On Secure Distributed Data Storage Under Repair Dynamics

Sameer Pawar
Salim El Rouayheb
Kannan Ramchandran

Electrical Engineering and Computer Sciences
University of California at Berkeley

Technical Report No. UCB/EECS-2010-18
http://www.eecs.berkeley.edu/Pubs/TechRpts/2010/EECS-2010-18.html

February 17, 2010

On Secure Distributed Data Storage Under Repair Dynamics

Sameer Pawar
University of California, Berkeley
Email: spawar@eecs.berkeley.edu

Salim El Rouayheb
University of California, Berkeley
Email: salim@eecs.berkeley.edu

Kannan Ramchandran
University of California, Berkeley
Email: Kannanr@eecs.berkeley.edu

***Abstract*— We address the problem of securing a dynamic distributed data storage system against a passive eavesdropper that can observe a fixed number of storage nodes. Distributed data storage system experiences node failures over time due to various reasons. These failed nodes are repaired in order to maintain the availability of data with certain fixed reliability. If an eavesdropper accesses a node while it is being added to the system to repair it from a failure, it will have access to all the data communicated to that node, which can potentially compromise the entire data stored in the system. We are interested in determining the secrecy capacity of such dynamic systems, i.e., the maximum amount of data that can be made reliably available to a legitimate user in the face of node failures and repairs in presence of eavesdropper without revealing any information to it about the data. We use the information flow graph to model these systems as multicast networks with compromised nodes. We provide a general upper bound on the secrecy capacity and show that this bound is tight in the *bandwidth limited regime* which is of significant importance for practical systems such as Internet-based peer-to-peer distributed storage systems.**

I. Introduction

Data storage devices have evolved significantly since the days of punched cards. Nevertheless, storage devices, such as hard disks or flash drives, are still bound to fail after long periods of usage, risking the loss of valuable data. To solve this problem and to increase the reliability of the stored data, multiple storage nodes can be networked together to redundantly store the data, thus forming a distributed data storage system. Applications of such systems are innumerable and include large data centers and peer to peer storage systems such as OceanStore [1], Total Recall [2] and DHash++ [3] that use a large number of nodes spread widely across the Internet to store files.

Codes for protecting data from erasures have been well studied in classical channel coding theory, and can be used here to increase the reliability of distributed storage systems. Fig. 1 illustrates an example where maximal distance separable (MDS) codes are used to store a file $\mathcal{F}$ of 4 symbols $(a_1, a_2, b_1, b_2) \in \mathbb{F}_5^4$ distributively on 4 different nodes, $v_1, \dots, v_4$, each of capacity 2 units. The MDS code implemented here ensures that any user, also called data collector, connecting to any 2 storage nodes can obtain the whole file $\mathcal{F}$. However, what distinguishes the scenario here from the erasure channel counterpart is that when a storage node fails, it needs to be repaired or replaced by a new node in order to maintain a desired level of system reliability. A straightforward repair mechanism would be to add a new replacement node of capacity 2, and make it act as a data collector by connecting, for this example, to 2 active, i.e., non-failed, nodes, so that it can download the whole file, construct the lost part of the data and store it. Another repair scheme that consumes less bandwidth is depicted in Fig. 1 where node v_1 fails and is replaced by node v_5. By making node v_5 connect to 3 nodes instead of 2, it is possible to decrease the total repair bandwidth from 4 to 3 units. Note that v_5 does not need to store the exact data that was on v_1; the only required property is that the data stored on all the active nodes v_2, v_3, v_4 and v_5 form an MDS code.

