A

PROJECT REPORT ON

DEVELOPMENT OF SECURE FOLDER APP FOR DESKTOP WITH ONE-PASSWORD AUTHENTICATION

BY

TAJUDEEN TAIW SAHEED

CSC/18/5867

SUPERVISED BY:

DR. OSUOLALE A. FESTUS

SUBMITTED TO:

THE DEPARTMENT OF COMPUTER SCIENCE,

SCHOOL OF COMPUTING,

THE FEDERAL UNIVERSITY OF TECHNOLOGY, AKURE,NIGERIA.

IN PARTIAL FULFILLMENT OF REQUIREMENTS FOR THE AWARD OF BACHELOR OF TECHNOLOGY (B.TECH) IN COMPUTER SCIENCE

CHAPTER ONE

INTRODUCTION

1.1 Background to the Study

Nakkeeran (2015) states that the proliferation of networked devices and internet services has led to concerns regarding the security of data stored on desktop computers. Bawa(2012) suggests that traditional computing methods rely on portable storage media and machine robustness for security, whereas cloud computing offers a distributed environment with enhanced security features.

Alshsayeji et al. (2018) highlight the importance of addressing reliability issues in cloud computing systems, as service outages can result from failures, which can affect the reliability and availability of cloud services. Kalanirnika and Sivagami (2015) propose proactive fault tolerance policies to predict and replace faults before they occur, ensuring the continuous operation of cloud computing systems.

Deng et al. (2012) emphasize the need for reliable fault tolerance techniques in cloud computing, particularly for scientific computations where accuracy and dependability are critical. Paul and Visuswasm (2012) introduce a smart checkpoint infrastructure for cloud service providers to enhance fault tolerance in real-time computing scenarios. Guler and Ozkasap (2017) develop compressed incremental checkpointing algorithms to reduce communication costs and improve throughput in replicated key-value stores deployed on cloud infrastructure. Sutaria et al. (2017) conduct a comprehensive survey on fault prediction and mitigation techniques in cloud computing, highlighting the importance of proactive measures to enhance system reliability.

In light of this findings, it is evident that ensuring the security and reliability of data stored on desktop computers, particularly in cloud-based environments, required robust fault tolerance mechanisms.

1.2 Motivation

The necessity for improved data security and privacy in contemporary computer environments led to the development of a desktop secure folder application with one-time password authentication. The weaknesses of conventional password-based authentication systems, which can be breached by phishing, brute force attacks, or password theft, have been brought to light by earlier cyber security research.

Researchers and Developers have looked into several authentication techniques to strengthen security in order to address these worries. One such technique is authentication using a one-time password (OTPs), methods that rely on time, event-based triggers, or a mix of the two are employed to make sure that every password is distinct and cannot be used again by hackers.

The need to safeguard private or sensitive files from unauthorized access is the driving force behind the integration of OTP authentication into a desktop secure folder application. By requiring users to enter a one-time password generated through a secure mechanism, the app adds an extra layer of security beyond traditional password protection, mitigating the risk of password theft or interception, as the OTPs are not reusable. The development of this type of application was made possible by prior research in secure file storage and access control, which looked into methods like encryption, access control lists, and biometric authentication to secure files and folders on desktop systems. However, the incorporation of OTP authentication adds a new dimension to the process enhancing resilience against various attack vectors and insider threats.

Furthermore, the necessity of protecting data kept on local devices has grown due to the spread of cloud storage services and remote work policies. Instead of depending exclusively

on cloud-based security measures, users can protect important data in their desktops with ease and strength by using a desktop secure folder application with OTP authentication.

In conclusion, the need to bolster data security in the face of emerging cyber dangers is the driving force for the creation of a desktop safe folder app with one-time password authentication. The program improves the confidentiality and integrity of saved files by using OTPs as an extra authentication element, making sure that only authorized users may access important data. This expands on earlier cyber security ad access control studies, providing a proactive method of safeguarding data on desktop systems in an increasingly interconnected digital landscape.

1.3 Objectives

The specific objectives of the research are to:

a. design a secure folder app using one-time password authentication;
b. implement (a); and
c. evaluate the performance of the developed application based on performance metrics.

1.4 Methodology

The proposed application comprises four(4) layers namely the user, task controller, fault detector, and fault tolerance layers. In the user layer, the use submits a request (R) along with quality of service (QoS) parameters via cloud clients to the controller layers as follows:

1. **User Layer:**
    a. User submits a request (R) comprising a set of tasks (S) and QoS parameters.

b. Request (R) includes tasks such as inventory taking, data analysis, and language processing.

c. QoS parameters include turnaround time, monetary cost, and computing resources.

2. **Task controller Layer**:

   (a) Houses the service manager (Sm) and the fault tolerance scheduler module (Fm).

   (b) Service manage(Sm) checks the database (Db) for computing capacity (cc), storage capacity(sc), usage history(uh), and failure history(fh).

   (c) Virtual Machine and forwards it to the scheduler module.

3. Fault Detector and Fault Tolerance Layer:

   (a) Fault detector module checks VM reliability based on failure probability.

   (b) VMs are ranked according to failure probability values.

   (c) Fault tolerance scheduler module selects appropriate fault tolerance technique (replication or check pointing) based on VM rankings.

4. Replication and Checkpointing:

(a) Replication module creates replicas of the request on the VM with the least failure probability.

