LETTER

Optimal Spot-Checking Ratio for Probabilistic Attacks in Remote Data Checking

Younsoo PARK[†], **Jungwoo CHOI**[†], **Young-Bin KWON**[†], **Jaehwa PARK**[†], ***Nonmembers***, ***and*** **Ho-Hyun PARK**[†a)], ***Member***

SUMMARY Remote data checking (RDC) is a scheme that allows clients to efficiently check the integrity of data stored at an untrusted server using spot-checking. Efforts have been consistently devoted toward improving the efficiency of such RDC schemes because they involve some overhead. In this letter, it is assumed that a probabilistic attack model is adopted, in which an adversary corrupts exposed blocks in the network with a certain probability. An optimal spot-checking ratio that simultaneously guarantees the robustness of the scheme and minimizes the overhead is obtained.

key words: *RDC, PDP, spot-checking, probabilistic attack, δ-robustness*

1. Introduction

Remote storage provided by a storage service provider (SSP) allows clients to store large amounts of data at affordable prices. Because clients might store private data that is not supposed to be revealed to others, the data stored at the server should be securely maintained. However, clients may not fully trust SSPs. Therefore, the importance of an auditing scheme for remote storage servers has been emphasized.

Under these circumstances, the provable data possession (PDP) scheme has been proposed [1], [2]. This scheme is capable of checking the integrity of data without downloading the original data from the perspective of the client and without retrieving the entire data from the perspective of the server. These requirements can be fulfilled by adopting spot-checking [3]. In spot-checking, the integrity of data is checked by randomly sampling a set of blocks rather than accessing the entire data set. Although spot-checking is suitable for detecting large corruptions, it is vulnerable to small corruptions against the entire data size. This problem has been resolved by introducing the notion of δ-robustness, which adopts forward error correction (FEC) codes such as the Reed-Solomon (RS) code [4]–[6] (δ-robustness will be defined in Sect. 2.1.)

Although many studies have investigated the efficiency of PDP schemes, some overhead is still incurred during spot-checking. For example, the auditing scheme introduced in [5] has a drawback in that even though a client requests for a download of only a small part of the data, whole parity blocks are redundantly downloaded together with the requested data in order to conceal the association between the data blocks and the parity blocks.

To overcome this problem, Dong et al. proposed a scheme that efficiently ensures δ-robustness by minimizing the parity redundancy in the downloaded data [7]. Although this scheme provides an optimal value for the number of parity blocks to be downloaded during one spot-check, it cannot suggest an optimal value for the number of spot-checked blocks to efficiently ensure δ-robustness for a file stored at a server.

According to the PDP scheme, if the number of spot-checked blocks is sufficient compared to the number of total blocks in a file, the δ-robustness of the file is ensured. However, if spot-checking is performed for too many blocks, the large number of samplings will lead to excessive overhead. On the other hand, if the number of spot-checked blocks is too small, δ-robustness cannot be ensured. In this sense, the present study aims to propose a technique for selecting the optimal number of spot-checked blocks that minimizes the overhead and ensures δ-robustness simultaneously.

2. System Model

2.1 Robust RDC Scheme

In the PDP scheme [4], a file F is represented as a finite ordered collection of f blocks: $F = b_1, \ldots, b_f$. To store F in the server, a client must transform F into an encoded $\tilde{F}$. An encoding algorithm such as [3], [4] and [5] divides F into k-block chunks. Then the (n,k) RS code is applied to each chunk which is called a constraint group. The redundancy (d) and error correcting capacity (t) for each constraint group are computed as $d = n - k$ and $t = d/2$ respectively [6].

A file F is encoded into $\tilde{F} : b_1, \ldots, b_f, c_1, \ldots, c_{\frac{f}{k}d}$. In the file $\tilde{F}$, k data blocks are constrained by d check blocks. The check blocks contain error correcting codes as redundancy information for data recovery. Next, it permutes the check blocks to hide the structure of the file. The output of the permutation is represented as $R : r_1, \ldots, r_{\frac{f}{k}d}$ which is an ordered collection of $\frac{f}{k}d$ blocks. As the result, the file $\tilde{F}$ forms a concatenation of the F and the redundancy information R. Let $\tilde{f} = f + \frac{f}{k}d$. At the end of this encoding, the file $\tilde{F}$ is encrypted by the RSA encryption.

