After several operations, the magnitude of the noise in the ciphertext exceeds the threshold at which a successful decryption is no longer possible even with the knowledge of a good basis. This limits the number of operations that can be performed with this scheme and is also referred to as *SomeWhat Homomorphic Encryption (SWHE)* scheme. Gentry proposed a remarkable bootstrapping method

*Figure 2. Homomorphic encryption with good (left) and bad (right) basis vectors mapping to a correct and incorrect result, respectively*



(i.e., recryption) to transform SWHE scheme into FHE scheme by evaluating the decryption function homomorphically. The recryption operation resets the noise inside the ciphertext and enables computation of arbitrary functions indefinitely.

Although Gentry's scheme is the first plausible mechanism for an FHE scheme, it has several inefficiencies both in terms of storage and computation. Messages are encrypted bitwise and in order to increase the noise threshold the ciphertext size must be large, which results expansion in storage space: For example, the size of a ciphertext encrypting 1-bit message could be multi-million bits, which presents an unacceptable data expansion ratio for most practical implementations. The homomorphic operations over very large ciphertexts are also compute-intensive and cost of the recryption operation is very high making Gentry's FHE scheme impractical.

Several FHE schemes and implementations have been proposed after Gentry's FHE scheme (Dijk, Gentry, Halevi, & Vaikuntanathan, 2010; Brakerski & Vaikuntanathan, 2011b, 2011a; Coron, Mandal, Naccache, & Tibouchi, 2011; Gentry & Halevi, 2011a; Naehrig, Lauter, & Vaikuntanathan, 2011; Smart & Vercauteren, 2010; Stehle & Steinfeld, 2010; Brakerski et al., 2012; Halevi & Shoup, n.d.; Gentry, Halevi, & Smart, 2012) to address the inefficiencies and make FHE more practical. (see Figure 2)

## BGV Scheme

At present the BGV scheme (Brakerski et al, 2012) and its implementation (Halevi & Shoup, n.d.) are one of the most promising works for a practical FHE. The BGV scheme is based on Ring Learning with Errors (RLWE) primitives (Lyubashevsky, Peikert, & Regev, 2010). In the BGV scheme both messages and ciphertexts are defined over polynomial rings.

Several methods are introduced by the BGV scheme to improve the performance of earlier FHE schemes. A ciphertext is partitioned into slots by using the techniques in (Smart & Vercauteren, 2011), where each slot can pack a multi-bit message. Packing multiple messages into one ciphertext also enables computing homomorphic operations in Single Instruction Multiple Data (SIMD) fashion. The expensive recrypt operation can be

avoided by using the leveled version of the BGV scheme. In the leveled version of the BGV scheme, homomorphic operations are performed up to $L$ levels. Since each homomorphic addition and multiplication increases the noise in the ciphertext, only a limited number of homomorphic operations can be performed. While homomorphic addition does not increase the noise level significantly, homomorphic multiplication roughly squares the noise amount. Thus the level $L$ is determined by the depth of multiplication operations for the function to be evaluated. The level of the function to be computed can be defined beforehand and then the parameters of the scheme can be adjusted during the key generation.

## Medical Data Analytics on ECG Data

An exhaustive list of medical data such as echo/MRI imaging data, subject drug treatment, and physiological monitoring signals are routinely acquired and used for assessing a patients' health state by healthcare organizations (HCO). Among the list of medical data, we have opted to limit our feasibility assessment to a simple, yet real, set of data acquired from a subject coming to the Emergency Department (ED) of the University of California San Francisco Hospital for chest pain (Shusterman et al, 2007) and shared by the THEW initiative (Courderc, 2010). This data contain recordings of a patient's heart rhythms for 24-hours, acquired by a 12-lead Holter system. The device was hooked up to the patients when they arrived at the ED. In order to demonstrate the feasibility of our concept, we used information about the patient's heart rate (HR). There are standard ECG measurements that a cardiologist needs to access from this information that require computational tasks. Among these, we selected five measurements to be extracted from ECG tracing as examples, these are: 1) the minimum HR, 2) the maximum HR, 3) the average heart rate, 4) the presence of abnormal cardiac beats, and 5) the frequency of the ectopic beats. These five quantifiers can be extracted from the annotation file of the ECG, i.e., the file containing the vital information about each cardiac beat type and duration as shown on Figure 3. These five ECG measurements provide essential analytic information to the cardiologist about the patient's heart state. First, the cardiologist will evaluate if the average heart rate is in normal ranges, and then the cardiologist will check if the heart rate variation during the recordings are appropriate based on the patient physical activity, finally the frequency of abnormal cardiac beats will be checked. These abnormal beats can be discriminated based on their morphology. They are often present in healthy individuals but they may be associated with some risk if their frequency of occurrence is too high.

## Structure of the Captured ECG Data

In general, the electrocardiogram (ECG) annotation file provides information which includes what type of cardiac contraction for each beat and the temporal distance between consecutive beats. The temporal distance is usually measured between two consecutive R peaks which is the peak of positive deflection in the QRS complex. Figure 3 shows the information extracted from a real ECG signal.

