IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received June 22, 2019, accepted July 9, 2019, date of publication July 19, 2019, date of current version September 9, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2929954

SMCACC: Developing an Efficient Dynamic Secure Framework for Mobile Capabilities Augmentation Using Cloud Computing

DIAA SALAMA ABD ELMINAAM[1], FARAH TURKEY ALANEZI [1], AND KHALID M. HOSNY[2]
[1]Department of Information Systems, Faculty of Computers and Informatics, Benha University, Benha 13511, Egypt
[2]Department of Information Technology, Faculty of Computers and Informatics, Zagazig University, Zagazig 44519, Egypt

Corresponding author: Diaa Salama Abd Elminaam (diaa.salama@fci.bu.edu.eg)

This work was supported in part by Benha University.

ABSTRACT Mobile capability development increases with the need to use it. However, mobile devices still lack computational resources. Mobile cloud computing is the solution to overcome these challenges. Extensive research has been conducted to solve these problems, and a large number of new techniques were developed. Many of these researches solve the resource problem by partitioning and offloading applications to the cloud to tap its full computational and storage availability. Other methods involve offloading part of the applications while retaining the rest for processing on the smartphone — the decision making in these techniques based on metrics such as power and CPU consumption. Also, small numbers of available solutions consider security issues. This paper proposed a new elastic framework named secure framework for mobile capabilities augmentation using cloud computing (SMCACC) that enables transparent use of cloud resources to augment the capabilities of resource-constrained mobile devices. A significant feature of this framework is the partition of a single application into multiple components. Mobile apps can be executed on the mobile device itself or offloaded to the cloud clone for execution. Thus, the elastic application can augment the capabilities of a mobile device to save energy for a mobile device. Besides, a hybrid cryptography method is used to secure data and take energy consumption in the considerations. The new proposed security protocols use a combination of both symmetric and asymmetric cryptographic techniques to avoids the disadvantages of the existing hybrid protocols. These methods help to protect users by securing data that offloaded to the cloud. The results of this framework without security show the resources consumed for executing the application on mobile and cloud are decreased approximately to half of the memory consumed for running app on the mobile-only. According to the security framework, the resources consumed for executing the application on mobile still decreased.

INDEX TERMS Mobile computing, mobile cloud computing, GPS calculations, hybrid security method, fingerprint.

I. INTRODUCTION

Mobile devices, including smartphones and tablets, are increasingly becoming an essential part of human life that make them the most effective and suitable tools for communication and entertainment. Also, it unbound by time and place.

On these smartphones, there is a wide variety of operating systems that have been developed to manage resources. Operating systems such as Android, IOS, Windows Mobile, and BlackBerry allow programmers to build custom applications. Smartphones are the dominant future computing devices with high user expectations for accessing the computationally intensive applications seen in powerful stationary computing machines [1].

Smartphone users are exceedingly connected to the Internet these days; they can capture and manage photos and videos, play music, movies, and complex games, and download a lot of complex applications. However, the increasing number of mobile apps available requires more resources in terms of storage and processing capabilities. Figure 1 shows the market share of some popular smartphone brands in millions of units.

The associate editor coordinating the review of this manuscript and approving it for publication was Tao Zhang.

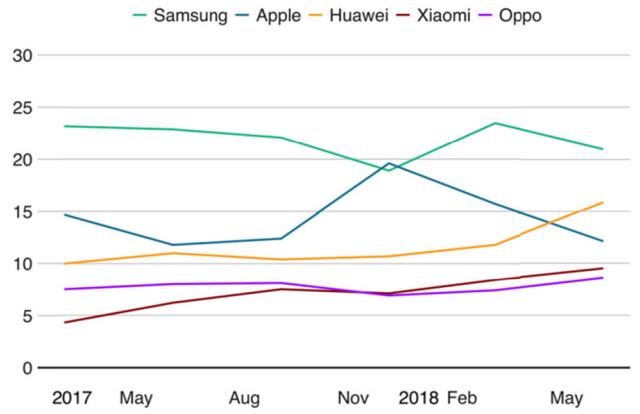

FIGURE 1. – Smartphone brand market share in millions of units.

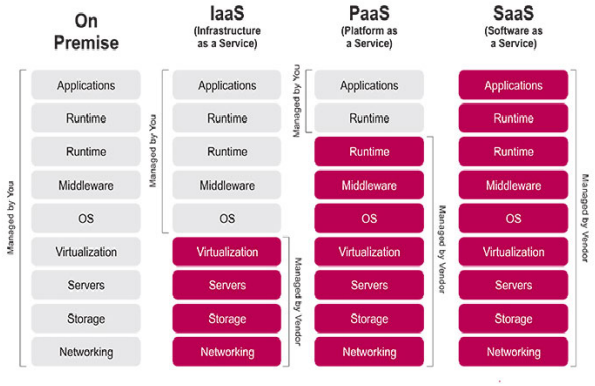

FIGURE 2. Cloud computing service models.

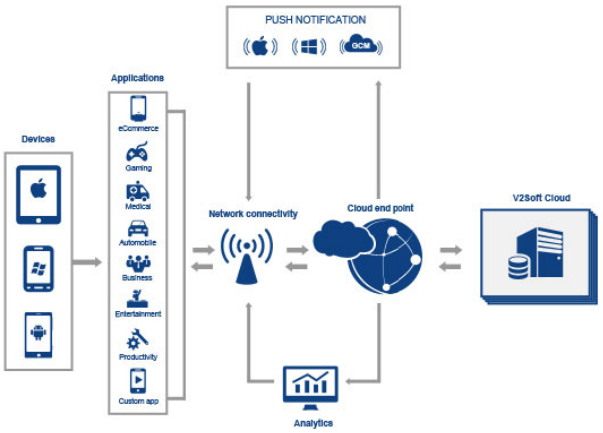

FIGURE 3. The architecture of mobile cloud computing.

Smartphones, compared to desktop computers, have less computing power, less storage capacity, fewer memory resources, and battery limitations. Demanding applications such as the ones mentioned above require more resources on mobile devices for better user experience. Hence smartphones are resource-constrained. Much research was conducted to address this problem, leading to four basic approaches to saving energy and extending battery life. Mobile cloud computing (MCC) is the solution to overcome the challenges.

Cloud computing is a kind of computing in which dynamic virtual and scalable resources supported as an internet service. Cloud services support software and hardware remotely from a location controlled by a third party for individuals and businesses [2]–[4].

The main goals of cloud computing are to improve the efficiency of the runtime without needful in investing in new infrastructure. The services model of could computing can be classified as the following [3]

- *infrastructure as a service (IaaS)*
- *data storage as a service (DaaS)*
- *hardware as a service (HaaS)*
- *software as a service (SaaS),*
- *platform as a service (PaaS).*

Cloud computing service models illustrate in Figure 2.

Mobile cloud computing is a new platform combining mobile devices and cloud computing into new infrastructure, wherein the cloud performs the heavy lifting for computing-intensive tasks and storing massive amounts of data.

Offloading is an essential method in mobile cloud computing (MCC) [5]–[11]. Due to the low battery capacity of mobile devices, a significant amount of research has performed on offloading. In [12], a combination of analysis and dynamic profiling modules is used to partition the application and specify which process should be transferred to the cloud. The authors in [13] monitor the remote execution of complete created VMs of smartphone system by using a profiler module and execution controller. **The main limitation of [12] and [13]** is the power consumption that is required for simple synchronisation with the emulation VM on the cloud [14]. Furthermore, the connection to the cloud is not secure.

The architecture of mobile cloud computing illustrated in Figure 3.

The purpose of this paper is to investigate the applicability of cloud computing in the area of smartphones applications. The focus lies on how cloud computing can improve mobile phone computational performance and usability. Because the computational resources of mobile phones are limited and because cloud computing could be a solution to improve the mentioned area, in order to obtain this goal, a new framework is developed.to improve security in the proposed framework, a new hybrid cryptography algorithm is proposed for achieving security in Mobile Cloud Computing framework.

The significant contributions of this paper are summarized as follow

- Building the proposed secure and cost-effective offloading schema for mobile cloud computing
- Developing a mobile-cloud infrastructure that will enable smartphone applications that are distributed both in terms of data and computation.

- This framework floods only intensive tasks to the cloud in a dynamically way based on four constraints, namely, the execution time of the tasks, CPU utilization, memory consumed, and energy consumption.
- building new hybrid cryptography algorithms for achieving security in the proposed framework.
- Four different types of mobile applications are used in experimental studies to distinguish between light and heavy mobile applications.

The remainder of the paper organized as follows: In section 2, an overview of previous works presented. In section 3, the materials and methods of the proposed system are described. In section 4, results and discussions are produced, before concluding and future work in sections 5 and 6.

II. RELATED WORKS

A. MOBILE CLOUD OFFLOADING SYSTEM OVERVIEW

A lot of different approaches proposed recently focusing on the challenges of mobile devices by moving computational tasks to cloud resources for remote execution [5], [6]. Some of these methods only transfer a process from the mobile device to the replicated virtual machine (VM) on the cloud [12], [13].

In Fernando *et al.*, [15], the authors provided an extensive survey of mobile cloud computing research while highlighting specific concerns of mobile cloud computing. The authors presented a classification based on the critical issues in this area and discussed different approaches to tackle these issues.

The synchronisation problem is handled by only offloading acute services in [16], rather than the complete process, to the cloud. Additionally, a model has been created to decide whether the service needs offloading or not. Although, the simplicity of the created model that prefer to make always remote execution, sometimes executing services on the mobile platform is crucial than offloading it to the cloud. It is vital to use some security techniques to secure the transferred data.

Other frameworks are proposing to offload intensive method only after partitioning the application [17]–[22]. These frameworks utilize an integer linear programming model similar to the proposed framework that is created to decide offloading decisions.

Many frameworks focus only on the limitations of a battery lifetime, energy consumption, and the required total response time, and ignoring the memory usage and the security of the offloading process [23]. On the other hand, in [24], a complete offloading to the full Android application from the mobile to the cloud is applied, where this is considered resource-consuming due in no small amount of data sent over the network. What is more, there is a crucial need for any security technique to save the application.

The main objective of the proposed method in [25] is to minimize the data transmission and energy-saving, which only offloads resource-intensive services and leverages the Software-as-a-Service model for the configuration of the rigorous services on the cloud server.

In [25], a dynamic resource allocation model for scheduling data-intensive applications on an integrated computation resource environment proposed. It composed of mobile devices, cloudlets, and public cloud. The allocation process is based on different restrictions related to the application structure, data size, and network configuration. It evaluated the performance of the proposed technique using many experiments. Results showed that the proposed technique improves the execution time for data-intensive applications by an average of 78%.

Other research can be found in [26]–[35].

B. SECURITY ISSUES ON CLOUD COMPUTING

A lot of different approaches proposed recently focusing on the challenges of security issues on cloud computing by using different encryption techniques [37]–[50]. Some of these methods only use a single encryption techniques methods and other used hybrid encryption.

The authors in [37] used a multi-cloud strategy to handle problems such as loss of privacy and loss of data. The proposed method addressed data confidentiality problem. The proposed method encrypts data through RSA before sending to the cloud. The system consists of two clouds, the application logic, and the data logic. When comparing the proposed system with the conventional system, the proposed system achieved security, integrity, and confidentiality.

In [45], the authors concentrate on storing data on cloud computing in an encrypted format using fully homomorphic encryption.

In [46], the authors applied the ElGamal algorithm to enhance cloud security and allow re-encrypting ciphertext in two levels (first level and second level). The encryption process takes place at the data owner side, CSP acts as a midway between the data owner and data user, CSP re-encrypts ciphertext with re-encryption key.

The authors in [45] present a new hybrid security method for achieving data security. Data is split into blocks of bits. Genetic algorithm is applied to every two blocks of bits. The results of each Genetic Algorithm are a block of ciphertext. Each ciphertext is stored in a different location.

Many frameworks focused only on the limitations of the security of cloud computing and suggested a new framework for a check on the availability of data over the cloud environment.

C. SECURITY ISSUES ON MOBILE CLOUD COMPUTING

Although the cloud-based approach can intensely extend the capability of mobile devices, the assignment of developing a secure and reliable mobile cloud offloading system remains challenging [51]. In recent years, numerous works about security in mobile cloud offloading and cloud computing has been presented [52]–[55].

Several security challenges are existing in the mobile cloud offloading scenario.

The authors in [56] proposed a technique with an outstanding feature of data integrity and confidentiality. The technique is based on the concept of RSA algorithm, Hash function to provide better security to the data stored on the mobile cloud. In this scheme, encryption is used to provide security to the data while in transit. Because the encrypted file is stored in the cloud, so the user can believe that his data is secure.

Garg and Sharma [57] proposed a secure data service that outsources data and security management to cloud in trusted mode. The secure data service allows mobile users to move data and data sharing overhead to cloud without disclosing any information.

Taking into consideration all of the mentioned work, other works considered security in MCC, memory usage constraints in their models. In this paper, a model that handles four different constraints in the offloading decision will be formulated. This model made the offloading decision dynamically at runtime. Besides, we provided this framework with new hybrid cryptography protocols to secured the offloaded data to the cloud. The proposed framework is tested with four different types of mobile applications that were developed using Android.