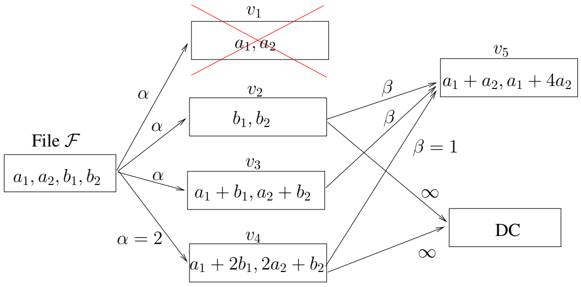

Fig. 1. An example of a distributed data storage system under repair. Node v_1 fails and is replaced by a new node v_5 that downloads $(b_1 + b_2), (a_1 + a_2 + b_1 + b_2)$ and $(a_1 + 4a_2 + 2b_1 + 2b_2)$ from v_2, v_3 and v_4 respectively to compute and store $(a_1 + a_2, a_1 + 4a_2)$. A data collector connecting to nodes v_2 and v_4 to recover the stored file is also depicted.

The above important observation was the basis of the original work of [4] where the authors showed that there exists a fundamental tradeoff between the amount of storage used at each node and the amount of bandwidth needed for repair. In other words, for a fixed repair bandwidth there is a minimal value beyond which the node storage capacity cannot be decreased, and vice versa. In this tradeoff, two optimal operational regimes were defined: *Minimum Storage Regeneration* (MSR) and *Minimum Bandwidth Regeneration* (MBR). When storing a file of size M, the first regime corresponds to the smallest possible storage space on each node which is $\frac{M}{k}$ which corresponds to using an MDS code. The second regime is characterized by an operating point that has minimum repair bandwidth that can be achieved by storing slightly more than $\frac{M}{k}$ at each node.

When a distributed data storage system is formed using nodes widely spread across the Internet, e.g. Internet based peer to peer system, individual nodes may not be secure and hence may fall victim to eavesdropping. This paper focuses on such scenarios where an eavesdropper can gain access to a certain number of storage nodes. If a node is compromised by an eavesdropper, its stored content as well as any incoming or outgoing messages to and from the compromised node are observed by the eavesdropper. The eavesdropper is assumed to be passive that is, has the ability to observe the data but not to modify it. The distributed storage system that is being compromised is always assumed to be dynamic with nodes continually failing and being repaired. Thus, the compromised nodes can belong to the original set of storage nodes that the system starts with, or even include some of the replacement nodes which might be observed while being repaired. Under this setting, it is interesting to see how much data can still be stored in the system while revealing no information about it to the eavesdropper, and what schemes can achieve this goal.

To answer this question, we follow the approach of [4] and model the distributed storage system as a multicast network that uses network coding. Under this model, the eavesdropper is an intruder that can access a fixed number of the network nodes of his or her choice. This eavesdropper model is natural for distributed storage systems and comes in contrast with the wiretapper model studied in the network coding literature [5]-[7] where the intruder can access network edges instead of nodes. Under this setting, we define the secrecy capacity of a given distributed storage system and make the following contributions. First, we give a general upper bound on the secrecy capacity as a function of the node storage capacity α and the repair bandwidth γ. Second, we demonstrate that this upper bound is achievable for an important regime that we call the *bandwidth limited regime* where the repair bandwidth of the system γ is limited to a fixed number Γ but no cap is set on the storage capacity α of the nodes.

This paper is organized as follows. In Section II we describe the system and security model, then we define the problem and give a summary of our results in Section III. In Section IV, we illustrate two special cases of distributed storage systems that are illuminative in understanding the general problem. In Section V, we derive an upper bound on the secrecy capacity of a distributed storage system. In Section VI we present a scheme that achieves this upper bound for the case of bandwidth limited regime, which is the operating regime of interest for most practical applications. We conclude in Section VII.

II. Model

A. Distributed storage system

A distributed storage system (DSS) is a dynamic network of storage nodes. These nodes include a source node that has an incompressible data file $\mathcal{F}$ of M symbols, or units, each belonging to a finite field $\mathbb{F}_q$. The source node is connected to n storage nodes $v_1, \dots, v_n$, each having a storage capacity of α units which may be utilized to save coded parts of the file $\mathcal{F}$. The storage nodes are individually unreliable and may fail over time. To guarantee a certain desired level of reliability, we assume that the DSS is required to always have n active, i.e., non-failed, storage nodes that are in service. Therefore, when a storage node fails, it is immediately replaced by a new node with same storage capacity α. The DSS should be designed in such a way as to allow any legitimate user, that we also call data collector, that connects to any k out of the n active storage nodes available at any given time, to be able to reconstruct the original file $\mathcal{F}$. We term this condition as the "*reconstruction property*" of distributed storage systems.