In this work, we have planned to assess the feasibility of implementing secure cloud-based monitoring using the ECG annotation file. The annotation file is a binary file containing two parts: 1) the header information and 2) the beat annotation. The header provides the information related to the original ECG, such as the number of leads, sampling frequency, recording time, and other technical specifications of the digital ECG signal. The header information is followed by the beat annotations, where each beat annotation segment consists of 4 bytes of binary data organized as three fields. First two fields are label information for classifying the recorded ECG beat type. The last field contains 2 bytes of information related to the temporal distance (i.e., *toc)* of the current beat from the last recorded beat. The size

*Figure 3. One-lead tracing in which the number on top of each cardiac beat signal represents the time distance in millisecond between the current displayed beats, while the other characters, such as V and S, denote irregular beats corresponding to potential heart conditions.*


of the annotation file depends on the length of the acquired ECG tracings. In our experiments, we will use a sample ECG annotation file from the THEW ECG database (Courderc, 2010), which has a 24-hour ECG tracing record of a patient and contains 87,896 beat annotations.

## THE DESIGN OF A CLOUD-BASED MEDICAL APPLICATION

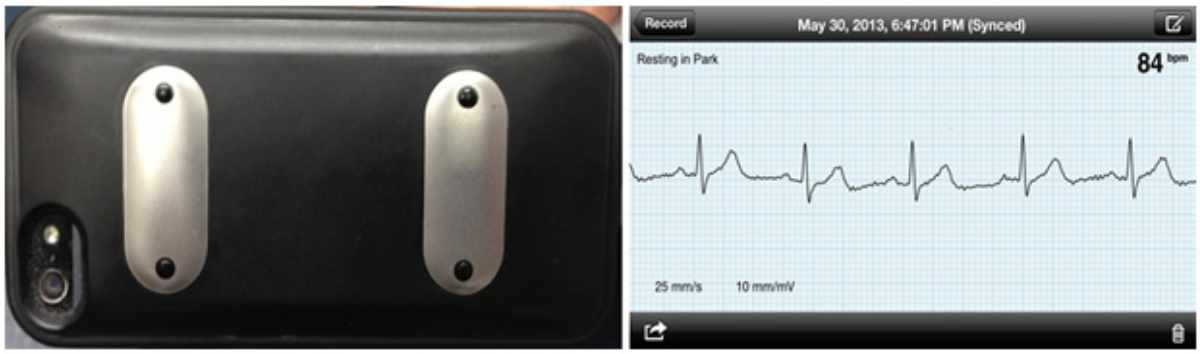
Our proposed cloud-based application is based on offloading almost entire computation to the cloud. Our application is based on mainly three distinct parts: 1) Real-time medical data acquisition devices, 2) Cloud-based storage and computation, 3) GUI (end) node. In the following subsections, we will analyze each part individually.

### Data Acquisition through Thin Devices

Acquisition devices are front-end of our cloud based medical application. These devices are capable of acquiring real-time medical data. Examples of such devices are disposable ECG patches attached to a patient or mobile ECG carts used in hospitals. Furthermore, with decades of research and development, current ECG recording technologies have matured enough to allow a patient to self-monitor at home. Figure 4 (left) shows a sample device from Alivecor (2013), which can be attached to a Smartphone and the software that is included with the device is capable of recording ECG samples. A sample ECG recording obtained from the device is shown in Figure 4 (right), which has sufficient accuracy to be useful in clinical diagnostics. To protect the patient's privacy, we assume the acquisition devices are capable of performing AES encryption of patient data and transmitting the encrypted data wirelessly (Fahad et al, 2012; Soyata et al, 2012b; Soyata et al, 2012c; Soyata et al, 2013).

Considering the significant computational difference of encrypting data between AES (National Institute of Standards and Technology, 2001) and FHE, it is unrealistic for an acquisition node to execute real-time FHE encryption, while AES encryption has trivial computational demands and available even in the least expensive devices. Therefore, we formulate the acquisition node is oblivious to FHE encryption and only responsible for encrypting the patient data with AES encryption, while the conversion of AES encrypted data to FHE encrypted data is performed in the cloud.

*Figure 4. (Left) Commercial ECG screening device from Alivecor. (Right) Recorded ECG data using the Alivecor device*




## AES to FHE Conversion Agent

Since homomorphic encryption cannot be performed during the acquisition phase, the data has to be transmitted into the cloud in AES-encrypted format. We propose to store all of the patient data in AES-encrypted format, since AES is a storage-neutral conversion (i.e., the AES-encrypted version of a 128-bit raw data occupies 128-bits also). While this completely solves the privacy of the stored data, conversion of AES-encrypted data to FHE-encrypted data has to be performed at some point, before any computation can be done by using FHE. We will experiment with a background AES to FHE conversion agent, a portion of the cloud software to continuously convert the AES-encrypted data into its FHE counterpart.