III. MATERIALS AND METHODS

The proposed framework was developed to help software owners in dividing the processing operations of their applications between those running on the mobile site and the server-side based on different metrics such as the execution time of the separated methods; memory consumed, the power consumed for each method. It also uses a hybrid cryptography method designed to secure the data stored and transferred in minimum time. First, we present the framework architecture, and then we present the implementation of the hybrid cryptography method that was added to secure the transferred data from any software.

A. FRAMEWORK ARCHITECTURE

In order to use the framework, the software should first upload an application on it, which will then be processed, as shown in Figures 4, 5, and 6.

Algorithm 1 Framework Execution Flow

i. *Get the implemented application*
ii. *Divide the project into a set of methods*
iii. *Calculate the execution time for running each method*
iv. *Determine the method with the most massive execution time*
v. *Store running configurations*

As shown in Fig 6, the framework architecture consists of five modules, namely, application methods, estimator, decision-maker, mobile manager, and cloud manager.

The framework starts to work at the method level, where the developers need to add all exhaustive methods at the developing step. These methods should require additional computation resources and can be offloaded to the cloud for remote execution.

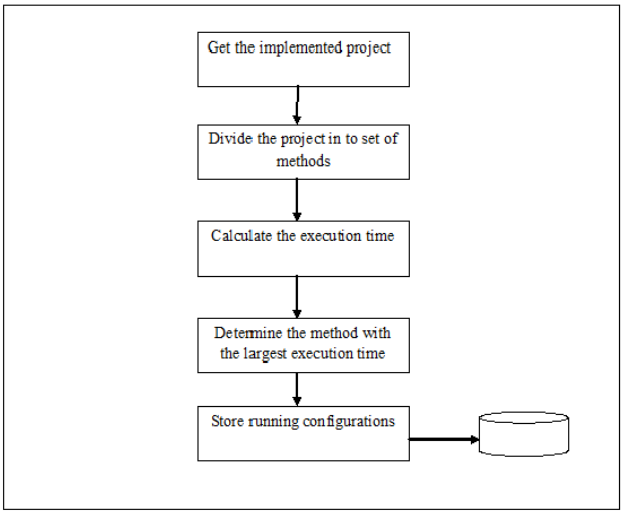

FIGURE 4. Main Framework architecture.

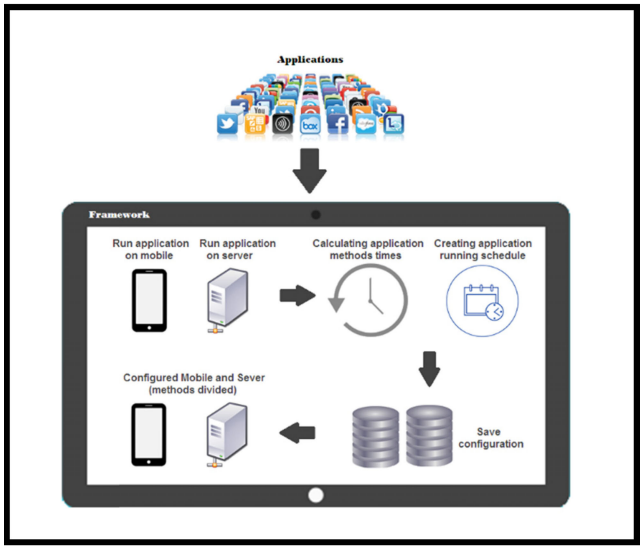

FIGURE 5. Framework procedure flow.

Estimator: The estimator module is responsible for identifying these methods for local execution on the mobile device and remote execution on the cloud with different input sizes by calculating the values of execution time, memory usage, CPU utilization, and energy consumption for each method.

Decision maker: it obtains the values of execution time, memory usage, CPU utilization, and energy consumption from the estimator module for each method. Then, the module creates a new file for each.

Mobile Manager: is deployed on the mobile side only if the methods executed on the mobile side, the file is updated with the new values. If the methods executed on the cloud side, the mobile manager decides to encrypt the offload data or not using one of our proposed hybrid cryptography algorithms.

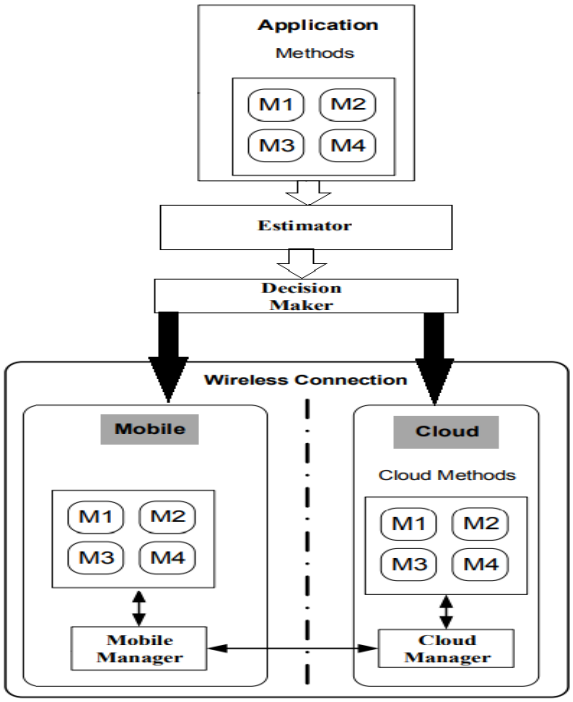

FIGURE 6. Detailed Framework architecture.

Cloud Manager: is deployed on the cloud side only if it received methods from the mobile manager to be executed on the cloud.

Therefore, the objective function should minimize the summation of four parameters as listed below.

- Cost of transferring methods from mobile to cloud
- Memory used
- CPU used
- Power Consumption

Also, take in consideration three constraints that must be handled as follows:

- **Minimize the memory used** by the application methods on the mobile device. This constraint can guarantee that the total memory for executing the application methods remotely on the cloud must be less than the total memory consumed for executing the methods of the application locally on the mobile device.
- **Minimize the total execution time**, that is, the second constraint, for the application. This constraint can guarantee that the total time for executing the application methods remotely on the cloud must be less than the total time for executing the methods of the application locally on the mobile device.
- **Minimize the total energy consumption**, that is, the last constraint for the objective function. This constraint deals with the energy consumed by executing the application method.

After dividing the methods into two sets (i.e., the methods to be run on the mobile side, and those run on the server-side), the application will run with the stored configuration.

The framework calculates the time, memory utilization, average CPU/processor cycles and average power/battery consumption for each method in the application to determine which methods should run on mobile and which should run on the server. The framework calculates these values based on the following formulas:

$$Time = (OP_{after} - OP_{before}) * 60 * 60 \quad (1)$$

where OP_{after} and OP_{before} are the operating system time after executing the method and before executing the method, respectively.

$$Memory = (AM_{after} - AM_{before}) * 1024 \quad (2)$$

where AM_{after} and AM_{before} are the allocated memory after executing the method and before executing the method, respectively.

$$CPU\ cycles = No.\ of\ Cycles \quad (3)$$

where $No.\ of\ Cycles$ are calculated in connection with the use of the method for the following subtasks:

1) *Load (5 cycles)*
2) *Store (4 cycles)*
3) *R-type (4 cycles)*
4) *Branch (3 cycles)*
5) *Jump (3 cycles)*

$$Power = (0.00148775) * No.\ of\ bytes \quad (4)$$

where (0.00148775) is the power consumption per byte.

B. HYBRID CRYPTOGRAPHY METHODS

The proposed framework uses a hybrid cryptography method developed in work [36]–[38] to secure information stored and transmitted through the cloud. The hybrid cryptography method aims to encrypt and secure the stored and transmitted fingerprints efficiently.

- **First Hybrid cryptography method**

The Encryption phase is shown in Figure 5-1. The plaintext is divided into n blocks, B_i. Each block consists of 128 bits. Then, it is divided into two parts P1 (0: n/2-1) blocks and P2 (n/2: n-1) blocks. If n is not integer number and has a fraction, NHCP protocol uses padding with null for the last block to be 128 bits.

The first n/2 blocks are encrypted using *(AES and ECC)* hybrid encryption algorithm, as illustrated below.

P_1 will be encrypted using *AES* by the key k_i, which is the secret key of *AES* encryption algorithm with size 128 bits. K_i is encrypted by *ECC* to produce K_j with length L.

$$M = \sum_{i=0}^{i=n/2-1} (Bi) \quad (5)$$

$$\boldsymbol{K_j = ECC_{enc}(TC_{PK}, k_{i-1})} \qquad \text{for } 0 < j \leq L - 1 \quad (6)$$

where ECC_{enc} is Elliptic Curve encryption function. It cyphers the input with the trust centre public key (TC_{PK}), which is used as a function to authenticate the key.

$$\boldsymbol{c_i = E_{AES}(K_j, B_i)} \quad (7)$$

where E_{AES} is the *AES* encryption function.

In parallel, the remaining $n/2$ blocks are encrypted using the *BLOWFISH- RSA* algorithm. *BLOWFISH- RSA* algorithm guarantees developing a stronger algorithm, as follows:

$$\boldsymbol{M} = \sum_{i=n/2}^{i=n-1} (Bi) \tag{8}$$

In this algorithm, two huge prime numbers are chosen; p and q. Then, $x = p \times q$, $\phi(x) = (p-1) \times (q-1)$. A number relatively prime to ϕ is chosen; d. Then, e is calculated such that $e \times d = 1 \bmod \phi(x)$, and Public key (e, x) is used for encrypting the key of BLOWFISH.

$$K_j = RSA_{enc}(k_{i-1}) \quad for\ 0 < j \leq L-1 \tag{9}$$

$$C_i = E_{BLOWFISH}(K_j, B_i) \tag{10}$$

MD-5 is applied to the cipher texts c_i and C_i. It is the best performance of hashing function security.

$$d_i = MD-5(c_i) \tag{11}$$

$$D_i = MD-5(C_i) \tag{12}$$

At the final stage of the encryption process, the two n/2 blocks are integrated to generate cypher text of n blocks, and it is sent to the sink node. The corresponding hash values (di and Di) with size 128 bits for each one are concatenated and sent to the sink node at the same time.

$$C = c_i + Ci \tag{13}$$

$$D = d_i + D_i \tag{14}$$

The proposed encryption algorithm is shown in Algorithm 1.

- **Strength of the First Hybrid Cryptography algorithm**

In the first Hybrid Cryptography algorithm, splitting the plain text improves the strength of the proposed cryptography algorithms. The intruder will be not able to identify which type of specific algorithm is applied to generate the ciphertext. Thus, it is impossible to decrypt the cypher text.

- **Second Hybrid cryptography methods**

In this paper, we build another two Hybrid cryptography methods shown in algorithm 2, and algorithm 3. Figure 7 shows the system encrypting the data using the Krishna encryption algorithm and triple DES encryption algorithm. It works on encrypting fingerprints images using the Krishna encryption algorithm and the triple DES encryption algorithm. Investigating previous work, the result shows that the first hybrid encryption algorithm takes the shortest time to encrypt data compared with all other algorithms.

The framework calculates the power consumption after applying the encryption algorithm based on the following formula.

$$Power = (0.00148775) * No.\ of\ bytes * ET \tag{15}$$

Algorithm The First Proposed Hybrid Encryption Algorithm

Input: *M (Plain text), k (secret key of AES encryption), s (128 bist size of the block);*

Output: *C (Ciphertext), c_i (encrypted text using AES with ECC), C_i(encrypted text using RSA), D (hashing value of cypher text);*

1. $n = M/s$;
2. *let* $i = 0$;
3. *do{*
4. $m = \sum_{i=0}^{i=n/2-1} (Bi)$ *The first part of plain text; what is m and where it is used, what is* B_i
5. ***for(j = 0; j <= n − 1; j++)***
6. *{*
7. $K_j = ECC_{enc}(TC_{PK}, k_{i-1})$;
8. *}*
9. $c_i = E_{AES}(K_j, B_i)$;
10. $d_i = MD\text{-}5\ (c_i)$;
11. i ++;
12. *}*
13. *while(i <n/2);*
14. $i = (n/2)$
15. *Let p and q two large prime numbers*
16. $x = p \times q$
17. $\phi(x) = (p-1) \times (q-1)$
18. *Let d a relatively prime number to* ϕ
19. $e \times d = 1\ mod\ \phi(x)$
20. *Let (e, x) public key of RSA.*
21. *do{*
22. $M = \sum_{i=n/2}^{i=n} (Bi)$ *second part of plain text which encrypted simultaneously with the first part;*
23. $K_j = RSA_{enc}(k_{i-1})$;
24. $C_i = E_{BLOWFISH}(K_j, B_i)$; ;
25. $D_i = MD\text{-}5\ (C_i)$;
26. *}*
27. *while(i < n);*
28. $C = c_i + C_i$;
29. $D = d_i + D_i$;

where (0.00148775) is the power consumption per byte, and ET is the encryption time.