We assume that nodes fail one at a time, and we denote by v_{n+i} the new replacement node added to the system to repair the i-th failure. In this work we focus on only symmetrical repair: new node connects to some d nodes, $d \geq k$, chosen, possibly randomly, out of the remaining active $n-1$ nodes and downloads β units of data from each node i.e., $\gamma = d\beta$. The process of replenishing redundancy to maintain the reliability of a DSS is referred to as the *"regeneration"* or *"repair"* process. Note that a new replacement node may download more data than what it actually stores. Moreover, the stored data can possibly be different than the one that was stored on the failed node, as long as the "reconstruction property" of the DSS is retained. A distributed storage system $\mathcal{D}$ is thus characterized as $\mathcal{D}(n,k)$. For instance, the DSS depicted in Fig. 1 corresponds to $\mathcal{D}(4,2)$ which is operating at the MSR operation point $(\alpha, \gamma) = (2, 3)$.

B. Flow Graph Representation

We adopt the same representation structure as in [4] where a DSS is cast as an information flow graph $\mathcal{G}$. The graph $\mathcal{G}$ is a directed acyclic graph with capacity constrained edges that consists of three kinds of nodes: a single source node s, input storage nodes x_{in}^i and output storage nodes x_{out}^i and data collectors DC_j for $i, j \in \{1, 2, \dots\}$. The source node s holds an information source S which has a specific realization the file $\mathcal{F}$. Each storage node v_i in the DSS is represented by two nodes x_{in}^i and x_{out}^i. To account for the storage capacity of v_i, these two nodes are joined by a directed edge of capacity α (see Fig. 2).

The repair process that is initiated every time a failure occurs, causes the DSS, and consequently the flow graph, to be dynamic and evolve with time. At any given time, each node in the graph is either active or inactive depending on whether it has failed or not. The graph $\mathcal{G}$ starts with only the source node s, the nodes $x_{in}^1, \dots, x_{in}^n$ connected respectively to the nodes $x_{out}^1, \dots, x_{out}^n$. Initially, only the source node s is active and is connected to the storage input nodes $x_{in}^1, \dots, x_{in}^n$ by outgoing edges of infinite capacity. From this point onwards, the source node s becomes and remains inactive, and the n input and output storage nodes become active. When a node v_i fails in a DSS, the corresponding nodes x_{in}^i and x_{out}^i become inactive in $\mathcal{G}$. If a replacement node v_j joins the DSS in the process of repairing a failure and connects to d active nodes $v_{i_1}, \dots, v_{i_d}$, the corresponding nodes x_{in}^j and x_{out}^j with the edge (x_{in}^j, x_{out}^j) are added to the flow graph $\mathcal{G}$, and node x_{in}^j is connected to the nodes $x_{out}^{i_1}, \dots, x_{out}^{i_d}$ by incoming edges of capacity β each. A data collector is represented by a node connected to k active storage output nodes through infinite capacity links enabling it to reconstruct the file $\mathcal{F}$ by downloading all the data stored on these nodes. The graph $\mathcal{G}$ constitutes a multicast network with the data collectors as destinations. An underlying assumption here is that the flow graph corresponding to a distributed storage system depends on the sequence of failed nodes. As an example, we depict in Fig. 2 the flow graph corresponding to the DSS $\mathcal{D}(4,2)$ of the previous section (see Fig. 1) when node v_1 fails.

