Emerging Security Mechanisms for Medical Cyber Physical Systems

Ovunc Kocabas, Tolga Soyata, and Mehmet K. Aktas

Abstract—The following decade will witness a surge in remote health-monitoring systems that are based on body-worn monitoring devices. These Medical Cyber Physical Systems (MCPS) will be capable of transmitting the acquired data to a private or public cloud for storage and processing. Machine learning algorithms running in the cloud and processing this data can provide decision support to healthcare professionals. There is no doubt that the security and privacy of the medical data is one of the most important concerns in designing an MCPS. In this paper, we depict the general architecture of an MCPS consisting of four layers: data acquisition, data aggregation, cloud processing, and action. Due to the differences in hardware and communication capabilities of each layer, different encryption schemes must be used to guarantee data privacy within that layer. We survey conventional and emerging encryption schemes based on their ability to provide secure storage, data sharing, and secure computation. Our detailed experimental evaluation of each scheme shows that while the emerging encryption schemes enable exciting new features such as secure sharing and secure computation, they introduce several orders-of-magnitude computational and storage overhead. We conclude our paper by outlining future research directions to improve the usability of the emerging encryption schemes in an MCPS.

Index Terms—Medical cyber physical systems, medical data privacy, homomorphic encryption, attribute-based encryption

1 Introduction

The coming decade will witness an explosive growth in systems that monitor a patient through body-worn inexpensive personal monitoring devices that record multiple physiological signals, such as ECG and heart rate [1], [2], or more sophisticated devices that measure physiological markers such as body temperature, skin resistance, gait, posture, and EMG [3], [4]. The emergence of these devices combined with user awareness for their importance in personal health monitoring even emerged trends to make such devices *fashionable* [5].

The unstoppable momentum in the development of such devices enabled the construction of complete patient health monitoring systems that can be clinically used [6], [7], [8]. The medical data that is acquired from patients by a distributed sensor network can be transmitted to private [9], [10] or public [11], [12], [13] cloud services. A set of statistical inference algorithms running in the cloud can determine the correlation of the patient data to known disease states. These correlations could be fed back to healthcare professionals as a means to provide decision support. Such systems, termed Medical Cyber-Physical Systems (MCPS), signal the beginning of a new Digital-Health (D-Health) era and a disruptive technology in human history.

Establishing MCPSs will require overcoming technological hurdles in building the architectural components of the MCPS such as sensors, cloud computing architectures, and fast Internet and cellular phone connections. Additionally, assuring the privacy of the personal health information during the transmission from the sensory networks to the cloud and from the cloud to doctors' mobile devices will necessitate the design of a sophisticated cryptographic architecture for an MCPS. While this design implies only *secure storage* using conventional encryption schemes, emerging encryption schemes provide options for *secure data sharing* and *secure computation*.

The contribution of this paper is two-fold: First, we survey conventional and emerging encryption schemes that can be used in designing an MCPS. Second, we provide an extensive evaluation of these schemes and compare them based on their ability to provide secure storage, secure data sharing, and secure computation.

The remainder of this paper is organized as follows: Section 2 provides a description of the architecture of an MCPS. Section 3 introduces the adversary models for designing a secure MCPS, followed by Section 4, which details the privacy requirements of each MCPS architectural component. Cryptographic methodologies, used in MCPSs, are detailed in the following three sections: Section 5 provides details for the conventional AES and ECC encryption. Sections 6 and 7 detail the emerging attribute-based and homomorphic encryption (HE) mechanisms, respectively. Section 8 presents an implementation case study of a medical application using homomorphic encryption. Section 9 details the setup for experiments and a quantitative and qualitative evaluation of all of these cryptosystems are provided in Section 10. Conclusions are drawn pertaining to the suitability of each cryptosystem for different MCPS architectural components in Section 11.

2 Medical Cyber Physical Systems

A typical MCPS architecture consists of four different layers: i) data acquisition layer, ii) data pre-processing layer,

- *O. Kocabas and T. Soyata are with the Department of Electrical and Computer Engineering, University of Rochester, Rochester, NY 14627. E-mail: {ovunc.kocabas, tolga.soyata}@rochester.edu.*
- *M.K. Aktas is with the University of Rochester Medical Center, Rochester, NY 14642. E-mail: mehmet_aktas@urmc.rochester.edu.*

Manuscript received 7 Sept. 2015; revised 7 Jan. 2016; accepted 12 Jan. 2016. Date of publication 22 Jan. 2016; date of current version 1 June 2016.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TCBB.2016.2520933

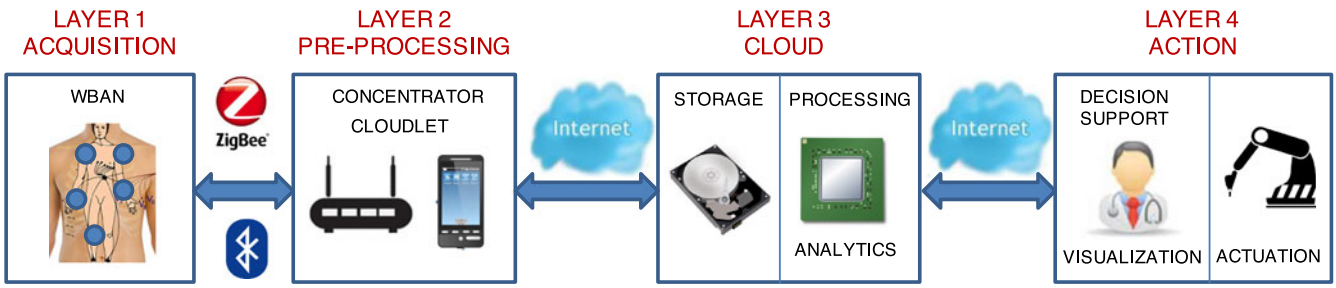

Fig. 1. Four layers of a typical Medical Cyber Physical System. Each layer is characterized by different constraints. The communication among the layers must be protected using different cryptographic standards.

iii) cloud processing layer, and iv) action layer. An architectural map of an MCPS is shown in Fig. 1. In this section, the details of operation and security requirements for each layer will be introduced.

2.1 Data Acquisition Layer

Data acquisition layer is typically a Body Area Network (BAN) consisting of wireless wearable sensors [6], [14] for specific medical applications such as blood pressure and body temperature monitoring [15], or data storage for on-demand access by doctors [16]. A BAN facilitates the collection of patient medical information and forwards this information to a nearby computationally-capable device such as a cloudlet [17]. Battery-operated active sensors in the BAN use Bluetooth or ZigBee protocols while battery-less passive sensors use RFID.

2.2 Data Concentration/Aggregation Layer

Due to the low computational power of the sensors that make up a BAN, an intermediate device, either a cloudlet or a concentrator is necessary. In [15], sensors transmit the gathered information to a gateway server (acting as a concentrator) through a Bluetooth connection. A concentrator is the most important building block of an IoT-based architecture [18], since it enables individually-weak devices to have strong overall functionality by concentrating the data from each device and sending the aggregated information to the cloud. A cloudlet is similar in purpose, but is designed to aggregate data from more powerful devices too, e.g., a smartphone. Typically a cloudlet is built from a dedicated computer and has a dedicated Internet connection [19], [20].

2.3 Cloud Processing and Storage Layer

Since accurate diagnosis requires long-term patient health monitoring information, secure *storage* is the most important function of the cloud [21], [22]. Additionally, government health regulations require the storage of medical records for an extended amount of time. Many cloud operators store medical data by signing a Business Associate Agreement (BAA). Medical institutions run their applications in their private cloud (i.e., datacenter), therefore using the cloud for the second important purpose: *processing*. However, as we will detail in Section 7, privacy-preserving processing in a public cloud is only feasible using advanced homomorphic encryption schemes. Third function of the cloud is *data analytics* to facilitate decision support for healthcare professionals [23], [24] by applying statistical inference algorithms to the acquired data and predicting patient health condition. These methods have recently received attention in remote health monitoring systems [25].

2.4 Action Layer

The action layer can provide either "active" or "passive" action. In *active action*, an actuator is used to turn the results of the algorithms that run in the cloud into the activation of an actuator such as a robotic arm. Examples of this type of action are robot-assisted surgery [26]. In *passive action*, no physical action is actually taken. The outcome of the analytics or medical application results are given to the requesting authority to provide decision support. An example of passive action is the visualization of a patient's long-term ($\geq$ 24-hr) Holter ECG monitoring, allowing the visualization of 20-30 patients' monitoring results by a doctor within 10-20 seconds [27].

3 MCPS Adversary Models

An essential part of designing a secure MCPS is determining system security requirements based on the capabilities of potential attackers. In this section, we study adversary models and side channel attacks related to the security vulnerabilities of an MCPS.

3.1 Adversary Models

An MCPS must be resilient to attacks on all four of its layers. An adversary model captures the capabilities of an attacker. We consider two adversary models [28]: active (i.e., *malicious*) and passive (i.e., *honest but curious*). An *active adversary* takes control of the host and can arbitrarily deviate from a specified protocol in order to steal secret information. Alternatively, a *passive adversary* follows the protocols correctly (*honest*), but can look at the encrypted data during the execution of protocols (*but curious*) to obtain information.

Data privacy is one of the features that an MCPS must provide at every level. All of the encryption schemes that are considered in this paper protect data privacy against an active adversary. The only exceptions are the case where there is an attack directly at the crypto-level that "breaks" the encryption through a brute-force attack. This could happen if the security parameters of an encryption scheme are chosen to be weak. Alternatively, a side channel attack could attempt to steal the secret/private key, as will be detailed in Section 3.2.

Correctness of the computed results (verification) is another feature that must be provided for an MCPS that aims to perform secure (encrypted) computations. As will

be detailed in Section 7, secure computation over medical data in a public cloud can only be achieved using homomorphic encryption schemes. However, homomorphic encryption schemes are *malleable* by design; an active adversary can modify the computation result without knowing the private key. Therefore the correctness of the computations cannot be guaranteed when an active adversary model is considered.

To summarize, an MCPS provides *only* data privacy against an active adversary, while it can guarantee *both* data privacy and correctness against a passive adversary. The passive adversary model has been widely used for determining the security requirements of many cloud-based secure computation systems [29], [30], [31]. We also assume that an adversary cannot collude with the parties that hold the secret/private key of the symmetric/public key encryption schemes, since this type of an attack cannot be protected against by using any encryption scheme. We further note that the correctness of the secure computation can be achieved by using techniques from verifiable computing [32] or homomorphic signatures [33]. However, these techniques introduce additional performance penalties to encryption schemes that are already too slow to be practical.

3.2 Side Channel Attacks

Although encryption schemes go through rigorous mathematical and theoretical cryptanalysis to provide security and privacy, the system can still leak information due to the vulnerabilities in its software and hardware implementations. Attacks based on such leaked information are called *side channel attacks*. These attacks can be prevented by using leakage resistant cryptography [34], albeit at the expense of severe performance penalties that make an MCPS impractical.

Side channel attacks concentrate on obtaining the secret/private key by using every layer of the *system*, rather than just the *data* that is being processed by the system. While many types of side channel attacks exist for nearly every encryption scheme [35], we restrict our focus on attacks on AES and Elliptic Curve Cryptography (ECC), which are the most common encryption schemes for building an MCPS. We will detail AES and ECC in Section 5.

Timing attacks are based on observing the execution time of the operations performed during encryption/decryption to reveal the secret key. Depending on the implementation, execution time of the operations can vary based on the bits of the secret key [36]. Timing attacks on AES usually observe cache memory access patterns during the execution of AES operations. Timing attacks on ECC target the scalar multiplication operation, and they can be prevented by using Montgomery's multiplication method [37], which performs the multiplication independent from the bits of the private key [38].