EVALUATION AND ANALYSIS

The proposed framework is evaluated using **four different types of mobile applications**, as shown in Table 1. The experimental results measure four parameters for running the application methods locally on a mobile device and when offloading the methods to the cloud by using the framework. These parameters include processing Time, CPU utilization, battery consumption, and memory usage.

IV. FRAMEWORK PROTOTYPE

In this section, we will introduce an example for dividing the application using the implemented framework. As shown in Figure 8 and Figure 9, the software owner selects the

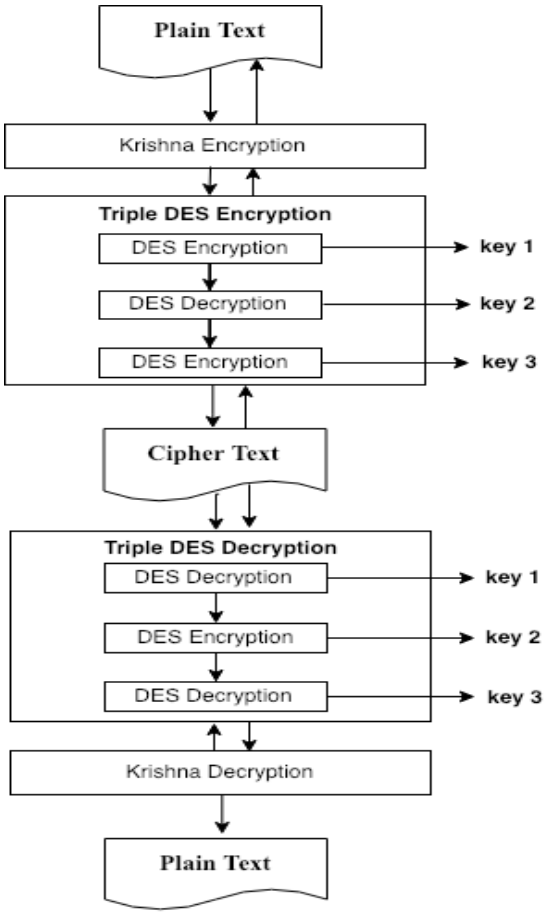

FIGURE 7. Hybrid encryption algorithm using Krishna and triple-DES algorithms.

application he wants to divide to yield the configuration of the running process of this application. Figure 9 illustrates the configuration of the running process for the car store application as an example. Figure 10 to Figure 16 shows how to use the fingerprint application. Figure 17 to Figure 24 demonstrate how to use the address book application. Figure 25 to Figure 34 shows how to use the car store application.

- ***Fingerprint application prototype***
- ***Test case: Name: Person 13, Password: user's fingerprint***

TABLE 1. Applications used in the experimental.

Application	Description
Fingerprint Print System	Detect Fingerprint Print for each user and his name
GPS calculation	Calculate GPS calculation such as distance between different number of point
Address Book	Store and retrieve names for all contact list
Car Store	Store the data for each car

Algorithm 2 Algorithm for Encryption

1. *Read Plain text file(ptF)*
2. *Krishna is used to encrypting (ptF) resulting in (Ck1)*
3. *Triple DES key1 is used to encrypt a (Ck1) resulting in (Ck2d1)*
4. *Triple DES key2 is used to decrypt (Ck2d1) resulting in (Ck3d2)*
5. *Triple DES key3 is used to encrypt (Ck3d2) resulting in (Ck4d3)*
6. *Final Cipher text Ck4d3 is that resulting from encryption using Krishna and Triple DES algorithms.*

Algorithm 3 Algorithm for Decryption

1. *The ciphertext (Ck4d3).*
2. *Triple DES key3 is used to decrypt (Ck4d3) resulting in (Ck3d2)*
3. *Triple DES key2 is used to encrypt (Ck3d2) resulting in (Ck2d1)*
4. *Triple DES key1 is used to decrypt (Ck2d1) resulting in (Ck1)*
5. *Krishna is used to decrypting a file (Ck1) resulting in (ptF)*

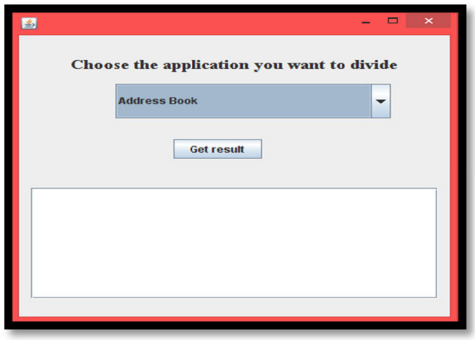

FIGURE 8. Framework interface.

- ***Address book application prototype***
- ***Car store application prototype***

A. EXPERIMENTAL AND EVALUATION RESULTS

1) FINGERPRINT IMPLEMENTATION SYSTEM

Each system is divided into two phases. The first phase is applying all processing on the mobile side, and the second phase is applying part of processing on the server-side while keeping the interface on the mobile side. Figure 35 illustrates the first phase, while Figure 36 illustrates the second phase.

- ***The first phase (mobile side)***
  - ***Preprocessing:***

Registration steps: to register a new person, several steps must be completed:

Step 1: A person will enter his name (X) and his fingerprint (FBY).

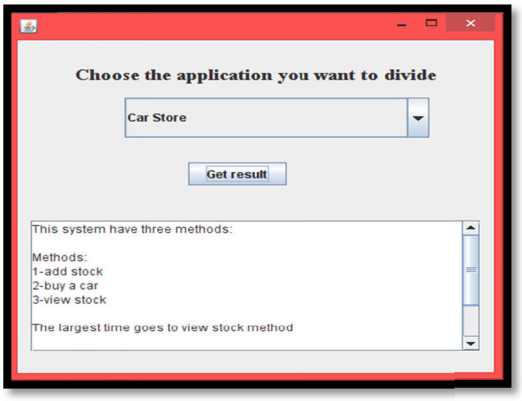

FIGURE 9. Example of getting the configuration of the car store application.

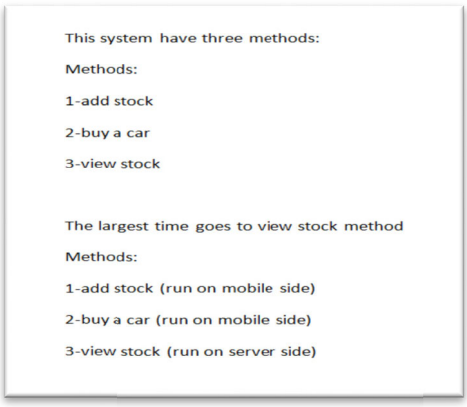

FIGURE 10. Configuration of the car store application.

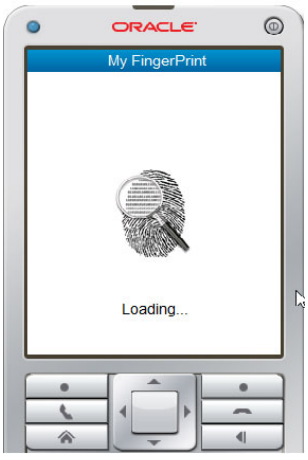

FIGURE 11. Opening of the fingerprint application.

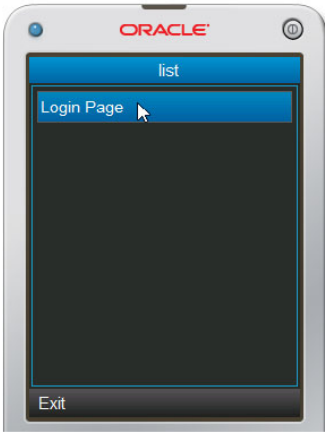

FIGURE 12. Login page.

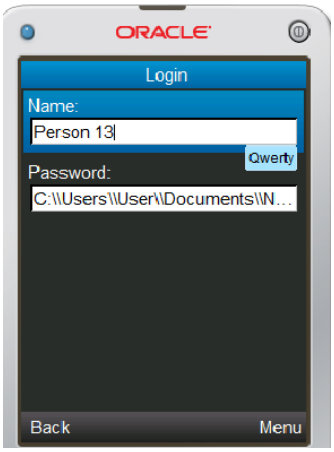

FIGURE 13. Entering username and password (user's fingerprint).

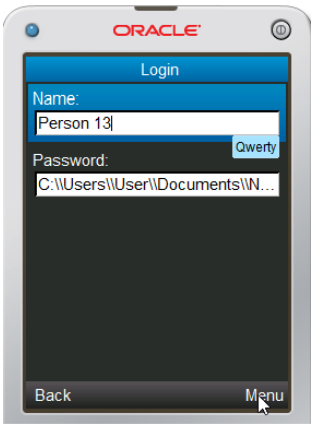

FIGURE 14. Submitting data.

Step 2: The system will convert a fingerprint (FBY) to an encoded fingerprint (FBY') using a base64 encoding algorithm.

Step 3: The system will save the person name and the encoded fingerprint (FBY') in the database ''Fingerprints.' A person (X) can register more than one fingerprint.

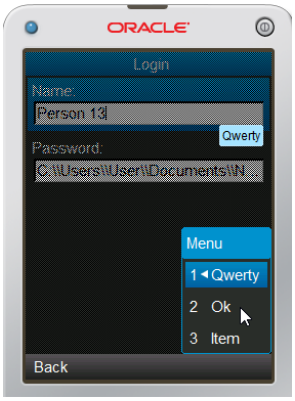

FIGURE 15. Sending data to server.

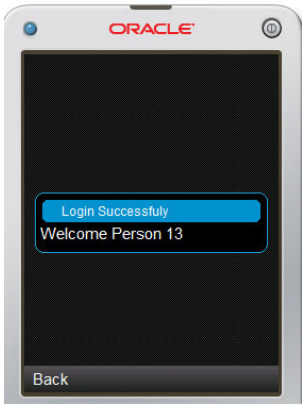

FIGURE 16. Verification screen.

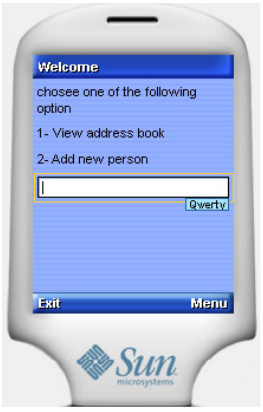

FIGURE 17. Application main menu.

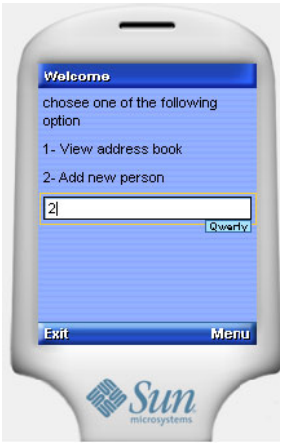

FIGURE 18. Choosing the (Add new Person) method.

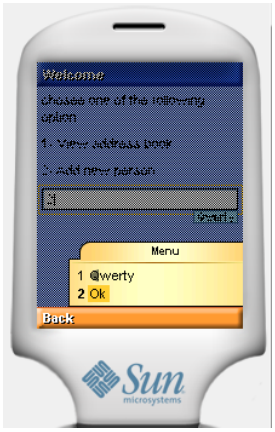

FIGURE 19. Opening of the fingerprint application.

FIGURE 20. Login page.

Login steps:

Step 1: A person will enter his name (X) and his fingerprint (FBY).

Step 2: The system will submit the person name (X) and his fingerprint (FBY) to the web service GetData (X, FBY).

Step 3: The system will convert the fingerprint (FBY) to an encoded fingerprint (FBY') using a base64 encoding algorithm.

Step 4: The system will retrieve all fingerprints (FBY_List) belonging to person (X) to form the database 'Fingerprints.'

Step 5: The system will compare all fingerprints (FBY_List) belonging to person (X) to form the database 'Fingerprints' with the encoded fingerprint (FBY').

Step 6: The system will return the results to the web service GetData (X, FBY).

FIGURE 21. Submitting user choice.

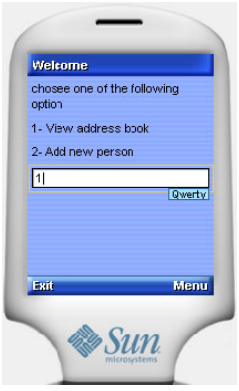

FIGURE 22. Choosing the (View address book) method.

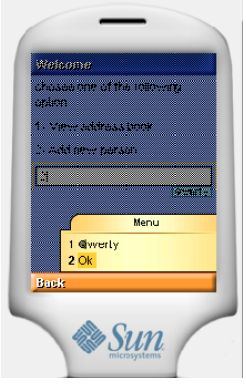

FIGURE 23. Submitting user choice.

Step 7: The system will return the results to the user (Login successfully - error).

- ***Print result:***

On the mobile side, printing the results requires several steps:

The system will return the results to the user (Login successfully - error).

- ***The second phase (server-side)***

Registration steps: on the server-side, to register a new person, several steps must be completed:

FIGURE 24. Viewing data stored in the address book.

