ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)
Vol. 2, Issue 3 (July - Sept. 2014)
International Journal of Advanced Research in Computer Science & Technology (IJARCST 2014)

Towards High Security and Fault Tolerant Dispersed Storage System with Optimized Information Dispersal Algorithm

[I]Hrishikesh Lahkar, [II]Manjunath C R
[I,II]Jain University, School of Engineering and Technology, Kanakapura Road, Bangalore, India

Abstract
A Robust Data Storage System is one of the classes of fault tolerant systems. It is built around protocols that enable retrieval of data where only a subset of servers needs to be contacted. It enables efficient sharing and retrieval of data and the ability to scale to a number of servers as needed. A Robust Data Storage System basically comprises two main considerations. Firstly, integrity and consistency of data and secondly, security.

Keywords
IDA, CRS, RS, Dispersed Storage, AES256, SHA2.

I. Introduction

Today information dispersal algorithm is being widely used. Data loss is the main concern in storage system and to prevent this multiple disks are used. Storage companies such as Cleversafe, Data Domain, Allmydata, Network Appliance and Panasas are using information dispersal algorithm for their product. Companies such as IBM, Microsoft and Hewlett Packard are also doing active research on information dispersal for storage systems.In dispersed storage system, k + m disks are used, where data and coding information are represented by k and m. Codeword is calculated from the data in encoding operation, and decoding operation recover the data from the available disks after one or more disk failures. Storage systems uses information dispersal algorithm, which ensure that at least k disks should survive the failures, the data can always be recovered.Security is provide by dispersal system without the use of encryption keys. Shamir's technique [1] and Rabin's information dispersal is a non-systematic erasure codes [2] is a (k, n) scheme. A client is required to have at least k block of data out of the n blocks to reconstruct the file. It is not possible for the client to reconstruct the file with less than k blocks. Several of these systems use these techniques to store the n data blocks at a different site, and assuming that the intruder will never able to authenticate himself to k of them does not use the encryption technique which required to store the encryption key securely, without the use of encryption strategies the storage system may be vulnerable to attacks.This paper focuses on improving the security and performance of encoding and decoding of data which will be dispersed across multiple servers. It does so by optimizing the information dispersal algorithm and combining it with all or nothing transform (AONT) [3]. In this paper we demonstrate how to increase the security and the performance of dispersed storage system. We compare the encoding performance and the decoding performance of our technique with traditional technique and show that our scheme is significantly improves upon the traditional scheme. Hence, we believe that our approach has great potential in storage systems to have wide impact.

II. Related Work

A. Information Dispersal Algorithm

An information dispersal algorithm is a technique to slice a file into n pieces in such a way that the file can be recovered from some subsets of slices. There are many application for Information dispersal algorithms such as secure and reliable information storage, fault-tolerant and efficient transmission of information in dispersed storage systems. Matrix-Vector product of (k, n) is a threshold scheme, illustrated in Figure 1. The data to be dispersed is broken into w bits length word. A generator matrix G is calculated from the data, which has n rows and k columns. The matrix G is multiplied by a k element vector D to generate an n-element vector codeword C. the codeword generated will be stored on a different storage node.

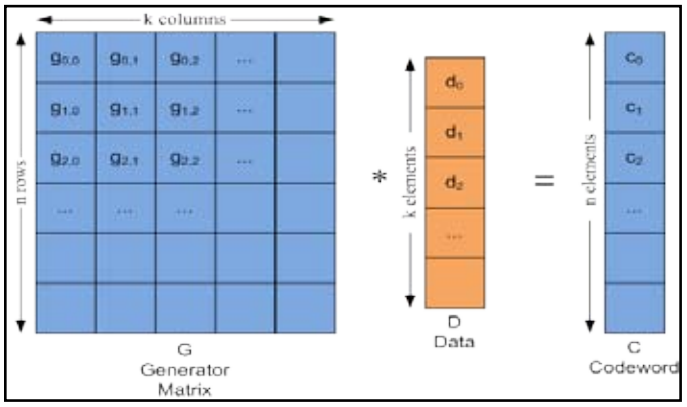

Fig.1: A basic matrix vector product.