C. Eavesdropper Model

In addition to the model adopted in [4], we assume the presence of an intruder "Eve" that can access up to l, $l < k$, nodes of its choice among all the storage nodes, $v_1, v_2, \dots$, possibly at different time instances as the system evolves. In the flow graph model, Eve is an eavesdropper that can access a fixed number l of nodes chosen from the storage input nodes $x_{in}^1, x_{in}^2, \dots$. Notice that while a data collector

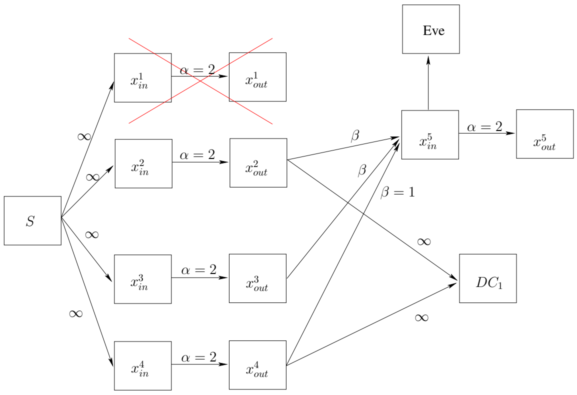

Fig. 2. The flow graph model of the DSS of Fig. 1 with $l = 1$. Eve accesses input node x_{in}^5 of the storage node v_5.

observes output storage nodes, i.e., the data stored on the nodes it connects to, Eve, has access to input storage nodes, and thus can observe, in addition to the stored data, all incoming messages to these nodes. We also assume that Eve has complete knowledge of the storage and repair scheme implemented in the DSS, thus, she can choose some of the l nodes among the initial n storage nodes, and, maybe, if it deems to her more profitable, wait till a certain failure occurs and then eavesdrops on the replacement node to observe its downloaded data. We assume that Eve is passive and can only observe the data without modifying it.

III. Problem Statement and Results

A. Secrecy Capacity

Under the presence of an eavesdropper, our objective is to maximize the utilization of the DSS for the data collectors by storing as much data as possible while guaranteeing perfect secrecy, i.e., making sure that no information is revealed to Eve. We start with few definitions. Let S be a vector taking arbitrary values in $\mathbb{F}_q^M$. S represents the incompressible data file of size M at the source node i.e., $H(S) = M$. Let $V_{in} := \{x_{in}^1, x_{in}^2, \dots\}$ and $V_{out} := \{x_{out}^1, x_{out}^2, \dots\}$ be the sets of input and output storage nodes respectively. For a storage node v_i, let D_i and C_i be the random variables representing its downloaded messages and stored content respectively. Thus, C_i represents the data that can be acquired by a data collector when connecting to node v_i, while D_i represents the total data revealed to Eve when it accesses node v_i. In general, the stored data C_i is a function of the downloaded data D_i.

Let V_{out}^a be the collection of all subsets of V_{out} of cardinality k consisting of nodes that are simultaneously active together at a certain instant in time. For any subset B of V_{out}, define $C_B := \{C_i : x_{out}^i \in B\}$. Similarly for any subset E of V_{in}, define $D_E := \{D_i : x_{in}^i \in E\}$. The reconstruction property can be written as

$$H(S|C_B) \quad = \quad 0 \quad \forall B \in V_{out}^a, \qquad (1)$$

and the perfect secrecy condition implies

$$H(S|D_E) \quad = \quad H(S), \forall E \subset V_{in} \text{ and } |E| \leq l. \qquad (2)$$

Given a DSS $\mathcal{D}(n,k)$ with l compromised nodes, its secrecy capacity, denoted by $C_s(\alpha,\gamma$, is then defined to be the maximum amount of data that can be stored in this system such that the reconstruction property and the perfect secrecy condition are simultaneously satisfied for all possible data collectors and eavesdroppers i.e.,

$$C_s(\alpha,\gamma) := \max_{\substack{H(S|C_B)=0 \quad \forall B \\ H(S|D_E)=H(S) \quad \forall E}} H(S) \qquad (3)$$

where $B \in V_{out}^a$, $E \subset V_{in}$ and $|E| \leq l$.