FIGURE 25. Choosing the (Add new car) method.

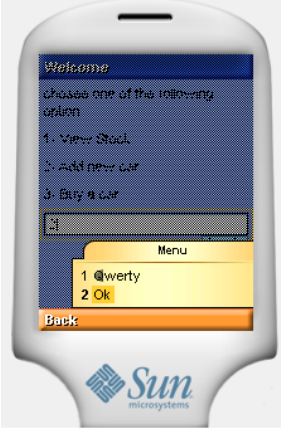

FIGURE 26. Submitting user choice.

Step 1: A person will enter his name (X) and his fingerprint (FBY).

Step 2: The system will convert the fingerprint (FBY) to an encoded fingerprint (FBY') using a base64 encoding algorithm.

Step 3: The system will save the person name and the encoded fingerprint (FBY') in the database 'Fingerprints.' A person (X) can register more than one fingerprint.

FIGURE 27. Entering car data.

FIGURE 28. Sending data to server.

FIGURE 29. Choosing the (Buy a car) method.

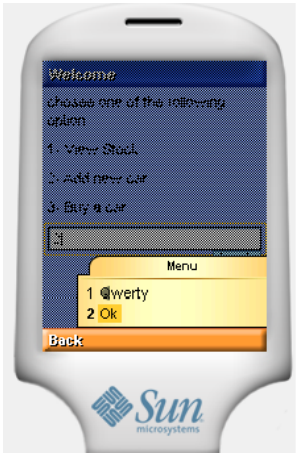

FIGURE 30. Submitting user choice.

FIGURE 31. Choosing a car.

FIGURE 32. Sending data to the server.

Login steps:

Step 1: A person will enter his name (X) and his fingerprint (FBY).

Step 2: The system will submit a person name (X) and his fingerprint (FBY) to the web service GetData (X, FBY).

Step 3: The system will convert the fingerprint (FBY) to an encoded fingerprint (FBY') using a base64 encoding algorithm.

Step 4: The system will retrieve all fingerprints (FBY_List) belonging to person (X) to form the database 'Fingerprints.'

FIGURE 33. Choosing the (View stock) method.

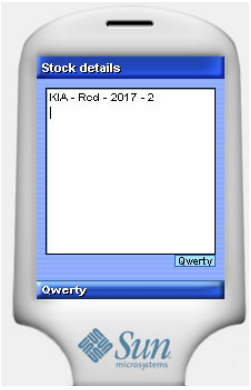

FIGURE 34. Viewing data exist in the stock.

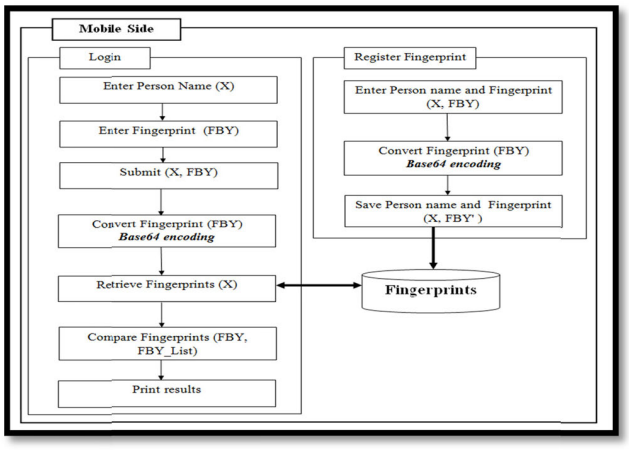

FIGURE 35. Applying to process on the mobile side.

Step 5: The system will compare all fingerprints (FBY_List) belonging to person (X) to form the database 'Fingerprints' with the encoded fingerprint (FBY').

Step 6: The system will return the results to the web service GetData (X, FBY).

2) GPS APPLICATION

In the experiments, the GPS application smartphone calculates some of GPS calculations such as distance between

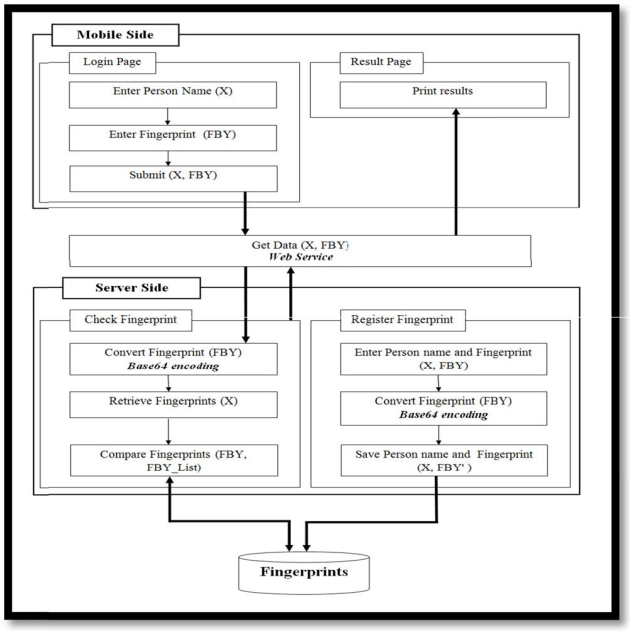

FIGURE 36. Applying to process on the server-side.

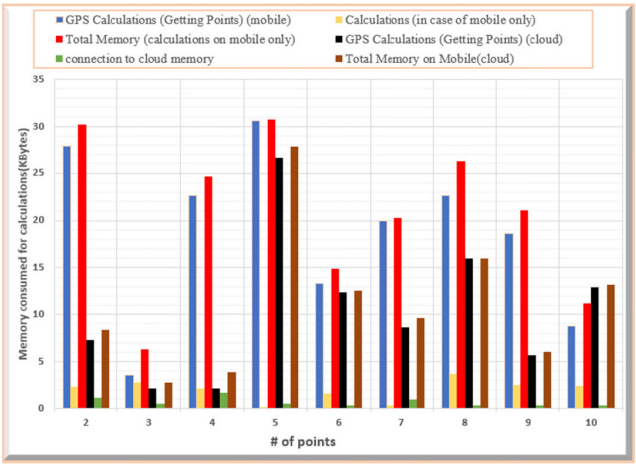

FIGURE 37. Total Memory consumed (Kbytes) for running experiment three.

different numbers of point's and a different number of parameters. the experiment done either the application run totally on mobile devices only or running using the proposed framework.

In the first step: comparison is conducted using two different types of GPS mode (using mobile GPS) and using the mobile network.

In the second step: the GPS mode of operations, we have to choose between manual or automatic calculation to get latitude or longitude for each point.

- If automatic calculation is selected, we have to enter the number of points, and the system gets points every **forty** second
- If manual calculation is selected, we have to click to get points

TABLE 2. GPS application calculations steps.

Step no.	*without security*	*with security AES /RSA encryption*	*with security - hybrid cryptography algorithms*
1	GPS reading the latitude and longitude for each point either by GPS for mobile (smartphone/ satellite) or from the mobile network (i.e., this step execute on a mobile device).		
2	The data (longitude and latitude for each point) is transmitted to the cloud server to perform a calculation on the cloud.	The data (longitude and latitude for each point) is encrypted using AES and then transmitted to the cloud server to perform a calculation on the cloud.	The data (longitude and latitude for each point) is encrypted using a hybrid cryptography algorithm and then transmitted to the cloud server to perform a calculation on the cloud.
2.1		Then Ciphertext is decrypted on the cloud.	
3	The different calculations such as the distance between two points or more using different algorithms calculations performed on the		
4	Calculating the results and the consuming resources to send and receive results for all processes.		
5		Results are Encrypted using RSA and send to the mobile device.	Results are Encrypted using hybrid cryptography algorithms and send to the mobile device.
6		The mobile device is decrypted cypher text. The application will perform calculations on a cloud server. Also , it calculates the results and the consuming resources such as Memory consumed for sending and receiving results, Memory consumed for distance calculations only, memory consumed for all process from getting points till receiving results, CPU usage, Time consumed for calculation, battery consumed to perform the transmitting data, time consumed for calculations and getting points.	

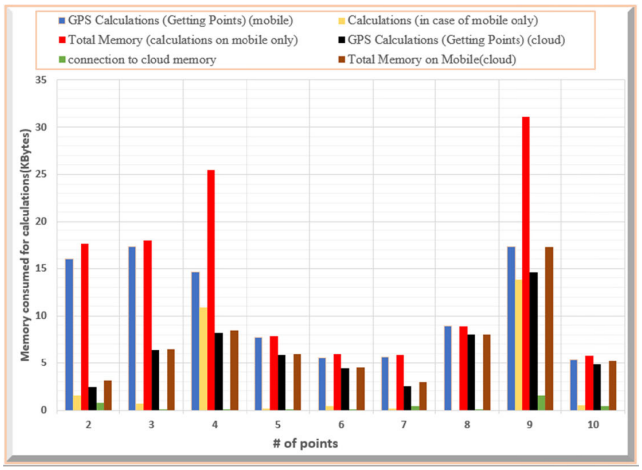

FIGURE 38. Memory Consumed (KBYTES) for running Experiment two.

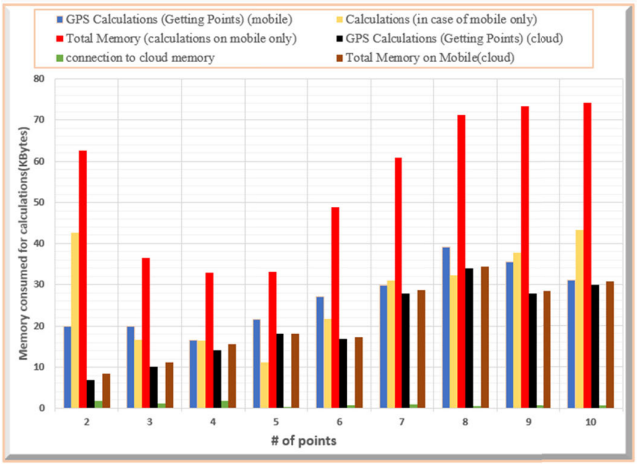

FIGURE 39. Total Memory consumed (Kbytes) for running experiment three.

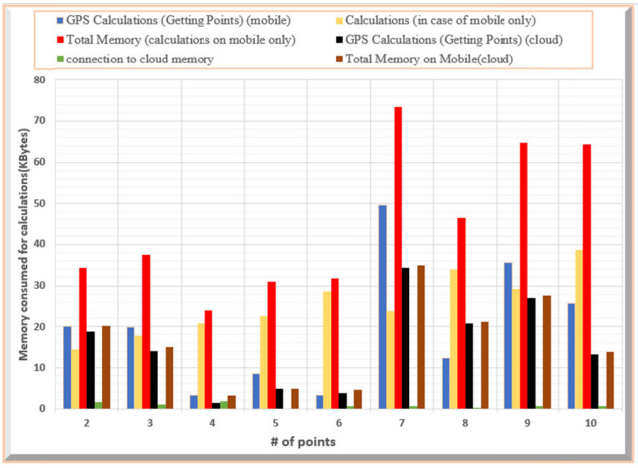

FIGURE 40. Total Memory consumed (Kbytes) for running experiment four.

TABLE 3. Memory consumed for execution application on mobile smartphone and cloud web services (Kbytes) (Getting points using GPS Satellite).

Mobile Calculations				Cloud Calculations				
# points	GPS Calculations (Getting Points)	Calculations	Total Memory	# of points	GPS Calculations (Getting Points)	connection to cloud memory	Memory consumed for calculation only	Total Memory on Mobile
2	27.9	2.3	30.2	2	7.3	1.1	138.3	8.4
3	3.6	2.7	6.3	3	2.1	0.5	137	2.7
4	22.6	2.1	24.7	4	2.1	1.7	139.7	3.9
5	30.6	0.1	30.7	5	26.6	0.5	139.7	27.9
6	13.3	1.6	14.9	6	12.4	0.3	141	12.6
7	20	0.3	20.3	7	8.6	0.9	141	9.6
8	22.6	3.7	26.3	8	16	0.3	142.3	16
9	18.6	2.5	21.1	9	5.7	0.3	142.3	6
10	8.8	2.4	11.2	10	12.9	0.3	143.6	13.2

In the third step: After selecting a method to get points either manually or automatic, we have to choose between calculation way on mobile or by the proposed framework

a) ***In the case of calculation on a mobile device only u***, the calculation is directed using the following steps

1. GPS reading to determine latitude and longitude for each point either by GPS for mobile (smartphone/satellite) or from a mobile network.
2. Then calculate the distance between two points or more using different algorithms.

TABLE 4. Memory consumed for execution application on mobile smartphone and cloud web services (getting points using network GPS).

