Secure Regenerating Code

Jian Li Tongtong Li Jian Ren
Department of ECE, Michigan State University, East Lansing, MI 48824-1226.
Email: {lijian6, tongli, renjian}@msu.edu

***Abstract*—Distributed storage plays a crucial role in the current cloud computing framework. After the theoretical bound for distributed storage was derived by the pioneer work of the regenerating code, Reed-Solomon code based regenerating codes, including the minimum storage regeneration (MSR) code and the minimum bandwidth regeneration (MBR) code, were developed. However, in the hostile network with passive eavesdroppers and active attackers, the data confidentiality and storage capacity of the network can be significantly affected. In this paper, we propose a secure MSR code that can combat against the passive eavesdroppers and active attackers in the network. We also provide theoretical analyses showing that our code can provide better security with less computational cost and bandwidth overhead.**

***Index Terms*—regenerating code, Reed-Solomon code, error-correction, passive eavesdropper, active attacker.**

I. Introduction

The main idea of distributed storage is that instead of storing the entire file in one server, we can split the file into n components and store them in n storage nodes. The original file can be recovered when we collect k file components. Employing an (n, k) Reed-Solomon code (RS code) is a straightforward and popular approach for distributed storage, where the data collector can reconstruct the file by connecting to any k storage nodes. However, when repairing or regenerating the contents of a failed node, the whole file has to be recovered first in the RS code approach, which is a waste of bandwidth. To deal with this issue, the concept of regenerating code was introduced in [1]. The bandwidth needed for repairing a failed node could be much less than the size of the whole file. In fact the regenerating code achieves an optimal tradeoff between bandwidth and storage within the minimum storage regeneration (MSR) and the minimum bandwidth regeneration (MBR) points.

The regenerating code can be divided into functional regeneration and exact regeneration. In the functional regeneration [2]–[4], the replacement node regenerates a new component that can functionally replace the failed component instead of being the same as the original stored component. In the exact regeneration [5]–[7], the replacement node regenerates the exact symbols of a failed node. [7] presented optimal exact constructions of MBR code and MSR code under product-matrix framework.

However, none of these schemes above can achieve data confidentiality and they will all fail in both regeneration and reconstruction under active attackers. In [8], the authors proposed to add CRC codes in the regenerating code to check the integrity of the data in hostile networks. Unfortunately, the CRC checks can be easily manipulated by the malicious nodes. In [9], the authors analyzed the error resilience of the regenerating code in the network with errors and erasures. They provided the theoretical error correction capability. But their scheme is unable to determine whether the errors in the network are successfully corrected. In [10] the authors proposed the universally secure regenerating code to achieve information theoretic data confidentiality. But the extra computational cost and bandwidth have to be considered for this code.

In this paper, we propose a secure MSR code that can protect the data against both passive eavesdroppers and active attackers. Theoretical analyses show that our code can provide better security with less computational cost and bandwidth overhead. The rest of this paper is organized as follows: In Section II, the preliminary of this paper is presented. In Section III, we propose the secure MSR code schemes. We conduct security and performance analyses in Section IV. The paper is concluded in Section V.

II. Preliminary and Assumptions

A. Regenerating Code

Regenerating code introduced in [1] is a linear code over finite filed GF_q with a set of parameters $\{n, k, d, \alpha, \beta, B\}$. A file of size B is stored in n storage nodes, each of which stores α symbols. A replacement node can regenerate the contents of a failed node by downloading β symbols from each of d randomly selected storage nodes. So the total bandwidth needed to regenerate a failed node is $\gamma = d\beta$. The data collector (DC) can reconstruct the whole file by downloading α symbols from each of $k \leq d$ randomly selected storage nodes. In [1], the following theoretical bound was derived:

$$B \leq \sum_{i=0}^{k-1} \min\{\alpha, (d-i)\beta\}. \quad (1)$$

From equation (1), a tradeoff between the regeneration bandwidth γ and the storage requirement α was derived. γ and α cannot be decreased at the same time. There are two special cases: minimum storage regeneration (MSR) point in which the storage parameter α is minimized;

$$(\alpha_{MSR}, \gamma_{MSR}) = \left(\frac{B}{k}, \frac{Bd}{k(d-k+1)}\right), \quad (2)$$

and minimum bandwidth regeneration (MBR) point in which the bandwidth γ is minimized:

$$(\alpha_{MBR}, \gamma_{MBR}) = \left(\frac{2Bd}{2kd-k^2+k}, \frac{2Bd}{2kd-k^2+k}\right). \quad (3)$$

978-1-4799-3512-3/14/$31.00 ©2014 IEEE

B. Adversarial Model

In this paper, we assume there are (l, m) eavesdroppers that can access the data stored in l nodes and monitor the data communication in m out of the l nodes. We also assume that some network nodes can be corrupted by the active attackers and send out bogus symbols to disrupt the data regeneration and reconstruction.