After encoding, the client sends $\tilde{F}$ to the server and stores metadata in its local storage. The server receives $\tilde{F}$

Manuscript received June 7, 2016.
Manuscript revised March 16, 2017.
Manuscript publicized April 26, 2017.
[†]The authors are with Chung-Ang University, Seoul, Korea.
a) E-mail: hohyun@cau.ac.kr
DOI: 10.1587/transinf.2016EDL8120

and saves it in the server-side storage. After that, the server computes a proof of data possession and sends it back to the client. Finally, the client checks validity of the proof.

In the challenge phase, an auditor (or client) generates some challenge block's index which is a collection of randomly sampled c blocks $i_1, \ldots, i_c$ where $1 \leq c \leq \tilde{f}$ and requests a proof of possession for the challenged blocks $b_{i_1}, \ldots, b_{i_c}$ to the server. In this *sampling* scheme named as spot-checking [4], the value $c/\tilde{f}$ is referred to as spot-checking ratio. The server sends the proof of the challenged blocks to the auditor. Then the auditor checks the validity of the proof.

If an adversary corrupts $x_{\tilde{F}}$ blocks among $\tilde{f}$ blocks, the probability that an auditor can detect the corruption, denoted as $P(detect_{\tilde{F}})$, can be computed as shown in Eq. (1) [4]. The detection is considered as successful in case the auditor finds one or more corrupted blocks through the spot-checking. Equation (1) depends on the number of challenged blocks c and corrupted block ratio $x_{\tilde{F}}/\tilde{f}$.

$$1 - \left(1 - \frac{x_{\tilde{F}}}{\tilde{f}}\right)^c = P(detect_{\tilde{F}}) \tag{1}$$

Adversarial model. Since the SSP's server is not fully trusted, an adversary dominating the server may try to cheat the auditor. For example, the adversary can corrupt part of stored data in the server after the server sends a proof to the auditor. If part of data is corrupted, the auditor can detect the corrupted blocks by the spot-checking at the next challenge phase. Since the challenging uses a sampling scheme, the auditor might fail to detect very small part of corruption.

Since the small size corruption can be recovered by the RS code, the adversary must destroy data of which size is large enough not to be recovered by the RS code. But the size should be small not to be detected by the spot-checking. In this situation, an RDC scheme must ensure δ-robustness to protect the adversary's attack and securely store the client's data. The δ-robustness is defined as follows [4]

Definition 1: When an original file F is encoded as $\tilde{F}$ and stored at a server, an RDC scheme provides δ-robustness when *i)* the auditor will detect corruption with high probability (more than $1 - \epsilon$) if an adversary corrupts more than a δ-fraction of $\tilde{F}$; or *ii)* the auditor will recover the data in F with high probability (more than $1 - \epsilon$) if an adversary corrupts at most a δ-fraction of $\tilde{F}$.

Definition 1 states that δ-robustness requires that the probability of failure to detect the data corruption by the spot-checking, i.e., $1 - P(detect_{\tilde{F}})$ or the probability of failure to recover the file by the RS code, i.e., $1 - P(recover_{\tilde{F}})$ should be less than ϵ. Therefore, ϵ must be a very small value. For example, ϵ was set to 10^{-10} in [4], [5], and [7].

2.2 Probabilistic Attack Model

If part of a file is corrupted, the probability that an auditor detects the corruption increases as the portion of corrupted data grows. Contrarily the probability that the file is recovered using the RS codes increases as the portion of corrupted data lessens. As a consequence, an adversary who is eager to achieve his/her malicious objective needs to determine a suitable ratio of corruption preventing detection by the spot-checking and recovery by the RS codes simultaneously.