Mobile Calculations				MCC calculations				
# points	GPS Calculations (Getting Points)	Calculations	Total Memory	# of points	GPS Calculations (Getting Points)	connection to cloud memory	Memory consumed for calculation only	Total Memory on Mobile
2	16	1.6	17.6	2.4	0.8	3.1	2	16
3	17.3	0.7	18	6.4	0.1	6.5	3	17.3
4	14.6	10.9	25.5	8.2	0.1	8.5	4	14.6
5	7.7	0.2	7.9	5.9	0.1	6	5	7.7
6	5.6	0.4	6	4.4	0.1	4.5	6	5.6
7	5.7	0.2	5.9	2.5	0.4	2.9	7	5.7
8	8.9	0	8.9	8	0.1	8	8	8.9
9	17.3	13.8	31.1	14.6	1.6	17.3	9	17.3
10	5.3	0.5	5.8	4.8	0.4	5.2	10	5.3

3. The Application will perform all calculations on a smartphone device and calculate the results and the consuming resources such as Memory consumed, CPU usage, Time consumed for calculation, battery consumed to
4. Calculate the resources consumed for running all applications methods on the mobile-only.

b) ***In the case of partition and offloading calculation on a cloud and mobile***, the comparison is conducted using three different types of operations.

We implement cloud clone application that enables the mobile applications developers to decide to perform all application processes on an Android mobile device or to divide the application processes to execute on mobile and cloud. The framework uses security techniques (***AES /RSA encryption and hybrid cryptography algorithms)*** and without using security as illustrated as in Table 2.

- **Mathematical Calculations**
  - **Distance using Haversine formula:**

In this experiment, distance calculations between two point using the 'haversine' formula is used to calculate the

TABLE 5. Memory consumed(Kbytes) manual calculation for getting points using GPS satellite.

Calculations on mobile Smartphone				Calculations on Cloud				
# of points	GPS Memory consumed	Calculation Memory consumed	Total Memory	# of points	GPS memory consumed	connection to cloud memory	Memory consumed for calculation only	Total Memory on Mobile
2	20	42.5	62.5	6.7	1.8	8.5	2	20
3	19.9	16.6	36.5	10.1	1.1	11.2	3	19.9
4	16.5	16.4	32.9	14.1	1.7	15.7	4	16.5
5	21.7	11.3	33	18.1	0.1	18.2	5	21.7
6	27.1	21.7	48.8	16.9	0.6	17.4	6	27.1
7	29.8	31	60.8	27.9	0.8	28.7	7	29.8
8	39	32.3	71.3	34	0.4	34.4	8	39
9	35.6	37.7	73.3	27.8	0.6	28.4	9	35.6
10	31	43.2	74.2	30	0.7	30.7	10	31

great-circle distance between two points

$$\mathbf{a} = \mathbf{sin^2(\Delta\varphi/2) + cos(\varphi_1).cos(\varphi_2).sin^2(\Delta\lambda/2)} \quad (16)$$

$$\mathbf{c} = \mathbf{2.atan2(\sqrt{a}, \sqrt{(1-a)})} \quad (17)$$

$$\mathbf{d} = \mathbf{R.c} \quad (18)$$

where $\Delta\varphi$ is latitude difference (lat2− lat1), $\Delta\lambda$ is longitude difference (long2− long1), R is earth's radius(mean radius = 6,371km)

- **Distance using Spherical low of Cosines: (spherical law of cosines formula)**

$$d = \text{acos}(\sin(\varphi 1).\sin(\varphi 2) + \cos(\varphi 1).\cos(\varphi 2).\cos(\Delta\lambda)).R \quad (19)$$

TABLE 6. Memory consumed(Kbytes) manual calculation for getting points using network GPS.

Mobile Calculations				Calculations on Cloud				
# of points	GPS Memory consumed	Calculation Memory consumed	Total Memory	# of points	GPS memory consumed	connection to cloud memory	Memory consumed for calculation only	Total Memory on Mobile
2	20	14.4	34.4	18.7	1.5	20.2	2	20
3	19.9	17.7	37.6	14.1	1	15.1	3	19.9
4	3.2	20.7	23.9	1.4	1.8	3.2	4	3.2
5	8.4	22.5	30.9	4.8	0	4.8	5	8.4
6	3.2	28.6	31.8	3.8	0.7	4.5	6	3.2
7	49.7	23.7	73.4	34.3	0.7	35	7	49.7
8	12.4	33.9	46.3	20.8	0.3	21.2	8	12.4
9	35.6	29.2	64.8	27	0.6	27.7	9	35.6
10	25.7	38.7	64.4	13.2	0.7	13.9	10	25.7

- **Low of Cosines Distance using Equirectangular Approximation:**

$$\mathbf{x} = \mathbf{\Delta\lambda.cos(\varphi)} \quad (20)$$

$$\mathbf{y} = \mathbf{\Delta\varphi} \quad (21)$$

$$\mathbf{d} = \boldsymbol{R}.\sqrt{(\boldsymbol{x}^2 + \boldsymbol{y}^2)} \quad (22)$$

- **Bearing**

$$\theta = \text{atan}_2(\sin(\Delta\lambda).\cos(\varphi_2), \cos(\varphi 1).\sin(\varphi_2) - \sin(\varphi_1).\cos(\varphi_2).\cos(\Delta\lambda)) \quad (23)$$

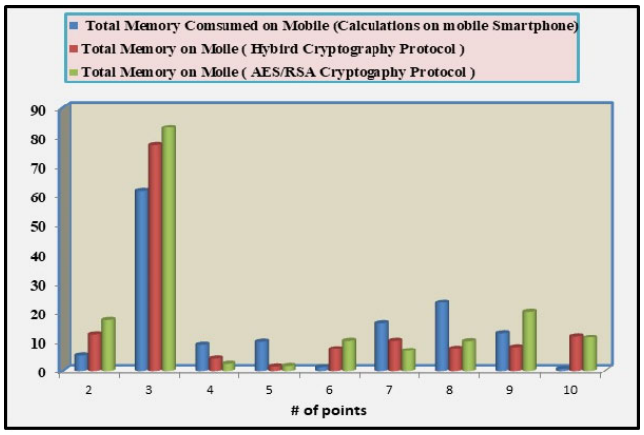

FIGURE 41. Total Memory Consumed (Kbytes) on Mobile Only for Execution Application on Cloud Web Services (Getting points automatically using GPS Satellite).

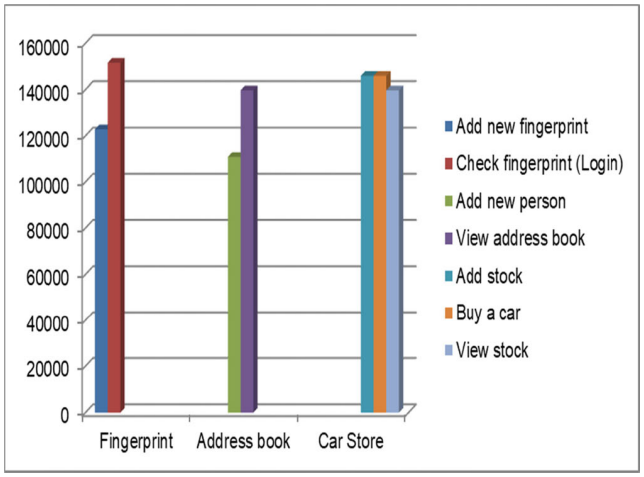

FIGURE 42. – Memory consumption in mobile.

B. RESULT FOR GPS APPLICATION

1) EXPERIMENT ONE: GETTING POINTS USING GPS SATELLITE WITHOUT SECURITY (AUTOMATIC CALCULATIONS)

Table 3 shows resources consumed in case of automatic calculation in case of points **automatically every forty seconds using GPS satellite**. The number of points range from two points till ten points (GPS calculation on mobile smartphone and calculation migrated to cloud and return results

TABLE 7. Comparing the performance while executing the two methods on the mobile site.

Function	Average time (s)	Average memory (Bytes)	Average processor cycles	Function
Add a new fingerprint	6.2	122880	11	182.81
Check a fingerprint (Login)	16.9	151552	22	225.47

TABLE 8. Comparing the performance while executing the two methods on the server side.

Function	Average time (s)	Average memory (Bytes)	Average processor cycles	Function
Add a new fingerprint	3.2	47523	9	70.70
Check a fingerprint (Login)	5.6	192768	16	286.79

TABLE 9. Performance while executing the 'add new fingerprint' method on the mobile side and the 'check fingerprint (log in)' method on the server side.

Function	Average time (s)	Average memory (Bytes)	Average processor cycles	Function
Add a new fingerprint **mobile site**	13.2	123380	12	138.55
Check a fingerprint (Login) **server-side**	12.6	193118	18	287.31

TABLE 10. Comparing the performance while executing the two methods on the mobile site.

Function	Average time (s)	Average memory (Bytes)	Average processor cycles	Function
Add a new person	6.2	110890	9	164.97
View address book	15.6	139754	6	207.91

TABLE 11. Comparing the performance while executing the two methods on the server side.

Function	Average time (s)	Average memory (Bytes)	Average processor cycles	Function
Add a new person	4.2	36784	6	54.72
View address book	7.1	176485	9	262.56

TABLE 12. Performance while executing the 'add new person' method on the mobile side and 'view address book' method on the server side.

Function	Average time (s)	Average memory (Bytes)	Average processor cycles	Function
Add a new person Mobile site	11.3	110922	9	192.12
View address book Server-side	9.6	176567	9	280.76

TABLE 13. Comparing the performance while executing the two methods on the mobile site.

Function	Average time (s)	Average memory (Bytes)	Average processor cycles	Function
Add stock	7.3	120970	9	189.77
Buy a car	12.6	145879	6	250.46
View stock	15.3	139752	12	223.11

TABLE 14. Comparing the performance while executing the two methods on the server side.

Function	Average time (s)	Average memory (Bytes)	Average processor cycles	Function
Add stock	5.4	35543	6	40.31
Buy a car	4.9	161125	9	266.97
View stock	6.1	186974	9	290.03

TABLE 15. Performance while executing the 'add new person' method on the mobile side and view address book method on the server side.

Function	Average time (s)	Average memory (Bytes)	Average processor cycles	Function
Add stock **mobile site**	12.3	121080	9	199.33
Buy a car **mobile site**	17.6	146779	6	255.76
View stock **server-side**	11.1	189733	9	300.13

TABLE 16. Result for all applications.

Application name	# of functions	Function names	Longest time	Functions run on the server	Functions run on mobile
Finger print	2	1- adding a new fingerprint 2- check fingerprint	check fingerprint	check fingerprint	adding new fingerprint
Address book	2	1- adding a new person 2- display contacts	display contacts	display contacts	adding new person
Car Store	3	1- add stock 2- buy a car 3- view stock	View stock	View stock	1- add stock 2- buy a car

to mobile) in case of distance range from approximately 100 meters tall 400 meters either in case of all calculation done on mobile device or application is partitioned and offloading on cloud to perform distance calculation on cloud.

TABLE 17. Performance while executing the 'add new person' method on the mobile side and view address book method on the server side.

Memory Consumed (Bytes)					
		Mobile Calculations		Cloud Calculations	
Projects	Methods	Memory Consumed	Total Memory	Memory Consumed	Total Memory
Fingerprint	**Add new fingerprint**	122880	274432	47523	240291
	Check fingerprint (Login)	151552		192768	
Address book	**Add a new person**	110890	250644	36784	213269
	View address book	139754		176485	
Car Store	**Add stock**	120970	406601	35543	383642
	Buy a car	145879		161125	
	View stock	139752		186974	

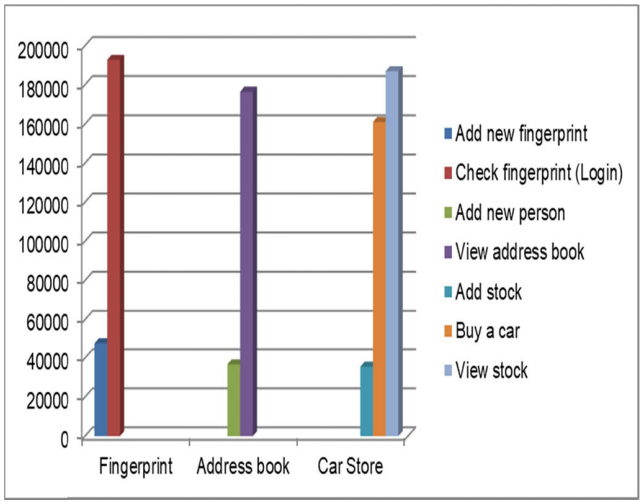

FIGURE 43. Memory consumption in cloud.

- **Results analysis for getting Points Automatically (GPS Satellite)**

Figure 37 shows the performance of executing the application on mobile or cloud in terms of memory consumed using different setting. the results show that the resources consumed on a mobile smartphone in case of MCC will **decrease approximately 48%** of memory consumed for running application on the mobile-only. Most of the resources are consumed on the cloud and minimize the resources consumed in the mobile smartphone.