III. Secure MSR Code Design

In this section, we will analyze secure MSR code on the MSR point with $d = 2k - 2$. The code based on the MSR point with $d > 2k - 2$ can be derived the same way through truncating operations.

A. Encoding of Secure MSR Code

According to equation (2), we have $\alpha = d/2, \beta = 1, B = (\alpha+1)\alpha$. If the actual file size is larger than B, we can divide the file into blocks of size B. We can then encode, regenerate, and reconstruct each block separately. So we will only consider a single block of the size B in this paper.

Suppose the file is $\mathbf{m} = [m_1, m_2, \ldots, m_B] \in GF_q^B$. To achieve the data confidentiality in distributed storage, the secure server will encode $\mathbf{m}$ using a code sequence generated from linear feedback shift register (LFSR) with the feedback polynomial [11] $y = x^{63} + x^{62} + 1$ with a randomly select initial secret state, before the MSR encoding. In practical implementation, we can choose LFSR's with polynomials of different orders to satisfy different security requirements. After generating B symbols $\mathbf{r} = [r_1, r_2, \ldots, r_B] \in GF_q^B$, the secure server will add $\mathbf{m}$ and $\mathbf{r}$ together to get the symbol vector $\mathbf{s}$ for MSR encoding:

$$\mathbf{s} = \mathbf{m} + \mathbf{r} = [s_1, s_2, \ldots, s_B]. \tag{4}$$

The next step is to encode the encrypted symbol vector $\mathbf{s}$ based on the product-matrix code framework [7]. We will arrange $\mathbf{s}$ into two symmetric matrices T_1, T_2:

$$T_1 = \begin{bmatrix} s_1 & s_2 & \cdots & s_\alpha \\ s_2 & s_{\alpha+1} & \cdots & s_{2\alpha-1} \\ \vdots & \vdots & \ddots & \vdots \\ s_\alpha & s_{2\alpha-1} & \cdots & s_{B/2} \end{bmatrix}, \tag{5}$$

$$T_2 = \begin{bmatrix} s_{B/2+1} & s_{B/2+2} & \cdots & s_{B/2+\alpha} \\ s_{B/2+2} & s_{B/2+\alpha+1} & \cdots & s_{B/2+2\alpha-1} \\ \vdots & \vdots & \ddots & \vdots \\ s_{B/2+\alpha} & s_{B/2+2\alpha-1} & \cdots & s_B \end{bmatrix}. \tag{6}$$

The codeword C is defined as

$$C = [\Phi \quad \Lambda\Phi] \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \Psi T, \tag{7}$$

where

$$\Phi = \begin{bmatrix} 1 & 1 & 1 & \ldots & 1 \\ 1 & \phi & \phi^2 & \ldots & \phi^{\alpha-1} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 1 & \phi^{n-1} & (\phi^{n-1})^2 & \ldots & (\phi^{n-1})^{\alpha-1} \end{bmatrix} \tag{8}$$

is a Vandermonde matrix and $\Lambda = \text{diag}[\lambda_1, \lambda_2, \cdots, \lambda_\alpha]$ such that the $\lambda_i \neq \lambda_j$ for $1 \leq i, j \leq \alpha, i \neq j$, where $\lambda_i \in GF_q$ for $1 \leq i \leq \alpha$, ϕ is a primitive element in GF_q, and any d rows of Ψ are linear independent. Then each row $\mathbf{c}_i$, $0 \leq i \leq n-1$, of the codeword matrix C will be stored in storage node i, in which the encoding vector ν_i is the i^{th} row of Ψ.

B. Regeneration and Reconstruction in the Normal Network

Since the encoding operation of the secure MSR code does not affect the product-matrix framework, a replacement node can regenerate the contents of a failed node through the regeneration process described in [7].