B. Results

The main objective of this paper is to determine the secrecy capacity of distributed storage systems. Under the flow graph model described above, the problem can be recast as finding the secrecy capacity of a special family of multicast networks implementing network coding where a certain collection of nodes is vulnerable to eavesdropping. Using this approach, we prove two main results. The first consists of an upper bound on the secrecy capacity of a DSS:

Theorem 3.1: [Upper Bound] For a distributed data storage system $\mathcal{D}(n,k)$ with $l < k$ compromised nodes, the secrecy capacity is upper bounded by

$$C_s(\alpha,\gamma) \leq \sum_{i=l}^{k-1} \min\{(d-i)\beta, \alpha\}. \qquad (4)$$

where d, β are such that $k \leq d \leq n-1, \gamma = d\beta$.

We also consider an important operational regime namely the *bandwidth limited regime*. In a bandwidth limited regime the repair bandwidth $\gamma = d\beta$ is capped to a fixed amount Γ (although one has a choice of d, β) while no constraint is imposed on the node storage capacity α. The secrecy capacity in this regime is defined as

$$C_s^{BL}(\Gamma) := \max_{\substack{\gamma \leq \Gamma \\ \alpha}} C_s(\alpha,\gamma)$$

For a fixed Γ the upper bound of Theorem 3.1 on the secrecy capacity can be shown to be maximized for the choice of $d = n-1$. In section VI, we demonstrate that this upper bound for $d = n-1$ can be achieved for a bandwidth limited regime. Thus establishing the following theorem,

Theorem 3.2: [Bandwidth Limited Regime] For a distributed data storage system $\mathcal{D}(n,k)$ with $l < k$ compromised nodes, the secrecy capacity for bandwidth limited regime is given by

$$C_s^{BL}(\Gamma) = \sum_{i=l}^{k-1}((n-1)-i)\frac{\Gamma}{n-1},$$

and is achieved with a storage capacity $\alpha^* = \Gamma$.

IV. Special Cases

Before we proceed to present the proofs of Theorems 3.1 and 3.2, we analyze two specific cases of distributed storage systems which help shed some light on the general problem.

A. Static Systems

A static version of the problem studied here corresponds to a DSS with ideal storage nodes that do not fail, and hence there is no need for any repair in the system. The flow graph of this system constitutes then a well-known multicast network studied in network coding theory called the combination network. Therefore, the static storage problem can be regarded as a special case of wiretap networks [5], [6], or equivalently, as the erasure-erasure wiretap-II channel studied in [10]. The secrecy capacity for such systems is $C_s(\alpha) = (k - l)\alpha$, and can be achieved using either nested MDS codes [10] or the coset codes of [9], [6].

Even though the above proposed solution is optimal for the static case, it can have a very poor security performance when applied directly to dynamic storage systems with failures. For instance one naive way for an MDS code to repair a lost coded symbol would be to download the whole file on the new replacement node and then generate the specific lost data. In this case if Eve accesses the new replacement node while it is downloading the whole file it will be able to reconstruct the original data. Hence, no secrecy scheme will be able to hide any part of the data from Eve and the secrecy rate for this scheme would be zero. However theorem 3.2 suggests that for some systems we can have a positive secrecy capacity.

This example highlights the new dimension that the repair process brings into the distributed storage picture. The dynamic nature of the DSS renders it intrinsically different from the static counterpart making the repair process a key factor that should be carefully designed in order not to jeopardize the whole stored data.

B. Systems Using Random Network Coding

Using the flow graph model, the authors of [4] showed that *random linear network codes* over a large finite field can achieve any point (α, γ) on the optimal storage-repair bandwidth tradeoff curve with a high probability. Consider an example of random linear network codes used in a compromised DSS $\mathcal{D}(4, 3)$ which stores $M = 6$ symbols and operates at the MBR regime (see Eq. (5)) corresponding to $d = 3, \beta = 1$ and $\alpha = 3$. In this case, each of the initial nodes $v_1, \ldots, v_4$ stores 3 independently generated random linear combinations of these $M = 6$ symbols. Assume now that node v_4 fails and is replaced by a new node v_5 that connects to v_1, v_2 and v_3 and downloads from each one of them $\beta = 1$ random linear combination of their stored data. Assume that after some time, node v_5 fails and is replaced by node v_6 in a similar fashion. Now if Eve accesses nodes v_5 and v_6 i.e., $(l = 2)$ while they were being repaired, it will observe 6 linear combinations of the original data symbols. If the field size is high enough (which is usually the case), then with high probability the information observed by Eve is independent, and she will be able to reconstruct the whole file.

