

Optimizing Business Outcomes through Data-Driven Decision-Making: Techniques for Complex Dataset Analysis

Assistant Professor P. Anjaneyulu, Assistant Professor D. Priyanka,
Assistant Professor Y. Manaswini

CSE Department
Aditya Institute of Technology and Management, Tekkali, Srikakulam, Andhra Pradesh,

Abstract- The widespread adoption of Cloud Computing technology in industry, education, and government sectors has made it a standard for IT implementation. Data leakage is one of them, particularly, the unauthorized transfer of information from one environment to any other domain. Data leakage has been a problem much before data was maintained digitally. It is therefore vital to prevent and detect this leakage so that the cloud service provider's reputation is not jeopardized. Furthermore it is integral that users' data confidentiality, integrity, and availability is not compromised. In most cases, data are handled by a third-party software whose security procedures are unknown to the user. This software serves as a bridge between the user and the cloud service provider. To resolve the issue of data leakage, several methods have already been proposed such as watermarking, cryptographic and probabilistic techniques. This paper, however, aims to use a revised version of the probabilistic approach by encrypting the user data even before it is uploaded through a portal. During the encryption process, a user ID is embedded into the encrypted file. When this file is accessed by another consumer, their user ID is also embedded into the file. Hence it makes it easier for the algorithm to detect the guilty agent by comparing the leaked file against the user file. A list of users who have accessed the file is thus maintained.

Index Terms- Cryptography, probabilistic techniques, S3 Bucket, Xampp, AES Algorithm.

I. INTRODUCTION

While efficient data sharing boosts organizational effectiveness, the peril of malicious entities compromising shared information poses a substantial threat to both outcomes and reputation.

A risk-based security report underscores a staggering 284% surge in leaked data records, exceeding 15.1 billion in 2019 alone, underscoring the growing challenge of data protection. Researchers are driven to formulate theories identifying the culprit responsible for data leaks. Strategies for data protection fall into four main categories: cryptography, watermarking, fingerprinting, and probabilistic methods. These approaches play a pivotal role in navigating the intricate realm of information security, mitigating risks linked to unauthorized data exposure.

Data protection methods can be broadly classified into four categories:

- Cryptography
- Watermarking

- Fingerprinting
- Probabilistic.

The major goal of these paper is to give authorized users safe web portal access to the stored data. Providing secure protection of data from unauthorized access is done by using encryption and decryption. And locating the leaking agents by means of embedded data. In order to encrypt files before putting them on a cloud server, we offer a gateway.

In the event of a data leak, the process of going back and finding the data is made easier by retaining access data that tracks the person and the file being accessed. The user will log in through the portal and encrypt the files before uploading them to the cloud storage. Additionally, the user has access to previously saved files that must first be decrypted by the gateway in order for the user to access them. The user may grant access to a different user, who must first log in to the portal in order to access the specified file. Additionally, the system saves the id for each file access. The user id is kept in a file when it is accessed. By comparing the value included in

the file with the user data, this makes it simpler for the algorithm to identify the user who started the leak.

II. LITERATURE SURVEY

Shruti D. Meshram et al This new system introduces the Agent Guilt Model, a way to figure out if an agent is likely responsible for a leak without changing the original sensitive data. This method looks at the leaked data and compares it with the data from various agents, checking for similarities. It also takes into account the likelihood that someone could have guessed certain information in other ways. To make the detection even better, fake or dummy objects can be mixed in with the real ones when giving them to the agents. This extra layer helps increase the chances of finding the agent at fault.[1]

Kumara Hulyurdurga et al To find data leaks, they pay attention to important things like the agent's ID and where they are. They watch out for any unauthorized access, which could happen if someone uses shared keys or gets data from devices not part of the organization's network. They use IP addresses to see if an agent is trying to get data from a different place. If an agent is caught doing something wrong, the distributor can either give a warning or kick the agent out of the system. Warnings are sent through email and alerts to the agent. If they decide to remove the agent, they delete the account, stopping them from logging in or accessing anything else. This active approach helps quickly deal with possible data breaches, making sure the organization's important information stays safe.[2]

Pachipala Yellamma et al They showed that it's possible to figure out if an agent might be the one causing a leak. They do this by comparing the agent's information with the leaked documents and data from other stores. They find the agents by looking at things like how much time they spend with the data, how often they open a particular file, and so on. By checking these factors, they can calculate the likelihood. If the probability is higher than a certain limit (threshold value), they can say that the agent might be responsible for the leak.[3]