For data reconstruction, DC can first reconstruct the encrypted symbol vector $\mathbf{s}$ following the reconstruction process presented in [7]. Then DC can generate the secret symbol vector $\mathbf{r}$ of size B, which is the same as the vector in the secure MSR code encoding, by using the same feedback polynomial and initial state. At last DC can reconstruct the original file $\mathbf{m}$ through

$$\mathbf{m} = \mathbf{s} - \mathbf{r}. \tag{9}$$

C. Combating Against Passive Eavesdroppers

For an (l, m) eavesdropper that can access the data stored in l nodes and monitor the data communication in m out of the l nodes, the secure MSR code can ensure that it is computationally infeasible for the eavesdropper to get access to any fraction of the original file.

In fact, if $l \geq k$, although the eavesdropper can reconstruct the encrypted symbol vector $\mathbf{s}$, the probability that it can recover the original file is $p = 1/(2^{63} - 1) \approx 10^{-19}$, which is the strength of cracking a block cipher with key size equal to 63 bits. This is because the initial state of the shift register was randomly selected by the security server and kept as a secret. In fact, if needed, the key size can be easily increased with minimum computational overhead.

If $l < k$, the eavesdropper cannot get the entire encrypted symbol vector $\mathbf{s}$ through reconstruction. If the encoding vector ν_i in a eavesdropped node i has only one non-zero element, the eavesdropper can get the encrypted symbols in some row of the matrix T. For example, suppose node 0 is eavesdropped and $\nu_0 = [1, 0, \ldots, 0]$, the eavesdropper can get the symbols of the first row of T. However, it is computational infeasible for the eavesdropper to recover the original symbols. If the encoding vector ν_i in a eavesdropped node i has at least two non-zero elements, the eavesdropper cannot even get the encrypted symbols in any row of the matrix T.

There are several advantages for applying the pseudo random vector generated by LFSR in the secure MSR code to combat against passive eavesdroppers for distributed storage. First, the data collector can be implemented in the secure server. So there is no need to transmit the initial state of the feedback register as the secret key among different nodes. The eavesdropper will not get the secret key unless it can break in the secure server. Second, since there is only one server that manages and uses the secret keys, the key management is very simple. For n files stored in the network, the security server

only needs to save n secret keys instead of $n(n-1)/2$ keys for the normal communication network consisting of n nodes.

D. Combating Against Active Adversaries

In the network where there are active adversaries, since the malicious node can modify the data, regeneration of the failed nodes or reconstruction of the original file using the schemes proposed in [7] may fail without being detected. In this section we will propose regeneration and reconstruction algorithms for the secure MSR code in this type of hostile network. For convenience, we define $\mathbf{V}_{i,j} = \left[\nu_i^T, \nu_{i+1}^T \cdots, \nu_j^T\right]^T$, where ν_t, $i \leq t \leq j$, is the t^{th} row of Ψ.

1) Regeneration: Suppose node z fails, there are two modes for regeneration of the secure MSR code. The replacement node z' will use the detection regeneration mode first to detect the erroneous regeneration if no error has been found before. If any erroneous regeneration is detected, z' will try to correct the errors using the recovery regeneration mode.

a) Detection Regeneration Mode: In the detection regeneration mode, z' will send regeneration requests to $d+1$ helper nodes instead of d nodes. Upon receiving the regeneration request, helper node i will calculate and send out the help symbol $p_i = \mathbf{c}_i \mu_z^T$, where μ_z is the z^{th} row of Φ. z' will perform Algorithm 1 to regenerate the contents of the failed node z and detect erroneous regenerations.

Algorithm 1 (Detection mode)**.** *z' regenerates symbols of the failed node z*

Suppose $p_i' = p_i + e_i$ is the response from the i^{th} helper node. If p_i has been modified by the malicious node i, we have $e_i \in GF_q \backslash \{0\}$. To detect whether $e_i = 0$, z' will calculate symbols and compare the results from two sets of helper nodes. (Without loss of generality, we assume $0 \leq i \leq d$.)

Step 1: Solve the equation $\mathbf{V}_{0,d-1} \cdot \mathbf{x}_1 = \mathbf{p}_1'$, where $\mathbf{p}_1' = [p_0', p_1', \cdots, p_{d-1}']^T$ are symbols collected from node 0 to node $d-1$.

Step 2: Solve the equation $\mathbf{V}_{1,d} \cdot \mathbf{x}_2 = \mathbf{p}_2'$, where $\mathbf{p}_2' = [p_1', p_2', \cdots, p_d']^T$ are symbols collected from node 1 to node d.

