Received July 26, 2017, accepted August 4, 2017, date of publication August 25, 2017, date of current version September 27, 2017.

Digital Object Identifier 10.1109/ACCESS.2017.2744670

CLOAK: A Stream Cipher Based Encryption Protocol for Mobile Cloud Computing

AMIT BANERJEE, MAHAMUDUL HASAN, MD. AUHIDUR RAHMAN, AND RAJESH CHAPAGAIN
Department of Computer Science, South Asian University, New Delhi 110021, India

Corresponding author: Mahamudul Hasan (m.hasan@students.sau.ac.in)

ABSTRACT Mobile device and its applications have revolutionized the way we store and share data. It is becoming a warehouse of users personal information. Unluckily, most of these data are stored in an unencrypted format, prone to security threats. In this paper, we propose a lightweight, computationally efficient protocol, called *CLOAK*, for the mobile device. CLOAK is based on stream cipher and takes the help of an external server for the generation and distribution of cryptographically secure pseudo-random number (CSPRN). In order to enhance the security of our protocol, we use the concept of symmetric key cryptography. We present three versions of the protocol referred as *s-CLOAK*, *r-CLOAK* and *d-CLOAK*, varying on the basis of the key selection procedure. In CLOAK, the core encryption/decryption operation is performed within the mobile devices to secure data at its origin. The security of CSPRN is ensured using deception method. In CLOAK, all messages are exchanged securely between mobile and the server with mutual identity verification. We evaluate CLOAK on Android smart phones and use Amazon Web services for generating CSPRN. Additionally, we present attack analysis and show that the brute force attack is computationally infeasible for the proposed protocol.

INDEX TERMS Cloud computing, mobile cloud computing, mobile device, security, stream cipher, encryption, decryption.

I. INTRODUCTION

Advancements in mobile technology, innovative applications and decreasing prices of smartphones, wearable computers and other Mobile devices (MD) have contributed significantly in increasing popularity of mobile devices in our modern lifestyle. Since, MDs are meant for personal usage, it is often used as a repository for storing users personal information, such as user profile, passwords, bank account information and medical records. More significantly, the data is stored in a clear text format in a MD, which can be easily retrieved and lead to serious security complications. Security threats on MD can be from various sources including malwares, third party applications, eavesdropping over wireless network, theft and lost devices. As a result, many companies do not allow employees to store corporate data in smartphones or use the corporate network through personal devices [1].

Mobile cloud computing (MCC) is an emerging research area focusing on supplementing the storage and computational requirement of MD by utilizing the cloud infrastructure [2]–[4]. By interacting with cloud, MD can deliver various services to the user, such as healthcare [5], mobile commerce [6] and online education [7]. Users can upload and store data (photos, medical records) from their MD to the cloud and can share them with others. In addition, MD can offload computation intensive tasks to the cloud to overcome its resources limitation and for saving battery [4]. However, security is a major concern in MCC [8], particularly for mobile applications sending unencrypted personal information over insecure wireless medium to the cloud. Data encryption is also required for protecting users data against external and internal attacks within the cloud environment [9].

Encryption/decryption algorithms are commonly used for providing security to user's personal information [10], [11]. Encryption is a process of converting plaintext (PT) data into an incomprehensible code called ciphertext (CT) and a decryption algorithm is used for inverting the CT to original PT. In this paper, we focus our discussion on the encryption and decryption of files for the MD. There are three basic approaches for the same.

- The encryption/decryption operations can be performed within the MD, which we refer as a mobile centric approach. Authors in [12] and [13], have studied the feasibility of implementing the standard symmetric and asymmetric cryptographic algorithms (DES, AES, RSA) in the MD. However, due to high computational complexity, the standard encryption algorithms are not efficient for the resource constrained MDs [12]. The performance can be improved by S-Box optimization and reducing the number of rounds [14], [15] but more lightweight encryption/decryption algorithm is required of the MDs.

- Secondly, the MD can offload files and perform the computation intensive encryption/decryption tasks to the cloud or an external server (ES). By offloading the task, MD can overcome its resource limitations and can efficiently handle large files in a relatively short time frame [16]. Researchers have proposed solutions to address the security concerns associated with offloading files, such as using a trusted third party (TTP) [17], secure channel [18], mobile VPN [19], file splitting [20], [21] and multipath TCP [22]. Most of these techniques depend upon intermediate server or infrastructure, which may not be feasible for many MCC applications, like instant photo uploading.
- An intermediate approach is to share the computation by encrypting the important parts of a file in the mobile device and offloading the remaining tasks to the cloud [23], [24].

In this paper, we propose a protocol for encrypting and decrypting files within the MDs in a mobile cloud environment, referred as *CLOAK*. Our goal is to secure personal information stored in MD (images, pdf, doc), of size in the range of 5-10 MBs. The CLOAK protocol is based on stream cipher and takes the help of a cloud or an external server (ES) for generating the key-stream or a cryptographically secure pseudo-random number (CSPRN). The advantage of using stream cipher as the basis of our protocol, is that it is less computation intensive compared to block cipher [25] and can easily be handled by existing MDs. Stream cipher is a cryptographically secure encryption algorithm, used in various protocols (WPA, TLS), applications (Internet Explorer, Google Chrome, Firefox) and in communication standards (GSM, 3GPP, LTE) [26]–[28]. The design considerations of our protocol are as follows:

- To design a lightweight encryption protocol for MD. We assume that a single file size of 5 to 10 MB is adequate for images and documents in txt, pdf, doc formats. The protocol must be able to handle such files on most MDs that are currently available in the market.
- The encryption/decryption operation must be performed in an acceptable time frame.
- The users must be able to control the encryption/decryption operation. This is important for establishing user's confidence on the system.
- All operations on the PT must be performed locally on the MD and the computations on ES should not affect the security of the protocol.
- Finally, and most importantly, the protocol must be cryptographically secure.

One of the major challenges of a stream cipher is the generation and distribution of the key-stream or CSPRN (C). In CLOAK, we offload this task to an external server (ES) in the cloud to save resources of the MDs. In addition, the cloud can be used for sharing the encrypted files with multiple recipients. To address the security of the CSPRN (C), we propose two level CSPRN modification. Firstly, the C is modified to C'' by the ES (i.e., $C \xrightarrow{S_k} C''$), before transmitting it to the MD. This ensures the security of C againsts the vulnerabilities of unreliable wireless media. Furthermore, we need to ensure that only the intended recipients should be able to decrypt the file. For this, we perform another modification on C in the MD to generate C'. The C' is used for the encrypting a file (i.e. $C \xrightarrow{k} C'$ and $CT = PT \oplus C'$) and can only be decrypted by the receipients having the key k. We investigate two randomized (*s-CLOAK* and *r-CLOAK*) and a deterministic approach (*d-CLOAK*) for generating C'.

Secrecy of the key is the basic requirement of all cryptographic algorithms and adversary may impose various attacks to retrieve the same. Since we are using C' for the encryption process, the generation procedure of $C \xrightarrow{k} C'$ plays a crucial role in the security of the protocol. In the proposed algorithms, we use the key (k) for generating C'. We show that, for an unknown k, it is computationally infeasible for an adversary to generate C' through a brute force attack. In addition, we also show that the protocol can resist attacks like two time pad, known plaintext, algebaric, Man-in-the-middle, insider, impersonation and DoS. We evaluate the performance of CLOAK on five different Android-based smartphones and use Amazon Web services (AWS) for CSPRN and study the complexity of the algorithm (i.e., time, space, processing power) by varying the file size.

Rest of the paper is organized as follows. In section-II, we present the related works. In section-III, we discuss the basics of the CLOAK protocol, generation of CSPRN and the security challenges. In section-IV, we discuss the modification of CSPRN and secure the message flow. Next, section-V, analyses the security threats of the CLOAK protocol. Section-VI, illustrates the important properties of the protocol and presents the experimental result on various Android based MDs. Finally, we conclude our discussion in section-VII.

II. RELATED WORKS

Encryption is a fundamental operation for securing users digital information against unauthorized access. A secure encryption algorithm is needed for the MD, as it is commonly used for storing and sharing users private and sensitive data, e.g., photos, videos, medical records. Moreover, encryption is also essential before offloading data from mobiles to remote cloud servers. Researchers have proposed various encryption/decryption approaches for the MD [12], [13], [23], [24]. In this section, we discuss the previous works related to our proposed protocol.

Stream cipher is a popular cryptographically secure encryption technique. Communication standards, like GSM, 3GPP, LTE [26] are using stream cipher for encrypting voice communication. Stream cipher is less computation intensive and it can easily be handled by existing MD. In [29] and [30], authors suggested that stream cipher is best suited for handling large data streams in resource constrained MD. Researchers are investigating new stream ciphers to render its advantages. It can be implemented on both hardware [31] and software [32], as listed in table-1. Some of the popular stream cipher includes A5/1, A5/2 used for voice encryption in GSM networks and RC4 used in SSL/TLS protocol (until 2015). These implementation of a stream cipher mainly differ in their key generation phase. Researchers have studied various key generation strategies, such as linear feedback shift registers (LFSR) [33], nonlinear feedback shift registers (NLFSR) [34], AES-CTR [35] for improving the efficiency of the stream ciphers and to counter attacks. However, the CLOAK protocol, proposed in this paper, is different from the above. The goal of our work is to implement a lightweight encryption protocol for resource constrained devices. Although, such devices can meet the computational requirement of a stream cipher, sharing the keystream can be a challenging task. To address this issue, we investigate the possibility of offloading the keystream generation and distribution task to an ES, such that it can be shared securely with multiple recipients. This is the main difference between CLOAK and other stream ciphers.

TABLE 1. Various stream ciphers.