Converting AES-encrypted data to FHE-encrypted data requires evaluating AES decryption function homomorphically. To estimate the cost of AES to FHE conversion we refer to (Gentry et al., 2012). In (Gentry et al., 2012), the authors implemented the AES-128 decryption function with the BGV scheme (Brakerski et al., 2012) and provided latency/throughput analysis with different design choices. An AES-128 decryption operates on blocks of 128-bit (i.e. 16B) data, where granularity of the operations is 1 Byte. In the first design, a ciphertext is set to hold 864 plaintext slots where each slot holds information for 1B message. With this setting 16 slots can be used to contain one AES-encrypted data, thus $\lfloor 864 \div 16 \rfloor = 54$ AES decryption operation can be performed in parallel. The overall evaluation runs in 36 hours; however since 54 AES decryptions have been performed in parallel, throughput is around 40 minutes per one AES decryption. In the second design, 16 ciphertexts are used and each ciphertext is set to hold 720 plaintext slots. Similar to the first design settings each slot holds information for 1B message, but this time each slot associated with different AES-encrypted data, thus 720 AES decryption operation can be performed in parallel. Although with this setting total evaluation time is around 5 days, throughput for one AES decryption is reduced to 5 minutes. Although the second design provided better throughput results than the first design, it requires larger memory to store all variables. Therefore, we will use the first design setting as our reference.

Based on the results were reported in (Gentry et al, 2012) to be around 36 hours for the decryption of 54 AES blocks (16B each), approximately 150 Sec is needed to convert 1B. Using these results as the basis, we calculate that, the AES to FHE conversion agent will need to process 87,896 beat annotations (175,792B assuming 2B per annotated element) to convert a 24-hour patient annotated recording to FHE. Therefore, the computation time for this conversion is approximately 7,324

hours. Using the estimated conversion time, the required speedup is around 305x to compute the results at the rate of arrival (i.e., 24-hours). We will show in the following subsection how, it is possible to parallelize this process to perform AES to FHE conversion at the rate of arrival if sufficient hardware parallelism is available.

## Storage and Computation in the Cloud

As previously mentioned, acquisition nodes are assumed to be capable of AES encryption and AES-encrypted version of the medical records permanently stored in the cloud. In order to operate on medical data with FHE, AES-encrypted medical records have to be converted to FHE-encrypted version. Although we note that AES to FHE conversion is compute-intensive, this conversion has to be performed only once.

The AES to FHE conversion can be performed offline while the conversion time will be exposed as a delay in providing the remotely monitored patient data to the doctor. This delay might not be important, since the doctor typically needs these results in a few days after the remote monitoring has been completed. This latency tolerance can be translated into further cost savings for the HCO, by performing AES to FHE conversion when the computation resources are less expensive. For instance, Amazon Web Services (AWS) offers Micro instances which have basic computation capabilities yet they can be rented at no cost.

In addition to the delay in providing AES to FHE conversion, a certain amount of compute-caching can also be performed offline. For example, assume a set of 10,000 results that need to be added to provide the average heart rate to the doctor. These results to add are generally in very predictable intervals, thereby generating predictable patterns in pre-computable results. As an example, to reduce the real-time compute strain in the cloud when the doctor is running the application, every 100 results can be summed, and the results can be cached in the storage area. Such a process can be accelerated using specialized accelerators (Guo et al, 2010; Soyata et al, 2012a) and computation optimization techniques (Soyata et al, 1993; Soyata & Friedman, 1994a; Soyata & Friedman, 1994b; Soyata et al, 1995; Soyata & Friedman, 1997; Soyata, 1999).

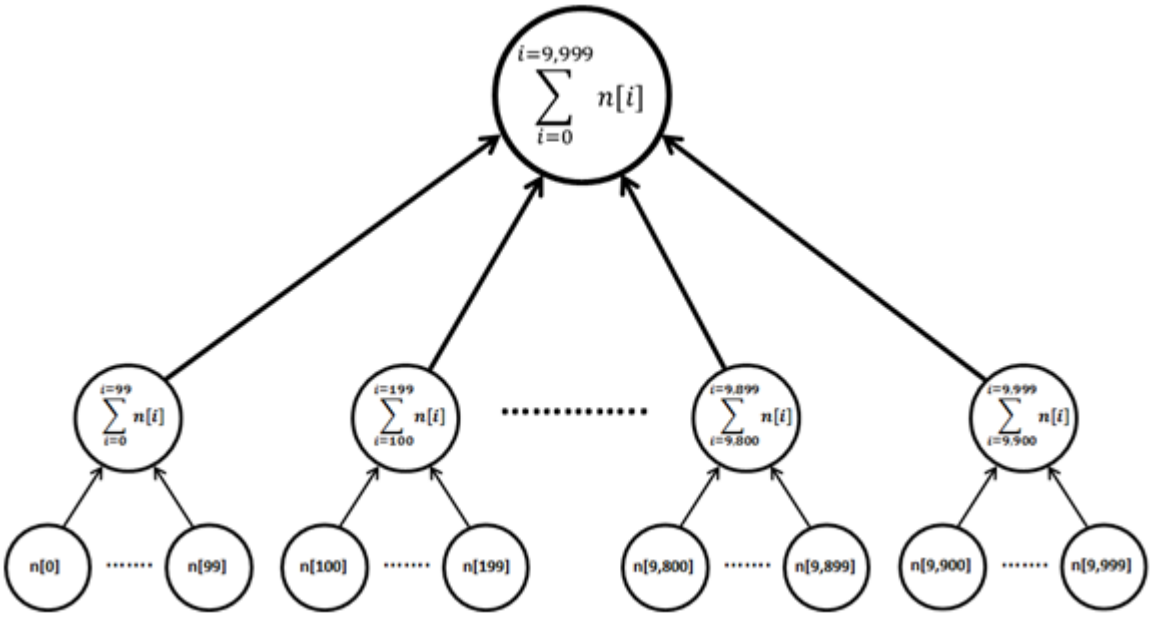
In this specific example, which is also demonstrated in Figure 5, a typical operation is to calculate the sum (and, thus, the average) of 10,000 numbers. It is feasible to pre-compute sums for 100-number chunks. Observing that, this will expand the required storage by 100x as compared to storing only the initial 10,000 results, this provides a trade-off between latency and storage by shifting the application execution time from offline to online computation. This idea can be further expanded by building a compute-cache that has a log-tree structure by calculating every 100, and every 10,000, etc., permitting computations to be sped up at the expense of higher storage.

