ARDS: A Model for Securing Data in Mobile Cloud Computing

Heba Sherif Sobhy, Amira Mohamed Kotb, Hesham N. Elmahdy

Information Technology Department, Faculty of Computers and Information

Cairo University, Cairo, Egypt.

Tel. (+202) 35674618, Fax. (+202) 33350109

Email: heba.sherif@fci-cu.edu.eg, a.kotb@fci-cu.edu.eg, ehesham@fci-cu.edu.eg

Abstract

For the last few years, mobile usage has grown significantly especially smartphones. Most people decided dispensing the use of their personal computers (i.e. desktops or laptops) and use mobile device instead. Because of mobile limitations, such as limited battery life, limited size of memory, and limited processor's power, the need for Mobile Cloud Computing (MCC) has been increased. The big issue of using MCC is data privacy and security. Data need to be secure while transmission and storage on cloud. This paper proposes a model to secure mobile user's data. This model uses AES, and RSA Digital Signature (ARDS) algorithms. This model provides confidentiality, authentication, and integrity of data stored on mobile cloud. Our model is compared with Garg & Sharma's model that uses RSA, Hashing, and DES algorithm.

Keywords: Mobile cloud computing, cloud, security, data storage, privacy, confidentiality, integrity, authentication.

1. Introduction

The question asked by anyone dealing with technology and business is, "What is cloud computing, and how it could be useful for my business?" [1].

Up to now, there is no accepted definition for cloud computing. The most widely used definition is the one defined by The National Institute of Standards and Technology (NIST), "Cloud computing is a model for enabling ubiquitous, convenient, on-demand network access to a shared pool of configurable computing resources (e.g., networks, servers, storage, applications, and services) that can be rapidly provisioned and released with minimal management effort or service provider interaction" [2]. The cloud computing paradigm provides resources to consumers as services. It is a successful paradigm for deploying web applications [3]. The strength of cloud computing is because its benefits like scalability, availability, flexibility, minimum cost, location independent, and so on [3, 4]. The main obstacle for cloud computing that limits its wide spread is data privacy and security.

For the last few years, the mobile usage has grown significantly especially smartphones. Most people decided dispensing the use of their personal computers (i.e. desktops or laptops) and use the mobile device instead. While increasing the use of mobile devices, users want larger storage capacity for their data. Mobile devices have limited resource such as limited battery life, limited size of memory, and limited processor's power [5, 6]. Because of these limitations, the need for Mobile Cloud Computing (MCC) has been increased. MCC integrates mobile computing, networking, and cloud computing [7]. By using MCC, most of the processing and data storage needed by application is moved from the mobile device to the cloud [8]. The big issue of using Mobile

Cloud Computing is data privacy and security. Data need to be secure while transmission and storage on cloud. There is a survey stated that 74% of IT Executives and Chief Information Officers are not willing to adopt cloud services because of the risks associated with security and privacy [8, 9, 10]. That's why several models and mechanisms have been presented by the researchers in the last few years [8]. The main target of this paper is to provide confidentiality, authentication, and integrity of data stored on the mobile cloud.

The remainder of this paper is organized as follows. Section 2 presents the related work to data security in MCC. The proposed model for data security in MCC is presented in section 3. Section 4 presents the evaluation of the proposed model and the results. The conclusion and future works are presented in section 5.

2. Related Work

In the last few years, many researches and mechanisms about data security in MCC are presented. Some of them are introduced in this section.

2.1. A new hybrid encryption protocol (NHEP) for securing data in MCC

Kader et al. [11] proposed a new hybrid encryption protocol (NHEP) to be used for MCC where ECC, AES, RSA, and Blowfish are used for authentication and confidentiality. MD5 is used for integrity. It works as follows:

Encryption Phase: The plaintext is divided into n blocks, each of length 128 bits. The first n/2 blocks are encrypted using AES, and ECC encrypts AES secret key. The second n/2 blocks are encrypted using Blowfish, and RSA encrypts Blowfish secret key. Hashing using MD5 is applied to both ciphertexts.

Decryption Phase: The ciphertext is divided into n blocks, each of length 128 bits. The first n/2 blocks are hashed. The generated hash is compared to the stored one. If the two hashes are the same, the first n/2 blocks are decrypted using ECC and AES. The second n/2 blocks are hashed. The generated hash is compared to the stored one. If the two hashes are the same, the second n/2 blocks are decrypted using RSA and Blowfish.

2.2. A model for secure sharing of data in MCC using Blowfish algorithm

Alam et al. [12] proposed a model for secure sharing of data in MCC using Blowfish algorithm. Data owner encrypts file using Blowfish and sends it to the cloud for storage. Data sharer downloads the file from cloud, and gets the key of it secretly from data owner to decrypt. Data owner chooses with whom to share his data.

2.3. A model for securing data in MCC using RSA, DES, and Hash Function

Garg et al. [8] proposed a mechanism to provide security for data in MCC using RSA algorithm, hash function, and DES algorithm.

2.3.1. Participants

The participants involved in this mechanism are:

- Data Owner (DO): The mobile user.
- Third Party Auditor (TPA): A trusted third party.
- Cloud Service Provider (CSP): The cloud used for storage.