The generator matrix for the data is created so that all combinations of k rows produce an invertible matrices which will gives us a method to recover D from k elements of the codeword. Codeword element is calculated from each row of G. We construct a new k×k matrix A from Generator Matrix G which is from k surviving elements. Next we convert matrix A to inverted matrix and multiply A−1 by the remaining elements to produce D. The Generator matrix G guarantees that A is invertible. Galois Field arithmetic [4], GF (2^w), is used to process the data where bitwise exclusive-or and multiplication is implemented. So dispersal is actually a part of well-known Reed-Solomon codes [5, 6].

B. Shamir's Secret Sharing Algorithm

One of the earliest algorithm for dispersal is Shamir's secret sharing algorithm [7] where data containing w bits are encoded in d_0.The matrix used to calculate generator matrix G for Shamir's algorithm is Vandermonde matrix [8]. In Vandermonde matrix $g_{i,j} = i^j$ and $n \leq 2^w$. In Shamir's algorithm, storage requirements is nb bytes and encoding operation for multiplication and XOR require O(knb). Shamir's algorithm provides a very strong security for its data.

C. Rabin's Information Dispersal Algorithm

Rabin's algorithms [2] improves the storage efficiency and performance whereas security provided is very less. D which

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

contains the data and message earlier now contains a word of data. So if we use Rabin's scheme, it require nb/k bytes of storage. Which will definitely improves the storage efficiency and the encoding performance. Since security guarantee is less, Rabin come up with a solution to encrypt the codeword with external encryption key but that does not solve our problem.

D. Reed Solomon (RS) Codes

To implement Reed Solomon Code [5, 6] we are considering a storage system which is composed of n disks array, each disk is of the same size. Each of n disks contains data and coding information. We can say that n=k+m where k hold the data and m hold the coding information often called as parity and is calculated from the data. Data disks are represented as $D_0, \ldots, D_{k-1}$ and the parity disks as $C_0, \ldots, C_{m-1}$. Since we are mainly interested in Maximum Distance Separable code [5] which is used to reconstruct the original data if m disk out of n disk fails. Encoding process partitions disk in to fix size strips. Each data disk contain one strip which is used to encode each parity strip. Stripe is the collection of k+m strips.In Reed Solomon Code, Strip is a w bit word where w should satisfy the condition $n \leq 2^w+1$ and $w < \{8,16,32,64\}$. In RS code the value of w can be chosen by the user if $n \leq 2^w+1$. Galois Filed arithmetic (GF (2^w)) is used to operate on each word w and the value of w is a number between 0-2^w-1. Since most system has fewer than 256 disk and to perform best, w=8 is largely used for implementation.Galois Filed arithmetic performs different addition, multiplication and division operation on each word so that the system perform to its best. The encoding and decoding operation of RS Code is a simple linear algebra. In RS code first we have to construct the Generator or Dispersal matrix G. Vandermonde matrix is used to construct the Generator matrix G. To create the codeword we multiply the Vandermonde matrix by k dataword. Now the codeword will be combination of k data and m code word. In Figure 2 a basic RS Code encoding is illustrated.

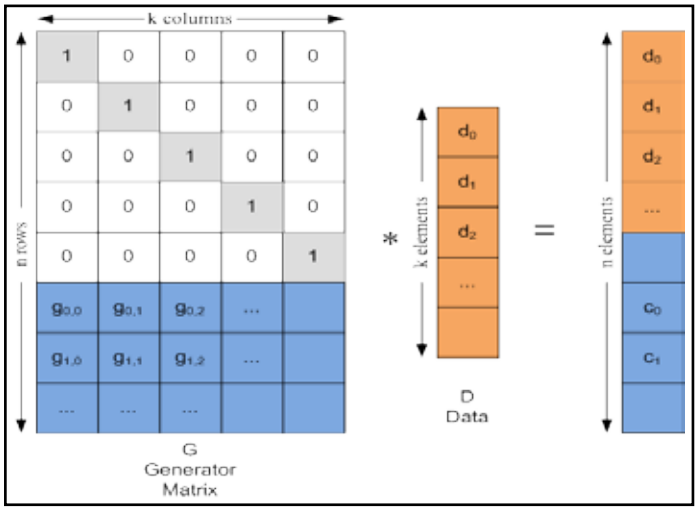

Fig.2: Reed Solomon Encoding

When k disk out of n disk fails, we are going to take the transpose of the Generator matrix (G^T) which will be inverted and multiplied by the surviving k disk to reconstruct the original data. RS code perform different addition and multiplication operation in (GF (2^w)) and addition is equivalent to exclusive-or and the multiplication operation is more complex, hence RS code are considered as more expensive.