## Storage Management in the Cloud

Considering the significant amount of storage that FHE requires, a natural question to ask is the total required amount of storage for each application. We conceptualize the cloud storage that an FHE-enabled application requires as composed of three separate areas: 1) The AES-data area, which is where the medical records are permanently stored in AES-encrypted format, 2) The FHE data-cache area, which is the FHE-encrypted copy of the original AES data, only for certain records, 3) The FHE compute-cache area, which is the pre-computed results for portions of the FHE data-cache.

We assume that, three distinct spaces will be allocated to each one of AES data, FHE data-cache, and FHE-compute cache. This implies a hierarchical storage which resembles closely a computer's memory subsystem, where, AES-data area is analogous to a computer disk, since the conversion from AES to FHE takes a long time,

*Figure 5. Compute-caching example*




FHE data-cache is analogous to computer memory, since there is a significant penalty in bringing the data in from the memory into the cache, and FHE compute-cache is analogous to L3 cache, where the results in this cache can be converted to useful results significantly faster than the ones in the FHE data-cache.

In this proposed tiered storage scheme, the FHE data-cache and FHE compute-cache are completely disposable, i.e., discarding any information in these caches only hurts performance, but does not cause data loss. This allows the cloud application to dynamically adjust the contents of each cache, thereby modulating the application response time vs. required storage and computation.

## Displaying the Application Results through GUI Devices

The backend of our application is the GUI device which runs the GUI portion of the medical application and displays the results to the doctor. Since the cloud is responsible for performing entire set of computations with FHE, the end result will be in the FHE-encrypted format when it is transferred to the GUI device. This necessitates that the GUI device has to perform decryption of FHE-encrypted ciphertexts. Furthermore, to avoid exposing the medical data at any point, decryption needs to be performed only on the Smartphone of the authorized personnel.

Considering that, within the FHE framework, the decryption has a fairly low compute-intensity as compared to the intermediate computations, this is feasible for the GUI device. Since most current Smartphones have multiple processor cores and are expected to amass an ever increasing computational power, it is reasonable to expect the decryption process to take close to real-time and acceptable to the user. Therefore the GUI end-node has to have minimal capability in 1) running an OS such as Android or iOS to provide a user interface to the doctor, and 2) perform homomorphic decryption.

## PERFORMANCE EVALUATION

In this section, we will evaluate calculating the average heart rate of a patient with two FHE schemes: Gentry's FHE scheme (Gentry, 2009) and the BGV scheme (Brakerski et al, 2012). We will use the library in (Gentry & Halevi, 2011b) for the former scheme, while the library in (Halevi & Shoup, n.d.) will be used for the latter.

We run our simulations on a computation node in UR Bluhive cluster (University of Rochester, Center for Integrated Research Computing, n.d.) which has two Intel Xeon E5450 processors, each with four cores running at 3GHz with 16GB RAM in total.

### Calculating the Average Heart Rate

In order to demonstrate the feasibility of our concept, we selected finding the average heart rate of the patient as our case study. To compute the average heart rate of a patient, we will use an ECG annotation file from the THEW ECG database (Courderc, 2010). The annotation file consists of 24-hour ECG data of the patient captured with a 12-lead Holter system sampling at 1,000Hz. The file contains 87,896 entries for temporal distance (*toc*) of consecutive heart beats and each *toc* value is represented by 12-bit number.

We calculate average heart rate of a patient during $N$ heart beats in two steps: 1) Accumulate the *toc* values for $N$ heart beats, 2) divide the final sum by $N$, and then finally multiply with sample acquisition time. The trivial division and multiplication operation for the second step is expensive to perform with the FHE, thus we require performing this step at the Smartphone. The first step will be computed completely in the cloud and the FHE encrypted result will be sent to the Smartphone along with FHE encrypted information related to the second step (i.e. $N$ and acquisition time). The Smartphone can decrypt the result from the first step, and information related to second step then it can perform trivial division and multiplication to find the average heart rate.