2) EXPERIMENT TWO: GETTING POINTS USING NETWORK GPS USING AUTOMATIC CALCULATIONS WITHOUT SECURITY

Table 4, and Figure 38 show the resources consumed in case of **automatic calculation**(getting points every 40-second

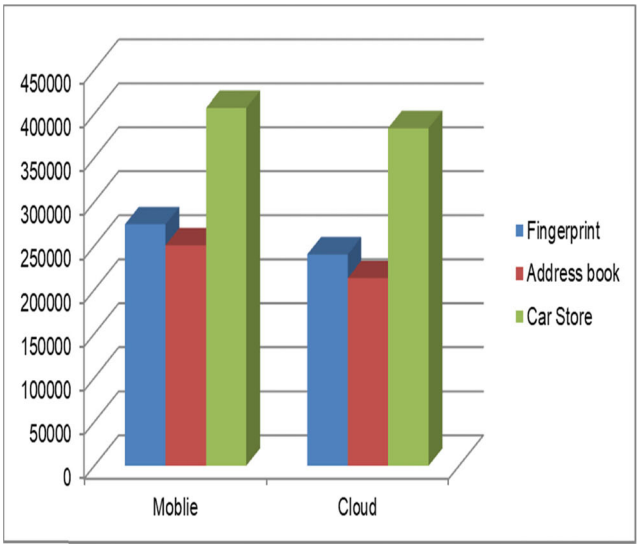

FIGURE 44. Comparison between memory consumption in mobile and cloud.

TABLE 18. Average time consumed for all projects.

Average time consumed (Seconds)					
		Mobile Calculations		Cloud Calculations	
Projects	Methods	Average time Consumed	Total time	Average time Consumed	Total time
Fingerprint	Add new fingerprint	6.2	23.1	3.2	8.8
	Check fingerprint (Login)	16.9		5.6	
Address book	Add new person	6.2	21.8	4.2	11.3
	View address book	15.6		7.1	
Car Store	Add stock	7.3	35.2	5.4	16.4
	Buy a car	12.6		4.9	
	View stock	15.3		6.1	

using **Network GPS** for execution application on cloud web services). The experimental setup as done as in Experiment (Automatic Calculations) getting Points using GPS satellite

- ***Results analysis for getting Points Automatically (Network GPS)***

The performance of executing the application on mobile or cloud in terms of memory consumed using different cases are determined. **The resources consumed on a mobile smartphone in case of MCC will decrease approximately to the half of memory consumed for running application on the mobile-only**.

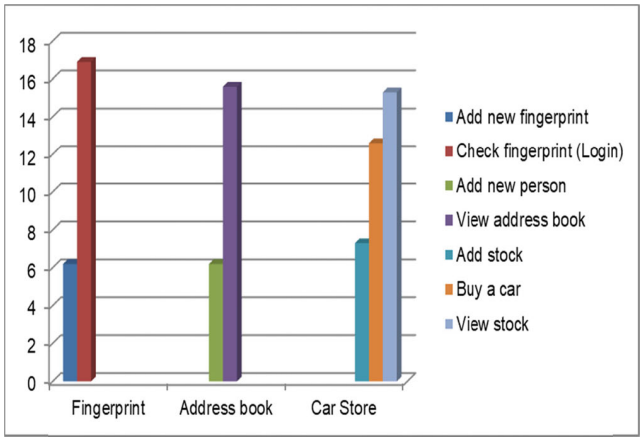

FIGURE 45. Average time consumption in mobile.

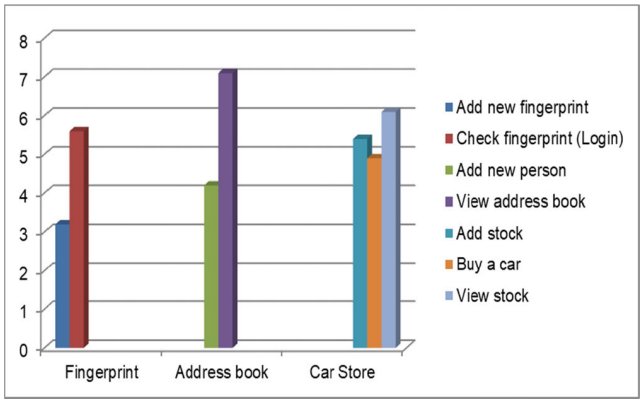

FIGURE 46. Average time consumption in cloud.

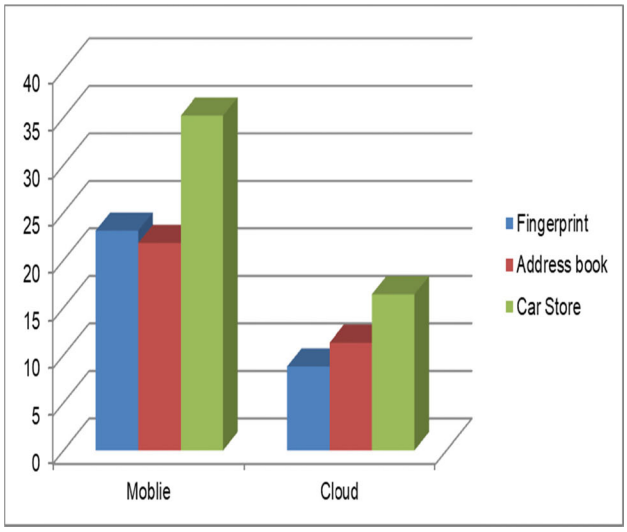

FIGURE 47. Comparison between average time consumption in mobile and cloud.

TABLE 19. Average processor cycles consumed for all projects.

Average processor cycles consumed (MHz)					
		Mobile Calculations		Cloud Calculations	
Projects	Methods	Average processor cycles Consumed	Total process or cycles	Average processor cycles Consumed	Total process or cycles
Fingerprint	Add new fingerprint	11	33	9	25
	Check fingerprint (Login)	22		16	
Address book	Add new person	9	15	6	15
	View address book	6		9	
Car Store	Add stock	9	27	6	24
	Buy a car	6		9	
	View stock	12		9	

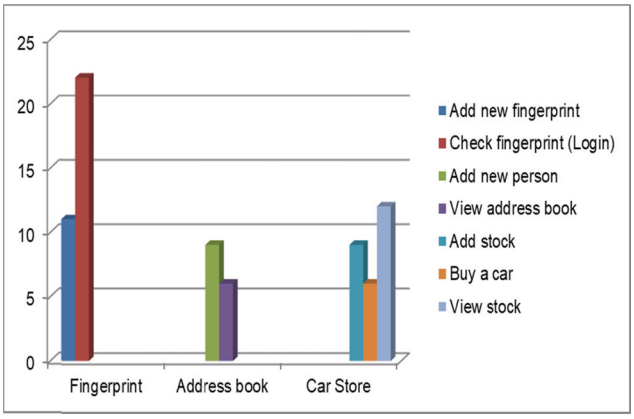

FIGURE 48. Average processor cycles consumption in mobile.

3) EXPERIMENT THREE: GETTING POINTS USING GPS SATELLITE USING MANUAL CALCULATIONS WITHOUT SECURITY

Table 5, and figure 39, show memory consumed (Kbytes) in case of getting the point **manually using GPS satellite** using the same setting as in the previous experiments

- ***Results analysis for getting Points manually using GPS satellite***

According to the partition algorithm, most of the resources consumed on a mobile smartphone will **decrease approximately 46% of memory consumed** for running application on the mobile-only.

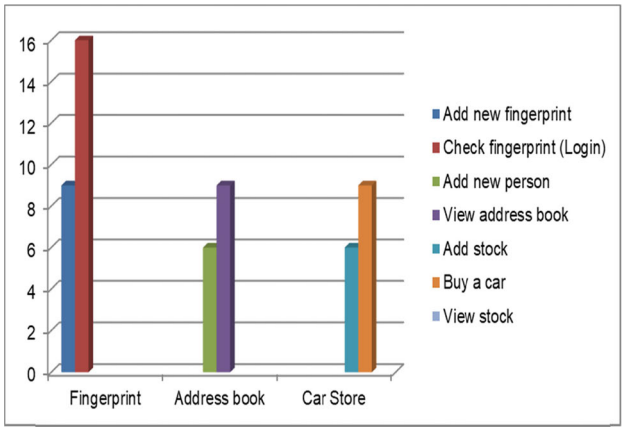

FIGURE 49. Average processor cycles consumption in cloud.

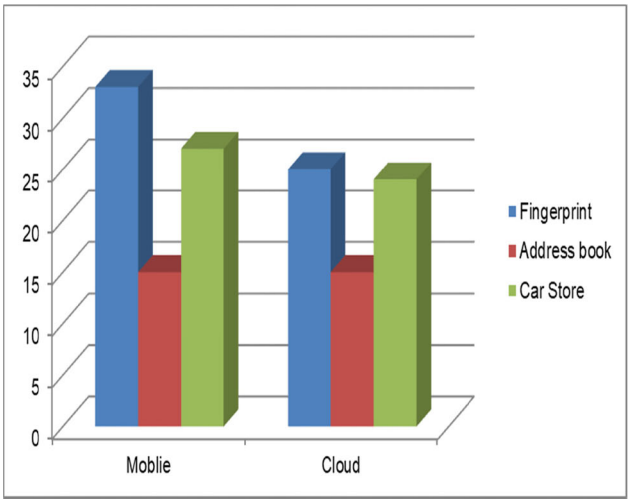

FIGURE 50. Comparison between average processor cycles consumption in mobile and cloud.

4) EXPERIMENT FOUR: GETTING POINTS USING NETWORK GPS USING MANUAL CALCULATIONS WITHOUT SECURITY

Table 6, figure 40, show the memory consumed (Kbytes) in case of getting the point **manually** using **Network GPS** for execution application in the same setting

- ***Results analysis for getting Points manually using Network GPS***

According to the partition algorithm, most of the resources consumed on a mobile smartphone will decrease to approximately 39% of memory consumed for running application on the mobile-only.

5) EXPERIMENT FIVE: GPS CALCULATIONS WITH SECURITY

a: EXPERIMENTAL (AUTOMATIC CALCULATIONS) GETTING POINTS USING GPS SATELLITE, AND NETWORK GPS

Figure 41 shows the resources consumed in case of automatic calculation in case of getting location automatically every 40 seconds using **GPS satellite** and taking security in con-

TABLE 20. Average power consumed for all projects.

Average power consumption (W)					
		Mobile Calculations		Cloud Calculations	
Projects	Methods	Average power Consumed	Total power	Average power Consumed	Total power
Fingerprint	Add new fingerprint	138.55	425.86	182.81	408.28
	Check fingerprint (Login)	287.31		225.47	
Address book	Add new person	164.97	372.88	54.72	317.28
	View address book	207.91		262.56	
Car Store	Add stock	189.77	663.34	40.31	597.31
	Buy a car	250.46		266.97	
	View stock	223.11		290.03	

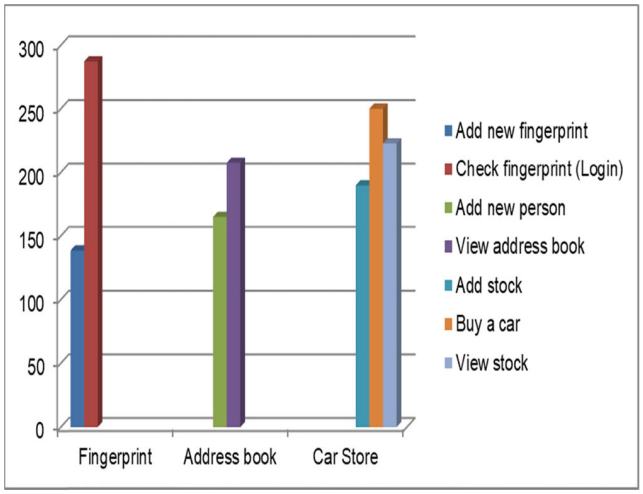

FIGURE 51. The average power consumed in mobile.

sideration (***with security AES/RSA encryption and with first proposed hybrid cryptography algorithm).*** Figure 42 shows the results using Network GPS.

- ***Results analysis for getting Points Automatically (GPS Satellite) (with security)***

- the partition algorithm using a different type of cryptography protocols or without using security shows

- According to the partition algorithm, most of the resources consumed on a mobile smartphone will increase approximately 20% of memory consumed for running application on the mobile-only.

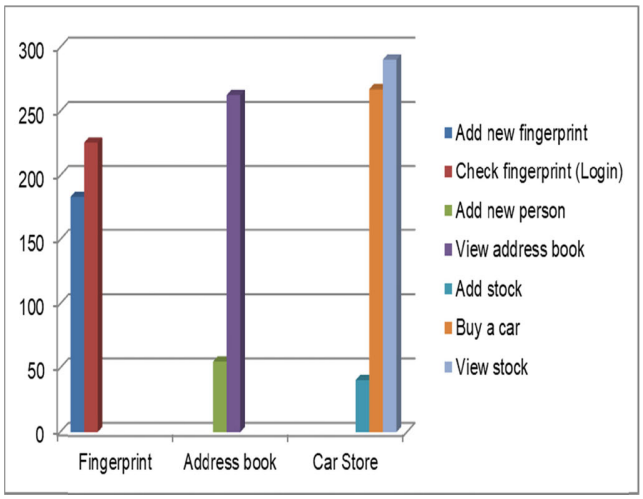

FIGURE 52. The average power consumed in the cloud.

b: RESULT FOR FINGERPRINT APPLICATION (2nd APPLICATION)

This system contains two functions:

1- Add a new fingerprint
2- Check a fingerprint (Login)

Where:

The average time in seconds is the mean value of the time taken to execute the method after running the system 50 times.

