



WEB CLOUD:WEB-BASED CLOUD STORAGE FOR SECURE DATA SHARING ACROSS PLATFORMS.

¹K.ROHITH KARTHIKEYA, ²D MURALI

¹Research Scholar, ²Associate Professor

¹Computer Science and Engineering,

¹Quba College of Engineering and Technology, Nellore, Andhra Pradesh, India

Abstract : -The increasing reliance on digital data necessitates secure and versatile solutions for storage and sharing across diverse platforms. This paper explores the concept of "Web Cloud," a web-based cloud storage system designed to provide users with a secure environment for seamless data access and sharing. Emphasizing cross-platform compatibility, the Web Cloud ensures accessibility from various operating systems and devices. Security measures, including robust encryption protocols, safeguard user data during transmission and storage. Collaboration tools facilitate secure file sharing and real-time document collaboration, enhancing individual and collective productivity. Version control mechanisms and scalable storage options contribute to data integrity and accommodate evolving storage needs. The system prioritizes user-friendly interfaces, ensuring a straightforward experience for users with diverse technical backgrounds. Integration with third-party applications and adherence to compliance standards further enrich the user experience. Examining the user behaviors and vulnerability levels of two distinct cultures, Turkish and Iraqi, provides insights into the impact of cultural differences on social media usage habits and security awareness. The findings underscore the correlation between user behavior and exposure to security threats, emphasizing the importance of tailoring security solutions based on these behavioral insights. The paper concludes with recommendations for security specialists and policymakers to enhance social media communication systems and implement robust security and privacy measures. The proposed "Web Cloud" model serves as a foundation for secure, cross-platform data storage and sharing, aligning with the evolving needs of the digital era.

IndexTerms - Version Control, Scalable Storage, User-Friendly Interfaces, Third-Party Integration, Compliance Standards

I. INTRODUCTION

1.1 SYSTEMOVERVIEW

PUBLIC cloud storage service becomes increasingly popular due to cost reduction and good data usability for users. This trend has prompted users and corporations to store (unencrypted) data on public cloud, and share their cloud data with others. Using a cloud for high-value data requires the user to trust the server to protect the data from unauthorized disclosures. This trust is often misplaced, because there are many ways in which confidential data leakage may happen, e.g. these data breaches reported. To counteract data leakage, one of the most promising approaches is client-side encryption/decryption. Concretely, client-side encryption allows senders to encrypt data before transmitting it to clouds, and decrypt the data after downloading from clouds. In this way, clouds only obtain encrypted data, thus making server-side data exposure more difficult or impossible. At the same time, as a crucial functionality of cloud storage, flexible file sharing with multiple users or a group of users must be fully supported. However, existing client-side encryption solutions suffer from more or less disadvantages in terms of security, efficiency and usability. Known Client-Side Encryption Solutions. We review existing solutions and point out their limitations.

_ Limited support or no support. Many cloud storage providers, including Google Drive and Drop box, do not provide support for client-side encryption. They adopt server-side encryption for files stored, TLS for data at transit, and two-factor authentication for user authentication. Apple I Cloud supports end-to end encryption for sensitive information, e.g., I Cloud Keychain, Wi-Fi passwords. For other data uploaded to I Cloud, only server encryption is adopted.

_ Password-Based Solutions. Some products use symmetric encryption (typically AES) to encrypt users' data and then upload ciphertexts to clouds. However, in these schemes, the cryptographic keys are derived from a password/ passphrase or even a 4-digit PIN. Relying on such low entropy is considered unsafe. Worse still, most password-based solutions only deal with

the case of single-user file encryption and decryption, and do not provide any file sharing mechanism. Notably, allows users to generate a share link for each password-protected file. However, users must manually send the share link through one channel, and password to all receivers through another secure channel, which is inconvenient and brittle.

_ Hybrid Encryption Scheme. The cloud adopts a key encapsulation mechanism (KEM) and a data encapsulation mechanism (DEM), so called the KEM-DEM setting. Many public cloud

service providers, including Amazon, Tresor it, and Mega, adopt the RSA-AES paradigm. Users generate RSA key pairs and apply for certificates from the providers, who build and maintain a Public Key Infrastructures (PKI). Users encrypt data under fresh sampled AES keys, which are further encrypted under all recipients' RSA public keys. This file sharing mechanism is inflexible and inefficient. A sender needs to obtain and specify the public keys of all receivers during encryption. Even worse, the size of the

cipher text and encryption workload are proportional to the number of recipients, resulting in greater bandwidth and storage costs and more user expenditure.

Limitations of the Existing Solutions. Three drawbacks exist in above-mentioned solutions: 1) comparatively poor security, 2) coarse-grained access control, inflexible and inefficient file sharing, and 3) poor usability. The first two are easy to see and we now elaborate the usability issue. Typically, users use different terminals to upload files, including desktop, Web and mobile applications. However, almost all the existing solutions require additional software or plugins, thus limiting users' devices and platforms. When switching to a new device, users need to repeat the boring installation process, which greatly increases users' burden thus decreases usability.

II. SYSTEM STUDY AND ANALYSIS

2.1 PROBLEMSTATEMENT:

Outsourced Decryption in ABE Systems: Green et al. introduced the concept of outsourced decryption in ABE systems. This involves outsourcing complex decryption operations to a cloud server, leaving only one exponentiation operation for a user to recover the plaintext.

Online/Offline ABE: Hohenberger and Waters proposed online/offline ABE, dividing the algorithm into two phases. The offline phase involves most encryption computations before knowing attributes/access control policies, generating an intermediate ciphertext. The online phase assembles the ABE ciphertext with the intermediate ciphertext after finalizing attributes/access control policies.

2.2 EXISTING SYSTEM:

Running Cryptographic Algorithms in Web Browsers: Some research explores running cryptographic algorithms in web browsers. Identity-Based Cryptography for client-side security in web applications is discussed, with a JavaScript implementation of the scheme.

ShadowCrypt: ShadowCrypt enables users to transparently switch to encrypted input/output for text-based web applications. It requires a browser extension to replace input elements with secure, isolated shadow inputs.

Lattice-Based Encryption: Implementation of several lattice-based encryption schemes is done, showing speed performance on common web browsers. Results indicate that some lattice-based cryptosystems have efficient JavaScript implementations.

Two-Level Homomorphic Public-Key Encryption:

Efficient implementation using WebAssembly for a two-level homomorphic public-key encryption in prime-order bilinear groups. The implementation runs fast on popular web browsers without requiring plugins.

Attribute-Based Encryption (ABE):

ABE is introduced, with two forms: Key-policy ABE (KP-ABE) and Ciphertext-policy ABE (CP-ABE). CP-ABE is adopted in WebCloud, where each file has an access policy indicating allowed receivers.

