A

Major Project Report

On

Secure Authentication Method using honey Key Password in Online Shopping

submitted in partial fulfillment of the requirements for the award of the degree of

Bachelor of Technology

In

COMPUTER SCIENCE AND ENGINEERING

by

Araganlapally Manidhar

(20EG105303)

Bashyakarla Sucharitha

(20EG105316)

Lakavath Sravanthi

(20EG105708)

Under the guidance of

Mrs. P. Swarajya Lakshmi

Assistant Professor

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

ANURAG UNIVERSITY

VENTAKAPUR (V), GHATKESAR (M), MEDCHAL (D), T.S - 500088

TELANGANA

(2023-2024)

DECLARATION

We hereby declare that the Report entitled "**Secure Authentication Method using honey Key Password in Online Shopping**" submitted for the award of Bachelor of technology Degree is our original work and the Report has not formed the basis for the award of any degree, diploma, associate ship or fellowship of similar other titles. It has not been submitted to any other University or Institution for the award of any degree or diploma**.**

Place: Anurag University,Hyderabad
Date :

Araganlapally Manidhar
20Eg105303

Bashyakarla Sucharitha
20EG105316

Lakavath Sravanthi
20EG105708

DEPARTMENT OF COMPUTER SCIENCE AND ENGNIEERING

CERTIFICATE

This is to certify that the Project entitled "**Secure Authentication Method using honey Key Password in Online Shopping**" being submitted by **Araganlapally Manidhar** bearing the Hall Ticket number **20EG105303, Bashyakarla Sucharitha** bearing the Hall Ticket number **20EG105316, Lakavath Sravanthi** bearing the Hall Ticket number **20EG105708** in partial fulfillment of the requirements for the award of the **Bachelor of Technology** in **Computer Science and Engineering** to the **Anurag University** is a record of bonafide work carried out by them under my guidance and supervision for the academic year 2023 to 2024

The results embodied in this Project have been verified and found to be satisfactory. The results embodies in the Project report have not been submitted to any other University or Institute for the award of any degree or diploma.

Signature of Supervisor	Signature of the Dean
Mrs. P Swarajya Lakshmi	Dr. G Vishnu Murthy
Assistant Professor	Dean, CSE
Department of CSE	

External Examiner

ACKNOWLEDGMENT

We would like to express our sincere thanks and deep sense of gratitude to project supervisor **Mrs.P. Swarajya Lakshmi, Assistant Professor, Department of Computer Science and Engineering**, Anurag University for her constant encouragement and inspiring guidance without which this project could not have been completed. Her critical reviews and constructive comments improved our grasp of the subject and steered to the fruitful completion of the work. Her patience, guidance and encouragement made this project possible.

We would like to acknowledge my sincere gratitude for the support extended by **Dr. G. VISHNU MURTHY**, **Dean, Department of Computer Science and Engineering,** Anurag University.

We also express my deep sense of gratitude to **Dr. V. V. S. S. S. BALARAM**, Academic coordinator. **Dr. T. SHYAM PRASAD,** Assistant Professor**,** Project Coordinator and Project review committee members, whose research expertise and commitment to the highest standards continuously motivated us during the crucial stage of my project work.

We would like to express special thanks to **Dr. V. VIJAYA KUMAR, Dean School of Engineering,** Anurag University, for his encouragement and timely Support in our B. Tech program.

Araganlapally Manidhar
20Eg105303

Bashyakarla Sucharitha
20EG105316

Lakavath Sravanthi
20EG105708

ABSTRACT

As online applications relying predominantly on password-only authentication, the vulnerability to password leaks from both internal and external adversaries remains a critical concern. This paper introduces an innovative solution, Honey PAKE (HPAKE), designed to overcome the limitations of existing methods, including honeywords for external attacks and augmented password-authentication key exchange (aPAKE) for insider threats. HPAKE implements a robust authentication mechanism during the registration process within online shopping platforms. Our unique registration protocol mandates users to upload a single text file and provide a secret key pair. Any modifications to these elements result in login restrictions, thereby reinforcing the authentication process. This protocol not only enhances the security of online shopping applications but also empowers the authentication server to proactively identify and address potential password leaks. Moreover, to mitigate fraudulent activities, specifically erroneous payments, users involved in such transactions will face account suspension. The integration of HPAKE signifies a substantial progression in securing user credentials and establishing a formidable defense against both internal and external threats prevalent in the online shopping domain.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

1. INTRODUCTION

1.1 Overview

The most often used authentication method in safety-related applications is the username/password paradigm . Tokens and biometrics are two examples of alternative authentication elements that necessitate the purchase of additional hardware, which is sometimes deemed too costly for a given application. Passwords, however, are vulnerable to dictionary attacks and are low-entropy secrets . They must therefore be safeguarded throughout transmission. Passwords are sent using SSL/TLS, which is the most commonly used protocol . However, Public Key Infrastructure (PKI) must be set up for this; PKI maintenance is costly. Furthermore, there is a risk of man-in-the-middle attacks while utilizing SSL/TLS . Even though the session is fully encrypted, if a user authenticates himself to a phishing website by sharing his password, the password will be stolen. One logical approach, given the inherent weakness of passwords, would seem to be to replace them with powerful secrets, such as cryptographically secure private keys. The UK National Grid Service (NGS) used this method for user authentication [4].This has made it much more difficult for grid computing technologies to become widely used. Therefore, we have to accept the fact that weak passwords are a reality. The Password-Authenticated Key Exchange (PAKE) is a research topic that has researchers actively investigating methods to accomplish password-based authentication without utilizing PKIs or certificates . The EKE protocol was introduced by Beloin and Merrit in 1992, marking the first significant event . The PAKE problem was shown to be at least solvable by the EKE protocol, despite certain documented flaws .Numerous protocols have been suggested since then. Many of them are just EKE variations that implement the "symmetric cipher" in different ways . The majority of the few methods that assert to be resistant to known attacks are patent-protected; two prominent examples are Lucent Technologies' EKE and Phoenix Technologies' SPEKE . Because of this, applying these strategies cannot immediately benefit the scientific community or the larger security business .The EKE and SPEKE protocols only provide heuristic security. Formal security proofs seem unlikely without introducing new assumptions or relaxing criteria, given the way the two methodologies were constructed; we shall describe the specifics in Section 4. In the section that follows, we will present an alternative method for resolving the CAE problem and demonstrate how it is free from the security flaws associated with the EKE and SPEKE protocols.

1.2 Problem Statement:

The prevalent use of password-only authentication in online shopping platforms has led to vulnerabilities from both external attacks and insider threats. Current methods, susceptible to password leaks and the risks associated with augmented password-authenticated key exchange (aPAKE), call for an innovative solution. This project introduces Honey Password Authenticated Key Exchange (HPAKE) to address these concerns. HPAKE implements a stringent registration process, mandating users to upload a text file and provide a secret key pair. This approach not only strengthens authentication but also proactively identifies potential password leaks. Given the limitations of existing methods in the realm of online shopping security, there is an urgent need for comprehensive solution like HPAKE. Its integration represents a substantial advancement, offering improved defense against both external adversaries and insider threats, thereby elevating the overall security of user credentials in the dynamic digital landscape.