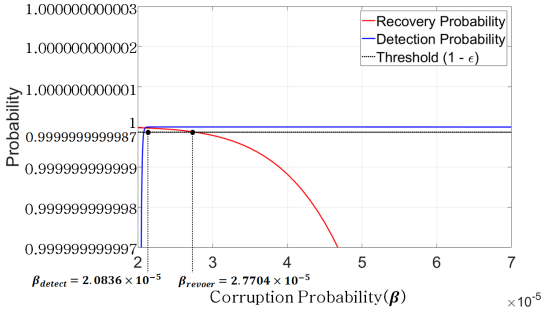

Fig. 1 δ-robustness is guaranteed at $c/\tilde{f} = 4\%$

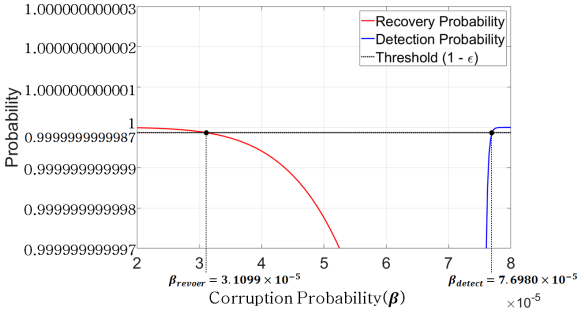

Fig. 2 δ-robustness is not guaranteed at $c/\tilde{f} = 2\%$

The πR scheme [4] disallows an adversary to identify that some blocks are original data F or redundancy information R. Therefore, the adversary is forced to corrupt only part of encoded file blocks randomly and such a behavior can be explained by a probabilistic attack model. Under this model, we assume that an adversary corrupts each block with a certain probability β which is large enough and small enough. The adversary's attack is considered as successful if the auditor fails to detect and recover the file corruption in the challenge phase.

Example 1 There is a file $\tilde{F}$ which has $\tilde{f}$ block at a server encoded by a (140, 130) RS code. The size of the encoded file $\tilde{F}$ is 4TB where a block size is 4KB. While a client was challenging c blocks of $\tilde{f}$ blocks, the file was exposed to a probabilistic attack attempted by an adversary. In the challenge phase, if an auditor extracts 4% of the file for spot-checking, the error detection probability and recovery probability according to the corruption probability can be presented as graphs, as shown in Fig. 1.

The blue solid line and red solid line denote the error detection probability and error recovery probability, respectively (The calculation of these probabilities will be explained in Sects. 3.1 and 3.2). In Fig. 1, either the error detection probability or the error recovery probability is greater than or equal to the threshold $1-\epsilon$ in the entire range of β. Thus, δ-robustness is guaranteed.

Figure 2 plots the error detection probability versus the

Table 1 Some probabilities in normal distribution

m	$P(x_{\tilde{F}} < E[x_{\tilde{F}}] - m\sigma[x_{\tilde{F}}])$ or $P(x_{\tilde{F}} > E[x_{\tilde{F}}] + m\sigma[x_{\tilde{F}}])$
5	2.8665×10^{-7}
6	9.8660×10^{-10}
7	1.2971×10^{-12}

error recovery probability when all the conditions are the same as in Fig. 1 except that the percentage of sampled data for spot-checking decreases to 2%. Figure 2 indicates that there exists a range of β where both the error detection probability and the error recovery probability are lower than $1-\epsilon$. Thus, δ-robustness is not guaranteed in this range. In Figs. 1 and 2, the error recovery probability is predetermined at the time of RS encoding because it only depends on the RS encoding parameters n and t as shown in Eqs. (11) and (12), whereas the error detection probability varies with the spot-checking ratio, $c/\tilde{f}$. Hence, we can achieve δ-robustness by adjusting the spot-checking ratio appropriately.

Assume that a block in the file $\tilde{F}$ is corrupted by a constant probability β and the events where each block is corrupted are mutually independent. Then, the event that $x_{\tilde{F}}$ blocks among the c spot-checked blocks are corrupted follows the binomial distribution B(c,β). When c becomes sufficiently large, the binomial distribution can be mapped to a normal distribution $\mathrm{N}(c\beta, c\beta(1-\beta))$. In the normal distribution, $P(x_{\tilde{F}} < E[x_{\tilde{F}}] - m\sigma[x_{\tilde{F}}])$ or $P(x_{\tilde{F}} > E[x_{\tilde{F}}] + m\sigma[x_{\tilde{F}}])$, the probabilities that $x_{\tilde{F}}$ is m times the standard deviation away from the mean are calculated for some m as shown in Table 1. (In the above formula, $E[x_{\tilde{F}}]$ and $\sigma[x_{\tilde{F}}]$ indicate the expected value of the random variable $x_{\tilde{F}}$ and the standard deviation of $x_{\tilde{F}}$, respectively.)