Power analysis attacks are based on observing the power consumption during the execution of cryptographic operations [39]. Power consumption can vary based on the bit values of the secret/private key, allowing an attack by either observing the power usage of devices (simple power analysis) or using statistical methods to capture information in the presence of measurement errors and noise (differential power analysis). Differential power analysis attacks are more powerful due to their noise tolerance in power measurements. Power analysis attacks on AES can be prevented by using randomized masks for AES operations [40] that scramble the relationship between the AES secret key and the intermediate values generated during each AES round. Power analysis based attacks on ECC-based encryption schemes can be mitigated by methods proposed in [41] that randomize intermediate computations to avoid information leakage about the private key from power consumption patterns.

Fault-based attacks are based on introducing faults to bits during the execution of cryptographic operations [42], [43], by applying a power glitch, magnetic field, light source, etc. This would cause errors in operations that can reveal the secret/private key to the attacker. In [44], the authors propose a method to thwart fault based attacks against AES by verifying the correctness of the encryption. The message is first encrypted and compared against the decrypted ciphertext to determine whether a fault was introduced during the encryption. Correctness of the decryption can be verified in a similar fashion by reversing the operations. Their method introduces significant hardware overhead. In [45], the authors propose a novel technique to detect faults based on Error Detecting Codes (EDC), which reduce the hardware overhead and latency. For ECC-based encryption schemes, fault-based attacks are focused on introducing error during the decryption to produce a point that is not on the elliptic curve [46]. These attacks can be mitigated by checking if the calculated point is on the elliptic curve and discarding incorrect computations. Implementations of various cryptographic architectures against fault-based attacks are proposed in [47], [48].

Cache attacks are based on measuring the cache access latency of the cryptographic instructions to recover the cache lines that store the secret key [49], [50]. The information about memory access patterns can be measured by running a malicious program in parallel with other processes. Cache attacks on AES implementations generally target the lookup tables that store S-Boxes [51]. Intel AES-NI instructions [52] can thwart cache attacks by making the cache access latency independent of the data and performing operations on the hardware without using lookup tables. Cache attacks on ECC exploit the precomputed values that are used during point addition in OpenSSL implementations [53]. ECC-based cache attacks can be prevented by i) using blinding scalar for point multiplication, ii) randomizing addition and multiplication chains, and iii) balancing number of additions and multiplications [53].

4 Data Privacy in an MCPS

According to the Health Insurance Portability and Accountability Act (HIPAA) [54], data privacy must be protected within every layer of an MCPS. Individual encryption schemes ensure that medical data is accessed by only the authorized parties, thereby providing data privacy on isolated data blocks. However, ensuring system-level security requires designing a crypto-architecture for the MCPS as a whole. In this section, system-level view of data privacy is studied the the details of individual encryption schemes are provided in Sections 5, 6 and 7.

4.1 Key Management Techniques

Regardless of the type of encryption scheme, communicating parties must agree on key(s) to encrypt/decrypt messages. In the public-key cryptography, *sender* uses the public key of the *receiver* to encrypt messages and the *receiver* uses his/her private key to decrypt encrypted messages. Every user in the system has a dedicated public and private key pair generated by a Public-Key Infrastructure (PKI). PKI is a trusted third party such as a certificate authority that authenticates the key pairs by binding them to the identity of users. For symmetric-key cryptography, both *sender* and *receiver* must share the same secret key to encrypt/decrypt messages. Both parties perform a key-exchange protocol, such as Diffie-Hellman key exchange, to generate the secret key. Once both parties share the same key, they can use symmetric-key cryptography to securely transfer the data.

4.2 Data Acquisition Privacy

The acquisition layer in Fig. 1 is composed of BAN sensor devices with limited computational capability and battery life [55]. Therefore, encryption schemes used to protect the communication within BAN sensors and BAN-to-cloudlet communications must not be computationally intensive. One possible option is to use the Zigbee protocol that is based on the AES encryption scheme and can easily be implemented using low cost microcontroller-based devices. Communicating devices have to agree on a secret-key before using AES encryption by using generic key exchange algorithm such as Diffie-Hellman (DH) [56] or its elliptic curve counterpart Elliptic Curve Diffie-Hellman (ECDH).

Communication of devices can be also secured by using biomedical signals. In [57], authors propose a low-power bio-identification mechanism using the interpulse interval (IPI) to secure the communication between BAN sensors. IPI is the distance between two R peaks and is available to all sensors. In [58], authors use physiological signals to agree on a secret key of the symmetric key cryptosystem for pairwise BAN sensor communication. Compared to ECDH, [58] features authentication capability, requires fewer clock cycles to execute, but has a larger memory footprint. Therefore, [58] offers a viable option for key agreement in BANs.

4.3 Data Sharing Privacy

In many real-world healthcare scenarios more than one party may need to access the data such as i) the patient being monitored, ii) his/her doctor, and iii) in an emergency, other health care personnel. In these cases, conventional encryption schemes cannot handle the sharing of the secret key among multiple parties. Encrypting the data using each party's public key is not a solution either since it creates duplicates of the data, which must be managed separately. Attribute based encryption (ABE) [59], [60], [61] allows secure sharing of data among multiple parties. ABE is a public-key crypto-system that provides fine-grained access control similar to Role Based Access Control [62]. Only the users whose credentials/attributes satisfy the rules determined by the access policy can retrieve the data. In [63], authors propose methods to secure data storage in BANs and distribute data access control. They use the ABE scheme [60] to control who accesses the patient data. ABE encryption is applied to data on a nearby local server and the communication between the BAN and the local server is secured using symmetric key encryption.

4.4 Data Computation Privacy

Conventional encryption schemes do not allow computations on encrypted data without first decrypting it. Decryption necessitates a trusted storage such as health-care organizations' datacenter or a private cloud. This eliminates the option to run analytics, monitoring algorithms (e.g., ECG monitoring [64]) or other algorithms in a public cloud to reduce health care costs. Fully Homomorphic Encryption (FHE) [65] allows computation on encrypted data. By using FHE, the data can be stored in untrusted storage environments, such as public clouds [66], and computations on the encrypted data can be performed without violating the privacy of the data. In [67], a privacy-preserving medical cloud computing system is proposed based on FHE. Authors show that simple operations, such as the computation of average, minimum and maximum heart rate can be implemented at a reasonable cost despite the complexity of FHE.

5 Data Privacy Using Conventional Encryption Schemes

In this section, we study the conventional AES and ECC encryption schemes, which can only guarantee data privacy. However, they are widely used due to their substantially lower resource requirements as compared to emerging schemes.

5.1 Advanced Encryption Standard (AES)

AES [51] is one of the most widely used symmetric key encryption algorithms and is accepted as an industry and a government applications standard. AES is optimized for speed, low memory footprint and energy efficiency. Its low resource intensity allows AES to run on a wide range of hardware platforms ranging from 8-bit microcontrollers to high-end desktops and servers.

5.1.1 AES Encryption and Decryption

AES is a block-cipher and operates on 128-bit blocks of data in multiple rounds (n_r). AES is specified for three different key sizes: AES-128 (128-bit key and $n_r = 10$), AES-192 (192-bit key and $n_r = 12$) and AES-256 (256-bit key and $n_r = 14$). AES represents both the *plaintext* (i.e., original data) and the *ciphertext* (encrypted data) using 128-bit blocks that are arranged as 4×4 matrices, defined as AES *states*. Each matrix entry is 1B = 8-bits and represents an element in the finite field $\mathbb{F}_{2^8}$ using the reduction polynomial $\mathbb{G}(x) = x^8 + x^4 + x^3 + x + 1$.

AES Encryption (Fig. 2) involves XOR, data shuffling, or replacement-by-lookup operations, making encryption very fast and power-efficient. AES Decryption uses the same operations in reverse order. AES encryption/decryption involves these four operations:

KeyExpansion generates a total of n_r+1 *round keys* from the AES secret key iteratively for n_r rounds of AES implementation. Each round key is 1 word = 32 b.

input : Plaintext Block $ptxt_b$, Secret Key sk
output: AES state $state$
$state = InitState(ptxt_b, sk)$
$AddKey(state, sk_0)$
for $i = 1$ **to** $n_r - 1$ **do**
  $SubBytes(state)$
  $ShiftRows(state)$
  $MixColumns(state)$
  $AddKey(state, key_i)$
$SubBytes(state)$
$ShiftRows(state)$
$AddKey(state, key_{n_r-1})$

Fig. 2. AES encryption algorithm. Decryption is achieved by reversing operations.

AddKey applies XOR operation to AES *state* with the roundkeys that are computed during KeyExpansion step. The secret key is used only during this step.

SubBytes applies a non-linear transform of AES *states* and transform each byte of the state using S-boxes.

ShiftRows cyclic left shifts the state matrix rows.

MixColumns applies transformation on the columns of the AES state based on operations in $\mathbb{F}_{2^8}$ and can be represented as a matrix multiplication.

When a plaintext is longer than the AES block size, AES encryption/decryption can be used by choosing one of these modes of operation: Electronic Code Book (ECB), Ciphertext Chain Blocking (CBC), and Counter (CTR). A recent proposal is Galois Counter Mode (GCM) [68], which provides authentication as well as confidentiality. GCM combines the speed of CTR mode with hashing to provide an authenticated encryption mechanism. Confidentiality of the messages is protected using AES and integrity of the communication is provided using a universal hash function.

5.1.2 AES Implementations

CPU instruction set implementations of AES, such as the Intel AES-NI [52] and ARM v8 Cryptography extensions [69], accelerate AES encryption/decryption and generally provide countermeasures against side channel attacks such as timing and cache-based attacks.

Embedded hardware implementations of AES encryption/decryption utilize restricted resources available in hardware platforms such as ASIC and FPGA. Efficient hardware implementations focus on the SubBytes step, which is the only non-linear step in AES. This step involves computing inverse of an element in $\mathbb{F}_{2^8}$, which is the most compute-intensive operation, followed by an affine transformation. Usually SubBytes can be computed by storing all possible combinations in an Substitution Box (S-Box) and use the S-Box as a lookup table. However, this requires additional hardware resources.

Several proposed optimizations [70], [71], [72] improve S-Box computation functionality by representing the AES finite field $\mathbb{F}_{2^8}$ as a composite field such as $\mathbb{F}_{(2^4)^2}$ or $\mathbb{F}_{((2^2)^2)^2}$ (i.e., *tower field*). While representing operations in the composite field requires additional back-and-forth conversions to $\mathbb{F}_{2^8}$, overall computation time is reduced due to the simplified intermediate operations.

Choosing a basis for the tower field is also crucial for the implementation, and three different choices exist for selecting a basis: polynomial [70], normal [71], and mixed [72]. While normal basis provides efficient inversion operation, polynomial basis provides better multiplication performance. In [72], the authors propose using both normals basis as a mixture, and show that the critical path delay can be improved compared to using polynomial-only or normal-only basis. Finite fields can have many irreducible polynomials; 432 possible options are considered in [71] up to 20 percent reduction in terms of gates is reported by picking the optimum choice. Efficiency of AES implementation in the tower field also depends on choosing the coefficients of irreducible polynomials. In [73], 16 possible choices are studied for choosing these coefficients and a reduction in gate size and critical path delay has been reported.

Implementations of AES-GCM are provided using dedicated hardware [74] or by using the instruction set support within Intel CPUs [75].

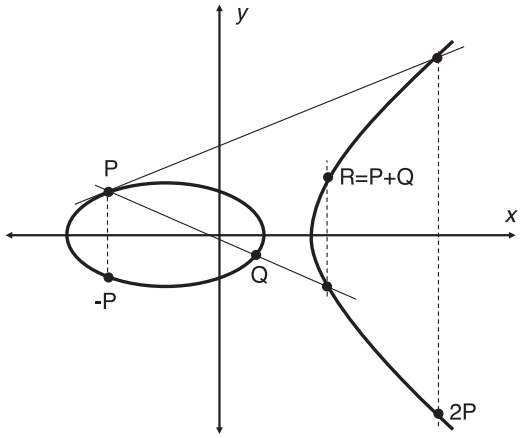

Fig. 3. An Elliptic Curve and the *point addition* and *point doubling* operations on this curve.