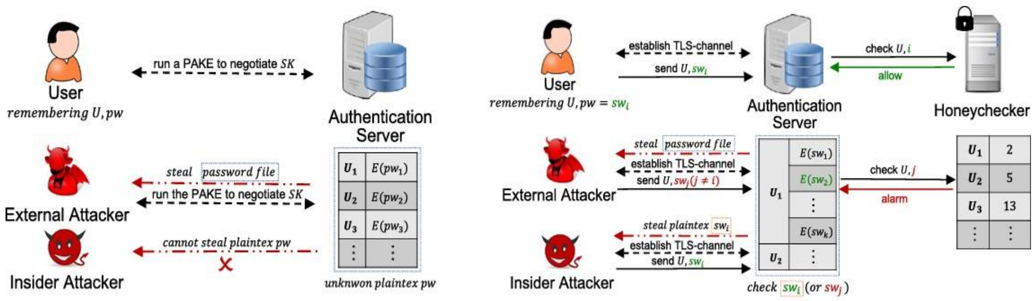
1.3 Problem Illustration:

In safety-related applications, the prevailing method of authentication remains the username/password paradigm, despite its susceptibility to dictionary attacks and interception during transmission. While alternatives like tokens and biometrics offer enhanced security, their adoption is hindered by the additional cost of hardware implementation. To secure password transmission, SSL/TLS encryption is commonly employed, necessitating the setup and maintenance of

Public Key Infrastructure (PKI), which can be financially burdensome. Moreover, the risk of man-in-the-middle attacks persists, especially when users unwittingly authenticate on phishing websites. Given these vulnerabilities, the exploration of stronger authentication methods, such as cryptographically secure private keys, becomes imperative. However, their complexity and cost hinder widespread adoption. In response, research into Password-Authenticated Key Exchange (PAKE) protocols, such as EKE and SPEKE, has intensified. Despite offering heuristic security, these protocols lack formal proofs and are often encumbered by patents, limiting their accessibility to the broader security community. Consequently, ongoing efforts focus on devising alternative authentication methods that address these shortcomings while ensuring robust security and affordability.

2. LITERATURE SURVEY

Against Insiders. In Figure 1a, Augmented password authentication key exchange (aPAKE) [9] is designed to allow a client and a server to establish a session key based on a password, where the client has the password plaintext and the server only holds the verifier. This technique prevents the server from knowing the password, and therefore resists the insider attacks. Since Bellovin and Merrit [9] introduced this notion, many researchers proposed various aPAKE schemes [10], [11], [12], [13] in order to improve the security and efficiency performance. Among them, OPAQUE [12] is the most well-studied scheme with the strongest security and thus, it recently is standardized by the Crypto Forum Research Group of the Internet Engineering Task Force (IETF) [14]. Against Outsiders. Honeyword technique [15] (see Figure 1b) is proposed to detect the password leakage for the most common password-only authentication systems, passwordover-TLS. This approach associates $t-1$ decoy and plausible looking passwords (i.e., honeywords) to each account. The honeywords and the real password are collectively called sweet words. If an attacker steals the password file, she cannot tell the real one and probably (with $1-1/t$ probability) log in with a honeyword. Then, the server can detect the password leakage from the “wrong” login. The follow-up works focus on the honeyword generation algorithms [16], [17] so as to produce more plausible-looking decoys and the detection methods [18] to improve reliability. Others. Password less authentication [19] or multi-factor authentication systems [20], [21] make good use of other factors, e.g., smartphone and fingerprint. They significantly reduce the risk of password leakage. If an attacker steals the password, she still needs additional factors to compromise account. Besides, in some of these designs, authentication server does not need to store the password-related data, so that even if the attacker compromises the storage file on server, she cannot carry out offline password guessing as long as other factors are secure. A typical design can be seen in [21], [22], [23] that a smart device (as an authentication factor) is used to store the password-related data, making systems resist offline guessing in the case of server compromise. Shortcomings. The techniques above, unfortunately, have the following shortcomings. The honeyword mechanism requires the client to send the password plaintext to the server (via a server-authenticated secure channel), otherwise the server cannot tell if the login password is real. Thus an insider can directly steal the plaintext of the login password without any guessing attacks. In aPAKE, the server has to store the verifiers in the password file for authentication. But an external attacker may steal the file and carry out guessing attacks [24] to recover the password. This vulnerability is inherent in aPAKE. And neither of these methods can provide a solution maintaining security against both insiders and outsiders. As for other (passwordless or multi-factor) approaches, they may provide stronger security relying on extra factors, which may bring disadvantages to deployability and usability. In this paper, we do not consider them and only focus on password-only authentication. According to the above discussion, we thus raise a question: “How could one design a fast and secure password-only authentication scheme that can resist both the insider and external attackers.



(a) aPAKE (b) Honeyword mechanism

Figure 2.1 Existing method

Author(s)	Strategies	Advantages	Disadvantages
Steven M. Bellovin and Michael Merritt	encrypted key exchange: a password-based protocol	• Reduce dictionary attacks • Protect users with weak passwords	• Complexity • Key management
WentingLi, Haibo Cheng,Ping Wang*Eand KaitaiLiang	Practical threshold multi factor authentication	• Flexibility • Usability	• Dependency on factors
Stanislaw Jarecki, Hugo Krawczyk, and Jiayu Xu1	OPAQUE: An Asymmetric PAKE Protocol Secure Against Pre Computation Attacks	• Secure against pre-computation attacks • Reduce offline dictionary attacks	• Scalability Challenges • Complexity in Implementation

Table 2.1. Literature Survey

3. PROPOSED METHOD

3.1 Illustration

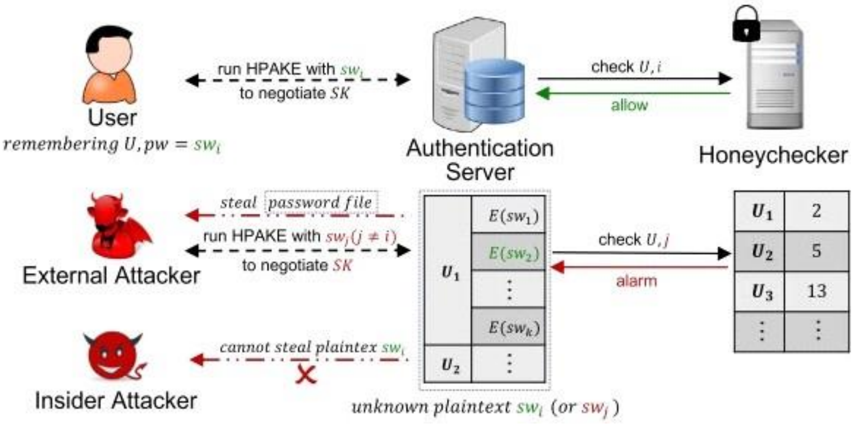


Fig 3.1 : Hpake

In order to provide security beyond the conventional bounds of aPAKE, we introduce the concept of honey PAKE (HPAKE), which enables the authentication server to detect password leakage. On top of the honeyword method, honey encryption, and OPAQUE, a standardized aPAKE, we also design an HPAKE structure. Our design's security is properly examined, with insider resistance and password breach detection achieved.

HPAKE offers both honeyword tactics and password leaking to **external attackers.** Through offline guessing attacks, the attacker will obtain a password list with one sweetword if the authentication service is compromised. Because the attacker can't distinguish which is real, it's likely that they use a honeyword together with HPAKE to compromise the account. As a result, a honey session key will be generated, and any use of the key will trigger an alarm.