Software Implementation	Hardware Inplementation
HC-128 [36]	
Rabbit [37]	GrainV1 [40]
Salsa20 [38]	MICKEY2.0 [41]
SOSEMANUK [35]	Trivium [42]
RC4 [39]	

In [12], authors explore the feasibility of implementing Advanced Encryption Standard (AES), SERPENT and TWOFISH algorithms in MD and studied the performance and computational cost of each algorithm. The analysis shows that the performance of TWOFISH is better compared to others in terms of CPU and memory utilization. However, due to high computational complexity, these algorithms are not efficient for the MD [12]. Authors in [14] and [15] have optimized the AES/DES encryption algorithm. In AES, the main functionalists are performed in the S-BOX, which comprises of two transformations, multiplicative inverse and followed by an affine transformation. In [15], authors focus on rectifying the S-BOX operations to make the algorithm lightweight and proposed a modified affine transformation with lesser time complexity than the original AES. The authors prove that the proposed optimized algorithm provides similar security as compare to the original version.

Researchers have proposed selective or partial encryption schemes to overcome the resource limitations of MD [23], [24]. In this, the important parts of a file are parsed for encryption within the MD and the remaining parts can be transferred to an external server or stored in the mobile in a plaintext format. Selective encryption can save resources in MD. In [23] authors eliminate the duplicate data in structured files (e.g. pdfs, ppt, docx) before encryption to reduce the amount of data at its source to save the storage space and network bandwidth. Similarly, in [24], data is fragmented both vertically and horizontally to determine the parts of the data for encryption.

Homomorphic encryption technique allows computational operations on the ciphertext to produce encrypted output and is more applicable for mobile cloud computing (MCC). MD can use homomorphic encryption to offload the encrypted data to the cloud, perform computational operations on the same and return the encrypted result back to the mobile. On decryption, it produces the same output as can be obtained by performing the operation on the plaintext. In [13], authors propose a light-weight homomorphic encryption (LHE) algorithm for the MD. The LHE algorithm has four essential services, i.e., key generation, data encryption, data recovery and data evaluation. The data evaluation algorithm allows both the addition and multiplication operations on the ciphertext.

Hardware based encryption is another approach for securing data in MD. In [43], authors use the Graphical Processing Unit (GPU) for handling computation intensive cryptographic operations. It utilizes the performance of the GPUs for non-graphical computations. Authors consider the XTS-AES based encryption algorithm and used the OpenCL APIs for GPU programming. The C library *libc* is used for interacting with the Andriod platform. The user's data is encrypted transparently during the read/write operations in the MD. Other research works related to hardware based encryption are [44], [45].

Attribute-Based Encryption (ABE) is based on public key encryption paradigm. In ABE, the secret key is defined by a set of attributes or policy on the attributes (e.g KP-ABE [46], CP-ABE [47]). The data can be encrypted with respect to subsets of attributes. For decryption, the set of user attributes must match with the attributes of the cipher text. In [48], authors propose an ABE scheme for the mobile cloud environment. The computational overhead of the mobile is reduced by delegating the part of encryption tasks to the cloud. There are four entities in [48]: user, Trusted Key Generator (TKG), Cloud Service Provider (CSP), and Token Service Provider (TSP). The TKG is responsible for generation and distributions of keys and the TSP uses token-controlled public key encryption for efficient key revocation.

Chen *et al.* [32] proposed an encryption algorithm called self-encryption (SE) based on stream cipher. In SE, the key stream is generated randomly within the MD by using user's PIN and a nonce. Then, the key is XORed with the plaintext for generating ciphertext. The ciphertext is stored in the user's mobile phone. However, the key stream and other parameters are stored in a secure external server. For decryption, the user needs to provide the PIN to the server for authentication. The server validates the user and forwards the key stream to the user. On receiving the keystream, the mobile can decrypt the stored data. In SE, the length of the keystream depends on the user's security demand. The SE algorithm is similar to our proposed algorithm. However, the difference is that in SE the keystream is generated inside the MD, whereas in CLOAK, the keystream is obtained from an external server/cloud. Moreover, SE uses a trusted third party server for storing the keystream and in CLOAK we study the feasibility of utilizing both trusted and untrusted external servers for the generation and distribution of the keystream.

III. IMPLEMENTATION

In this section, we discuss the implementational details of our proposed protocol. We begin by introducing a general overview for the generation of CSPRN and the basic overview of the proposed CLOAK protocol. Then introduce the security issues of CLOAK.

A. BASIC CLOAK ARCHITECTURE

CLOAK is a light-weight, stream cipher based encryption protocol for secure data communication between two MDs. The two fundamental operations of a stream cipher are key generation and XORing. These are independent operations and can be performed separately. This is the fundamental idea of CLOAK. In CLOAK, the key generation operation can be performed in an ES/cloud and the XORing operation is performed in the MD to generate the CT.

There are three main components of our protocol are clients, the external server (ES) and the communication media (CM), figure-1. A client can be a smartphone, tablet or a PC interested in performing the encryption/decryption operation. In MCC, an ES is often used for offloading the computationally intensive tasks from resource constrained MD [12]. In CLOAK, we use an ES for generating the CSPRN. The ES can be specifically configured according to the requirement of an application and the workload. The communications between MD and cloud ES can take place via any wireless communication media such as Wi-fi, 3G, 4G, UMTS, LTE. The commonly used notations in CLOAK protocol is shown in table-2.

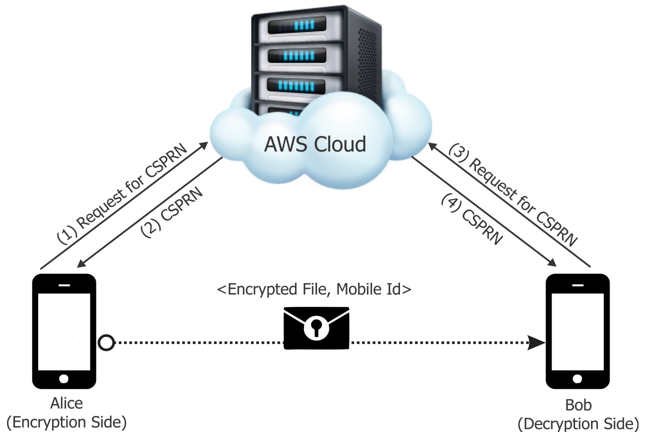


FIGURE 1. Basic architecture of proposed protocol.

Figure-1 shows the basic architecture of the CLOAK protocol. In this, we consider a scenario where a user say, Alice, wants to share a file present in her mobile device to another user, say Bob. We assume that the MDs of both Alice and Bob are registered with the ES. For secure communication, CLOAK uses a pre-shared key (S_k) between ES and MD, defined during the registration of the MD. For encryption, Alice first selects the intended file and sends a request to the ES specifying unique user-id (*uid*), file-name (*fn*) and CSPRN-size (*cs*). The *uid* can be a user-defined password, device-id or IMEI number of the MD. The $< uid, fn >$ is meant for unique identification of a file originating from an MD. For the security purposes, the size of the CSPRN (*cs*) should be greater then 128 bits [49].

On receiving this request, the ES produces CSPRN using algorithm-1 and returns the CSPRN back to Alice. To make the size of CSPRN (*cs*) equal to PT (say *n*), a pre-processing operation, i.e. replication or truncation on CSPRN may be required. In case, $cs < n$, the CSPRN can be replicated $(|n|/|cs|)$ times. Similarly, if the $cs > n$, truncation can be performed to discard the extra bits. Then Alice can encrypt

TABLE 2. Notations used in CLOAK.

Notation	Description
MD	Mobile Devices
ES	External Server
PT	Plaintext
CT	Ciphertext
CM	Comuunication Media
C,C',C''	Cryptographically Secure Pseudo-random number (CSPRN), Key-stream
PRN	Pseudo-random number
OTP	One Time Password
OOB	Out-of-band Channel
fn	File Name
cs	CSPRN Size
un	User Name
uid	Unique User ID
s	Seed
k	Key
S_k	Pre-shared key between MD and ES
T_k	Token
T_s	Time Stamp

Algorithm 1 CSPRN Generation

```
Function CSPRN_Gen (CSPRN size: cs)
    $s \leftarrow$ random_num() ;          /* Key or Seed */
    $sn \leftarrow$ random_num() ;            /* Seq Num */
    $CSPRN \leftarrow$ NULL ;          /* Init. CSPRN */
    $n \leftarrow \lceil cs/128 \rceil$;
    while $n > 0$ do
        $CSPRN \leftarrow CSPRN + AES(s, sn)$;
        $sn \leftarrow sn + 1$;
        $n \leftarrow n - 1$;
    return CSPRN;
```

the file using this CSPRN and can either save the CT in SD card or store it in the ES, which completes the encryption process. For sharing the encrypted file, Alice sends the $< uid, fn >$ to Bob. In the decryption phase, Bob sends a request to the ES specifying $< uid, fn >$, for the CSPRN. The ES consults its database with these parameters, generates the CSPRN of size cs and forwards it to Bob. Finally, Bob can decrypt the file using the CSPRN.

In CLOAK, XORing is the only operation performed in the MD. For encryption, the PT is XORed with the CSPRN to generate the CT and in decryption, the CT is again XORed with the same CSPRN to retrieve the original PT. In our protocol, to handle the memory limitations of the MD, we perform chunk-wise XORing operation by incrementally reading the file and CSPRN in chunks of equal sizes. In general, XORing is a simple operation with less computation and memory requirement, which can be easily implemented in MD. In our experiment, we find that the read/write operation for a chunk size of 512 bytes from an external memory (SD card) takes more time compared to the XORing operation. Moreover, by offloading the CSPRN generation task to the ES, the MD can save resources. So, the CLOAK protocol is mobile centric and it does not need to exchange data in a PT format.