5.2 Elliptic Curve Cryptography

Elliptic Curve Cryptography emerged as a public key cryptosystem that achieves the same security level of RSA using a shorter key size [76], [77]. Fig. 3 depicts an example elliptic curve. Security of ECC is based on hardness of the elliptic curve discrete logarithm problem (ECDLP). ECDLP is defined as finding an integer k for given two points on the elliptic curve G and $k{\cdot}G$. The fastest algorithm to solve the ECDLP [78] requires approximately $\sqrt{p}$ steps for an elliptic curve on prime field $\mathbb{F}_p$. Choosing a 160-bit prime p in ECC achieves the same security level as a $1{,}024$-bit RSA.

Reduced storage and bandwidth requirements combined with efficient arithmetic operations make ECC suitable for resource-limited devices in an MCPS acquisition layer (see Fig. 1). ECC allows more sophisticated crypto-operations such as key sharing and encryption with data integrity, however, does not provide a mechanism for encrypted computation. Elliptic Curve Arithmetic is based on generalized discrete logarithm over elliptic curves. Elliptic curves over real numbers are defined as the set of points (x, y) that satisfying

$$y^2 = x^3 + a \cdot x + b,$$

where a and b are chosen such that $4 \cdot a^3 + 27 \cdot b^2 \neq 0$. Points on the elliptic curve together with a special point O (called

input : Message m, receiver's public key Q_B
output: U, C, tag
Set random $u \in \mathbb{Z}_p$
Compute $U = u \cdot G$
Compute $S(x_s, y_s) = u \cdot Q_B$
Generate (k_{ENC}, k_{MAC}) = KDF(x_s)
Encrypt C = ENC(m, k_{ENC})
Generate tag = HMAC(C, k_{MAC})

Fig. 4. ECIES encryption pseudo-code.

input : Ctxt C, tag, U, receiver's private key k_b
output: m
Compute $S(x_s, y_s) = U \cdot k_B$
Generate (k_{ENC}, k_{MAC}) = KDF(x_s)
Compute tag_B = HMAC(C, k_{MAC})
Check $tag_B == tag$
Decrypt m = DEC(C, k_{ENC})

Fig. 5. ECIES decryption pseudo-code.

point at infinity, which is not on the curve), form a group. Arithmetic operations over the elliptic curves (graphically described in Fig. 3) are:

Point addition adds two points $P(x_p, y_p)$ and $Q(x_q, y_q)$ of the group on the elliptic curve to find point $R(x_r, y_r)$, which is also on the elliptic curve.

Point doubling computes the double of point $P(x_p, y_p)$ as $2P$.

Point inversion calculates the inverse of point $P(x_p, y_p)$ as $-P(x_p, -y_p)$ such that $P + (-P) = O$.

Scalar multiplication of a point P by a scalar k is $k \cdot G = \underbrace{G + G + G + \cdots + G}_{k}$, which is computed by repeated point additions, similar to the repeated multiplications to compute modular exponentiation in RSA.

5.3 EC Diffie-Hellman Key Exchange

ECC is widely used for key exchange, similar to the Diffie-Hellman (DH) key-exchange protocol [56]. Regular DH can be converted to its ECC counterpart by replacing modular multiplications with point additions and modular exponentiations with repeated point additions. A shared session key between two parties (A and B) is established using ECCDH as follows: First, both parties agree on an elliptic curve on prime field $\mathbb{F}_p$ and a point P on the curve. Then, A and B select an integer k_A and k_B as their private key. Based on their private keys, they compute a point Q_A, Q_B on the curve by performing repeated additions. They exchange their computations without being able to discover each others' private key due to the hardness of the ECDLP problem. Finally, each party performs another point multiplication with his/her private key to find a common point Q_{AB} on the elliptic curve, which can be used as the shared secret key for a symmetric cipher.

5.4 EC Integrated Encryption Scheme (ECIES)

One of the standard ways to use ECC for public-key cryptography is the ECIES method [79], as shown in Fig. 4. ECIES provides data confidentiality by using a symmetric-key encryption such as AES. Integrity of the data is protected by message authentication code (MAC). Elliptic curves are employed to generate an encryption key (k_{ENC}) and a MAC key (k_{MAC}).

In ECIES, the sender generates a session key pair that will be used only for the current encryption. Session key is generated by choosing an element $u \in \mathbb{Z}_p^*$ and computing elliptic curve point $U = u \cdot G$. Based on the session key, a shared secret value is generated by using the receiver's public key as $S = u \cdot Q_B = u \cdot k_b \cdot G$. A standard Key Derivation Function (KDF) [80] inputs the shared secret value to generate two keys: k_{ENC} and k_{MAC}. Finally, message m is encrypted as $C = ENC(m, k_{ENC})$ using a symmetric key encryption and the key k_{ENC}. The tag of the ciphertext C is $tag = HMAC(C, k_{MAC})$, which is calculated using a keyed-hash message authentication code (HMAC). Finally the sender transfers C, tag and U (session key) to the receiver.

In ECIES decryption (Fig. 5), the receiver generates a shared secret $S = U \cdot k_b = u \cdot k_b \cdot G$ and k_{ENC} and k_{MAC} keys from S using KDF. Authenticity of C is verified by comparing the sender tag to $tag_B = HMAC(C, k_{MAC})$. If both of the tags match, the message is retrieved as $m = DEC(C, k_{ENC})$, otherwise C is discarded.

6 Secure Data Sharing Using Attribute Based Encryption (ABE)

In conventional public-key cryptography [76], [81], a user has two keys: The *public key* is shared with anyone that wants to send encrypted data to the user, while the *private key* is used to decrypt the received messages and is not shared with anyone. In many real-world healthcare scenarios more than one party may need to access the data, requiring duplicates of data by encrypting it using each party's public key. Attribute-based encryption [59], [60] is a public-key encryption that enables secure data sharing by multiple users. The data is encrypted using an access policy based on credentials (i.e., *attributes*). Only the users whose credentials satisfy the access policy can access data. The attributes can be the profession (e.g., Doctor, Nurse) or the department (e.g., Cardiology, Emergency) of a user. An access policy P can be defined as conjunctions, disjunctions and (k, n)-threshold gates of attributes such as

$$(\text{Doctor} \wedge \text{Cardiology}) \vee (\text{Nurse} \vee \text{Emergency})$$

which grants access to a Doctor from Cardiology OR a nurse OR an Emergency personnel. We provide details for two existing types of ABE: Ciphertext-Policy ABE (CP-ABE) and Key-Policy ABE (KP-ABE).

6.1 Ciphertext-Policy ABE (CP-ABE)

CP-ABE scheme provides a fine-grained access control to encrypted data similar to Role-Based Access control schemes [62]. Private key of a user is associated with user credentials. Ciphertexts specify an access policy and only users whose credentials satisfy the policy requirements can decrypt them. The data can be encrypted without the knowledge of users beforehand and the policy can be specified afterwards, enabling the future reassignment of keys. CP-ABE scheme consists of four algorithms [61]:

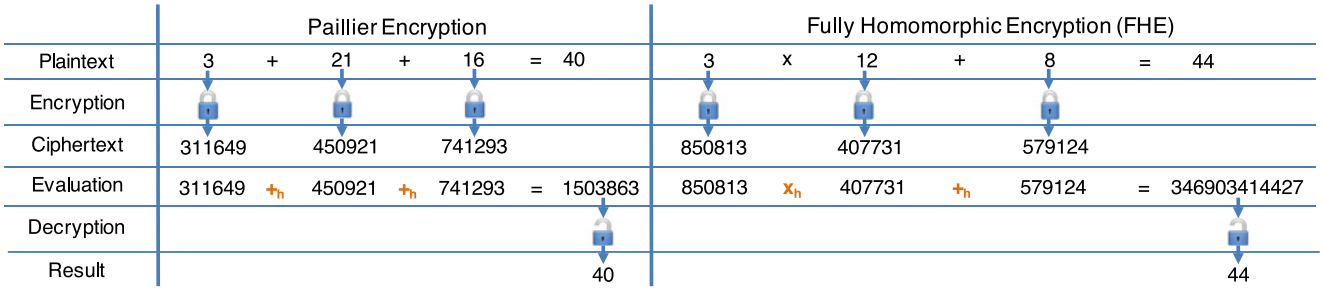

Fig. 6. Paillier and FHE homomorphic encryption schemes enable encrypted (secure) computation.

Setup. Generates a master key (k_M) and public parameters (Params). A bilinear group $\mathbb{G}_0$ of order prime p and a generator g is chosen. Two random exponents $\alpha, \beta \in \mathbb{Z}_p$ are selected to compute the parameters:

$$h = g^{\beta}, f = g^{1/\beta}, e(g,g)^{\alpha},$$

where $e(g,g)^{\alpha}$ is the bilinear mapping $\mathbb{G}_0 \rightarrow \mathbb{G}_T$.

Public parameters are then published as Params $= (\mathbb{G}_0, g, h, f, e(g,g)^{\alpha})$ and k_M is selected as $k_M = (\beta, g^{\alpha})$.

Key generation. Takes k_M as input and a set of attributes S specific to a user and generates a private key (k_{PRIV}) by choosing a random $r \in \mathbb{Z}_p$ and computing $D = g^{(\alpha+r)/\beta}$. For each attribute $s_j \in S$, a random $r_j \in \mathbb{Z}_p$ is selected to compute following:

$$D_j = g^{r} \cdot H(s_j)^{r_j}, \tilde{D}_j = g^{r_j},$$

where $H(s_j)$ is the hash of s_j that maps string s_j to a group element in $\mathbb{G}_0$. Private key k_{PRIV} is published as

$$k_{PRIV} = (D = g^{(\alpha+r)}, \forall s_j \in S : D_j, \tilde{D}_j).$$

Encryption. Takes Params, an access policy represented as a tree T defined over all possible attributes and message M to generate ciphertext C.

Decryption. Inputs Params, k_{PRIV}, and ciphertext C to generate M. Decryption will be successful if user's k_{PRIV} satisfies the access structure embedded in C.

6.2 Key-Policy ABE (KP-ABE)

In KP-ABE [59], [60] the access policy is encoded into the users' private key and a ciphertext is labeled with a set of attributes. KP-ABE schemes place the access policy on the private key of the users and the attributes are associated with the ciphertexts. A recently proposed ABE scheme [82], which is based on KP-ABE, is proposed as a *lightweight* ABE solution to provide security for resource constrained devices such as Internet-of-Things (IoTs). This scheme is based on ECC instead of bilinear pairings. Bilinear pairings are very expensive for resource constrained devices and lightweight ABE scheme improves both communication and computation overhead by using ECC. Specifically, [82] uses ECIES [79] to provide both data confidentiality and data integrity. This scheme is composed of the following four steps:

Setup. In this step, a central attribute authority who is responsible for key generation, generates public parameters (Params) and master key (k_M). The setup is based on the the universal set of attributes U. For each attribute i in U, a point on elliptic curve P_i is generated by choosing a random $r_i \in \mathbb{Z}_q^*$ and then computing $P_i = r_i \cdot G$. Then a random $r \in \mathbb{Z}_q^*$ is chosen as k_M and master public key is set to $PK = r \cdot G$. Finally Params is published as the set Params $= \{PK, P_1, P_2, \ldots, P_{|U|}\}$.

Key generation. Takes k_M and access policy P and generates decryption key (k_{DEC}).