The average memory in bytes is the mean value of the memory utilized to execute the method after running the system 50 times.

The average CPU cycles are the mean value of the CPU cycles taken to execute the method after running the system 50 times.

The average power is the mean value of the power taken to execute the method after running the system 50 times.

c: RESULT FOR ADDRESS BOOK APPLICATION

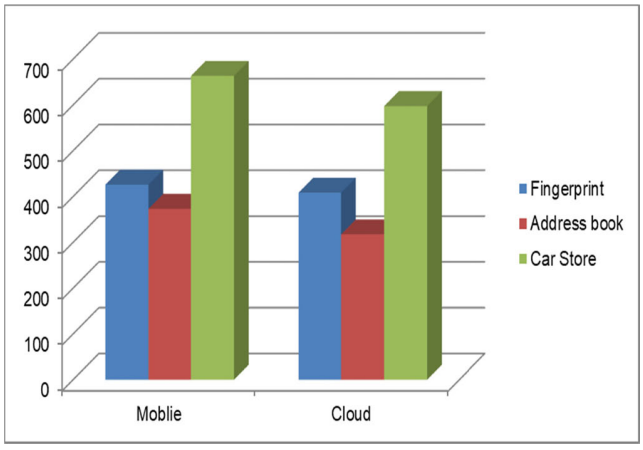

FIGURE 53. Comparison between average power consumed in mobile and cloud.

d: RESULT FOR CAR STORE APPLICATION

TABLE 21. Performance with encrypting the transferred data while executing the 'add new fingerprint' method on the mobile side and 'check fingerprint (log in)' method on the server side.

Function	Average time (s)	Average memory (Bytes)	Average CPU/processor cycles	Average power/battery consumption (W)
Add new fingerprint mobile site	18.3	123380	12	138.55
Check fingerprint (Login) server-side	17.6	193118	18	287.31

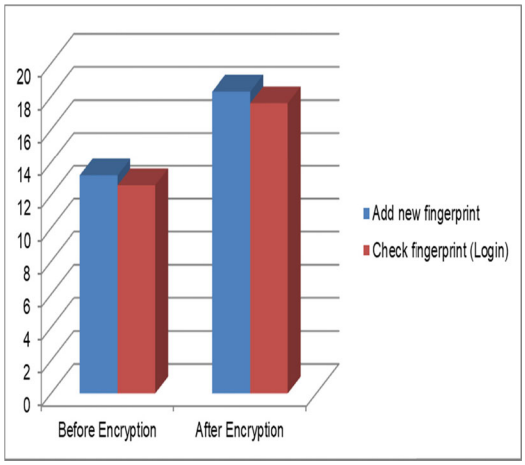

FIGURE 54. Time consumed before and after encryption in fingerprint.

e: RESULT MATRIX FOR ALL SYSTEMS

Table 16 illustrates the results for all the applications and shows the separation of methods implemented by the framework on the project's methods.

Table 17 shows all the memory consumed by the applications and the separation implemented by the framework on the project's methods.

f: COMPARISON BETWEEN TIMES CONSUMED AFTER USING THE FRAMEWORK AND APPLYING THE HYBRID CRYPTOGRAPHY METHODS.

- ***Fingerprint application***

TABLE 22. Performance with encrypting the transferred data while executing the 'add new fingerprint' method on the mobile side and 'check fingerprint (log in)' method on the server side.

Function	Average time (s)	Average memory (Bytes)	Average CPU/processor cycles	Average power/battery consumption (W)
Add a new person mobile site	16.3	111922	9	201.12
View address book server side	14.6	177567	9	290.76

- ***Address book application***

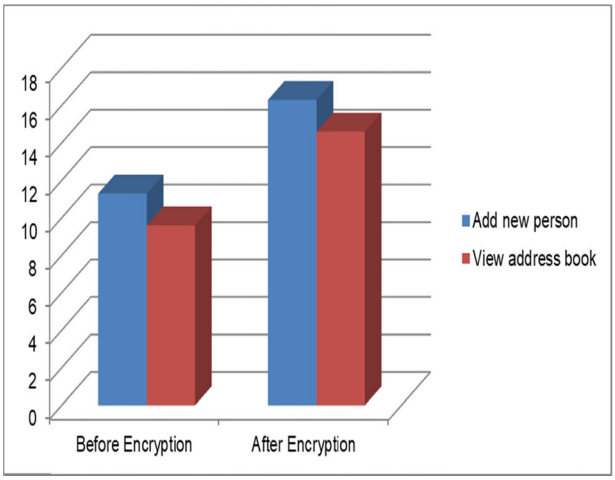

FIGURE 55. Time consumed before and after encryption in the address book.

- ***Car store application***

TABLE 23. Performance with encrypting the transferred data while executing the 'add new fingerprint' method on the mobile side and 'check fingerprint (log in)' method on the server side.

Function	Average time (s)	Average memory (Bytes)	Average CPU/processor cycles	Average power/battery consumption (W)
Add stock mobile site	17.3	122080	9	200.33
Buy a car mobile site	21.6	148779	6	260.76
View stock server-side	16.1	189993	9	320.13

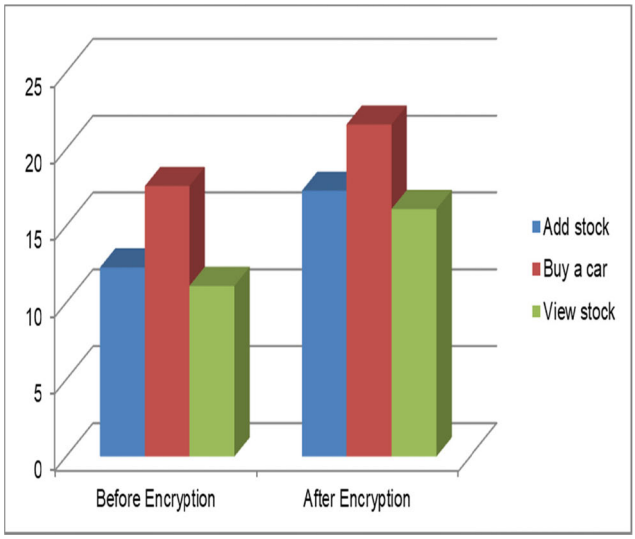

FIGURE 56. Time consumed before and after encryption in 'car store'.

V. CONCLUSION

In this paper, an optimized framework is proposed to improve the efficiency of offloading computation from the mobile device to the cloud. This framework can offload only the application methods that consume substantial mobile resources. This framework divides application processing methods into two sets, one running on the mobile site and the other running on the server-side based on the execution time of the separated methods. It also presents an extension for mobile cloud computing models by adding a hybrid cryptography method. When using the framework, the running of any application is separated into two parts. The framework contributes to developing both light and heavy mobile applications, such as GPS calculation, fingerprint, and face recognition. These applications can benefit from our proposed model while saving energy and improving performance compared to previous techniques. Also, this approach allows mobile applications to leverage cloud resources by allowing some of the services to run on the cloud, thus alleviating resource constraints stemming from the mobile devices themselves. Mobile applications can be readily divided into a group of services without modifying the application source code or the OS.

Finally, our framework protects service data on the cloud using security, hence minimizing outside risks. Using this framework and accounting for the added hybrid cryptography method, the aggregate processing time increased by 0.8 to 1 s.

This framework is tested using four mobile application. the first mobile application is GPS calculations. The first application performed GPS calculations.

Comparison is conducted using two different types of GPS mode (using mobile GPS), and using mobile network. for each type of GPS .longitude and latitude for each point can be obtained on it either manually or automatic.

Firstly: in case of getting Points using GPS satellite (automatic calculation)

- The memory consumed using different distance and number of points on a mobile smartphone will decrease approximately 48 % of memory consumed for running application on the mobile-only.

Secondly: in case of getting Points using GPS satellite (manual calculation)

- The resources consumed on a mobile smartphone will decrease approximately 46% of memory consumed for running application on the mobile-only.

Thirdly: in case of getting Points using Network GPS (automatic calculation)

- The memory consumed using different cases on a mobile smartphone will decrease approximately to the half of memory consumed for running application on the mobile-only.

Fourthly: in case of getting Points using Network GPS (manual calculation)

- The memory consumed using different cases on a mobile smartphone will decrease approximately 39% of memory consumed for running application on the mobile-only.

The second mobile application is a fingerprint. The system aims to store and retrieve data to and from the cloud efficiently. The system was developed based on two phases: first, applying all the processing steps to register a new user, converting, checking and saving fingerprints on the mobile side; then, applying all the processing steps to register a new user, converting and checking fingerprints, and saving them on the server-side while keeping the interface on the mobile side. Extensive experiments were performed to study the efficiency of the implemented system. The system was tested on various database sizes. The size of the images containing the fingerprints ranged from 10 kb to 480 kb. The two phases were tested, and the time is taken to apply all processing steps, including registering a new user, converting and checking fingerprints and saving fingerprints on the mobile side was calculated. The system ran 50 times for each database size to yield an average time. The result points to an average time taken to process the data of 16 seconds.

The time is taken by applying all the processing steps of registering a new user, converting and checking fingerprints and saving fingerprints on the server-side while maintaining an interface on the mobile side was calculated. The average time consumed to process the data is 4.11 seconds. The proposed system proves that applying all the processing steps of registering a new user, converting and checking fingerprints and saving fingerprints on the server-side while keeping the interface on the mobile side. It is more efficient than applying all the processing steps of registering a new user, converting and checking fingerprints and saving fingerprints on the server-side while keeping the interface on the mobile side.

REFERENCES

[1] B. Gao, L. He, X. Lu, C. Chang, K. Li, and K. Li, "Developing energy-aware task allocation schemes in cloud-assisted mobile workflows," in *Proc. IEEE Int. Conf. Comput. Inf. Technol.; Ubiquitous Comput. Commun.; Dependable, Autonomic Secure Comput.; Pervasive Intell. Comput.*, Oct. 2015, pp. 1266–1273.

[2] I. Elgendy, W. Zhang, C. Liu, and C.-H. Hsu, "An efficient and secured framework for mobile cloud computing," *IEEE Trans. Cloud Comput.*, to be published.

[3] R. D. Shobha, M. Pounambal, and V. Saritha, "An efficient algorithm for dynamic task offloading using cloudlets in mobile cloud computing," *Int. J. Commun. Syst.*, vol. e3890, pp. 1–10, Jan. 2019.

[4] D. S. A. Elminaam, H. M. A. Kader, M. M. Hadhoud, and S. M. El-Sayed, "Elastic framework for augmenting the performance of mobile applications using cloud computing," in *Proc. 9th Int. Comput. Eng. Conf. (ICENCO)*, Dec. 2013, pp. 134–141.

[5] A. Botta, W. De Donato, V. Persico, and A. Pescapé, "Integration of cloud computing and Internet of Things: A survey," *Future Gener. Comput. Syst.*, vol. 56, pp. 684–700, Mar. 2016.

[6] L. Yang, J. Cao, S. Tang, D. Han, and N. Suri, "Run time application repartitioning in dynamic mobile cloud environments," *IEEE Trans. Cloud Comput.*, vol. 4, no. 3, pp. 336–348, Jul./Sep. 2016.

[7] Y. Wang, R. Chen, and D.-C. Wang, "A survey of mobile cloud computing applications: Perspectives and challenges," *Wireless Pers. Commun.*, vol. 80, no. 4, pp. 1607–1623, 2015.

[8] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2013.