3. Proposed Approach

3.1 Minimum Threshold to Detect Errors

According to Definition 1, to guarantee δ-robustness from the perspective of error detection, the probability that the adversary successfully performs an attack, denoted as $1 - P(detect_{\tilde{F}})$, should be less than ϵ. It can be written as Eq. (2).

$$1 - P(detect_{\tilde{F}}) \leq \epsilon \tag{2}$$

Equation (1) can be transformed into Eq. (3).

$$1 - P(detect_{\tilde{F}}) = \left(1 - \frac{x_{\tilde{F}}}{\tilde{f}}\right)^c \tag{3}$$

Since $x_{\tilde{F}}/\tilde{f}$ means the ratio of the damaged blocks to the whole file blocks, it is expressed as $Damage_{\tilde{F}}$.

$$1 - P(detect_{\tilde{F}}) = (1 - Damage_{\tilde{F}})^c \tag{4}$$

By combining Eqs. (2) and (4),

$$(1 - Damage_{\tilde{F}})^c \leq \epsilon \tag{5}$$

If we rewrite Eq. (5) for $Damage_{\tilde{F}}$,

$$1 - \epsilon^{\frac{1}{c}} \leq Damage_{\tilde{F}} \tag{6}$$

Equations (2)~(6) state that the ratio of damaged blocks should be larger than or equal to $1 - \epsilon^{\frac{1}{c}}$ in order to detect adversary's attacks with a probability of $1 - \epsilon$ or higher. Therefore, the minimum damaged block ratio necessary for detecting data corruption, $Damage_{detect-min}$, can be expressed as

$$Damage_{detect-min} = 1 - \epsilon^{\frac{1}{c}} \tag{7}$$

If we multiply Eq. (7) by $\tilde{f}$, we can obtain a threshold Th_{detect} that represents the minimum number of damaged blocks (i.e., the minimum value of $x_{\tilde{F}}$). We can express this threshold as

$$Th_{detect} = Damage_{detect-min} \times \tilde{f} = (1 - \epsilon^{\frac{1}{c}}) \times \tilde{f} \tag{8}$$

As mentioned in Sect. 2.2, the distribution of $x_{\tilde{F}}$ is a normal distribution $\mathrm{N}(c\beta, c\beta(1-\beta))$ for a large c. In this distribution, spot-checking will be failed if $x_{\tilde{F}}$ is less than Th_{detect}. If we set Th_{detect} to be $E[x_{\tilde{F}}] - m\sigma[x_{\tilde{F}}]$, the probability that the spot-checking will fail to detect the damaged blocks becomes $P(x_{\tilde{F}} < E[x_{\tilde{F}}] - m\sigma[x_{\tilde{F}}])$. Since the probability of failure must satisfy Eq. (2), we can express this relationship by Eq. (9).

$$P(x_{\tilde{F}} \leq E[x_{\tilde{F}}] - m\sigma[x_{\tilde{F}}]) \leq \epsilon \tag{9}$$

Equation (9) indicates that ϵ is affected by $E[x_{\tilde{F}}] - m\sigma[x_{\tilde{F}}]$. If we set m to 6, ϵ will be 9.8660×10^{-10} as shown in Table 1. This value is greater than ϵ of [4], [5], and [7]. However, if we set m to 7, ϵ will be 1.2971×10^{-12}. Notice that this value is smaller than ϵ of [4], [5], and [7]; hence, our proposed method guarantees δ-robustness of which level is stronger than [4], [5], and [7]. Equation (10) represents the minimum threshold value of $x_{\tilde{F}}$ when $m = 7$.

$$Th_{detect} = E[x_{\tilde{F}}] - 7\sigma[x_{\tilde{F}}] = c\beta - 7\sqrt{c\beta(1-\beta)} \tag{10}$$

Let the solution of Eq. (10) be β_{detect}. If β is less than β_{detect}, an auditor cannot detect the corruption with high probability $1 - \epsilon$. In this case, the δ−robustness is not ensured from the perspective of error detection.

Given the parameters of $\tilde{f}$, c, and ϵ, we can obtain the value of β_{detect} by computing Th_{detect} in Eq. (8) and substituting it into Eq. (10). Figure 1 and Fig. 2 show the values for $\beta_{detect} = 2.0836 \times 10^{-5}$ and 7.6980×10^{-5}, which are calculated in this way when c=46,253,494 and c=23,126,747, respectively, provided that $\epsilon = 1.2971 \times 10^{-12}$, $\tilde{f} = 1,156,337,354$. The blue lines in both figures represent the values of $P(x_{\tilde{F}} \geq Th_{detect})$ on varying β at the same condition of $\tilde{f}$ and c as above under the normal distribution $\mathrm{N}(c\beta, c\beta(1-\beta))$. Note that Th_{detect} is obtained from Eq. (8).