III. Proposed Method

In this paper, Rabin's scheme is further modified to achieve improved computational performance, security and integrity by combining the All-Or-Nothing Transform with Optimized Cauchy Reed-Solomon code. We used a modified version of Rivest's All-or-nothing Transform as a preprocessing operation over the data. This operation we can say that is a (s+1, s+1) threshold scheme and data is composed of s words of size wA. None of the original words could be decoded because data is encoded into s+1 different words, unless all encoded words (s+1) are available. The key use in this technique is encoded with the data. By employing the AONT technique we achieve numerous benefits such as no external keys are required, performance is better than traditional scheme and storage requirement is very less.In modification of Rivest's All or Nothing Transform we show how the data is first encoded before sending it to the information dispersal algorithm and also how the data will be decoded after reconstructing the data by the information dispersal algorithm. In Figure 3 and Figure 4 illustrate how the data is sliced and reconstructed from the subset of slices.

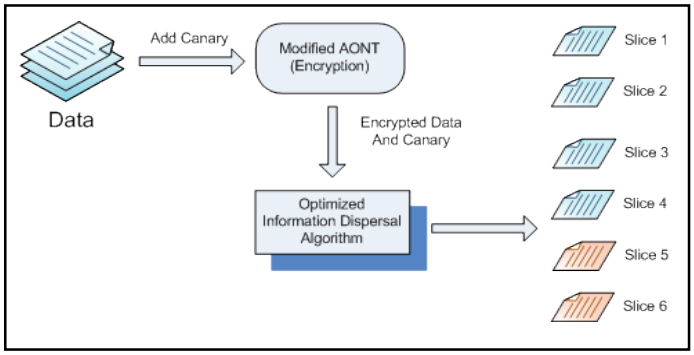

Fig. 3: Dispersal of AONT package using CRS

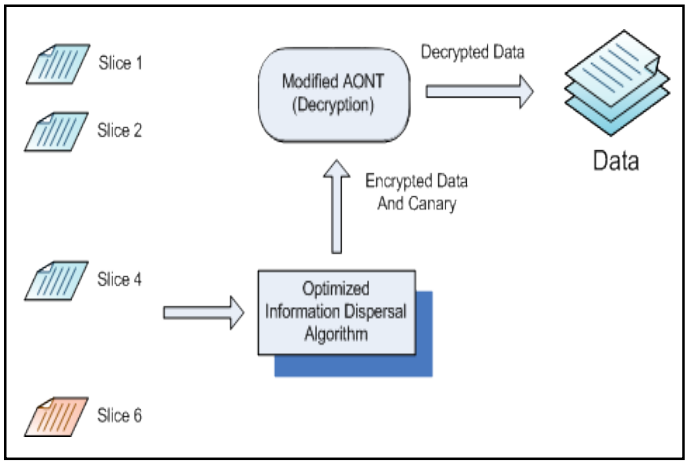

Fig. 4: AONT is recovered from Surviving Slices

To perform the All or Nothing Transform, the data is first composed of s words $d_0,\ldots\ldots d_{s-1}$, each word is of W_A bits in length. Each codeword c_i is calculated using a random key K.

$$c_i = d_i \oplus (K,i+1)$$

Where we used a encryption algorithm such as AES256 [9], a key-based algorithm. Codeword, c_k, will be calculated from the function of K and a hash value from the other codewords so it will be not possible for an attacker to guess K from any word or data.

Along with the data we add an extra data ds called canary [10]. The canary has a fixed and known value so that we can check the integrity of data during decoding operation. Next we calculate the hash value h of s+1 codeword using SHA-256 [11] hash algorithm along with it we generate c0,….cs as described above.
Our proposed work is depict by several diagrams along with AONT and Cauchy Reed-Solomon coding. In Figure 3, data is first passed to the AONT package where a canary is added to the data, and a random key is used to encrypt the data and canary. Now from this encrypted data a hash value is computed and a difference is calculated by combining the hash value and the random key by bitwise exclusive-or. The AONT package is formed by appended the difference to the encrypted data. The result of the AONT package is then use as an input to the Information Dispersal Algorithm as show in the Figure 4.

