Secure Clustered Distributed Storage Against Eavesdropping

Beongjun Choi, Student Member, IEEE, Jy-yong Sohn, Student Member, IEEE, Sung Whan Yoon, Member, IEEE, and Jaekyun Moon, Fellow, IEEE.

Abstract—This paper investigates interplay among storage overhead, bandwidth requirement and security constraint in clustered distributed storage. In the model used in our analysis, storage nodes are dispersed in multiple clusters. When a node fails, necessary content gets restored by downloading data from different nodes that may possibly be in other clusters. The bandwidth required for transferring data for node repair is assumed more scarce for cluster-to-cluster links than the links connecting intra-cluster nodes. Eavesdropping takes place on links across clusters only, and a fraction of the total number of clusters is assumed compromised. When a cluster is compromised, any repair traffic going in and out of it is eavesdropped. For this clustered model with eavesdroppers, we analyze the security of distributed storage systems (DSSs) and provide guidelines on designing system solutions for securing the data. First, under the setting of functional repair, we derive a general upper bound on the secrecy capacity, the maximum data size that can be stored in DSSs with perfect secrecy. In the practically important bandwidth-limited regime where the node storage size is equal to the repair bandwidth, the upper bound is shown to be achievable through proposed code constructions. Moreover, we obtain a closed-form expression for the required system resources - node storage size and repair bandwidth - to store a given amount of data with perfect secrecy. Secondly, we investigate the behavior of secrecy capacity as the number of compromised clusters increases. According to our mathematical analysis, the secrecy capacity decreases as a quadratic function until the number of compromised clusters reaches a certain threshold. Finally, based on the fundamental relationship between the system resources and the secrecy capacity, we provide a guideline on balancing intra- and cross-cluster repair bandwidths depending on the given system security level.

Index Terms—Distributed storage system, Secrecy capacity, Eavesdropper, Network coding

I. INTRODUCTION

A distributed storage system (DSS) is a network of storage nodes to reliably store massive amounts of data under frequent node failure events. Good examples are the large-scale data centers that employ storage nodes widely spread over the Internet [2]–[5]. The file to be stored is erasure-coded and distributed to \( n \) storage nodes, so that the legitimate user is able to reconstruct the complete file by contacting any \( k \) out of \( n \) active storage. In order to support reliable storage, a DSS is typically able to repair nodes in the case of node failure events using predefined coding scheme. If an active node fails, a “newcomer node” joins the system by downloading data from any \( d < n \) active helper nodes.

Clusters naturally arise in the form of racks in data centers but may also exist in many different variations that may involve wireless connection links as well. For example, wireless local area networks connected together in wireless links may be seen as forming clusters. Connection speeds within the cluster can be faster than those across clusters. When a node fails, a newcomer node gets established within the same cluster, which promptly downloads necessary data from any node within or outside the same cluster. Repair bandwidth across clusters is generally more restricted than within cluster, which adds significant twist to the trade-off picture of storage overhead versus repair bandwidth [6].

Storage nodes and links in DSSs can be vulnerable to intruder attacks, and guaranteeing system security against possible threats is a major challenge. Intruder models for DSSs are generally a node-based model [7], [8] where the intruder directly invades individual storage nodes. Much research efforts have gone into securing DSSs against these types of intruders [7]–[17]. Considering the clustered nature of storage nodes in many real world DSSs, however, conventional intruder models may not fully capture the behavior of the attackers. In this paper, we utilize what we call the cluster eavesdropper model, where cross-cluster repair links can be vulnerable to eavesdroppers whereas intra-cluster links are assumed secure. We assume that a certain number of clusters are compromised, and when a cluster is compromised any repair traffic that crosses its boundary is eavesdropped. Thus, in our cluster eavesdropper model an eavesdropper cannot access the individual storage nodes residing in the clusters, but can monitor data that go in and out of a certain number of clusters during the node repair process.

Under this model, this paper attempts to provide answers to the following key questions on securing clustered DSSs: What is the maximum amount of information that can be stored securely given the available cross- and intra-cluster repair bandwidths? As a larger number of clusters become compromised, sending data through other clusters will generally incur a bigger capacity loss. How rapidly would capacity be degraded as a function of the number of compromised clusters? How should the repair burden be balanced through intra-cluster and cross-cluster links in the presence of the given security threats?
A. Contributions

To our best knowledge, secrecy issues reflecting clustering characteristic of DSS have rarely been studied. Compared to previous works in securing DSS, we utilize a new node repair model suitable for a clustered DSS wherein the repair bandwidths assigned to intra- and cross-cluster links are imbalanced [6]. Under the setting of functional repair, which does not require the contents of the replacement node to be exactly the same as the failed node, we derive a general upper bound on the secrecy capacity, i.e., the maximum data size that can be stored in DSS with perfect secrecy, when clusters could be compromised by eavesdroppers. We demonstrate achievability of the bound in the practically important bandwidth-limited regime, where the node storage size $\alpha$ is equal to the repair bandwidth $\gamma$, by presenting secure coding schemes achieving the bound. The suggested coding schemes have an exact repair property, where regeneration directly restores the exact copies of the failed node, and are applicable for general system parameters $(n, k)$ with maximal helper nodes. The coding schemes are based on a concatenation of the symbol distribution rule in [18] and the nested maximum distance separable (MDS) code in [19]. Based on the analytical results on secrecy capacity, we obtain a closed-form expression for the required amount of resources - node storage size and repair bandwidth - to securely store a given amount of data. Finally, we provide some key insights on the system behavior under the security constraint. In particular, we show that secrecy capacity is represented as a quadratically decreasing function of the number of compromised clusters $L_c$ when $L_c$ is small enough, and remains at a constant value as $L_c$ is greater than a certain threshold. Moreover, we obtain the optimal ratio of cross- to intra-cluster repair bandwidth to maximize secrecy capacity, depending on the eavesdropper’s capability. Interestingly, the optimal solution depends largely on whether the portion of the compromised clusters is larger than $1 - \sqrt{1 - \frac{R}{1 - R}}$, where $R = k/n$ is the code rate.

B. Related Work

There have been some suggestions on modeling intruder behaviors in a DSS [7]–[9]. Roughly, intruder models are designed and classified into two types in DSSs. One is an active intruder model in which the attacker can maliciously change stored or transmitted data in the system, while the other is a passive eavesdropper model where the attacker only observes data without altering it. The passive eavesdropper model is classified further into node-based intruders [7], [8] and link-based intruders [9]. The node-based eavesdropper model suggested in [7], [8] allows access to individual storage nodes in the system. The model in [7] has a parameter $l$ which represents the total number of accessible nodes by the eavesdropper. In this model the intruder eavesdrops stored contents and downloaded data (during repair process) from $l$ compromised storage nodes. On the other hand, the authors of [8] divide the compromised storage nodes into two groups. They use two parameters $l_1, l_2$, such that an eavesdropper observes stored contents up to $l_1$ nodes and has an access to downloaded data up to $l_2$ nodes. When $l_1 = l_2 = l$, the two eavesdropper models suggested in [7], [8] become identical. Unlike the node-based eavesdroppers, the link-based eavesdroppers suitable for a DSS have been rarely discussed in the literature. The authors of [9] classify data centers into two groups, local and remote, and assume transmitted data from remote sites to a local site are more likely to be eavesdropped compared to transmitted data within a local site. In this aspect, our cluster eavesdropper model is similar to the model of [9] in that cross-cluster transmission are more likely to be compromised. However, the present paper reflects asymmetric repair more suitable for multiple-cluster scenarios while [9] still uses the conventional symmetric repair model of [20].

In the early work of Dimakis et al. [20], a fundamental trade-off relationship between node storage size $\alpha$ and repair bandwidth $\gamma$ is found to reliably store a file with given size. Repair-efficient codes which follow the fundamental trade-off curve are called regenerating codes. In the literature, two extreme points in the trade-off curve have been extensively studied: the minimum storage regenerating (MSR) point (minimize the storage size $\alpha$) and the minimum bandwidth regenerating (MBR) point (minimize the repair bandwidth $\gamma$).

There have been much discussions on how to securely store data against intruders [1], [7]–[17]. The authors of [7] define secrecy capacity as the maximum securely stored data in a system against passive eavesdropper with perfect secrecy. The authors of [7] provide a tight upper bound on secrecy capacity and present a coding scheme which achieves the upper bound in the bandwidth-limited regime. In [8], secure codes based on the product matrix codes are proposed. The suggested scheme achieves an upper bound presented in [7] at the MBR point with any set of system parameter values for $n, k$ and $d$. They also construct low-rate information-theoretically secure MSR codes with a constraint that an eavesdropper can observe stored data but no downloaded data ($l_2 = 0$). An improved upper bound on secrecy capacity in the MSR region and secure codes based on maximum rank distance codes are obtained in [10]. More secure codes following the fundamental trade-off curves are suggested and analyzed in [11]–[13]. The authors of [14] consider multiple node failure scenarios and employ an exact-repair secure coding scheme for the MBR region. Relaxing the perfect secrecy condition, the scenario of eavesdropper accessing a portion of the stored data is considered in [9], [15]. The trade-off between storage size and repair bandwidth in exact-repair secure codes in the presence of eavesdropper is investigated in [16], [17].

Although several researchers have already considered the clustered nature of storage nodes [6], [21]–[23], secrecy problems have not been previously addressed in a sufficient fashion in the context of clustered DSSs. The work of [24] discusses secrecy capacity of a heterogeneous DSS where each storage node has distinct storage size and requires different repair bandwidth. The model, however, does not reflect the clustered structure. Of the several existing DSS models, we basically adopt the model suggested in [6]. This model is suitable to a practical scenario where the available bandwidth associated with cross-cluster traffic is restricted compared to that for intra-cluster traffic. When recovering a failed node, helper nodes provide different amounts of data depending on whether
a regenerated node is in the same cluster or not.

C. Organization

This paper is organized as follows. We provide a threat model and an overview of the system for the clustered DSS in Section II. We provide results on secrecy capacity and derive the feasible region of system resources in Section III and IV. In Section V, we analyze secrecy capacity with various system parameters in the bandwidth-limited regime. Finally, secure code construction is provided in Section V-D and we draw conclusions and provide future research directions in Section VII.

II. SYSTEM AND THREAT MODEL

A. Clustered DSS model [6]

The system model for clustered DSS in [6] is reviewed. Suppose a source wants to store $M$ symbols of data in a clustered DSS, where each symbol belongs to a finite field $\mathbb{F}_2$. The source encodes the data to ensure reliability and distributes the encoded data to $n$ storage nodes, each of which has capacity to store $\alpha$ symbols over $\mathbb{F}_2$. Active $n$ storage nodes are distributed in $L$ clusters where the number of nodes per cluster is fixed to $n_1 = n/L$. A clustered DSS employs erasure coding so that a data collector should be able to reconstruct the entire data by retrieving encoded data from any choice of $k < n$ storage nodes.

In a DSS, individual storage nodes frequently fail to provide a reliable data [25]. In order to store message for a long period of time, system provides the node repair process to keep $n$ active storage nodes. When an active node fails, a replacement node joins the system by receiving $\beta_1$ symbols from each of the $d_1$ helper nodes in the same cluster, and $\beta_c$ symbols from each of the $d_c$ helper nodes in all other clusters. A clustered DSS considers a realistic scenario wherein the bandwidth assigned for intra-cluster repair is more abundant than that for cross-cluster repair, it is assumed that

$$\beta_1 \geq \beta_c.$$  

In case of node failures, a failed node and corresponding newcomer node reside in the identical cluster to keep constant number $n_1$ of active storage nodes per cluster. The total repair bandwidth to regenerate a single replacement node is expressed as $\gamma = \gamma_1 + \gamma_c$ where $\gamma_1 = d_1 \beta_1$ is the intra-cluster repair bandwidth and $\gamma_c = d_c \beta_c$ is the cross-cluster repair bandwidth. Finally, we define $q$ and $r$ as the quotient and remainder of $k$ divided by $n_1$. We summarize the parameters used in the paper in Table I.

Unlike other models dealing with a clustered DSS [23], [26], the present model assumes that individual storage nodes does not aggregate data within a cluster. Considering general scenarios where nodes are clustered and connected each other with limited resources, the present model without aggregation is practically well-suited. Now, we summarize a main result of [6] in the following lemma which obtains capacity, i.e., the maximum amount of reliably storable data, of a clustered DSS.