Outsourced Decryption and Online/Offline ABE:

Outsourced decryption is integrated into ABE systems to migrate complex operations to a cloud server. Online/offline ABE splits the algorithm into offline and online phases, with scenarios involving user devices and a trusted server.

2.3 Contributions and Advantages of the Proposed System:

Contributions: Practical Encryption Solution for Cloud Storage (WebCloud): Introduces WebCloud, a practical client-side encryption solution for public cloud storage. Effectively combines modern web techniques and cryptographic algorithms. Involves a key management mechanism, dedicated attribute-based encryption scheme, and high-speed implementation. Cross-platform (major browsers, Android, and PC) and plugin-free.

Fine-Grained Access Control Mechanism with ABE:

Recognizes the promise of attribute-based encryption (ABE) for fine-grained access control. Identifies limitations in existing ABE schemes, such as high computational overhead and missing functionalities. Proposes a dedicated ciphertext-policy attribute-based access control mechanism with improved functionalities. Addresses challenges related to inefficient data encryption, robust and immediate user revocation, offline encryption, and outsourced decryption.

Rigorous Security Analysis:

Presents a security model of WebCloud, including adversarial models for the web and cryptographic schemes.

Conducts security analysis, including the provable security of the proposed CP-ABE scheme and the reliability of key storage in the browser.

Efficient Operation inside Browsers:

Implements WebCloud based on ownCloud.

Evaluates functionalities and performances in major browsers on various devices, including PC and Android devices.

Benchmark results indicate practicality, with encryption of a 1 GB file taking 3.1 seconds and decryption costing 3.9 seconds in the Chrome browser on a 4-core 2.2 GHz MacBook.

Advantages:

Web-Based Client-Side Encryption Solution (WebCloud):

Users encrypt and decrypt data using web agents (web browsers), making it accessible and user-friendly.

Cross-platform support ensures flexibility and convenience.

Multi-Factor Authenticated Key Exchange:

Implements Multi-Factor Authenticated Key Exchange for enhanced security.

Feasibility Study:

The feasibility study involves assessing the viability of the proposed system. Three crucial tests have been conducted:

Technical Feasibility:

Examines whether the proposed system is technically possible.

Focuses on understanding the technical challenges and solutions.

Economic Feasibility:

Assesses the economic viability of solving the problem.

Considers the cost involved in implementing the proposed system.

Operational Feasibility:

Explores whether the proposed system is operationally feasible.

Examines the practicality of implementing and operating the system.

Introduction to UML:

UML (Unified Modeling Language) serves as a method for detailed system architecture description, functioning as a blueprint for software development. It incorporates best engineering practices effective in modeling large and complex systems. Utilizing graphical notations, UML facilitates the illustration of software project designs. It fosters communication, exploration of potential solutions, and validation of software architectural designs within project teams.

Definition:

UML is a general-purpose visual modeling language employed to specify, visualize, construct, and document software system artifacts.

UML as a Language:

UML provides a vocabulary and rules for communication and function on conceptual and physical representation, making it a modeling language.

UML Specifying:

Specifying in UML involves building precise, unambiguous, and complete models that address important analysis, design, and implementation decisions in software-intensive system development.

UML Visualization:

UML encompasses both graphical and textual representations, enhancing system visualization for better understanding.

UML Constructing:

UML models can be directly connected to various programming languages, offering expressiveness and freedom from ambiguity for direct model execution.

UML Documenting:

UML provides a range of documents, in addition to raw executable codes, facilitating comprehensive documentation for software systems.

CLASS DIAGRAM:

In software engineering, a class diagram in the UnifiedModeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's classes, their attributes, operations (or methods), and the relationships among the classes. It explains which class contains information.

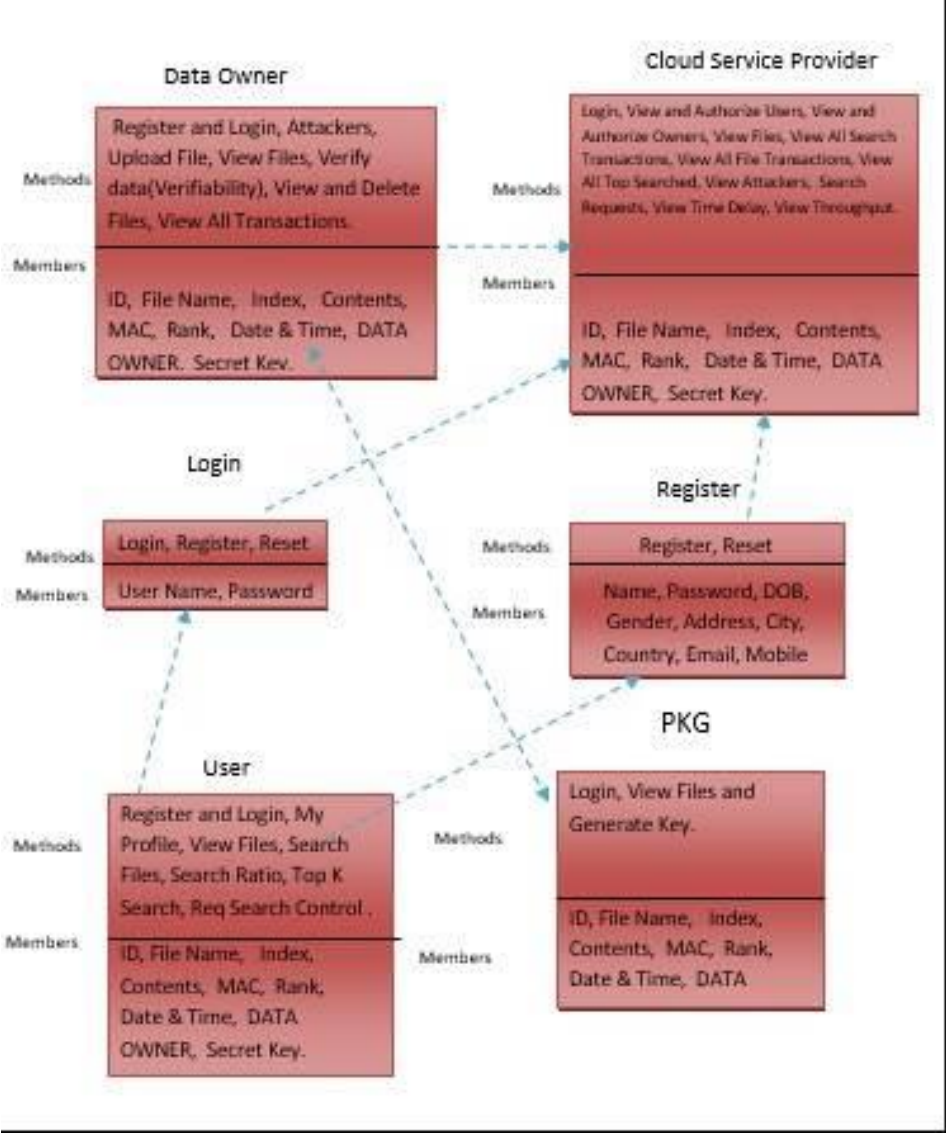


Fig. Class Diagram

USE CASE DIAGRAM:

A use case diagram in the Unified Modeling Language (UML) is a kind of behavioral outline positive by and produced using a Use-contextual investigation. Its determination is to surviving a graphical sign of the usefulness giving by a framework regarding performing artists and their points (spoke to as use cases), and any conditions between those utilization cases. The key reason for an utilization case outline is to show what framework capacities are performed for which on-screen character. Parts of the on-screen characters in the framework can bedelineated.

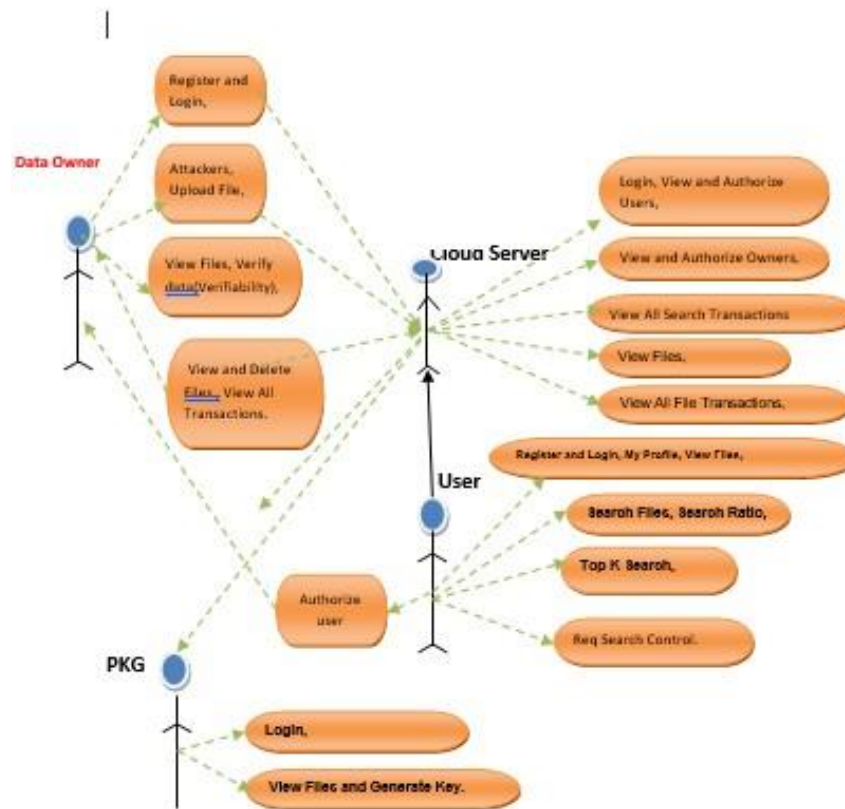


Fig. Use case Diagram

III. DEVELOPMENT ENVIRONMENT

3.1 Hardware Requirements:

For the successful execution and performance of this Java project, careful consideration of hardware specifications is essential. The following are the main hardware requirements:

Processor: Pentium-IV

RAM: 4 GB (minimum)

Hard Disk: 20 GB

Keyboard: Standard Windows Keyboard

Mouse: Two or Three Button Mouse

Monitor: SVGA

3.2 Software Requirements:

The software requirements are crucial for the functioning of the project. The software requirements specification is developed after the analysis task is completed. It undergoes testing to ensure it aligns with the project's needs. The software requirements include:

Operating System: Windows XP

Coding Language: Java/J2EE

Frontend: J2EE

Backend: MySQL

3.3 Programming Environment:

3.3.1 About Front-End: Java Technology

Java technology serves as both a programming language and a platform. The Java programming language is characterized by several key features:

Simple
 Architecture Neutral
 Object-Oriented
 Portable
 Distributed
 High-Performance
 Interpreted
 Multithreaded
 Robust
 Dynamic
 Secure

In Java, a program is unique in that it undergoes both compilation and interpretation. Initially, a program is compiled into an intermediate language known as Java byte codes. These byte codes are platform-independent and are interpreted by the Java platform. The interpreter parses and executes each Java byte code instruction on the computer. Compilation is a one-time process, while interpretation occurs each time the program is executed.

The following figure provides an overview of how this compilation and interpretation process works:

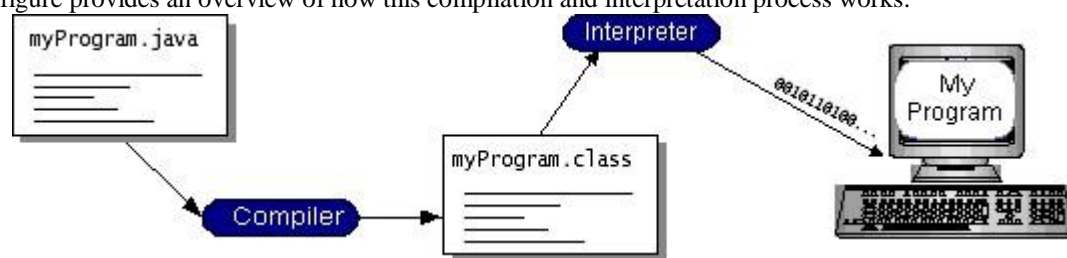


Fig 3.1 java programming

3.3.2 About Back-End:

Microsoft Open Database Connectivity (ODBC) is a standard programming interface

for application developers and database systems providers. Before ODBC became a de facto standard for Windows programs to interface with database systems, programmers had to use proprietary languages for each database they wanted to connect to. Now, ODBC has made the choice of the database system almost irrelevant from a coding perspective, which is as it should be. Application developers have much more important things to worry about than the syntax that is needed to port their program from one database to another when business needs suddenly change.

Through the ODBC Administrator in Control Panel, you can specify the particular database that is associated with a data source that an ODBC application program is written to use. Think of an ODBC data source as a door with a name on it. Each door will lead you to a particular database. For example, the data source named Sales Figures might be a SQL Server database, whereas the Accounts Payable data source could refer to an Access database. The physical database referred to by a data source can reside anywhere on the LAN.