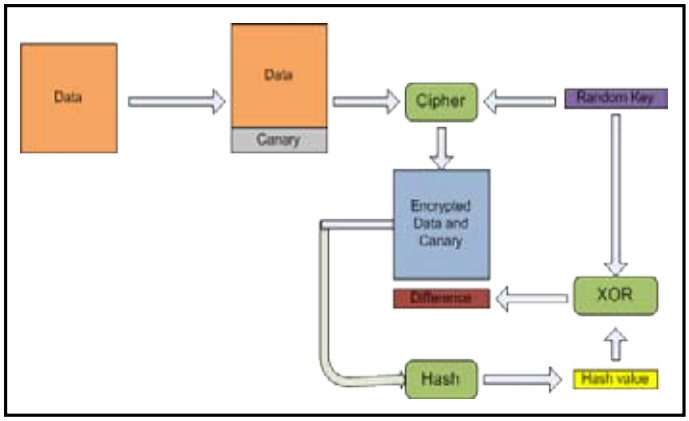

Fig. 3: Encoding of AONT package

Our second modification is to implement a variant of RS Code. Since RS code are expensive due to the use of Vandermonde matrix, we will replace Vandermonde matrix with Cauchy matrix [12] to from Cauchy Reed Solomon [13] (CRS) Code. Modification of RS code is perform in two ways. First Cauchy matrix will be used to generate the Generator matrix instead of Vandermonde matrix. Secondly the complex multiplication operation of RS code will be replaced by XOR operation.

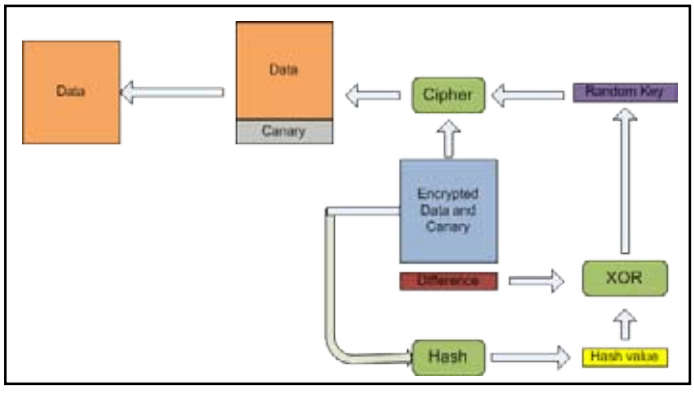

Fig. 4: Decoding of AONT package

In Reed Solomon Code single words are used for computation but CRS uses entire strip or data packet instead of word where each strip is represented as w packet. So now only XOR operation are performed for coding operation and complex multiplication are eliminated. We perform the XOR operation on all data packets to construct the code packet. Each row of Transpose Generator matrix GT has coding packet which contain 1 bit of data.

Figure 5 illustrate the matrix vector product of Cauchy Reed Solomon Code.

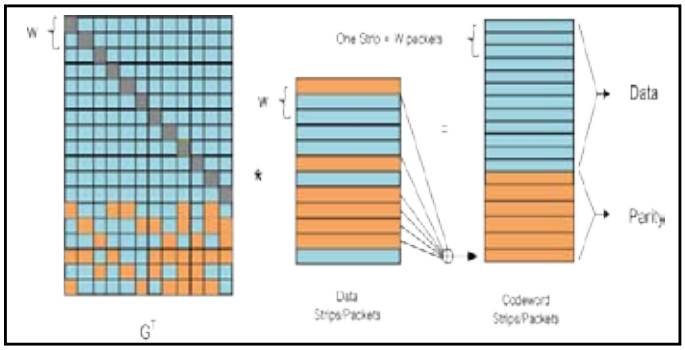

Fig. 5: CRS example for k = 4 and m = 2

Here the packet size are multiple of the systems word size which will make XOR more efficient. Also since w is not related to word size hence w times packet size gives us the strip size and we can take any value of w instead of {8,16,32,64} as long as $n \leq 2^w$. Decoding operation in Cauchy Reed Solomon is same as RS Code. Rows of the G^T which are correspond to those data packets that are found in failed disk are deleted. The matrix G^T is then inverted and multiplied with the data packets of the surviving disk to reconstruct the original data. The IDA computes n−k coding slices by splitting the input into k slices and store them in separate locations. The data slices are retrieved from these location. In CRS we can specify the threshold number of slices and without the threshold slices it is not possible to recover the AONT packages. After recovering the AONT package, next step is to reverse the AONT operation. In reversing operation we have to first calculate the hash value (h) of the data which is encrypted using AES256 algorithm. The last block of our AONT package contain k⊕ h which gives us the value of h. By performing exclusive-or operation on the last block we can calculate the (k⊕ h⊕ h) which will give us the random key k since h⊕h equals zero. Next we have to use this key to decrypt the encrypted data and to check the corruption in data we use the canary.