The above analysis shows that secrecy rate achieved for this system by a random network coding is zero. But according to Theorem 3.2 which we will prove in section VI, secrecy capacity of the DSS $D(4, 3)$ operating at MBR point corresponding to $d = 3, \beta = 1$ and $\alpha = 3$ is equal to one unit when $l = 2$. While random network codes are very appealing for use in distributed storage systems due to their decentralized nature and low complexity, the above analysis shows that this may not always be the case when security is a desired property. This also is in contrast with the case of multicast networks where an intruder can access a fixed number of edges instead of nodes [5], wherein, random network coding performs as good as any deterministic secure code [7].

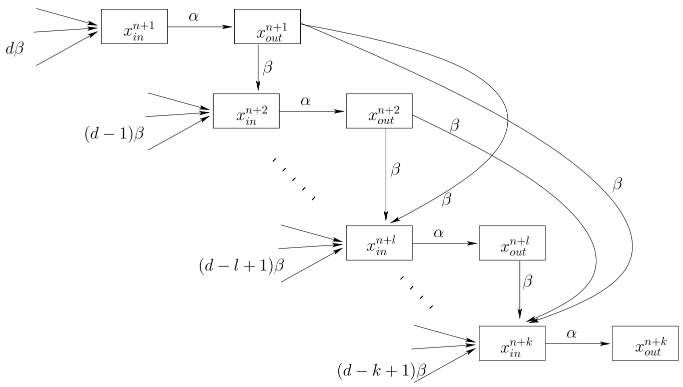

Fig. 3. Part of the flow graph corresponding to a DSS when nodes $v_1, \ldots, v_k$ fail successively and are repaired by nodes $v_{n+1}, \ldots, v_{n+k}$. Nodes $v_{n+1}, \ldots, v_{n+l}$ are compromised by Eve while they were being repaired.

V. Upper Bound on Secrecy Capacity

In this section we derive the upper bound of Theorem 3.1. Consider a DSS $\mathcal{D}(n, k)$ with $l < k$, operating at point (α, γ) with $d\beta = \gamma, k \leq d \leq n - 1$. Consider a specific cut in the flow graph $\mathcal{G}$ of a DSS. Assume that the nodes $v_1, v_2, \ldots, v_k$ have failed consecutively and were replaced during the repair process by the nodes $v_{n+1}, v_{n+2}, \ldots, v_{n+k}$ respectively as shown in Fig. 3.

Now suppose that Eve accesses nodes in $E = \{v_{n+1}, v_{n+2}, \ldots, v_{n+l}\}$ while they were being repaired, and consider a data collector connected to the nodes in $B = \{v_{n+1}, v_{n+2}, \ldots, v_{n+k}\}$. The reconstruction property implies $H(S|C_B) = 0$ by Eq. (1), and the perfect secrecy condition implies $H(S|D_E) = H(S)$ by Eq. (2). We can therefore write

$$\begin{aligned}
H(S) &= H(S|D_E) - H(S|C_B)\\
&\overset{(1)}{\leq} H(S|C_E) - H(S|C_B)\\
&\overset{(2)}{=} H(S|C_E) - H(S|C_E, C_{B\setminus E})\\
&= I(S, C_{B\setminus E}|C_E)\\
&\leq H(C_{B\setminus E}|C_E)\\
&= \sum_{i=l+1}^{k} H(C_{n+i}|C_{n+1}, \ldots, C_{n+i-1})\\
&\overset{(3)}{\leq} \sum_{i=l+1}^{k} \min\{(d - i + 1)\beta, \alpha\}
\end{aligned}$$

Inequality (1) follows from the fact that the stored data C_E is a function of the downloaded data D_E, (2) $C_{B\setminus E} := \{C_{n+l+1}, \ldots, C_{n+k}\}$, (3) follows from the fact that each node can store at most α units, and for each replacement node we have $H(C_i) \leq H(D_i) \leq d\beta$, also from the topology of the network (see Fig. 3) where each node x_{in}^{n+i} is connected to each of the nodes $x_{out}^{n+1}, \dots, x_{out}^{n+i-1}$ by an edge of capacity β. The upper bound of Theorem 3.1 then follows directly from the definition of Eq. (3).