B. PSEUDO RANDOM NUMBER GENERATION

Pseudo-random number (PRN) is a stream of random or pseudo-random characters, used for generating the ciphertext in a stream cipher. It is a set of values or elements that are statistically random but is derived from a known starting point, called seed and typically the elements are repeated after a fixed interval [50]. The PRN is generated using a deterministic process and is reproducible. It is called "pseudo" random because the generator can reproduce the sequence for a specific seed value and thus the PRNs are not entirely random. In addition to cryptography, PRN is also used for simulations (e.g. for the Monte Carlo method [51], electronic games (e.g. for procedural generation [52], etc.

However, in stream ciphers, we generally use cryptographically secure pseudo-random numbers (CSPRNs). The CSPRNs are unpredictable i.e., for some given output bits of the key stream it is computationally infeasible to compute the subsequent bits. Another way of defining CSPRN is that, for a given 'n' consecutive bits of a key stream, no polynomial time algorithm can predict the next or preceding bits of the key stream. There is various method for generation of CSPRN, such as Middle Square Method (John Von Neumann 1946) [53], Linear Congruential generator (LCG) [54], Cubic Congruential generator (CCG).

In our implementation, we use the Advanced Encryption Standard (AES) for generating CSPRN [55]. AES is a secure and widely used symmetric-key based cryptographic algorithm, published by National Institute of Standards and Technology (NIST) in 2001 [56]. The encryption algorithm of AES requires two parameters: plaintext and a secret key. The main design principle of AES is based on substitution and permutation, that makes it faster for both software and hardware implementation [57]. The core functionality of AES occurs inside the substitution box or S-Box. By using AES, 128 bits data block can be encrypted by using three different key lengths of 128, 192 and 256 bits with 10, 12 and 14 rounds, respectively.

In CLOAK, we implement AES-128 bits for generating of 128 bits of a CSPRN. The parameters passed to the AES encryption algorithm, are seed/key (s) and sequence number (sn), figure-2. The sn is an integer that can be user defined or can be selected randomly. Similarly, the key (s) can be a user defined string or an integer chosen randomly by the system. These are the initial parameters of the AES algorithm, used for generating the first 128 bits of the CSPRN. The sn is incremented at each iteration for producing the subsequent 128 bits of the CSPRN (C). Subsequently, the ES needs to send C to the MD. To secure the transmission, the ES modifies C by XORing it with a pre-shared secret key S_k, which is defined the MD at the time of registration with ES. The ES generate $C'' \leftarrow C \oplus S_k$) and transmits C'' to the MD, as shown in Fig-2. If the size of $S_k < C$, the ES can replicate S_k to make the size equal to C. Notice that, the generation of multiple C'' with the same secret key S_k, has no effect on two-time pad attack. In addition, the generation of C'' also helps CLOAK to avoid other attacks, such as known plaintext attack and algebraic attack, as discussed in section-V.

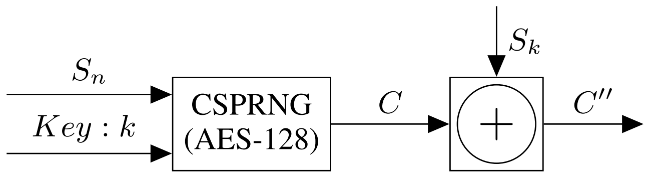


FIGURE 2. Cryptographically Pseudo random number generator.

We installed a Linux based server on AWS and implemented AES (128 bit) encryption algorithm on the server. To fetch the CSPRN, users can send a request to the server from it's mobile, mentioning the $< uid, fn, cs >$, as discussed above. For simplicity, we used the PHP rand-function() for generating the key s and sequence number sn in the ES. Since, we are using 128 bit AES, the size of the CSPRN is the multiple of 128 bits. In each iteration, the algorithm adds 128 bits to the key stream to make it equal to the requested cs, shown in algorithm-1. The initial values of s and sn must be same for both encryption and decryption operations. For this, the ES stores $< uid, fn, cs, sn, s >$ in it's database. When the ES receives a decryption request with $< uid, fn >$, it retrieves the corresponding sn and s from its database, generates the same CSPRN (size cs) and sends it back to the requesting MD. For more security, the ES can use encryption before storing the data in the database.

C. SECURITY ISSUES OF CLOAK

The security of CLOAK depends upon the security of its components (i.e, MD, ES and the communication channels). In the following, we analyze the security of these components in detail. The idea is to explore the vulnerabilities and to emphasise the security concerns of the CLOAK protocol.

1) *Security of MD:* Ensuring the security of the MD is the responsibility of OS and researchers have proposed various mechanism to overcome the security challenges of the same [58]–[60]. In CLOAK, we assume that the XORING and the read/write operation on the PT/CT can be preformed securely within the MD. This is the basic presumption of any encryption algorithm, i.e., the device on which the cipher is performed should be secure.
2) *Security of ES:* In MCC, the use of an ES or cloud is increasing to overcome the resource constraints of the MDs. It is performed by offloading the computation intensive operations to the ES. Providing security of a shared platform against internal and external attacks is a challenging task and is currently a major research issue for the Cloud Service Providers (CSPs). In our case, security of the CSPRN generator against modification or deletion of code/data is the responsibility of the CSPs. However, the protocol must ensure that the data obtained from a compromised ES (leaked or modified data) has no effect on the security of the protocol.
3) *Security of CM:* One of the basic requirement of a stream cipher is to protect both the CT and CSPRN (Key-stream) from the adversary. This is the most challenging task for the CLOAK protocol since the communications between MDs and ES can take place over an unreliable wireless medium in the MCC environment. Thus, the CLOAK protocol must ensure that the adversary retrieves no information about the PT, from CSPRN and/or CT.

Thus, in the CLOAK protocol, the main security challenge is to protect the CSPRN and CT pair from the adversary. Note that, the CSPRN and CT can be compromised in one of the following ways: (a) by fetching the CSPRN from ES and CT from the CM or (b) by compromising the two communication channels used for exchanging data between the ES and MDs, figure-1. We address this issue by modifying C (i.e. $C'' \leftarrow C \oplus S_k$) on ES before transmitting to the MD. In addition again we modify the the original CSPRN (C) by using deception techniques within the mobile devices (MDs) to generate a new CSPRN (i.e. $C \overset{k}{\rightarrow} C'$) and use C' to produce CT. The idea is to deceive the adversary. Moreover, the CLOAK protocol must be immune to other security challenges, such as two-time pad/reused key attack, Known plaintext attack, algebraic attack, man-in-the-middle, DoS and impersonation attacks.

IV. SECURING CLOAK

In this section, we try to address the above security challenges in detail. We begin our discussion with the deception technique, whereby we investigate techniques for modifying the original CSPRN within the mobile devices, i.e. $C \overset{k}{\rightarrow} C'$, for producing CT. Furthermore, to handle other security attacks, as discussed above, we modify the basic CLOAK protocol by securing the message communication between the CLOAK entities.

A. MODIFYING CSPRN

The security of a stream cipher depends on the security of the key-stream, i.e. CSPRN. Since we consider an unreliable communication medium, we investigate two randomized approaches (s-CLOAK and r-CLOAK) and a deterministic approach (d-CLOAK) for generating modified CSPRN (C'), figure-3. In figure-4, we show the block diagrams of all three approaches. For all approaches, we assume that the MD has received the CSPRN (C) from ES.

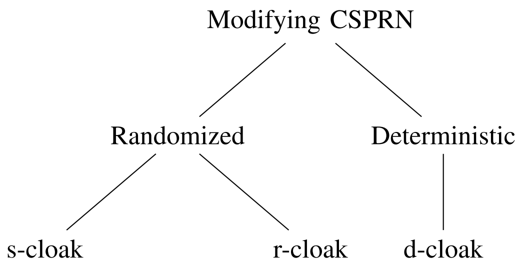


FIGURE 3. Approached for modifying CSPRN.

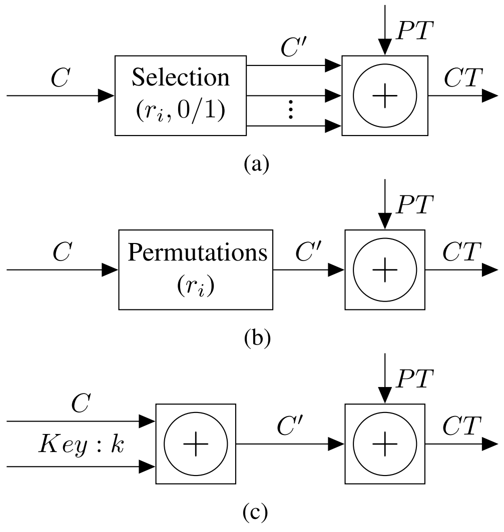


FIGURE 4. Block Diagram of s-CLOAK, r-CLOAK, d-CLOAK.

1) s-CLOAK

We begin by introducing a naïve approach by logically considering C to be cyclic of size n, figure-5a. In this, the MD selects a random number r in the range of $(1 \ldots n)$, which corresponds to a bit location in C. A modified CSPRN C' can be produced by shifting C by r bits, i.e. $C'[i] = C[(n \pm i + r - 1) \mod n]$, $\forall$ i bits. The shift can either be in the clockwise (0) or anticlockwise (1) direction, determined by a random variable. The cipher-text (CT) can be produced by:

$$CT[i] = PT[i] \oplus C[(n \pm i + r - 1) \mod n], \quad \forall \text{ i bits}$$

Thus, in naïve approach, we use two random variables, one for determining the number of shifts and other for the direction, figure-5a. The key-pair $(r, 0/1)$ can be shared between the communicating parties, as in any symmetric key cryptographic algorithm and can be used by the decryptor for generating the same C'. The algorithmic complexity of this approach is $O(n)$. However, notice that, for a given C and CT, a brute force attack can retrieve the PT in $O(n)$ by shifting through all bits of C.