IV. Security Evaluation

Our model for security evaluation is one where servers belong to different domains. Servers may fail due to power failure or water damage which are nonsecurity related events, or the security of the system may be compromised; for example an attacker can steal data form the servers. All the techniques which we mentioned has a good level of security. Without having all k slices it is not possible to decode the data. However, if an attacker has less than k-1 slices and has some idea of what data he is looking, then those technique differ greatly. So we will see how the different algorithm behaves in security evaluation.
Shamir has great security called information theoretic security where it is not possible for attackers to get any information from less than k. In Shamir's algorithm d0 has 2^w potential values that can generate k-1 slices of size w. To determine the actual value of slices one need the k^{th} slice.
In Rabin's algorithm, missing slices can be easily guessed if the attacker knows the generator matrix and the recognizable pattern of the data. Attacker can get the k recognizable word from the k-1 slices using the 2w possibilities for each words.
In our proposed method to decode any of the data, one should have all of the encrypted data to decode it because to discover k we need all the data. However attacker can easily verify the

ISSN : 2347 - 8446 (Online)
ISSN : 2347 - 9817 (Print)

predetermined value of D if he has K and one slice, which is same as Rabin. But in our method attacker will find it difficult to figure out the value of K, suppose the attacker has the first slice where D's first encoded data is present that means d_0+E(K,1). To discover K, enumeration is the only way, so to discover its real value 2^{w}_{A} values must be tested which is not possible because it will require trillion computers to test trillion keys per second and it will take more than thousand years.

V. Performance Evaluation

For evaluating the performance of our proposed system, we considered the performance of encoding and decoding of our method and compare it with the existing method. When evaluating the encoding operation we took a large data file partitioned in to n pieces each of which contain k+m data/coding. Each of the n pieces are stored in different disk and calculate the encoding performance. Due to limited amount of memory of most computer and experimenting on large file requires two fix size buffer, Data Buffer and Coding Buffer. We then partitioned the Data and Coding Buffer into k and m blocks. The encoder reads the data, of size of the Data Buffer, from the large file and then perform the encoding operation and stored it in the Coding Buffer. Next it takes the content of Data and Coding Buffer and store it to k+m files. This process is repeated until all the file is encoded and record the encoding and the total time.

A. Encoding Performance

To test the performance of our proposed method we encode a file for w<=32. We select the best packet which we consider here is [12, 4] configuration shown in Figure 6. The result for [12, 4] is displayed in Figure with comparison between RS and CRS codes. From the figure we can conclude that CRS has the best performance compared to the RS. Encoding speed of CRS is 260MB/sec as compared to 50MB/sec of Reed Solomon because generator matrix is not optimized in Reed Solomon code and hence XOR operation increases which affects the encoding speed.

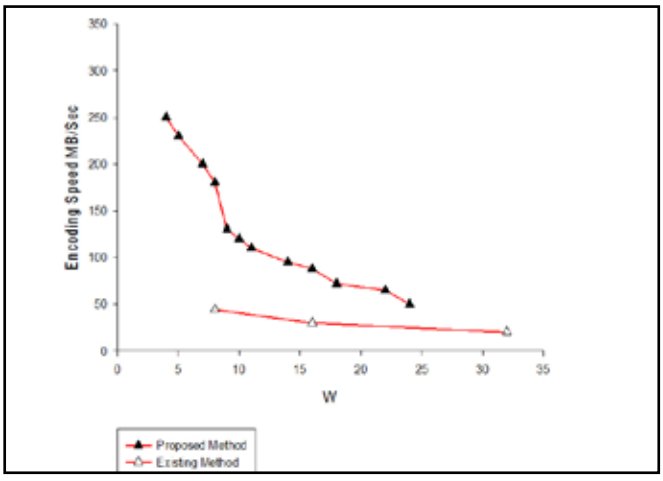

Fig. 6: Encoding Performance for [12, 4].