For the **insiders**: HPAKE achieves the same level of security as aPAKE by guaranteeing that the password plaintext is never disclosed to the client. The key exchange specifically completes the authentication. Additionally, using the actual password to execute the program will produce a real session key, and only instructions that have been encrypted with that key will be permitted. Consequently, the plaintext password cannot be stolen by the server or insider.

3.2 User flow:

3.2.1 Admin Login:

The admin navigates to the website's admin login page. Enters their username and password into the designated fields. Clicks on the "Login" button to proceed . Upon successful authentication, the admin is redirected to the admin dashboard. The dashboard provides an overview of key metrics and recent activities. From the dashboard, the admin selects the "Manage Products" option. They are presented with a list of existing products and options to add new products. The admin can add new products by providing details such as name, description, price, and images. They can also edit or delete existing products as needed. The admin can also choose to simply view the list of products without making any changes. This allows them to review the inventory and ensure accuracy. Next, the admin selects the "Manage Users" option. They can view a list of registered users along with their details such as username, email, and account status. The admin has the ability to search for specific users and filter the list based on criteria such as account status or registration date. Within the user management section, the admin can access a separate tab or option specifically for detecting and managing fraud users. This tab displays users flagged for suspicious activity, such as multiple failed login attempts or irregular purchasing behavior. The admin can review the details of these users and take appropriate action, such as blocking their accounts. In cases where fraudulent activity is confirmed or for other reasons deemed necessary by the admin, they can choose to block specific user accounts. This action prevents the blocked users from accessing the website and making any further transactions.

Blocked users may be notified of the action via email or through a notification within their account.

3.2.2 User login:

The user navigates to the registration page of the online shopping website. Enters the required information such as username, password, gender, date of birth, mobile number, email, address, and uploads a file named "Filename." Clicks on the "Register" button to create an account. After registration, the user proceeds to the login page for subsequent access. Enters their username and password into the respective fields. Uploads the same file named "Filename" used during registration for authentication purposes. Clicks on the "Login" button to initiate the login process. Upon successful login attempt, the server initiates the secret key generati- on process. Using the user's password and the contents of the uploaded file, the server generates a unique secret key for the session. Once logged in, the user can navigate to their account details section. They can view and manage their personal information such as username, gender, date of birth, mobile number, email, and address. The user can browse through the available products on the online shopping website. They can view product details, such as name, description, price, and images. When the user finds a product they wish to purchase, they can add it to their shopping cart. They proceed to checkout, where they can review their order and enter payment details to complete the purchase. Once the user has completed their session, they can choose to logout from the website. This terminates their current session and ensures the security of their account.

3.2.3 Server Login:

The server accesses the login page designed for administrators .Enters the server's username and password into the respective fields. Clicks on the "Login" button to authenticate. Upon successful login, the server is directed to the admin dashboard. The dashboard provides options for managing users. The server selects the "View Users" option to access user information. From the list of users, the server chooses a specific user for whom to generate and a secret key. The server initiates the process to generate a unique secret key for the selected user. Once the secret key is generated, the server sends it securely to the user through a designated channel, such as email or direct message. This user flow outlines the server's actions, including logging in, viewing users, and sending a secret key to a specific user.

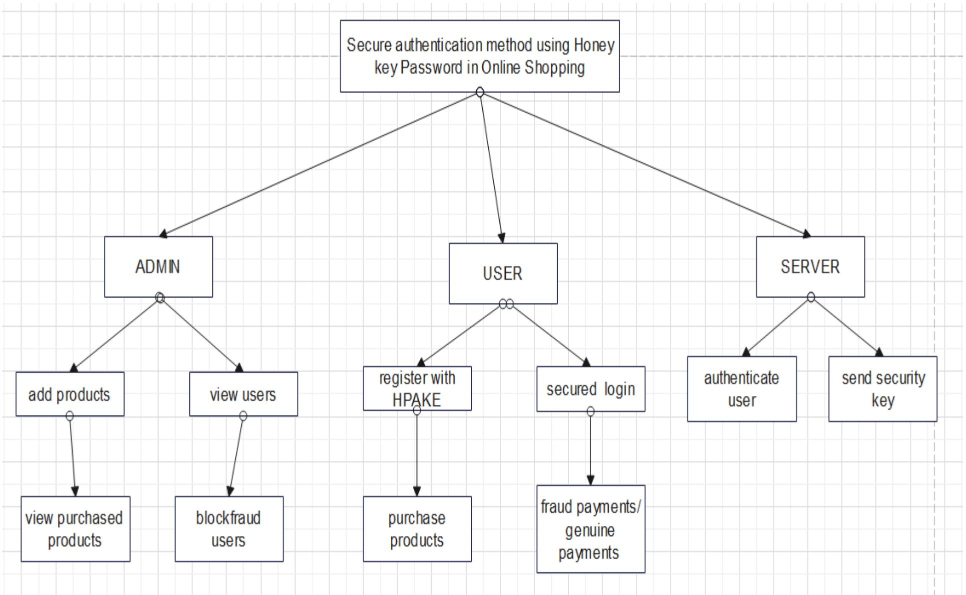


Figure 3.2.3.1 Flowchart

4. DESIGN

4.1 UML Diagrams

UML, or Unified Modeling Language, is a graphical tool essential for designing software systems. It offers standardized visual models for representing object-oriented software structures. UML diagrams are crucial for clear and organized communication of design concepts. These diagrams are indispensable in software design, aiding developers in understanding and analyzing intricate systems. They serve as effective tools for conveying design ideas to team members, stakeholders, and clients, ensuring that the software meets required standards of functionality, performance, and quality. Moreover, UML diagrams help in detecting and rectifying errors early in the development process, thereby saving time and reducing costs. They come in two main types: structural diagrams, which depict the static structure of the system, and behavioral diagrams, which illustrate dynamic interactions and Workflows. In summary, UML diagrams play a vital role in streamlining the software development process, providing developers with a clear and efficient means of conceptualizing and refining software systems.

4.1.1 Use Case Diagram

A use case diagram in the Unified Modeling Language (UML) is a type of behavioral diagram defined by and created from a Use-case analysis. Its purpose is to present a graphical overview of the functionality provided by a system in terms of actors, their goals (represented as use cases), and any dependencies between those use cases. The main purpose of a use case diagram is to show what system functions are performed for which actor. Roles of the actors in the system can be depicted.