2.3.2. Used Algorithms

RSA: is a widely used public-key cryptosystem (Asymmetric-key Cryptography). It is used for data transmission, digital signature, key exchange, and encryption/decryption of small data. It uses a key pair (i.e. public, and private keys) of variable size. To be secured, key has to be of length more than 1024 bits. The practical RSA implementations should use a padding scheme to be secured. The padding scheme means appending some random values to message [13]. In RSA with PKCS#1 (v1.5), the most commonly used padding

scheme, data to be encrypted should be less than key length by at least 11 bytes. For example, if a key of length 128 bytes (1024 bits) is used, message length should be of length, at most, 117 bytes.

Cryptographic Hash Function: produces a message-digest of fixed length for mapping a message of variable size. This hash function is unique for each message. The message is padded before computing the hash function. The most widely used cryptographic hash functions are:

- MD5 generates a 128-bit hash value.
- SHA-1 generates a 160-bit hash value.
- SHA-2 generates a hash value of length 224, or 256, or 384, or 512 bits [13].

DES: Data Encryption Standard is a symmetric-key cryptography algorithm. It uses a key of length 56 bits, and a data block of length 64 bits. Each block of data is processed in 16 rounds either for encryption or for decryption [13, 14].

2.3.3. Mechanism

It is divided into four phases as follows:

Key Generation Phase: Both DO and TPA use RSA for generating public and private keys.

Key Sharing Phase: TPA sends its public key to DO using a secure channel. Then, DO stores it on his/her mobile device.

Upload Phase: DO chooses a file F to upload, and encrypts it with his/her public key using RSA. Then DO generates a hash function for the encrypted file and encrypts this hash with TPA's public key using RSA. Finally, DO re-encrypts file with TPA's public key using RSA, and sends the encrypted file and hash to TPA. TPA stores the encrypted hash and decrypts the file using its private key. Then TPA generates a secret-key using DES, stores it, and encrypts the file using it. Finally, TPA sends the encrypted file to CSP for storage.

Download Phase: DO requests a file from TPA. Then TPA requests it from CSP. Finally, CSP sends it to TPA. TPA decrypts the file using the stored DES key, and generates a new hash for verifying the integrity of file. Then TPA compares the stored hash with the generated one, and sends the result which indicates file's correctness to DO with the file. DO checks the file's correctness, and then decrypts it with his/her private key using RSA.

3. Proposed Model (ARDS)

Our proposed model uses AES, and RSA Digital Signature algorithms to provide data security in MCC. The proposed architecture of ARDS model is shown in Figure 1.

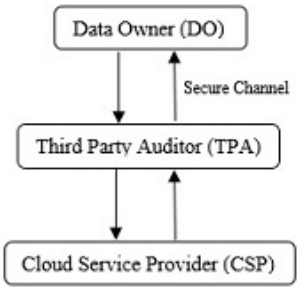

Figure 1. ARDS architecture

3.1. Participants

The participants involved in this mechanism are:

- Data Owner (DO): Mobile user who wants to store his/her data on mobile cloud.
- Third Party Auditor (TPA): A trusted third party that do some processing instead of mobile.

- Cloud Service Provider (CSP): Provides storage services to mobile users.

3.2. Used Algorithms

This model uses RSA digital signature for authentication and integrity of files. It uses AES to provide confidentiality by encrypting/decrypting files.

RSA Digital Signature: RSA algorithm is discussed in section 2.3.2. RSA Digital Signature is the most widely used digital signature technique. It is used for authentication and message integrity. The algorithm generally works as shown in Figure 2 [13]. RSA digital signature generally depends on RSA cryptography and message hashing [15]. The message has to be firstly hashed, then this hash is signed by the sender using RSA [13, 15].

1. Private and Public keys are generated using RSA.
   $K_{pr} = d, K_{pub} = (n, e)$
2. The signature s for a message x is generated by encrypting this message with the generated private key using RSA.
   $s = sig\ (x, K_{pr}) \equiv x^d \bmod n$
3. The verification is done by decrypting signature s with the generated public key using RSA.
   $x' = (s, K_{pub}) \equiv s^e \bmod n$
   If x and x' are equivalent; signature is valid.
   If x and x' are not equivalent; signature is invalid.

In practice, the message has to be first hashed producing a message-digest. Then this digest is signed.

Figure 2. RSA digital signature

AES: Advanced Encryption Standard (AES), also known as Rijndael, is a symmetric-key block cipher. AES is adopted by the U.S. government, and now used by the whole world. It uses a 128-bit block size, with key of sizes 128, 192, or 256 bits. Each block of data is processed in a number of rounds either for encryption or for decryption. All the rounds do the same operation. The number of rounds differs according to the key size as shown in Table 1. In 1997, the National Institute of Standards and Technology (NIST) announced that it needs a successor for DES and Triple-DES algorithms. This successor can provide data security for 20-30 years and to be used worldwide. AES proves itself and became the most popular symmetric-key cryptographic algorithm over DES and 3DES. AES is more secure, and faster than both DES and 3DES [13, 14].

Table 1. No. of rounds and key sizes for AES

Key Size	Number of rounds
128-bit	10
192-bit	12
256-bit	14

3.3. Mechanism

This mechanism is divided into four phases as follows:

Key Generation Phase: DO uses RSA for generating his/her public and private keys to be used for digital signature. Then he/she generates a 128-bit secret key using AES for encrypting files.