Encryption. Takes input attribute set S, message M and public key parameters Params to generate the corresponding ciphertext. For each attribute i in S, $C_i = r_i \cdot P_i$ is computed by choosing random $r_i \in \mathbb{Z}_q^*$. Encryption of the M is done by using secret key for the symmetric-key cryptography generated by ECIES to compute C. Finally the MAC of the message is computed as $MAC_M = HMAC(M, k_{MAC})$, where k_{MAC} is the y-coordinate of the elliptic curve $Q = r \cdot PK$. Ciphertext is published as the set $\{S, C, MAC_M, C_1, C_2, \ldots C_{|S|}\}$

Decryption. Takes ciphertext set $\{S, C, MAC_M, C_1, C_2, \ldots C_{|S|}\}$ encrypted using the attribute set S and uses decryption key k_{DEC} for the policy P to decrypt message M.

7 Secure Computation Using Homomorphic Encryption

Conventional encryption schemes are extremely lightweight, but do not allow computations on encrypted data. Homomorphic encryption schemes enable computation of meaningful operations on encrypted data without observing the actual data. By using HE, both storage and computation can be outsourced to public cloud operators, eliminating data privacy concerns in case of medical cloud computing. An HE scheme transforms into a Fully Homomorphic Encryption scheme if it can evaluate *arbitrary* functions. To evaluate arbitrary functions over ciphertexts, FHE schemes need to perform both homomorphic addition and homomorphic multiplication, which translates to addition and multiplication of the plaintext messages, respectively [83].

First plausible FHE scheme was proposed by Gentry in 2009. Schemes proposed before [84], [85], [86], [87] were *partially* homomorphic and they could perform only homomorphic addition or homomorphic multiplication. Fig. 6 shows the difference between the partially homomorphic Paillier scheme [86] and an FHE scheme. The Paillier scheme (left) is only additively-homomorphic, thereby allowing only addition operations on ciphertexts. FHE (right) allows both homomorphic additions and multiplications, thus permitting arbitrarily complex computations. Currently, FHE schemes are not practical since they require heavy computational and storage resources [88]. Improving the performance of FHE

remains an active research area. In this section, we will provide the details of Paillier and a recent FHE implementation called the Brakerski-Gentry-Vaikuntanathan (BGV) scheme [89].

7.1 Paillier Encryption Scheme

Paillier Encryption scheme [86] is a public-key cryptosytem that is additively-homomorphic. Operations on ciphertexts encrypted with Paillier scheme result in additions of messages without observing them. Due to its additive homomorphism, Paillier scheme is widely used in many practical applications [90]. Security of the Paillier scheme is based on difficulty of finding the nth residue of composite numbers: Given z and n^2, where $n = p \cdot q$ is a composite number, it is hard to find y that observes the following relationship

$$z = y^n \mod n^2.$$

Paillier encryption scheme consists of five algorithms:

Setup. Selects two large primes p and q randomly and independently to generate composite number $n = p \cdot q$.

Key generation. Calculates $\lambda = lcm(p-1, q-1)$ which is least common multiplier of $p-1$ and $q-1$. Random $g \in \mathbb{Z}^*_{n^2}$, which is a generator for the $\mathbb{Z}^*_{n^2}$, is selected and its multiplicative inverse$\bmod n$ is calculated as

$$\mu = (L(g^\lambda \mod n^2))^{-1} \bmod n,$$

where L is the function that computes $L(k) = (k-1)/n$. Finally, public key is selected as $k_{PUB} = (n, g)$ and private is selected as $k_{PRIV} = (\lambda, \mu)$.

Encryption. Encrypts the message m with random $r \in \mathbb{Z}^*_{n^2}$ to ciphertext c using k_{PUB} as follows:

$$c = g^m \cdot r^n \mod n^2.$$

Decryption. Decrypts the ciphertext c to the message m using k_{PRIV} as follows

$$m = L(c^\lambda \mod n^2) \cdot \mu \mod n.$$

Homomorphic addition. Addition of the plaintexts m_1 and m_2 $(m_1 + m_2 \bmod n)$ corresponds to the multiplication of their ciphertexts (c_1 and c_2) as detailed below:

$$\begin{aligned} c_1 &= g^{m_1} \cdot r_1^n && \bmod n^2 \\ c_2 &= g^{m_2} \cdot r_2^n && \bmod n^2 \\ c_3 = c_1 \cdot c_2 &= g^{(m_1 + m_2 \bmod n)} \cdot (r_1 \cdot r_2)^n && \bmod n^2. \end{aligned}$$

7.2 BGV Scheme

Several FHE implementations have been proposed to date [89], [91], [92], [93], [94] to improve performance of Gentry's initial FHE scheme [65]. Currently, the BGV scheme [89] is one of the most promising candidates for a practical FHE scheme, incorporating many optimizations. The expensive bootstrapping operation [65] is avoided by a variant of FHE called *leveled* FHE that employs a better noise management technique called *modulus-switching*. Ciphertexts encrypt multiple messages to reduce storage overhead and execute homomorphic operations in parallel similar to SIMD-fashion.

7.2.1 Leveled FHE

Leveled FHE scheme allows performing cascaded homomorphic multiplications ($\times_h$) without causing decryption errors. Right after encryption, each ciphertext is set to a level L and L is reduced by one after each $\times_h$ until it reaches $L = 1$, at which point further $\times_h$ operations can cause decryption errors. While leveled FHE provides better performance, it requires the computation of L beforehand [95].

7.2.2 Message Space

In the BGV scheme, plaintexts are represented as an element in polynomial ring $GF(p^d)$, where p is a prime number that defines the range of polynomial coefficients and d is the degree of the polynomials. Homomorphic addition and multiplication of ciphertexts correspond to addition and multiplication of plaintexts in the $GF(p^d)$, respectively. When $GF(2)$ is selected as the polynomial ring (i.e., $p = 2,\ d = 1$), the messages are represented as bits; in $GF(2)$, homomorphic addition and multiplication of ciphertexts translate to XOR, AND operations on the plaintexts, respectively, enabling the computation of arbitrary functions by representing them as a binary circuit using a combination of XOR, AND gates.

7.2.3 Message Packing

Representing plaintexts as polynomial rings in $GF(p^d)$ allows using Chinese Remainder Theorem to partition plaintexts into ℓ independent "slots" [96]. Multiple messages can be packed into the plaintext by assigning a message to each plaintext slot. For $GF(2)$, each slot represents single bit and messages can be packed by concatenating their bitwise representation.

7.2.4 SIMD Operations

Packing enables the SIMD execution of the same operation in parallel for ℓ-slots. BGV offers SIMD execution of homomorphic operations for performance improvement. We use four orthogonal operations available in BGV:

Homomorphic addition ($+_h$). Corresponds to a slot-wise XOR of plaintexts in $GF(2)$. $+_h$ does not affect the level L of the BGV scheme.

Homomorphic multiplication ($\times_h$). Corresponds to a slot-wise AND operation of plaintexts in $GF(2)$. $\times_h$ operation reduces the level L of the ciphertext by one. Therefore, the depth of multiplications will determine the required level of the BGV scheme.

Rotate ($>>>_h$, $<<<_h$). Provides rotation of slots similar to a barrel shifter and slots will wrap around based on the rotation direction, thereby potentially garbling the data contained in the neighboring slots. This will be corrected using Select operations.

Select (sel_{mask}). Chooses between the slots of two plaintexts based on an unencrypted selection mask vector. Select operation can be used to mask out the bits that are diffused from other messages after a Rotate.

8 Secure Computation Case Study

In this section, we provide a secure computation implementation case study for a simple medical application. Computations in this application are performed on encrypted

medical data in a public cloud using the Paillier and BGV homomorphic encryption schemes.

8.1 Medical Application

Our target MCPS is a remote patient health monitoring system [67] that transmits patient ECG signals from the patient's house (Layer 1 in Fig. 1) into the cloud (Layer 3). Patient medical data is assumed to be encrypted using one of the homomorphic encryption schemes to provide data privacy during transmission. Since both of these HE schemes are very resource-intensive, as discussed in Section 7, the intermediate pre-processing layer (Layer 2) is assumed to aid the HE computationally. From the encrypted ECG recordings, we will provide certain statistics and detection results to the doctor (Layer 4) as our case study application.

The statistics we will provide are the average heart rate of a patient. The detection results we will provide are for the "detection of the long-QT syndrome," which is a cardiac condition that can cause fatalities [7], [67]. Quantitatively, the goal of this application is to continuously monitor the "QT_c" metric of a patient's heartbeats and alert the doctor when QT_c exceeds a clinical threshold. Typically, QT_c is between 300-600 ms and $QT_c > 500$ ms is considered to be too long (i.e., long QT syndrome). The QT_c metric is defined as the *corrected* QT, which is calculated from the QT and RR intervals in an ECG recording. One of the most common methods in computing QT_c from QT and RR is to use Bazett's formula [97] : $QT_c = \frac{QT}{\sqrt{RR}}$.

8.2 Computations Using Paillier

Paillier scheme is an additive homomorphic encryption, therefore we will use Paillier for only the average heart rate computation. Calculating the average heart rate using Paillier involves accumulating the encrypted messages by using its additive homomorphic property. We note that to compute the average, the accumulated value needs to be divided by number of ECG samples. However, this division will be difficult to implement using Paillier. Therefore, we will return two ciphertexts: 1) accumulated sum and 2) number of ECG samples; the receiver can decrypt both ciphertexts and compute the actual average. Accumulating N ciphertexts (c_i) using Paillier is performed as follows:

$$c_{sum} = \prod_{i=0}^{i=N} c_i = \prod_{i=0}^{i=N} g^{m_i} \cdot r_i^n c_{sum}$$
$$= (g^{\sum_{i=0}^{i=N} m_i \bmod n}) \cdot \left(\prod_{i=0}^{i=N} r_i\right)^n \bmod n^2,$$

where decryption of c_{sum} will yield the sum of N messages (i.e., $\sum_{i=0}^{i=N} m_i \bmod n$).

8.3 Computations Using BGV

We use the *leveled* BGV scheme to implement LQTS detection and average heart rate calculation. We determine the required BGV level L by determining the multiplication-depth of each computation. As we will show later, the multiplication depth (the chain of cascaded multiplications) depends on two variables: bit-length of messages (k) and number of ciphertexts (N). BGV ciphertexts pack multiple k-bit messages based on number of plaintext slots, which varies based on level L.

input : Ciphertexts $\boldsymbol{X}$ and $\boldsymbol{Y}$
output: Ciphertext $\boldsymbol{R} = \boldsymbol{X} >_h \boldsymbol{Y}$
$\boldsymbol{E} = \boldsymbol{X} +_h \boldsymbol{Y} +_h \boldsymbol{1}$
$\boldsymbol{M} = \boldsymbol{E}$
for $i = 1$ **to** k **do**
  $\boldsymbol{T} = (\boldsymbol{M} >>>_h i)\ sel_{mask}\ \boldsymbol{1}$
  $\boldsymbol{M} = \boldsymbol{M} \times_h \boldsymbol{T}$
  $i = i \cdot 2$
$\boldsymbol{Q} = (\boldsymbol{Y} +_h \boldsymbol{1}) \times_h \boldsymbol{X}$
$\boldsymbol{R} = \boldsymbol{M} \times_h \boldsymbol{Q}$

Fig. 7. BGV implementation of comparison.

8.3.1 Long QT Syndrome (LQTS) Detection

LQTS detection requires the following comparison that we discussed in Section 8.1: $\frac{QT}{\sqrt{RR}} > th$, where th is the 500 ms clinical threshold. We rewrite the formula as $QT_h > RR_h$, which avoids the square-root, therefore making it more suitable for a BGV implementation. In this rearrangement, $QT_h = QT^2$ and $RR_h = RR \cdot th^2$, which reduces the original computation to a single comparison operation. In other words, the acquisition layer of the MCPS (Layer 1 in Fig. 1) transmits $RR_h = RR \cdot th^2$ and $QT_h = QT^2$ rather than RR and QT.