### Results Based on the Gentry's FHE scheme

In Gentry's FHE scheme (Gentry, 2009), encryption is performed on individual bits. In other words, encrypting a message of $m$-bits will generate $m$ ciphertexts. Homomorphic operations on the ciphertexts correspond to bit-wise arithmetic. Specifically, homomorphic addition results in XOR operation and homomorphic multiplication results in AND operation of the message bits. Table 1 presents the execution times for each FHE primitive on the cluster node.

In order to perform integer additions with bit-wise operations, we choose to implement Ripple Carry Adder. First, we calculate the sum and carry homomorphically for each bit and then the carry is forwarded to next level computation. The noise inside the ciphertext grows during carry computation which involves homomorphic multiplication. To prevent decryption errors, we need to perform recryption operations for the carry before forwarding to next level computation. Based on the results presented in Table 1, recryption operation takes longer than the rest of the operations and thus 99.9% of the execution time for adding two $m$-bit number is spent during recryption operation.

To analyze computational and storage requirements of Gentry's FHE scheme, we calculate the average heart rate of the patient during one-hour. The patient record for one-hour consists of approximately 4,096 *toc* values where each *toc* value is a 12-bit number. A 24-bit accumulator is chosen to prevent overflow for adding 4,096 12-bit numbers. Computing one-hour average heart rate finished in approximately 700 hours on the cluster node. Each ciphertext has a size of roughly 0.1MB and storing one-hour of patient record requires $\approx$ 4.8GB of storage space . Since each ciphertext encrypts one-bit, this is equal to storage expansion of 800,000X. Our experiment results indicate that using Gentry's FHE scheme is impractical both in terms of computation and storage.

*Table 1. Execution time of the operations for the Gentry's FHE scheme*

| Operation | Execution Time |
|---|---|
| Encryption | 1.45 Sec |
| Decryption | 0.2 Sec |
| Recryption | 24.95 Sec |
| Addition | < 1 μs |
| Multiplication | 1.79 ms |

## Results Based on the BGV scheme

In the BGV scheme (Brakerski et al., 2012), messages and ciphertexts are defined over polynomial rings. Homomorphic addition and multiplication of ciphertexts will correspond to ring additions and multiplications respectively. Table 2 presents the execution times for each FHE primitive on the cluster node.

To perform additions of *toc* values with polynomial rings we use the methods described in (Naehrig, M., et al, 2011) to encode each *toc* value. In (Naehrig et al, 2011), each message is represented by its binary encoding and each bit of the message is set as one of the coefficients of the message polynomial. Homomorphic additions correspond to polynomial additions and as long as the coefficients of the plaintext do not exceed plaintext space *p,* correctness are assured. The final result after computation can be recovered by first decrypting the ciphertext and then evaluating the resulting polynomial at 2.

To analyze computational and storage requirements of the BGV scheme, we calculate the average heart rate of a patient during 24 hours. To represent a 12-bit *toc* value we choose to work with polynomials of degree 12. We set the parameters of the BGV scheme which enable us to pack 200 slots in each ciphertext. Since each ciphertext can pack 200 *toc* values, accumulating the 87,896 *toc* values can be performed by $\lceil 87{,}896 \div 200 \rceil = 440$ additions. Based on our simulations on the cluster node, computing 24-hour average heart rate takes approximately 70 ms. In terms of storage, one ciphertext has a size of roughly 65KB and storing entire patient records require 28 MB of storage space. Each ciphertext encrypts 200 *toc* values with 12-bit each, which corresponds to a data expansion ratio of $65{,}000 \times 8/200 \times 12 \approx 217$. While computing the average is slow compared to its no-encryption version, the BGV scheme is very close to providing the result in real-time with a moderate expansion in storage.

We perform following experiment to investigate maximum achievable speedup by utilizing the parallelism in the cloud. We look at the parallelism at the process - level, since the library in (Halevi & Shoup, n.d.) is not thread-safe. We launch multiple concurrent processes and assign each process independent portions of the data.

*Table 2. Execution time of the operations for the BGV scheme*

| Operation | Execution Time |
|---|---|
| Encryption | 1.65 sec |
| Decryption | 0.65 sec |
| Addition | 0.11 ms |
| Multiplication | 0.8 sec |

*Table 3. Multi-process runtime using a dual-socket Xeon server*

| Process | Runtime (ms) | Speedup | Efficiency (%) |
|---|---|---|---|
| 1 | 69.8 | 1.00 | 100 |
| 2 | 35.8 | 1.95 | 97.4 |
| 4 | 21.5 | 3.25 | 81.4 |
| 8 | 11.75 | 5.96 | 74.5 |

The results of each process can be combined later through OS-level pipes. Table 3 presents the speedup due to process-level parallelism on the cluster node. The speedup column is normalized to the single-thread runtime. The Efficiency column indicates the percentage speedup compared to the ideal speedup due to parallelism (i.e., *N* threads for *N* times speedup).

## CONCLUSION AND FUTURE WORK

In this chapter, a long-term health monitoring system is introduced to achieve the end goal of detecting patient health issues by continuously monitoring the ECG data acquired outside the Healthcare Organization (HCO). This system consists of ECG acquisition devices, a cloud-based medical application, and back-end devices that display the monitoring results. While such a system can be trivially implemented by today's technology by using existing ECG devices, cloud computing resources and highly capable Smartphones, one important issue arises when the intended application is a medical application: The protection of Personal Healthcare Information (PHI).