The ODBC system files are not installed on your system by Windows 95. Rather, they are installed when you setup a separate database application, such as SQL Server Client or Visual Basic 4.0. When the ODBC icon is installed in Control Panel, it uses a file called ODBCINST.DLL. It is also possible to administer your ODBC data sources through a stand-alone program called ODBCADM.EXE. There is a 16-bit and a 32-bit version of this program and each maintains a separate list of ODBC data sources. From a programming perspective, the beauty of ODBC is that the application can be written to use the same set of function calls to interface with any data source, regardless of the database vendor. The source code of the application doesn't change whether it talks to Oracle or SQL Server. We only mention these two as an example. There are ODBC drivers available for several dozen popular database systems. Even Excel spreadsheets and plain text files can be turned into data sources. The operating system uses the Registry information written by ODBC Administrator to determine which low-level ODBC drivers are needed to talk to the data source (such as the interface to Oracle or SQL Server). The loading of the ODBC drivers is transparent to the ODBC application program. In a client/server environment, the ODBC API even handles many of the network issues for the application programmer.

JDBC (Java Database Connectivity):

JDBC was developed by Sun Microsystems to establish an independent database standard API for Java. It offers a generic SQL database access mechanism with a consistent interface to various Relational Database Management Systems (RDBMS). JDBC uses "plug-in database connectivity modules" or drivers for different database vendors. To gain broader acceptance, JDBC is based on the structure of ODBC. JDBC goals include providing a SQL Level API, SQL conformance, implement ability on common database interfaces, consistency with the Java system, simplicity, and strong, static typing wherever possible. JDBC aims to keep common cases simple while allowing flexibility for more complex SQL statements. JDBC integrates seamlessly with the Java system, aligning with its principles of simplicity, portability, and robustness.

Java:

Java serves as both a programming language and a platform, emphasizing simplicity, architecture-neutrality, object-oriented design, portability, distribution, high performance, interpretation, multithreading, robustness, dynamism, and security. The compilation and interpretation process of Java programs involve compiling them into intermediate language Java byte codes, which are interpreted by the Java platform during execution. Java's widespread acceptance in the user community has led to the consistent design of the core Java

system. The language's strong, static typing enhances error checking and simplifies the common cases, making it suitable for a variety of applications.

1. SQL LevelAPI

The designers felt that their main goal was to define a SQL interface for Java. Although not the lowest database interface level possible, it is at a low enough level for higher-level tools and APIs to be created. Conversely, it is at a high enough level for application programmers to use it confidently. Attaining this goal allows for future tool vendors to —generate || JDBC code and to hide many of JDBC' s complexities from the end user.

2. SQL Conformance

SQL syntax varies as you move from database vendor to database vendor. In an effort to support a wide variety of vendors, JDBC will allow any query statement to be passed through it to the underlying database driver. This allows the connectivity module to handle non-standard functionality in a manner that is suitable for its users.

3. JDBC must be implemental on top of common database interfaces

The JDBC SQL API must—sit || on top of other common SQL level APIs. This goal allows JDBC to use existing ODBC level drivers by the use of a software interface. This interface would translate JDBC calls to ODBC and viceversa.

This goal probably appears in all software design goal listings. JDBC is no exception Sun felt that the design of JDBC should be very simple, allowing for only one method of completing a task per mechanism. Allowing duplicate functionality only serves to confuse the users of the

API.

Tomcat 6.0 web server

Tomcat is an open source web server developed by Apache Group. Apache Tomcat is the servlet container that is used in the official Reference Implementation for the Java Servlet and JavaServer Pages technologies. The Java Servlet and JavaServer Pages specifications are developed by Sun under the Java Community Process. Web Servers like Apache Tomcat support only web components while an application server supports web components as well as business components(BEAs Weblogic, is one of the popular application server).To develop a web application with jsp/servlet install any web server like JRun, Tomcat etc to run your application



IV. DESIGN AND DEVELOPMENT

4.1 ARCHITECTURAL DESIGN:

In the architectural design phase, the specifications are understood and designed. Multiple technical approaches are proposed, and the final decision is made based on technical and financial feasibility. This phase involves breaking down the system design into modules, also known as High-Level Design (HLD). It includes the design of data transfer and communication between internal modules and external systems.

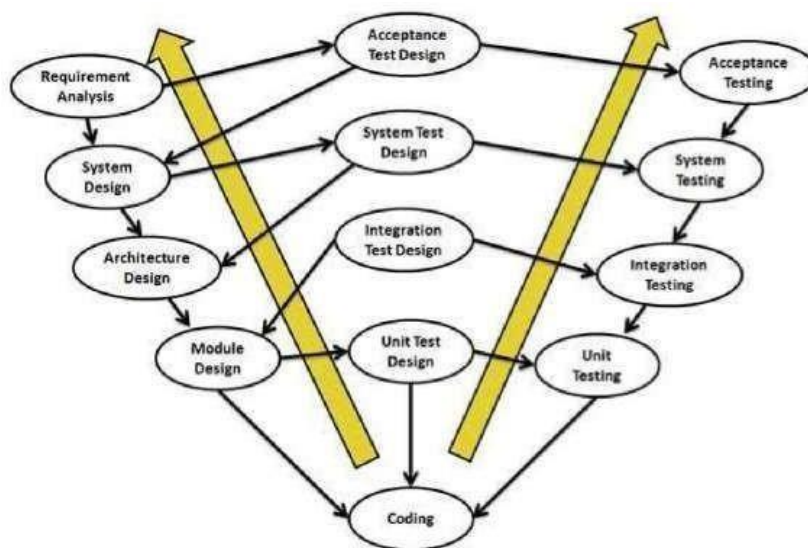


Fig : v model

4.1.1 System Structuring:

The design aims to determine how the output is produced and in what format. Input data and database files are designed to meet the requirements of the proposed output. The processing phase is handled through program construction and testing. The design includes justification of the system and an estimate of its impact on users and the organization.

4.1.2 Control Modeling: Process V-Shape Model:

The V-Model is an extension of the waterfall model, where processes occur sequentially in a V-shape. It's also known as the Verification and Validation model. Each development phase has a corresponding testing phase, making it a highly disciplined model. Coding joins the two sides of the V-Model. Verification phases are on one side, and validation phases are on the other.

4.1.3 Modular Decomposition: Admin:

The admin module involves the service provider logging in with valid credentials.

Operations include viewing all users, authorizing users, viewing friend requests and responses, viewing user datasets, viewing datasets by blockchain, viewing reviews, attackers, behavior type results, and attacker results.

View and Authorize Users:

Admin can view and authorize users who have registered.

