Enhancing the Security and Privacy of WBAN-Based Distributed Storage Systems

Sufyan T. Faraj Al-Janabi
College of Computer,
University of Anbar
Ramadi, Iraq
sufyantaih@yahoo.com

Ali J. Dawood
College of Computer,
University of Anbar
Ramadi, Iraq

Abeer Dawood Salman
College of Computer,
University of Anbar
Ramadi, Iraq

Abstract— **The sensors of a Wireless Body Area Network (WBAN) that are spread inside and outside of a human body collect medical information and transfer it into the back-end server that is managed by the hospital or medical center; where the professionals analyze this data. This server is considered a very critical device because it has sensitive information. Therefore, the security and privacy of that server must be defended. Networked storage systems have become an efficient way to use in WBAN. It can be divided into two types: Centralized storage systems and Distributed Storage Systems (DSSs). Storing data at a single server is simple way but it can lead to a single point of failure (whether being a normal failure or due to security attacks). On the other hand, the structure of DSS can tolerate the openness and failure. In this paper, enhanced security and privacy DSS system is developed using public key cryptography to store patient information. The storage of information is distributed among individual nodes spread in the network and simultaneously the security requirements (like confidentiality, reliability, authentication, and dynamic integrity) are achieved by utilizing more powerful algorithms. The main phases of operation of the proposed system are described. Also, the results of system prototype simulation are discussed at the end of the paper.**

Keywords— distributed storage; privacy; security; sensors; WBAN; wireless networks

I. INTRODUCTION

Networked storage systems have gained prominence in recent years; this storage has become an efficient way to use in Wireless Body Area Networks (WBANs). These can be divided into two main types. The first one is based on center point. This strategy is called "centralized storage" that is distinguished by having a central server control of all computing power, processing, and back-ups. All users are sharing the resources of that server [1]. Using this type of storage usually cannot provide the level of dependability or/and availability because it leads to a single point of failure [2].

The other type of storage does not depend on any center point and it is called Distributed Storage System (DSS). The definition of distributed storage systems according to Tanenbaum is a "collection of independent computers that appear to the users of the system as a single computer" [3]. The environment of DSS is usually an ad-hoc network that consists of a group of individually unreliable data storage nodes but are as a group used to store data files over long periods of time with high reliably [4]- [6]. The structure of DSS must be tolerant to the openness and failure, and must achieve security, scalability, concurrency, and transparency [3].The storage must be distributed and redundant to ensure these security and operational requirements [7]. The reliability of this system came from using its redundancy especially when node fails. The redundancy usually is achieved using two schemes: the first is *replication code* and the second is *erasure code*, or a mix between them [7] [8].

DSS is used to ensure security and privacy of the high level of WBANs architecture, where the patient information is stored. The aim of this paper is to present an enhanced system capable for providing the confidentiality, dependability, integrity, and authentication as the same time with using DSS environment.

II. RELATED WORK

In this section a survey of some significant related works that deal with a data storage system in WBANs is presented:

- In 2007, Ball, Grant, So, Spurrett, and de Lemos [9] proposed fragmentation redundancy-scattering (FRS) technique that is fragmented the confidential information into insignificant fragments, and scattered these fragments in a redundant fashion through a network. Two algorithms are developed to maintain a constant number of fragment replicas: one based on the game of life, and the other based on roaming ants. This paper researched the use of autonomous agents combined with an intrusion tolerance technique for providing secure and dependable storage for ad hoc networks.

- In 2009, Wang, Ren, Lou and Zhang [10] proposed a scheme achieve most security requirements in data storage. In the initial data storage process, they utilized perfect secret sharing and erasure coding to guarantee data confidentiality and dependability. Based on the principle of algebraic signatures, they constructed efficient dynamic data integrity checking scheme to ensure the integrity of data shares. A weak point of this scheme is that it does not allow a third party to carry out integrity checks. This is quite unsuitable in WBAN applications because we want the local server to verify the integrity of the collected data.
- In 2012, Han, Omiwade, and Zheng [11] designed progressive data retrieval PDR algorithm for distributed storage systems that is highly computation-efficient and communication-optimal algorithm. The communication and computation costs for data retrieval are minimized by utilizing intermediate computation results and retrieving only the minimum data required for successful data reconstruction, respectively. The proposal shows that decentralized fountain codes and PDR for distributed storage systems are most suitable to networks without Byzantine storage nodes because of the associated minimal computation cost for fountain codes, and the minimal data retrieval cost of PDR.
- In 2013, Juarez, Oggier and Datta [12] proposed a new decentralized erasure coding that is reduce the network traffic required to archive replicated data in distributed storage systems. The proposed approach exploits the presence of data that is already replicated through the system and distributes the redundancy generation among those nodes that store part of this replicated data, which in turn reduces the total amount of data transferred during the encoding process. By storing additional replicated blocks at nodes executing the distributed encoding tasks, the necessary network traffic for archiving can be further reduced.