Significant liability is associated with mishandling PHI in the U.S.A., whether intentional or unintentional. The Health Insurance Portability and Accountability Act (HIPAA, n.d.) mandates strict regulations on protection PHI. Due to the unacceptable risks associated with mishandling PHI (due to whatever reason, including hardware or software malfunction or an intentional security breach), cloud operators, such as Amazon (Amazon, n.d.) do not sign a Business Associate Agreement (BAA) which shifts a portion of the liability to the cloud operator. Without a form of a guarantee that the PHI will be safe during cloud based operation, HCO's cannot take the risk and host their medical application in the cloud. This non-starter renders all of the benefits of cloud computing useless to an HCO.

This chapter formulates a system, in which the cloud can execute the medical application without the concern of PHI protection. This is achieved by an encryption system, called, Fully Homomorphic Encryption (FHE), which permits operations on encrypted data. Since the cloud operator can operate on data that it cannot observe, the data is secure even if there is a security breach. Only the parties with a private key can decrypt the data that was initially encrypted with FHE. Therefore, the protection of PHI implies protecting the private keys, which is the same responsibility as protecting passwords when accessing a computer.

We argue that, by providing such a tool for cloud operators to operate on encrypted data, and making the password protection the responsibility of the HCO, cloud operators will be motivated to sign a BAA. In fact, we have observed this at the University of Rochester Medical Center, where a small cloud backup company is willing to sign a BAA as long as the key is not stored in their system and the responsibility of the protection of the private keys lies 100% with the HCO. It is the conclusion of this chapter that, the same concept will eventually extend to the execution of a medical application when small operators sign a BAA to run a medical application as long as they are not

storing the private keys. Managing the privacy of these keys is a significantly easier task for an HCO as compared to managing the privacy of the entire datacenter they are operating. This concept, therefore, holds the key to revolutionizing the US healthcare.

## ACKNOWLEDGMENT

This work was supported in part by the National Science Foundation grant CNS-1239423 and a gift from Nvidia corporation.

## REFERENCES

Alivecor. (2013). *ECG screening made easy*. Retrieved from http://www.alivecor.com

Amazon. (n.d.). *Amazon web services (AWS)*. Retrieved from http://aws.amazon.com

Babai, L. (1985). On Lovász' lattice reduction and the nearest lattice point problem. *STACS*, *85*, 13–20.

Boneh, D., Goh, E. J., & Nissim, K. (2005). Evaluating 2-DNF formulas on ciphertexts. In *Theory of cryptography* (pp. 325–341). Berlin: Springer. doi:10.1007/978-3-540-30576-7_18

Brakerski, Z., Gentry, C., & Vaikuntanathan, V. (2012). (Leveled) fully homomorphic encryption without bootstrapping. In *Proceedings of the 3rd Innovations in Theoretical Computer Science Conference* (pp. 309-325). ACM.

Brakerski, Z., & Vaikuntanathan, V. (2011). Efficient fully homomorphic encryption from (standard) LWE. In Proceedings of Foundations of Computer Science (FOCS), (pp. 97-106). IEEE.

Brakerski, Z., & Vaikuntanathan, V. (2011). Fully homomorphic encryption from ring-LWE and security for key dependent messages. In *Proceedings of Advances in Cryptology–CRYPTO 2011* (pp. 505–524). Berlin: Springer. doi:10.1007/978-3-642-22792-9_29

Cohen, J. D., & Fischer, M. J. (1985). A robust and verifiable cryptographically secure election scheme. In *Proceedings of Foundations of Computer Science* (pp. 372–382). IEEE. doi:10.1109/SFCS.1985.2

Coron, J. S., Mandal, A., Naccache, D., & Tibouchi, M. (2011). Fully homomorphic encryption over the integers with shorter public keys. In *Proceedings of Advances in Cryptology–CRYPTO 2011* (pp. 487–504). Berlin: Springer. doi:10.1007/978-3-642-22792-9_28

Couderc, J. P. (2010). The telemetric and holter ECG warehouse initiative (THEW): A data repository for the design, implementation and validation of ECG-related technologies. In Proceedings of Engineering in Medicine and Biology Society (EMBC), (pp. 6252-6255). IEEE.

Damgård, I., & Jurik, M. (2001). A generalisation, a simplification and some applications of Paillier's probabilistic public-key system. In *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography: Public Key Cryptography* (pp. 119-136). Berlin: Springer-Verlag.

Diffie, W., & Hellman, M. (1976). New directions in cryptography. *IEEE Transactions on Information Theory*, *22*(6), 644–654. doi:10.1109/TIT.1976.1055638

ElGamal, T. (1985). A public key cryptosystem and a signature scheme based on discrete logarithms. *IEEE Transactions on Information Theory*, *31*(4), 469–472. doi:10.1109/TIT.1985.1057074

Fahad, A., Soyata, T., Wang, T., Sharma, G., Heinzelman, W., & Shen, K. (2012). SOLARCAP: Super capacitor buffering of solar energy for self-sustainable field systems. In *Proceedings of SOC Conference* (SOCC), (pp. 236-241). IEEE.