Key Sharing Phase: DO sends his/her public key to TPA using a secure channel.

Upload Phase: DO chooses a file F to upload, and encrypts it with his/her secret key using AES. Then DO generates a hash function for the encrypted file, and signs this hash with his/her private key using RSA digital signature. Finally, DO sends the encrypted file and signature to TPA. TPA checks authentication and integrity of

file by generating a new hash, and verifying the signature with DO's public key. Then TPA compares the verified signature (i.e. old hash) with the generated one, and stores the signature. Then TPA generates a new 128-bit secret-key for file F using AES, stores it, and encrypts the file using it. Finally TPA sends the encrypted file to CSP for storage.

Download Phase: DO requests a file from TPA. Then TPA requests it from CSP. Finally, CSP sends it to TPA. TPA decrypts file using the stored secret key. Then TPA checks authentication and integrity of file by generating a new hash, verifying the signature with DO's public key, and compares the verified signature (i.e. old hash) with the generated one. Finally, TPA sends the result of comparison which indicates the correctness of file with the file to DO. DO checks the correctness of file, and then decrypts it with his/her secret key using AES.

The mechanism is summarized in Figure 3.

Key Generation Phase:
DO: $\{K_{pub}, K_{pr}\}$ = GenKey ()
DO: K_{AES} = GenKey ()
Key Sharing Phase:
DO → TPA: K_{pub}
Upload Phase:
DO: F' = E (F, K_{AES})
DO: H (F') = Hash (F')
DO: Signature = Sign (H (F'), K_{pr})
DO → TPA: F', Signature
TPA: generated H (F') = Hash (F')
TPA: verified H (F') = Verify (Signature, K_{pub})
TPA: Compare (generated H (F'), verified H (F'))
TPA: Store (Signature)
TPA: K_F = GenKey ()
TPA: Store (K_F)
TPA: F'' = E (F', K_F)
TPA → CSP: F''
Download Phase:
DO → TPA: Request (F'')
TPA → CSP: Request (F'')
CSP → TPA: F''
TPA: F' = D (F'', K_F)
TPA: generated H (F') = Hash (F')
TPA: verified H (F') = Verify (Signature, K_{pub})
TPA: Compare (generated H (F'), verified H (F'))
TPA → DO: result, F'
DO: F = D (F', K_{AES})

Figure 3. ARDS mechanism

4. Evaluation and Results

In the previous section a model for securing data in MCC was introduced. The main target of it is to provide confidentiality, authentication, and integrity for data stored on the mobile cloud. The model, named ARDS, uses AES and RSA digital signature. It is compared with Garg & Sharma's model which was introduced in section 2.3. This section states the environment used for the development, security analysis, computational overhead, storage requirements, processing time, and data overhead of both models.

4.1. Environment

The model consists of data owner, third party auditor, and cloud service provider. Data owner is a mobile device with Android 5.0.2 (Lollipop) operating system, Quad-core 1.5 GHz processor, and with 3 GB RAM. Third party auditor uses Windows 7, 64-bit operating system Intel(R) Core(TM) i5, 2.4 GHz processor, and with 4 GB RAM. TPA is simulated in JAVA. For the cloud, dropbox is used. Padded RSA with PKCS#1 (v1.5) is used with 1024-bit (128-byte) keys. AES algorithm is used with 128-bit (16-byte) key. The cryptographic hashing algorithm used is SHA-2 with length 256 bits (32 bytes).

4.2. Security Analysis

4.2.1. Garg & Sharma's Model

This model provides confidentiality of data stored on the mobile cloud. But it does not provide authentication and integrity of data in a correct way.

Confidentiality: Confidentiality of data is provided during data transmission and data storage. The file is encrypted while transmission from DO to TPA, and from TPA to CSP. This avoids any intruder from knowing any information from the transferred file. The file stored on the cloud is encrypted twice; once with DO's key, and once with a key generated specifically for this file by TPA. This avoids CSP and anyone access file on the cloud from knowing the content of this file.

Authentication and Integrity: The file and the hash of file are signed by TPA's public key. Any intruder (e.g. other user) knows TPA's public key can change file, computes a hash for the new file, and then signs the new file and hash using TPA's public key. Moreover, TPA does not verify integrity of file before sending it to the cloud for storage. Integrity of file is verified only while downloading file from the cloud.

4.2.2. ARDS Model

This model provides confidentiality, authentication and integrity of data stored on the mobile cloud.

Confidentiality: Confidentiality of data is provided while transmission and storage of data. The file is encrypted when it is transmitted from DO to TPA, and from TPA to CSP. This avoids any intruder from knowing any information from the transferred file. The file stored on the cloud is encrypted with DO's key, and then re-encrypted with the key generated by TPA specifically for this file. This avoids CSP and anyone access file on the cloud from knowing the content of this file.

Authentication and Integrity: The file is hashed and this hash is signed by DO's private key. No intruder can change file while transmission because no one know the private key of DO to generate a hash and sign it. TPA verifies authentication and integrity of file before sending it to the cloud for storage, and again while downloading it from the cloud.

4.3. Computational Overhead

4.3.1. Garg & Sharma's Model

The computational overhead of Garg & Sharma's model according to phases is as follows:

Key Generation Phase: Both DO, and TPA performs public and private keys generation.