III. The Proposed Security and Privacy of WBAN-Based DSS

Before beginning to explain the specifics of the proposed system, the initialization process must be illustrated. Initialization consists of two phases: The first is building a simple Microsoft Access Database 2010 that is containing patient's information; the second is creating network connection between servers. Microsoft Visual C# has been used in order to write the software because it enjoys many characteristics such as its simplicity for implementing graphical user interface (GUI), and it is considered one of the top of the currently used programming languages.

The overall system consists of many servers: one *primary server* (PS), collection of *storage servers (SSs)*, and one or more *reader servers (RSs)*. The main phases of the system operation are distributed among all of these servers. The phases of distributed data security and privacy system in WBAN are illustrated in Figure 1. Below is a brief description of the sever types:

- ***PS***: It is a server that resides in the hospital and contains the main database (DB) which includes the patient information.
- ***SS***s: These servers are distributed in the network and used to store data after encoding and spreading by PS.
- ***RS***: At least one of such servers is needed to read the patient data that is stored at SS.

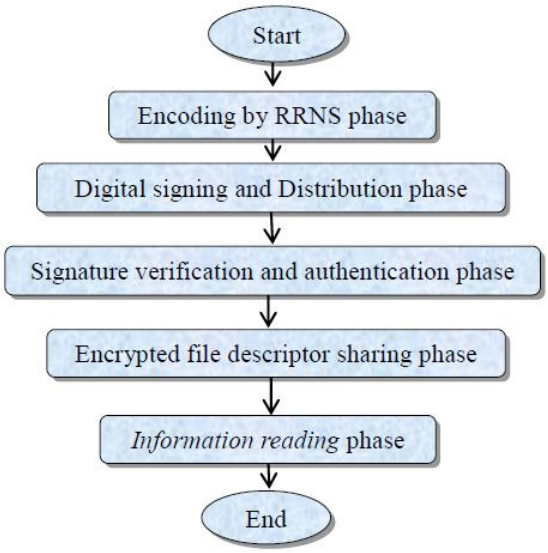

Fig. 1. Simplified System Model Phases.

A. Encoding By RRNS Phase

PS uses the Redundant Residue Number System (RRNS) technique for encoding the sensitive data. This technique needs some parameters which are: *h*, *r*, *moduli*, and *bytes*, where *h* is non-redundant residue, *r* is redundant residue, *moduli* is the key for encoding and decoding process, and *byte*s represents the number of the characters that are based on it the information is split [2]. It is important to note that the RRNS is a technique for secret sharing (or secure information distribution) among multiple parties rather than a traditional encryption technique (such as AES, Blowfish, etc.). The use of such a secret sharing technique is crucial for the proposed system. Traditional encryption algorithms are not suitable to be used instead of it in any direct or simple manner. Indeed, the security of RRNS can be increased to the required level by increasing the key size.

Two libraries of moduli (keys) are constructed to be used in the system: One is called *'moduli'* as shown in Table 1, and the other library is called *'modules of big prime numbers'* as shown in Table 2. Only small sample parts of these libraries are shown in these two tables. These keys must meet the requirements such as being pairwise prime, positive integers, and without loss of generality.