Step 3: Compare $\mathbf{x}_1$ with $\mathbf{x}_2$. If they are the same, compute $\mathbf{c}_z = \mu_z T_1 + \lambda_z \mu_z T_2$ as described in [7]. Otherwise, errors are detected in the help symbols. Switch to recovery regeneration mode.

We have the following theorem from [12] to guarantee the detection probability of Algorithm 1.

Theorem 1. *The probability for the bogus symbols received from the malicious nodes to be detected using Algorithm 1 is at least $1 - 1/q$.*

b) Recovery Regeneration Mode: In the recovery regeneration mode, z' will send requests to the other $n-1$ nodes. Helper node i will still send out the help symbol $p_i = \mathbf{c}_i \mu_z^T$. z' will perform Algorithm 2 to correct the errors and regenerate the contents of z.

Algorithm 2 (Recovery mode)**.** *z' regenerates symbols of the failed node z*

We can successfully regenerate the symbols in node z when the errors in the received help symbols ${p_i}'$ from $n-1$ helper nodes can be corrected. Without loss of generality, we assume $0 \leq i \leq n-2$.

Step 1: Decode $\mathbf{p}'$ to $\mathbf{p}_{cw}$, where $\mathbf{p}' = [p_0', p_1', \cdots, p_{n-2}']^T$ can be viewed as an MDS code with parameters $(n-1, d, n-d)$ since $\mathbf{V}_{0,n-2} \cdot \mathbf{x} = \mathbf{p}'$. If the i^{th} positions of $\mathbf{p}_{cw}$ and $\mathbf{p}'$ are different, mark node i as corrupted.

Step 2: Solve $\mathbf{V}_{0,n-2} \cdot \mathbf{x} = \mathbf{p}_{cw}$ and compute $\mathbf{c}_z = \mu_z T_1 + \lambda_z \mu_z T_2$ as described in [7].

2) Reconstruction: Similar to the failed node regeneration, there are two modes for reconstruction of the secure MSR code. DC will first use the detection reconstruction mode to retrieve the original file and detect the erroneous reconstructions. If an incorrect reconstruction is detected, DC will try to correct the errors using the recovery reconstruction mode.

a) Detection Reconstruction Mode: In the detection reconstruction mode, DC will send reconstruction requests to $k+1$ storage nodes instead of k nodes. Upon receiving the request, node i will send out the symbol vector $\mathbf{c}_i$. DC will use Algorithm 3 to perform the detection reconstruction.

Algorithm 3 (Detection mode)**.** *DC reconstructs the original file*

Suppose $\mathbf{c}_i' = \mathbf{c}_i + \mathbf{e}_i$ is the response from the i^{th} storage node. If $\mathbf{c}_i$ has been modified by the malicious node i, we have $\mathbf{e}_i \in (GF_q)^\alpha \backslash \{\mathbf{0}\}$. To detect whether there are errors, we will reconstruct the original file from two sets of storage nodes then compare the results. (Without loss of generality, we assume $0 \leq i \leq k$.)

Step 1: Solve T_1, T_2 using the method similar to [7], where $\Psi_{DC1} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = \mathbf{V}_{0,\alpha_l} \begin{bmatrix} T_1 \\ T_2 \end{bmatrix} = {R_1}'$ and ${R_1}' = [{\mathbf{c}_0'}^T, {\mathbf{c}_1'}^T, \cdots, {\mathbf{c}_\alpha'}^T]^T$ are the symbols collected from node 0 to node $k-1=\alpha$.

Step 2: Similar to **Step 1**, solve $\widetilde{T}_1, \widetilde{T}_2$ where $\Psi_{DC2} \begin{bmatrix} \widetilde{T}_1 \\ \widetilde{T}_2 \end{bmatrix} = {R_2}'$ and ${R_2}' = [{\mathbf{c}_0'}^T, \cdots, {\mathbf{c}_{\alpha-1}'}^T, {\mathbf{c}_{\alpha+1}'}^T]^T$ are the symbols collected from node 0 to node $k = \alpha + 1$ except node α.

Step 3: Compare $[T_1, T_2]$ with $[\widetilde{T}_1, \widetilde{T}_2]$. If they are the same, reconstruct the encrypted symbol vector $\mathbf{s}$ from T_1 and T_2. Then DC can reconstruct the original file $\mathbf{m}$ following Section III-B. Otherwise, errors are detected in the received symbols. Switch to recovery reconstruction mode.

