Security Analysis for the Deduplication-friEndly Watermarking Scheme

1 Security Goal

The Deduplication-friEndly Watermarking scheme (DEW) aims to protect ownership of the outsourced multimedia files in a deduplication-based cloud storage system, which implies: 1) any untrusted parties, including both the honest-but-curious cloud server and the malicious data users, should not obtain the original file; and 2) any parties which do not possess the original file should not be able to pass the PoW check. Therefore, security of DEW is captured in Theorem 1, 2 and 3 (corresponding proofs are provided in Section 2, 3 and 4, respectively).

Theorem 1. *The cloud server cannot obtain the original file.*

Theorem 2. *The malicious data user cannot obtain the original file.*

Theorem 3. *Any parties which do not possess the original file cannot pass the PoW check.*

2 Proof of Theorem 1

Proof. (sketch) The could server can have access to the watermarked file (f_t), the checksum of the file($checksum_f$), the PoW tags, g^{α_j}, and the PoW proofs, including μ_j and $tail$. In the following, we analyze the cloud server could not obtain the original file based on aforementioned possessions:
1) By having access to f_t, the cloud server cannot obtain the original file by removing the watermark directly. Without any additional knowledge, the only strategy for the cloud server to obtain the original file is either brute-force guessing the original file directly or conducting a removal attack in which brute-force guessing the watermark location (denoted as event X) and content (denoted as event Y) simultaneously. However, possessing nothing other than f_t the cloud server cannot determine the correctness of brute-force guessing, which means restoring the original file from the watermarked file directly is infeasible.
2) By additionally having the $checksum_f$, which implies the knowledge of watermark location, the could server could not obtain the original file. Recall that, the watermark is determined by two independent factors, i.e., the watermark location and the watermark content. Only when exact watermark content is removed from exact location can the watermarked file be restored to the original file correctly. Based on $checksum_f$ and f_t, the strategy for the cloud server to restore the original file is still brute-force guessing, but $checksum_f$ can help the cloud server to determine if the guessed watermark location is correct. Let event

A_2 denotes one successful removal attack when given $checksum_f$, the probability of which is multiplication of probability of X and Y as the two events are independent. Suppose c_0, l_0 be total number of elements in the image matrix and the length of watermark bit stream ($l_0 \leq c_0$), respectively. The probability of identifying watermark location then is $P(X) = \frac{1}{\binom{c_0}{l_0}}$, and the probability of event Y is: $P(Y) = \frac{1}{2^{l_0}}$. Therefore, the probability of A_2 is

$$P(A_2) = P(X \bigcap Y) = P(X) \times P(Y) = \frac{1}{2^{l_0}} \times \frac{1}{\binom{c_0}{l_0}} < \frac{1}{2^{l_0}}.$$

It is clearly that when l_0 is large enough, the probability of the removal attack is negligible even when $checksum_f$ is disclosed to the cloud server.
3) By further having access to the PoW tags, the cloud server cannot obtain the original file. The event A_3 denotes leaking the original file due to having the PoW tags besides the $cheksum_f$. Firstly, consider n PoW tags as n equations of $n \times s + s + n$ unknowns, i.e., $n \times s$ symbols $\{m_{ij}\}$, s coefficients $\{\alpha_j\}$, and n $H_3(i||I_{wm})$. Given n equations, it is infeasible to solve all unknowns directly since the number of which is greater than the number of equations. Besides, the unknowns m_{ij} cannot be extracted from each PoW tag due to aggregation.

As the PoW tags itself does not reveal any knowledge about the original file, in order to restore the original file, the cloud server may either brute-force the original file in the content space with a probability $\frac{1}{c_1}$, considering the space for the image file contains c_1 elements, or brute-force the watermark by identifying the watermark location and the content exactly at the same time. However, checking the correctness of brute-force guessing is unavoidable not matter which strategy the cloud server adopt.

Compared to encryption, the search scale of the original file is narrowed down much because of the nature of watermarking that the majority proportion of the original file is exposed to public. However, without knowing the random bit stream, the cloud server either brute force guess n $H(i||I_{wm})$ or the watermark bit stream for checking the correctness of guessing file content by recomputing the PoW tags and comparing to the real PoW tags. Therefore, the probability $P(A_3)$ is

$$P(A_3) = \frac{1}{c_1} \times max(\frac{1}{2^{n\beta}}, \frac{1}{2^{l_0}}).$$

If the cloud server attempts to brute-force the watermark first, the watermark location can be identified through exploiting $checksum_f$ with probability $P(X) = \frac{1}{\binom{l_0}{c_0}}$. Then restoring the original file turns to brute-force guessing the watermark bit stream. In this case, the probability $P(A_3)$ is equal to $P(A_2)$.