To implement homomorphic comparison, we start out by designing a 4-bit comparator that computes:

$$X > Y = (x_3\overline{y_3} \oplus x_2\overline{y_2}e_3 \oplus x_1\overline{y_1}e_3e_2 \oplus x_0\overline{y_0}e_3e_2e_1),$$

where X and Y are the two 4-bit *plaintext* values that are being compared, x_i is the value of bit i of X, $\overline{y_i}$ is the inverse of bit i of Y, and e_i denotes the bitwise equality $(x_i == y_i)$. To perform this comparison homomorphically, we will use the notation $\boldsymbol{X}$ and $\boldsymbol{Y}$ to denote the *ciphertexts* that correspond to the plaintexts X and Y, respectively. Homomorphic comparison can be performed by evaluating

$$\boldsymbol{X} > \boldsymbol{Y} = (\boldsymbol{X} \times_h \boldsymbol{Y'} \times_h \boldsymbol{M}),$$

where $\boldsymbol{Y'}$, $\boldsymbol{M}$ encrypt $\overline{y_i}$, $(1\ e_3\ e_3e_2\ e_3e_2e_1)$, respectively.

Fig. 7 presents the generalized k-bit BGV implementation of this homomorphic comparison. Ciphertexts X and Y encrypt QT^2 and $RR \cdot th^2$, respectively. Comparison requires $\log_2 k + 1$ depth for ciphertexts packing k-bit messages. Specifically, $\log_2 k$ depth is needed to compute mask $\boldsymbol{M}$ from $\boldsymbol{E}$, followed by single multiplication at the end. Once the comparison is finished, results of the comparisons needs to be aggregated to extend the detection results over multiple ECG samples. Aggregation can be performed using the OR operation as

$$\boldsymbol{X} \vee_h \boldsymbol{Y} = \boldsymbol{X} +_h \boldsymbol{Y} +_h (\boldsymbol{X} \times_h \boldsymbol{Y}),$$

which has a multiplication depth of 1. To aggregate N comparison results, the OR operation can be applied in a binary tree fashion, requiring $\lceil \log_2 N \rceil$ depth. Therefore, the minimum required level for LQTS detection is $L > (\log_2 k + 1 + \lceil \log_2 N \rceil)$. We note that after each rotation operation ($>>>_h$), a selection operation is applied to mask bits that are diffused from neighboring messages.

input : Ciphertexts $\boldsymbol{X}$ and $\boldsymbol{Y}$
output: Ciphertext $\boldsymbol{S}$
$\boldsymbol{G} = \boldsymbol{X} \times_h \boldsymbol{Y}$
$\boldsymbol{P} = \boldsymbol{X} +_h \boldsymbol{Y}$
for $i = 1$ **to** $num\_stages + 1$ **do**
  $\boldsymbol{G''} = \boldsymbol{G}$
  $\boldsymbol{P''} = \boldsymbol{P}$
  $\boldsymbol{G'} = (\boldsymbol{G} <<<_h i)\ sel_{mask}\ \boldsymbol{0}$
  $\boldsymbol{P'} = (\boldsymbol{P} <<<_h i)\ sel_{mask}\ \boldsymbol{1}$
  $\boldsymbol{P} = \boldsymbol{P'} \times_h \boldsymbol{P''}$
  $\boldsymbol{G'} = \boldsymbol{G'} \times_h \boldsymbol{P''}$
  $\boldsymbol{G} = \boldsymbol{G'} \vee_h \boldsymbol{G''}$
  $i = i \cdot 2$
$\boldsymbol{S} = \boldsymbol{P} +_h ((\boldsymbol{G} <<<_h 1)\ sel_{mask}\ \boldsymbol{0})$

Fig. 8. BGV implementation of KSA.

8.3.2 Average Heart Rate (HR)

Average HR is computed by accumulating N ciphertexts that encrypt multiple k-bit RR interval values. We use a combination of Carry Save Adder (CSA) and Kogge-Stone Adder (KSA) to achieve low multiplication-depth. Specifically, we use CSA adders to compress N ciphertexts down to two ciphertexts and add remaining ciphertexts using a KSA adder to compute final sum.

CSA adders operate on three variables X, Y, Z to generate carry $C = (XY \vee XZ \vee YZ) << 1$ and sum $S = (X \oplus Y \oplus Z)$. The multiplication depth is determined by the carry computation and is equal to 3 due to the multiplications and the OR operation. This depth can be reduced to one by replacing OR with XOR within a CSA adder [67]. CSA adders can be combined in a tree fashion, to compress N ciphertexts to two. The depth d of the CSA compression tree is equal to [98]:

$$\left\lceil \frac{\log_2(N/2)}{\log_2(3/2)} \right\rceil + 1 \leq d.$$

After compressing N ciphertexts down to two, we use KSA to add the final two ciphertexts. KSA is a parallel-prefix adder that performs operations in logarithmic-depth. Fig. 8 shows the implementation of KSA [99] using BGV. KSA starts by computing Generate (G) and Propagate (P) values from inputs X and Y, which has a depth of 1. G and P are updated in $\log_2 k$ stages, where each stage has a depth of 2 for computing G (1 for $\times_h$, 1 for $\vee_h$). Therefore, KSA requires depth of $2 \log_2 k + 1$. Therefore, minimum required level L for accumulating N ciphertext that packs k-bit messages is $L > \left(\left\lceil \frac{\log_2(N/2)}{\log_2(3/2)} \right\rceil + 1\right) + (2 \log_2 k + 1)$.

9 Experimental Setup

We run our experiments on an Intel Xeon W3565 workstation (4 cores, 8 threads) with 24 GB RAM, running 64-bit Ubuntu 15.04. Our results are based on single-threaded execution times, since most of the existing libraries do not have an efficient multi-threaded implementations. We use two open-source libraries:

Charm library [100] provides a high-level framework for designing cryptosystems. Charm is based on Python, but compute intensive operations are implemented in C and has comparable performance to native C implementations. We use Charm for benchmarking the performance of conventional and ABE encryption schemes.

TABLE 1
Parameter Selection for 128-bit Security

ECIES [80]	Elliptic curve: $\mathbb{F}_p$ with $p = 256$-bit prime Symmetric-key encryption: AES-128 MAC: HMAC-SHA1 (160-bit)
CP-ABE [61]	Bilinear Pairing: Supersingular curve over $\mathbb{F}_p$, $p = 1,536$-bit prime Access Policy: 10 attributes
KP-ABE [82]	Elliptic curve: $\mathbb{F}_p$ with $p = 256$-bit number p Symmetric-Key encryption: AES-128 MAC: HMAC-SHA1 (160-bit) Access Policy: 10 attributes
Paillier [86]	$p, q = 3{,}072$-bit prime

HElib library [101] is a state-of-the-art FHE library that implements the BGV scheme [89]. Medical applications presented in Section 8.3 are implemented by using the primitives in HElib that were listed in Section 7.2.4.

9.1 Data Set

To simulate the acquired patient data in the acquisition layer of the MCPS (Layer 1 in Fig. 1), we use the THEW database [102], [103]. THEW is a large corpus of 24-hour anonymized Holter ECG recordings of real patients, sampled at the rate of 1,000 Hz. The ECG data represents summary of the each heart beat and provides information of QT and RR intervals in terms of number of samples acquired (toc). 24-hour ECG data contains 87,896 samples and each toc value is represented as 16-bit unsigned integer.

9.2 Security Level of Encryption Schemes

We use 128-bit security for encrypting medical data, which is the recommended security level for federal government data by NIST [104]. Table 1 presents the parameter selection of encryption schemes based on a 128-bit security level. For BGV, we use the analysis provided in [105] for setting the security parameters.

9.3 BGV Setup

Runtime and storage requirements of BGV are tightly related to the BGV level L, which depends on the bit-length of the messages (k) packed in plaintexts and the number of ciphertexts required for computation (N) as described in Section 8.3. We set the level L to the lowest value that allows the execution of application without causing decryption errors. We use different k values for LQTS detection and Average HR. Since LQTS detection performs comparison operation, we choose $k = 16$, which is the bit-length of the toc values in the dataset. For the Average HR, we choose $k = 32$ by padding toc values with 0's to prevent overflow during accumulation. The number of ciphertexts, required to encrypt the dataset (N) depends on the number of plaintext slots (ℓ). Table 2 presents the ℓ options for different BGV levels. Each ciphertext can pack $\lfloor \ell/k \rfloor$ messages that enables SIMD-like parallel homomorphic operations.

10 Evaluation

In this section, we compare the performance of different encryption schemes based on their encryption/decryption

TABLE 2
# of Plaintext Slots at Different BGV Levels

BGV Level L	# of slots (ℓ)
$1 \leq L < 12$	630
$12 \leq L < 22$	682
$22 \leq L < 68$	1,285

times, evaluation times (only for homomorphic schemes) and ciphertext sizes.

10.1 Comparison of the Encryption Schemes

Table 3 summarizes the secure storage, secure computation and secure data sharing capabilities of the encryption schemes presented in Sections 5, 6 and 7. Conventional encryption schemes cannot provide secure computation, unless medical data is stored in a trusted private cloud (e.g., the data center of the hospital), where decryption is possible without violating the privacy.

Secure data sharing is limited to the users who have the secret key of AES and the private key of ECIES. ABE cannot perform computations on encrypted data, but provides fine-grained secure data sharing capability in a public cloud setting.

Homomorphic encryption schemes provide secure computation in a public cloud: Paillier only performs homomorphic addition, thereby allowing a limited set of operations, while BGV enables arbitrary computations, but requires more resources than Paillier. Both schemes limit data sharing to the users who have the private key.

10.2 Data Privacy in Acquisition, Preprocessing

Acquisition devices, such as the sensors in BANs, have strict resource requirements. Therefore the communication between BAN sensors (Layer 1 in Fig. 1) and BAN-to-Cloudlets (Layer 1 to Layer 2) must be secured using lightweight encryption schemes. We will use AES-128 for encrypting medical data captured by the sensors in a BAN. Symmetric-key of AES-128 is shared using the Elliptic Curve Diffie-Hellman key-exchange.

ECDH is used once to generate the same secret key between communicating parties. During the key exchange, two parties exchange a single ciphertext that represents a point in the elliptic curve. This ciphertext contains the (x, y) coordinates, each represented as a p-bit integer in $\mathbb{F}_p$. A 256-bit $\mathbb{F}_p$ is selected for the elliptic curve to achieve 128-bit security. Therefore, the exchanged ciphertext has a size of $2 \cdot (256/8) = 64$ B. Both parties need to perform elliptic curve point multiplications to a generate secret key for AES.

TABLE 3
Comparison of Different Encryption Schemes

Scheme	Encryption	Computation	Data Sharing
Conventional	AES	NA	Limited
	ECIES	NA	Limited
Attribute-based	KP-ABE	NA	Fine-Grained
	CP-ABE	NA	Fine-Grained
Homomorphic	Paillier	Partial	Limited
	BGV	Full	Limited

TABLE 4
Requirements of Encrypting 24-hr ECG Data Using Different Encryption Schemes

Encryption	Enc. (sec)	Dec. (sec)	Ctxt (MB)
ECIES	40.3	38.7	8.4
KP-ABE	439.5	615.3	56.7
CP-ABE	58 K	32.5 K	708.1
Paillier	49.2 K	48.3 K	128.8
BGV	3,956	1,868	44.4 K

Our Charm library simulation for this shows a total runtime of 0.23 ms.

Once the secret key is generated, medical data can be securely transferred by using AES-128. Our Charm library simulation for AES-128 encryption and decryption times are 0.2 and 0.23 μs, respectively. These are the performance results for the AES-CBC mode of operation that is used in the OpenSSL library implementation.