Upload Phase: DO performs two encryption for file, one hash function generation, and one encryption for the hash. While TPA performs one decryption for file, one key generation, and one encryption for file.

Download Phase: DO performs one decryption for file. While TPA performs one decryption for file, one hash function generation, and one decryption for the hash.

The computational overhead of this model is summarized in Table 2.

Table 2. Computational overhead of Garg & Sharma's model

Key Generation Phase			
	Cryptography	**Hashing**	**Key Generation**
DO	0	0	1
TPA	0	0	1
Upload Phase			
	Cryptography	**Hashing**	**Key Generation**
DO	3	1	0
TPA	2	0	1
Download Phase			
	Cryptography	**Hashing**	**Key Generation**

DO	1	0	0
TPA	2	1	0

4.3.2. ARDS Model

The computational overhead of ARDS model according to phases is as follows:

Key Generation Phase: DO performs public and private keys generation. Moreover, he/she generates a secret key for files.

Upload Phase: DO performs one encryption for file, one hash function generation, and one encryption for the hash. While TPA performs one hash function generation, one decryption for the hash, one key generation, and one encryption for file.

Download Phase: DO performs one decryption for file. While TPA performs one decryption for file, one hash function generation, and one decryption for the hash.

The computational overhead of this model is summarized in Table 3.

Table 3. Computational overhead of ARDS model

Key Generation Phase			
	Cryptography	**Hashing**	**Key Generation**
DO	0	0	2
TPA	0	0	0
Upload Phase			
	Cryptography	**Hashing**	**Key Generation**
DO	2	1	0
TPA	2	1	1
Download Phase			
	Cryptography	**Hashing**	**Key Generation**
DO	1	0	0
TPA	2	1	0

4.4. Storage Requirements

4.4.1. Garg & Sharma's Model

The storage required by Garg & Sharma's model according to participants is as follows:

- DO stores his/her public and private keys, and the public key of the TPA.
- TPA stores its public and private keys, an encrypted hash for each file, and a secret key for each file.
- CSP stores encrypted files uploaded by mobile user.

The storage requirements of this model is summarized in Table 4.

Table 4. Storage requirements of Garg & Sharma's model

Participants	**Storage Requirements**
DO	1. His/her public and private keys. 2. TPA's public key.
TPA	1. Its public and private keys. 2. **For each file:** encrypted hash of file. 3. **For each file:** secret key.
CSP	Encrypted files uploaded by DO.

4.4.2. ARDS Model

The storage required by ARDS model according to participants is as follows:

- DO stores his/her public key, private key, and secret key for files.
- TPA stores public key of DO, a signature (encrypted hash) for each file, and a secret key for each file.
- CSP stores encrypted files uploaded by mobile user.

The storage requirements of this model is summarized in Table 5.

Table 5. Storage requirements of ARDS model

Participants	Storage Requirements
DO	1. His/her public and private keys. 2. His/her secret key for files.
TPA	1. DO's public key. 2. **For each file:** signature of file. 3. **For each file:** secret key.
CSP	Encrypted files uploaded by DO.

4.5. Processing Time

Both Garg & Sharma's model, and ARDS model are implemented in the same environment stated above. The processing time is measured in seconds. Each time value is the average of 5 runs done in different times. The average processing time is calculated for each phase as follows:

Key Generation Phase: The average processing time needed by each model in this phase is shown in Table 6 and Figure 4. It is clear that Garg & Sharma's model is slower than ARDS model. DO's processing in Garg & Sharma's model is slightly faster than that of ARDS model because DO generates an asymmetric-key pair only, but in ARDS model he/she generates an extra symmetric-key. In ARDS model there is no TPA's processing.

Table 6. Average processing time in key generation phase

Garg & Sharma's Model			ARDS Model		
DO	**TPA**	**Total**	**DO**	**TPA**	**Total**
0.118	0.05	0.168	0.125	0	0.125

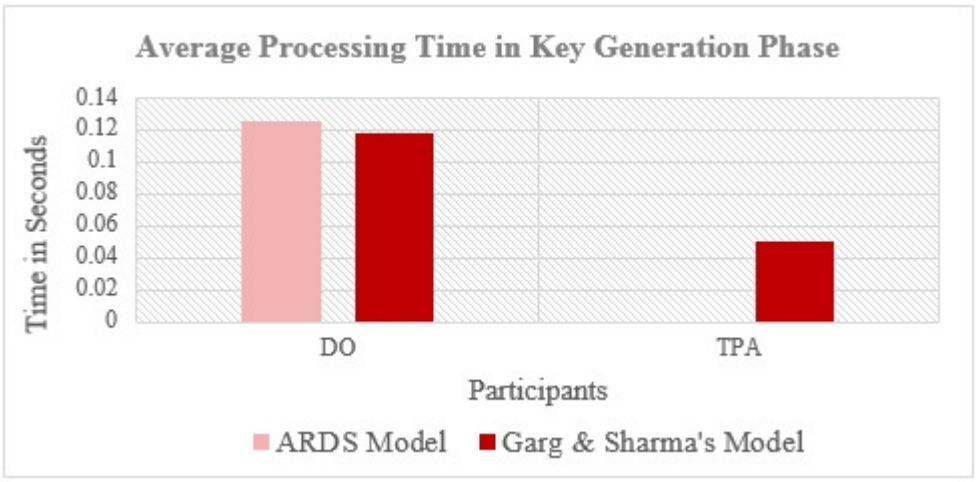

Figure 4. Average processing time in key generation phase

Upload Phase: DO's average processing time needed by each model in upload phase is shown in Table 7 and Figure 5. It is clear from them that Garg & Sharma's model is the slower. The main reason is that encrypting large data with asymmetric algorithm like RSA is not preferred yet as it takes too long time. Moreover, DO encrypts file using RSA twice. That's why there is a great difference between the processing of the two models especially when file is large (i.e. files of sizes 1 MB, 2 MB, and 5 MB). TPA's average processing time needed by each model in upload phase is shown in Table 8 and Figure 6. It is clear that Garg & Sharma's model is the slower. The main reason is that TPA decrypts file using RSA before encrypting it with DES and this takes too long time.