TABLE I. MODULI LIBRARY.

m1	65536	m2	65533	m3	65531
m4	65529	m5	65527	m6	65525
m7	65521	m8	65519	m9	65509
m10	65503	m11	65501	m12	65497
m13	65491	m14	65489	m15	65479

TABLE II. MODULES OF BIG PRIME NUMBERS.

m1	6461333093	m2	6461333101	m3	6461333117
m4	6461333143	m5	6461333171	m6	6461333227
m7	6461333267	m8	6461333269	m9	6461333311
m10	6461333321	m11	6461333353	m12	6461333357
m13	6461333369	m14	6461333387	m15	6461333399

PS selects (h+r) keys from the built library, and finds the range of these moduli by applying Eq. (1):

$$M = \prod_{p=1}^{h} m_p \quad (1)$$

PS takes the patient data as one file and converts it to binary form and splits the binary bits result into records of b bits based on the number of the byte used. For example, if PS used first library of moduli, it can split every 16 bits of records, and it divide every 32 bits if PS used the second library. The number of the records s is found by Eq. (2):

$$s = \text{no. of characters in file / bytes} \quad (2)$$

After that PS converts every record into decimal form X. If X is less than the moduli, the corresponding residue will be equal to decimal content and the storage system will be disclosed. To solve this problem, a random number $0 \leq \text{rand} < M$ is added to these decimal digits. Then the residues are calculated by applying Eq. (3):

$$x_p = X \bmod m_p \quad (3)$$

As a result of this phase, $s(h+r)$ residues are formed; they will be propagated to the SSs later. The distribution process is done without constraining. Only different residues belong to same record are distributed to a different server.

B. Digital Signing and Distribution Phase

After the residues are calculated, they must be a signed and then spread to storage servers. The signing process is done by using Digital Signature Algorithm (DSA) [13]. The result of signature process generates signature values ss and rr. These values are appended with the residues. h+r reliable storage servers (SSs) are chosen from a number of trusted servers. These SSs must be connected to the network that is created by primary server (PR). Finite loop of the distribution process begins with first SS and ends at the last SS.

C. Signature Verification and Authentication Phase

This phase is divided into two sub phases: verification, and storage and authentication phases. The verification phase is done when SSs receive the residues from PS. The storage process is accrued when SS ensures that the residues have not been modified during the transmission.

- ***Verification phase:*** The verification process is started by breaking the incoming message into the three parts: *residues*, *ss*, and *rr* (results of signature process). All these values are entered to the verification process to product the verification value v that is compared with the rr, if they are equal, the message is not modified.
- ***Storage and authentication*:** This sub phase is responsible for storage and achieving authentication at the same time. Each residue is stored with identification (ID) of its location as shown in Table 3 bellow.

TABLE III. FORM OF STORING RESIDUES.

$\mathbf{ID_1}$	Residue_1
$\mathbf{ID_2}$	Residue_2
$\mathbf{ID_s}$	Residue_s

The ID_s must be sent to the PS in order to store them in the file descriptor (FD). To enforce the confidentiality and ensure the authentication, RSA algorithm (See also [13] for example) is used. IDs are encrypted and signed with the public key of the PS and relative SS private key. This process is repeated for every SS where $s(h+r)$ IDs are arrived to the PS. For most small to medium scale systems, the public keys of relevant parties can be assumed to be manually distributed using an out-of-band style. However, in large scale applications a trusted third party (or public-key infrastructure) might be required.

D. Encrypted File Descriptor Sharing

This phase is considered the core for recovering original information on reader server RS after spreading the residues to set of reliable SSs since the RS cannot read the data before sharing the FD. This phase is divided into several sub phases. The first group of sub phases deals with receiving and decryption the IDs. And the second group deals with creating, signing, encrypting, and sending the FD, as follows:

Sub phases on the Identifiers:

- *Receiving:* When PS receives IDs from the h+r SSs, PS keeps $s(h+r)$ encrypted IDs that represent the positions of storage residues at SSs.

- *Decryption the IDs:* Because SS cipher the IDs with the public key of the PS, only the PS can decipher to recover the right IDs.

Sub phases on the file descriptor:

- *Creating FD:* In order to create FD, *I/O Stream* Class has been used. The contents of FD are: *moduli*, *added random number*, the value of *h*, *r*, *bytes*, and also the *IPs* of the SSs with its decrypted IDs.
- *Signing FD:* When PS intends to send FD to the RS, the signing process is done by applying the DSA algorithm on that FD. The two signature parameters (*ss1*, *rr1*) are calculated.
- *Encrypting FD:* The DSA signs FD only and does not encrypt it, therefore it must be encrypted with public key of the RS that is wanted to share the FD.
- *Sending FD:* PS selects the address (IP) of the RS and it is begins to send signed and encrypted FD.