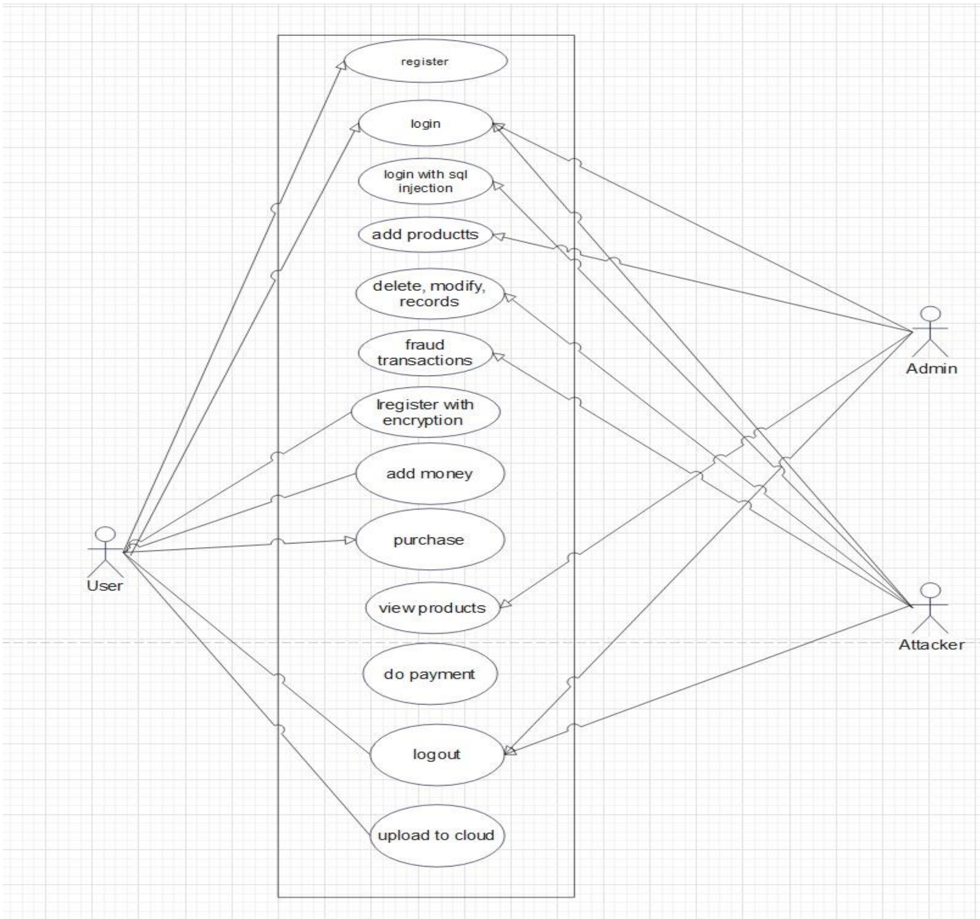


Figure 4.1.1.1 Use case Diagram

User Actor:

Register: The user can register for a new account on the online shopping platform.

Login: Allows the user to log in to their account.

Register with Encryption: Similar to registration, but with the added security measure of encryption for sensitive data.

Add Money: The user can add funds to their account to make purchases.

Purchase Products: Enables the user to browse and buy products from the online store.

Logout: Allows the user to securely log out of their account.

Upload File to Cloud: The user can upload files to the cloud storage provided by the online shopping platform.

Admin Actor:

Login: The admin can log in to the system using their credentials.

Add Products: Allows the admin to add new products to the online store.

View Products: Enables the admin to view the list of available products in the online store.

Logout: The admin can securely log out of their account after completing tasks.

Attacker Actor:

Login with SQL Injection: The attacker exploits vulnerabilities in the login system using SQL injection to gain unauthorized access.

Delete/Modify Records: The attacker can delete or modify records in the system, causing data corruption or loss.

Fraud Transactions: Allows the attacker to carry out fraudulent transactions within the system.

Logout: The attacker can log out of the system after executing malicious activities.

4.1.2 Sequence Diagram

A sequence diagram is a visual representation that illustrates the interactions and communication flow between different components or objects in a system over time. It demonstrates how these entities collaborate and exchange messages to achieve specific functionalities. In a sequence diagram, the vertical axis typically represents time, while horizontal lines, called lifelines, represent individual entities or objects involved in the interaction. Arrows between lifelines indicate the flow of messages or method calls between these entities, depicting the sequence of actions and responses. Sequence diagrams are invaluable tools in software engineering for designing and understanding the behavior of systems, as they provide a clear and intuitive visualization of the runtime interactions between various components. They aid in identifying potential bottlenecks, understanding the order of operations, and ensuring that system requirements are met. Additionally, sequence diagrams facilitate communication among stakeholders by offering a concise and structured depiction of system behavior, enabling effective collaboration and decision-making during the software development process. Overall, sequence diagrams play a crucial role in analyzing, designing, and documenting the dynamic behavior of complex systems

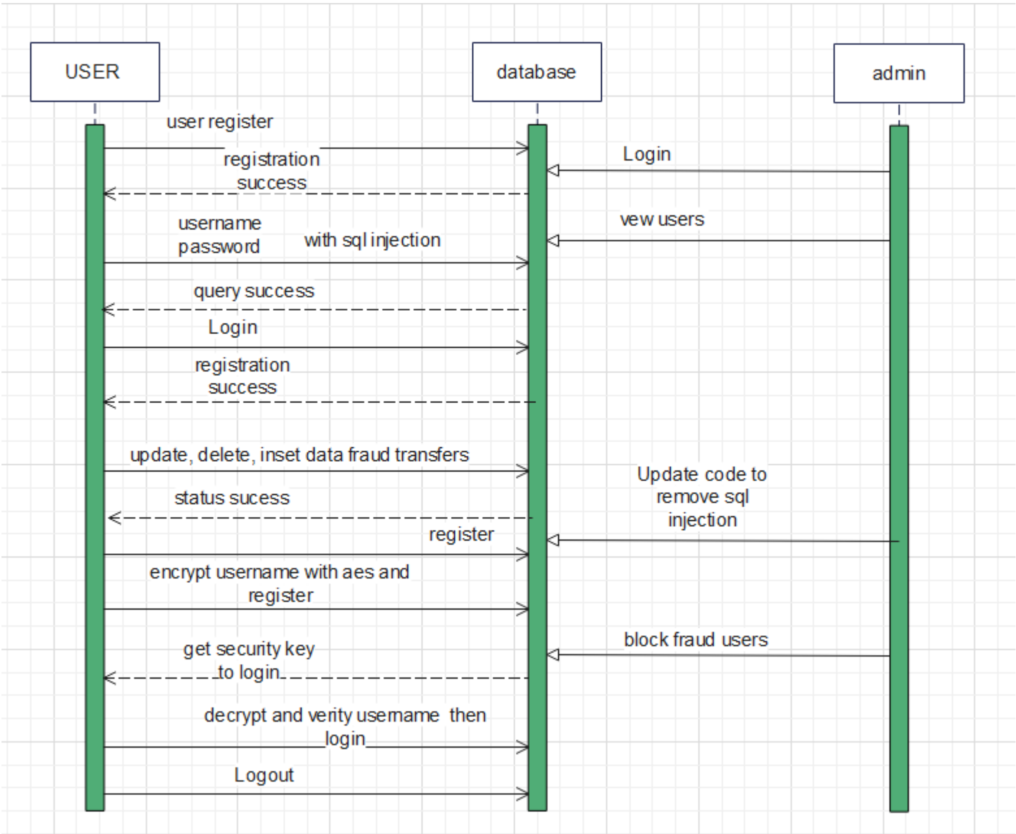


Figure 1.2.1 Sequence Diagram

The sequence diagram would show the following steps:

1. The user registers with a username and password.

2. The registration is successful, and the username and password are stored in the database.

3. The user attempts a login using the stored username and password.

4. The login is successful, and the user is able to access their account.

5. The user performs actions such as updating, deleting, or inserting data in their account.

6. The admin logs in and views the list of users.

7. The admin updates the code to remove any vulnerabilities to SQL injection.

8. The admin registers new users and blocks any suspicious or fraudulent accounts.

9. The user's encrypted username is decrypted with the security key during login.

4.1.3 Class Diagram

A class diagram serves as a visual blueprint for software systems, outlining the structure and connections between different components. In simple terms, it depicts the building blocks of a program, known as classes, and how they interact with each other. Each class represents a specific entity or object in the system, detailing its attributes (characteristics) and methods (actions). Associations between classes indicate relationships, illustrating how they collaborate to achieve system functionality. These diagrams are crucial for software development, providing a clear overview of the system's architecture, aiding in communication among developers, and facilitating the design and maintenance of complex software projects

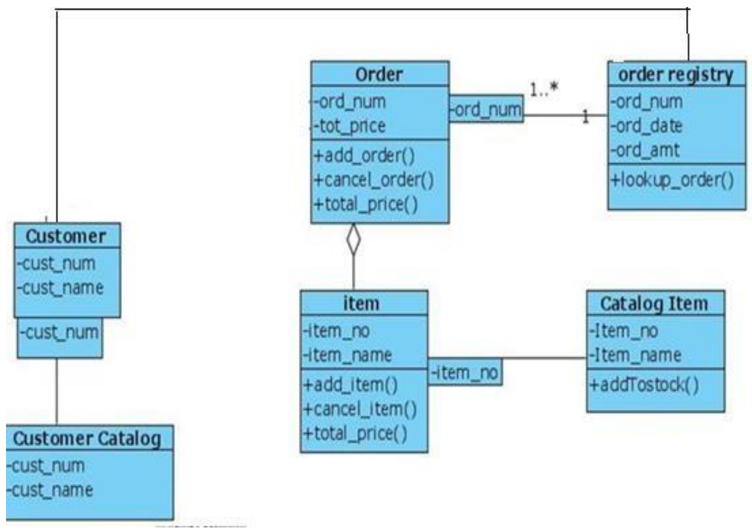


Figure 4.1.2.1 Class Diagram

4.1.4 Collaboration Diagram:

A collaboration diagram, also known as a communication diagram, is a type of interaction diagram in the Unified Modeling Language (UML) that illustrates the interactions and relationships between objects or components within a system to achieve a specific functionality or behavior. In a collaboration diagram, the emphasis is placed on how objects or components interact with each other to accomplish a task or respond to an event. It typically represents these interactions through messages exchanged between objects, which are depicted as labeled arrows between the objects. These messages can represent method calls, signals, or other forms of communication. The main purpose of a collaboration diagram is to visualize the dynamic aspects of a system's behavior, showing how objects collaborate to fulfill certain requirements or achieve specific objectives. By depicting the flow of messages between objects, developers can gain insights into the runtime behavior of the system and identify potential design flaws or areas for optimization. Overall, collaboration diagrams serve as valuable tools for understanding and communicating the runtime behavior and interactions within a system, aiding in system design, analysis, and debugging processes.

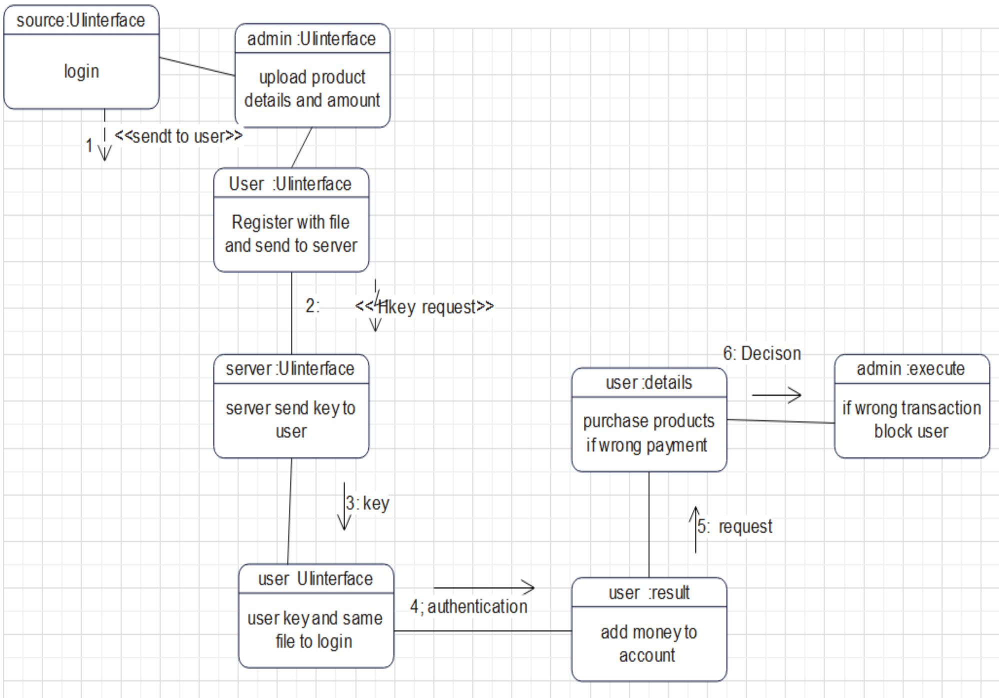


Figure 4.1.4.1 Collaboration Diagram

4.1.5 State Diagram:

A state diagram, also known as a state machine diagram or statechart diagram, is a graphical representation used to depict the various states of an object or system and the transitions between those states. It is a behavioral diagram that shows the different states an object can be in over time, as well as the events that cause transitions between these states. In a state diagram, each state is represented as a node, typically drawn as a rounded rectangle, and transitions between states are represented by arrows, showing the flow from one state to another. Events trigger these transitions, and conditions or actions associated with transitions may also be depicted. Additionally, the diagram may include initial and final states to denote the starting and ending points of the system's behavior.