In addition, the cloud server can accumulate PoW tags of different files, which, however, is useless in terms of obtaining the original file due to collision resistance property and masking design in the PoW tag. For collision resistance, suppose the coefficients ($\{\alpha_j\}$) are selected randomly, the probability of collision among at least two PoW tags of different files is

$$P_1 = 1 - \frac{2^q - 1}{2^q} \times \frac{2^q - 2}{2^q} \times ... \times \frac{2^q - (N' - 1)}{2^q},$$

where N' denotes the number of the PoW tags a DEW system will produce throughout the runtime. The collision is rare when given a large enough 2^q in system setup phase. For instance, the probability of collision among no more than 1.71×10^{15} 160-bits PoW tags is less than 10^{-18}. On the other hand, if the cloud server eventually finds a collision such that

$$\sum_{j=1}^{s} \alpha_j m_j + H(i||I_{wm}) = \sum_{j=1}^{s} \alpha_j^* m_j^* + H^*(i||I_{wm}^*)$$

for two different file blocks, the combination of symbols and coefficients can be extracted with a probability $\frac{1}{2^\beta}$ due to using mask. Therefore, accumulating PoW tags does not give any advantage to the cloud server on facilitating brute-force guessing. In conclusion, the probability of A_3 is

$$P(A_3) = max(\frac{1}{c_1} \times max(\frac{1}{2^{n\beta}}, \frac{1}{2^{l_0}}), P(A_2)),$$

which is small when given large β and l_0.

4) The cloud server cannot obtain the original file by additionally having access to the PoW proof. Let the event A_4 stands for obtaining the original file by exploiting the PoW proof. Similar to PoW tags, c PoW proofs $\{\mu_j | 1 \leq j \leq s\}$ derived from c challenged blocks can be viewed as equations, which are unsolvable because the number of unknowns, $c \times s$ symbols $\{m_{ij}\}$ and s random masks $\{x_j\}$ selected by the prover, is greater than the number of equations . The *tail* itself does not reveal masks used in PoW tags and proofs due to aggregation and hardness of DLP [1].

Additionally, suppose the cloud server may know which file block consists of public content by exploiting the $cheksum_f$ with the probability $P(X)$, it may deliberately select t file blocks from public proportion as challenges, remaining $(c + 1 - t) \times s$ unknowns in the replied s PoW proofs. Specially, if the cloud server can selects $t = c - 1$ such file blocks, what remain unknown are m_{ij} in the only watermarked file block and $\{x_j\}$, which can be determined only by brute-force guessing. Once the remaining m_{ij} are correctly guessed, the $\{x_j\}$ are fixed, and vice versa. The success probability are $\frac{1}{2^s - 1}$ and $\frac{1}{2^{sq}}$ respectively. Obviously $\frac{1}{2^{sq}} < \frac{1}{2^s - 1}$. To confirm the correctness of guessing, the cloud server may recompute *tail* and compare the result to the real *tail*, which requires the cloud server to brute-force c unknown $H(i||I_{wm})$ with probability $P(Y)$, at the same time. otherwise, recomputing *tail* is infeasible. In this case, the probability of A_4 is

$$P(A_4) = \frac{1}{\binom{l_0}{c_0}} \times \frac{1}{2^s - 1} \times P(Y).$$

Further, the cloud server could not obtain advantage from accumulating the PoW proofs. Similar to the PoW tag, the PoW proof is collision resistant. Be-

sides, since masks in different PoW proofs for different file are generated randomly and independently, the cloud server cannot infer file content from one to another. The probability of A_4 by accumulating the PoW proof is

$$P(A_4) = \frac{p_1}{2^{sq}}.$$

Therefore, the probability of event A_4 is

$$P(A_4) = max(\frac{1}{\binom{l_0}{c_0}} \times \frac{1}{2^s - 1} \times P(Y), \frac{p_1}{2^{sq}})$$

which means infeasibility for the cloud server to obtain the original file from the PoW proof under large enough l_0 and s.

3 Proof of Theorem 2

Proof. (sketch) The malicious data users can retrieve the compressed matrix of the watermarked file (i.e., M_{tr} and M_{tc}) and g^{α_j} during PoW. Besides, it is allowed to have access to either the watermarked file (f_t) or the watermark itself. In the following, we will analyze the malicious data user cannot obtain the original file by having the aforementioned possessions.
1) When the malicious data user has M_{tr}, M_{tc}, g^{α_j}, and f_t, which means no more information than what the cloud server has. As we already analyzed in Appendix 2 that the cloud server cannot obtain the original file, the malicious data user also cannot restore the original file.
2) The malicious data user who possesses the watermark besides M_{tr}, M_{tc}, and g^{α_j} cannot obtain the original file. Let B_1 be the event that the malicious data user obtains the original file in this case. Compared to f_t, the compressed matrix only reduces, rather than increases the effective information which can be utilized by the malicious data user. To be more specific, restoring the original file from M_{tr} or M_{tc} is equivalent to partitioning each integer in the two matrices into exact elements. Suppose the original $n \times s$ elements matrix is compressed to an $n \times d$ one, the probability for an adversary to figure out an exact partition from the $n \times d$ compressed matrix is

$$P(B_1) = \frac{1}{2^{8s}},$$

where '8' means every element in the pixel matrix is represented as 8-bits, and s is the number of the symbols within individual row (or column), which is small for a large enough s. Therefore, restoring the original file from the compressed matrix is infeasible.