Abhijeet Singh et al To reduce the risk, this system suggests putting a watermark on important data before sharing it. This watermark helps track where the data came from and adds an extra layer of protection. It stops people from getting to the data without permission and makes it easier to find out who's responsible for any problems by including fake objects in the data. This paper looks into various ways of putting watermarks, like using DCT, DWT, wavelet, and invisible methods, and explains how they are important for making data more secure.[4]

Ishu Gupta et al They share a model to find out who's responsible when something goes wrong. The idea is to give a bit of data to different people, making sure they don't have all the data. They also mix in some fake things with the real stuff when giving it out. This careful sharing and adding fake things make it more likely to figure out who did something wrong. If the person giving out the data sees it in a place it shouldn't be, they compare it with what they gave out using something called Bigraph. This helps them figure out who might be at fault by looking at the chances. In the future, they want to keep improving how they give out and share data to make sure really important stuff stays safe.[5]

Bhagwan D. Thorat et al In this new system, they use a technique called perturbation, which means making the information a bit less sensitive before giving it to agents. They create a system to watch agents for possible wrongdoing and come up with ways to share things with many agents to increase the chances of finding someone leaking information. They also consider adding "fake" things to the original data. These things aren't real, but they look real to the agent. In the world of intelligence, these fake things act like a special mark for the whole set of data without changing any individual parts. If an agent with a fake thing leaks information, the person sharing the data can figure out which agents are to blame.[6]

Shital Chattar et al They made a system where if you want to get a file someone shared, they create a special key. To actually get the file, you have to ask the person who shared it for this special key. Once they give you the key, you can then get the file. If someone tries to get the file without the right key by trying to guess it, the system will notice and tell them they might be leaking information.[7]

| S.No | Author | Topic | Year | Outcome |
|------|------------------------------|--|------|--|
| 8 | Sushilkumar N. Holambe Et al | The Guilt Detection Approach in Data Leakage Detection | 2015 | This Paper Detect data leaks by tracing origin from distributors to agents, using allocation algorithms and access controls to prevent unauthorized access and identify potential leaks. |
| 9 | S. Jenila et al | Guilt Model Process for Identifying Data Leakage and Guilty Agent in Data transmission | 2012 | Data distributor assigns private objects to detect and identify leaking agents, protecting sensitive data. |
| 10 | Niketa Kirdat et al | Data Leakage Detection and File Monitoring in Cloud Computing | 2018 | Efficient distribution techniques and guilt probability calculation aid in detection and prevention, enhancing data protection. |

III. METHODOLOGY

1. Problem Statement

The end users are not given access to a secure cloud storage system by the currently in use cloud computing solutions. While the user can encrypt data independently and upload them to the cloud, the solutions do not offer any way for the user to self-encrypt the files there. Instead, the cloud does not encrypt the information in a way that ensures the user will be the only one who can access them. They don't offer a mechanism to verify the data leak either. Our solution will offer a user interface where the user can upload files to the cloud after they have been encrypted, assisting in the identification of the guilty agent who leaked the file.

2. Existing System

Existing methods for data protection include cryptographic, fingerprinting, and probabilistic approaches. Cryptographic methods employ multi-authority CP-ABE techniques to hold malicious users accountable, while watermarking embeds secret codes for identifying malevolent parties. Fingerprinting techniques use a privacy-preserving approach, scanning and computing intersections of data for leak detection. Probabilistic methods calculate the likelihood of a user being malevolent, offering advantages in identifying leakers unaffected by shared data modifications. However, drawbacks include compromised data in cryptographic methods, loss of identification capability in watermarking, limited detection scope in fingerprinting, and the inefficiency of some probabilistic approaches

Limitations of Existing System

- **Cryptographic Method:** Once the key used is revealed, no one can stop the data from being compromised. It is also unable to identify the malicious party.
- **Fingerprinting Method:** It can only identify leaks that are purposefully caused by individuals. The method's inability to pinpoint the malicious entity responsible for data leaks is another drawback.
- **Probabilistic Method:** Despite the existence of probabilistic approaches that deal with explicit data requests, none of them provide actual results for sample data requests.

3. Proposed Method

The proposed method introduces a web portal for encrypting and sharing files on a cloud server. Users log in to encrypt and upload files, with access data tracking user and file interactions. User IDs embedded in accessed files aid in tracing the source of leaks. Admin- controlled user registration adds an extra security layer, and since all files are encrypted, they remain secure and unmodifiable without downloading. This method provides a unique approach to tracing guilty

agents and enhances data security through encrypted file management.