**Lemma 1** (Theorem 1 of [6]). The capacity $C$ of a clustered DSS with maximum number of helper nodes $(d_1 = n_1 - 1, d_c = n - n_1)$ is

$$C = \sum_{i=1}^{n_1} \sum_{j=1}^{g_i} \min \{\alpha, \rho_i \beta_1 + (n - \rho_i - j - \sum_{m=1}^{i-1} g_m) \beta_c\},$$

where

$$\rho_i = n_1 - i,$$

$$g_m = \begin{cases} q + 1, & m \leq r \\ q, & \text{otherwise} \end{cases}.$$  

**Remark 1** (Proposition 2 of [6]). For every $(i, j)$ with $i \in [n_1], j \in [g_i]$, the following inequality holds.

$$\gamma \geq \rho_i \beta_1 + (n - \rho_i - j - \sum_{m=1}^{i-1} g_m) \beta_c$$

B. Cluster eavesdropper model

We introduce the cluster eavesdropper model of a clustered DSS. Note that in case of node failures, the necessary content gets restored by downloading data from different nodes that may possibly be in other clusters. A cluster eavesdropper compromises a certain number $L_c$ of total clusters, and eavesdrops repair traffic of both incoming and outgoing data of compromised clusters. Repair traffic residing in the clusters and contents stored in individual nodes are not directly observed by a cluster eavesdropper. We assume that a cluster eavesdropper has sufficient computational power and complete knowledge of the repair protocol, like in other eavesdropper models [7], [8] for a DSS. The secrecy capacity of a clustered DSS is defined as the maximum amount of data that can be

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>$n$</td>
<td>number of storage nodes</td>
</tr>
<tr>
<td>$k$</td>
<td>number of DC-contacting nodes</td>
</tr>
<tr>
<td>$L$</td>
<td>number of clusters</td>
</tr>
<tr>
<td>$n_1 = n/L$</td>
<td>number of nodes in a cluster</td>
</tr>
<tr>
<td>$\alpha$</td>
<td>storage capacity of each node</td>
</tr>
<tr>
<td>$d = d_1 + d_c$</td>
<td>total number of helper nodes</td>
</tr>
<tr>
<td>$d_1$</td>
<td>number of intra-cluster helper nodes</td>
</tr>
<tr>
<td>$d_c$</td>
<td>number of cross-cluster helper nodes</td>
</tr>
<tr>
<td>$\beta_1$</td>
<td>intra-cluster repair bandwidth (per node)</td>
</tr>
<tr>
<td>$\beta_c$</td>
<td>cross-cluster repair bandwidth (per node)</td>
</tr>
<tr>
<td>$\gamma_1 = d_1 \beta_1$</td>
<td>intra-cluster repair bandwidth</td>
</tr>
<tr>
<td>$\gamma_c = d_c \beta_c$</td>
<td>cross-cluster repair bandwidth</td>
</tr>
<tr>
<td>$\gamma = \gamma_1 + \gamma_c$</td>
<td>total repair bandwidth</td>
</tr>
<tr>
<td>$q = \lfloor k/n_1 \rfloor$</td>
<td>quotient of $k$ divided by $n_1$</td>
</tr>
<tr>
<td>$r = \text{mod}(k, n_1)$</td>
<td>remainder of $k$ divided by $n_1$</td>
</tr>
</tbody>
</table>

\[1\] It is assumed that only single node failure occurs in the system since a vast majority of data recovery required in practice is due to single node failure [25].
stored in the system with perfect secrecy requirement, i.e., without revealing any data to the cluster eavesdropper. To be more precise, for a given message vector \( m \) to be stored in the system and a vector of revealed symbols \( e \) which the eavesdropper observes, the condition

\[
H(m|e) = H(m)
\]

represents the perfect secrecy requirement, where \( H \) is an entropy function. The legitimate data collector can access the stored data by retrieving information from a given number of nodes, while a cluster eavesdropper cannot obtain any information about the data.

In our cluster eavesdropper model, we assume that only the data used for node repair can be observed by the eavesdropper. In other words, the cluster eavesdropper has no access to data when the source stores them to nodes or when the data collector retrieves them from nodes. We made this assumption due to the following reason. In the case of node repair, each node in other clusters helps \( \beta_c \) symbols to regenerate a failed node. In the case of data collection, however, each node provides all of its contents (\( \alpha \) symbols) to a data collector, which is far greater than \( \beta_c \). (Similarly, in the case of data storing, each node receives \( \alpha \) symbols from the source.) Thus, in the system designer’s perspective, securing the channel for data storing/collection has a higher priority, compared to securing the channel for node repair.

A simple example on how the attacker eavesdrops data in a clustered DSS is shown in Fig. 1. There are \( n = 9 \) storage nodes which are dispersed in \( L = 3 \) clusters. A single cluster is compromised which is illustrated as a smooth square with dashed line. Suppose a node in the compromised cluster fails. Then, a node is regenerated by receiving \( \beta_I \) units of data from a node within the cluster and \( \beta_c \) units of data from each node in the other clusters. Notice that links from nodes in the uncompromised clusters must pass through the boundary of the compromised cluster, which is observed by eavesdropper. Therefore, \( 6\beta_c \) symbols are eavesdropped. Here, compromised and uncompromised links are illustrated as dashed and solid links in the figure, respectively.

III. Secrecy Capacity Against Cluster Eavesdropper

In this section, we summarize our main results, the secrecy capacity of a clustered DSS against a cluster eavesdropper. Here, we assume that the number of helper nodes is set to the maximum value \((d_I = n_I - 1, d_c = n - n_I)\) since maximizing the number of helper nodes maximizes secrecy capacity as stated in the following proposition. Under this assumption, parameters \( n, k, L, \alpha, \beta_I \) and \( \beta_c \) completely specify a clustered DSS. We denote such system as \((n, k, L, \alpha, \beta_I, \beta_c)\) clustered DSS.

**Proposition 1.** Let \( \gamma, \gamma_c \) be specified for a clustered DSS. Setting \( d_I \) and \( d_c \) to the maximum possible values maximizes secrecy capacity in the bandwidth-limited regime.

**Proof.** See Appendix D-A.

A. Upper bound on secrecy capacity

Consider a \((n, k, L, \alpha, \beta_I, \beta_c)\) clustered DSS and a cluster eavesdropper who compromises \( L_c \) clusters. Under the functional repair setting in the regeneration process, we present a theorem which provides an upper bound on secrecy capacity against a cluster eavesdropper.

**Theorem 1.** The secrecy capacity of a \((n, k, L, \alpha, \beta_I, \beta_c)\) clustered DSS against \( L_c \) cluster eavesdropping is upper-bounded by

\[
C_s(\alpha, \beta_I, \beta_c) \leq C_U
\]

where

\[
C_U = \begin{cases}
L_c \sum_{i=1}^{n_I} \min\{\alpha, (n_I - i)\beta_I\} + C_r, & n_I L_c < k \\
q \sum_{i=1}^{n_I} \min\{\alpha, (n_I - i)\beta_I\} + \sum_{i=1}^{n'} \min\{\alpha, (n_I - i)\beta_I\}, & \text{otherwise}
\end{cases}
\]

Here, \( q \) and \( r \) are defined in Table I and \( C_r \) is capacity [6] of a reduced \((n', k', L', \alpha', \beta_I, \beta_c)\) clustered DSS with parameters

\[
n' = n - n_I L_c, \\
k' = k - n_I L_c, \\
L' = L - L_c, \\
\alpha' = \min\{\alpha, (n_I - 1)\beta_I + (n' - n_I)\beta_c\}.
\]

**Proof.** See Appendix A-A.

**Remark 2.** Note that there are two cases where eavesdropping does not occur. Under these settings, secrecy capacity is equal to capacity obtained in [6] which considers the clustered model without security constraints.

(a) When \( L_c = 0 \), the upper bound \( C_U \) obtained in Theorem 1 is reduced to capacity of a \((n, k, L, \alpha, \beta_I, \beta_c)\) clustered DSS.

(b) When \( \beta_c = 0 \), the upper bound \( C_U \) reduces to capacity of a \((n, k, L, \alpha, \beta_I, \beta_c = 0)\) clustered DSS. This can be obtained as follows. Note that capacity expression of a
Theorem 1. Consider a \( (n, k, L, \alpha, \beta_1, \beta_c) \) clustered DSS. In the bandwidth-limited regime, the secrecy capacity against \( L_c \) cluster eavesdropping is determined as
\[
C^\text{BL}_{c} (\beta_1, \beta_c) = R
\]
where \( R \) is given by (2).

Proof. As stated in Theorem 5 and 6, we present code constructions where \( R \) symbols can be stored with perfect secrecy under the desired setting with the presence of a cluster eavesdropper. Thus, a clustered DSS securely stores \( R \) symbols, i.e.,
\[
C^\text{BL}_{c} (\beta_1, \beta_c) \geq R.
\]

Combining this with Corollary 1 completes the proof. \( \Box \)

IV. FEASIBLE \( (\alpha, \beta_1, \beta_c) \) TO STORE DATA WITH PERFECT SECRECY

In the previous section, we derived an upper bound of secrecy capacity under general system parameters and resources \( (n, k, L, \alpha, \beta_1, \beta_c) \), and the exact value of secrecy capacity in the bandwidth-limited regime. Using these results, we analyze required resources \( (\alpha, \beta_1, \beta_c) \) to store \( M_s \) symbols of data with perfect secrecy, given the system parameters \( (n, k, L) \). We introduce an important parameter
\[
e \triangleq \beta_c / \beta_1,
\]
which is a ratio of cross- to intra-cluster repair bandwidth. Since we consider practical scenarios where cross-cluster...

Remark 3. The upper bound \( C^U_c \) is obtained based on an information flow graph; such, the analysis inevitably bears some similarities to that on secrecy capacity of the non-clustered DSS model in [20]. However, there are some significant additional challenges in pursuing analysis for the clustered model. In the non-clustered model that assumes symmetric repair (i.e., \( \beta_1 = \beta_c \)), identifying \( l \) compromised nodes for the worst-case scenario (when the intruder eavesdrops the maximum amount of information) suffices to find the least upper bound. On the other hand, in the clustered model with asymmetric repair (i.e., \( \beta_1 \neq \beta_c \)), identifying the order of failed nodes as well as \( L_c \) compromised clusters in the worst-case scenario is required to obtain the least upper bound.

B. Secrecy capacity in the bandwidth-limited regime

Suppose a clustered DSS satisfies \( \alpha \geq \gamma \). From Remark 1, the capacity \( C \) stated in (1) is expressed as
\[
C = \sum_{i=1}^{n_i} g_i \min \{ \alpha, (n_l - i) \beta_1 \} + \sum_{j=1}^{k_j} (\rho_j \beta_1 + (n - \rho_j - j - (\sum_{m=1}^{i-1} g_m) \beta_c) \}
\]
This regime is said to be bandwidth-limited since only the repair bandwidths \( \beta_1 \) and \( \beta_c \) limit the capacity \( C \). Notice that if node storage size \( \alpha \) is strictly larger than \( \gamma \), \( \alpha \) can be reduced without affecting the capacity. Since reducing system resources is desired, we say that the bandwidth-limited regime is the regime satisfying the equality, \( \alpha = \gamma \). Based on Theorem 1, the secrecy capacity in the bandwidth-limited regime is upper bounded as follows, with the bound being a function of intra- and cross-cluster repair bandwidths \( \beta_1 \) and \( \beta_c \). Applying the bandwidth-limited condition \( \alpha = \gamma = (n_l - 1) \beta_1 + (n - n_l) \beta_c \) to Theorem 1 directly completes the proof.

Corollary 1. Consider a \( (n, k, L, \alpha, \beta_1, \beta_c) \) clustered DSS. In the bandwidth-limited regime, the secrecy capacity against \( L_c \) cluster eavesdropping is bounded by
\[
C^\text{BL}_{c} (\beta_1, \beta_c) \leq R
\]
with
\[
R = \begin{cases} 
L_c \beta_1 \frac{n_l(n_l-1)}{2} + C_r, & n_l L_c < k \\
q \beta_1 \frac{n_l(n_l-1)}{2} + \sum_{i=1}^{r} (n_l - i) \beta_1, & \text{otherwise}
\end{cases}
\]
where \( C_r \) is the capacity of a reduced \( (n', k', L', \alpha', \beta_1, \beta_c) \) clustered DSS with parameters
\[
n' = n - n_l L_c, \\
k' = k - n_l L_c, \\
L' = L - L_c
\]
and \( \alpha' = \min \{ \alpha, (n_l - 1) \beta_1 + (n' - n_l) \beta_c \} \).

The upper bound of secrecy capacity in the bandwidth-limited regime obtained in the above corollary turns out to be the exact value of secrecy capacity, as stated in the following theorem.
bandwidth is less than intra-cluster repair bandwidth, the range of \( \epsilon \) is expressed as
\[
0 \leq \epsilon \leq 1.
\]
Note that \( \epsilon = 0 \) is the case that cross-cluster communication does not occur in the system, thus eavesdropping cannot happen. Thus, secrecy capacity is equal to capacity, and its analysis is identical to [6]. We only consider the non-trivial case for
\[
0 < \epsilon \leq 1.
\]
As stated in Table I, total repair bandwidth of a clustered DSS with a maximal number of helper nodes is given as
\[
\gamma = (n_j - 1)\beta_t + (n - n_j)\beta_c.
\]
Combining (3) with (4), intra- and cross-cluster repair bandwidth can be expressed as functions of \( \gamma \) and \( \epsilon \):
\[
\begin{align*}
\beta_t &= \frac{\gamma}{(n_j - 1) + (n - n_j)\epsilon}, \\
\beta_c &= \frac{\gamma\epsilon}{(n_j - 1) + (n - n_j)\epsilon}.
\end{align*}
\]
Using these expressions, the required resources \((\alpha, \beta_t, \beta_c)\) can be expressed more compactly by using resources \((\alpha, \gamma)\) for some given ratio \(\epsilon\). For example, secrecy capacity \(C_s(\alpha, \beta_t, \beta_c)\) can be expressed as \(C_s(\alpha, \gamma)\) for a given \(\epsilon\).