We have the following theorem from [12] to guarantee the detection probability of Algorithm 3.

Theorem 2. *The probability for the bogus symbols received from the malicious nodes to be detected using Algorithm 3 is at least $1 - (1/q)^{2(\alpha-1)}$.*

b) Recovery Reconstruction Mode: In the recovery reconstruction mode, DC will send requests to all the n storage nodes. Node i will still send out the symbol vector $\mathbf{c}_i$.

Let $R' = [{\mathbf{c}'_0}^T, {\mathbf{c}'_1}^T, \cdots, {\mathbf{c}'_{n-1}}^T]^T$, we have

$$\Psi_{DC} \begin{bmatrix} T'_1 \\ T'_2 \end{bmatrix} = [\Phi_{DC} \quad \Lambda_{DC}\Phi_{DC}] \begin{bmatrix} T'_1 \\ T'_2 \end{bmatrix} = \mathbf{V}_{0,n-1} \begin{bmatrix} T'_1 \\ T'_2 \end{bmatrix} = R',$$

$$\Phi_{DC}T'_1\Phi_{DC}^T + \Delta_{DC}\Phi_{DC}T'_2\Phi_{DC}^T = R'\Phi_{DC}^T.$$

Let $C = \Phi_{DC}T'_1\Phi_{DC}^T$, $D = \Phi_{DC}T'_2\Phi_{DC}^T$, and $\widehat{R}' = R'\Phi_{DC}^T$, then

$$C + \Delta_{DC}D = \widehat{R}'.$$

Since C, D are both symmetric, we can solve the non-diagonal elements of C, D as follows:

$$\begin{cases} C_{i,j} + \lambda_i \cdot D_{i,j} = \widehat{R}'_{i,j} \\ C_{i,j} + \lambda_j \cdot D_{i,j} = \widehat{R}'_{j,i}. \end{cases}$$

Because matrices C and D have the same structure, here we only focus on C (corresponding to T'_1). It is straightforward to see that if node i is malicious and there are errors in the i^{th} row of R', there will be errors in the i^{th} row of $\widehat{R}'$. Furthermore, there will be errors in the i^{th} row and i^{th} column of C. Define $T'_1\Phi_{DC}^T = \widehat{T}'_1$, we have $\Phi_{DC}\widehat{T}'_1 = C$. Here we can view each column of C as an $(n-1, \alpha, n-\alpha)$ MDS code because Φ_{DC} is a Vandermonde matrix. The length of the code is $n-1$ since the diagonal elements of C is unknown. Suppose node j is a legitimate node. If the number of the malicious nodes τ satisfies: $2\tau + 1 \leq n - \alpha$, then the j^{th} column of C can be recovered and the error locations (corresponding to the corrupted nodes) can be pinpointed. The non-diagonal elements of C can be recovered. So DC can reconstructs T_1 using the method similar to [7]. For T_2, the recovering process is similar.

At last DC can reconstruct the original file $\mathbf{m}$ from T_1 and T_2 following Section III-B.

IV. Security Analysis and Performance Evaluation

In this section, security analyze and performance evaluation of the proposed secure MSR will be performed.

A. Security Analysis

1) Passive Eavesdroppers: In Section III-C, we have demonstrated that the secure MSR code is safe against passive eavesdroppers regardless of the value of (l, m). For the universally secure MSR code in [10] (We will refer the code to universal MSR in this paper) pre-designed for the (l, m) eavesdropper, if the number of the eavesdropped nodes is larger than l or the number of the nodes of which the communication is monitored is larger than m, the whole storage system is no longer secure.

2) Active Attackers: The security against active attackers can be analyzed through the error correction capability of the MSR codes. For the regeneration, both the secure MSR code and the universal MSR code can correct errors from up to $\lfloor n-d \rfloor/2$ nodes influenced by active attackers. But the secure MSR code has more flexible error correction capabilities. The universal MSR code can correct up to τ errors by downloading symbols from $d + 2\tau$ nodes. However, the number of errors may vary in the symbols sent by the helper nodes. When there is no error or the number of errors is far less than τ, downloading symbols from extra nodes will be a waste of bandwidth. When the number of errors is larger than τ, the decoding process will fail without being detected. In this case, the symbols stored in the replacement node will be erroneous. If this erroneous node becomes a helper node later, the errors will propagate to other nodes. The secure MSR code can detect the erroneous decodings using Algorithm 1. If no error is detected, the replacement node only needs to download symbols from one more node than that in the normal network, while the extra cost for the universal MSR code is 2τ. If errors are detected in the symbols received from the helper nodes, the secure MSR code can correct the errors using Algorithm 2.