Proposed Algorithm

- **Start:** The process starts on the Homepage.
- **Login:** The user logs in to the system.
- **Authentication:** The system authenticates the user.

```

if ($_POST['user'] == "admin" && $_POST['pwd'] == "admin1234") {
    include("Admin.html");
} else {
    while ($row = mysqli_fetch_array($sql)) {
        if ($row['username'] == $_POST['user'] && $row['password'] == $_POST['pwd']) {
            include("UserAccount.php");
            $flag = 1;
            break;
        }
    }
    if ($flag == 0) {
        echo "Incorrect credentials!";
    }
}

```

4. User Account: Based on the user's role (admin or user), different options are available:

Admin

- **Add User:** The admin can add a new user to the system.
- **Similarity of Documents:** The system calculates the similarity between upload file and other documents using cosine similarity.

```

//Calculating d1 and d2
$d1d2=0;
foreach($doc1 as $y=>$y_value){
    if(array_key_exists($y,$doc2)){
        $d1d2 += ($y_value*$doc2[$y]);
    }
}

//Calculating result
if($d1==0 || $d2==0)
    $res=0;
else
    $res = ($d1d2/($d1*$d2))*100;
// echo "<h3> CosineSimilarityCal.php</h3>".$res;
include("CosineSimilarityResult.html");
}

```

Fig.1.Similarity of documents

- **Find Guilty:** It will find the agent who leaked the files and display their userID in the system. We can find by using the code in Fig.3.3.3
- **User Details:** If the file is leaked then the admin can view details of the users in the system by using userID.
- **Search Data from db and Display:** After entering userID the system will search the details in database and display it.

```

<?php
$connection = mysqli_connect("localhost", "root", "");
$db = mysqli_select_db($connection, 'project1');
if (isset($_POST['search'])) {
    $UID = $_POST['UID'];
    $query = "select * from user where UID = '$UID' ";
    $query_run = mysqli_query($connection, $query);
    while ($row = mysqli_fetch_array($query_run)) {
        ?>
    }
}

```

Fig.2. It will search the data from database and display it.

User

- **Upload Files:** The user can upload files to their cloud storage bucket.
- **View/Share Files:** The user can view and share files stored in their bucket.
- **Download:** The user can download files from their bucket.
- **Encryption & Decryption:** The user can encrypt or decrypt files in their bucket by using AES algorithm.

```

1  <code>{?php
2  require('aes.php');
3
4  $file = "File.pdf";
5  $dest = "File.pdf.encrypt";
6  $encryptedFile = encryption($file,"1234",$dest);
7  echo $encryptedFile;
8  $dest = "Decrypted_File.pdf";
9  $decryptedFile = decryption($encryptedFile,"1234",$dest);
10 echo $decryptedFile;
11
    </code>

```

Fig.3 Encrypting and Decrypting the files

- **Share:** The user can share files with other users.
- **Delete:** The user can delete files from their bucket.
- **Key Generation:** If the user chooses to encrypt or decrypt files, a key is generated by using SHA-256 algorithm.

```

> MajorProject > MajorProject > Login&Registration > KeyGeneration.php > ...
1  <code>{?php
2
3  reference
4  function generateKey($uid){
5  $key = "88++(%^)(%^)".$uid;
6  return hash("sha256",$key);
7
8
9
    </code>

```

Fig.4 Generate the Key for UserID

- **Store in S3:** The file is stored in an S3 bucket.
- **Upload Files to Target Users Bucket:** If the admin chooses to share files with specific users, the files are uploaded to the target users' buckets.

```

//Uploading file on Target User Bucket
try{//putting an object
    $result = $s3->putObject([
        'Bucket' => $targetBucket,
        'Key' => $sourceKeyname,
        'SourceFile' => $encryptedFile2
    ]);
} catch (Aws\S3\Exception\S3Exception $e) {
    echo $e->getMessage() . PHP_EOL;
}
    </code>

```

Fig.5 Upload the files to target user bucket

End: The process ends.

Advantages of Proposed System

- The ability to trace the guilty agent is the uniqueness of this proposed system.

- While uploading or sharing a file, the file gets encrypted and the user id is embedded in it. This creates a trail of access that can be used to trace the guilty agent.
- The proposed method provides a web portal wherein the organization's personnel must be added as a registered users by the admin to upload and to share files. This constraint provides an additional layer of security.
- Since all the files are encrypted, they cannot be accessed or modified without downloading.