The AES-GCM mode can be used to provide both confidentiality and integrity. AES-GCM mode can be implemented efficiently by using the techniques introduced in Section 5.1.2. By using the Intel AES-NI instruction set extensions, the optimized code that is published on Intel's website [75] resulted in AES-GCM encryption and decryption run times of 0.06 μs per 128-bit block.

The performance of AES-GCM mode can be further improved by using ASIC/FPGA implementations. A fully pipelined ASIC implementation of AES-GCM is presented in [74], which can run at 429.2 MHz and perform encryption/decryption in ≈ 2.3 ns per block.

10.3 Secure Storage

Once the medical data is captured, it is transferred to a more computationally capable device such as a smartphone or a cloudlet. This data can be encrypted using different encryption schemes based on the desired capability (i.e., sharing, computation). For example, before transferring the data to a public cloud, AES-128 can be used at the acquisition layer, which can be converted to FHE in the cloud using AES-to-FHE conversion schemes [105]. Table 4 lists execution times and storage requirements for ciphertexts for different encryption schemes. Encryption (Enc.) and Decryption (Dec.) columns list the required time to encrypt/decrypt 24-hr ECG data, consisting of 87,896 samples as described in Section 9.1. Ctxt column shows the space required for storing encrypted data.

10.3.1 ECIES

For ECIES, we select AES-128 for symmetric-key cryptography and HMAC-SHA1 for HMAC. The ciphertext generated by the ECIES encryption has three components: a point on the elliptic curve, an AES-128 encrypted message and a *tag* generated by HMAC-SHA1. A point on the elliptic curve has two 256-bit coordinates, the AES-128 encrypted message is 128-bits and the *tag* from HMAC-SHA1 is 160-bits. Therefore total ciphertext size is equal to $(2 \cdot 256 + 128 + 160)/8 = 100$ B. Encryption and decryption operations using ECIES require 0.46 and 0.44 ms, respectively based on Charm results.

TABLE 5
BGV Results for Computing the Average Heart Rate and LQTS Detection L is the BGV Level and N is the Number of Ciphertexts Required to Store the Encrypted ECG Samples for a Given Monitoring Interval

	Monitor. Interval	N	L	Enc. (sec)	Dec. (sec)	Ctxt (MB)	Exec. (min)
	1 min	3	14	0.20	0.19	3.4	0.4
Avg	15 min	44	21	0.29	0.29	4.8	2.8
HR	1 hr	92	23	1.36	0.63	15.0	16.5
(k = 32)	3 hr	275	26	1.59	0.73	17.6	56.1
	24 hr	2,198	31	1.80	0.85	20.2	502
	1 min	2	7	0.05	0.01	0.9	0.1
	15 min	24	11	0.08	0.03	1.3	2.5
LQTS	1 hr	88	13	0.18	0.15	2.9	32.7
(k = 16)	3 hr	262	15	0.21	0.19	3.4	117
	24 hr	2,093	18	0.26	0.25	4.3	1,165

10.3.2 Attribute-Based Encryption (ABE)

For ABE, we consider two candidates: CP-ABE scheme from [61] and the recent KP-ABE scheme from [82]. We evaluate both schemes based on an access policy P, consisting of 10 attributes.

A ciphertext in the CP-ABE scheme consists of the set C', C, C_y, C'_y, where C_y and C'_y are generated for each attribute in the policy P. Each element in the ciphertext is a point on the elliptic curve, which is represented as two coordinates in the 1,536-bit prime field $\mathbb{F}_p$. Therefore, the total size of a ciphertext in the CP-ABE scheme is $(2 \cdot (1 + 1 + 10 + 10) \cdot 1{,}536)/8 = 8{,}448$ B. Encryption and decryption operations are performed in 660 and 700 ms, respectively based on Charm results.

In the KP-ABE scheme, a ciphertext consists of the set C, *tag*, and C_i, where a different C_i is generated for each attribute in the policy P. C is the 128-bit ciphertext, encrypted using AES-128. The *tag* is generated using HMAC-SHA1 and 160-bits. Each C_i is a point on the elliptic curve, which is represented as two coordinates in the 256-bit prime field $\mathbb{F}_p$. The total size of a ciphertext in the KP-ABE scheme is $(128 + 160 + (2 \cdot 10 \cdot 256))/8 = 676$ B. Encryption and decryption operations are performed in 5 and 7 ms, respectively based on Charm results.

The KP-ABE scheme is more efficient and requires less storage, compared to CP-ABE. This is a result of using elliptic curves to generate keys for efficient AES and HMAC operations instead of bilinear pairings found in CP-ABE. CP-ABE can provide an easy implementation if the hospital is already using a Role-Based Access System.

10.3.3 Paillier

Ciphertexts in Paillier are represented as 12,288-bits integers in the prime field $\mathbb{F}_p$. This is due to the fact that ciphertexts are integers in$\bmod n^2$ where $n = p \cdot q$. We choose the security parameter as 128-bits, which requires 3,072-bit primes for p and q to be selected. Encryption and decryption operations are performed in 560 and 550 ms, respectively according to Charm results.

10.3.4 BGV

In the BGV scheme, ciphertext sizes depend on the BGV level L. The resource requirements reported in Table 4 are based on $L = 31$ for computing the 24-hour average heart rate. A 20.2 MB ciphertext can encrypt 1,285 plaintext slots or 40 32-bit messages ($\lfloor \frac{1{,}285}{32} \rfloor = 40$). Encryption and decryption operations are performed in 1.8 and 0.85 sec, respectively based on HElib results.

TABLE 6
Execution Time and Ciphertext Size Comparison of Different Encryption Schemes, Normalized to AES. The Evaluation Time of the Homomorphic Schemes are Normalized to Paillier

Scheme	Implement. Source	Enc. time	Dec. time	Ctxt size	Eval. time
AES	ASIC [74]	0.01	0.01	1	
AES	Intel [75]	0.3	0.3	1	
AES	Charm [100]	**1**	**1**	**1**	
ECIES	Charm	2.3 K	1.9 K	6.3	
KP-ABE	Charm	25 K	30.4 K	42	
CP-ABE	Charm	3.3 M	1.6 M	528	
Paillier	Charm	2.8 M	2.4 M	96	**1**
BGV	HElib [101]	9 M	3.6 M	1.3 M	3.1 K

10.4 Secure Computation

We evaluate the secure computation options for an MCPS using the Paillier and BGV schemes.

10.4.1 Computation Using the Paillier Scheme

Average heart rate computation using the Paillier scheme requires performing homomorphic addition of multiple ciphertexts. Single homomorphic addition requires 0.11 ms based on Charm results. Therefore, computing the average heart rate for the 24-hour ECG data takes 9.7 seconds, which involves the homomorphic addition of 87,896 ciphertexts.

10.4.2 Computations Using the BGV Scheme

Table 5 presents the HElib results of LQTS detection and Average heart rate for the 24-hour ECG data, containing 87,896 toc values. Rows of the table represent the partitioning of the data used in computations. For example, LQTS detection using 1-min ECG interval checks for the LQTS event every minute, while 24-hour ECG interval operates on all 24-hour data and returns a single result. ECG intervals can be adjusted to reflect the condition of a patient; a patient in critical condition might require monitoring results every minute, while a healthy patient just needs one result per day.

For each application, we determined L using the guidelines in Section 8.3. Both the LQTS detection and average heart rate computation require higher L for longer ECG intervals, since longer intervals require an increased number of ciphertexts (N), thereby increasing both the execution time and the required storage space. However, longer ECG intervals require less network traffic by producing aggregated results over many ciphertexts.

10.5 Summary of Results

Table 6 summarizes our results. Encryption/Decryption times and ciphertext sizes are normalized to AES for every scheme. Evaluation times are normalized to Paillier for the homomorphic schemes. Using the Charm library [100], we

show that multiple orders-of-magnitude computational time and storage space penalty must be incurred to enable secure data sharing and secure computation. Using the Charm [100] and the HElib [101] libraries, we demonstrate the performance of the homomorphic schemes in the last two lines of Table 6; Paillier requires four orders-of-magnitude lower storage for ciphertexts, but only allows restricted set of secure computations and performs evaluations 3,100× faster than BGV. ASIC [74] and Intel AES-NI optimized [75] versions of the AES run 1–2 orders-of-magnitude faster than the generic C software implementation [100].

11 Conclusions

In this paper, we define a Medical Cyber Physical System as a four-layer system consisting of data acquisition, data aggregation, cloud, and action layers. We survey conventional and emerging encryption schemes based on their ability to provide secure storage, secure data sharing, and secure computation. Conventional encryptions such as AES and ECIES do not allow any operation other than secure storage, while the emerging Attribute-Based Encryption allows secure data sharing based on the credentials of the sharing parties. Alternatively, secure computation on encrypted data is only feasible using the emerging Fully Homomorphic Encryption schemes.

Through our experimental analysis, we show that due to the substantial differences among these algorithms in terms of storage and computational requirements, it is not possible to provide a single encryption/decryption scheme that is superior to all of the others. Therefore, we analyze six different encryption schemes based on four metrics: i) encryption time, ii) decryption time, iii) ciphertext size, and iv) evaluation time. While the first two metrics provide information about the computational intensity of the encryption scheme, the third metric shows the expansion of the amount of data in its encrypted form, determining its storage and transmission characteristics. Clearly, the fourth metric is only relevant to the techniques that provide computation in encrypted format, such as FHE and Paillier.

Our first experimental analysis shows that the encryption and decryption times under a given encryption scheme are comparable (e.g., within $\pm 20\%$ for ECIECS encryption versus decryption), although the variation among different schemes is significant. For example, normalizing to AES, attribute-based encryption schemes (KP-ABE and CP-ABE) are $25{,}000\times$ and $3.3\,\mathrm{M}\times$ slower, respectively, while homomorphic encryption schemes (Paillier and FHE) are $2.8\,\mathrm{M}\times$ and $9\,\mathrm{M}\times$ slower. These results underline the vast computational penalty that must be paid to enable secure sharing and secure computation.

Our second analysis focuses on determining the amount of storage required for the encrypted version (i.e., ciphertext) of a given plaintext. Normalizing to AES, ECIES requires $6.3\times$ more space, while attribute-based encryption schemes (KP-ABE and CP-ABE) still show a significant disadvantage, requiring $42\times$ and $528\times$ more storage for the encrypted data. On the other hand, homomorphic encryption schemes (Paillier and FHE) exhibit a $96\times$ and $1.3\,\mathrm{M}\times$ storage expansion. Consequently, these storage disadvantages translate to vast communication overheads when transmitting encrypted data.

Our final analysis compares the two homomorphic encryption schemes that can perform secure computation on ciphertexts. We conclude that while the encryption and decryption of the Paillier scheme are almost as slow as BGV, evaluation of a ciphertext using Paillier is $3,100\times$ faster, however the evaluation operations that are permitted by Paillier are substantially more restrictive (only additions can be performed on ciphertext).

Based on these analyses, we conclude that a one-size-fits-all encryption scheme simply does not exist for designing an MCPS. Among the six different schemes studied in this paper, AES is the clear winner in terms of computation and storage requirements, while the other five suffer substantial storage and computation overheads. Therefore, to construct exciting new MCPSs that can take advantage of these emerging encryption schemes, their significant speed-up is necessary either through theoretical advancements or by utilizing GPUs, ASICs, or FPGA-based hardware accelerators.

Acknowledgments

This work is supported in part by the US National Science Foundation grant CNS-1239423. The authors thank Prof. Muthuramakrishnan Venkitasubramaniam and anonynous reviewers for their insightful discussions.