For the reconstruction, the evaluation result is similar to the regeneration. Both the secure MSR code and the universal MSR code can correct errors from up to $\lfloor n-k+1 \rfloor/2$ nodes. However, the secure MSR code is more flexible. The universal MSR code can correct at most τ errors with support from 2τ additional helper nodes. The secure MSR code only requires symbols from one additional node to detect the erroneous reconstructions using Algorithm 3. The errors can then be corrected using the corresponding processes in Section III-D2.

B. Computational Cost

1) Encoding: The secure MSR code encoding consists of the generation of encrypted symbol vector $\mathbf{s}$ and one matrix multiplication between matrices of size $n \times d$ and $d \times \alpha$. There are B shift operations, B exclusive-or operations and B addition operations for the generation of $\mathbf{s}$. For the matrix multiplication, there are $nd\alpha$ addition operations and $nd\alpha$ multiplication operations. So the total computational cost is $3B + 2nd\alpha$ and the average cost per symbol is

$$COST_{enc-1} = (3B + 2nd\alpha)/B. \tag{10}$$

For the universal MSR code encoding, there is one matrix multiplication between matrices of size $n \times d$ and $d \times \alpha$. So the cost is $2nd\alpha$. Since only $B^* = (k-l)(\alpha-m)$ symbols can be stored securely against the (l, m) passive eavesdroppers according to [10], the average cost per symbol is

$$COST_{enc-2} = \frac{2nd\alpha}{(k-l)(\alpha-m)}. \tag{11}$$

In Table I we list the ratio

$$R_{enc} = COST_{enc-1}/COST_{enc-2} \tag{12}$$

to compare the average computational cost between the secure MSR code and the universal MSR code for the parameter $\alpha = 10$, $\beta = 1$, $B = 110$, $k = 11$, $n = 25$, $m = 0$ and $0 \leq l \leq 9$. We use different l, the number of nodes from which the eavesdropper can access the data, to evaluate the performance of the two MSR codes. From the table we can see that the average computational cost of the secure MSR code is becoming lower compared to the universal MSR code as there are more eavesdropped nodes in the network.

TABLE I
COMPARISON OF THE SECURE MSR AND THE UNIVERSAL MSR CODE

l	0	1	2	3	4	5	6	7	8	9
average computational cost ratio (%)										
R_{enc}	103	94	85	75	66	56	47	38	28	19
R_{reg}	100	91	82	73	64	55	46	36	27	18
R_{rec}	106	97	87	77	68	58	48	39	29	19
bandwidth overhead of the universal MSR code (%)										
OD	0	9	18	27	36	45	55	64	73	82

This is because when the number of the eavesdropped nodes increases, utilizing the universal MSR code means we have to perform operations for more random numbers inserted into the data matrix to prevent any possible information leakage according to [10]. Then the average computational cost for the real data becomes higher.

2) Regeneration: The regeneration operations for the secure MSR code and the universal MSR code in the normal network are the same. So the ratio of the average computational cost between the secure MSR code and the universal MSR code is

$$R_{reg} = \frac{B^*}{B} = \frac{(k-l)(\alpha-m)}{\alpha(\alpha+1)}. \tag{13}$$

In Table I we also list R_{reg} for the same sets of parameters in Section IV-B1. From the table we can also conclude that the secure MSR code has lower average computational cost for node regeneration due to the same reason.

3) Reconstruction: The reconstruction of the secure MSR code consists of the reconstruction operations described in [7] and one operation to remove $\mathbf{r}$. The computational cost is $(14/3)\alpha^3 + (13/2)\alpha^2 + (9/2)\alpha$ for the reconstruction operations described in [7], which requires one matrix multiplication between matrices of size $k \times \alpha$ and $\alpha \times k$, $k(k-1)/2$ 2-variable linear equation sets, two $\alpha \times \alpha$ matrix inversions and two matrix multiplications between $\alpha \times \alpha$ matrices. So the average cost per symbol is

$$COST_{rec-1} = (3B + (14/3)\alpha^3 + (13/2)\alpha^2 + (9/2)\alpha)/B.$$

Similar to the analysis of Section IV-B1, the average computational cost for the universal MSR code is

$$COST_{rec-2} = \frac{(14/3)\alpha^3 + (13/2)\alpha^2 + (9/2)\alpha}{(k-l)(\alpha-m)}.$$

In Table I we list

$$R_{rec} = COST_{rec-1}/COST_{rec-2}$$

for the same sets of parameters in Section IV-B1. Similarly, the results also show that the secure MSR code has lower average computational cost.