A. Parameters

In this subsection, we define parameters that used in the corollaries as follows. Here, \([k]\) stands for a set \(\{1, \cdots, k\}\).
\[
n' = n - n_jL_c,
\]
\[
k' = k - n_jL_c,
\]
\[
g_t = \begin{cases} q + 1, & t \leq r \\ q, & \text{otherwise}, \end{cases}
\]
\[
h_t = \min_{s \in [n_j]} \sum_{i=1}^{s} (g_i - L_c) \geq t, \quad t \in [k'],
\]
\[
z_t = \begin{cases} (n' - n_t - t + h_t)\epsilon + (n_j - h_t), & t \in [k'] \\ \infty, & t = 0, \end{cases}
\]
\[
x_t = \begin{cases} (n_j - t) & t \in [n_j - 1] \\ 0, & t = n_j, \\ \infty, & t = n_j + 1, \end{cases}
\]
\[
y_t = \begin{cases} (n_j - t + 1)\epsilon & t \in [k'] \\ 0, & t = 0, \end{cases}
\]
\[
u_t = \begin{cases} \text{the minimum of } \{x_i\}_{i=1}^{n_j} \cup \{y_j\}_{j=1}^{k'}, & t \in [n_j + k'] \\ 0, & t = 0, \end{cases}
\]
\[
p_t = \arg \max_{i \in [n_j - 1] \cup \{0\}} x_i \leq \gamma, \quad \gamma \in (u_t \alpha, u_{t+1} \alpha],
\]
\[
q_t = \arg \max_{j \in [k'] \cup \{0\}} y_j \leq \gamma, \quad \gamma \in (u_t \alpha, u_{t+1} \alpha],
\]
\[
r_t = \sum_{i=t+1}^{n_j} L_c/x_i, \quad s_t = \sum_{j=t+1}^{k'} \frac{1}{y_j},
\]
\[
a_t = L_c p_t + q_t, \quad b_t = r_p_t + s_q_t,
\]
\[
\tau = \{\tau \in [k' + n_j - 2] \cup \{0\} : u_t \alpha < \gamma \leq u_{t+1} \alpha\},
\]
\[
c_t = \begin{cases} (q + 1)t, & 0 \leq t < r \\ qt + r, & r \leq t \leq n_j - 1, \end{cases}
\]
\[
d_t = \begin{cases} \sum_{i=t+1}^{n_j} \frac{q}{x_i} + \sum_{i=t+1}^{n_j} \frac{r}{x_i}, & 0 \leq t < r \\ \sum_{i=t+1}^{n_j} \frac{r}{x_i}, & r \leq t \leq n_j - 1. \end{cases}
\]

B. Main results

We say that a point \((\alpha, \gamma)\) is feasible to store \(M_s\) symbols with perfect secrecy if and only if \(C_s(\alpha, \gamma) \geq M_s\). Following Corollaries 2, 3 reveal infeasible \((\alpha, \gamma)\) points to store \(M_s\) symbols with perfect secrecy. Corollary 2 covers the case for eavesdropping capability satisfying \(L_c < k/n_j\) while Corollary 3 handles \(L_c \geq k/n_j\) case.

**Corollary 2.** Consider a clustered DSS for storing \(M_s\) symbols of data with perfect secrecy against \(L_c < k/n_j\) cluster eavesdropping. For any \(\gamma < \gamma^*(\alpha)\), it is impossible to store the \(M_s\) symbols with perfect secrecy. The threshold value \(\gamma^*(\alpha)\) can be obtained as
\[
\gamma^*(\alpha) = \begin{cases} \infty, & \alpha \in (0, M_s/L_c), \\ \frac{M_s - a_1 \alpha}{b_1}, & \alpha \in \left[\frac{M_s}{a_1 + b_1}, \infty\right), \quad (t = k' + n_j - 2, k' + n_j - 3, \cdots, 1), \\ \frac{M_s}{b_1}, & \alpha = \frac{M_s}{a_1 + b_1}, \infty. \end{cases}
\]

**Proof.** See Appendix B-A.

**Corollary 3.** Consider a clustered DSS for storing \(M_s\) symbols of data with perfect secrecy against \(L_c \geq k/n_j\) cluster eavesdropping. For any \(\gamma < \gamma^*(\alpha)\), it is impossible to store the \(M_s\) symbols with perfect secrecy. The threshold value \(\gamma^*(\alpha)\) can be obtained as
\[
\gamma^*(\alpha) = \begin{cases} \infty, & \alpha \in (0, M_s/L_c), \\ \frac{M_s - a_1 \alpha}{b_1}, & \alpha \in \left[\frac{M_s}{a_1 + b_1}, \frac{M_s}{a_1 + b_1}, \infty\right), \quad (t = \tau - 1, \tau - 2, \cdots, 1), \\ \frac{M_s}{b_1}, & \alpha = \frac{M_s}{a_1 + b_1}, \infty. \end{cases}
\]

**Proof.** See Appendix B-B.

\[\Box\]
Remark 4. Every \((\alpha, \gamma^*(\alpha))\) pair satisfies \(C^U_s(\alpha, \gamma^*(\alpha)) = M_s\).

Consider a \((n, k, L, \alpha, \beta_L, \beta_c)\) clustered DSS to store \(M\) symbols without the secrecy concern. It is known that it can store \(M\) symbols with minimum storage overhead \(\alpha = M/k\) at the expense of repair bandwidth if and only if \(\epsilon \geq \frac{1}{n-k}[6]\). However, in a clustered DSS with perfect secrecy constraint, it is impossible to store \(M_s\) symbols with storage overhead \(\alpha = M_s/k\). The lower bound of storage size \(\alpha\) to secure \(M_s\) symbols of data is stated as follows.

Theorem 3. A clustered DSS cannot store \(M_s\) symbols of data with perfect secrecy when \(\alpha < \frac{M_s}{k-L_c}\).

Proof. See Appendix A-B.

We illustrate \((\alpha, \gamma^*(\alpha))\) pairs with various \(\epsilon\) values in Fig. 2. The points below each curve represent an infeasible region to store \(M_s = 85\) with perfect secrecy. Here, the case with \(\epsilon = 0\) corresponds to the scenario where a failed node is completely repaired without contacting nodes in other clusters. The \(\epsilon = 1\) case becomes a symmetric repair where \(\beta_L = \beta_c\). For every \(\epsilon\), it is shown that the \((\alpha, \gamma)\) point is infeasible for all \(\alpha < \alpha^\text{min}\) where \(\alpha^\text{min} = \frac{M_s}{k-L_c} \approx 1.037\).

Fig. 2: The set of \((\alpha, \gamma^*(\alpha))\) points to store \(M_s = 85\) symbols with perfect secrecy, under the setting of \(n = 100, k = 85, L = 10\) and \(L_c = 3\)

Now, we provide a feasible region to store \(M_s\) symbols in the following corollary.

Corollary 4. Consider a clustered DSS for storing \(M_s\) symbols with perfect secrecy for a given \(0 < \epsilon \leq 1\). It is possible to securely store data with resources \((\alpha, \gamma)\) where

\[
\alpha \geq \gamma^\text{min}, \quad \gamma \geq \gamma^\text{min}
\]

with

\[
\gamma^\text{min} = \begin{cases} 
M_s/b_0, & L_c < k/n_1 \\
M_s/d_0, & L_c \geq k/n_1.
\end{cases}
\]  

(11)

Proof. First, we show that \((\alpha, \gamma) = (\gamma^\text{min}, \gamma^\text{min})\) is a feasible point to store \(M_s\) symbols with perfect secrecy. In the bandwidth-limited regime where \(\alpha = \gamma\), secrecy capacity is explicitly derived in Theorem 2, achieving the upper bound proposed in Theorem 1. Therefore,

\[
C_s(\alpha, \gamma) = C_s(\gamma^\text{min}, \gamma^\text{min}) = C^U_s(\gamma^\text{min}, \gamma^\text{min}) = M_s.
\]

Trivially, it is possible to store \(M_s\) symbols with perfect secrecy for an arbitrary \((\alpha, \gamma)\) point where \(\alpha \geq \gamma^\text{min}\) and \(\gamma \geq \gamma^\text{min}\), which utilizes more system resources than \((\alpha, \gamma) = (\gamma^\text{min}, \gamma^\text{min})\).

Fig. 3: Feasible and infeasible regions to store \(M_s = 8\) symbols with perfect secrecy, under the setting of \(n = 15, k = 8, L = 3, \epsilon = 1/7\) and \(L_c = 1\)

An example of \((\alpha, \gamma^*(\alpha))\) points and corresponding feasible/infeasible regions are illustrated in Fig. 3. The set of \((\alpha, \gamma^*(\alpha))\) points and the infeasible region are obtained from Corollary 3, while the feasible region is obtained from Corollary 4. Whether the blue shaded region is feasible or not remains unknown.

Here we shed light on the difference between the minimum bandwidth point in this paper and the Minimum Bandwidth Regenerating (MBR) point in [6], [20]. Note that both points have the same definition, the point having the minimum storage given the minimum bandwidth constraint, while the functional regeneration is not guaranteed for the former point. It is well known that \(\alpha = \gamma\) generally holds for the MBR points of [6], [20]. However, considering the set of \((\alpha, \gamma^*(\alpha))\) pairs obtained from the above corollaries, \(\alpha = \gamma\) does not hold at the minimum bandwidth point. The reason for this difference is that the present paper targets a secure data storage against eavesdroppers, not a fault-tolerant data storage as in [20] and [6]. The minimum bandwidth point is depicted as a square mark while the point with minimum bandwidth satisfying \(\alpha = \gamma\) is shown as a square mark in Fig. 3.

In Fig. 4, we illustrate the set of \((\alpha, \gamma^*(\alpha))\) points for a various number of compromised clusters \(L_c\). Note that \(L_c = 0\) corresponds to the case where there is no eavesdropper. When \(L_c = 0\), every communication link is secure, and thus the least amount of resources are required. As the power of eavesdropper becomes stronger, the more resources are required to secure data. The \(L_c = 10\) case is a system
A. Secrecy capacity when \( L_c < k/n_I \)

As the number of compromised clusters becomes larger, the amount of securely storable data in a system is clearly decreased. Let us first see how rapidly secrecy capacity degrades as a function of \( L_c \). For fair comparison, we fixed system parameters \((n, k, L, \alpha, \gamma)\) and use \((\epsilon, L_c)\) as variables. We define the secrecy capacity loss in the bandwidth-limited regime as

\[
\Delta C_s^{BL}(L_c) := C_s^{BL}(L_c + 1) - C_s^{BL}(L_c).
\]

where every cluster is compromised, i.e., every cross-cluster communication is monitored by a cluster eavesdropper. Fig. 5 illustrates \( \gamma^{min} \), the minimum required bandwidth to achieve the secrecy capacity, defined in (11) as a function of \( L_c \). It is shown that \( \gamma^{min} \) gradually increases with \( L_c \) and then gets saturated.

V. IMPLICATIONS IN THE BANDWIDTH-LIMITED REGIME

In this section, we focus on the bandwidth-limited regime and answer key questions related to securely storing data in a clustered DSS against cluster eavesdropping. The bandwidth \( \gamma \) required for communication is generally the more scarce resource than storage node size \( \alpha \), given the abundance of cheap commodity disks. Since the required bandwidth for repairing failed nodes is minimized in this regime, we feel this is a practical important scenario, reflecting the real world.
should be able to assign cross-cluster bandwidth more freely. When distributing data into large number of clusters, capacity as a function of cross-cluster communication can be eavesdropped, setting zero capacity. Also, it is observed that the loss of secrecy is also observed in [6]. In contrast, the secrecy capacity hardly affects the capacity of the entire system, which is known that capacity is an increasing function of the secrecy capacity degrades when increasing secrecy capacity.

Proof. From Theorem 2, the secrecy capacity with \( L_c \geq k/n \) is expressed as

\[
C_s^{BL} = q \sum_{i=1}^{n_I} (n_I - i) \beta_I + \sum_{i=r+1}^{n_I} (n_I - i) \beta_I \\
= (q + 1) \sum_{i=1}^{n_I} (n_I - i) \beta_I + \sum_{i=r+1}^{n_I} (n_I - i) \beta_I \\
= \frac{(q + 1)(n_I - 1)n_I - (n_I - r - 1)(n_I - r)}{2} \beta_I. \quad (13)
\]

Combining (13) with (5) completes the proof.

Fig. 7 illustrates the secrecy capacity as a function of \( \epsilon \) with \( L_c \geq k/n \), under the setting of \( n = 1000, k = 850, L = 100 \) and \( \alpha = \gamma = 1 \) when \( L \) is small.

C. Optimal \( \epsilon \) to maximize secrecy capacity

Suppose that there exists no eavesdropper \( (L_c = 0) \) in the system (where the secrecy capacity is equal to capacity). It is known that capacity is an increasing function of \( \epsilon \) [6], also implied in Fig. 6. This is consistent with the previous work on asymmetric repair in [24] which shows that the symmetric repair maximizes capacity of the system. As \( L_c \) becomes larger, the secrecy capacity degrades with different speeds depending on \( \epsilon \), and it is seen that the curves intersect at one point, as seen in Fig. 6. Denote this point as \( L_c^* \). As \( L_c \) exceeds this number, repair across clusters always degrade secrecy capacity; when \( L_c \) is below this critical number, cross-cluster repair always helps secrecy capacity. \( L_c^* \) can be explicitly derived as a function of system parameters as follows.