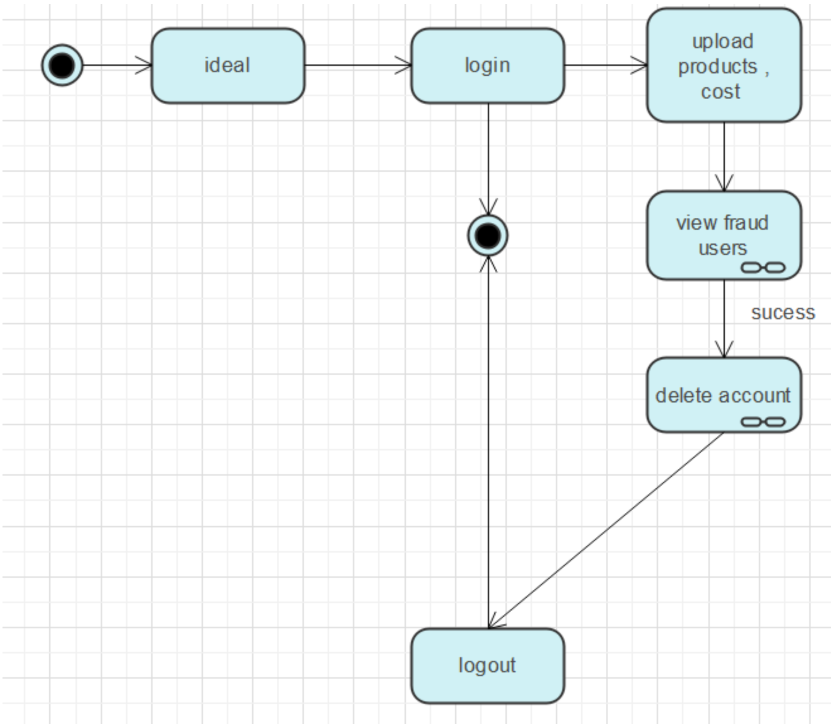


Figure 4.1.5.1 Admin State Diagram

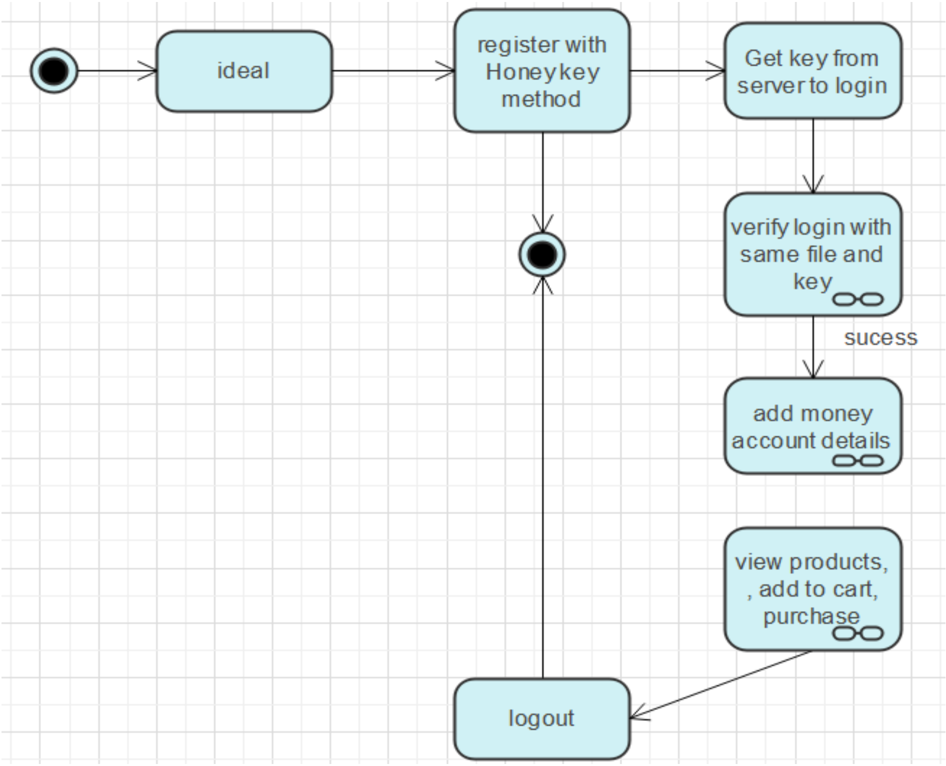


Figure 4.1.5.2 User State Diagram

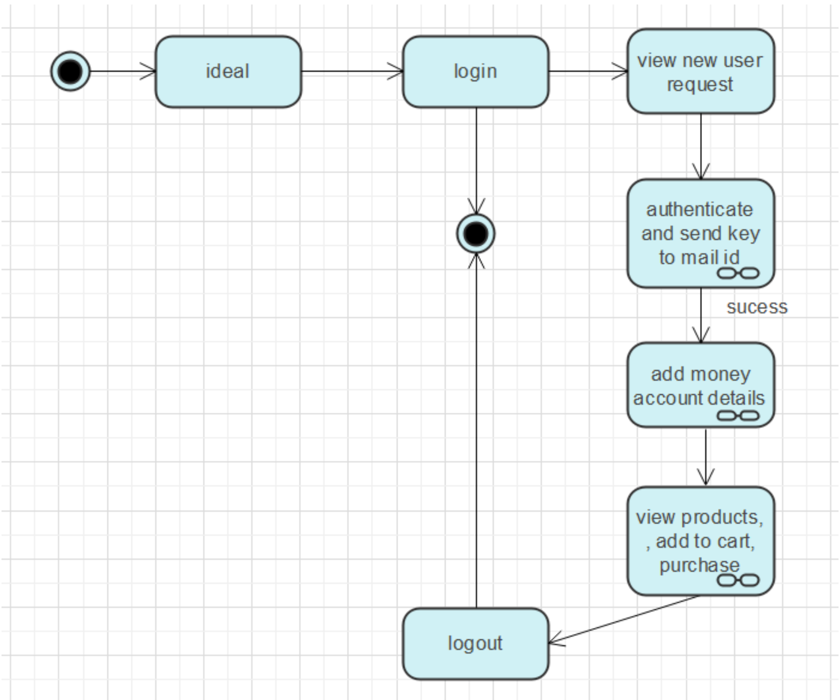


Figure 4.1.5.3 Server State Diagram

4.1.6 Activity Diagram

An activity diagram is a type of UML (Unified Modeling Language) diagram used to visually represent the flow of control or the sequence of activities in a system, process, or workflow. It typically consists of various activities, represented by nodes, connected by arrows to indicate the sequence in which the activities are performed. Each activity can represent a single operation, a group of operations, or a decision point within the system. The arrows between activities show the order in which the activities are executed, with forks and joins indicating parallel or concurrent execution paths. Activity diagrams are useful for modeling complex business processes, software workflows, or any sequence of actions that involve multiple actors or components. They provide a clear and concise visualization of how the system functions, helping stakeholders to understand, analyze, and communicate the behavior and logic of the system effectively