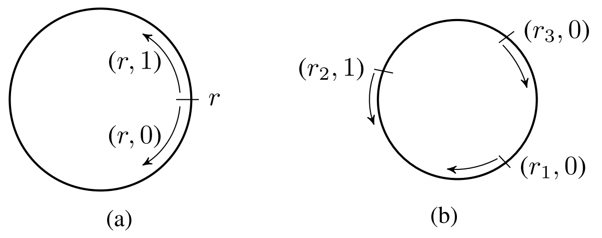


FIGURE 5. s-CLOAK with randomized key-pair $k = \{r_i, 0/1\}$.

To increase unpredictability, the above process can be repeated m times. We refer it as s-CLOAK. For this, the MD needs to select m pair of random numbers $(r_i, 0/1)$, for i = 1 to m to generate m modified CSPRNs $C' = C_1, C_2, \ldots, C_m$, shown in figure-5b, and perform the XORing operation to generate the CT, as follows:

$$CT = PT \oplus C_1 \oplus C_2 \oplus \ldots \oplus C_m$$

In s-CLOAK, the total size of the key-pair increases by a factor of m and the complexity of the algorithm as $O(nm)$, as it performs m XORing operations on n bits. Moreover, if m is equivalent to n, then the maximum complexity of the algorithm increases to $O(n^2)$. Also notice that the order in which the random numbers are selected, has no effect on the CT.

2) r-CLOAK

r-CLOAK is another randomized approach for modifying CSPRN. In this, we again use a set of m random numbers $(r_1, r_2, \ldots, r_m)$, indicating the bit positions in the range of $0 \ldots (n-1)$. However, unlike s-CLOAK, the order of the random numbers is important in r-CLOAK, which increase the randomness of the system. Another important distinction is that, r-CLOAK performs a single XOR operation; whereas s-CLOAK requires m-XORing to produce the CT, figure-4.

In r-CLOAK, the first m-bits of C' is obtained from the initial position of the m-random numbers. Then, the random numbers are incremented by one to obtain the next m-bits of C'. All successive bits of C' can be obtained by the same procedure, as shown in figure-6. The notation r_i^j indicates a shift of j bits from r_i in C. The relationship between the bits of C and C' is shown below. Notice that, if we change the order of the random numbers, the C' changes.

$$C'[jm+i] = C[r_i^j] = C[(r_i \pm j) \mod n], \ \forall\, i = 1 \ldots m, \quad j = 0 \ldots \lceil n/m \rceil$$

The complexity of the r-CLOAK is $O(n)$, as the pre-processing of $C \rightarrow C'$ requires $O(n)$ and also $PT \rightarrow CT$ can be generated in $O(n)$. The key shared between encryptor and decryptor is $(r_1, r_2, \ldots, r_m)$. In figure-6, the next bit is obtained by incrementing the random numbers. However, we can also use the decrement operation or randomly choose between the two operations for each r_i. In the later case, the r-CLOAK generates a sequence of key-pairs, similar to s-CLOAK.

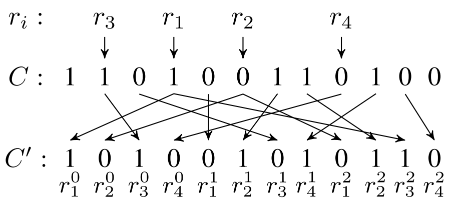

r_i :
r_3
r_1
r_2
r_4
C : 1 1 0 1 0 0 1 1 0 1 0 0
C' : 1 0 1 0 0 1 0 1 0 1 1 0
r_1^0 r_2^0 r_3^0 r_4^0 r_1^1 r_2^1 r_3^1 r_4^1 r_1^2 r_2^2 r_3^2 r_4^2

FIGURE 6. r-CLOAK with randomized key-pair: $k = \{r_i, 0/1\}$.

Both s-CLOAK and r-CLOAK can be implemented using a block/chunk-wise, which is important for memory constrained mobile devices (MDs). Instead of handling a large file, the MDs can recursively read a small chunk of its main memory, apply any of the above algorithms and write the result back to the secondary storage. The MD can either use the same set of m-random numbers or it can choose a different set for each chunk. For both s-CLOAK and r-CLOAK, the key can be generated by random function [61], [62] or pseudo-random number generator [53], [54]. The later is more preferable, as only the seed value needs to be shared between the encryptor and decryptor. Notice that, secrecy of C' depends upon the random numbers. We discuss the brute force attack for both s-CLOAK and r-CLOAK in section-VI.

3) d-CLOAK

This is a deterministic approach where a predetermined secret key k is used for generating modified CSPRN C'. In comparison with the previous approaches, in d-CLOAK the key k can either be user defined string or can be generated randomly in the MD. The C' is generated by XORing C with the secret key k, i.e. $C' = C \oplus k$. A pre-processing operation on the secret key, i.e. replication or trimming, may be required to make the size of k equal to PT. Finally, the CT is generated by performing another XORing operation between PT and C', as shown in figure-4.

Same as other symmetric key algorithms, the three approaches discussed above, need to exchange the key/key-pair between the encryptor and decryptor. Since this is not the main focus of this paper, we skip the details. However, popular techniques for exchanging keys include, secure channel [18], out-of-band channel (OOB) [63] and Diffie-Hellman algorithm [64]. The secrecy of the keys is the primary requirement of all cryptographic algorithms and it is also true for the above approaches.

The decryption is the inverse of the encryption procedure, for the above approaches. To retrieve the plain text, decryptor needs to have the C, key/key-pair and CT. The decryptor can obtain C form ES, CT from either ES or MD and the key from the MD. Then the decryptor generates C' using the above approaches and finally can obtain the PT by XORing CT with C'.

B. SECURING THE MESSAGE FLOW

In MCC, the message exchanges between MD and the ES takes place over insecure CM and is susceptible to various security threats. In this section, we address the issue by securing the messages exchanged in the CLOAK protocol. The goal is to protect all parameters used for fetching the CSPRN from the external ES. We assume that, all users are registered with the ES with user-name un and unique user-id (uid) for accessing its services. We also assume that the mobile device and the external ES, use a common one-way hashed function (e.g. SHA-1 and SHA-2 [65]) for protecting their respective messages. Figure-7, shows the message flow of our protocol.

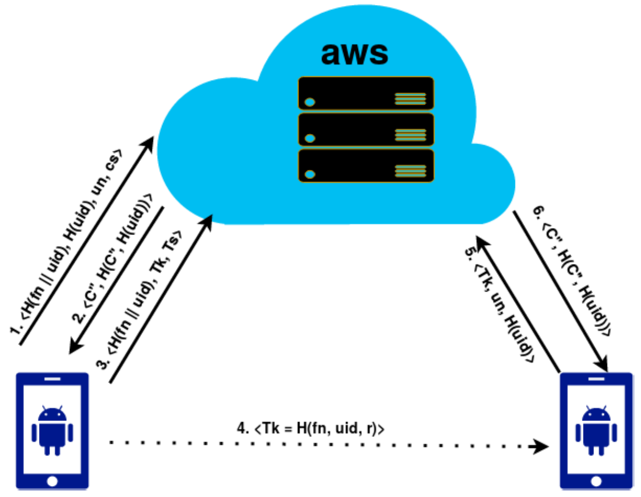


FIGURE 7. Message Flow in CLOAK.

In figure-7, message 1 and 2 is exchanged between MD and ES to obtain the CSPRN. The message 1 includes: $\langle un, cs, H(uid), H(fn \parallel uid)\rangle$, where $H(fn \parallel uid)$ is the hash value of concatenated filename fn and uid. The pair $\langle un, H(uid)\rangle$ is the credentials for validating user's identity. The parameters $H(fn \parallel uid)$ and cs are the unique file identifier and size of the CSPRN. On receiving this message, ES validates the user and on successful verification it generates $C'' \leftarrow C \oplus S_k$, as discussed in section-III-B. Then, the ES stores the unique file name $H(fn \parallel uid)$, $H(uid)$, cs, sn and s in its database. For more security, the ES can send one time password (OTP) to users mobile through an out-of-band channel (OOB) [63], for verifying user authenticity.

Then, the ES generates a hash value $h' = H(C'', H(uid))$ to prepare message 2, containing $\langle C'', h'\rangle$ and forwards it to the MD. The OTP can also be included in h'. On receiving this message, the mobile checks the integrity of the message by calculating $h'' = H(C'', H(uid))$ by comparing h'' with h'. If $h'' = h'$, then the MD considers C'' as valid. Finally, the MD retrieves C by XORing C'' with its pre-shared key S_k and uses it for the encryption operation in s-CLOAK, r-CLOAK or d-CLOAK protocol. Notice that, both MD and ES mutually validate each other before accepting any message.

The encryptor may like to share the file with multiple MDs. For this, the encryptor generates a token T_k and distributes it to the intended users. A mobile with a valid token can request CSPRN form the external ES and decrypt the ciphertext, as discussed above. Token generation and distribution process is shown in message 3 and 4 of figure-7. The token T_k is generated by hashing fn, uid and a random number r (i.e. $T_k = H(fn \parallel uid \parallel r)$). The encryptor creates message 3 containing $< H(fn \parallel uid), T_k, T_s >$, associating T_k with the filename for a timestamp T_s. Then the message is forwarded to the ES. The ES stores the information of this message in its database. The token is forwarded to the intended mobile users via a secure channel, shown in message 4.

With a valid T_k, the decryptor can request the ES for CSPRN. For this, the decryptor sends message 5 containing T_k, with user credential $\langle un, H(uid)\rangle$ for its verification. The ES validates the decryptor and checks the validity of the token T_k. If the T_k is found in the database, the sever checks corresponding timestamp (T_s) and retrieves the necessary parameters (sn, k) for generating the CSPRN. Finally, the CSPRN is forwarded to the decryptor as message 6, which is similar to message 2.