References

[1] (2015). FitBit Inc. flex: Wireless activity + sleep wristband [Online]. Available: https://www.fitbit.com/flex
[2] (2015). Apple Inc. Apple watch [Online]. Available: https://www.apple.com/watch/
[3] S. Xu, Y. Zhang, L. Jia, K. E. Mathewson, K.-I. Jang, J. Kim, H. Fu, X. Huang, P. Chava, R.Wang, S. Bhole, L.Wang, Y. J. Na, Y. Guan, M. Flavin, Z. Han, Y. Huang, and J. A. Rogers, "Soft microfluidic assemblies of sensors, circuits, and radios for the skin," *Science*, vol. 344, no. 6179, pp. 70–74, 2014.
[4] D. Kim, R. Ghaffari, N. Lu, and J. A. Rogers, "Flexible and stretchable electronics for biointegrated devices," *Annu. Rev. Biomed. Eng.*, vol. 14, pp. 113–128, 2012.
[5] A. Schneider. (2015, Jun.). Tech makeover [Online]. Available: http://www.innewyork.com/jun15/tech-gift-guide
[6] A. Pantelopoulos and N. G. Bourbakis, "A survey on wearable sensor-based systems for health monitoring and prognosis," *IEEE Trans. Sys., Man, Cybern., Part C: Appl. Rev.*, vol. 40, no. 1, pp. 1–12, Jan. 2010.
[7] A. Page, O. Kocabas, T. Soyata, M. K. Aktas, and J. Couderc, "Cloud-based privacy-preserving remote ECG monitoring and surveillance," *Ann. Noninvasive Electrocardiol.*, vol. 20, no. 4, pp. 328–337, 2014.
[8] M. Hassanalieragh, A. Page, T. Soyata, G. Sharma, M. K. Aktas, G. Mateos, B. Kantarci, and S. Andreescu, "Health monitoring and management using internet-of-things (IoT) sensing with cloud-based processing: Opportunities and challenges," in *Proc. IEEE Int. Conf. Serv. Comput.*, Jun. 2015, pp. 285–292.
[9] Care Cloud. (2013). [Online]. Available: http://www.carecloud.com/
[10] Dr Chrono. (2013). [Online]. Available: https://drchrono.com/
[11] (2015). Amazon Web Services [Online]. Available: http://aws.amazon.com
[12] (2015). Google Cloud Platform [Online]. Available: https://cloud.google.com/
[13] (2015). Microsoft Windows Azure [Online]. Available: http://www.microsoft.com/windowazure
[14] A. Benharref and M. A. Serhani, "Novel cloud and SOA-based framework for E-Health monitoring using wireless biosensors," *IEEE J. Biomed. Health Inf.*, vol. 18, no. 1, pp. 46–55, Jan. 2014.
[15] S. Babu, M. Chandini, P. Lavanya, K. Ganapathy, and V. Vaidehi, "Cloud-enabled remote health monitoring system," in *Proc. Int. Conf. Recent Trends Inform. Tech.*, Jul. 2013, pp. 702–707.

[16] C. O. Rolim, F. L. Koch, C. B. Westphall, J. Werner, A. Fracalossi, and G. S. Salvador, "A cloud computing solution for patient's data collection in health care institutions," in *Proc. Int. Conf. eHealth, Telemed., Social Med.*, Feb. 2010, pp. 95–99.

[17] T. Soyata, R. Muraleedharan, S. Ames, J. H. Langdon, C. Funai, M. Kwon, and W. B. Heinzelman, "COMBAT: Mobile Cloud-based cOmpute/coMmunications infrastructure for BATtlefield applications," in *Proc. SPIE*, May 2012, pp. 84 030K–84 030K.

[18] W. Zhao, C. Wang, and Y. Nakahira, "Medical application on internet of things," in *Proc. IET Int. Conf. Commmun. Technol. Appl.*, Oct. 2011, pp. 660–665.

[19] T. Soyata, R. Muraleedharan, C. Funai, M. Kwon, and W. Heinzelman, "Cloud-vision: Real-time face recognition using a mobile-cloudlet-cloud acceleration architecture," in *Proc. IEEE Symp. Comput. Commun.*, Jul. 2012, pp. 59–66.

[20] N. Powers, A. Alling, K. Osolinsky, T. Soyata, M. Zhu, H. Wang, H. Ba, W. Heinzelman, J. Shi, and M. Kwon, "The cloudlet accelerator: Bringing mobile-cloud face recognition into real-time," in *Proc. Globecom Workshops*, Dec. 2015.

[21] G. Nalinipriya and K. R. Aswin, "Extensive medical data storage with prominent symmetric algorithms on cloud - a protected framework," in *Proc. IEEE Int. Conf. Smart Struct. Syst.*, Mar. 2013, pp. 171–177.

[22] A. F. Hani, I. V. Paputungan, M. F. Hassan, V. S. Asirvadam, and M. Daharus, "Development of private cloud storage for medical image research data," in *Proc. Int. Conf. Comput. Inf. Sci.*, Jun. 2014, pp. 1–6.

[23] Y. Mao, Y. Chen, G. Hackmann, M. Chen, C. Lu, M. Kollef, and T. C. Bailey, "Medical data mining for early deterioration warning in general hospital wards," in *Proc. IEEE 11th Int. Conf. Data Mining Workshops*, Dec. 2011, pp. 1042–1049.

[24] O. Kocabas and T. Soyata, "Medical data analytics in the cloud using homomorphic encryption," in *Handbook of Research on Cloud Infrastructures for Big Data Analytics*, P. R. Chelliah and G. Deka, Eds. Hershey, PA, USA: IGI Global, Mar. 2014, ch. 19, pp. 471–488.

[25] B. Rao, "The role of medical data analytics in reducing health fraud and improving clinical and financial outcomes," in *Proc. IEEE 26th Int. Symp. Comput.-Based Med. Syst.*, Jun. 2013, pp. 3–3.

[26] G. Barbash and S. Glied, "New technology and health care coststhe case of robot-assisted surgery," *New Engl. J. Med.*, vol. 363, no. 8, pp. 701–704, 2010.

[27] A. Page, M. K. Aktas, T. Soyata, W. Zareba, and J. Couderc, "QT clock to improve detection of QT prolongation in long QT syndrome patients," *Heart Rhythm*, vol. 13, no. 1, pp. 190–198, Jan. 2016.

[28] O. Goldreich, *Foundations of Cryptography: Volume 2, Basic Applications*. Cambridge, U.K.: Cambridge Univ. Press, 2004.

[29] N. Cao, C. Wang, M. Li, K. Ren, and W. Lou, "Privacy-preserving multi-keyword ranked search over encrypted cloud data," *IEEE Trans. Parallel Distrib. Syst.*, vol. 25, no. 1, pp. 222–233, Jan. 2014.

[30] D. Boneh, C. Gentry, S. Halevi, F. Wang, and D. J. Wu, "Private database queries using somewhat homomorphic encryption," in *Proc. Appl. Cryptography Netw. Security*, 2013, pp. 102–118.

[31] V. Nikolaenko, U. Weinsberg, S. Ioannidis, M. Joye, D. Boneh, and N. Taft, "Privacy-preserving ridge regression on hundreds of millions of records," in *Proc. IEEE Symp. Security Privacy*, 2013, pp. 334–348.

[32] R. Gennaro, C. Gentry, and B. Parno, "Non-interactive verifiable computing: Outsourcing computation to untrusted workers," in *Proc. CRYPTO*, 2010, pp. 465–482.

[33] D. Boneh and D. M. Freeman, "Homomorphic signatures for polynomial functions," in *Proc. EUROCRYPT*, 2011, pp. 149–168.

[34] S. Dziembowski and K. Pietrzak, "Leakage-resilient cryptography," in *Proc. IEEE 49th Annu. IEEE Symp. Found. Comput. Sci.*, 2008, pp. 293–302.

[35] Y. Zhou and D. Feng, "Side-channel attacks: Ten years after its publication and the impacts on cryptographic module security testing." *IACR Cryptol. ePrint Archive*, vol. 2005, p. 388, 2005.

[36] P. C. Kocher, "Timing attacks on implementations of diffie-hellman, rsa, dss, and other systems," in *Proc. Adv. Cryptol.*, 1996, pp. 104–113.

[37] P. L. Montgomery, "Speeding the pollard and elliptic curve methods of factorization," *Math. Comput.*, vol. 48, no. 177, pp. 243–264, 1987.

[38] J. López and R. Dahab, "Fast multiplication on elliptic curves over GF(2^m) without precomputation," in *Proc. Cryptographic Hardw. Embedded Syst.*, 1999, pp. 316–327.

[39] P. Kocher, J. Jaffe, and B. Jun, "Differential power analysis," in *Proc. Adv. Cryptol.*, 1999, pp. 388–397.

[40] T. S. Messerges, "Securing the aes finalists against power analysis attacks," in *Proc. Fast Softw. Encryption*, 2001, pp. 150–164.

[41] J.-S. Coron, "Resistance against differential power analysis for elliptic curve cryptosystems," in *Proc. Cryptographic Hardw. Embedded Syst.*, 1999, pp. 292–302.

[42] D. Boneh, R. A. DeMillo, and R. J. Lipton, "On the importance of checking cryptographic protocols for faults," in *Proc. EUROCRYPT*, 1997, pp. 37–51.

[43] X. Guo, D. Mukhopadhyay, C. Jin, and R. Karri, "Security analysis of concurrent error detection against differential fault analysis," *J. Cryptographic Eng.*, vol. 5, no. 3, pp. 153–169, 2015.

[44] R. Karri, K. Wu, P. Mishra, and Y. Kim, "Fault-based side-channel cryptanalysis tolerant rijndael symmetric block cipher architecture," in *Proc. IEEE Int. Symp. Defect Fault Tolerance VLSI Syst.*, 2001, pp. 427–435.

[45] G. Bertoni, L. Breveglieri, I. Koren, P. Maistri, and V. Piuri, "Error analysis and detection procedures for a hardware implementation of the advanced encryption standard," *IEEE Trans. Comput.*, vol. 52, no. 4, pp. 492–505, Apr. 2003.

[46] I. Biehl, B. Meyer, and V. Müller, "Differential fault attacks on elliptic curve cryptosystems," in *Proc. CRYPTO*, 2000, pp. 131–146.

[47] M. Mozaffari-Kermani, R. Azarderakhsh, and A. Aghaie, "Reliable and error detection architectures of pomaranch for false-alarm-sensitive cryptographic applications," *IEEE Trans. VLSI Syst.*, vol. 23, no. 12, pp. 2804–2812, Dec. 2015.

[48] S. Bayat-Sarmadi, M. Mozaffari-Kermani, and A. Reyhani-Masoleh, "Efficient and concurrent reliable realization of the secure cryptographic sha-3 algorithm," *IEEE Trans. Comput.-Aided Des. Integrated Circuits Syst.*, vol. 33, no. 7, pp. 1105–1109, Jul. 2014.

[49] D. J. Bernstein. (2005). Cache-timing attacks on AES, Tech. Rep. [Online]. Available: http://cr.yp.to/papers.html#cachetiming

[50] D. A. Osvik, A. Shamir, and E. Tromer, "Cache attacks and countermeasures: The case of AES," in *Proc. Topics Cryptol.*, 2006, pp. 1–20.

[51] N. F. Pub, "Advanced encryption standard (AES)," Federal information processing standards publication, vol. 197, pp. 441–0311, 2001.

[52] S. Gueron, "Intels new aes instructions for enhanced performance and security," in *Proc. Fast Softw. Encryption*, 2009, pp. 51–66.

[53] B. B. Brumley and R. M. Hakala, "Cache-timing template attacks," in *Proc. 15th Int. Conf. Theory Appl. Cryptol. Inf. Security: Adv. Cryptol.*, 2009, pp. 667–684.