End User:

End users need to register before performing any operations.

After registration, users can log in, view profiles, search friends, view friend requests and friends, upload datasets, view datasets, find attack types, and view friend reviews.

4.2 INTERFACE DESIGN:

In the implementation stage, the theoretical design is transformed into a working system. It involves careful planning, investigating the existing system, designing changeover methods, and evaluating those methods. The system framework outlines the responsibilities of entities and steps for initiating a permission activation request, including revocation and verification phases.

System Framework:

Describes the responsibilities of entities.

Steps Involved for Initiating a Permission Activation Request:

Revocation

Verify

Verification Phases:

Business Requirement Analysis: Understanding customer expectations and needs.

System Design: Detailed design of the complete system.

Architectural Design: Designing architectural specifications.

Module Design: Specifying detailed internal designs for system modules.

Coding Phase: Actual coding of system modules based on the design.

4.3 COMPONENT DESIGN:

V-Model Application:

Similar to the waterfall model, the V-Model application is of a sequential type.

Suitable scenarios to use V-Model include well-defined, stable requirements, fixed product definition, known technology, and a short project duration.

Pros:

Highly disciplined and suitable for smaller projects with well-understood requirements.

Simple, easy to understand, and manage.

Cons:

High risk and uncertainty.

Not suitable for complex, object-oriented, or long-term projects.

4.4 DATABASE DESIGN:

Database and Tables:

A Database Management System (DBMS) manages databases and runs operations on structured data.

Examples of DBMS include Oracle, DB2, MySQL, etc.

Description:

DBMS controls organization, storage, management, and retrieval of data.

It includes a modeling language, data structures, a query language, report writer, transaction mechanism, and maintains data integrity.

DBMS accepts requests for data from application programs and ensures data security.

SQL (Structured Query Language):

SQL is used to manipulate relational databases tied closely with the relational model.

SQL statements include data definition (DDL) for creating, altering, and dropping schema objects, and data manipulation (DML) for inserting, updating, deleting, and querying data.

V. SYSTEM TESTING AND IMPLEMENTATION

5.1 TESTING

The goal of testing is to acquire errors. Testing is that the technique of trying to get each possible error or weakness in an extremely work product. It provides the way to observe the practicality of parts, sub-assemblies, assemblies and or a finished product it is the

technique of effort code with the concentrating of guaranteeing that the software meets its requirements and user hopes and does not fail in an undesirable manner. There are numerous sorts of check. Every check sort reports a designated testing demand.

Testing objectives:

The key objective of testing is to determine a mass of errors, systematically and with minimum effort and time. Stating formally, we can say, testing is a process of executing a program with resolved of discover an error.

- A successful test is one that determines an as however undiscovered error.
- A good test case is one that has possibility of discover an error, if it exists.
- The test is insufficient to detect possibly present errors.
- The software more or less approves to the quality and unswerving standards.

5.1.1 Unit Testing & Test Cases:

Unit Testing: A unit is the smallest piece of source code, also known as a module.

The purpose of unit testing is to expose non-compliance with functional requirements and ensure the structural implementation aligns with the design.

5.1.2 Integration Testing & Test Cases:

Integration Testing:

Tests integrated software components to verify they run as one system.

It focuses on revealing problems that arise from the combination of components.

5.1.3 User Acceptance Testing & Test Cases:

User Acceptance Testing (UAT):

A crucial phase requiring significant input from end users.

Ensures the system meets functional requirements.

5.1.4 Output Testing & Test Cases:

Test Strategy and Approach:

Ground testing done manually, and functional tests documented in detail.

Test Objectives:

Ensure all field submissions work properly.

Pages are activated from identified links.

Entry screen, messages, and responses are not delayed.

Features to be Tested:

Validate correct format for accesses.

Disallow duplicate entries.

Verify all links redirect to the correct page.

5.1.5 Validation Testing & Test Cases:

Test Objectives: Functional tests confirm that the tested functions align with business and technical requirements.

Functional Testing Focus:

Valid Input: Accept known categories of valid input.

Invalid Input: Reject known categories of invalid input.

Functions: Exercise identified functions.

Output: Exercise identified classes of application outputs.

Systems/Procedures: Invoke interfacing systems or procedures.

Association and Arrangement:

Scope is based on requirements, key functions, or unique experiments.

Identifies additional tests and determines the effective value of current tests before completing functional testing.

5.2 SYSTEM SECURITY:

System testing guarantees that the entire coordinated programming framework meets prerequisites. It tests a design to guarantee known and unsurprising results. A sample of framework testing is the arrangement situated framework combination test. Framework testing depends on procedure portrayals and streams, stressing pre-driven procedure connections and mix focuses.

White Box Testing

It is a testing in which the product analyzer has information of the internal workings, structure and dialect of the product, or if nothing else its motivation. It is reason. It is utilized to test ranges that can't be gotten a handle on from a discovery level.

Black Box Testing

It is the testing the product with no information of within workings, structure or dialect of the part being tried. Discovery tests, as most different sorts of tests, must be composed from a complete source report, for example, prerequisite or necessities archive, for example, determination or necessities record. It is a trying in which the product under test is dealt with, as a discovery you can't "see" into it. The test gives inputs and reacts to yields without considering how the product functions.

Unit Testing:

Unit testing is by and large appeared as a major aspect of a joined code and unit test period of the product lifecycle, in spite of the fact that it is not exceptional for coding and unit testing to be directed as two unmistakable stages.

Integration Testing

Software integration testing is the incremental combination analysis of two or more joint software components on a single platform to generate failures created by boundary faults. The task of the integration test is to design those components or s/w applications, e.g. modules in a software system or – one step up – software presentations at the company level

– interact without faults.