E. Information Reading Phase

Reading information process works by several steps. These steps are distributed among the RS and SS, as follows:

- **Processes in the Reader Server:** RS first receives a message from PS containing signed and encrypted FD. This message is split into: *FD*, and the value of DSA parameters. FD is decrypted with RS's privet key to ensure the confidentiality and authentication. The verification process is achieved on the decrypted FD. Comparison between the signature and verification values (*v* and *rr*) is done to check if the message is modified or not. After verification process is completed, the FD content is isolated. RS needs to obtain the residues that are stored at different SSs, therefore the extracted IDs from FD must be sent to the SSs depending on the extracted IP. The sending process must be secure so the IDs are encrypted with the public keys of the SSs using RSA algorithm (As the messages to be encrypted are very small in size, there is a non-significant public key encryption/decryption time).
- **Processes in the Storage Server:** All SSs receive the identifiers that are specified for the locations of storing residues. These IDs are encrypted by RS already. Therefore, each of them must be decoded with the SS's private key and then compared these with its IDs to check if all received IDs are compatible with local IDs. Then the equivalent residues are sent back to the RS.
- **Processes in** the **Reader Server:** The sent residues from SSs are received at RS to enable it to recover the patient information. These residues result from RRSN algorithm. In order to recover the original data, these residues must be decoded. The decoding process is explained in the next section.

IV. INFORMATION DECRYPTION

Three techniques have been investigated in the proposed system for decrypting the information, as described below.

A. Chinese Remainder Theorem (CRT)

This technique has been applied at first on the RRNS. It works as follow: Range of moduli are calculated by applying Eq. (1). These keys are obtained from FD. Then, we apply Eq. (4) to recover the original decimal integer number.

$$X = \left(\sum_{p=1,h} x_p \frac{M}{m_p} b_p\right) mod\ M \qquad (4)$$

For each $p \in [1, h]$, b_p is multiplicative inverse of M/m_p modulo m_p, that is;

$$\left(b_p \frac{M}{mp}\right) mod\ m_p = 1.$$

Eq. (4) is applied for *h* residues.

B. Base Extension (BEX) with Mixed Radix Conversion (MRC)

In spite of CRT is a classical algorithm but the implementation of it is computationally intensive for large moduli values when dealing with modular operations with a large value of range (M) [14]. In order to avoid processing of large integers, the alternative method that is widely used is the base extension (BEX) operation in conjunction with the mixed radix conversion (MRC) method [15]. MRC is formulated as follows:

Take *h* of the received residues (x_1, x_2, …, x_h) with corresponding moduli (m_1, m_2, m_3, . . ., m_n) and find set of mixed radix digits (a_1, a_2, . . ., a_n) by applying Eq. (6), the decimal equivalent of the residues can be determined by applying Eq. (5)

$$X = a1 + a2mi + a3m1m2 + \cdots + an \prod_{i=1}^{h-1} mi \qquad (5)$$

where the mixed radix digits (MRD) are given as:

$$a_1 = x_1$$

$$a_2 = ((x_1 - a_1){m_1}^{-1})m_2$$

$$a_h = (\left(\left(\ldots\left((x_h - a_1){m_1}^{-1} - a_2\right){m_2}^{-1} - \cdots -\right)a_{h-1}\right){m_{(h-1)}}^{-1})m_h \qquad (6)$$

C. New Chinese Remainder Theorem (CRT I)

New Chinese Remainder Theorem (CRT I) is a modified version of the traditional Chinese Remainder Theorem. It is designed to make the computations faster and efficient without any overheads. The Chinese Remainder Theorem demands a slow large modulo operation while the Mixed Radix Conversion requires finding the mixed radix digits which is a slow process. In the CRT I, the weighted number can be retrieved faster because the operations are done in parallel, without depending on other results [16]. This Theorem operates as follow:

Given the *h* residue numbers (x_1, x_2,..., x_h); with corresponding moduli (m_1, m_2,... ,m_h) the weighted number *X* can be computed using Eq. (7):

$$X = [x_1 + k_1 m_1(x_2 - x_1) + k_2 m_1 m_2(x_3 - x_2) + \cdots$$

$$+ k_{n-1} m_1 m_2 \dots m_{n-1}(x_n - x_{n-1})] \bmod m_1 m_2 m_{n-1} m_n \quad (7)$$

where;

$$k_1 = (m_1)^{-1} \bmod m_2 m_3 \dots m_n,$$

$$k_2 = (m_1 m_2)^{-1} \bmod m_3 m_4 \dots m_n$$

$$k_{(n-1)} = (m_1 m_2 .. m_{n-1})^{-1} \bmod m_n \quad (8)$$

As shown above, all these algorithms aim to find the decimal numbers that represent the patient information. To recover this data, these numbers are reconverted to binary form, and then the original characters are obtained. After all that, the information is added to the new DB at RS and putted in the record specified for each patient.

V. System Performance

In this section, the performance of the implemented prototype system is discussed based on the code efficiency of RRNS with different key values and the algorithms used for decoding RRNS. The main reason for this is to give indication on the best methods that are used to produce a secure and efficient system at the same time. Our measurements have been done using systems with AMD Turion X2 processor (2GHz) and 2GB RAM running Windows 7 OS.

A. Code Efficiency of RRNS

In our implementation, PS splits the data into records according to the value of moduli such as it can use 2 or 5 bytes. Suppose PS is using the keys from Table 1, the largest value of the module is 65536 that is equal to 2^{16}. This means that PS can split a file into every 16 bit (2 byte) only with at least h=2 because if h =1 the range M according to Eq. (1) will be an error because 2^{16}>M, and this is not allowed in the RRNS, therefore the value of *h* must be greater than 1. Code Efficiency (CE) is calculated according to Eq. (9):

$$\emptyset = b/e \quad (9)$$

where

$$b = \log M,$$

$$e = 16(h+r)$$

.

If the patient data was large, splitting file characters to every 16 bits will produce large numbers of records and the time of the encoding will increase. Therefore, PS can use the moduli at Table 2 that can split the data to every 32 bits (5 bytes) because the largest value of these keys is 6461333947. The CE is calculated as the same Eq. (9) except that $e = 32\ (h + r)$. The comparison of using different byte numbers is shown in the Table 4. As shown in the table, RRNS with big moduli value results in increasing CE.

TABLE IV. Code Efficiency with 2- and 5-byte moduli.

h+r	CE (16 bits)	CE(32 bits)
2+1	0.999656758	1.018412026
4+3	0.99967964	1.018412027
6+5	0.999698678	1.018412027
9+5	0.999712175	1.018412027

B. The Performance of Decoding algorithms

In this sub section, the decoding algorithms are evaluated. The performance of these algorithms is calculated depending on the following parameters: Execution time and cyclomatic complexity (CC). The Time (in millisecond) of executing each algorithm is shown in Figure 2, where it can be noted that CRT I is faster in reading information and recovering the original decimal number. Even when the PS encodes the data with large modules, CRT I remain executes with the least time.

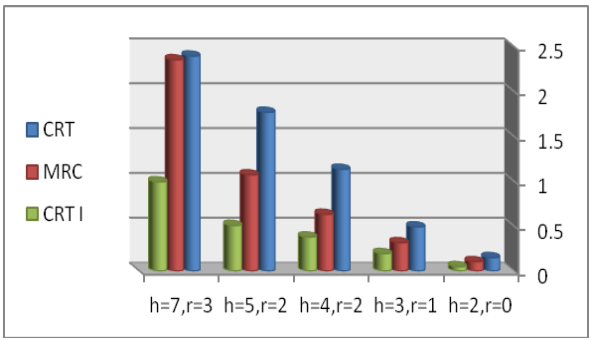

Fig. 2. Execution Time of the Decoding Algorithms.

CC measurement has been used to show the complexity of a program. This metric is based on a control flow representation of the program. Control flow describes a program as a graph which consists of nodes and edges. The difference between the above reading algorithms is illustrated in the Figure 3. The CRT had lower value of CC.