[9] T. Xing, D. Huang, S. Ata, and D. Medhi, "MobiCloud: A geo-distributed mobile cloud computing platform," in *Proc. 8th Int. Conf. Netw. Service Manage.*, Oct. 2012, pp. 164–168,

[10] X. Chen, "Decentralized computation offloading game for mobile cloud computing," *IEEE Trans. Parallel Distrib. Syst.*, vol. 26, no. 4, pp. 83–974, Apr. 2014.
[11] M. V. Barbera, S. Kosta, A. Mei, and J. Stefa, "To offload or not to offload? The bandwidth and energy costs of mobile cloud computing," in *Proc. IEEE INFOCOM*, Apr. 2013, pp. 1285–1293.
[12] B.-G. Chun, S. Ihm, P. Maniatis, M. Naik, and A. Patti, "CloneCloud: Elastic execution between mobile device and cloud," in *Proc. 6th Conf. Comput. Syst.*, Apr. 2011, pp. 301–314.
[13] S. Kosta, A. Aucinas, P. Hui, R. Mortier, and X. Zhang, "ThinkAir: Dynamic resource allocation and parallel execution in the cloud for mobile code offloading," in *Proc. IEEE INFOCOM*, Mar. 2012, pp. 945–953.
[14] W. Zhang, Y. Wen, K. Guan, D. Kilper, H. Luo, and W. DO, "Energy-optimal mobile cloud computing under stochastic wireless channel," *IEEE Trans. Wireless Commun.*, vol. 12, no. 9, pp. 4569–4581, Sep. 2013.
[15] N. Fernando, S. W. Loke, and W. Rahayu, "Mobile cloud computing: A survey," *Future Generat. Comput. Syst.*, vol. 29, no. 1, pp. 84–106, 2013.
[16] I. Giurgiu, O. Riva, D. Juric, I. Krivulev, and G. Alonso, "Calling the cloud: Enabling mobile phones as interfaces to cloud applications," in *Proc. 10th ACM/IFIP/USENIX Int. Conf. Middleware*. Berlin, Germany: Springer-Verlag, Nov. 2009, p. 5.
[17] S. Yi, C. Li, and Q. Li, "A survey of fog computing: Concepts, applications and issues," in *Proc. Workshop Mobile Big*, Jun. 2015, pp. 37–42.
[18] A. U. R. Khan, M. Othman, S. A. Madani, and S. U. Khan, "A survey of mobile cloud computing application models," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 393–413, 1st Quart., 2014.
[19] C. Shi, V. Lakafosis, M. H. Ammar, and E. W. Zegura, "Serendipity: Enabling remote computing among intermittently connected mobile devices," in *Proc. 13th ACM Int. Symp. Mobile Ad Hoc Netw. Comput.*, Jun. 2012, pp. 145–154.
[20] W. Zhang, S. Han, H. He, and H. Chen, "Network-aware virtual machine migration in an overcommitted cloud," *Future Gener. Comput. Syst.*, vol. 76, pp. 428–442, Nov. 2017.
[21] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Proc. Int. Conf. Mobile Comput., Appl., Services*, vol. 76, 2010, pp. 59–79.
[22] E. Cuervo, A. Balasubramanian, D.-K. Cho, A. Wolman, S. Saroiu, R. Chandra, and P. Bahl, "MAUI: Making smartphones last longer with code offload," in *Proc. Int. Conf. Mobile Syst., Appl., Services*, 2010, pp. 49–62.
[23] D. Kovachev, T. Yu, and R. Klamma, "Adaptive computation offloading from mobile devices into the cloud," in *Proc. IEEE 10th Int. Symp. Parallel Distrib. Process. Appl.*, Jul. 2012, pp. 784–791.
[24] F. Xia, F. Ding, J. Li, X. Kong, L. T. Yang, and J. Ma, "Phone2Cloud: Exploiting computation offloading for energy saving on smartphones in mobile cloud computing," *Mobile Cloud Comput. Inf. Syst. Frontiers*, vol. 16, no. 1, pp. 95–111, Mar. 2014.
[25] M. Alkhalaileh, R. N. Calheiros, Q. V. Nguyen, and B. Javadi, "Dynamic resource allocation in hybrid mobile cloud computing for data-intensive applications," in *Proc. Int. Conf. Green, Pervasive, Cloud Comput.* Cham, Switzerland: Springer, May 2019, pp. 176–191.
[26] M. Shiraz, A. Gani, A. Shamim, S. Khan, and R. W. Ahmad, "Energy efficient computational offloading framework for mobile cloud computing," *J. Grid Comput.*, vol. 13, no. 1, pp. 1–18, 2015.
[27] B. Zhou, A. V. Dastjerdi, R. N. Calheiros, S. N. Srirama, and R. Buyya, "A context sensitive offloading scheme for mobile cloud computing service," in *Proc. IEEE Int. Conf. Cloud Comput.*, Jun./Jul. 2015, pp. 869–876.
[28] S. Guo, B. Xiao, Y. Yang, and Y. Yang, "Energy-efficient dynamic offloading and resource scheduling in mobile cloud computing," in *Proc. IEEE INFOCOM*, Apr. 2016, pp. 1–9.
[29] W.-Z. Zhang, H.-C. Xie, and C.-H. Hsu, "Automatic memory control of multiple virtual machines on a consolidated server," *IEEE Trans. Cloud Comput.*, vol. 5, no. 1, pp. 2–14, Jan./Mar. 2017.
[30] Y. Li, M. Chen, W. Dai, and M. Qiu, "Energy optimization with dynamic task scheduling mobile cloud computing," *IEEE Syst. J.*, vol. 11, no. 1, pp. 96–105, Mar. 2017.
[31] D. S. Abdul, "Reliable the resources of mobile devices in cloud computing," *Int. J. Adv. Comput. Technol.*, vol. 10, no. 1, pp. 61–70, Mar. 2018.
[32] A. A. Taha, D. S. A. Elminaam, and K. M. Hosny, "An improved security schema for mobile cloud computing using hybrid cryptographic algorithms," *Far East J. Electron. Commun.*, vol. 18, no. 4, pp. 521–546, Apr. 2018.
[33] S. M. El-Sayed, H. M. A. Kader, M. M. Hadhoud, and D. S. A. Elminaam, "Mobile cloud computing framework for elastic partitioned/modularized applications mobility," *Int. J. Electron. Inf. Eng.*, vol. 1, no. 2, pp. 53–63, Dec. 2014.
[34] D. S. A. Elminaam, H. M. A. Kader, M. M. Hadhoud, and M. S. El-Sayed, "GPS test performance: Elastic execution applications between mobile device and cloud to reduce power consumption," *Int. J. Comput. Sci. Netw. Secur.*, vol. 13, no. 12, pp. 6–13, Dec. 2013.
[35] N. Vallina-Rodriguez and J. Crowcroft, "Energy management techniques in modern mobile handsets," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 1, pp. 179–198, 1st Quart., 2013.
[36] D. S. A. Elminaam, H. M. A. Kader, M. M. Hadhoud, and S. M. El-Sayed, "Increase the performance of mobile smartphones using partition and migration of mobile applications to cloud computing," *Int. J. Technol. Enhancements Emerg. Eng. Res.*, vol. 2, no. 5, pp. 1–10, May 2014.
[37] M. Sulochana and O. Dubey, "Preserving data confidentiality using multi-cloud architecture," in *Proc. 2nd Int. Symp. Big Data Cloud Comput. (ISBCC)*, vol. 3, no. 4, 2015, pp. 1–6.
[38] L. Tawalbeh and N. S. Darwazeh, "A secure cloud computing model based on data classification," *Procedia Comput. Sci.*, vol. 52, pp. 1153–1158, 2015.
[39] P. Ratha, D. Swain, B. Paikaray, and S. Sahoo, "An optimized encryption technique using an arbitrary matrix with probabilistic encryption," in *Proc. 3rd Int. Conf. Recent Trends Comput. (ICTRC)*, vol. 5, no 4, 2015, pp. 1–7.
[40] K. El Makkaoui and A. Beni-Hssane, "Fast cloud-RSA scheme for promoting data confidentiality in the cloud computing," *Procedia Comput. Sci.*, vol. 113, pp. 33–40, 2017.
[41] V. Ponnuramu and L. Tamilselvan, "Encryption for massive data storage in cloud," *Comput. Intell. Data Mining*, vol. 2, no. 2, pp. 27–37, 2015.
[42] Z. Kartit, A. Azougaghe, H. K. Idrissi, M. El Marraki, M. Hedabou, M. Belkasmi, and A. Kartit, "Applying encryption algorithm for data security in cloud storage," in *Advances in Ubiquitous Networking*, vol. 3, no. 3. Singapore: Springer, 2016.
[43] N. Sengupta and R. Chinnasamy, "Contriving hybrid DESCAST algorithm for cloud security," in *Proc. 11th Int. Multi-Conf. Inf. Process.*, vol. 5, no. 2, 2015, pp. 1–10.
[44] N. Balkish, A. M. Prasad, and V. Suma, "An efficient approach to enhance data security in cloud using recursive blowfish algorithm," in *Proc. 48th Annu. Conv. Comput. Soc. India*. Cham, Switzerland: Springer, vol. 1, no. 1, 2014, pp. 575–582.
[45] M. M. Poteya, C. A. Dhote, and D. H. Sharmac, "Homomorphic encryption for security of cloud data," in *Proc. 7th Int. Conf. Commun.*, vol. 1 no. 1, 2016, pp. 1–7.
[46] L. Xiong, Z. Xu, and Y. Xu, "A secure re-encryption scheme for data services in a cloud computing environment," *Concurrency Comput., Pract. Exper.*, vol. 27, no. 17, pp. 4573–4585, 2015.
[47] S. K. S. ShaluMall, "A new security framework for cloud data," in *Proc. 8th Int. Conf. Adv. Comput. Commun.*, vol. 1, no. 1, 2018, pp. 1–11.
[48] V. R. Balasaraswathi and S. Manikandan, "Enhanced security for multi-cloud storage using cryptographic data splitting with dynamic approach," in *Proc. IEEE Int. Conf. Adv. Commun., Control Comput. Technol.*, vol. 2, no 3, May 2014, pp. 1190–1194.
[49] S. P. Kumar and R. Subramanian, "An efficient and secure protocol for ensuring data storage security in cloud computing," *Int. J. Comput. Sci. Issues*, vol. 8, no. 1, p. 261, 2011.
[50] A. Sachde and M. Bhansali, "Enhancing cloud computing security using AES algorithm," *Int. J. Comput. Appl.*, vol. 67, no. 9, pp. 1–5, 2013.
[51] Z. Sanaei, S. Abolfazli, A. Gani, and R. Buyya, "Heterogeneity in mobile cloud computing: Taxonomy and open challenges," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 1, pp. 369–392, 1st Quart., 2013.
[52] I. A. Elgendy, M. Elkawkagy, and A. Keshk, "An efficient framework to improve the performance of mobile applications," *Int. J. Digit. Content Technol. Appl.*, vol. 9, no. 5, pp. 43–54, 2015.
[53] D. S. A. Elminaam, F. T. Elanezi, and K. M. Hosny, "An efficient framework for mobile cloud computing," in *Proc. 32th Int. Bus. Inf. Manage. Assoc. (IBIMA)*, Seville, Spain, Nov. 2018, pp. 5783–5796.
[54] D. S. A. Elminaam and Y. M. Wazery, "Resource sharing security in cloud computing environment," *Int. Arab J. e-Technol.*, vol. 5, no. 2, pp. 47–57, Jun. 2018.
[55] F. Liu, P. Shu, H. Jin, L. Ding, J. Yu, D. Niu, and B. Li, "Gearing resource-poor mobile devices with powerful clouds: Architectures, challenges, and applications," *IEEE Wireless Commun.*, vol. 20, no. 3, pp. 14–22, Jun. 2013.

[56] A. A. Taha, E. Dr-Diaa S. S. Abdelminaam, M. Khalid, and K. Hosny, "Enhancement the security of cloud computing using hybrid cryptography algorithms," *Int. J. Adv. Comput. Technol.*, vol. 9, no. 3, pp. 36–42, Dec. 2017.
[57] P. Garg and V. Sharma, "An efficient and secure data storage in mobile cloud computing through RSA and Hash function," in *Proc. Int. Conf. Issues Challenges Intell. Comput. Techn. (ICICT)*, Feb. 2014, pp. 334–339.

DIAA SALAMA ABD ELMINAAM was born in 1982, KafrSakr, Sharkia, Egypt. He received the B.Sc. degree (Hons.) from the Faculty of Computers & Informatics, Zagazig University, Egypt, in 2004, and the master's degree in information system, specializing in cryptography and network security, and the Ph.D. degree in information system from the Faculty of Computers and Information, Menufia University, Egypt, in 2009 and 2015, respectively. He has been an Assistance Professor with the Information Systems Department, Faculty of Computers and Information, Benha University, Egypt, since 2011. He has worked on several research topics. He has contributed more than 40 technical articles in the areas of wireless networks, wireless network security, information security, Internet applications, cloud computing, mobile cloud computing, the Internet of Things, and machine learning in international journals, international conferences, local journals, and local conferences. His major research interests include cryptography, network security, the IoT, big data, cloud computing, and deep learning.

FARAH TURKEY ALANEZI was born in 1986, Kuwait. She received the B.Sc. degree from the Faculty of Information Technology and Computing-Arab Open University, Kuwait, in 2009. She is currently pursuing the master's degree with the Faculty of Computer and Information, Benha University, Egypt, in 2016. She has worked on several research topics. She has contributed more than 2 technical articles in the areas of cloud computing, and mobile cloud computing in international journals, and international conferences. Her major research interest includes security on mobile cloud computing.

KHALID M. HOSNY was born in 1966, Zagazig, Egypt. He received the B.Sc., M.Sc., and Ph.D. degrees from Zagazig University, Egypt, in 1988, 1994, and 2000, respectively, where he is currently a Professor in information technology with the Faculty of Computers and Informatics. From 1997 to 1999, he was a Visiting Scholar, University of Michigan, Ann Arbor, and University of Cincinnati, Cincinnati, USA. His research interests include image processing, pattern recognition, multimedia, computer vision, and cloud computing. He has published three edited books and more than 70 articles in international journals. He is a Senior Member of ACM. He is serving as an editor and a scientific reviewer for more than 35 international journals.