[54] (2015). US Department of Health and Human Services. Health Insurance Portability and Accountability Act [Online]. Available: http://www.hhs.gov/ocr/privacy/

[55] T. Soyata, L. Copeland, and W. Heinzelman, "RF energy harvesting for embedded systems: A survey of tradeoffs and methodology," *IEEE Circuits Syst. Mag.*, Feb. 12, 2016, Doi: 10.1109/MCAS.2015.2510198.

[56] W. Diffie and M. Hellman, "New directions in cryptography," *IEEE Trans. Inf. Theor.*, vol. 22, no. 6, pp. 644–654, Nov. 2006.

[57] C. Poon, Y. Zhang, and S. Bao, "A novel biometrics method to secure wireless body area sensor networks for telemedicine and m-health," *IEEE Commun. Mag.*, vol. 44, no. 4, pp. 73–81, Apr. 2006.

[58] K. K. Venkatasubramanian, A. Banerjee, and S. Gupta, "PSKA: Usable and secure key agreement scheme for body area networks," *IEEE Trans. Inf. Technol. Biomed.*, vol. 14, no. 1, pp. 60–68, Jan. 2010.

[59] A. Sahai and B. Waters, "Fuzzy identity-based encryption," in *Proc. EUROCRYPT*, 2005, pp. 457–473.

[60] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Security*, 2006, pp. 89–98.

[61] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Security Privacy*, 2007, pp. 321–334.

[62] D. F. Ferraiolo and D. R. Kuhn, "Role-based access controls," *arXiv preprint arXiv:0903.2171*, 2009.

[63] M. Li, W. Lou, and K. Ren, "Data security and privacy in wireless body area networks," *IEEE Wireless Commun.*, vol. 17, no. 1, pp. 51–58, Feb. 2010.
[64] A. Page, T. Soyata, J. Couderc, M. Aktas, B. Kantarci, and S. Andreescu, "Visualization of health monitoring data acquired from distributed sensors for multiple patients," in *Proc. IEEE Global Telecommun. Conf.*, Dec. 2015.
[65] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proc. 41st Annu. ACM Symp. Theory Comput.*, 2009, pp. 169–178.
[66] O. Kocabas, T. Soyata, J. Couderc, M. K. Aktas, J. Xia, and M. Huang, "Assessment of cloud-based health monitoring using homomorphic encryption," in *Proc. 31st IEEE Int. Conf. Comput. Des.*, Ashville, VA, USA, Oct. 2013, pp. 443–446.
[67] O. Kocabas and T. Soyata, "Utilizing homomorphic encryption to implement secure and private medical cloud computing," in *Proc. IEEE 8th Int. Conf. Cloud Comput.*, Jun. 2015, pp. 540–547.
[68] D. McGrew and J. Viega. (2004). The galois/counter mode of operation (GCM). [Online]. Available: *Submission to NIST. http://csrc. nist. gov/CryptoToolkit/modes/proposedmodes/gcm/gcm-spec. pdf*
[69] A. A. Group, "Armv8 instruction set overview," 2011.
[70] S. Morioka and A. Satoh, "An optimized S-Box circuit architecture for low power AES design," in *Proc. Cryptographic Hardw. Embedded Syst.*, 2003, pp. 172–186.
[71] D. Canright, *A Very Compact S-Box for AES*. New York, NY, USA: Springer, 2005.
[72] Y. Nogami, K. Nekado, T. Toyota, N. Hongo, and Y. Morikawa, "Mixed bases for efficient inversion in $F((2^2)^2)^2$ and conversion matrices of subbytes of AES," in *Proc. Cryptographic Hardw. Embedded Syst.*, 2010, pp. 234–247.
[73] X. Zhang and K. K. Parhi, "On the optimum constructions of composite field for the aes algorithm," *IEEE Trans. Circuits Syst. II*, vol. 53, no. 10, pp. 1153–1157, Oct. 2006.
[74] A. Satoh, T. Sugawara, and T. Aoki, "High-performance hardware architectures for Galois counter mode," *IEEE Trans. Comput.*, vol. 58, no. 7, pp. 917–930, Jul. 2009.
[75] S. Gueron and M. E. Kounavis, "Intel® carry-less multiplication instruction and its usage for computing the GCM mode," *White Paper*, 2010.
[76] N. Koblitz, "Elliptic curve cryptosystems," *Math. Comput.*, vol. 48, no. 177, pp. 203–209, 1987.
[77] V. Miller, "Use of elliptic curves in cryptography," in *Proc. Adv. Cryptol.*, 1986, pp. 417–426.
[78] J. M. Pollard, "Monte Carlo methods for index computation," *Math. Comput.*, vol. 32, no. 143, pp. 918–924, 1978.
[79] D. Hankerson, A. J. Menezes, and S. Vanstone, *Guide to Elliptic Curve Cryptography*. New York, NY, USA: Springer, 2006.
[80] V. G. Martínez, L. H. Encinas, and C. S. Ávila, "A survey of the elliptic curve integrated encryption scheme," vol. 80, no. 1024, pp. 160–223, 2010.
[81] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, 1978.
[82] X. Yao, Z. Chen, and Y. Tian, "A lightweight attribute-based encryption scheme for the internet of things," *Future Gener. Comput. Syst.*, vol. 49, pp. 104–112, 2015.
[83] O. Kocabas and T. Soyata, "Towards privacy-preserving medical cloud computing using homomorphic encryption," in *Enabling Real-Time Mobile Cloud Computing through Emerging Technologies*, T. Soyata, Ed. Hershey, PA, USA: IGI Global, 2015, ch. 7, pp. 213–246.
[84] S. Goldwasser and S. Micali, "Probabilistic encryption & how to play mental poker keeping secret all partial formation," in *Proc. STOC*, 1982, pp. 365–377.
[85] T. El Gamal, "A public key cryptosystem and a signature scheme based on discrete logarithms," in *Proc. Adv. Cryptol.*, 1985, pp. 10–18.
[86] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Proc. 17th Int. Conf. Theory Appl. Cryptographic Techn.*, 1999, pp. 223–238.
[87] D. Boneh, E. Goh, and K. Nissim, "Evaluating 2-dnf formulas on ciphertexts," in *Proc. Conf. Theory Cryptography*, 2005, pp. 325–341.
[88] O. Kocabas, R. Gyampoh-Vidogah, and T. Soyata, "Operational cost of running real-time mobile cloud applications," in *Enabling Real-Time Mobile Cloud Computing through Emerging Technologies.*, T. Soyata, Ed. Hershey, PA, USA: IGI Global, 2015, ch. 10, pp. 294–321.
[89] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *Proc. 3rd Innovations Theoretical Comput. Sci. Conf.*, 2012, pp. 309–325.
[90] R. L. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection: Conveying the utility of homomorphic encryption and multiparty computation," *IEEE Signal Process. Mag.*, vol. 30, no. 1, pp. 82–105, Jan. 2013.
[91] M. van Dijk, C. Gentry, S. Halevi, and V. Vaikuntanathan, "Fully homomorphic encryption over the integers," in *Proc. EUROCRYPT*, 2010, pp. 24–43.
[92] Z. Brakerski and V. Vaikuntanathan, "Fully homomorphic encryption from Ring-LWE and security for key dependent messages," in *Proc. 31st Annu. Conf. Adv. Cryptol.*, 2011, vol. 6841, p. 501.
[93] Z. Brakerski and V. Vaikuntanathan, "Efficient fully homomorphic encryption from (standard) LWE," in *Proc. IEEE 52nd Annu. Symp. Found. Comput. Sci.*, 2011, pp. 97–106.
[94] N. P. Smart and F. Vercauteren, "Fully homomorphic encryption with relatively small key and ciphertext sizes," in *Proc. 13th Int. Conf. Practice Theory Public Key Cryptography*, 2010, pp. 420–443.
[95] A. Page, O. Kocabas, S. Ames, M. Venkitasubramaniam, and T. Soyata, "Cloud-based secure health monitoring: Optimizing fully-homomorphic encryption for streaming algorithms," in *Proc. Globecom Workshops*, Dec. 2014, pp. 48–52.
[96] N. P. Smart and F. Vercauteren. (2011). Fully homomorphic SIMD operations [Online]. Available: http://eprint.iacr.org/2011/133
[97] H. C. Bazett-group, "An analysis of the time-relations of electrocardiograms," *Ann. Noninvasive Electrocardiol.*, vol. 2, no. 2, pp. 177–194, 1997.
[98] J. E. Savage, *Models of Computation: Exploring the Power of Computing*, 1st ed. Reading, MA, USA: Addison-Wesley, 1997.
[99] P. M. Kogge and H. S. Stone, "A parallel algorithm for the efficient solution of a general class of recurrence equations," *IEEE Trans. Comput.*, vol. 22, no. 8, pp. 786–793, Aug. 1973.
[100] J. A. Akinyele, C. Garman, I. Miers, M. W. Pagano, M. Rushanan, M. Green, and A. D. Rubin, "Charm: A framework for rapidly prototyping cryptosystems," *J. Cryptographic Eng.*, vol. 3, no. 2, pp. 111–128, 2013.
[101] S. Halevi and V. Shoup. (2015). [Online]. Available: https://github.com/shaih/HElib
[102] J. Couderc, "The telemetric and holter ECG warehouse initiative (THEW): A data repository for the design, implementation and validation of ECG-related technologies," in *Proc. Conf. IEEE Eng. Med. Biol. Soc.*, 2010, pp. 6252–6255.
[103] A. Page, T. Soyata, J. Couderc, and M. K. Aktas, "An open source ECG clock generator for visualization of long-term cardiac monitoring data," *IEEE Access*, vol. 3, pp. 2704–2714, Dec. 2015, http://ieeexplore.ieee.org/xpl/articleDetails.jsp?arnumber=7358049
[104] E. Barker and A. Roginsky, "Transitions: Recommendation for transitioning the use of cryptographic algorithms and key lengths," *NIST Special Publication*, vol. 800, p. 131A, 2011.
[105] C. Gentry, S. Halevi, and N. P. Smart, "Homomorphic evaluation of the AES circuit," in *Proc. CRYPTO*, 2012, pp. 850–867.

Ovunc Kocabas received the BS degree in microelectronics engineering from Sabanci University, Istanbul, Turkey, in 2006, and the MS degree in electrical and computer engineering from Rice University, Houston, TX, in 2011. He received the PhD degree from the University of Rochester, ECE, on December 15, 2015. His research interests include secure cloud computing, computer security, system design, and high-performance computer architecture design. He published six conference papers and one book chapter to date in his research areas.

Tolga Soyata received the BS degree in electrical and communications engineering from Istanbul Technical University in 1988, the MS degree in electrical and computer engineering from Johns Hopkins University in 1992, and the PhD degree in electrical and computer engineering from the University of Rochester in 1999. He joined the University of Rochester ECE Department in 2008, where he is currently an assistant professor - research. He manages the CUDA Research Center and CUDA Teaching Center programs for the University of Rochester, and Xilinx University Program and MOSIS Educational Program for the ECE Department. He teaches courses in VLSI ASIC Design, GPU Parallel Programming, and FPGA-based Advanced Digital Design. His current research interests include cyber physical systems and many aspects of digital health (D-Health). He is a member of the IEEE.

Mehmet K. Aktas received the BA degree in biology from the University of Rochester in 2002 and completed the medical school education at SUNY Upstate Medical University. He completed Internal Medicine residency training at the Cleveland Clinic and then proceeded to the University of Rochester Medical Center (URMC), where he completed advanced fellowships in cardiovascular diseases and cardiac pacing and electrophysiology. He received the MBA degree from the University of Rochester's Simon School. He is on the faculty at URMC as an associate professor of medicine. He is board certified in internal medicine, cardiovascular diseases and cardiac pacing, and electrophysiology. His clinical work involves the treatment of patients with a variety of complex heart rhythm disorders. His research is focused on improved risk stratification of patients with heart rhythm disorders and development of systems to enable early detection of arrhythmias.