3.2 Maximum Threshold to Recover Errors

As explained in Sect. 2.1, a single constraint group that is

encoded by an (n,k) RS code consists of n blocks and its error recovery capacity is $t(t = d/2)$. Let c blocks among $\tilde{f}$ blocks in a file be exposed to probabilistic attacks after the challenge phase. Further let $recover_{\tilde{F}}$ denote the event that all these blocks are recovered by the RS codes and $recover_i$ denote the event that the i-th constraint group is recovered. The number of constraint groups of the exposed blocks, g, can be written as $\lceil c/n \rceil$.

The (n, k) RS coding can recover the constraint group as long as up to t blocks among n blocks in one constraint group are damaged. Therefore, the probability of a successful recovery of the i-th constraint group can be written as

$$P(recover_i) = \sum_{l=0}^{t} \binom{n}{l} (\beta)^l (1-\beta)^{n-l} \tag{11}$$

where the attack probability is β and $\binom{n}{l}$ denotes the binomial coefficient which represents a combination of l objects from n objects. If the events $recover_i$ for $1 \le i \le g$ are independent, the probability of success to recover the attacked file can be written as

$$\begin{aligned} P(recover_{\tilde{F}}) &= [P(recover_i)]^g \\ &= \left[\sum_{l=0}^{t} \binom{n}{l} (\beta)^l (1-\beta)^{n-l} \right]^g \end{aligned} \tag{12}$$

To guarantee δ-robustness from the perspective of error recovery, the probability that the adversary successfully performs an attack, denoted as $1 - P(recover_{\tilde{F}})$, should be less than ϵ. This can be expressed as

$$1 - P(recover_{\tilde{F}}) \le \epsilon \tag{13}$$

In Eq. (12), $P(recover_{\tilde{F}})$ decreases as β increases. Therefore, Eq. (13) does not hold if β increases above a certain value. In the range below a certain value of β, Eq. (13) is satisfied and δ-robustness is thus guaranteed. This value of β is denoted as $\beta_{recover}$. It is the maximum value of β that guarantees δ-robustness. It can be calculated by

$$1 - \left[\sum_{l=0}^{t} \binom{n}{l} (\beta_{recover})^l (1-\beta_{recover})^{n-l} \right]^g = \epsilon \tag{14}$$

Now, we can define another threshold $Th_{recover}$, the maximum number of damaged blocks (i.e., the maximum value of $x_{\tilde{F}}$) such that the file is recovered with a probability of $1 - \epsilon$ or greater. As with Eq. (10), if we make $E[x_{\tilde{F}}] + m\sigma[x_{\tilde{F}}]$ equal to $Th_{recover}$, the probability that the (n,k) RS coding fails to recover the file $\tilde{F}$ becomes $P(x_{\tilde{F}} \ge E[x_{\tilde{F}}] + m\sigma[x_{\tilde{F}}])$. It can be written as

$$\begin{aligned} Th_{recover} &= E[x_{\tilde{F}}] + m\sigma[x_{\tilde{F}}] \\ &= c\beta_{recover} + 7\sqrt{c\beta_{recover}(1-\beta_{recover})} \end{aligned} \tag{15}$$

If $\beta > \beta_{recover}$, an auditor cannot recover the corruption with high probability $1 - \epsilon$, and the δ−robustness is not ensured from the perspective of error recovery. Given the parameters of n, t, c and ϵ, we can calculate $\beta_{recover}$ from Eq. (14) and substitute it into Eq. (15) to obtain $Th_{recover}$. Figure 1 and 2 show the values for $\beta_{recover} = 2.7704 \times 10^{-5}$ and 3.1099×10^{-5}, which are calculated in this way when c=46,253,494 and c=23,126,747 respectively, provided that $\epsilon = 1.2971 \times 10^{-12}$, $n = 140$, and $t = 5$. The red lines in both figures were calculated by Eq. (12) on varying β at the same condition of n, t and c as above.