Theorem 4. With various \( \epsilon \), \( C_s^{BL}(L_c) \) intersects on one point \( L_c^* \). The intersection point is given by

\[
L_c^* = (L - 1/2) - \sqrt{(L - 1/2)^2 - \frac{\omega}{n_I^2(n_I - 1)}} \quad (14)
\]

where \( \omega = qn_I(n_I - 1)(n - qn_I - 2r) + r(r - 1)(n - n_I) \).

Proof. See Appendix A-C.

The intersection point \( L_c^* \) is further approximated in the following corollary. We observed that the approximation was very accurate as \( L \) grows.

Corollary 7. The intersection point (14) is approximated as

\[
L_c^* \approx L_{c,approx} = L(1 - \sqrt{1 - R(1 - R)})
\]

where \( R = k/n \) is the code rate.

Proof. See Appendix B-D.
In this case, capacity is a decreasing function of $\epsilon$. If $L_c/\alpha$ is smaller than this, then it is recommended to increase $\epsilon$ to 1 to maximize secrecy capacity. Otherwise, minimizing $\epsilon$ to 0 would maximize secrecy capacity. It is shown that as the rate is closer to 0.5, the system can tolerate a larger number of compromised clusters before cross-cluster should be disallowed. Notice that $L_{c,approx}^*$ is upper bounded as

$$L_{c,approx}^* = L(1 - \sqrt{1 - R(1 - R)}) \leq L(1 - \sqrt{3}/2) \approx 0.134L.$$ 

If a cluster eavesdropper occupies more than 13.4% of clusters, reducing $\epsilon$ is always beneficial to maximize the secrecy capacity.

Moreover, we depict secrecy capacity as a function of $\epsilon$ in Fig. 9 where the system parameters are consistent with Fig. 6. If $L_c$ is lower than the critical $L_c^* = 6.629$, the secrecy capacity is an increasing function of $\epsilon$. Otherwise, the secrecy capacity is a decreasing function of $\epsilon$.

### D. Secrecy capacity as a function of $L$

Suppose $L_c$ is set to a constant value. Now see what happens to secrecy capacity as $L$ changes. Fig. 10 illustrates secrecy capacity as a function of $L$ for different values of $\epsilon$. Since we assume that all clusters have the identical number $n_1 = n/L$ of nodes, we only plotted for the set of $L$ values which divide $n$. As $L$ becomes larger, the secrecy capacity tends to decrease up to a certain point, and then increases, for all positive values of $\epsilon$. When $L$ is small enough, storage nodes are distributed to a few number of clusters. Data transmission using intra-cluster links occupies a large portion of total communication. Thus, Eve can access a smaller amount of data, which increases secrecy capacity in small $L$. When $L$ is large enough, an eavesdropper with fixed $L_c$ observes only a fraction of total clusters. Accessibility of Eve to communication links is limited in large $L$, which again increases secrecy capacity. Here, again note that distributing data to nodes in a small number of clusters is susceptible to cluster failure scenarios which may cause permanent data loss. This would mean that when $L_c$ is fixed, increasing the number of clusters is highly beneficial.

![Fig. 8: $L_{c,approx}^*/L$ versus code rate $k/n$.](image)

![Fig. 9: Capacity and secrecy capacity as a function of $\epsilon$, under the setting of $n = 1000, k = 850, L = 100$ and $\alpha = \gamma = 1$. In this case, $L_c^* = 6.629$ holds.](image)

![Fig. 10: Secrecy capacity as a function of $L$, under the setting of $n = 100, k = 80, \alpha = 1, \gamma = 1$ and $L_c = 3$](image)

### VI. Secure Code Constructions at the Bandwidth-Limited Regime

In this section, we provide explicit coding schemes to securely store data comprising of $C_n^{BL}$ symbols in the bandwidth-limited regime, where each symbol belongs to a finite field $F_s$ of size $s$. In the case of $\beta_c = 0$, note that a cluster eavesdropper cannot observe any data regardless of $L_c$, since cross-cluster communication does not occur in this setting. Thus, secrecy capacity is equal to the capacity. An explicit code construction achieving the capacity in the bandwidth-limited regime is obtained in [18], thus it trivially achieves the secrecy capacity when $\beta_c = 0$. Hereafter, we focus on nontrivial cases where $\beta_c \neq 0$. The following secure codes are applied to clustered DSS with arbitrary system parameters...
(n, k, L, α, β_1, β_c) and eavesdropping capability L_c, when the following conditions are guaranteed: 1) the number of helper nodes is maximized, i.e., d_l = n_l - 1 and d_c = n - n_l, 2) n is divisible by L, 3) α = γ (bandwidth-limited condition), 4) \chi := \beta_c/\beta_1 is an integer, and 5) field size s is greater than or equal to θ, where θ is defined in (15). From now on, we set β_c = 1 and β_1 = γ without a loss of generality. The coding for arbitrary β_c > 1 is completed by repeatedly applying the code with β_c = 1 in a parallel manner.

Before providing the code construction rules, we introduce a two dimensional representation of nodes in a clustered DSS (Fig. 11). Consider a \((n, k, L)\) clustered DSS where each cluster has \(n_l = n/L\) storage nodes. For \(l \in [L]\) and \(j \in [n_l]\), the \(j^{th}\) node in \(l^{th}\) cluster is represented as node \(N(l, j)\). The set of active \(n\) storage nodes is represented as

\[ N = \{ N(l, j) : l \in [L], j \in [n_l] \}. \]

![Fig. 11: Two dimensional representation of a clustered DSS](image)

A. Code Construction on \(n_lL_c < k\)

For a \((n, k, L, α, β_1 = γ, β_c = 1)\) clustered DSS, we present a secure code construction to store data in the bandwidth-limited regime, when the capability of a cluster eavesdropper is restricted to \(L_c < k/n_l\). Note that storage size \(α\) of each node in the bandwidth-limited regime is determined as

\[ α = γ = d_lβ_1 + d_cβ_c = (n_l - 1)γ + (n - n_l). \]

We first introduce the main concept of the wiretap channel problem [27], a solution to which will be used in our code construction. Consider the following situation. Alice wants to convey \(R\) message symbols to Bob where each symbol is independent and uniformly chosen in the field \(\mathbb{F}_s\). Alice uses an erasure channel with transmission length \(θ > R\), and knows that Bob receives arbitrary \(M\) among \(θ\) encoded symbols and Eve wiretaps \(K ≤ M\) out of \(θ\) symbols. This channel is denoted as \(h_{wt}(θ, M, K)\). The role of Alice is to encode \(R\) message symbols into \(θ\) symbols in a way that the channel can convey all the message symbols to Bob, while Eve cannot retrieve any information on the message symbols. We summarize the result of [19] for securing \(R\) symbols with perfect secrecy in the following lemma. Note that a random vector of size \(K\) is used in the encoding process to guarantee perfect secrecy.

**Lemma 2** (Main result of [19]). Consider a wiretap channel \(h_{wt}(θ, M, K)\). The secrecy capacity of this channel, i.e., the maximum number of message symbols that can be conveyed with perfect secrecy, is \(R = M - K\). Moreover, for a given message vector \(s = [s_1, \ldots, s_R] ∈ \mathbb{F}_s^R\), the secrecy capacity is achievable by conveying a vector of encoded symbols \(c = [c_1, \ldots, c_0] ∈ \mathbb{F}_s^R\) generated by using a nested MDS code:

\[ c = [k] s \begin{bmatrix} G_K & \mathbf{0} \\ G_S & \mathbf{0} \end{bmatrix} = kG_K + sG_S, \]

where \(k = [k_1, \ldots, k_K] ∈ \mathbb{F}_s^K\) is a random vector evenly distributed over \(\mathbb{F}_s^K\), while \(G_K\) and \(G = [G_K | G_S]\) are generator matrices of MDS codes with size \(K×θ\) and \(M×θ\), respectively.

In the construction of the suggested coding scheme, we use several parameters

\[ \theta = (χ - 1) \left( \binom{n_l}{2} L + \binom{n}{2} \right), \quad (15) \]

\[ M = kα - \frac{1}{2} (χ - 1)(qL_c^2 + r^2 - k) - \frac{k(k - 1)}{2}, \quad (16) \]

\[ K = n_l^2L_c(L - \frac{L_c + 1}{2}), \quad (17) \]

and a matrix \(V_t\) defined as follows. Consider a fully connected graph \(G_t\) with \(t\) vertices. Then, \(V_t\), the incidence matrix of \(G_t\) is a \(t \times \binom{t}{2}\) matrix given by:

\[ V_t(j, i) = \begin{cases} 1, & \text{if } i^{th}\text{ }edge \text{ is connected to } j^{th}\text{ node} \\ 0, & \text{otherwise} \end{cases} \quad (18) \]

An example of incidence matrix \(V_4\) is given in Fig. 12.

Using the parameters defined above, the explicit code construction rule for securely storing \(M - K\) message symbols is provided in Algorithm 1. According to Theorem 5, the code in Algorithm 1 is shown to achieve the secrecy capacity of \((n, k, L, α, γ, 1)\) clustered DSS under the condition of \(L_c < k/n_l\). Here, \(K\) random symbols are used when generating encoded symbols.

Before stating Theorem 5, we provide some physical insights into two important parameters, \(M\) and \(K\) used in Algorithm 1. First, the properties of the symbol distribution rule in Step 2 of Algorithm 1 is described in Lemma 3. Second, the physical meaning of \(K\) is stated in Lemma 4.
Algorithm 1 Secure code construction on $n_1 L_c < k$

**Input:** System parameters $n, k, L, L_c, \chi$.
Message vector $s \triangleq [s_1, \ldots, s_{M-K}] \in F_s^{M-K}$

**Output:** Symbols stored on each node in $N$.

1. **Step 1.** Generate encoded symbols $\{c_1, \ldots, c_{\theta}\}$
   Applying the code in Lemma 2 to secure $s$ in $h_{\text{disc}}(\theta, M, K)$, generate an encoded vector $c \triangleq [c_1, \ldots, c_{\theta}] = [\lfloor k \rfloor s] [G_K]^{-1} [G_S]^{-1}$.

2. **Step 2.** Distribute encoded symbols $\{c_1, \ldots, c_{\theta}\}$ to nodes under the following rules:
   - Node $N(l, j)$ stores a symbol $c_i$, if and only if $V_{n_1}(l, j) + 1 = i$.
   - Node $N(l, j)$ stores $c_i$ if and only if $V_{n_1}(l, j) + 1 = i$.
   - Node $N(l, j)$ stores $c_i$ if and only if $V_{n_1}(l, j) + 1 = i$.

**Lemma 3** (Theorem 2 and Lemma 2 in [18]). Consider a $(n, k, L, \alpha, \chi, 1)$ clustered DSS in the bandwidth-limited regime. The symbol distribution rule stated in Step 2 of Algorithm 1 satisfies the following properties:

- (a) Each encoded symbol $c_i$ is duplicated and stored in two different nodes.
- (b) Any choice of two storage nodes in different clusters shares one encoded symbol.
- (c) Any choice of two storage nodes in the same cluster shares $\chi$ encoded symbols.
- (d) Any failed node is exactly regenerated by retrieving a given number of symbols ($\chi$ or 1) from helper nodes.
- (e) When a data collector receives data from $k$ nodes, at least $M$ encoded symbols stored in nodes are accessible.

**Lemma 4.** Consider a $(n, k, L, \alpha, \chi, 1)$ clustered DSS and a cluster eavesdropper who compromises $L_c$ clusters. When using the distribution rule stated in Step 2 of Algorithm 1, the maximum number of encoded symbols observed by a cluster eavesdropper is $K = n_1^2 L_c (L - L_c + 1)$.

**Proof.** Recall that a coded symbol stored in both the compromised cluster and another cluster can be observed by a cluster eavesdropper. Note that any two storage nodes in different clusters share a single encoded symbol according to property (b) in Lemma 3. Thus, the number of coded symbols stored in a compromised cluster and another compromised cluster is obtained as $(L - L_c)^2 n_1^2$. Similarly, the number of coded symbols stored in a compromised cluster and a non-compromised cluster is obtained as $L_c (L - L_c + 1) n_1^2$. Combining these numbers results in $K = (L - L_c)^2 n_1^2 + L_c (L - L_c + 1) n_1^2 = n_1^2 L_c (L - L_c + 1)$.

According to property (e) in Lemma 3, a data collector observes at least $M$ among $\theta$ encoded symbols by contacting $k$ nodes. Similarly, a cluster eavesdropper observes at most $K$ out of $\theta$ encoded symbols according to Lemma 4. Thus, from the statement of Lemma 2, maximally $M - K$ message symbols can be stored in the system with perfect secrecy by applying a nested MDS code. By showing that $M - K$ is equal to $R$ in (2), Theorem 5 below proves that the code in Algorithm 1 achieves the upper bound on secrecy capacity.

**Theorem 5.** Consider a $(n, k, L, \alpha, \chi, 1)$ clustered DSS and a cluster eavesdropper who compromise $L_c$ clusters under condition $n_1 L_c < k$. The code constructed by Algorithm 1 achieves the secrecy capacity.

**Proof.** See Appendix A-D.

In addition, we summarize properties of the suggested capacity-achieving code in the following remark.

**Remark 5.** The code constructed by Algorithm 1 satisfies the following.

- (a) Exact regeneration: Any failed node is exactly regenerated by retrieving a given number of symbols ($\chi$ or 1) from helper nodes.
- (b) Data reconstruction: A data collector can reconstruct the message vector $s$ by contacting any $k$ among $n$ nodes.
- (c) Perfect secrecy: A cluster eavesdropper cannot obtain any information on the message $s$.