V. ATTACK ANALYSIS

The security threats on CLOAK can be imposed in two ways. An attacker may either try to find vulnerabilities in the ES or on CM. In this section, we consider both issues and perform the attack analysis on the CLOAK protocol.

A. BRUTE FORCE ATTACK

For Brute force attack, we assume that the adversary has obtained the CT and CSPRN (C), and wants to find the PT. In CLOAK, since $PT \neq C \oplus CT$, the attacker may either try to guess the C' or try to generate C' from C. The former case is equivalent to finding the secret key S in d-CLOAK. So, if the size of C' or S is $\geq$ 128 bits, then a brute-force attack needs to consider more than 2^{128} combinations to find the PT, which is computationally infeasible [49]. For the later case, the complexity of generating $C \rightarrow C'$ depends on the number of random numbers (i.e. m). In the following, we compute m for a given PT of size n ($\gg m$), that makes the brute-force attack computationally infeasible for s-CLOAK and r-CLOAK.

For s-CLOAK, to compute $C \rightarrow C'$, a brute force attack needs to determine the m random numbers r_i's in the range of $0 \ldots (n-1)$. For this, the attacker needs to consider all $\binom{n}{m}$ combinations of choosing r_i's, which can be bounded using Sterlings approximation [66] as: $\left(\frac{n}{m}\right)^m \leq \binom{n}{m} \leq \left(\frac{en}{m}\right)^m$. In s-CLOAK, since each r_i is associated with 0 or 1 with equal probability, the computational complexity of generating $C \rightarrow C'$ greater than $(\frac{2n}{m})^m$. This can be used for computing the minimum number of random numbers required to make the brute-force attack computationally infeasible. For example, for a known PT of size $n = 1\text{Mb} = 2^{20}$ bits, the complexity of finding $m \geq 8$ random numbers is more than 2^{128}. Similarly, for a PT size of n = 8 Mb, similar complexity can be obtained by choosing $m \geq 6$. In r-CLOAK, the generation of C' depends on the order of the random numbers. Thus, a brute force attack needs to consider all permutations of m random numbers. The complexity for the same is ${}^{n}P_m$ and it can be bounded as $\left(\frac{n}{e}\right)^m \leq {}^{n}P_m \leq \left(n\right)^m$. Again, associating each r_i with 0 and 1 with equal probability, the complexity of finding r_i's increases by a factor of 2^m, i.e., $\left(\frac{2n}{e}\right)^m$. This can be used to calculate the m for r-CLOAK to make the attack computationally infeasible.

B. TWO TIME PAD / REUSED KEY ATTACK

In a stream cipher, encrypting two different messages, say m_1 and m_2 using same key k is called a two time pad attack or reused key attack. That is, if an attacker intercepts both $c_1 \leftarrow m_1 \oplus k$ and $c_2 \leftarrow m_2 \oplus k$, then it can easily find the plaintext. For this, the attacker XORs c_1, c_2 (that is, $m_1 \oplus m_2 \leftarrow c_1 \oplus c_2$) and then determines m_1 and m_2 by using frequency analysis on $m_1 \oplus m_2$. However, this attck is not possible in CLOAK, as it encrypts each file with a different CSPRN obtained form a randomly generated sequence number sn and seed s using AES, shown in algo-1.

C. KNOWN PLAINTEXT ATTACK AND ALGEBRAIC ATTACK

A known plaintext attack tries to determine the secret key (or key stream in case of a stream cipher) from the known bits of a plaintext and its corresponding cipher text. Similarly, in an algebraic attack, an attacker tries to retrieve the secret key by finding and solving a system of the equation over a finite field [67]. Both attacks try to determine the secret key using different methodology. A known plaintext attack is not possible in CLOAK. This is because, from the known bits of a PT and CT, the attacker can only determine the corresponding bits of C'. To determine the subsequent bits of C', the attacker needs to know the original CSPRN (C). However, as discussed in section-III-B, the ES modifies C using S_k before transmitting to the MD. Thus, the C can only be determined if the shared secret (S_k) is known to the attacker. Similarly, an adversary must determine C for a successful algebraic attack. For this, the attacker must perform the algebraic attack on the CSPRN generation procedure, i.e. on AES algorithm in CLOAK. However, according to [68], [69], the algebraic attack is computationally infeasible on AES-128 using XL, XSL algorithms [70].

D. MAN-IN-THE-MIDDLE ATTACK

In Man-in-the-middle (MIM) attack, the attacker can eavesdrop and modify the messages exchanged between two the communicating parties. Both parties, unaware of the attacker, think that they are communicating directly with each other; but an MIM attacker can control this conversation by changing the original messages. In CLOAK, for MIM attack, the attacker may try to snoop on messages 1-6, in figure-7. As Hashing is used in all the messages, the attacker is unable to manage, predict or modify any information passed between the communicating parties. Also, the messages are verified mutually by checking the user's credentials and comparing the hash values.

E. INSIDER ATTACK

In CLOAK, an insider attack can take place to compromise the CSPRN or the user credentials stored on the ES. However, in our protocol, the ES stores the digest of all important parameters ($H(fn||uid)$, $H(uid)$, $H(fn||uid||r)$) in its database, that can not be used for retrieving any meaningful information. The information stored in the plain text format (sn, k, cs, T_s) can not be used without meaningful interpretation of the above parameters. Thus, the insider attack can not take place in CLOAK.

F. DENIAL OF SERVICE (DoS) ATTACK

In CLOAK, the ES validates the credentials of the user before providing its services. Also, the mobile checks the integrity of the message received from the ES by comparing the hash digest. Due to mutual verification, DoS attack is not possible in CLOAK.

G. IMPERSONATION ATTACK

For this, we consider two cases, i.e. mobile user impersonation and CSPRN impersonation. In CLOAK, user impersonation attack can happen while the mobile is requesting CSPRN from the ES. This can be avoided by verifying the authenticity of the user using OTP, as discussed above. Similarly, the same OTP can be used for countering the CSPRN impersonation by an attacker, by hashing the OTP with the CSPRN.

H. CHOSEN IV ATTACK

In stream ciphers, Initial Vector (IV) is generally used to generate the pseudo-random numbers using PRNG(K, IV) function. However, using the same IV for generating multiple PRNs is unsafe. A chosen IV attack tries to choose an IV to generate the PRN with a known key (K). In CLOAK, the CSPRN is generated using the AES-128 bit and the IVs are generated randomly for each file. So, Chosen IV attack is not possible in CLOAK.

VI. PERFORMANCE EVALUATION

In the following section, we discuss some significant properties of CLOAK. Subsequently to understand the complexity of our protocol, we present our experimental evaluations on five MDs.

A. PROPERTIES OF CLOAK

1) CSPRN (C) IS UNPREDICTABLE

As mentioned above, in a stream cipher, the PRN should be unpredictable. That is, for a given i consecutive bits of a PRN sequence, finding the remaining bits should be computationally infeasible. An unpredictable PRN is a cryptographically secure PRN (CSPRN). Now we show that algo-1, used for PRN generation is unpredictable. Algo-1, uses the Advanced Encryption Standard (AES) algorithm with sequence number (sn) and seed (s) as the input parameters, generated by using a random function. The AES algorithm has the *confusion* and *diffusion* properties, identified by Claude Shannon in 1945 [71], required for generating secure ciphers. In AES, confusion is based on the substitution principle, whereby the bits of a ciphertext is correlated with several bits of the secret key. The diffusion ensures that a single bit change in the plaintext should statistically alter half of the bits in the ciphertext and vice versa. The confusion and diffusion properties ensure that, for a given 128-bit output of AES, the probability that the i^{th} bit is 0 or 1 is 0.5. Moreover, since the sn is incremented at each iteration, each 128-bit output of the algorithm is distinct. Thus, the entire key stream generated by the algo-1 is unpredictable and produces cryptographically secure PRN.

2) FOR UNKNOWN k, GENERATION OF C' IS COMPUTATIONALLY INFEASIBLE

In s-CLOAK and r-CLOAK, the generation of $C \rightarrow C'$ is directly related to the m random numbers. If the random numbers are unpredictable and uniformly distributed, then as shown in the brute-force attack, the generation of $C \rightarrow C'$ is computationally infeasible. In d-CLOAK, C' is produced by XORing C with replicated k. The XOR operation on a string of unknown distribution with a uniformly distributed CSPRN, is uniformly distributed. Hence, C' is uniformly distributed and for $k > 128$ bits, generation of C' is computationally infeasible, as discussed above.

3) CLOAK IS SECURE CIPHER

The CLOAK protocol is similar to One Time Pad (OTP) encryption. In CLOAK, the length of the key-stream greater than 128 bits and the key-stream is different for each file, as shown in algo-1. Moreover, the proposed protocol is designed by considering the security challenges of the MCC environment. Thus, CLOAK is a secure cipher.

4) SMOOTH KEY DISTRIBUTION

The distribution of key-stream is a challenging task of a stream cipher [72], as the same key is used for both encryption and decryption operations. In CLOAK, the CSPRN generation and distribution is handled by the ES. The MD always downloads the CSPRNs from the ES, as required. It is assumed that the ES infrastructure can easily meet the demands of multiple users. Thus, the key distribution of our proposed protocol requires the minimum involvement of the MD.

5) LESS COMPUTATIONAL OVERHEAD ON MD

In CLOAK, the operations performed by the MD are read, write, XORing, calculating hash values and communication with ES. These are basic operations with low computational complexity and can be performed with limited computational resources.