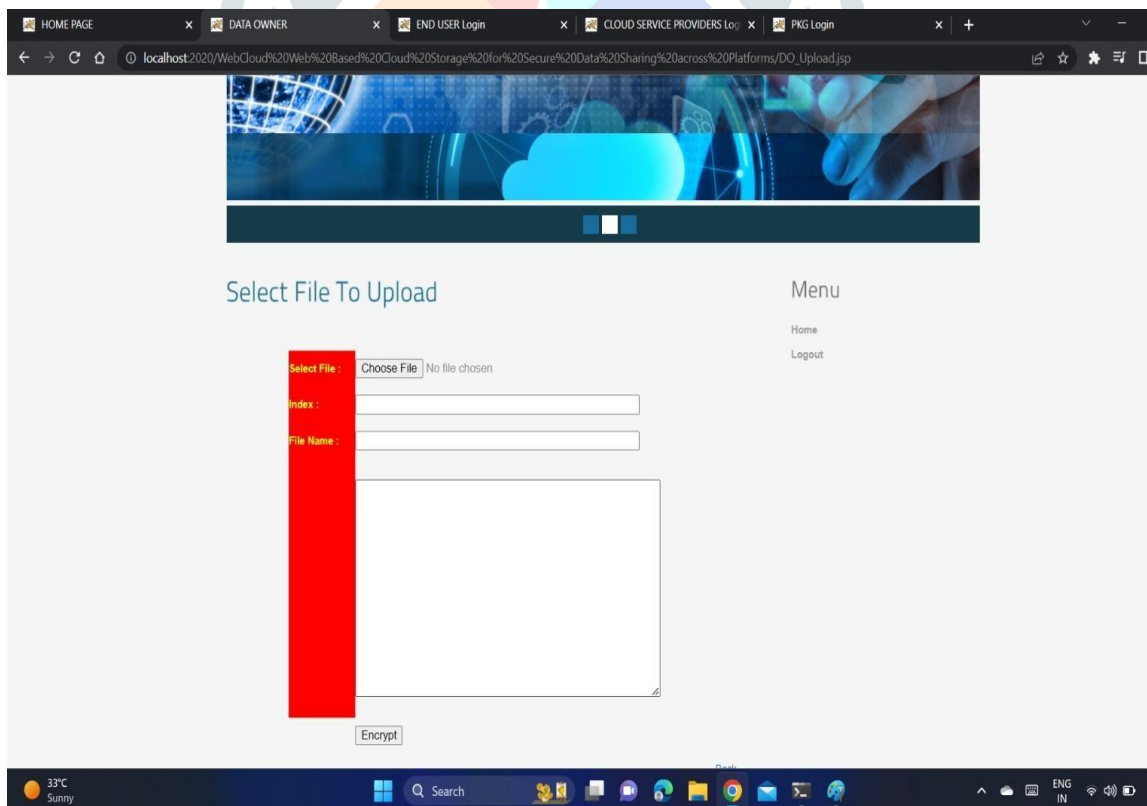
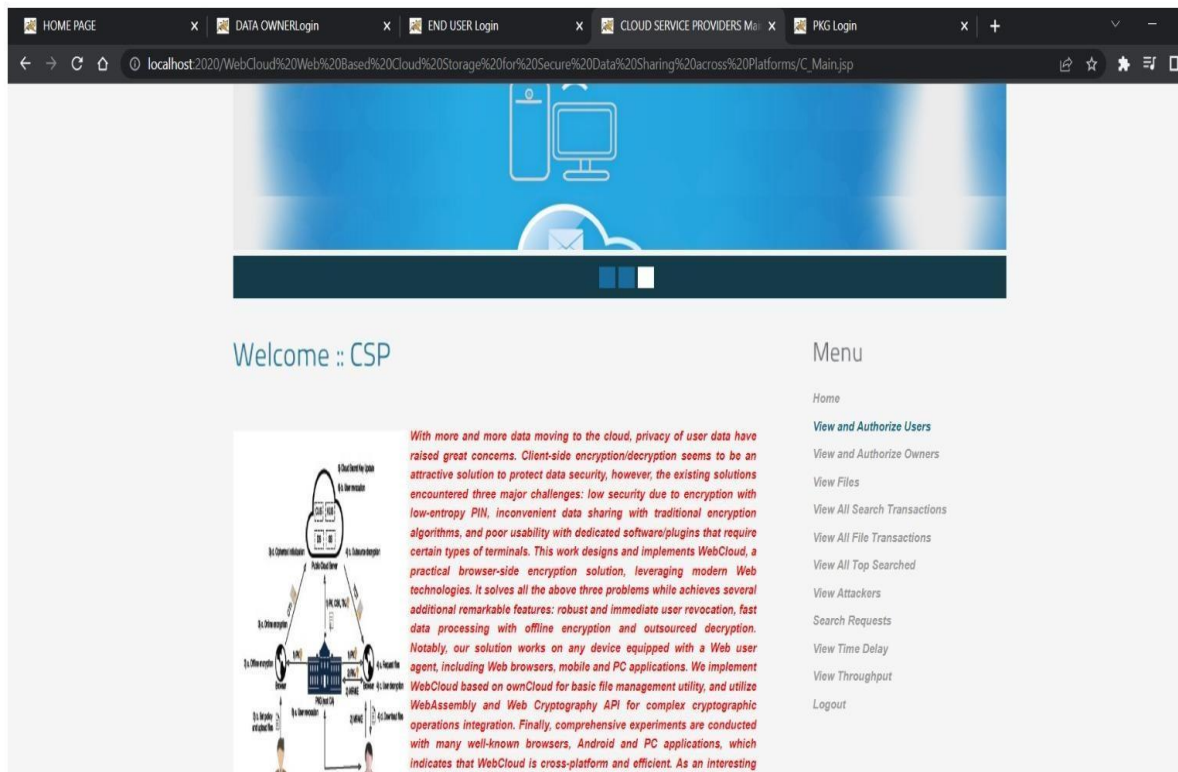
Test Results: All the test cases stated above passed effectively. No defects met.