We provide an example of the suggested code in a $(6, 4, 3, 6, 2, 1)$ clustered DSS against a cluster eavesdropper with eavesdropping capability $L_c = 1$. Note that this regime is bandwidth-limited since $\alpha = \chi = 1$, $\alpha = 1$, $\chi = 1$, and $K = 8$, respectively. In order to store a message vector $s = [s_1, \ldots, s_8]$ of size $M - K$ with perfect secrecy, the encoded vector $c = [c_1, \ldots, c_8] = [\lfloor k \rfloor s] [G_K]^{-1} [G_S]^{-1}$ of size $\theta$ is generated in Step 1 of Algorithm 1 by applying a code in Lemma 2. Then, the coded symbols are distributed to nodes under the rule described in Step 2. Fig. 13 shows that how the coded symbols are distributed to nodes. Here we illustrate a special case where the cluster containing nodes $v_1$ and $v_2$ is compromised by a cluster eavesdropper. Now, we show that this code satisfies three properties stated in Remark 5.

- (a) **Exact regeneration:** Suppose a node $N(2, 1)$ fails. Then, the node can be exactly repaired by receiving one symbol from nodes $N(1, 1), N(1, 2), N(3, 1)$ and $N(3, 2)$ while $\chi = 2$ symbols from $N(2, 2)$.
- (b) **Data reconstruction:** A data collector observes at least $M = 16$ coded symbols by contacting $k = 4$ nodes. From property (e) in Lemma 3, the message $s$ can be reconstructed from $M$ symbols.
- (c) **Perfect secrecy:** Notice that $K = 8$ encoded symbols $\{c_2, c_3, c_4, c_5, c_6, c_7, c_8\}$ are at risk of being observed by a cluster eavesdropper. This is because those encoded symbols are stored in both the compromised cluster and another cluster, so they have chance to be transmitted from/to the compromised cluster when regenerating failed nodes. From Lemma 2, an eavesdropper cannot obtain any message from $K$ encoded symbols.
B. Code Construction on \( n_1L_c \geq k \)

Consider a \((n, k, L, \alpha, \beta_1 = \chi, \beta_c = 1)\) clustered DSS in the bandwidth-limited regime. We present a code construction to securely store data when sufficiently large clusters are compromised by a cluster eavesdropper, i.e., \( L_o \geq k/n_1 \). First we define

\[
\eta = L_X \binom{n_1}{2}
\]  

and a matrix \( W_{L,n_1} \) used in the code construction. Consider a complete \( L \)-partite graph \( G_{L,n_1} \) where each partition has \( n_1 \) vertices. An incidence matrix \( W_{L,n_1} \) of \( G_{L,n_1} \) is a \( n_1 L \times \binom{n_1}{2} \) matrix given by

\[
W_{L,n_1}(j,i) = \begin{cases} 
1, & \text{if } i^{th} \text{ edge is connected to } j^{th} \text{ node} \\
0, & \text{otherwise.} \end{cases}
\]

An example of incidence matrix \( W_{L,n_1} \) when \( L = 3 \) and \( n_1 = 2 \) is provided in Fig. 14.

Using these definitions, we provide an explicit code construction in Algorithm 2, which store \( R \) message symbols with perfect secrecy.

**Algorithm 2** Secure code construction on \( n_1L_c \geq k \)

**Input:** System parameters \( n, k, \alpha, L, \chi \)

Message vector \( s = [s_1, \ldots, s_R] \in \mathbb{F}_s^{1 \times R} \) where \( R \) is obtained from system parameters as in (2)

**Output:** Symbols stored on each node \( N \)

**Step 1.** Generate encoded symbols \( \{c_1, \ldots, c_\eta\} \) and dummy symbols \( \{c_{\eta+1}, \ldots, c_k\} \)

Generate an encoded vector \( c \triangleq [c_1, \ldots, c_\eta] = sG \) where \( G \) is a generator matrix of a \((\eta, R)\) MDS code.

**Step 2.** Distribute encoded symbols \( \{c_1, \ldots, c_\eta\} \) to nodes under the following rules:

- For \( t \in [\chi] \), node \( N(l,j) \) stores 
  \( c_{(t-1)\chi+(t-1)(\chi+1)+i} \) if and only if \( V_n(j,i) = 1 \).
- Node \( N(l,j) \) stores symbol \( c_{\eta+i} \) if and only if \( W_{L,n_1}(j,i) = 1 \).

Theorem 6 below shows that the code in Algorithm 2 achieves the secrecy capacity of the system with \( L_c \geq k/n_1 \), which can be proved as follows. First, a cluster eavesdropper cannot access any of \( \eta \) encoded symbols \( \{c_1, \ldots, c_\eta\} \), which is proved in Lemma 5. Next, a data collector can access at least \( R \) out of \( \eta \) encoded symbols \( \{c_1, \ldots, c_\eta\} \), which is proved in Lemma 6. Note that a data collector can obtain message vector \( s \) of size \( R \) by decoding \((\eta, R)\) MDS codes defined in Step 1 of Algorithm 2. Combining these results, we conclude that a message vector \( s \) of size \( R \) can be stored in the system with perfect secrecy, which completes the proof of Theorem 6. The formal statements of Lemmas 5, 6 and Theorem 6 are provided as follows.

**Lemma 5.** In Algorithm 2, each encoded symbol in \( \{c_1, \ldots, c_\eta\} \) is stored on two distinct nodes in a single cluster. Therefore, such symbols are stored with perfect secrecy against a cluster eavesdropper since they cannot be eavesdropped during any repair events.

**Proof.** When distributing an encoded symbol \( c_t \) such that \( t \leq \eta \), we use the first rule in Step 2. From the symbol allocation rule, it is directly obtained that symbol \( c_t \) is stored on nodes in \( t^{th} \) cluster where \( I = \left( x_1^{(t-1)}) \right) \).

**Lemma 6.** Consider \( \theta \) symbols \( \{c_1, \ldots, c_\theta\} \) are distributed to \( \eta \) nodes following the rule specified at step 2 in Algorithm 2. Then, a data collector observes at least \( R \) symbols in \( \{c_1, \ldots, c_\eta\} \) by contacting \( k \) out of \( \eta \) nodes.

**Proof.** See Appendix C-A.

**Theorem 6.** Consider a \((n, k, \alpha, \chi, 1)\) clustered DSS in the bandwidth-limited regime and a cluster eavesdropper who compromise \( L_c \) clusters under condition \( n_1L_c \geq k \). The code construction described in Algorithm 2 achieves the secrecy capacity.

As an example, Fig. 15 represents the distribution rule of Algorithm 2 to store \( \theta = 18 \) symbols \( \{c_1, \ldots, c_{18}\} \) in a \((6, 4, 3, 6, 2, 1)\) clustered DSS. You may easily find that the first \( \eta = 6 \) coded symbols \( \{c_1, \ldots, c_6\} \) are stored in a single cluster, thus they cannot be observed by a cluster eavesdropper. Finally, we summarize the properties of the suggested code in Remark 6.

**Remark 6.** The code constructed by Algorithm 2 satisfies the following.

(a) Exact regeneration: Any failed node is exactly regenerated by retrieving a given number of symbols (\( \chi \) or 1) from helper nodes.

(b) Data reconstruction: A data collector can reconstruct the message vector \( s \) by contacting any \( k \) among \( \eta \) nodes.

(c) Perfect secrecy: A cluster eavesdropper cannot obtain any information on the message \( s \).

**VII. CONCLUSION**

We have presented a cluster eavesdropper model for practical clustered DSSs. Our model covers scenarios where data is distributed to nodes in multiple clusters and a cluster eavesdropper observes links from cluster to cluster. Based on an analysis of information flow graph, we derived the maximum amount of data stored in the system with perfect secrecy. Using
the results, we obtained feasible points of resources - node storage size and repair bandwidth - to securely store data with given size. We provide some key insights into how to best allocate resources in order to secure data. Furthermore, an explicit coding scheme to achieve the theoretical upper bound of secrecy capacity is suggested for the practically important bandwidth-limited regime. In realistic scenarios where data fragments are dispersed in multiple clusters, security of the system can be evaluated using our results. The present paper only deals maximal helper nodes in the cluster system. Due to potential latency issues and multiple node failures of storage nodes, expanding our results to some fixed number of helper nodes would be meaningful future work.

**APPENDIX A**

**PROOF OF THEOREMS**

A. Proof of Theorem 1

Before proving the theorem, we introduce information flow graph, an useful mathematic tool used in [20] to analyze dynamic node failure events and repair processes. Information flow graph is a directed acyclic graph consisting of three types of nodes: the source node $S$, the data collector node $DC$ and the storage nodes $v_i$, where $i$ is the node label. In the graph, storing encoded data into $n$ storage nodes is represented by $n$ directed edges from the source $S$ to $n$ storage nodes. Each storage node $v_i$ consists of the input node $v_i^{in}$ and the output node $v_i^{out}$, connected via a link whose capacity is $\alpha$, the node storage size. In the case of node failures, a replacement node joins the system by retrieving data from remaining nodes, each contributes to given amount ($\beta_1$ or $\beta_2$) of edge capacity. Finally, the retrieval process from the data collector $DC$ is illustrated by the $k$ directed edges from the $v_i^{out}$ nodes to $DC$. An example of information flow graph in a $(4, 3, 2, \alpha, \beta_1, \beta_2)$ clustered DSS is shown in Fig. 16.

Due to the dynamic node failure and repair processes in a DSS, the number of possible flow graphs can be infinite. We define $\mathcal{G}(n,k,L,\alpha,\beta_1,\beta_2)$ as the set of all possible flow graphs in a $(n, k, L, \alpha, \beta_1, \beta_2)$ clustered DSS. An information flow graph $G \in \mathcal{G}(n,k,L,\alpha,\beta_1,\beta_2)$ can be expressed as $G = (V(G), E(G))$ where $V(G)$ is the set of all vertices and $E(G)$ is the set of all directed edges in $G$. For a given $G \in \mathcal{G}(n,k,L,\alpha,\beta_1,\beta_2)$, a cut is defined as a set $c \subset E(G)$ of edges which satisfies the following: every directed path from the source to the data collector must pass at least one edge in $c$. We define a cut-value $w(c)$ as the sum of edge capacities for the edges in $c$. Note that a cut $c$ can be alternatively expressed as a pair of two exclusive vertex sets $(U, \bar{U})$ where $U \subset V(G)$ and $\bar{U} = V(G)\setminus U$ (i.e., $V$ exclusive of $U$). In the $(U, \bar{U})$ representation, the elements of a cut are determined by the set of all directed edges from $U$ to $\bar{U}$.

As in Fig. 17, data revealed to a cluster eavesdropper are fully characterized by a set of directed edges $E_e$ in the information flow graph. For a given flow graph $G \in \mathcal{G}(n,k,L,\alpha,\beta_1,\beta_2)$, the data collector node $DC$ can be evaluated using our results. The present paper only deals maximal helper nodes in the cluster system. Due to potential latency issues and multiple node failures of storage nodes, expanding our results to some fixed number of helper nodes would be meaningful future work.
G(n, k, L, α, β₁, β₂) and a set of compromised data links E_c, the upper bound on secrecy capacity of a (n, k, L, α, β₁, β₂) clustered DSS can be obtained as

\[ \text{Secrecy capacity} \leq w(c|E_c) \]

where \( w(c|E_c) \) is the sum of edge capacities of a cut without compromised edges. It is intuitive to see that the system cannot store data more than \( w(c|E_c) \) with perfect secrecy, since DC can retrieve a maximum amount of secure data by ignoring all data from the compromised links.

Now we prove our main theorem by specifying an information flow graph \( G \in G(n, k, L, \alpha, \beta_1, \beta_2) \), a cut \( c \in E(G) \) and a set of compromised links \( E_c \in E(G) \). Here, we use two dimensional representation of nodes as in Fig. 11. Consider a \( (n, k, L, \alpha, \beta_1, \beta_2) \) clustered DSS and a cluster eavesdropper who compromises \( L_c \) clusters. The source node stores data in nodes \( v_i, i \in [n] \), where the location of each node is expressed as

\[ v_i = N(x_i, y_i), \quad t \in [n] \]

where

\[ x_i = \lfloor i/n_1 \rfloor, \]
\[ y_i = \text{mod}(i - 1, n_1) + 1. \]

Without a loss of generality, we assume that \( 1^{\text{st}}, \ldots, L_c^{\text{th}} \) clusters are compromised by a cluster eavesdropper. First, we describe obtaining a flow graph \( G \) and a set of compromised data links \( E_c \). Consider \( k \) successive node failures and repair events. \( k \) storage nodes among \( \{v_i\}_{i=1}^n \) are successively failed and regenerated to \( v_{n+1}, \ldots, v_{n+k} \) where node \( v_{n+i}, i \in [k] \) indicates \( i^{\text{th}} \) regenerated node. The data collector DC contacts \( k \) regenerated nodes \( v_{n+1}, \ldots, v_{n+k} \) to retrieve data. The two dimensional location of \( i^{\text{th}} \) failed and corresponding regenerated node is specified as \( N(l_i, j_i) \) where

\[ l_i = \begin{cases} \lfloor i/n_1 \rfloor, & \text{if } i \leq n_1 L_c \\ (i - n_1 L_c) - \sum_{m=1}^{n_1-1} (g_m - L_c) + L_c, & \text{otherwise} \end{cases} \]

\[ j_i = \begin{cases} \text{mod}(i - 1, n_1) + 1, & \text{if } i \leq n_1 L_c \\ \min\{v : \sum_{m=1}^{n} (g_m - L_c) \geq i - n_1 L_c\}, & \text{otherwise} \end{cases} \]

and

\[ g_m = \begin{cases} q + 1, & \text{if } m \leq r \\ q, & \text{otherwise}. \end{cases} \]

We provide an example to obtain the location of \( k \) regenerated nodes in Fig. 19. A node with an index \( i \) in the figure stands for the location of \( i^{\text{th}} \) regenerated node, \( v_{n+i} \). For each index \( i \leq n_1 L_c \), \( v_{n+i} \) is located in the \( [i/n_1]^{\text{th}} \) compromised clusters. Also, for an index \( i > n_1 L_c \), note that selection that remaining \( k - n_1 L_c \) failed nodes follows the identical selection rule defined in Definition 3, 4 of [6]. The procedure provides an unique information flow graph \( G \) and the set of compromised links \( E_c \) by a cluster eavesdropper.