It is clear from all of the above that ARDS model is faster than Garg & Sharma's model in uploading files. The whole average processing time needed by both models to upload files is summed up and summarized in Table 9.

Table 7. DO's average processing time in upload phase

Model / File Size	Garg & Sharma's Model	ARDS Model
117 bytes	0.051	0.024
128 bytes	0.053	0.025
512 bytes	0.057	0.026
1 KB	0.059	0.027
5 KB	0.155	0.029
10 KB	0.296	0.032
50 KB	0.759	0.042
100 KB	2.666	0.059
1 MB	26.251	0.357
2 MB	52.588	0.626
5 MB	132.18	1.496

Table 8. TPA's average processing time in upload phase

Model / File Size	Garg & Sharma's Model	ARDS Model
117 bytes	0.071	0.036
128 bytes	0.074	0.039
512 bytes	0.074	0.046
1 KB	0.087	0.048
5 KB	0.185	0.052
10 KB	0.209	0.055
50 KB	0.733	0.059
100 KB	1.372	0.075
1 MB	13.159	0.107
2 MB	26.719	0.163
5 MB	64.594	0.279

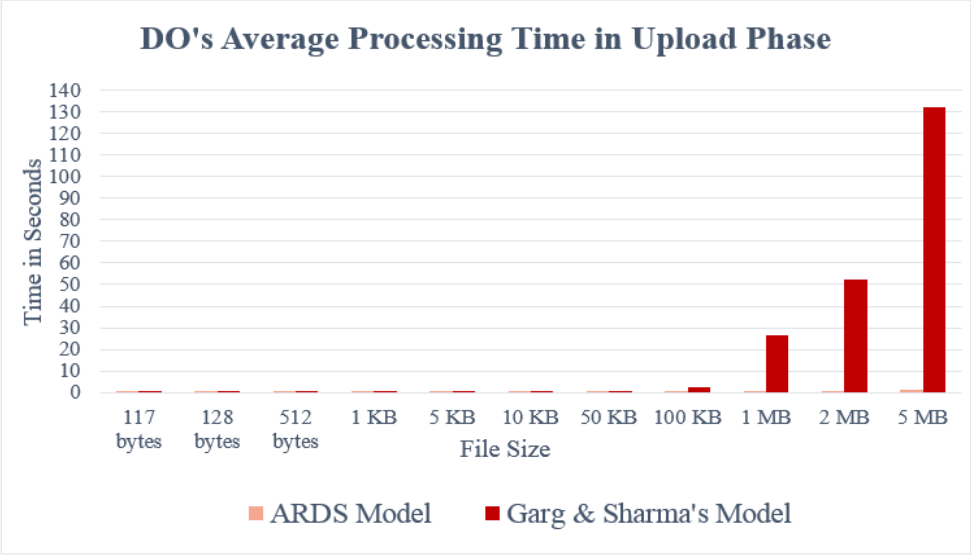

Figure 5. DO's average processing time in upload phase

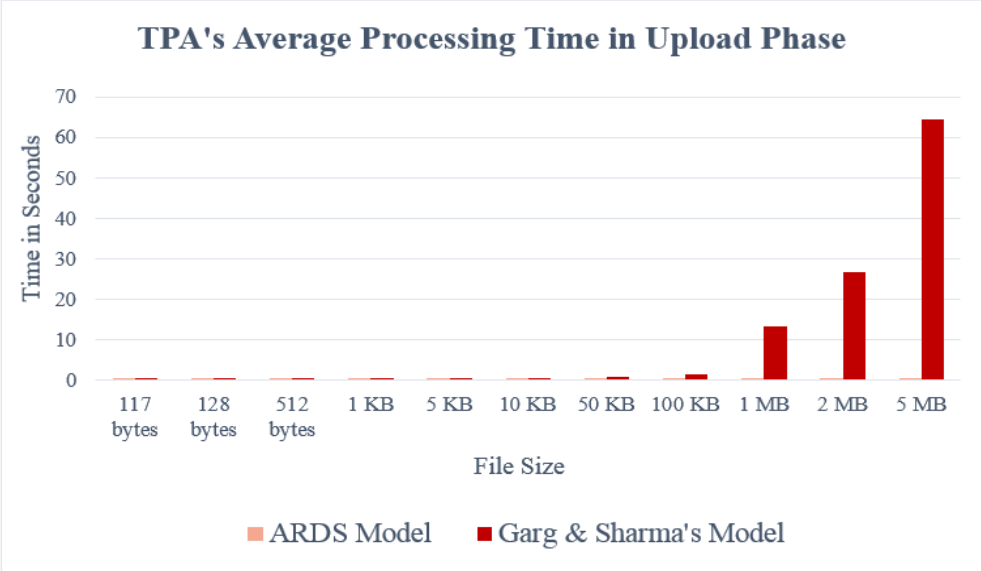

Figure 6. TPA's average processing time in upload phase

Table 9. Average processing time in upload phase

Model / File Size	Garg & Sharma's Model			ARDS Model		
	DO	**TPA**	**Total**	**DO**	**TPA**	**Total**
117 bytes	0.051	0.071	0.122	0.024	0.036	0.06
128 bytes	0.053	0.074	0.127	0.025	0.039	0.064
512 bytes	0.057	0.074	0.131	0.026	0.046	0.072
1 KB	0.059	0.087	0.146	0.027	0.048	0.075
5 KB	0.155	0.185	0.34	0.029	0.052	0.081
10 KB	0.296	0.209	0.505	0.032	0.055	0.087
50 KB	0.759	0.733	1.492	0.042	0.059	0.101
100 KB	2.666	1.372	4.038	0.059	0.075	0.134
1 MB	26.251	13.159	39.41	0.357	0.107	0.464
2 MB	52.588	26.719	79.307	0.626	0.163	0.789
5 MB	132.18	64.594	196.774	1.496	0.279	1.775

Download Phase: DO's average processing time needed by each model in download phase is shown in Table 10 and Figure 7. It is clear from them that Garg & Sharma's model is the slower because of decrypting large data with RSA. TPA's average processing time needed by each model in download phase is shown in Table 11 and Figure 8. It is clear from them that Garg & Sharma's model is slightly slower because of decrypting the stored hash using RSA.

It is clear from all of the above that ARDS model is the faster in downloading files. The whole average processing time needed by both models to download files is summed up and summarized in Table 12.

Table 10. DO's average processing time in download phase

Model / File Size	Garg & Sharma's Model	ARDS Model
117 bytes	0.026	0.014
128 bytes	0.03	0.014
512 bytes	0.036	0.017
1 KB	0.049	0.018
5 KB	0.098	0.022
10 KB	0.115	0.026
50 KB	0.343	0.032
100 KB	0.63	0.044
1 MB	5.874	0.301
2 MB	11.883	0.487
5 MB	29.55	1.289

Table 11. TPA's average processing time in download phase

Model / File Size	Garg & Sharma's Model	ARDS Model
117 bytes	0.072	0.029
128 bytes	0.09	0.031
512 bytes	0.111	0.037
1 KB	0.125	0.041
5 KB	0.14	0.05
10 KB	0.19	0.055
50 KB	0.23	0.059
100 KB	0.257	0.064
1 MB	0.295	0.118
2 MB	0.329	0.135
5 MB	0.397	0.212