B. EXPERIMENTAL RESULTS

The performance of an algorithm depends on the complexity and hardware capability of the device on which it is executed (i.e. processor, memory, cache). The two main factors affecting the performance of CLOAK are the time required for downloading CSPRN and the time required to perform the read, write and XOR operations in MD. To evaluate these factors we use five MDs of different configurations, shown in table-3. We place the CSPRN generator on the AWS cloud. In the following, we show the performance evaluation of r-CLOAK and d-CLOAK, as they are computationally similar to each other, i.e. ($O(n)$) algorithms.

We begin by discussing the performance of r-CLOAK and d-CLOAK on *M-1* for different file sizes between 1 MB to 10 MB, as shown in figure-8 and 9, respectively. For each file size, we repeat the experiment 50 times and present an average of the total time required for the encryption and decryption operations. The total time includes the following:

- *CSPRN Time:* The time required for sending CSPRN request to the external ES, generating CSPRN in the ES, downloading and modifying CSPRN in MD.
- *XOR Time:* Reading the plaintext or ciphertext from external memory, XORing it with CSPRN and writing the result back to the external memory.

TABLE 3. Mobile device configuaration.

Mobiles	M-1	M-2	M-3	M-4	M-5
Model name	YU Yureka	Xiaomi MI3	Samsung GT-I9505	Asus Z00LD	Oneplus A001
OS	Lollipop 5.1	Kitkat 4.4.4	Jelly Bean	Lollipop 5.0	Lollipop 5.1
API level	22	19	18	21	22
CPU	Octa-core 1.5 GHz	Quad-core 2.3 GHz	Quad-core 1.9 GHz	Quad-core 1.2 GHz	Quad-core 2.5 GHz
Chipset	Qualcomm Snapdragon 615	Qualcomm Snapdragon 800	Qualcomm Snapdragon 600	Qualcomm Snapdragon 410	Qualcomm Snapdragon 801
RAM	2GB	2GB	2GB	2GB	3 GB
GPU	Adreno 405	Adreno 330	Adreno 320	Adreno 306	Adreno 330
Battery	Li-Po 2500 mAh	Li-Ion 3050 mAh	Li-Ion 2600 mAh	Li-Po 3000 mAh	Li-Po 3100 mAh

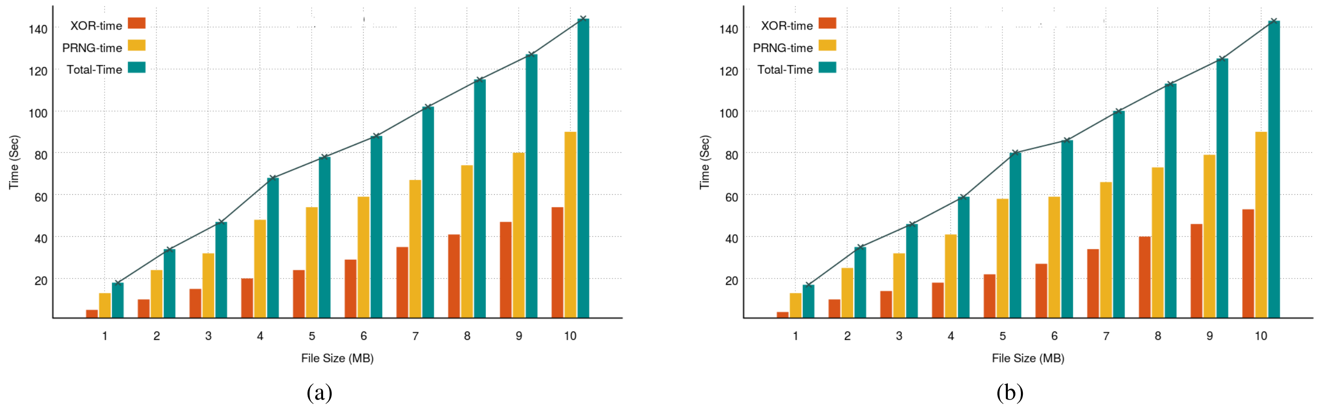


FIGURE 8. **Encryption/Decryption time for r-CLOAK for M5. (a) Encryption (r-CLOAK). (b) Decryption (r-CLOAK).**

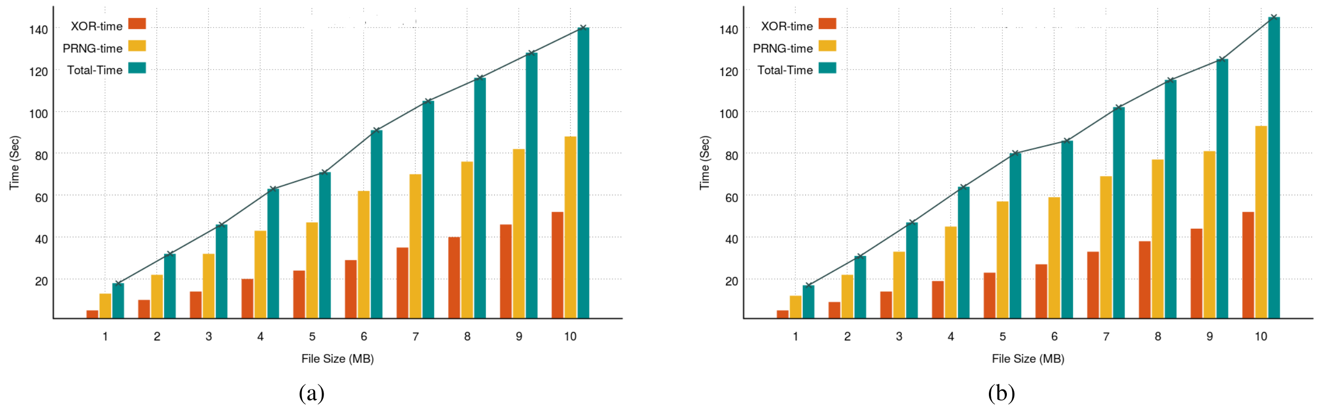


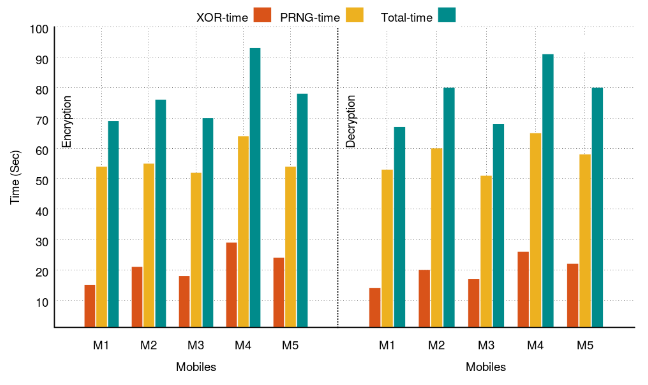
FIGURE 9. **Encryption/Decryption time for d-CLOAK for M5. (a) Encryption (d-CLOAK). (b) Decryption (d-CLOAK).**

As shown in figure-8 and 9, the total time for encryption and decryption increases with increasing file size and for all cases, the CSPRN time is more compared to XORing time. The CSPRN time depends on various factors, such as the location of the ES, the bandwidth of the underlay networks and the workload on the ES. In addition, since the size of C' is same as the file size, the CSPRN time is also directly proportional to the file size. Our experimental result shows that the total time varies linearly with increasing file size.

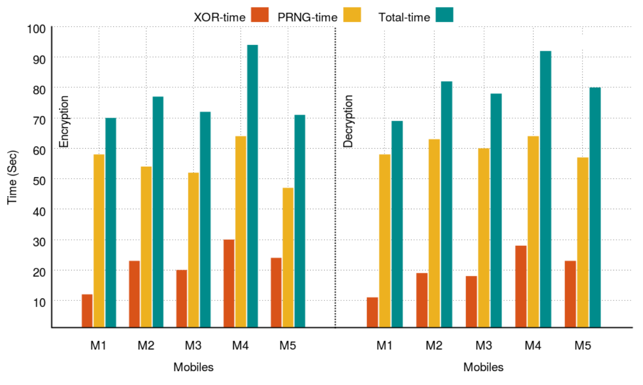
With growing demand of efficient and computationally intensive applications, the hardware specifications of MD are improving day by day [73], [74]. However, there is a very big gap between a highly configured mobile device and its lower counterpart. A resource intensive application may not have the desired performance on all MD. So, we test the performance of CLOAK on MD with different configurations. In this experiment, we fixed the file size as 5 MB and check the performance of both algorithms on five MD listed in table-3. The time complexity is shown in figure-10. The experimental result shows that our algorithm can be executed on a wide range of MD. However, the performance can vary depending upon the configuration. MD with more core processors has better performance.

The installable APK (Android Package Kit) size of r-CLOAK and d-CLOAK application is $1.3MB$ and $1.4MB$, respectively. The space occupied after installation may vary depending on the versions of the Android operating system running in the MD. We have installed both applications on all MD is shown in table-3 and found the r-CLOAK and d-CLOAK applications occupy approximately $3.5 - 4.5MB$ storage space after installation in the mobile phones. Hence, the application is light weight and it can be installed on MD with less memory.

We used the "GSam Battery Monitor" [75] Android application to measure the battery performance of our application on the *Xiaomi MI3* mobile device having Li-Ion 3050 mAh battery. To measure the battery consumption, we launched our application and perform the encryption operation on five files ranging from 1MB to 5MB. We notice a 1% decrease in the battery level, which includes the power consumed by the screen, wifi and other background processes. According to data recorded by GSam app, the battery consumption of CLOAK is only 5% of the total 1% battery consumption. Hence, the CLOAK protocol is efficient and can operate on MD with low configuration.



(a)



(b)

FIGURE 10. Encryption/Decryption time of 5Mb file on different devices. (a) r-CLOAK. (b) d-CLOAK.