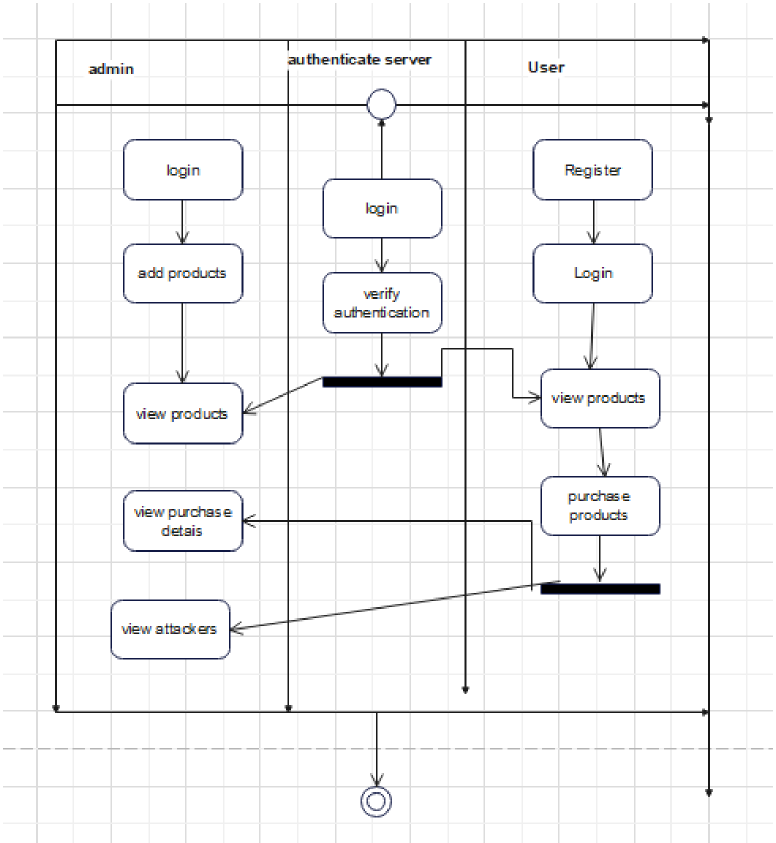


Figure 4.1.6.1 Activity Diagram

4.1.7 Component Diagram:

A component diagram is a type of UML (Unified Modeling Language) diagram that depicts the components of a system and their relationships. It illustrates how various software components or modules interact with each other to form a coherent system architecture. Components are represented as rectangles, each encapsulating the functionality they provide. Dependencies between components are depicted using arrows, indicating the direction of communication or dependency. Typically, a component diagram is used during the design phase of software development to visualize the high-level structure of the system and its constituent parts. It helps software architects and developers to understand the organization of the system, identify dependencies between components, and plan for integration and deployment. Additionally, component diagrams facilitate communication among stakeholders by providing a clear representation of the system's architecture and its components.

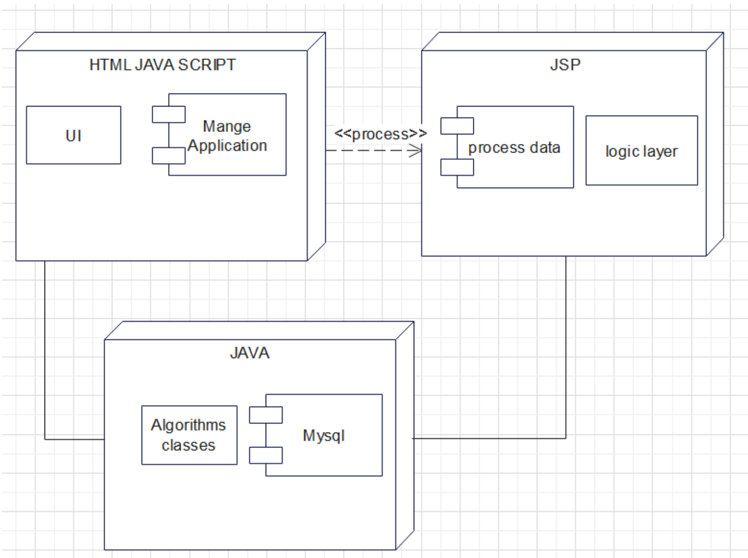


Figure 4.1.7.1 Component Diagram

4.1.8 Deployment Diagram

A deployment diagram is a type of UML (Unified Modeling Language) diagram that illustrates the physical deployment of software components within a computing environment. It shows how software artifacts, such as executable files, libraries, and databases, are distributed across hardware nodes, such as servers, workstations, or devices. Nodes are represented as boxes, while artifacts are depicted as rectangles, often connected by deployment relationships indicating which artifacts are hosted or executed on which nodes. Deployment diagrams are particularly useful for understanding the configuration and arrangement of software and hardware elements in a distributed system, including networks, servers, routers, and other infrastructure components. They help system architects and developers to plan for deployment, scalability, and performance optimization by visualizing how software components are distributed across different nodes and how they interact with each other over a network. Deployment diagrams also facilitate communication among stakeholders by providing a clear representation of the system's physical architecture and deployment topology.

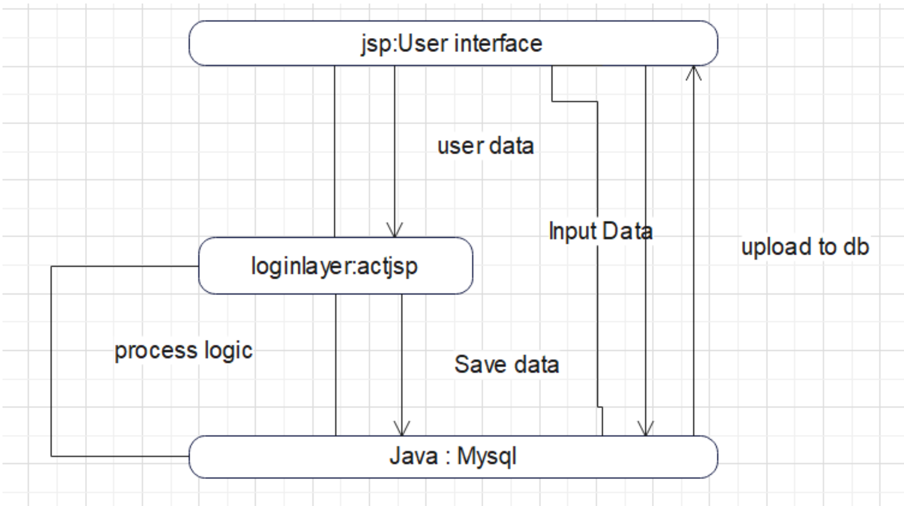


Figure 4.1.8.1 Deployment Diagram

5. IMPLEMENTATION

5.1 MODULES:

5.1.1 Online Shopping Website:

Using this module web application is developed which has online shopping features where seller can use admin module to upload products and buyer can view products and purchase. This application provides option for payment, add products to cart, view products, search products get conformation from admin on purchase, use attacker module to show internal atatcks. Show security methods to secure authentication process.

Figure5.1.1.1 shopping website home page

5.1.2 Admin Module:

This module is part on online shopping website where admin and login to application add products with cost and product details and verify users as attackers or normal users and block users who are attackers. Admin can verify users for purchasing products and get confirmation.

figure 5.1.2.1 Admin Module

5.1.3 User Module:

This module is part of online shopping website where users can register with application by entering valid user name and password along with text file data which is used for every time login. User must give same text fiile every time and apply Honey Password-Authenticated Key Exchange when he login to application which will encrypt and send key to authentication server who will verity and validate user . If Password-Authenticated Key Exchange is success then only user is considered as normal user else he is considered as attacker.

Figure 5.1.3.1 user Module

5.1.4 Authentication Server Module:

This module is used as middle layer between user registration process and login process verification for verifying Honey Password-Authenticated Key Exchange process. Every time new user registers this server will store a security key which is unique based on user input data . If same data is uploaded by user while login then only authentication server will give validation else authentication exchange will be failed.