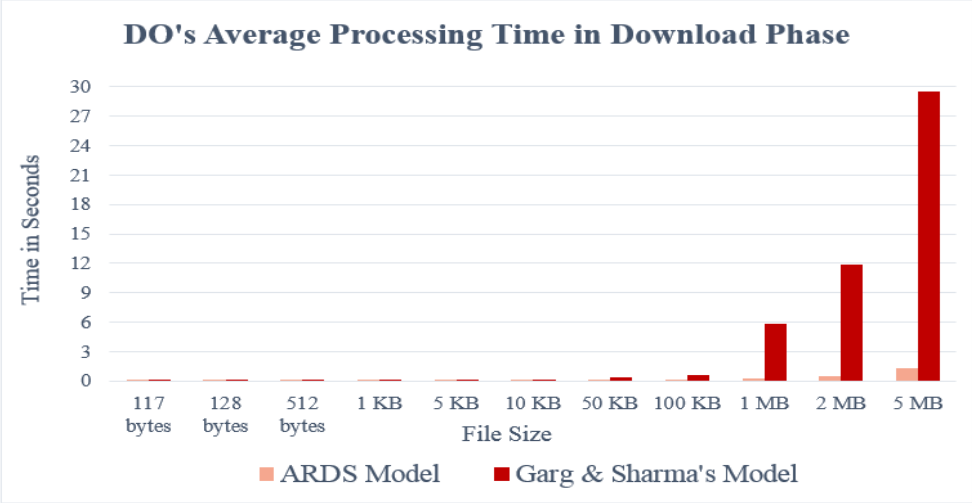

Figure 7. DO's average processing time in download phase

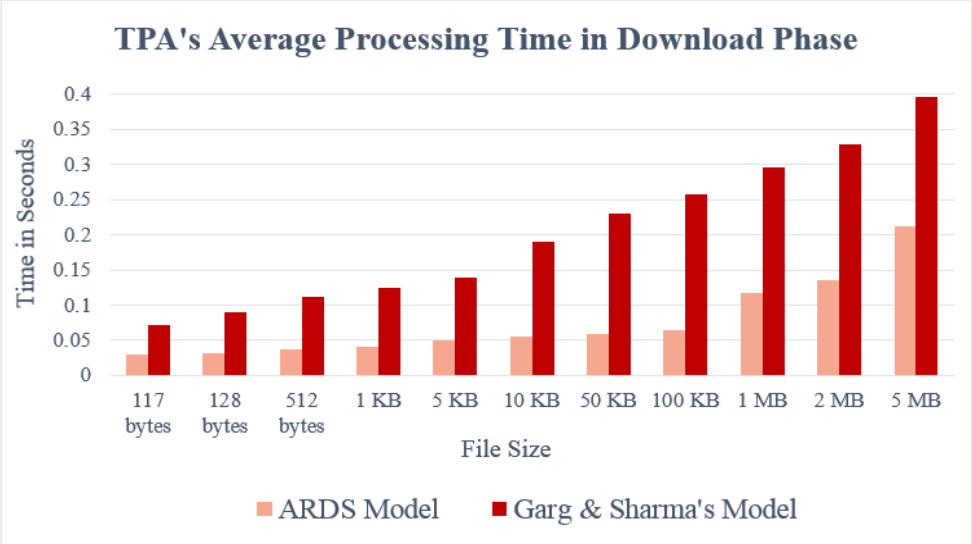

Figure 8. TPA's average processing time in download phase

Table 12. Average processing time in download phase

Model / File Size	Garg & Sharma's Model			ARDS Model		
	DO	**TPA**	**Total**	**DO**	**TPA**	**Total**
117 bytes	0.026	0.072	0.098	0.014	0.029	0.043
128 bytes	0.03	0.09	0.12	0.014	0.031	0.045
512 bytes	0.036	0.111	0.147	0.017	0.037	0.054
1 KB	0.049	0.125	0.174	0.018	0.041	0.059
5 KB	0.098	0.14	0.238	0.022	0.05	0.072
10 KB	0.115	0.19	0.305	0.026	0.055	0.081
50 KB	0.343	0.23	0.573	0.032	0.059	0.091
100 KB	0.63	0.257	0.887	0.044	0.064	0.108
1 MB	5.874	0.295	6.169	0.301	0.118	0.419
2 MB	11.883	0.329	12.212	0.487	0.135	0.622
5 MB	29.55	0.397	29.947	1.289	0.212	1.501

4.6. Data Overhead

The size of encrypted files stored on the cloud is compared in both models. This comparison is shown in Table 13, and the data overhead (i.e. added bytes) is shown in Figure 9. It is clear from them that file size is enlarged in Garg & Sharma's model. The main reason is that this model uses padded RSA to be secured. Padded RSA with 128-byte key length chunks data into 117-byte block length. Each block is padded to be of length 128 bytes. While in ARDS model; only the last block of data will be padded when file is not dividable by block size.