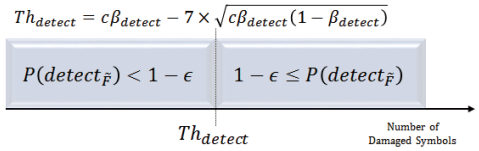

Fig. 3 Relationship between Th_{detect} and $P(detect_{\tilde{F}})$

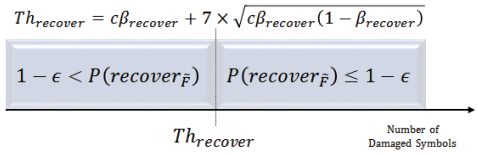

Fig. 4 Relationship between $Th_{recover}$ and $P(recover_{\tilde{F}})$

3.3 Interval to Ensure δ-robustness

Figures 3 and 4 show the probabilities related to Th_{detect} and $Th_{recover}$ on a horizontal line.

On the left-hand side of Th_{detect} in Fig. 3, δ-robustness cannot be guaranteed because the error detection probability is lower than the threshold. On the right-hand side of $Th_{recover}$ in Fig. 4, δ-robustness cannot be guaranteed because the error recovery probability is lower than the threshold. When we combine the two figures on a single horizontal line, $Th_{detect} < Th_{recover}$ should hold for guaranteeing δ-robustness for any β. Equation (8) indicates that Th_{detect} (hence, β_{detect} by Eq. (10)) is significantly affected by c, whereas Eq. (14) indicates that $\beta_{recover}$ (hence, $Th_{recover}$ by Eq. (15)) is significantly affected by t and g (hence, c because $g = \lceil c/n \rceil$).

However, $Th_{recover} < Th_{detect}$ may hold if we choose a small c[†] for spot-checking or use a low value of t for file encoding. In this case, δ-robustness cannot be guaranteed because both the error detection probability and the error recovery probability are lower than the threshold, as seen in Fig. 2. This might allow an adversary to have a chance to attack without being detected and also prevent recovery simultaneously.

4. Analysis

In order to identify the conditions under which δ-robustness is guaranteed, it is necessary to consider the parameters $c/\tilde{f}$, n, t, and ϵ, which might influence Th_{detect} and $Th_{recover}$. Note that ϵ was already set to $P(x_{\tilde{F}} > E[x_{\tilde{F}}] + 7\sigma[x_{\tilde{F}}]) = 1.2971 \times$

[†]Hereinafter we will use $c/\tilde{f}$ instead of c because c varies with file size.

Table 2 Values of Th_{detect} and $Th_{recover}$ with varying t and $c/\tilde{f}$

$c/\tilde{f}$		1%	2%	3%	10%	20%
Th_{detect}		2,737	1,368	912	273	136
$Th_{recover}$	t=1	**0.4193**	**0.4988**	**0.5520**	**0.7463**	**0.8878**
	t=2	**12.24**	**15.54**	**17.89**	**27.27**	**34.91**
	t=3	**76.68**	**103.6**	**124.2**	**217.6**	306.0
	t=4	**277.1**	**403.4**	**506.8**	1,036	1,606
	t=5	**758.1**	**1,179**	1,542	3,553	5,884
	t=6	**1,712**	2,801	3,768	9,404	16,227
	t=7	3,345	5,670	7,780	20,471	36,269
	t=8	5,837	10,153	14,129	38,561	69,571

10^{-12} in Sect. 2, and n was also fixed at the time of encoding files. Consequently, Th_{detect} and $Th_{recover}$ are determined by the remaining two parameters, $c/\tilde{f}$ and t. Table 2 lists the values of Th_{detect} and $Th_{recover}$ with varying $c/\tilde{f}$ and t while n is set to 140 and ϵ is set to 1.2971×10^{-12} for a 4TB file.

Among the values in Table 2, the bold-type values represent the cases in which $Th_{recover}$ is less than Th_{detect}. In these cases, δ-robustness is not guaranteed, as seen in Fig. 2. By contrast, the values in regular font represent the cases in which $Th_{recover}$ is larger than Th_{detect}, i.e., the cases in which δ-robustness is guaranteed, as seen in Fig. 1.