VI. Bandwidth Limited Regime

In most distributed data storage systems, such as Internet-based peer-to-peer systems, storage is an inexpensive resource while inter-node communication is costly. Such systems usually have an upper limit Γ on the amount of repair bandwidth consumed γ, while can be optimized with no constraint on the storage α.

In this section we focus on such scenario and will prove Theorem 3.2. Let $d = n - 1$. As the examples studied in Section IV pointed out, the main difficulty of this problem is due to the dynamic nature of the network. We will demonstrate that in a bandwidth limited regime for $d = n-1$, with careful choice of the network code it is possible to transform the problem of secrecy over a dynamic DSS into a static problem of secrecy over point to point channel equivalent to the erasure-erasure wiretap channel-II [10]. Then we show that using nested MDS codes at the source one can achieve the secrecy capacity of the equivalent wiretap channel.

A. Exact Regeneration Codes

Let $d = n - 1, \beta = \Gamma/d$ and $\alpha = d\beta$ which corresponds to an MBR operating point. It was shown in [8] that exact regeneration codes, i.e., codes that allow any new node to store the same data as the failed one it is replacing, can be constructed for this regime. This result is essential to establish our proof. Next, we summarize this construction. For a DSS (n, k) operating at the MBR point if the file size is M then we have following relation between α, β, d, k, M [4]:

$$(\alpha, \beta) = (\frac{2Md}{2kd - k^2 + k}, \frac{2M}{2kd - k^2 + k}) \qquad (5)$$

We will show an achievable scheme for $\beta = 1$ which implies $M = kd - \frac{k(k-1)}{2}$ and $\alpha = d$. For any larger values of β required, the file can be split into chunks of size M, each of which can be separately encoded using the construction for $\beta = 1$.

Denote the source symbols of the file by the column vector $S = (s_1, s_2, \dots, s_M)^T$. Let $\theta = d(d+1)/2 = (n-1)n/2$. The construction of [8] involves using an (θ, M) MDS code that encodes the information vector S into θ coded symbols denoted by $Y = (y_1, \dots, y_\theta)$. The exact regeneration code can be easily described using an auxiliary complete graph over n vertices $u_1, \dots, u_n$ that consists of θ edges. Suppose the edges are indexed by the coded symbols $y_1, \dots, y_\theta$. The code then consists of storing on the node v_i the indexes of the edges adjacent to vertex u_i in the complete graph.

This complete graph endows the code with a special property that every coded symbol is stored on exactly two storage nodes and any pair of two nodes have exactly one edge and hence exactly one distinct coded symbol in common. This property ensures that any data collector can download exactly M coded symbols by connecting to k nodes and thus recover the data since code is (θ, M) MDS code. Moreover since $\alpha = d = n - 1$ any new replacement node can download one coded common symbol from each of the remaining $n - 1$ active nodes which guaranties the exact regeneration of the failed node.

B. Equivalence to a Wiretap Channel

Given the previous construction of exact regeneration codes, we can now explain the transformation of the dynamic storage system into a static point-to-point channel. Towards that end, we make the following observations. First, since the exact regeneration codes described here are operating at the MBR point, all the data communicated during the repair process is stored at the new replacement node without any further compression. Thus accessing a node during repair process (i.e., downloaded data) is equivalent to accessing it after the repair process i.e. stored data. Second, the exact regeneration codes by definition restore a failed node with the exact lost data. So, even though there are failures and repair, the data storage system looks exactly the same at any point of time.