B. Decoding Performance

Decoding performance tested by converting the encoding program to perform decoding. Here m random data drives are chosen by the decoder. It sets the buffer value of those encoded drives and perform the decoding operation. Data drives are decoded because the decoding is the one of the hardest in data drive. In Figure 7 we shown the performance of [12, 4] for decoding operation. The performance of RS decoder identical to RS encoder because encoding process is optimized and uses standard RS decoding. In decoding also CRS performs much better than RS code as shown in Figure. The RS code decode at a speed of 50Mb/sec whereas CRS perform at a speed of 166Mb/sec.

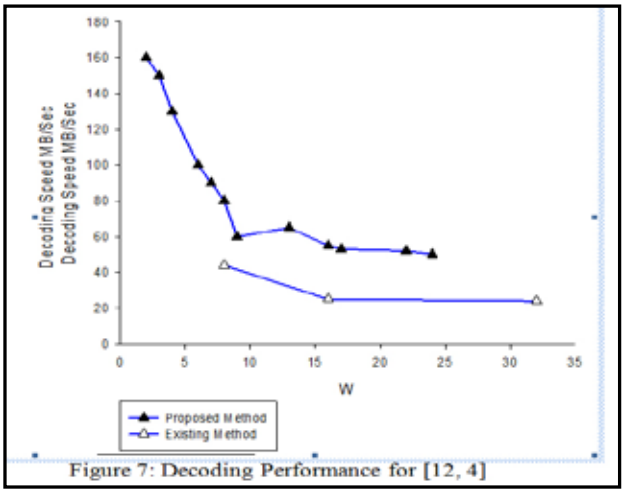

Fig. 7: Decoding Performance for [12, 4]

VI. Conclusion

Dispersed storage system are widely used for reliability, scalability and availability. Security in dispersed storage system is provided by k, n threshold scheme along with the secure storage of encryption key.

In this paper we described a new dispersal algorithm where AONT is combined with the Cauchy Reed Solomon codes to achieve a high level of security. We also compared the performance of Cauchy Reed Solomon codes with Reed Solomon codes. The performance of CRS coding is much better than RS coding because we can chose w as small as possible. During encoding and decoding dense matrix should not be used for generator matrix. While this is very useful but more work should be done. First, to improve the result, multiple machine should be used to test our method. Second, In CRS code, since multiple XOR operation are performed, those operation can be scheduled in different possible ways to yield improvements in performance.

References

[1] *SHAMIR, A. How to share a secret. Communications of the ACM 22, 11 (November 1979), 612–613.*

[2] *RABIN, M. O. Efficient dispersal of information for security, load balancing, and fault tolerance.Journalof the Association for Computing Machinery 36, 2 (April 1989), 335–348.*

[3] *RIVEST, R. All-or-nothing encryption and the package transform. In 4th International Workshop on Fast Software Encryption (1997), pp. 210–218.*

[4] *RIZZO, L. Erasure codes based on Vandermonde matrices. Gzipped tar file posted at http://planete- bcast.inrialpes.fr/ rubrique. php3?id rubrique=10,1998.*

[5] *MACWILLIAMS, F. J., AND SLOANE, N. J. A. The Theory of Error-Correcting Codes, Part I. North- Holland publishing Company, Amsterdam, New York, Oxford, 1977.*

[6] *REED, I. S., AND SOLOMON, G. Polynomial codes over certain finite fields. Journal of the Society for Industrial and Applied Mathematics 8 (1960), 300–304.*

[7] *SHAMIR, A. How to share a secret. Communications of the ACM 22, 11 (November 1979), 612–613.*

[8] *SHAMIR, A. How to share a secret. Communications of the*

ACM 22, 11 (November 1979), 612–613.

[9] DAEMEN, J., AND RIJMEN, V. The Design of Rijndael, AES—The Advanced Encryption Standard. Springer-Verlag, New York, 2002.

[10] AYCOCK, J. Computer Viruses and Malware (Advances in Information Security). Springer-Verlag, New York, 2006.

[11] WANZHONG, HONGPENG. Design and optimized implementation of the SHA-2 hash algorithms. ASIC, 2007. ASICON '07.

[12] LUBY, M. Code for Cauchy Reed-Solomon coding. Uuencoded tar file: http://www.icsi.berkeley.edu/luby/cauchy.tar.uu, 1997.

[13] PLANK, J. S., AND XU, L. Optimizing Cauchy Reed-Solomon codes for fault-tolerant networkstorageapplications. In NCA-06: 5th IEEE International Symposium on Network Computing Applications (Cambridge, MA, July 2006).