4 Proof of Theorem 3

Proof. (sketch) In the following, we first show correctness of the verification process.

1) Suppose the PoW tags are derived correctly and the PoW proofs are computed honestly:

$$\begin{aligned} g^{\sigma} &= g^{\sum\limits_{(i,v_i\in Q)} v_i\sigma_i} \\ &= g^{\sum\limits_{j=1}^{s}\sum\limits_{(i,v_i\in Q)} \alpha_j(v_i m_{ij})+v_i F_\kappa(i)} \\ &= g^{\sum\limits_{j=1}^{s}\sum\limits_{(i,v_i\in Q)} \alpha_j(\mu_j - x_j)+v_i F_\kappa(i)} \\ &= (g^{\sum\limits_{(i,v_i)\in Q} v_i F_\kappa(i)}(g^{\sum\limits_{j=1}^{s}\alpha_j x_j})^{-1}) \cdot g^{\sum\limits_{j=1}^{s}\alpha_j\mu_j} \\ &= (g^{\sum\limits_{(i,v_i)\in Q} v_i F_\kappa(i)}(\prod_{j=1}^{s} g^{\alpha_j x_j})^{-1}) \cdot \prod_{j=1}^{s} g^{\alpha_j\mu_j} \\ &= tail \cdot \prod_{j=1}^{s}(g^{\alpha_j})^{\mu_j} \end{aligned}$$

As shown above, the verification equation holds if the PoW tags are derived correctly and the PoW proof is computed honestly.
2) The malicious user cannot pass the verification in the PoW using another file. To be more specific, it is difficult for the malicious data user to find a different file that derives the PoW tags which are identical to what computed from particular original one, using which to produce valid PoW proofs. Let B_1 denotes the event that the malicious data user finds one such different file block. In a PoW round, given a PoW tag for a specific original file block $\sigma = \sum\limits_{j=1}^{s} \alpha_j m_j + H(i||I_{wm})$ which is fixed by the initial uploader in *Initial Upload* phase. The malicious data user attempts to find another file block m'_j, for $1 \le j \le s$, such that the equation $\sigma = \sum\limits_{j=1}^{s} \alpha_j m'_j + H(i|I_{wm})'$ holds without knowing the coefficients $\{\alpha_j\}$ and I_{wm} for the file. If the malicious data user occasionally knows the PoW tag, it is not difficult for the malicious data user to figure out a $s+1$ partition of the PoW tag such that $\sigma = \sum\limits_{j=1}^{s+1} y_j$. Therefore, the underlying fake file m'_j deriving the identical PoW tag and fake challenge value v' need to satisfy the following equation:

$$y_j = \begin{cases} \alpha_j m'_j, 1 \le j \le s \\ v' \cdot H(i||I_{wm}), j = s+1 \end{cases}$$

The coefficients $\{\alpha_j\}$ are kept secret to the prover, which cannot be computed from $\{g^{\alpha_j}\}$ according to DLP hardness assumption. In addition, the random bit stream I_{wm} cannot be extracted from the compressed matrix due to compression. Therefore, the only way for the adversary to determine one specific m'_j and v' is guessing with a probability $\frac{1}{2^q}$, where q is the length of the symbol.

Correspondingly, the probability of event C_1 is

$$P(C_1) = \frac{1}{2^{(s+1)q}},$$

which is equal to forging valid PoW proofs, and is negligibly small for large enough s and q. Conclusively, it is infeasible to find such a file that help the malicious data user to pass the PoW.
3) By possessing incomplete file, the malicious data user could not pass the PoW. For small files, the cloud server will check all the file blocks, but for large files, only randomly selected subset of file blocks will be checked. This is common in the literature of cloud storage integrity checking, since it could be very expensive to check the entire data if the file is large. According to PDP [2], by randomly checking a certain number of file blocks, the verifier can detect data corruption with a high probability. For example, if the adversary corrupts 1% of the entire data, by randomly checking 460 blocks, the verifier can detect the corruption with 99% probability. In PoW scenario, the data corruption is equivalent to the difference between the file being checked and the original file.

Therefore, the probability for the malicious data user who attempt to use incomplete file to pass the PoW is

$$P(C_2) = \frac{\binom{t}{n-w}}{\binom{t}{n}},$$

where n and t stands for the number of file blocks and challenge blocks respectively. The probability would approximate 0 if t approximates n as proven in [2]. Specially, since the new PoW is a probabilistic checking, the cloud server can run multiple PoW executions for one prover to achieve adequate confidence.

References

1. D. Gordon, *Discrete Logarithm Problem*, pp. 352–353. Boston, MA: Springer US, 2011.
2. G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable data possession at untrusted stores," in *Proceedings of the 14th ACM conference on Computer and communications security*, pp. 598–609, Acm, 2007.