VII. CONCLUSION

In this paper, we present a light-weight, stream cipher based encryption/decryption protocol for the mobile devices. The protocol is designed for the MCC environment. We handle the challenges of insecure wireless media by modifying the CSPRN and securing the message communication. The three variants of the proposed protocol are referred as s-CLOAK, r-CLOAK, and d-CLOAK, varying on the modification procedure of CSPRN. The s-CLOAK and r-CLOAK are randomized approaches, while the d-CLOAK is deterministic. We found CLOAK can resist various security challenges like brute force attack, MIM and Impersonation attacks. In addition, we studied the security of the messages exchanged between MD and the ES. To evaluate the protocol, we developed applications for android mobile phone and used the Amazon Web Service (AWS) for placing the CSPRN generator as ES. We have studied the performance of the protocol on five different MDs. Our experimental result shows that the proposed protocol can handle large files in an acceptable time frame.

ACKNOWLEDGMENT

Mahamudul Hasan wants to acknowledge ICT division of Ministry of Posts, Telecommunications and Information Technology, Government of the People's Republic of Bangladesh for their support in his Ph.D. program.

REFERENCES

[1] P. D'Arcy and L. E. Marketing, "CIO strategies for consumerization: The future of enterprise mobile computing," Tech. Rep., 2011.

[2] 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., 2014.

[3] R. Kemp, N. Palmer, T. Kielmann, and H. Bal, "Cuckoo: A computation offloading framework for smartphones," in *Mobile Computing, Applications, and Services*. Santa Clara, CA, USA: Springer, 2012, pp. 59–79.

[4] H. T. Dinh, C. Lee, D. Niyato, and P. Wang, "A survey of mobile cloud computing: architecture, applications, and approaches," *Wireless Commun. Mobile Comput.*, vol. 13, no. 18, pp. 1587–1611, 2013.

[5] W.-T. Tang, C.-M. Hu, and C.-Y. Hsu, "A mobile phone based homecare management system on the cloud," in *Proc. 3rd Int. Conf. Biomed. Eng. Inform. (BMEI)*, vol. 6. Oct. 2010, pp. 2442–2445.

[6] M. Yesudas, S. Gupta, and H. Ramamurthy, "Cloud-based mobile commerce for grocery purchasing in developing countries," *IBM J. Res. Develop.*, vol. 58, nos. 5–6, pp. 16:1–16:7, Sep. 2014.

[7] X. Chen, J. Liu, J. Han, and H. Xu, "Primary exploration of mobile learning mode under a cloud computing environment," in *Proc. Int. Conf. E-Health Netw., Digit. Ecosyst. Technol. (EDT)*, vol. 2. 2010, pp. 484–487.

[8] Z. Xiao and Y. Xiao, "Security and privacy in cloud computing," *IEEE Commun. Surveys Tuts.*, vol. 15, no. 2, pp. 843–859, 2nd Quart., 2013.

[9] U. T. Mattsson, "A practical implementation of transparent encryption and separation of duties in enterprise databases: Protection against external and internal attacks on databases," in *Proc. 7th IEEE Int. Conf. E-Commerce Technol. (CEC)*, Jul. 2005, pp. 559–565.

[10] C. Wang, G. Wang, Y. Sun, and W. Chen, "ARM realization of storage device encryption based on chaos and AES algorithm," in *Proc. 4th Int. Workshop Chaos-Fractals Theor. Appl. (IWCFTA)*, 2011, pp. 183–187.

[11] M. Sivasankari and S. Sujatha, "Performance metric analysis for mobile encryption algorithm using bouncy castle toolkit," in *Proc. Int. Conf. Emerg. Trends Electr. Comput. Technol. (ICETECT)*, 2011, pp. 1097–1101.

[12] B. A. O. Montoya, M. A. Muñoz, and S. T. Kofuji, "Performance analysis of encryption algorithms on mobile devices," in *Proc. 47th Int. Carnahan Conf. Secur. Technol. (ICCST)*, Oct. 2013, pp. 1–6.

[13] M. R. Baharon, Q. Shi, and D. Llewellyn-Jones, "A new lightweight homomorphic encryption scheme for mobile cloud computing," in *Proc. IEEE Int. Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Auton. Secure Comput., Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM)*, Oct. 2015, pp. 618–625.

[14] H. Wang, H. Zheng, B. Hu, and H. Tang, "Improved lightweight encryption algorithm based on optimized S-box," in *Proc. 5th Int. Conf. Comput. Inf. Sci. (ICCIS)*, Jun. 2013, pp. 734–737.

[15] O. B. Sahoo, D. K. Kole, and H. Rahaman, "An optimized S-box for advanced encryption standard (AES) design," in *Proc. Int. Conf. Adv. Comput. Commun. (ICACC)*, Aug. 2012, pp. 154–157.

[16] J. Y. Huang and I. E. Liao, "A searchable encryption scheme for outsourcing cloud storage," in *Proc. IEEE Int. Conf. Commun., Netw. Satellite (ComNetSat)*, Jul. 2012, pp. 142–146.

[17] M. Thamizhselvan, R. Raghuraman, S. G. Manoj, and P. V. Paul, "A novel security model for cloud using trusted third party encryption," in *Proc. Int. Conf. Innov. Inf., Embedded Commun. Syst. (ICIIECS)*, Mar. 2015, pp. 1–5.

[18] M. Ahmed, Y. Xiang, and S. Ali, "Above the trust and security in cloud computing: A notion towards innovation," in *Proc. IEEE/IFIP 8th Int. Conf. Embedded Ubiquitous Comput. (EUC)*, Dec. 2010, pp. 723–730.

[19] C. Xenakis, N. Loukas, and L. Merakos, "A secure mobile VPN scheme for UMTS," in *Proc. 12th Eur. Wireless Conf. Enabling Technol. Wireless Multimedia Commun. (Eur. Wireless)*, Apr. 2006, pp. 1–6.

[20] G. L. Prakash, M. Prateek, and I. Singh, "Data encryption and decryption algorithms using key rotations for data security in cloud system," in *Proc. Int. Conf. Signal Propag. Comput. Technol. (ICSPCT)*, Jul. 2014, pp. 624–629.

[21] W. Jiang, Z. Zhao, and C. de Laat, "An autonomous security storage solution for data-intensive cooperative cloud computing," in *Proc. IEEE 9th Int. Conf. eSci. (eScience)*, Oct. 2013, pp. 369–372.