C. Bandwidth Overhead

There is no additional bandwidth overhead for the regeneration and reconstruction of the secure MSR code since the amount of data we want to store in the matrix T in equation (7) satisfies the MSR bound in [1]. For the universal MSR code, $B - B^* = B - (k-l)(\alpha-m)$ random numbers have to be fit in the matrix T. So on average in the packets sent out by the storage nodes, the ratio of the fake information to the whole information is

$$OD = (B - B^*)/B. \tag{14}$$

This is the additional bandwidth overhead for the regeneration and reconstruction of the universal MSR code. Table I lists this overhead for the same sets of parameters in Section IV-B1. And for the similar reason, the overhead becomes higher as l increases.

V. CONCLUSION

In this paper, we develop a secure minimum storage regeneration (MSR) code against the passive eavesdroppers and active attackers. We propose the encoding, regeneration and reconstruction algorithms for the secure MSR code. Our theoretical analyses show that the secure MSR code is more efficient in defending against the passive eavesdroppers and more flexible in defending against the active attackers than the universal MSR code. Our analyses also demonstrate that the secure MSR code has lower computational cost and bandwidth overhead.

REFERENCES

[1] A. Dimakis, P. Godfrey, Y. Wu, M. Wainwright, and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Transactions on Information Theory*, vol. 56, pp. 4539 – 4551, 2010.

[2] Y. Wu, A. G. Dimakis, and K. Ramchandran, "Deterministic regenerating codes for distributed storage," in *45th Annu. Allerton Conf. Control, Computing, and Communication*, 2007.

[3] A. Duminuco and E. Biersack, "A practical study of regenerating codes for peer-to-peer backup systems," in *Distributed Computing Systems, 2009. ICDCS '09. 29th IEEE International Conference on*, pp. 376 – 384, June 2009.

[4] K. Shum, "Cooperative regenerating codes for distributed storage systems," in *2011 IEEE International Conference on Communications (ICC)*, pp. 1–5, 2011.

[5] Y. Wu and A. G. Dimakis, "Reducing repair traffic for erasure coding-based storage via interference alignment," in *IEEE International Symposium on Information Theory, 2009. ISIT 2009.*, pp. 2276–2280, 2009.

[6] N. Shah, K. Rashmi, P. Kumar, and K. Ramchandran, "Interference alignment in regenerating codes for distributed storage: Necessity and code constructions," *IEEE Transactions on Information Theory*, vol. 58, pp. 2134 – 2158, 2012.

[7] K. Rashmi, N. Shah, and P. Kumar, "Optimal exact-regenerating codes for distributed storage at the msr and mbr points via a product-matrix construction," *IEEE Transactions on Information Theory*, vol. 57, pp. 5227–5239, 2011.

[8] Y. Han, R. Zheng, and W. H. Mow, "Exact regenerating codes for byzantine fault tolerance in distributed storage," in *Proceedings IEEE INFOCOM*, pp. 2498 – 2506, 2012.

[9] K. Rashmi, N. Shah, K. Ramchandran, and P. Kumar, "Regenerating codes for errors and erasures in distributed storage," in *International Symposium on Information Theory (ISIT) 2012*, pp. 1202–1206, 2012.

[10] N. B. Shah, K. V. Rashmi, K. Ramchandran, and P. V. Kumar, "Privacy-preserving and secure distributed storage codes," *http://www.eecs.berkeley.edu/~nihar/publications/privacy_security.pdf/*.

[11] M. George and P. Alfke, "Linear feedback shift registers in virtex devices," *http://www.xilinx.com/support/documentation/application_notes/xapp210.pdf,*.

[12] J. Li, T. Li, and J. Ren, "Beyond the mds bound in distributed cloud storage," in *IEEE INFOCOM 2014, accepted, to appear*.