Table 2 shows that δ-robustness might be guaranteed or not depending on $c/\tilde{f}$ in spite of the same t value. For example, δ-robustness is guaranteed when t is 5 and spot-checking is performed using 3% extraction, whereas δ-robustness is not guaranteed when $c/\tilde{f}$ is as low as 2%. As $c/\tilde{f}$ decreases from 3% to 2%, $1/c$ increases and Th_{detect} in Eq. (8) also increases. Then, Th_{detect} switches sides with respect to $Th_{recover}$ in Fig. 1, as shown in Fig. 2. Now, we can define the cross point of Th_{detect} and $Th_{recover}$. In order to determine this cross point, we need to identify a suitable c for satisfying both Eqs. (16) and (17).

$$Th_{detect} = Th_{recover} \tag{16}$$

$$\begin{aligned} &c\beta_{detect} - 7\sqrt{c\beta_{detect}(1-\beta_{detect})} \\ &\quad = c\beta_{recover} + 7\sqrt{c\beta_{recover}(1-\beta_{recover})} \end{aligned} \tag{17}$$

If we find the c value satisfying Eq. (17) in the case of $t = 5$ and then calculate $c/\tilde{f}$, the cross point appears around 2.90%. This implies that δ-robustness can be guaranteed if we set $c/\tilde{f}$ to 2.9% or higher. This finding demonstrates that the 10% ratio taken in [4] and [7] is a sufficiently high spot-checking ratio to guarantee δ-robustness. According to Table 2, we ensure that for each t value, a minimum value of c exists such that δ-robustness is guaranteed. Because t is usually predetermined at the time of encoding, we can control only c after a file is loaded at a server. In this study, we denote this minimum value as c_{min}.

In summary, the case of $Th_{detect} < Th_{recover}$ leads to inefficiency because $c > c_{min}$, whereas when $Th_{recover} < Th_{detect}$, δ-robustness is not ensured because $c < c_{min}$. Therefore, $c_{min}/\tilde{f}$ is the best spot-checking ratio in the range where δ-robustness is ensured.

5. Conclusion

We proposed a mechanism for determining the optimal spot-checking ratio by applying a probabilistic attack model to existing RDC schemes. During this process, we defined the probabilities β_{detect} and $\beta_{recover}$ that are necessary for guaranteeing δ-robustness in terms of error detection and error recovery depending on the user's circumstances. We also described the steps for calculating the thresholds Th_{detect} and $Th_{recover}$, which are related to β_{detect} and $\beta_{recover}$. Based on the analysis of these thresholds, we described the steps for calculating $c_{min}/\tilde{f}$, i.e., the minimum spot-checking ratio. Thus, $c_{min}/\tilde{f}$ can be considered as the core parameter in this study. It is expected that this core parameter will minimize the spot-checking overhead, thereby providing an opportunity to efficiently use the limited resources in cloud systems.

Acknowledgements

This research was supported by the Chung-Ang University Research Scholarship Grants in 2015 and the National Research Foundation of Korea (NRF-2014R1A1A2056266 and NRF-2016R1D1A1B03933895).

References

[1] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," Proceedings of the 14th ACM Conference on Computer and Communications Security (CCS '07), pp.598–609, ACM, 2007.

[2] D. Xiao, L. Yang, C. Liu, B. Sun, and S. Zheng, "Efficient Data Possession Auditing for Real-World Cloud Storage Environments," IEICE Transactions on Information and Systems, vol.E98-D, no.4, pp.796–806, April 2015.

[3] R. Curtmola, O. Khan, and R. Burns, "Robust Remote Data Checking," Proceedings of the 4th ACM International Workshop on Storage Security and Survivability (StorageSS '08), pp.63–68, ACM, 2008.

[4] G. Ateniese, R. Burns, R. Curtmola, J. Herring, O. Khan, L. Kissner, Z. Peterson, and D. Song, "Remote Data Checking Using Provable Data Possession," ACM Trans. Inf. Syst. Secur., vol.14, no.1, pp.1–34, June 2011.

[5] B. Chen and R. Curtmola, "Robust Dynamic Provable Data Possession," 32nd International Conference on Distributed Computing Systems Workshops (ICDCSW), pp.515–525, IEEE, June 2012.

[6] J.S. Plank and L. Xu, "Optimizing Cauchy Reed-Solomon codes for fault-tolerant network storage applications," Fifth IEEE International Symposium on Network Computing and Applications (NCA '06), pp.173–180, IEEE, 2006.

[7] L. Dong, J. Park, J. Hur, and H.-H. Park, "An Enhanced Remote Data Checking Scheme for Dynamic Updates," KSII Transactions on Internet and Information Systems (TIIS), vol.8, no.5, pp.1744–1765, 2014.