Finally, we specify a cut \( c = (U, \bar{U}) \) of \( G \). Notice that there exists \( 2 + 2(n+k) \) vertices in \( G \) which are the source node \( S \), the data collector node DC and the \( (n+k) \) input and output storage nodes \( \{v_i^{in}, v_i^{out}\}_{i=1}^{n+k} \). Define \( w^*_i \) as the sum of non-compromised incoming edge capacities from \( \{v_i^{out}\}_{i=1}^n \) to \( i^{\text{th}} \) regenerated input node \( v_{n+i} \), and a set \( T \) as

\[ T = \{i \in [k] : w^*_i \geq \alpha\}. \]

We split the set of vertices into two disjoint groups \( U \) and \( \bar{U} \) to specify a cut \( c \) as follows.

\[ U = \{S, v_i^{in}, v_i^{out}, v_{n+i} : i \in [n], t \in T\} \]
\[ \bar{U} = \{DC, v_i^{in}, v_i^{out} : i \in [k], t \notin T\} \]

Then a cut \( c = (U, \bar{U}) \) satisfies

\[ w(c|E_c) = \sum_{i=1}^{k} \min\{\alpha, w^*_i\}. \]

Obtaining a graph \( G \), compromised edges \( E_c \) and cut \( c \) is illustrated in Fig. 18.

What remains is obtaining a value of \( w(c|E_c) \) with a function of system parameters. For convenience, we write \( C^{(1)} \) and \( C^{(2)} \) as follows, with an understanding that \( w(c|E_c) = C^{(1)} + C^{(2)} \). If \( k \leq n_1 L_c \), we regard as \( C^{(2)} = 0 \).

\[ C^{(1)} = \sum_{i=1}^{\min\{n_1 L_c, k\}} \min\{\alpha, w^*_i\} \]
\[ C^{(2)} = \sum_{i=n_1 L_c+1}^{k} \min\{\alpha, w^*_i\} \]

First, we specify \( w^*_i \) for \( i \leq n_1 L_c \). Consider \( w^*_i \) when \( L_c \geq 1 \). Notice that the first \( n_1 \) replacement nodes \( v_{n+i}, i \in [n] \) are located in the first cluster, which is compromised by a cluster eavesdropper. When the first replacement node \( v_{n+1} \) connects to the remaining \( n-1 \) helper nodes to retrieve data, every link coming from other clusters is observed by a cluster eavesdropper. Therefore, links from nodes \( \{v_i^{out}\}_{i=2}^{n_1} \) to \( v_{n+1}^{in} \) (each node contributes \( \beta_1 \)) are the only secure links, while links from \( \{v_i^{out}\}_{i=n_1+1}^{n+k} \) (each node contributes \( \beta_2 \)) are not. Thus, \( w^*_1 = (n_1 - 1)\beta_1 \). Next, consider \( w^*_2 \) where the second replacement node \( v_{n+2} \) is located in the first cluster.
Substituting (A.6) to (A.4) explicitly gives $C_s^{(1)}$. For $k \leq n_1 L_c$,

$$C_s^{(1)} = \sum_{i=1}^{n_{1q}} \min \{ \alpha, \text{mod}(n_1 - t, n_1) \beta I \} + \sum_{i=n_1+1}^{k} \min \{ \alpha, \text{mod}(n_1 - t, n_1) \beta I \}
= q \sum_{i=1}^{n_1} \min \{ \alpha, (n_1 - i) \beta I \} + \sum_{i=1}^{n_1} \min \{ \alpha, (n_1 - i) \beta I \}.$$  

and for $k > n_1 L_c$,

$$C_s^{(1)} = \sum_{i=1}^{n_{1L_c}} \min \{ \alpha, \text{mod}(n_1 - t, n_1) \beta I \}
= L_c \sum_{i=1}^{n_1} \min \{ \alpha, (n_1 - i) \beta I \}.$$  

Now we obtain $C_s^{(2)}$ for $k > n_1 L_c$. For given $i$ where $i \in [k \setminus [n_1 L_c]]$, the $i^{th}$ replacement node $v_{n+i}^{\text{in}}$ is connected to $\{v_{n+i+1}^{\text{in}}, \ldots, v_{n+q}^{\text{in}}\}$ in set $U$ and $\{v_{n+i+1}^{\text{out}}, \ldots, v_{n+q}^{\text{out}}\}$ in set $\bar{U}$. Notice that $\{v_{n+i+1}^{\text{in}}, \ldots, v_{n+q}^{\text{in}}\}$ in $U$ are located in clusters not observed by a cluster eavesdropper. Therefore, incoming edges from $U$ to $v_{n+i}^{\text{in}}$ are secure links. An important observation is that selection of $i^{th}$ failed nodes $i \in [k \setminus [n_1 L_c]]$ among remaining $(n_1 L_c)$ nodes follows an ordering rule suggested in Definition 3, 4 of [6]. Thus, value of $C_s^{(2)}$ is the cut-value (which turns out to capacity) in a clustered DSS without nodes in the compromised clusters. Therefore, $C_s^{(2)}$ can be rewritten as

$$C_s^{(2)} = C_r$$

where $C_r$ is capacity of the $(n - n_1 L_c, k - n_1 L_c, L - L_c, \alpha, \beta I, \beta I)$ clustered DSS. Note that repaired bandwidth of a reduced clustered DSS is $(n_1 - 1) \beta I + (n - n_1 L_c - n_1) \beta I$. Since the feasible node storage size should be less than or equal to repair bandwidth, $C_r$ can be rewritten as a capacity of $(n - n_1 L_c, k - n_1 L_c, L - L_c, \alpha', \beta I, \beta I)$ clustered DSS where $\alpha' = \min \{ \alpha, (n_1 - 1) \beta I + (n - n_1 L_c - n_1) \beta I \}$. An upper bound of secrecy capacity $w(c \setminus E_c)$ is obtained by combining $C_s^{(1)}$ and $C_s^{(2)}$, which completes the proof.

B. Proof of Theorem 3

Case 1) $1 \leq L_c < k/n_1$: From (B.8) and (B.10), we know

$$a_{k'+n_1-1} + b_{k'+n_1-1} u_{k'+n_1-1} = k - L_c$$

and

$$a_r + b_r \bar{\gamma} = \{a_r + n_1 - 1 + (n - n_1)\epsilon\} \alpha.$$
From (B.6), notice that
\[ C_s^U(\alpha, \gamma) = \begin{cases} a_t \alpha + b_t \gamma, & \gamma \in (u_t \alpha, u_{t+1} \alpha], \\ 0, & \gamma = 0 \end{cases} \]
is a continuous increasing function of \( \gamma \) and \( \{b_t\} \) is a decreasing sequence of \( t \) where \( a_t, b_t \) and \( \tau \) are defined in Section IV-A.

Therefore, the following inequality holds
\[ a_t \alpha + b_t \gamma < a_t \alpha + b_t \gamma = (k - L_c) \alpha, \]
which results in
\[ \frac{M_s}{k - L_c} < \frac{M_s}{k - \tau} \leq \frac{M_s}{k - \gamma}. \]
Thus, every \((\alpha, \gamma)\) pair with \( \alpha < \frac{M_s}{k - L_c} \) is infeasible for all \( \gamma \in [0, \infty) \) from Corollary 2.

Case 2) \( L_c \geq k/n_1 \): Since \( q = \lfloor k/n_1 \rfloor \), the following inequality trivially holds.
\[ \frac{M_s}{k - L_c} \leq \frac{M_s}{k - q} \]
Therefore, \( \alpha < \frac{M_s}{k - L_c} \) is infeasible for all \( \gamma \in [0, \infty) \) from Corollary 3.

C. Proof of Theorem 4

Define \( C_s^a \) as below, which is a constant with respect to \( L_c \).
\[ C_s^a = \frac{\gamma}{2} \frac{(g + 1)(n_I - 1)n_I - (n_I - r - 1)(n_I - r)}{(n_I - 1)} \]
Combining with Proposition 2 which says
\[ C_s^b \geq (L_c - (L - 1/2))^2 + c - a(L - 1/2)^2, \]
an intersection point \( L_c^*=C_s^a \) is expressed as
\[ L_c^* = (L - 1/2) - \sqrt{(L - 1/2)^2 - \frac{c - C_s^a}{a}}. \]
Note that parameters \( a \) and \( c \) are function of \( \epsilon \). What remains is to prove the following equality
\[ \frac{c - C_s^a}{a} = \frac{w}{n^2(n_I - 1)}, \]
which implicitly shows that intersection point \( L_c^* \) is independent of \( \epsilon \). This is shown in (A.8), which completes the proof.

D. Proof of Theorem 5

A message vector with size \( M - K \) can be stored with perfect secrecy in a \((n, k, L, (n_I - 1) \chi + (n - n_I), \chi, 1)\) clustered DSS by using Algorithm 1. All the remaining to prove is that \( M - K \) achieves the upper bound \( R \) of secrecy capacity. In (2), \( R \) is expressed as
\[ R = \chi L_c \frac{n_I(n_I - 1)}{2} + C_r, \]
where \( C_r \) is the capacity of a reduced \((n', k', L', \alpha', \chi, 1) = (n - n_I L_c, k - n_I L_c, L - L_c, (n_I - 1) \chi + (n' - n_I), \chi, 1)\)
clustered DSS. We state the following remark, which explicitly expresses capacity in the bandwidth-limited regime.

Remark 7 (Proposition 2 in [18]). Capacity \( C \) of a \((n, k, L, \alpha, \chi, 1)\) clustered DSS in the bandwidth-limited regime is
\[ C = k\alpha - \frac{1}{2}(\chi - 1)(q/n_1^2 + r^2 - k) - \frac{k(k - 1)}{2}, \quad (A.7) \]
where \( q = \lfloor k/n_1 \rfloor \) and \( r = \text{mod}(k, n_1) \).

From Lemma 7, the capacity of a reduced clustered DSS \( C_r \) is expressed as
\[ C_r = k'\alpha - \frac{1}{2}(\chi - 1)(q'n_1^2 + r'^2 - k') - \frac{k'(k' - 1)}{2}, \]
where \( q' = \lfloor k'/n_1 \rfloor \) and \( r' = \text{mod}(k', n_1) \). We can numerically derive an equality
\[ M - K = R \]
with a simple calculation, where \( M \) and \( K \) are defined in (16) and (17), respectively.

Appendix B
Proof of Corollaries

A. Proof of Corollary 2

Consider a \((n, k, L, \alpha, \beta_I, \beta_c)\) clustered DSS. Using the expressions of \( \epsilon \) and \( \gamma \) given in (3) and (4), the upper bound of secrecy capacity is expressed as
\[ C_s^U(\alpha, \gamma) = L_c \sum_{i=1}^{n_I} \min \{ \alpha, \frac{(n_I - i)\gamma}{n_I - 1 + (n - n_I)\epsilon} \} + C_r \quad (B.1) \]
where \( C_r \) is the capacity of a \((n', k', L', \alpha', \beta_I, \beta_c)\) clustered DSS.

From the proof of Theorem 1 in [6], note that capacity of a clustered distributed storage can be expressed as
\[ C_r = \sum_{i=1}^{n'} \min \{ \alpha, \frac{(n' - n_I - i + h_i)\epsilon + (n_I - h_i)}{n_I - 1} \}, \]
where
\[ n' = n - n_I L_c, \]
\[ k' = k - n_I L_c, \]
\[ h_i = n - n_I L_c, \]
\[ h_i = \min \{ g_i - L_c \geq t, t \in [k'] \}. \]
Define sequences \( \{x_t\}, \{y_t\}, \{z_t\} \) and \( \{u_t\} \) as in Section IV-A. Then, the upper bound in (B.1) is simply reduced to
\[ C_s^U(\alpha, \gamma) = L_c \sum_{i=1}^{n_I} \min \{ \frac{\alpha}{x_t}, \frac{\gamma}{y_t} \} + \sum_{i=1}^{k'} \min \{ \alpha, \frac{\gamma}{y_t} \}, \quad (B.2) \]
which is a continuous function of \( \gamma \).

Remark 8. \( \{x_t\}, \{y_t\} \) and \( \{u_t\} \) are non-negative increasing sequences, while \( \{z_t\} \) is a non-negative decreasing sequence.