By the property of the used exact code, the i^{th} node accessed by the data collector or Eve shares $i - 1$ symbols with the previously observed $(i-1)$ nodes and, as a result, will only reveal $(\alpha - (i-1)) = (d - (i-1))$ new symbols. An Eve accessing l distinct nodes will therefore observe some $\mu := \sum_{i=1}^{l}(d - (i-1))$ symbols out of θ encoded ones. Similarly, a data collector accessing k nodes will observe some $\nu := \sum_{i=1}^{k}(d - (i-1))$ symbols out of θ. Thus, the exact regeneration codes have transformed the problem of secrecy over a dynamic DSS with repair into a problem of secrecy over the equivalent point to point static erasure-erasure wiretap channel. In this equivalent channel source transmits θ symbols. A legitimate receiver observes the transmitted θ symbols over an erasure channel with $\theta - \nu$ erasures. The transmitted symbols are also observed by an eavesdropper/wiretapper through another erasure channel with $\theta - \mu$ erasures. This channel is similar to the erasure-erasure wiretap channel of type-II studied in [10] with slight difference. In the channel studied in [10] erasures can occur at any locations while in this equivalent channel erasures of only certain combinations can occur. Nevertheless we can only do worse by assuming that erasures are completely random with all possible combinations. Since we are showing achievability the above assumption is valid. From [10], we know that if the (θ, M) MDS code we used in constructing exact regeneration codes is an nested MDS code it can achieve the secrecy rate of $\nu - \mu$, i.e.,

$$\sum_{i=1}^{k}(d - (i-1)) - \sum_{i=1}^{l}(d - (i-1)) = \sum_{i=l}^{k-1}((n-1) - i)$$

This rate is achieved for every 1 unit of β. Thus, the total secrecy rate achieved for $\beta = \Gamma/(n-1)$ is,

$$\sum_{i=l}^{k-1}((n-1) - i)\frac{\Gamma}{n-1}$$

thus completing the proof of Theorem 3.2.

VII. Conclusion

In this paper we considered dynamic distributed data storage systems that are subject to eavesdropping. Our main objective was to determine the secrecy capacity of such systems, i.e., the maximum amount of data that these systems can store without revealing any information to the intruder. Modeling such systems as multicast networks with compromised nodes, we gave an upper bound on the secrecy capacity and showed that it can be achieved in the practically important bandwidth limited regime where the nodes have a sufficient storage capacity. Finding the general expression of the secrecy capacity of distributed storage systems, and more generally of multicast networks with a fixed number of compromised nodes, remains an open problem that we hope to address in future work.

REFERENCES

[1] S. Rhea, C. Wells, P. Eaton, D. Geels, B. Zhao, H. Weatherspoon, and J. Kubiatowicz, "Maintenance-free global data storage," *IEEE Internet Computing*, pp. 4049, September 2001.

[2] R. Bhagwan, K. Tati, Y.-C. Cheng, S. Savage, and G. M. Voelker, "Total recall: System support for automated availability management," in *NSDI*, 2004.

[3] F. Dabek, J. Li, E. Sit, J. Robertson, M. Kaashoek, and R. Morris, "Designing a dht for low latency and high throughput," 2004.

[4] A. Dimakis, P. Godfrey, Y. Wu, M. Wainright and K. Ramchandran, "Network coding for distributed storage systems," *IEEE Trans. Inform. Theory*, 2005.

[5] N. Cai and R. W. Yeung, "Secure network coding," in *IEEE Internat. Symp. Inform. Th. (ISIT02)*, Jun. 2002.

[6] S. E. Rouayheb and E. Soljanin, "On wiretap networks II," in *IEEE Internat. Symp. Inform. Th. (ISIT'08)*, 2007.

[7] D. Silva and F. R. Kschischang, "Security for wiretap networks via rank-metric codes," in *IEEE Internat. Symp. Inform. Th. (ISIT'08)*, 2008.

[8] Rashmi K.V, N. B. Shah, P. V. Kumar and K. Ramchandran,"Exact Regenerating Codes for Distributed Storage," *Submitted on Arxiv*.

[9] L. H. Ozarow and A. D. Wyner,"Wire-Tap Channel-II," *AT&T Bell lab tech. journal*.

[10] Arunkumar S, S. W. Mclaughlin, "MDS codes on erasure-erasure wiretap channel," *arXiv:0902.3286v1*.