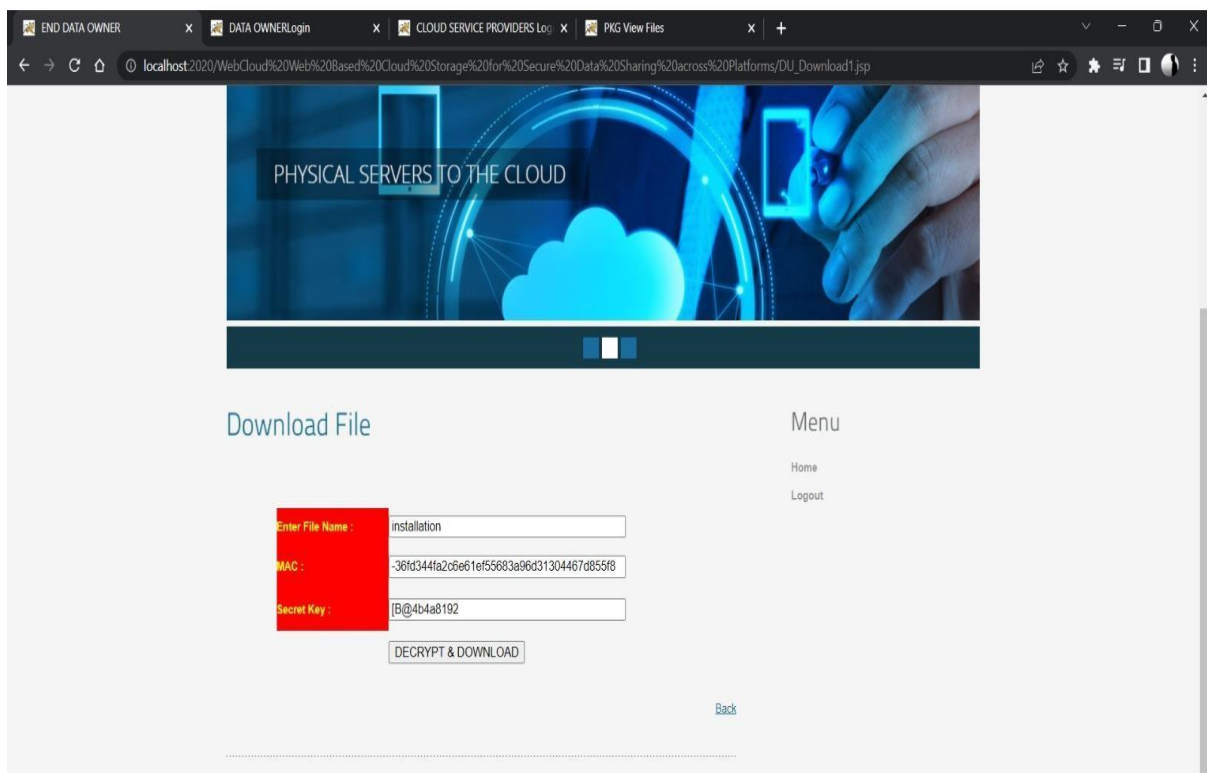
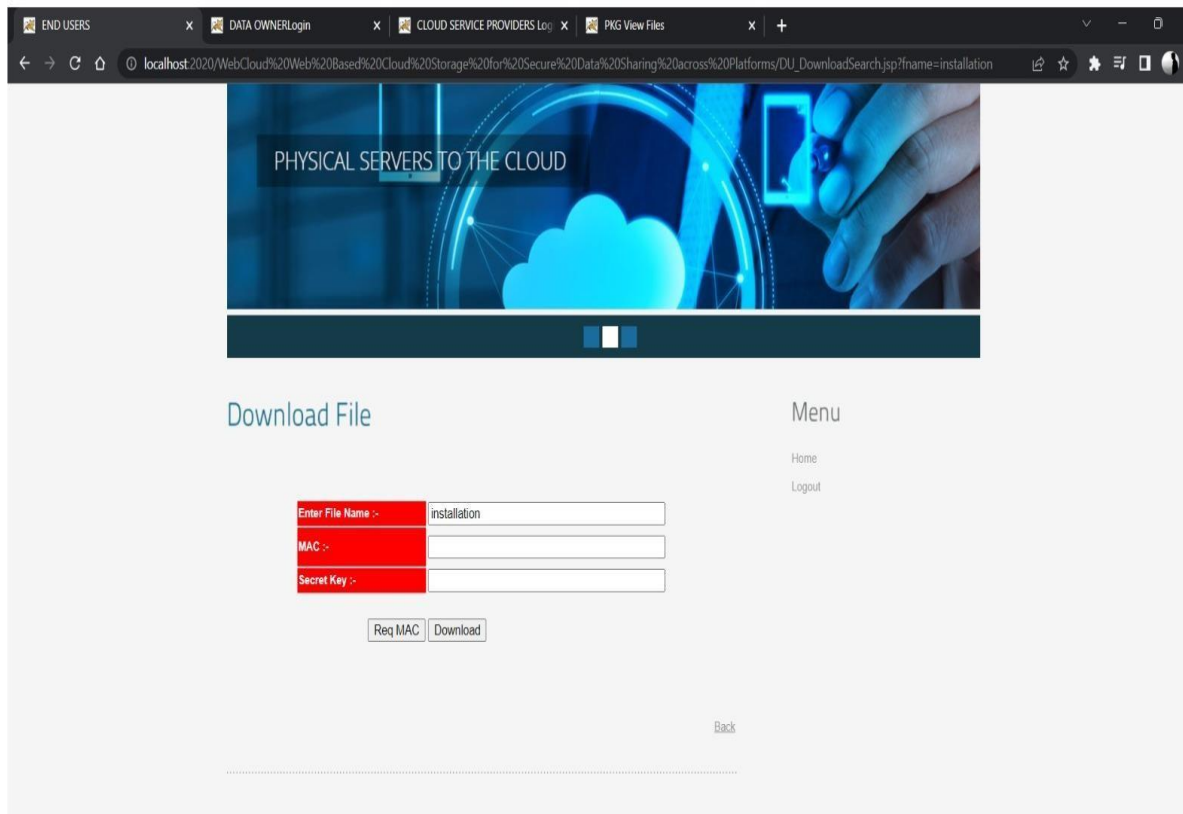
5.3 SYSTEM ENHANCEMENT

Test case id	Test case description	Actual value	Entered value	Status
1	Register user details in registration page	Fill all the fields while registering user	All the fields are filled	Pass
2	Give user name in text box	User name must be given in alphabets	User name given in alphabets and numeric values	Fail
3	Password to be entered in password box	Password must be given correctly	Password is entered wrongly	Fail
4	Phone number must be entered in phone number box during registration	Phone number must be given in 10 digits	Phone number given in 10 digits	Pass
5	Validating the functionality of Browse button	System should select the corresponding file	selected the file what we expected	Pass

Fig : Testcase Template

OUT SCREENS:





VI. CONCLUSION:

The proposed Web Cloud presents a practical client-side encryption solution tailored for public cloud storage within a web environment. Key aspects and findings of the study include:

Web Cloud Implementation:

- A client-side encryption solution designed for public cloud storage in the web setting.
- Users perform cryptographic operations exclusively within web browsers.

Security Analysis:

- Rigorous security analysis of Web Cloud.
- Evaluation of the proposed security model and the cryptographic scheme's reliability.

Performance Evaluation:

- Implementation of Web Cloud based on own Cloud.
- Comprehensive performance evaluation in major browsers on various devices, including PCs and Android devices.

Practicality and Viability:

- Experimental results demonstrate the practicality and viability of the proposed solution.
- CP-AB-KEM Scheme:

The design of Web Cloud naturally includes a dedicated CP-AB-KEM scheme.

This scheme has potential applications in various contexts beyond public cloud storage.

Feature Enhancement:

- To further improve the system and enhance its features, the following considerations can be explored:

Usability Improvements:

- Focus on refining the user interface and experience within web browsers.
- Implement user-friendly features to enhance accessibility.
- Extended Security Measures:
- Continuously monitor and address emerging security threats.
- Consider the integration of additional security measures to enhance data protection.