Proof. It is trivial to show \( \{x_t\} \) is a non-negative increasing sequence and \( \{u_t\} \) is an increasing sequence. \( \{z_t\} \) is a non-
Therefore, negative sequence since it satisfies
\[ z_t = (n' - n_t - t + h_t)\epsilon + (n_t - h_t) \]
\[ \geq (n' - n_t - t + h_t)\epsilon + (n_t - h_t) \epsilon \]
\[ = (n' - t)\epsilon > 0 \]
for all \( t \in [k'] \). It follows that \( \{y_t\} \) and \( \{u_t\} \) are non-negative sequences.

By definition, note that \( h_t \) satisfies
\[ h_{t+1} = \begin{cases} h_t + 1, & t \in T \\ h_t, & t \in [k' - 1] \backslash T \end{cases} \]  \hspace{1cm} (B.3)
where
\[ T = \{ \sum_{i=1}^{l} (g_i - L_c) : l \in [n_t] \}. \]
Combining with a definition of sequence \( \{z_t\} \),
\[ z_{t+1} - z_t = \begin{cases} -1, & t \in T \\ -\epsilon, & t \in [k' - 1] \backslash T. \end{cases} \]
Thus, \( \{z_t\} \) is a decreasing sequence, which implies \( \{y_t\} \) is an increasing sequence.

Next, we specify a valid region of repair bandwidth \( \gamma \). Since intra-cluster bandwidth per node \( \beta_t \) cannot exceed a node storage size \( \alpha \), inequality \( \beta_t \leq \alpha \) is trivially satisfied. Therefore, \( \gamma \) is bounded by
\[ \gamma = (n_t - 1)\beta_t + (n - n_t)\beta_e \]
\[ = (n_t - 1 + (n - n_t)\epsilon)\beta_t \leq \{n_t - 1 + (n - n_t)\epsilon\} \alpha. \]
and we define threshold \( \bar{\gamma} \) of repair bandwidth as
\[ \bar{\gamma} \triangleq \{n_t - 1 + (n - n_t)\epsilon\} \alpha. \]  \hspace{1cm} (B.4)

Here, notice that the sequence \( \{x_t\} \) satisfies
\[ x_t\alpha \leq \bar{\gamma}, \quad \forall t \in [n_t - 1]. \]
The rest of the proof depends on the range of \( \epsilon \).

**Case 1** \( \frac{1}{\alpha} \leq \epsilon \leq 1 \): Each element of sequence \( \{z_t\} \)
satisfies
\[ z_t \geq 2k' \]
\[ \geq (n' - n_t - k' + h_t)\epsilon + (n_t - h_t) \epsilon \]
\[ = (n - k)\epsilon \geq 1. \]  \hspace{1cm} (B.5)
Therefore, sequence \( \{y_t\} \) satisfies
\[ y_t\alpha \leq \{n_t - 1 + (n - n_t)\epsilon\} \alpha = \bar{\gamma}, \quad \forall t \in [k']. \]

For given \( \gamma \leq \bar{\gamma} \), there exist unique \( p, q \) which satisfy
\[ x_p\alpha \leq \gamma < x_{p+1}\alpha, \quad p \in [n_t - 1] \cup \{0\}, \]
\[ y_q\alpha \leq \gamma < y_{q+1}\alpha, \quad q \in [k'] \cup \{0\}. \]

Let \( p, q \) be an integer that satisfying the above inequalities. Then, the upper bound \( C^U_s \) (B.2) is simplified to
\[ C^U_s(\alpha, \gamma) = (L_c + p + q)\alpha + (r_p + s_q)\gamma, \quad \gamma \leq \bar{\gamma} \]
where \( r_p \) and \( s_q \) are defined in Section IV-A.

We can express \( C^U_s \) on different ranges of \( \gamma \),
\[ C^U_s(\alpha, \gamma) = \begin{cases} a_t\alpha + b_t\gamma, & \gamma \in (u_t\alpha, u_{t+1}\alpha], \\ 0, & \gamma = 0 \end{cases} \]
\[ \begin{cases} t \in [k' + n_t - 2] \cup \{0\} \\ 0, \end{cases} \]  \hspace{1cm} (B.6)
where \( a_t, b_t \) are defined in Section IV-A. Notice that \( u_{k' + n_t - 1} = \bar{\gamma} \), since \( u_{k' + n_t - 1} = \bar{\gamma} \) is the second largest element of \( \{x_i\}_{i=1}^{n_t} \). Let \( x_i, y_j \) satisfy
\[ x_i, y_j \leq \bar{\gamma}, \quad i \in [n_t - 1], j \in [k'], \]
\[ x_{n_t - 1} = \bar{\gamma}, \quad x_{n_t} = \infty. \]

**Remark 9.** \( \{b_t\} \) is a non-negative decreasing sequence.

**Proof.** First, every element of sequence \( \{b_t\} \) is nonnegative since \( r_p \) and \( s_q \) are summations of non-negative elements. In
order to prove \( \{b_t\} \) is a decreasing sequence, let
\[
\begin{align*}
    b_t &= r_p' + s_q', \\
    b_{t+1} &= r_p'' + s_q''.
\end{align*}
\]
Then, inequalities
\[
p' \leq p'', \quad q' \leq q''
\]
are satisfied. Combining with a definition of \( r_p \) and \( s_q \) in Section IV-A,
\[
b_t - b_{t+1} = \sum_{t=p'+1}^{p''} \frac{1}{x_t} + \sum_{t=q'+1}^{q''} \frac{1}{y_t} \geq 0.
\]
Thus, \( \{b_t\} \) is a non-negative decreasing sequence.

Since \( C_s^U \) is a continuous function of \( \gamma \) and \( b_t \) is non-negative decreasing sequences, \( C_s^U \) is shown in Fig. 20 as a function of \( \gamma \). Notice that \( C_s^U(\alpha, \gamma) < M_s \) if and only if \( \gamma < \gamma^*(\alpha) \) from the figure. Consider a \((\alpha, \gamma)\) pair with \( \gamma < \gamma^*(\alpha) \); Since \( C_s^U(\alpha, \gamma) \) is a monotonic increasing function of \( \gamma \),
\[
C_s(\alpha, \gamma) \leq C_s^U(\alpha, \gamma) < M_s
\]
holds. Thus, it is impossible to securely store \( M_s \) with given resources \((\alpha, \gamma)\).

Using (B.6), the threshold value \( \gamma^*(\alpha) \) is expressed as
\[
\gamma^*(\alpha) = \begin{cases} 0, & M_s = 0 \\ \frac{M_s-a_{t+1}}{b_t}, & M_s \in (E_t, E_{t+1}], \\ \infty, & M_s \in (E_{t+n_t+k'-1}, \infty) \end{cases}
\]
where
\[
E_t = C_s^U(\alpha, u_t\alpha) = (a_t + b_tu_t)\alpha.
\]
(B.7)
\( \gamma^*(\alpha) \) is alternatively expressed as a function of \( \alpha \) as in (7).

Notice that the following equality holds for \( t = k + n_t - 1 \).
\[
a_{k'+n_t-1} + b_{k'+n_t-1}u_{k'+n_t-1} = L_e(n_t - 1) + k' = k - L_e
\]
(B.8)

Case 2) \( 0 < \epsilon < \frac{1}{n-k}, k' < n_t \): Under the constraint \( k' < n_t \), inequality \( h_{k'} \leq n_t - 1 \) holds. Therefore, each

element of \( \{z_t\}_{t=0}^k \) satisfies
\[
z_t \geq z_{k'} \geq (n_t - h_t) \geq 1.
\]
The remaining proof is identical to the case \( \frac{1}{n-k} \leq \epsilon < 1 \).

Case 3) \( 0 < \epsilon < \frac{1}{n-k}, k' \geq n_t \): Notice that \( h_{k'} = n_t \) holds for \( k' \geq n_t \). Since \( \{z_t\} \) is a decreasing sequence and \( z_{k'} = (n - k)\epsilon < 1 \) there exists an unique \( \tau \in [k' + n_t - 2] \cup \{0\} \) such that
\[
u_{\tau} \alpha < \bar{\gamma} \leq u_{\tau + 1} \alpha.
\]
From an analysis similar to the \( \frac{1}{n-k} < \epsilon < 1 \) case, we obtain
\[
C_s^U(\alpha, \gamma) = \begin{cases} 0, & \gamma = 0 \\ a_{\tau} + b_{\tau} \gamma, & \gamma \in (u_{\tau} + u_{\tau + 1} \alpha], \\ a_{\tau} \alpha + b_{\tau} \gamma, & \gamma \in (u_{\tau} + \bar{\gamma}) \end{cases}
\]
for \( \tau \) such that
\[
u_{\tau} \alpha < \gamma \leq u_{\tau + 1} \alpha, \quad \tau \in [k' + n_t - 2] \cup \{0\}
\]
where parameters are given in Section IV-A.

Fig. 21 illustrates (B.9) as a function of \( \gamma \). From the figure, \( C_s^U(\alpha, \gamma) < M_s \) holds if and only if \( \gamma < \gamma^*(\alpha) \). Threshold \( \gamma^*(\alpha) \) is obtained as
\[
\gamma^*(\alpha) = \begin{cases} 0, & M_s = 0 \\ \frac{M_s-a_{t+1}}{b_t}, & M_s \in (E_t, E_{t+1}], \\ \infty, & M_s \in (E_{t+n_t+k'}, \infty) \end{cases}
\]
where \( E_t \) is defined in (B.7) and \( \bar{E} \) is expressed as
\[
\bar{E} = C_s^U(\alpha, \bar{\gamma}) = a_{\tau} \alpha + \bar{b}_{\tau} \bar{\gamma} = (a_{\tau} + n_t - 1 + (n - n_t)\epsilon) \alpha.
\]
Threshold \( \gamma^*(\alpha) \) is alternatively expressed as a function of \( \alpha \) as in (9).
From (B.11), the upper bound expressed as
\[
\gamma < \gamma^* (\alpha) = \min \left\{ \frac{\alpha_n}{n_1 - 1}, \frac{\alpha_{n_1-1}}{n_1-2} \right\}
\]
for all \( n \). The upper bound is obtained by
\[
\gamma \leq \gamma^* (\alpha, \gamma)
\]
where (B.13) comes from the following remark.

Remark 10. Capacity \( C \) of a \( (n, k, L, \alpha, \beta_1, \beta_c) \) clustered DSS in the bandwidth-limited regime is
\[
\tilde{C} = k \alpha - \frac{1}{2} (\beta_1 - \beta_c) (q n_1^2 + r^2 - k) - \beta_c \frac{k(k-1)}{2}
\]
where \( q = \lfloor k/n_1 \rfloor \) and \( r = \text{mod}(k,n_1) \).

Proof. The capacity of a \( (n, k, L, \alpha, \gamma, 1) \) clustered DSS in the bandwidth-limited regime is explicitly obtained in Remark 7. Since the capacity is a linear function of \( \gamma \) with a given \( \epsilon \) in this regime, \( C \) is obtained as \( \beta_c \) times a capacity \( C' \) of \( (n, k, L, \alpha, \beta_c, \beta_1, \beta_c) \) clustered DSS where
\[
C' = k \alpha - \frac{1}{2} (\beta_1 - \beta_c - 1) (q n_1^2 + r^2 - k) - \frac{k(k-1)}{2}
\]

Case 2) \( L_c \geq k/n_1 \): From Theorem 2, secrecy capacity is independent of \( L_c \) when \( L_c \geq k/n_1 \). Thus, \( \Delta C_s^{BL} (L_c) = 0 \).

D. Proof of Corollary 7

The exact expression of intersection point \( L_c^* \) is given in (14). We write
\[
\frac{\omega}{n_1^2 (n_1 - 1)} = \frac{q(n - q n_1 - 2 r)}{n_1} + \frac{r (r - 1) (L - 1)}{n_1 (n_1 - 1)}
\]

C. Proof of Corollary 5

Case 1) \( L_c < k/n_1 \): From Theorem 2, \( C_s^{BL} \) is expressed as
\[
C_s^{BL} = L_c \beta_1 \frac{n_1 (n_1 - 1)}{2} + C_s^{BL},
\]
where \( C_s^{BL} \) is capacity of a reduced \( (n', k', L', \alpha', \beta_1, \beta_c) \) clustered DSS. Thus, \( C_s^{BL} \) can be rewritten as
\[
\Delta C_s^{BL} (L_c) = - C_s^{BL} (L_c) + C_s^{BL} (L_c + 1)
\]
\[
= \beta_1 \frac{n_1 (n_1 - 1)}{2} + C_s^{BL} (L_c + 1) - C_s^{BL} (L_c)
\]
\[
= - \beta_c n_1^2 (L - L_c - 1)
\]

Graph 22: \( C_s^U (\alpha, \gamma) \) as a function of \( \gamma \) when \( L_c \geq k/n_1 \)

\[
\gamma^* (\alpha) = \min \left\{ \frac{\alpha_n}{n_1 - 1}, \frac{\alpha_{n_1-1}}{n_1-2} \right\}
\]
and
\[
E_t = C_s^U (\alpha, x_t) = (c_t + d_t x_t) \alpha.
\]
Alternatively, \( \gamma^* (\alpha) \) is expressed as (10).
Notice that \( \Delta_1 = o(L) \). Therefore,
\[
L^*_c \approx L - \sqrt{L^2 - \frac{\omega}{n_1^2(n_1 - 1)}} \\
= L(1 - \sqrt{1 - \mathcal{R}(1 - \mathcal{R})} + o(1/L)) \\
\approx L(1 - \sqrt{1 - \mathcal{R}(1 - \mathcal{R})}).
\]

**Appendix C**

**Proof of Lemmas**

**A. Proof of Lemma 6**

Before proving Lemma 6, we provide Remark 11, which says that symbol distribution rules of Algorithms 1 and 2 are identical by properly relabeling the indices of coded symbols.