Table 13. Size of encrypted files in both models

Model / File Size	Garg & Sharma's Model	ARDS Model
117 bytes	128 bytes	128 bytes
128 bytes	256 bytes	128 bytes
512 bytes	640 bytes	512 bytes
1 KB	1.125 KB	1 KB
5 KB	5.5 KB	5 KB
10 KB	11 KB	10 KB
50 KB	54.75 KB	50 KB
100 KB	109.5 KB	100 KB
1 MB	1.094 MB	1 MB
2 MB	2.188 MB	2 MB
5 MB	5.47 MB	5 MB

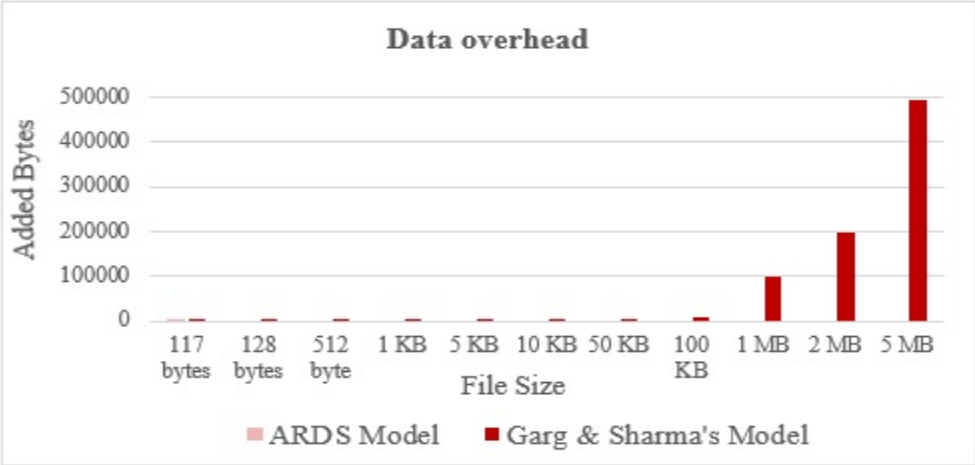

Figure 9. Data overhead in both models

5. Conclusion and Future Works

The main objective of this paper is to propose a model for securing data in mobile cloud computing. This model should take into account the mobile device's limitations such as, battery life, processor's power, and memory size. A number of frameworks were implemented to secure data in mobile cloud computing. Some of these frameworks were presented here, and one of them was chosen to evaluate our model according to it. The chosen model was introduced by Garg and Sharma in [8]. Garg & Sharma's model uses Hashing, RSA, and DES algorithms to secure data in MCC. Our proposed model, named ARDS, uses AES and RSA digital signature to provide authentication, confidentiality, and integrity of data stored on the mobile cloud. By using this model, users can save their data regardless its size, and without caring about privacy and security.