Scalability and Flexibility:

- Explore ways to make the system more scalable, accommodating the growing needs of users.
- Ensure flexibility to adapt to evolving technological landscapes.

Integration with Emerging Technologies:

- Stay abreast of emerging technologies and explore integration possibilities.
- Consider compatibility with evolving web standards and cryptographic advancements.

Community and User Feedback:

- Establish channels for user feedback and community involvement.
- Leverage user insights to identify areas for improvement and new feature implementations.

VII REFERENCES

- [1] R. Gross, A. Acquisti, and H. J. Heinz, "Information revelation and privacy in online social networks," in Proc. ACM Workshop Privacy Electron. Soc., 2005, pp. 71_80.
- [2] J. Nagy and P. Pecho, "Social networks security," in Proc. 3rd Int. Conf. Emerg. Secur. Inf., Syst. Technol., 2009, pp. 321_325.
- [3] S. Kemp, "The state of digital in April 2019: All the numbers you need to know," Tech. Rep., 2019. [Online]. Available: <https://wearesocial.com/us/blog/2019/04/the-state-of-digital-in-april-2019-all-the-numbers-you-need-to-know/>
- [4] G. Faces and N. Places, "A Nielsen report on social networking's new global footprint," Nielsen Company, New York, NY, USA, Tech. Rep., 2009.
- [5] L. F. Cranor, J. Reagle, and M. S. Ackerman, "Beyond concern: Understanding net users' attitudes about online privacy," in The Internet Upheaval: Raising Questions, Seeking Answers in Communications Policy. Cambridge, MA, USA: MIT Press, 2000, pp. 47_70.
- [6] (2018). 5 Soruda Facebook Verilerini "Usulsuz Kullanmakla" Suçlanan Cambridge Analytica. [Online]. Available: <https://www.bbc.com/turkce/haberler-dunya-43469094>
- [7] E. Christo_des, A. Muise, and S. Desmarais, Privacy and Disclosure on Facebook: Youth and Adult's Information Disclosure and Perceptions of Privacy Risks. Guelph, ON, Canada: Univ. of Guelph, 2011.
- [8] D. O'Brien and A. M. Torres, "Social networking and online privacy: Facebook users' perceptions," Irish J. Manage., vol. 31, no. 2, p. 63, 2012.
- [9] N. Aldhafferi, C. Watson, and A. S. M. Sajeev, "Personal information privacy settings of online social networks and their suitability for mobile internet devices," 2013, arXiv:1305.2770.
- [10] M. Madden, "Privacy management on social media sites," Pew Internet Rep., 2012, pp. 1_20.

- [11] K. Williams, A. Boyd, S. Densten, R. Chin, D. Diamond, and C. Morgenthaler, "Social networking privacy behaviors and risks," Seidenberg School CSIS, Pace Univ., New York, NY, USA, Tech. Rep., 2009.
- [12] N. B. Ellison, C. Stein_eld, and C. Lampe, "The bene_ts of Facebook`friends': Social capital and college students' use of online social network sites," *J. Comput.-Mediated Commun.*, vol. 12, no. 4, pp. 1143_1168, Jul. 2007.
- [13] P. B. Brandtzæg and J. Heim, "Why people use social networking sites," in *Proc. Int. Conf. Online Communities Social Comput.* San Diego, CA, USA: Springer, 2009, pp. 143_152.
- [14] S. M. Ghafari, A. Beheshti, A. Joshi, C. Paris, A. Mahmood, S. Yakhchi, and M. A. Orgun, "A survey on trust prediction in online social networks," *IEEE Access*, vol. 8, pp. 144292_144309, 2020.
- [15] A. Beheshti, V. Moraveji-Hashemi, S. Yakhchi, H. R. Motahari-Nezhad, S. M. Ghafari, and J. Yang, "personality2vec: Enabling the analysis of behavioral disorders in social networks," in *Proc. 13th Int. Conf. Web Search Data Mining (WSDM)*. New York, NY, USA Association for Computing Machinery, Jan. 2020, pp. 825_828, doi:10.1145/3336191.3371865.
- [16] V. Moustaka, Z. Theodosiou, A. Vakali, A. Kounoudes, and L. G. Anthopoulos, "Enhancing social networking in smart cities: Privacy and security borderlines," *Technol. Forecasting Social Change*, vol. 142, pp. 285_300, May 2019.
- [17] M. Tsay-Vogel, J. Shanahan, and N. Signorielli, "Social media cultivating perceptions of privacy: A 5-year analysis of privacy attitudes and self disclosure behaviors among Facebook users," *New Media Soc.*, vol. 20, no. 1, pp. 141_161, Jan. 2018, doi: 10.1177/1461444816660731.
- [18] S. Das, T. H.-J. Kim, L. A. Dabbish, and J. I. Hong, "The effect of social in_uence on security sensitivity," in *Proc. 10th Symp. Usable Privacy Secur. (SOUPS)*. Menlo Park, CA, USA: USENIX Association, Jul. 2014, pp. 143_157. [Online]. Available: https://www.usenix.org/conference/soups2014/proceedings/pre_sentation/das
- [19] Sar_, "Çocuk ve bili³im sanaldan gerçe³e sorunlar: Çözüm önerileri ve iyi uygulama örnekleri. SAMER yay_nlar_, accessed: Jul. 14, 2019," Tech. Rep., 2013. [20] M. Anderson, "Parents, teens and digital monitoring," *Internet, Sci. Tech.*, Pew Res. Center, Washington, DC, USA, Tech. Rep., 2016.
- [21] A. A.-A. Mohamed, "Online privacy concerns among social networks' users," *Cross-Cultural Commun.*, vol. 6, no. 4, pp. 74_89, 2011.
- [22] M. Fire, R. Goldschmidt, and Y. Elovici, "Online social networks: Threats and solutions," *IEEE Commun. Surveys Tuts.*, vol. 16, no. 4, pp. 2019_2036, 4th Quart., 2014.
- [23] J. Baltazar, J. Costoya, and R. Flores, "The real face of koobface: The largest web 2.0 botnet explained," *Trend Micro Res.*, vol. 5, no. 9, p. 10, 2009.
- [24] Z. Mao, N. Li, and I. Molloy, "Defeating cross-site request forgery attacks with browser-enforced authenticity protection," in *Proc. Int. Conf. Financial Cryptogr. Data Secur. Accra Beach, Barbados*: Springer, 2009, pp. 238_255.
- [25] V. B. Livshits and W. Cui, "Spectator: Detection and containment of JavaScript worms," in *Proc. USENIX Annu. Tech. Conf.*, 2008, pp. 335_348.