System Architecture

In this system users must first create an account with a username and password. Once logged in, users can upload files to their S3 bucket. These files can be anything from documents, audios and videos. For security purposes, all uploaded files are encrypted using SHA, a secure hashing algorithm. This helps to protect the files from unauthorized access. The encrypted files are then stored in Amazon S3 buckets, which are cloud storage containers. This allows users to access their files from anywhere with an internet connection. Users can view or share any of their stored files. When a file is shared, the recipient receives a link that they can use to access it. Users can also download any of their files from the cloud storage buckets to their local device. There is also an admin panel that allows authorized users to manage user accounts and system settings. In this we can find the guilty agent by using user Id's from original file and leaked file. After finding the guilty agent it will show the details of the user who leaked the files.



Fig.6. Architecture of system

4. Creation of S3 Bucket

AWS stands for Amazon Web Services in its entire form. For creating the S3 bucket we are using AWS cloud. It is a platform that provides scalable, adaptable, user-friendly, and affordable cloud computing solutions. AWS is an extensive, user-friendly computing platform provided by Amazon. The platform is created using a combination of packaged software as a service (SaaS), infrastructure as a service (IaaS), and platform as a service (PaaS) solutions. S3 bucket, a service provided by AWS, is used to store files.

AWS's Simple Storage Service (S3): It is an object storage service, includes a public cloud storage resource known as an Amazon S3 bucket. The objects that are stored in Amazon S3 buckets, which resemble file folders, are made up of data and the metadata that describes it.

5. AES Algorithm

The Advanced Encryption Standard (AES) was made by a group called NIST back in 2001 to keep electronic information safe and secret. Even though it's a bit harder to make, people still use AES a lot because it's way tougher to crack than older methods like DES and triple DES. AES works by taking chunks of data, each with 128 bits, and turning them into secret code. It can use different lengths of secret keys, from 128 to 256 bits, depending on how secure you want things to be. When AES does its job, it takes 128 bits of data in and spits out 128 bits of secret code. It does this by mixing up the data in a special way called a substitution-permutation network, which is like a big puzzle that scrambles everything up to keep it safe from prying eyes.

Encryption

It reads 16-byte blocks from the source file, encrypts them using AES-256 in CBC mode with the key and IV, and writes the encrypted blocks to the destination file, using the previous ciphertext block as the IV for the next block.

```
function encryption($source, $key, $iv) {
    // Create random IV (16 bytes)
    $iv = substr(md5($key), 0, 16);
    $iv = openssl_random_pseudo_bytes(16);
    $iv = hex2bin($iv);

    // Open the file to be encrypted as the file
    $file = fopen($source, "r");
    if (!$file) {
        return "Error: Cannot open source file";
    }

    // Read the first 16 bytes of the file
    $data = fread($file, 16);
    $cipher = openssl_encrypt($data, "aes-256-cbc", $key, OPENSSL_RAW_DATA, $iv);
    // Write the first 16 bytes of the ciphertext to the destination file
    $file = fopen($destination, "w");
    fwrite($file, $cipher);
    return $cipher;
}
```

Fig.7. Encryption of files

Decryption

The code extracts the first 16 bytes from the SHA-256 hash of the \$key argument to use as the encryption key, and then reads the first 16 bytes from the source file to use as the initialization vector (IV).The code reads 16-byte blocks from the source file, decrypts them using the AES-256-CBC algorithm with the key and IV, writes the decrypted blocks to the destination file, and updates the IV for the next block using the first 16 bytes of the current ciphertext.

```
function decryption($source, $key, $iv) {
    // Create random IV (16 bytes)
    $key = substr(md5($key), 0, 16);
    $iv = hex2bin($iv);

    // Open the file to be decrypted as the file
    $file = fopen($source, "r");
    if (!$file) {
        return "Error: Cannot open source file";
    }

    // Read the first 16 bytes of the ciphertext
    $data = fread($file, 16);
    // Decrypt the first 16 bytes of the ciphertext using the key and IV
    $cipher = openssl_decrypt($data, "aes-256-cbc", $key, OPENSSL_RAW_DATA, $iv);
    // Write the first 16 bytes of the plaintext to the destination file
    $file = fopen($destination, "w");
    fwrite($file, $cipher);
    return $cipher;
}
```

Fig.8 Decryption of files

6. SHA-256 Algorithm

The SHA-256 algorithm is a cryptographic method used to secure digital information by generating a unique, fixed-size hash value from input data. This hash value is a string of 64 characters, made up of numbers and letters, that represents the original data in a way that's virtually impossible to reverse-engineer. Think of it as a digital fingerprint: no matter how big or small the input data, SHA-256 always produces a fixed-length output. This makes it highly reliable for ensuring data integrity and authenticity. Due to its robustness and resistance to hacking attempts, SHA-256 is widely utilized in various security applications, including password hashing, digital signatures, and blockchain technology.