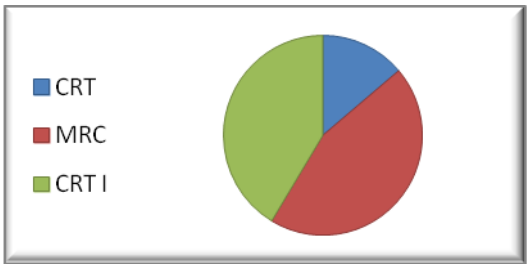

Fig. 3. Cyclomatic complexity of decoding algorithms.

Table 5 shows the times in milliseconds of encoding and decoding algorithms with varying number of (*h*+*r*) to a file have 74 characters. The result is applied when the number of the number of byte used equals 2 and when it equals 5. When the number of byte =2, the number of records is larger than when PS encoding with 5 byte. As well as the time of encoding and decoding is increased at 2 byte. The decoding time includes the time of the decoding algorithm not whole decoding process. Figure 4

shows the difference between the time of encoding and decoding process.

TABLE V. TIMES OF ENCODING AND DECODING ALGORITHMS

records	h+r	Time of Encoding	Time of CRT	Time of MRC	Time of CRT I
37	2+1	3307.1776	0.1586	0.0979	0.1105
37	4+2	4099.4608	1.1305	1.6863	0.3784
37	6+3	4282.9582	2.689	1.6513	0.8263
37	9+6	6216.5772	3.8943	4.8485	2.2793
15	2+1	2829.9167	0.3998	0.1241	0.0998
15	4+2	2984.1987	1.9807	0.635	0.383
15	6+3	3331.1054	1.7227	1.3872	0.8865
15	9+6	3833.5385	3.4132	3.1174	2.7628

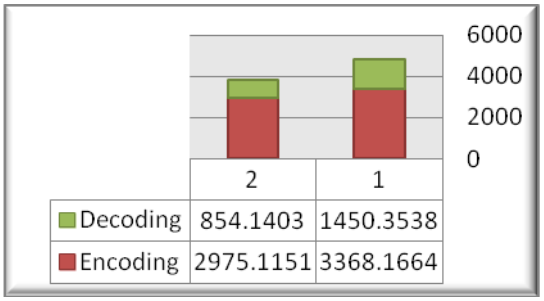

Fig. 4. Time of Encoding & Decoding Process.

C. *The Performance of the Authentication and Integrity Algorithms*

In this sub section, the performance of functions that ensure the security and privacy of the system are evaluated on the hand of time and complexity. This evaluation is based on the two tables of keys. The performance of the following operations is explained in Figures 5 & 6 below, and the time is calculated in millisecond;

- Signing and verifying of the residues
- Encrypting and decrypting of storage locations
- Creating FD
- Signing, encrypting, verifying and decrypting FD

VI. CONCLUSION

The DSS that is applied in proposed system can be considered the best way to store the critical data because it avoids single point of failure because it distribute data among a set of trusted servers. It also provides most security requirements such as the dependability that is achieved when the PS uses the redundancy where the RS can reconstruct the data up to $d \leq r$ residue erasures and $\frac{r-d}{2}$ corrupted residues. Confidentiality and authenticity are ensured in the system at two levels: One by using RRNS technique where in order to recover the patient information on the RS, the original keys (moduli) must be known to allow of RS to decrypt the data, and second by using Public key cryptography algorithms that ensure the authentication and non-repudiation servicers to encrypted the messages that are exchanged between servers. Integrity is achieved be using digital signature algorithm. The results of system prototype implementation have shown that whenever the value of moduli was big prime number, the code efficiency become large and the size of the sent message will be decreased. Results also shown that the CRT I is more efficient than the others decoding algorithm to decode RRNS. In order to avoid vulnerability to single point of attack when an attacker can have an access to all data if he/she could hack one RS, more access control techniques have to be applied to RSs (as end systems). This can be a point of further research.

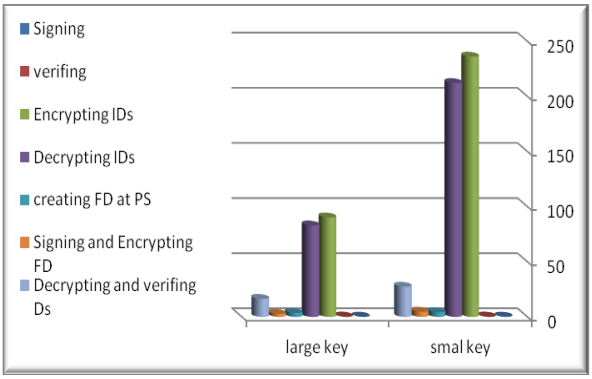

Fig. 5. Time of Authentication and Integrity Ams.