**Remark 11.** By properly relabeling indices of encoded symbols \( \{c_1, \ldots, c_\eta\} \), the distribution rule of encoded symbols stated in Algorithm 2 is identical to the rule stated in Algorithm 1. Thus, the distribution rule of Algorithm 2 satisfies all properties provided in Lemma 3.

**Proof.** Recall the encoded symbol distribution rules:

**Step 2 of Code 1.** Distribute encoded symbols \( \{c_1, \ldots, c_\eta\} \) to nodes under the following rules (Construction 2 in [18]):

- First rule: Node \( N(l, j) \) stores a symbol \( c_{i_1} \) if and only if \( V_n(n_1(l - 1) + j, i_1) = 1 \).
- Second rule: For \( t \in [\chi - 1] \), node \( N(l, j) \) stores a symbol \( c_{(t+1)(\chi-1)+j} \) if and only if \( V_{n_1}(j, i_2) = 1 \).

**Step 2 of Code 2.** Distribute encoded symbols \( \{c_1, \ldots, c_\eta\} \) to nodes under the following rules:

- First rule: For \( t \in [\chi] \), node \( N(l, j) \) stores \( c_{(t-1)(\chi)+j} \) if and only if \( V_{n_1}(j, i_1) = 1 \).
- Second rule: Node \( N(l, j) \) stores symbol \( c_{n+i_2} \) if and only if \( W_{L,n_1}(j, i_2) = 1 \).

We will show that the distribution rules of codes 1 and 2 can be made identical by properly relabeling the indices. Note that the rule of distributing encoded symbols comes from the incidence matrix of a given graph. We prove the statement of Theorem 5 by showing that two graphs corresponding to the rules of codes 1 and 2 are identical.

The encoded symbol distribution rule of code 1 is obtained from a single complete graph \( V_n \) (obtained from the first rule) and \( (\chi - 1) \) complete graphs \( V_{n_1} \) per cluster (obtained from the second rule). On the other hand, that of code 2 comes from \( \chi \) complete graphs \( V_{n_1} \) per cluster (obtained from the first rule) and a single complete \( L \)-partite graph \( W_{L,n_1} \) (obtained from the second rule). We conclude that two graphs are identical for the following reasons: any two nodes in the same cluster share \( \chi \) edges, while any two nodes in other clusters share a single edge. In Fig. 23, we provide an example of graphs corresponding to the encoded symbol distribution rules. A graph on the left side corresponds to the rule of code 1, while the right side corresponds to that of code 2. It is easy to see that the two graphs are the same.

Consider a scenario that \( \theta \) coded symbols \( \{c_1, \ldots, c_\eta\} \) are distributed to a \( (n, k, L, (n_1 - 1) \chi + n - n_1, \chi, 1) \) clustered DSS under the distribution rules described in Algorithm 2. Then, by using Lemma 5 and Remark 11, each symbol in set \( T = \{c_1, \ldots, c_\eta\} \) is stored on two distinct nodes in a single cluster, while each of the rest symbols \( \{c_{\eta+1}, \ldots, c_\theta\} \) is stored on two distinct nodes in different clusters. A data collector selects totally \( k \) nodes \( (k_1, k_2, \ldots, k_L \) nodes from \( 1^{st}, 2^{nd}, \ldots, L^{th} \) cluster) to retrieve coded symbols. Without a loss of generality, we assume \( k_1 \geq k_2 \geq \cdots \geq k_L \).

Here, we define a contacting vector as \( v = [k_1, \ldots, k_L] \). A set of feasible contacting vectors is obtained as
\[
V = \{ [k_1, \ldots, k_L] : \sum_{i=1}^{L} k_i = k, n_1 \geq k_1 \geq k_2 \geq \cdots \geq k_L \geq 0 \}.
\]

From properties (a) and (b) in Lemma 3, we know that any two nodes in a single cluster share \( \chi \) encoded symbols in set \( T \). Consider a \( j^{th} \) cluster with \( k_j \) contacting nodes by a data collector. The number of coded symbols in set \( T \) retrieved by a data collector from \( j^{th} \) cluster is expressed as
\[
T_j = \sum_{i=1}^{k_j} (n_1 - i) \chi.
\]

Therefore, the minimum number \( S \) of encoded symbols in set \( T \) retrieved by a data collector among possible contacting vector \( v \) is expressed as
\[
S = \min_{v \in V} \sum_{j=1}^{L} T_j = \min_{v \in V} \sum_{j=1}^{L} \sum_{i=1}^{k_j} (n_1 - i) \chi.
\]

Consider an arbitrary contacting vector \( v = [k_1, \ldots, k_L] \in V \). Assume there exists some \( j \) such that
\[
0 < k_{j+1} \leq k_j < n_1.
\]

Then, selecting a node in the \( j^{th} \) cluster instead of \( (j+1)^{th} \) cluster strictly decreases the number of symbols in set \( T \) retrieved by a data collector. Therefore, contacting vector \( v' = [k'_1, \ldots, k'_L] \) minimizes retrieved symbols in set \( T \) where
\[
k'_j = \begin{cases} n_1, & \text{if } j \leq \lfloor k/n_1 \rfloor \\ \text{mod}(k, n_1), & \text{if } j = \lfloor k/n_1 \rfloor + 1 \\ 0, & \text{otherwise.} \end{cases}
\]

The value of \( S \) is derived as follow, which is equal to \( R \) stated...
in (2),
\[ S = \sum_{j=1}^{L} \sum_{i=1}^{k'} (n_I - i) \chi \]
\[ = \left\lfloor \frac{k}{n_I} \right\rfloor \sum_{i=1}^{n_I} (n_I - i) \chi + \sum_{i=1}^{\text{mod}(k,n_I)} (n_I - i) \chi \]
\[ = \left\lfloor \frac{k}{n_I} \right\rfloor \frac{n_I(n_I - 1)}{2} \chi + \sum_{i=1}^{\text{mod}(k,n_I)} (n_I - i) \chi = R \]

**APPENDIX D**

**Proof of Propositions**

**A. Proof of Proposition 1**

From Theorem 1, the upper bound of secrecy capacity with the maximum number of helper nodes is obtained by the cut-value of secure links (A.3), where a flow graph, compromised clusters and a cut are determined as in the proof of Theorem 1. Consider an information flow graph with a general number of helper nodes in the same setting. Then, a cut-value of secure links with arbitrary \( d_I \) and \( d_c \) intra- and cross- helper nodes is expressed as

\[ W(d_I, d_c) = \sum_{i=1}^{k} \min\{\alpha, w_i^*(d_I, d_c)\}, \]

where \( w_i^*(d_I, d_c) \) is defined as the sum of non-compromised incoming edge capacities from \( \{v_{in}^{out}i\}_{i=1}^{n} \) to \( i \)th regenerated input node \( v_{in}^{out}i \) with \( d_I \) and \( d_c \) helper nodes from the same and other clusters. Then, \( W(d_I, d_c) \) is the upper bound of secrecy capacity with general \( d_I \) and \( d_c \).

In the bandwidth-limited regime where \( \alpha = \gamma \), the cut-value is simply expressed as

\[ W(d_I, d_c) = \sum_{i=1}^{k} w_i^*(d_I, d_c). \]

The exact value of secrecy capacity with the maximum helper nodes, \( C_{s,\text{max}} \), is obtained in the present paper as follows, which is the direct result of Theorem 2:

\[ C_{s,\text{max}} = W(d_{I,\text{max}}, d_{c,\text{max}}), \]

where

\[ d_{I,\text{max}} = n_I - 1, \]
\[ d_{c,\text{max}} = n - n_I. \]

We prove the proposition by showing

\[ w_i^*(d_I, d_c) \leq w_i^*(d_{I,\text{max}}, d_{c,\text{max}}), \quad \forall d_I, d_c \quad (\text{D.1}) \]

for all \( i \leq n_I L_c \) and

\[ \sum_{i=n_I L_c + 1}^{k} w_i^*(d_I, d_c) \leq \sum_{i=n_I L_c + 1}^{k} w_i^*(d_{I,\text{max}}, d_{c,\text{max}}), \quad \forall d_I, d_c \quad (\text{D.2}) \]

which leads to the inequality

\[ W(d_I, d_c) \leq C_{s,\text{max}} \quad \forall d_I, d_c. \quad (\text{D.3}) \]

The physical meaning of (D.3) is that it is impossible to securely store data more \( C_{s,\text{max}} \) symbols with arbitrary \( d_I \) and \( d_c \). Thus, setting \( d_I \) and \( d_c \) to the maximum possible values maximizes secrecy capacity.

Case 1) \( i \leq n_I L_c \): Notice that the \( i \)th replacement node is located in the compromised cluster. Thus, every cross-cluster traffic coming to the replacement node is observed by a cluster eavesdropper. Generally, \( w_i^* \) is expressed as

\[ w_i^*(d_I, d_c) = \max\{(d_I + 1 - x), 0\} \beta_I \]
\[ = \max\{1 - \frac{x - 1}{d_I}, 0\} \gamma_I \]

where \( x \in [n_I] \) and \( y \in [L_c - 1] \cup \{0\} \). For fixed \( x \), \( y \) and \( \gamma_I \), \( w_i^*(d_I, d_c) \) is a decreasing function of \( d_I \) and independent of \( d_c \). Therefore, (D.1) holds for \( i \leq n_I L_c \).

Case 2) \( n_I L_c < i \leq k \): As stated in the proof of Theorem 1, the right hand side of (D.2) corresponds to the capacity \( C_{r} \) of a reduced \((n', k', L', \alpha', \beta_1, \beta_c)\) clustered DSS with maximal helper nodes,

\[ C_{r}(d_I, d_c) = \sum_{i=n_I L_c + 1}^{k} w_i^*(d_{I,\text{max}}, d_{c,\text{max}}) \]

where \( n' = n - n_I L_c, k' = k - n_I L_c \) and \( L' = L - L_c \). Since maximizing the number of helper nodes maximizes capacity of a clustered DSS, as stated in Proposition 1 in [6], (D.2) directly holds.

**B. Proof of Proposition 2**

Note that \( C_{s}^{BL}(0) \) corresponds to the case without an eavesdropper, which is the capacity of a clustered DSS in the bandwidth-limited regime. The exact value of capacity is obtained as below from Remark 10, by setting \( \epsilon = \beta_c / \beta_1 \) and bandwidth-limited condition \( \alpha = \gamma \).

\[ C_{s}^{BL}(0) = k \gamma - \frac{\gamma}{2} \left( 1 - \epsilon \right) (\gamma a_1^2 + r^2 - k) + k (k - 1) \epsilon \]
\[ = \frac{\gamma}{2} \left( n_I - 1 \right) + \left( n - n_I \right) \epsilon. \]

Combining with Corollary 5, which says \( C_{s}^{BL}(L_c) \) is a quadratic function of \( L_c \), with the axis of symmetry \( L_c = L - 1/2 \), gives the unique expression of \( C_{s}^{BL} \).

**REFERENCES**


Sung Whan Yoon (M’17) received the M.S. and Ph.D. degrees in electrical engineering from the Korea Advanced Institute of Science and Technology (KAIST), Daejeon, South Korea, in 2013 and 2017, respectively. He is currently a postdoctoral researcher in KAIST from 2017. His research interests are in the area of coding theory, distributed system and deep learning algorithms. Recently, his primary interest is focused on meta-learning and continual learning of deep neural networks. Also, he is highly interested in hardware-friendly learning algorithms based on distributed resources (data, computation and communication). He was a co-recipient of the IEEE International Conference on Communications best Paper Award in 2017.

Jaekyun Moon (F’05) received the Ph.D degree in electrical and computer engineering at Carnegie Mellon University, Pittsburgh, PA, USA. He is currently a Professor of electrical engineering at KAIST. From 1990 through early 2009, he was with the faculty of the School of Electrical and Computer Engineering at the University of Minnesota, Twin Cities. He consulted as Chief Scientist for DSPG, Inc. from 2004 to 2007. He also worked as Chief Technology Officer at Link-A-Media Devices Corporation. His research interests are in the area of channel characterization, signal processing and coding for data storage and digital communication. Prof. Moon received the McKnight Land-Grant Professorship from the University of Minnesota. He received the IBM Faculty Development Awards as well as the IBM Partnership Awards. He was awarded the National Storage Industry Consortium (NSIC) Technical Achievement Award for the invention of the maximum transition run (MTR) code, a widely used error-control/modulation code in commercial storage systems. He served as Program Chair for the 1997 IEEE Magnetic Recording Conference. He is also Past Chair of the Signal Processing for Storage Technical Committee of the IEEE Communications Society. He served as a guest editor for the 2001 IEEE ISAC issue on Signal Processing for High Density Recording. He also served as an Editor for IEEE TRANSACTIONS ON MAGNETICS in the area of signal processing and coding for 2001-2006. He is an IEEE Fellow.