This file uses "sha-256" algorithm with the user id to generate a key to encrypt the files.

```
<? Php
function generateKey($uid) {
    $key = "&&**(%^)" . $uid; return hash("sha256", $key);
}
?>
```

7. Cosine Similarity

Cosine similarity is a statistic that may be used to assess how similar data objects are, regardless of size. Python's Cosine Resemblance function can be used to compare two texts for similarity. Cosine similarity treats each data object in a dataset as a vector. The following is the formula to determine the cosine similarity between two vectors:

$$\text{Cos}(x, y) = \frac{x \cdot y}{\|x\| * \|y\|}$$

where,
 $x \cdot y$ = product (dot) of the vectors 'x' and 'y'.
 $\|x\|$ and $\|y\|$ = length of the two vectors 'x' and 'y'.
 $\|x\| * \|y\|$ = cross product of the two vectors 'x' and 'y'.

The cosine similarity is advantageous because it allows for the possibility that there may be a smaller angle between two similar data objects that are separated by a large Euclidean distance due to their size. The resemblance increases with decreasing angle. The cosine similarity, when plotted on a multi-dimensional space, captures the direction (the angle) rather than the magnitude of the data objects.

8. FFMpeg:

FFmpeg is a free and open-source software paper consisting of a suite of libraries and programs for handling video, audio, and other multimedia files and streams.

At its core is the command-line ffmpeg tool itself, designed for processing of video and audio files. It is widely used for format transcoding, basic editing (trimming and concatenation), video scaling, video post-production effects and standards compliance

IV. RESULT



Fig.9. Similarity of two documents

This page is designed to check the similarity between two pdf documents based on the concept of cosine similarity. This displays the percentage of similarity between those two documents.



Fig.10. Results of similarity of documents

The figure 4.2 shows the system behave when it checks for the similarity between the same documents.



Fig.11. Uploading leaked file

This is the page where the admin can identify the guilty agent who leaked the file by uploading theselected file. It displays the respective branch of user ids through which the file is shared



Fig.12. Finding leaking agent

Here the user with id 4 is identified as the guilty agent. The users with ids 1, 4, 2 have accessed this video file. User 1 is the root user who uploaded the file, further it is shared to user

Now user 4 have shared this file to user 2 and hence user 4 is guilty agent.



Fig.13. Search user by user id



Fig.14. User details

V. CONCLUSION

This paper introduces a PHP-based web interface, ensuring a robust and swift system for storing and sharing PDF files. Utilizing the AES algorithm for file encryption guarantees high-level security. The adoption of user-specific AWS S3 buckets for storing user files ensures seamless storage and sharing of large volumes of data. The integration of unique user IDs during file upload and sharing establishes a traceable access trail, aiding in identifying the source of data leaks. The system not only enables the verification of the guilty party's identity but also imposes an additional layer of security by requiring explicit administrator approval for organization personnel to access the system. Access to AWS S3 buckets is restricted to the individual holding the AWS Credentials, bolstering the overall security of the system.

Future Scope

The paper can also be expanded to add access roles while sharing the files to include users who may share a certain file and those who may not. This paper has the potential to be developed to work with large enterprises where each department would have an administrator user and the administrators of all the departments would be managed by a „master“ administrator. It can be developed to create a hierarchy of user roles as well.

REFERENCES

1. Meshram, S. D., & Chavan, H. K. Agent Guilt Model And Fake Object Distribution for Identifying Data Leakage.

2. Huliurdurga, K., Jadhav, S., Patil, R., & Lawand, D. (2022). Data Outflow Detection with Guilty Agent Tracking.
3. Yellamma, P., Kumar, P. D., Reddy, K. S. P., Harsha, L. S., & Jagadeesh, N. (2020). Probability of data leakage in cloud computing. *International Journal of Advanced Science and Technology*, 29(6), 3444-3450.
4. Singh, A., & Anand, A. (2017). Data leakage detection using cloud computing. *International Journal Of Engineering And Computer Science*, 6(4).
5. Gupta, I., & Singh, A. K. (2018). A probabilistic approach for guilty agent detection using bigraph after distribution of sample data. *Procedia Computer Science*, 125, 662-668.
6. Thorat, B. D., & Devale, P. R. (2014). Identifying Guilty Agent for Data Leakage Detection System. *International Journal*, 2(3).
7. Papadimitriou, P., & Garcia-Molina, H. (2010). Data leakage detection. *IEEE Transactions on knowledge and data engineering*, 23(1), 51-63.