(b) Number of replicas (p) for a request is determined dynamically based on failure probability thresholds.

(c) Check-pointing module saves request data for resumption in case of failure

5. FTScheduler Algorithm:

(a) Determines fault tolerance technique (replication or checkpointing) based on VM availability and request type.

(b) Utilizes a partitioned check-pointing strategy for optimal check-pointings.

6. Evaluation:

(a) The proposed model will be simulated using CloudSim.

(b) Evaluation metrics include throughput response time, robustness, and trend analysis.

By following this methodology, the secure desktop folder app can be developed with robust fault tolerance mechanism to ensure data integrity and availability.

Mathematical model:

Let's denote the total time required for syncing notes with cloud storage as $T_{cloud\_sync.}$

The model can be represented as follows:

$T_{cloud\_sync} == B_{size} X (T_{upload} + T_{download})^{1.14}$

Where:

Bsize is the storage capacity provided by the cloud service.

T_{upload} and $T_{download}$ are functions of internet speed (x),

Server response time(y), and network congestion level(z).

The functions T_{upload} and $T_{download}$ can be represented as:

$T_{upload,} T_{download} = f(x,y,z)$

Now, let's break down the components:

(1) Internet speed(x): This factor represents the upload and download speeds of the user's internet connection. Higher speeds result in faster syncing times.
(2) Server response time(y): This factor indicates how quickly the cloud server responds to requests from the application. Lower response times lead to faster syncing.
(3) Network congestion level(z): This factor reflects the congestion level of the network through which data is being transmitted. Higher congestion levels can slow down data transfer.

1.5 Organization of Project

The rest of this project is organized as follows:

Chapter Two presents the related works and extensively reviewed existing literature, to investigate existing loopholes and justify the need to carry out this research. Chapter Three discusses the methodology used in the design and the overall analysis of the system. Chapter Four presents the implementation and results and evaluation while Chapter Five concludes the research with recommendations drawn from this research and the contributions made to knowledge.

1.6 References

Soares, L.R. (2015). Android Programming for Beginners. Packt Publishing

Oorschot, P.C. van, & Stubblebine, S.G. (2012). Password Authentication with Insecure Communication. Springer.

Jovanovic, M., & kocovic, A. (2015). Secure Password Authentication Protocol Using One-Time Passwords. CRC Press.

Cryptanalysis of the On-Time Pad. IEEE Secuirty & Privacy, 11(1), 16-21.

Stajano, F., & Anderson, R. (1999). "The Resurrecting Duckling: Security Issues for Ad-hoc Wireless Networks." Security Protocols Workshop.

Perrig, A., et al (2001). Turtles all the way down: Research challenges in user authentication. Proceedings of the 2nd USENIX Workshop on Hot Topics in Security.

Corley, C.S., et al. (2015). Towards an understanding of security decision making in mobile app development. In 2015 IEEE/ACM 37th IEEE International Conference on software Engineering.

Mitchell, J.C. (1997). Foundations for Programming Languages. MIT Press.

Ross, A., & Jain, A. (200). BioMetrics: A Tool for Information Security. IEEE Transactions on Information Forensics and Security, 1(2), 125—143.

Janiszewski, J.A, et al. (s019). Effects of biometric fingerprint authentication on smartphones and tablets. International Journal of Human-Computer Interaction, 35(4-5), 353-366.

Wheeler, D.A., & Gifford, R.D. (2002). Recovery of Passowords Using Improved Guessing. In 11th USENIX Security Symposium.

Google Developers. (2022). Fingerprint Authentication | Android Developers. Retrieved from https:://developer.android.com/training/sign-in/biometric-auth#kotlin

Reschke, J. K. (2012). The OAuth 2.o authorization Framework. RFC6749.

Katzenbeisser, S., & Peticolas, F.A>P. (2000). Information Hiding Techniques for Steganography and Digital Watermarking. Artech House.

Scheier, B. (1996). Applied Cryptography. John Wiley & Sons.

Smith, J., Johnson, R., & Williams, E. (2018). Security vulnerabilities I note-taking applications. Journal of Cybersecurity, 5(2), 123-137.

Jones, A., & Brown, M. (2019). User perceptions of security in note-taking applications: A survey study. Information Security Journal, 28(4), 321-335.

Huang, K., et al. (2018). Fingerprint authentication system: A review. In 2018 7th Int. Conf. Inf. Commun. Technol. Electron. Appl.ICCEA.

Patel, S., Gupta, N., & Kumar, A. (2020). Securing note-taking applications with biometric authentication and blockchain technology. International Journal of Information Security, 19(3), 265-279.

Wang, Q., & Chen, L. (2021). Usability of biometric authentication in mobile applications: A comparative study. Human-Computer Interaction Research, 10(2), 145-159.

Lee, K., Park, S., & Kim, D. (2022). Security implications of cloud synchronization in note-taking applications

Schulz, A. (2017). “Electron: From Beginner to Pro.” Apress.

Reilly, J. (2019). “Electron in Action.” Manning Publications.

Smith, C. (2018). “Building Cross-Platform Desktop Applications with Electron.” Packt Publishing.

Malhotra, P., & Vashishtha, S. 92018). Development of Desktop Applications using Electron Framewok Journal of Engineering Research & Technology, 7(12), 228-232.

.