Fig. 6.

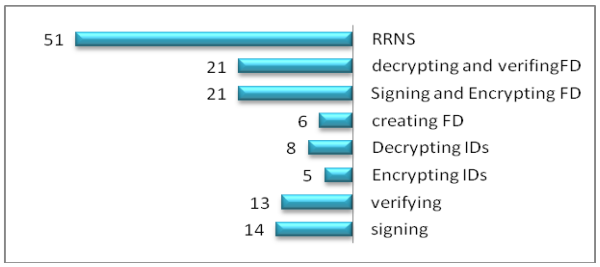

Fig. 7. Cyclomatic complexity of system.

REFERENCES

[1] MountainView ITSM, "Centralized vs. Distributed Computing", White Paper Discussion, October 29, 2007.

[2] Chessa and P. Maestrini, "Dependable and Secure Data Storage and Retrieval in Mobile, Wireless Networks", *International Conference on Dependable Systems and Networks*, 2003.

[3] Oliver Schneider, "Trust Aware Social Networking: A Distributed Storage System based on Social Trust and Geographical Proximity", Diplom Thesis, FU Berlin, Berlin, Germany, January 22, 2009.

[4] Salim El Rouayheb, Kannan Ramchandran," Fractional Repetition Codes for Repair in Distributed Storage Systems," *arXiv:1010.2551v1 [cs.IT]*, 2010

[5] Sameer Pawar, Salim El Rouayheb, Kannan Ramchandran," Securing Dynamic Distributed Storage Systems against Eavesdropping and Adversarial Attacks", *IEEE Transactions on Information Theory*, pp. 6734-6753, 2011.

[6] Alexandros G. Dimakis, P. Brighten Godfrey, Martin J. Wainwright ,Kannan Ramchandran, "The Benefits of Network

Coding for Peer-to-Peer Storage Systems", *Third Workshop on Network Coding, Theory, and Applications*, January 2007.

[7] Tanakorn Chareonvisal, "Implementing Distributed Storage System by Network Coding in Presence of Link Failure", Master Thesis, School of Electrical Engineering Kungliga Tekniska Högskolan Stockholm, Sweden, September 2012.

[8] Frederique Oggier, Anwitaman Datta" Self-repairing Homomorphic Codes for Distributed Storage Systems", *Infocom 2011, The 30th IEEE International Conference on Computer Communications*, July 2010.

[9] Rudi Ball, James Grant, Jonathan So, Victoria Spurrett, and Rogério de Lemos, "Dependable and Secure Distributed Storage System for Ad Hoc Networks", *Lecture Notes in Computer Science*, Volume 4686, pp. 142-152, Springer-Verlag, Berlin Heidelberg, 2007.

[10] Qian Wang, Kui Ren, Wenjing Lou, and Yanchao Zhang, "Dependable and Secure Sensor Data Storage with Dynamic Integrity Assurance", *IEEE INFOCOM*, 2009.

[11] Y. S. Han, S. Omiwade, and R. Zheng, "Progressive Data Retrieval for Distributed Networked Storage", *IEEE Conference on Parallel and Distributed Systems*, 2012.

[12] Lluis Pamies-Juarez, Frédérique Oggier, and Anwitaman Datta, "Decentralized Erasure Coding for Efficient Data Archival in Distributed Storage Systems", *Proceedings of the 14th International Conference on Distributed Computing and Networking (ICDCN)*, 2013.

[13] William Stallings, *Cryptography and Network Security Principles and Practices*, Fifth Edition, Prentice-Hall, 2011.

[14] Avik Sengupta, "Redundant Residue Number System Based Space-Time Block Codes", Thesis, Kansas State University, Manhattan, Kansas, 2012.

[15] K. Amusa and E. Nwoye, "Novel Algorithm For Decoding Redundant Residue Number Systems (RRNS) Codes", *IJRRAS*, July 2012.

[16] Narendran Narayanaswamy, "Optimization of New Chinese Remainder Theorems Using Special Moduli Sets", Thesis, Louisiana State University, December 2010.