FDA. (2013). *FDA safety communication: Cybersecurity for medical devices and hospital networks*. Retrieved from http://www.fda.gov/medicaldevices/safety/alertsandnotices/ucm356423.htm

Gentry, C. (2009). *A fully homomorphic encryption scheme.* (Doctoral Dissertation). Stanford University, Palo Alto, CA.

Gentry, C., & Halevi, S. (2011). Fully homomorphic encryption without squashing using depth-3 arithmetic circuits. In Proceedings of Foundations of Computer Science (FOCS), (pp. 107-109). IEEE.

Gentry, C., & Halevi, S. (2011). Implementing Gentry's fully-homomorphic encryption scheme. In *Proceedings of Advances in Cryptology–EUROCRYPT 2011* (pp. 129–148). Berlin: Springer. doi:10.1007/978-3-642-20465-4_9

Gentry, C., Halevi, S., & Smart, N. P. (2012). Homomorphic evaluation of the AES circuit. In *Proceedings of Advances in Cryptology–CRYPTO 2012* (pp. 850–867). Berlin: Springer. doi:10.1007/978-3-642-32009-5_49

Goldwasser, S., & Micali, S. (1982). Probabilistic encryption & how to play mental poker keeping secret all partial information. In *Proceedings of the Fourteenth Annual ACM Symposium on Theory of Computing* (pp. 365-377). ACM.

Google. (n.d.). *Google app. engine*. Retrieved from http://code.google.com/appengine

Guo, X., Ipek, E., & Soyata, T. (2010). Resistive computation: avoiding the power wall with low-leakage, STT-MRAM based computing. *ACM SIGARCH Computer Architecture News*, *38*(3), 371–382. doi:10.1145/1816038.1816012

Halevi, S., & Shoup, V. (n.d.). *HElib*. Retrieved from https://github.com/shaih/HElib

HIPAA. (n.d.). *Wikipedia*. Retrieved from http://en.wikipedia.org/wiki/Hipaa

Hoang, D. B., & Chen, L. (2010). Mobile cloud for assistive healthcare (MoCAsH). In *Proceedings of Services Computing Conference* (APSCC), (pp. 325-332). IEEE.

Hoang, D. T., Niyato, D., & Wang, P. (2012). Optimal admission control policy for mobile cloud computing hotspot with cloudlet. In *Proceedings of Wireless Communications and Networking Conference* (WCNC), (pp. 3145-3149). IEEE.

Kocabas, O., Soyata, T., Couderc, J. P., Aktas, M., Xia, J., & Huang, M. (2013). Assessment of cloud-based health monitoring using homomorphic encryption. In *Proceedings of the 31st IEEE International Conference on Computer Design*. IEEE.

Leaf. (n.d). *World's thinnest 3-lead ECG patch*. Retrieved from http://www.clearbridgevitalsigns.com/brochures/CardioLeaf_ULTRA_Brochure.pdf

Lobodzinski, S., & Laks, M. (2012). New devices for every long-term ecg monitoring. *Cardiology Journal*, *19*(2), 210–214. doi:10.5603/CJ.2012.0039 PMID:22461060

Lyubashevsky, V., Peikert, C., & Regev, O. (2010). On ideal lattices and learning with errors over rings. In *Proceedings of Advances in Cryptology–EUROCRYPT 2010* (pp. 1–23). Berlin: Springer. doi:10.1007/978-3-642-13190-5_1

Micciancio, D. (2001). Improving lattice based cryptosystems using the Hermite normal form. In *Cryptography and lattices* (pp. 126–145). Berlin: Springer. doi:10.1007/3-540-44670-2_11

Microsoft. (n.d.). *Windows Azure*. Retrieved from http://www.microsoft.com/windowazure

Naehrig, M., Lauter, K., & Vaikuntanathan, V. (2011). Can homomorphic encryption be practical? In *Proceedings of the 3rd ACM Workshop on Cloud Computing Security Workshop* (pp. 113-124). ACM.

NIST. (2001). Advanced encryption standard (AES) [). Washington, DC: NIST.]. *Federal Information Processing Standard*, *FIPS-197*, 1.

Paillier, P. (1999). Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Advances in Cryptology—EUROCRYPT'99* (pp. 223–238). Berlin: Springer. doi:10.1007/3-540-48910-X_16

Paypal. (n.d). *Paypal*. Retrieved from https://www.paypal.com

PROCEED. (n.d.). *Programming computation on encrypted data*. Retrieved from http://www.darpa.mil/Our_Work/I2O/Programs/PROgramming_Computation_on_EncryptEd_Data_(PROCEED).aspx

Rivest, R. L., Adleman, L., & Dertouzos, M. L. (1978). On data banks and privacy homomorphisms. *Foundations of Secure Computation*, *32*(4), 169–178.