A simulation program was developed for both models (Garg & Sharma's model and the proposed ARDS model). The two models were compared, and the result was that ARDS model provides better performance, storage, and security than Garg & Sharma's model because of the following:

1. Garg & Sharma's model signs the file and hash of file using TPA's public key which can be known easily. It is known to other users. While ARDS model signs the hash of file using DO's private key which is known only to that DO.

2. Garg & Sharma's model doesn't verify integrity of file before sending it to the cloud for storage. It is verified only after downloading file. While ARDS model verifies integrity of file before sending it to the cloud and after downloading it from the cloud.

3. Garg & Sharma's model uses RSA algorithm for the encryption and decryption of files. Encrypting or decrypting large data with asymmetric algorithm like RSA is not preferred yet as it takes too long time. The file is encrypted twice on behalf of the mobile device using RSA. While ARDS model encrypts and decrypts files using AES algorithm which is fast and much secured symmetric algorithm. The file is encrypted only one time on behalf of the mobile device.

4. Garg & Sharma's model uses RSA for encrypting and decryption of files and using such algorithm should be accompanied with a padding scheme which means enlargement of ciphertext. For example, in RSA with PKCS#1 (v1.5), the most commonly used padding scheme, data to be encrypted should be less than key length by at least 11 bytes (i.e. If we use a key of size 128 bytes, data is chunked into 117-byte block length. Each block is padded to be of length 128 bytes). While ARDS model uses AES which results in a ciphertext nearly equal to the plaintext except if plaintext is not dividable by block size, the last block will be padded.

Data security in mobile cloud computing is a huge research area. There are several challenges and issues should be addressed to help the deployment of mobile cloud computing. Some of the future works are proposed as follows:

1. Reducing the processing done on behalf of the mobile device more by trying well-secured algorithms other than AES, and RSA digital signature.

2. The mobile device can be lost easily, so we need to find a way to secure storage of keys (i.e. Recovery and backups of keys).

3. The mobile can be stolen easily at any time so, we need to add some additional authentication for the data owner. We can use passwords, one time passwords (OTP), biometrics, and so on.

References

1. S. Zhang, X. Chen, S. Zhang, and X. Huo, "The comparison between cloud computing and grid computing," In Proc. Int. Conf. on Computer Application and System Modeling (ICCASM), IEEE, vol.11, 2010, pp.72-75.
2. P. Mell and T. Grance, "The nist definition of cloud computing," 2011. [Online]. Available: http://csrc.nist.gov/publications/nistpubs/ 800-145/SP800-145.pdf
3. N. A. Abd-El Azim, and A. H. El Bastawissy, "Transactions management in cloud computing," Egyptian Computer Science Journal, vol.38, no.1, 2014.
4. A. Ahmad, M. M. Hassan, and A. Aziz, "A multi-token authorization strategy for secure mobile cloud computing," In Proc. Int. Conf. on Mobile Cloud Computing, Services, and Engineering (MobileCloud), IEEE, 2014, pp.136-141.
5. D. Jana, and D. Bandyopadhyay, "Management of security and privacy issues of application development in mobile cloud environment: A survey," In Proc. Int. Conf. on Recent Advances and Innovations in Engineering (ICRAIE), IEEE, 2014, pp.1-6.
6. S. Abolfazli, Z. Sanaei, M. H. Sanaei, M. Shojafar, and A. Gani, "Mobile cloud computing: The-state-of-the-art, challenges, and future research," 2015.

7. N. Aminzadeh, Z. Sanaei, and S. H. A. Hamid, “Mobile storage augmentation in mobile cloud computing: Taxonomy, approaches, and open issues,” Simulation Modelling Practice and Theory, vol.50, 2015, pp. 96-108.

8. P. Garg, and V. Sharma, “An efficient and secure data storage in mobile cloud computing through RSA and Hash function,” In Proc. Int. Conf. on Issues and Challenges in Intelligent Computing Techniques (ICICT), IEEE, 2014, pp.334-339.

9. S. Subashini, and V. Kavitha. “A survey on security issues in service delivery models of cloud computing,” Journal of Network and Computer Applications, vol.34, no.1, 2011, pp.1-11.

10. A. N. Khan, M. L. M. Kiah, S. U. Khan, and S. A. Madani, “Towards secure mobile cloud computing: A survey.” Future Generation Computer Systems, vol.29, no.5, 2013, pp. 1278-1299.

11. H. M. A. Kader, M. M. Hadhoud, S. M. El-Sayed, and D. S. AbdElminaam, “Performance evaluation of new hybrid encryption algorithms to be used for mobile cloud computing,” International Journal of Technology Enhancements and Emerging Engineering Research, vol.2, no.4, 2014.

12. M. M. Alam, S. Hati, D. De, and S. Chattopadhyay, “Secure sharing of mobile device data using public cloud,” In Proc. of CONFLUENCE, Fifth International Conference on Confluence The Next Generation Information Technology Summit, IEEE, 2014, pp.149-154.

13. C. Paar, and J. Pelzl, “Understanding cryptography: a textbook for students and practitioners,” Berlin, Germany, Springer verlage, 2010.

14. W. Stallings, “Cryptography and network security: principles and practices,” Edition 4, Prentice Hall, 2006.

15. K. H. Rahouma, “Design of a multi-digital signature protocol,” Egyptian Computer Science Journal, vol.28, no.1, 2006.