[22] C. Pollalis, P. Charalampou, and E. Sykas, "HTTP data offloading using multipath TCP proxy," in *Proc. IEEE Int. Conf. Comput. Inf. Technol., Ubiquitous Comput. Commun., Dependable, Auton. Secure Comput., Pervasive Intell. Comput. (CIT/IUCC/DASC/PICOM)*, Oct. 2015, pp. 777–782.
[23] S. Song, B. Y. Choi, and D. Kim, "Selective encryption and component-oriented deduplication for mobile cloud data computing," in *Proc. Int. Conf. Comput., Netw. Commun. (ICNC)*, Feb. 2016, pp. 1–5.
[24] R. K. N. S. Krishna, T. J. V. R. K. M. K. Sayi, R. Mukkamala, and P. K. Baruah, "Privacy-preserving data management in mobile environments: A partial encryption approach," in *Proc. IEEE 13th Int. Conf. Mobile Data Manage.*, Jul. 2012, pp. 167–175.
[25] M. Hell, T. Johansson, and W. Meier, "Grain: A stream cipher for constrained environments," *Int. J. Wireless Mobile Comput.*, vol. 2, no. 1, pp. 86–93, 2007.
[26] F. Xiu-Tao, "ZUC algorithm: 3GPP LTE international encryption standard," *Inf. Secur. Commun. Privacy*, vol. 12, no. 12 p. 031, 2011.
[27] S. S. Gupta, A. Chattopadhyay, and A. Khalid, "HiPAcc-LTE: An integrated high performance accelerator for 3GPP LTE stream ciphers," in *Progress in Cryptology—INDOCRYPT*. Chennai, India: Springer, Dec. 2011, pp. 196–215.
[28] E. Biham and O. Dunkelman, "Cryptanalysis of the A5/1 GSM stream cipher," in *Progress in Cryptology—INDOCRYPT*. Calcutta, India: Springer, 2000, pp. 43–51.
[29] A. Biryukov, "Block ciphers and stream ciphers: The state of the art," in *IACR Cryptology ePrint Archive*. 2004, p. 94.
[30] A. Shamir, "Stream ciphers: Dead or alive?" in *Proc. ASIACRYPT*, 2004, p. 78.
[31] D. Hwang, M. Chaney, S. Karanam, N. Ton, and K. Gaj, "Comparison of FPGAtargeted hardware implementations of eSTREAM stream cipher candidates," in *Proc. State Art Stream Ciphers Workshop, (SASC)*, 2008, pp. 151–162.
[32] Y. Chen and W. S. Ku, "Self-encryption scheme for data security in mobile devices," in *Proc. 6th IEEE Consum. Commun. Netw. Conf.*, Jan. 2009, pp. 1–5.
[33] G. Rose, "A stream cipher based on linear feedback over GF(2^8)," in *Information Security and Privacy*. Brisbane, Australia: Springer, 1998, pp. 135–146.
[34] M. Hell, T. Johansson, A. Maximov, and W. Meier, "A stream cipher proposal: Grain-128," in *Proc. IEEE Int. Symp. Inf. Theory*, Jul. 2006, pp. 1614–1618.
[35] C. Berbain *et al.*, "Sosemanuk, a fast software-oriented stream cipher," in *New Stream Cipher Designs* (Lecture Notes in Computer Science), vol. 4986. 2008, pp. 98–118.
[36] H. Wu, "The stream cipher HC-128," in *New Stream Cipher Designs* (Lecture Notes in Computer Science), vol. 4986. Springer-Verlag, Apr. 2008, pp. 39–47.
[37] M. Boesgaard, M. Vesterager, T. Pedersen, J. Christiansen, and O. Scavenius, "Rabbit: A new high-performance stream cipher," in *Fast Software Encryption*. Lund, Sweden: Springer, 2003, pp. 307–329.
[38] D. J. Bernstein, "The Salsa20 family of stream ciphers," in *New Stream Cipher Designs* (Lecture Notes in Computer Science), vol. 4986. The ESTREAM Finalists, 2008, ch. 84, pp. 84–97.
[39] G. Paul and S. Maitra, *RC4 Stream Cipher and its Variants*. Boca Raton, FL, USA: CRC Press, 2011.
[40] C. de Cannière, Ö. Küçük, and B. Preneel, "Analysis of Grain's initialization algorithm," in *Proc. Cryptol. Africa 1st Int. Conf. Prog. Cryptol.*, 2008, pp. 276–289.
[41] S. Babbage and M. Dodd. (2006). *The Stream Cipher MICKEY 2.0, ECRYPT Stream Cipher*. [Online]. Available: http://www.ecrypt.eu.org/stream/p3ciphers/mickey/mickey/p3.pdf
[42] C. de Cannière and B. Preneel, "Trivium," in *New Stream Cipher Designs*. Berlin, Germany: Springer, 2008, pp. 244–266.
[43] M. A. Alomari and K. Samsudin, "A framework for GPU-accelerated AES-XTS encryption in mobile devices," in *Proc. IEEE Region 10 Conf. (TENCON)*, Nov. 2011, pp. 144–148.
[44] D. Arora, A. Raghunathan, S. Ravi, M. Sankaradass, N. K. Jha, and S. T. Chakradhar, "Exploring software partitions for fast security processing on a multiprocessor mobile SoC," *IEEE Trans. Very Large Scale Integr. (VLSI) Syst.*, vol. 15, no. 6, pp. 699–710, Jun. 2007.
[45] S. Harper and P. Athanas, "A security policy based upon hardware encryption," in *Proc. 37th Annu. Hawaii Int. Conf. Syst. Sci.*, Jan. 2004, p. 8.
[46] V. Goyal, O. Pandey, A. Sahai, and B. Waters, "Attribute-based encryption for fine-grained access control of encrypted data," in *Proc. 13th ACM Conf. Comput. Commun. Secur.*, 2006, pp. 89–98.
[47] J. Bethencourt, A. Sahai, and B. Waters, "Ciphertext-policy attribute-based encryption," in *Proc. IEEE Symp. Secur. Privacy (SP)*, May 2007, pp. 321–334.
[48] T. Ishiguro, S. Kiyomoto, and Y. Miyake, "A key-revocable attribute-based encryption for mobile cloud environments," in *Proc. Int. Conf. Secur. Cryptograph. (SECRYPT)*, Jul. 2013, pp. 1–11.
[49] C. Paar and J. Pelzl, *Understanding Cryptography: A Textbook for Students and Practitioners*. Berlin, Germany: Springer, 2009.
[50] (2015). *Pseudo-Random Numbers. (N.D.) Computer Desktop Encyclopedia (1981–2015)*. [Online]. Available: http://encyclopedia2.thefreedictionary.com/pseudo-random+numbers
[51] E. Cashwell and C. Everett, *Monte Carlo Method*. London, U.K.: Pergamon Press, 1959.
[52] M. Hendrikx, S. Meijer, J. Van Der Velden, and A. Iosup, "Procedural content generation for games: A survey," *ACM Trans. Multimedia Comput., Commun., Appl. (TOMM)*, vol. 9, no. 1, p. 1, 2013.
[53] J. von Neumann "John von Neumann," in *Proc. Quantum Algebra Symmetry*, 2010, p. 576.
[54] P. L'ecuyer, "Tables of linear congruential generators of different sizes and good lattice structure," *Math. Comput. Amer. Math. Soc.*, vol. 68, no. 225, pp. 249–260, 1999.
[55] A. E. Standard, "Federal information processing standards publication 197," in *Proc. FIPS PUB*, 2001, pp. 46–53.
[56] N. F. Pub, "197: Advanced encryption standard (AES)," *Federal Inf. Process. Standards Publication*, vol. 197, pp. 0311–441, Nov. 2001.
[57] K. Chen *et al.*, "Dragon: A fast word based stream cipher," in *Information Security and Cryptology—ICISC*. Seoul, South Korea: Springer, 2004, pp. 33–50.
[58] M. Lange, S. Liebergeld, A. Lackorzynski, A. Warg, and M. Peter, "L4Android: A generic operating system framework for secure smartphones," in *Proc. 1st ACM Workshop Secur. Privacy Smartphones Mobile Devices*, 2011, pp. 39–50.
[59] M. Ongtang, S. McLaughlin, W. Enck, and P. McDaniel, "Semantically rich application-centric security in android," *Secur. Commun. Netw.*, vol. 5, no. 6, pp. 658–673, 2012.
[60] D. Barrera, H. G. Kayacik, P. C. van Oorschot, and A. Somayaji, "A methodology for empirical analysis of permission-based security models and its application to android," in *Proc. 17th ACM Conf. Comput. Commun. Secur.*, 2010, pp. 73–84.
[61] H. Niederreiter, *Random Number Generation and Quasi-Monte Carlo Methods*. Philadelphia, PA, USA: SIAM, 1992.
[62] S. K. Park and K. W. Miller, "Random number generators: Good ones are hard to find," *Commun. ACM*, vol. 31, pp. 1192–1201, Oct. 1988.
[63] A. D. Griefer, "Message transmission using out-of-band signaling channel," U.S. Patent 5 615 213 A, Mar. 25, 1997.
[64] R. Canetti and H. Krawczyk, "Analysis of key-exchange protocols and their use for building secure channels," in *Advances in Cryptology—EUROCRYPT*. Innsbruck (Tyrol), Austria: Springer, 2001, pp. 453–474.
[65] N. Sklavos and O. Koufopavlou, "On the hardware implementations of the SHA-2 (256, 384, 512) hash functions," in *Proc. Int. Symp. Circuits Syst. (ISCAS)*, vol. 5. May 2003, pp. V-153–V-156.
[66] T. K. Moon and W. C. Stirling, *Mathematical Methods and Algorithms for Signal Processing*. Englewood Cliffs, NJ, USA: Prentice-Hall, 2000.
[67] A. Canteaut *et al.*, "Open problems related to algebraic attacks on stream ciphers," in *Coding and Cryptography* (Lecture Notes in Computer Science), vol. 3969, Ø. Ytrehus, Ed. Berlin, Germany: Springer-Verlag, 2005, pp. 120–134. [Online]. Available: http://dx.doi.org/10.1007/11779360_10
[68] R.-P. Weinmann, "Evaluating algebraic attacks on the AES," M.S. thesis, Techn. Univ. Darmstadt, Darmstadt, Germany, 2003.
[69] S. Murphy and M. Robshaw, "Comments on the security of the AES and the XSL technique," *Electron. Lett.*, vol. 39, no. 1, pp. 36–38, 2003.
[70] N. Courtois and J. Patarin, "About the XL algorithm over GF(2)," in *Proc. CT-RSA*, vol. 2612. 2003, pp. 141–157.
[71] C. E. Shannon, "A mathematical theory of communication," *ACM SIGMOBILE Mobile Comput. Commun. Rev.*, vol. 5, no. 1, pp. 3–55, 2001.
[72] R. A. Rueppel, *Analysis and Design of Stream Ciphers*. Berlin, Germany: Springer, 2012.
[73] A. Carroll and G. Heiser, "An analysis of power consumption in a smartphone," in *Proc. USENIX Annu. Tech. Conf.*, vol. 14. Boston, MA, USA, 2010, pp. 1–41.
[74] J. G. Caudill, "The growth of m-learning and the growth of mobile computing: Parallel developments," *Int. Rev. Res. Open Distrib. Learn.*, vol. 8 no. 2, Jun. 2007.
[75] (2015). *GSam Battery Monitor (Google Play Store)*. [Online]. Available: https://play.google.com/store/apps/details?id=com.gsamlabs.bbm&hl=en/

AMIT BANERJEE received Ph.D. degree from the Department of Computer Science, National Tsing Hua University, Taiwan, in 2009. Since 2011, He has been an Assistant Professor with the Department of Computer Science, South Asian University, New Delhi, India. From 2009 to 2011, he served as an Engineer with the Industrial Technology Research Institute, Taiwan. His research interest includes cloud computing, IoT, network security, mobile *ad hoc*, and sensor networks.

MAHAMUDUL HASAN received the B.Sc. Eng. degree in computer science and telecommunication engineering from the Noakhali Science and Technology University, Bangladesh, the master's degree in computer science from the Department of Computer Science, South Asian University, New Delhi, India, in 2013, where he is currently pursuing the Ph.D. degree. His research interest includes mobile computing, cloud computing, IoT, network security. He is also serving as a Lecturer with the Department of Computer Science and Telecommunication Engineering, Noakhali Science and Technology University, Bangladesh. He received prestigious ICT and Bangabandhu fellowship from the Bangladesh Government for Ph.D. and master's studies, respectively.

MD. AUHIDUR RAHMAN received B.Sc. Eng. in computer science and telecommunication engineering from Noakhali Science and Technology University, Bangladesh, in 2014, the master's degree in computer science from South Asian University, New Delhi, India, in 2016. He is currently serving as a Lecturer with the Institute of Information Technology, Noakhali Science and Technology University, Bangladesh.

RAJESH CHAPAGAIN received the bachelor's degree in computer application form University of Pune, Pune, India. He received the master's degree in computer science from South Asian University, New Delhi, India, in 2016.