Rivest, R. L., Shamir, A., & Adleman, L. (1978). A method for obtaining digital signatures and public-key cryptosystems. *Communications of the ACM*, *21*(2), 120–126. doi:10.1145/359340.359342

Salesforce. (n.d.). *Salesforce customer relationship management (CRM)*. Retrieved from http://www.salesforce.com

Shusterman, V., Goldberg, A., Schindler, D. M., Fleischmann, K. E., Lux, R. L., & Drew, B. J. (2007). Dynamic tracking of ischemia in the surface electrocardiogram. *Journal of Electrocardiology*, *40*(6), S179–S186. doi:10.1016/j.jelectrocard.2007.06.015 PMID:17993319

Smart, N. P., & Vercauteren, F. (2010). Fully homomorphic encryption with relatively small key and ciphertext sizes. In *Proceedings of Public Key Cryptography–PKC 2010* (pp. 420–443). Berlin: Springer. doi:10.1007/978-3-642-13013-7_25

Smart, N. P., & Vercauteren, F. (2011). Fully homomorphic SIMD operations. In *Proceedings of Designs, Codes and Cryptography*. Academic Press.

Soyata, T. (1999). *Incorporating circuit level information into the retiming process*. (Doctoral Dissertation). University of Rochester, Rochester, NY.

Soyata, T., Ba, H., Heinzelman, W., Kwon, M., & Shi, J. (2013). *Accelerating mobile-cloud computing: A survey*. Academic Press.

Soyata, T., & Friedman, E. G. (1994). Retiming with non-zero clock skew, variable register, and interconnect delay. In *Proceedings of the 1994 IEEE/ACM International Conference on Computer-Aided Design* (pp. 234-241). IEEE.

Soyata, T., & Friedman, E. G. (1994). Synchronous performance and reliability improvement in pipelined ASICs. In *Proceedings of ASIC Conference and Exhibit*, (pp. 383-390). IEEE.

Soyata, T., Friedman, E. G., & Mulligan, J. H. Jr. (1993). Integration of clock skew and register delays into a retiming algorithm. In *Proceedings of Circuits and Systems* (pp. 1483–1486). IEEE. doi:10.1109/ISCAS.1993.394015

Soyata, T., Friedman, E. G., & Mulligan, J. H. Jr. (1995). Monotonicity constraints on path delays for efficient retiming with localized clock skew and variable register delay. [). IEEE.]. *Proceedings of Circuits and Systems*, *3*, 1748–1751.

Soyata, T., Friedman, E. G., & Mulligan, J. H. Jr. (1997). Incorporating interconnect, register, and clock distribution delays into the retiming process. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, *16*(1), 105–120. doi:10.1109/43.559335

Soyata, T., & Liobe, J. (2012). pbCAM: Probabilistically-banked content addressable memory. In *Proceedings of SOC Conference* (SOCC), (pp. 27-32). IEEE.

Soyata, T., Muraleedharan, R., Funai, C., Kwon, M., & Heinzelman, W. (2012). Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture. In Proceedings of Computers and Communications (ISCC), (pp. 59-66). IEEE.

Soyata, T., Muraleedharana, R., Langdonb, J., Funaia, C., Amesc, S., Kwond, M., & Heinzelmana, W. (2012). COMBAT: Mobile-cloud-based compute/communications infrastructure for battlefield applications. [). SPIE.]. *Proceedings of the Society for Photo-Instrumentation Engineers*, *8403*, 84030K–1. doi:10.1117/12.919146

Stehlé, D., & Steinfeld, R. (2010). Faster fully homomorphic encryption. In *Proceedings of Advances in Cryptology-ASIACRYPT 2010* (pp. 377–394). Berlin: Springer. doi:10.1007/978-3-642-17373-8_22

University of Rochester, Center for Integrated Research Computing. (n.d.). *Bluehive cluster*. Retrieved from http://www.circ.rochester.edu/wiki/index.php/BlueHive_Cluster

Van Dijk, M., Gentry, C., Halevi, S., & Vaikuntanathan, V. (2010). Fully homomorphic encryption over the integers. In *Proceedings of Advances in Cryptology–EUROCRYPT 2010* (pp. 24–43). Berlin: Springer. doi:10.1007/978-3-642-13190-5_2

## KEY TERMS AND DEFINITIONS

**Advanced Encryption Standard (AES):** Widely used symmetric-key cryptography published by National Institute of Standards and Technology (NIST).

**Cloud Computing:** A distributed computing system that relies on use of shared resources connected by the Internet to manage data and perform computations.

**Electrocardiogram (ECG):** Recording electrical activity of the heart to measure and diagnose abnormal rhythms of the heart.

**Encryption:** Encoding the contents of a message such that only authorized parties can access the message.

**Graphical User Interface (GUI):** An interface that allows visualization of information in graphical format.

**Holter System:** Portable monitor used for recording electrical activity of a patient continuously during 24-48 hours of daily activity.

**Homomorphic Encryption:** An encryption system capable of performing meaningful operations on the encrypted messages without accessing the original message.

**Lattice-Based Cryptography:** Cryptographic systems in which primitives are based on the hardness of lattice problems.

**Long Term Health Monitoring:** Monitoring patients during extended period of time for diagnosing and treating health issues at an early stage.

**Mobile-Cloud Task Partitioning:** Partitioning and assigning different subtasks to mobile devices or to cloud based on computational resource requirements of each subtask.