5.1.5 Honey key Mechanism:

Honeyword Mechanism Honeyword technique [15] has been proposed to detect password leakage for password-over-TLS. As shown in Figure 1b, honeyword mechanism directly generates several honeywords and stores them (in the form of hash value) on the authentication server along with the real password. It stores the index of the real password in the list on another server called honeychecker. When one logs in with a username U and a password pw (where pw is sent to the authentication server via the TLS channel), then the authentication server checks if pw is a sweetword: 1) If it is not, deny this login. 2) If it is the i-th sweetword, the authentication server sends (U, i) to the honeychecker (via a secure channel). Then the honeychecker checks if the index i is correct for U: a) If it is, allow this login. b) Otherwise, raise an alarm of password leakage and take actions according to the pre-defined security policy. This mechanism only does slight modification on the server side for password-over-TLS, and therefore maintains its advantages on deployability. Besides, since its interface is very simple, the honeychecker can be easily enhanced to avoid being compromised

5.1.7 Honey Key encryption:

Honey Encryption Honey encryption [25], [32] is a novel encryption method, which can yield decoy messages for incorrect keys as shown in Figure 4b. It introduces a probabilistic encoder to encode the message M to a (fixed- length) uniform bit string S and then encrypts S by a carefully-chosen traditional encryption scheme (see Figure 5). The encoder is designed according to the message distribution M, which can be uniform or nonuniform (e.g., for the password vaults [32]). The encoder should guarantee that decoding a random bit string will yield a message sampled from M. Formally, for an arbitrary adversary (maybe with unlimited computing resources) A, (M0, S0) and (M1, S1) are indistinguishable (we denote (M0, S0) ~ (M1, S1)), where S0 ←$ {0, 1} l (i.e., randomly selecting a l-bit string), M0 ← Decode(S0), M1 ←p M (i.e., sampling a message from M according to the message distribution p), S1 ← Encode(M1), and l is the length of the bit strings. More specifically, |Pr[A(M0, S0) = 1 : S0 ←$ {0, 1} l , M0 ← Decode(S0)] − Pr[A(M1, S1) = 1 : M1 ←$ M, S1 ←

Encode(M1)]| is negligible. Please note that the traditional encryption scheme used in honey encryption should yield a random bit string for each incorrect keys. Therefore, for each incorrect key K′ , the honey encryption scheme will produce a l-bit string S ′ and further a plausible-looking message M′ on M. In the design of HPAKE, we use honey encryption to encrypt the user's private key kU , which is a uniformly random number on Zm2 . Designing an encoder for kU is simple. To encode kU , we directly select an integer number from [round(kU 2 l/m2),round((kU + 1)2l/m2)) (⊆ [0, 2 l)) as S, where round is the rounding function; to decode S, we find the corresponding interval and obtain kU . With the encoder, for each incorrect key rw, the honey encryption scheme can produce a plausible-looking private key on Zm2.

5.2 Sample Code:

5.2.1 Admin.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>HPAKE Honey Password-Authenticated Key Exchange for Fast and Safer
Online Authentication</title>
<meta name="keywords" content="" />
<meta name="description" content="" />
<link href="templatemo_style.css" rel="stylesheet" type="text/css" />
</head>
  <%
      if (request.getParameter("m1") != null) {%>
    <script>alert('Login Successfully');</script>
    <%}
      if (request.getParameter("m2") != null) {%>
    <script>alert('Login Failed ');</script>

    <%}
    %>
<body>
      <div class="templatemo_container">
```

```
<div id="templatemo_header">
<div id="templatemo_logo_area">
        <div id="templatemo_logo_left">
   
  </div>
    <h1><font size="4">HPAKE Honey Password-Authenticated Key
Exchange for Fast and Safer Online Authentication</h1>

  <div id="templatemo_logo_right">
   
  </div>
            </div>

</div>
<div id="templatemo_top_section">
    <div id="templaetmo_top_section_top">
  <div id="templatemo_top_section_glow">
        <div id="templatemo_menu">
    <div id="templatemo_menu_left">
    </div>
    <ul>
          <li><a href="index.html">Home</a></li>
          <li><a href="admin.jsp" class="current">Admin</a></li>
          <li><a href="user.jsp">User</a></li>
           <li><a href="server.jsp">Server</a></li>
    </ul>
             </div>
  <br><br>
  <h1>Abstract</h1>
  <a href="#"><img src="images/card1.jpg" alt="Free CSS Templates"
border="0" /></a>


        <div class="cleaner"></div>
        </div><!-- end of glow -->
  </div>
  <div id="templaetmo_top_section_bottom"></div>

</div>



  <form name="f" action="admin_act.jsp" method="post" >  <center>
```

```
                <h1><span><font color="white">Admin Login</font></span>
</h1><br>
<table>

      <tr>
           <td>
              <strong><font size="4" color="white">Username:</font></strong>
             <input type="text" name="username" id="userName1" placeholder=
Username style="height:30px; width:170px"></input>
           </td>
      </tr>



      <tr>
           <td>
             <strong><font size="4" color="white">Password: </font></strong>
             <input type="password" name="password" id="password1" placeholder=
Password style="height:30px; width:170px"></input>
           </td>
      </tr>



           <tr>
           <td>
              <input type="submit" value="Login"  style="height:30px; width:65px" />
           </td>
        </tr>


         </table>
     </center>
    </form>
        <br>




      <div class="cleaner"></div>

         </div><!-- End Of Container -->



   <div id="templatemo_footer">
         <a href="#"></a>
   </div>
```

```
</body>
</html>

<%@page import="java.sql.*"%>
<%@page import="frauddetection.Dbconnection"%>
<%@ page session="true" %>
<%

   String username = request.getParameter("username");
   String password = request.getParameter("password");


   try{


       Connection con = Dbconnection.getConnection();
       Statement st =con.createStatement();

       ResultSet rs = st.executeQuery("select * from admin where
username='"+username+"' and password='"+password+"'");

       if(rs.next())
       {

           response.sendRedirect("admin_home.jsp?m1=success");

       }
      else
       {
           response.sendRedirect("admin.jsp?m2=LoginFail");
               }
               }



   catch(Exception e)
   {

   }
%>
```

5.2.2 Server.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>HPAKE Honey Password-Authenticated Key Exchange for Fast and Safer
Online Authentication</title>
<meta name="keywords" content="" />
<meta name="description" content="" />
<link href="templatemo_style.css" rel="stylesheet" type="text/css" />
</head>
  <%
        if (request.getParameter("m1") != null) {%>
      <script>alert('Login Successfully');</script>
      <%}
        if (request.getParameter("m2") != null) {%>
      <script>alert('Login Failed ');</script>

      <%}
      %>
<body>
        <div class="templatemo_container">
        <div id="templatemo_header">
        <div id="templatemo_logo_area">
                <div id="templatemo_logo_left">
           
          </div>
             <h1><font size="4">HPAKE Honey Password-Authenticated Key
Exchange for Fast and Safer Online Authentication</h1>

          <div id="templatemo_logo_right">
           
          </div>
                          </div>

      </div>
      <div id="templatemo_top_section">
           <div id="templaetmo_top_section_top">
        <div id="templatemo_top_section_glow">
                <div id="templatemo_menu">
             <div id="templatemo_menu_left">
```

```
                </div>
                <ul>
                        <li><a href="index.html">Home</a></li>
                        <li><a href="admin.jsp">Admin</a></li>
                        <li><a href="user.jsp" >User</a></li>
                         <li><a href="server.jsp" class="current">Server</a></li>
               </ul>
                            </div>
             <br><br>
             <h1>Abstract</h1>
             <a href="#"><img src="images/card1.jpg" alt="Free CSS Templates"
border="0" /></a>


                    <div class="cleaner"></div>
                  </div><!-- end of glow -->
         </div>
         <div id="templaetmo_top_section_bottom"></div>

     </div>




        <form name="f" action="serveract.jsp" method="post" >  <center>
                <h1><span><font color="white">Server Login</font></span>
</h1><br>
<table>

     <tr>
           <td>
               <strong><font size="4" color="white">Username:</font></strong>
             <input type="text" name="username" id="userName1" placeholder=
Username style="height:30px; width:170px"></input>
           </td>
      </tr>



     <tr>
           <td>
              <strong><font size="4" color="white">Password: </font></strong>
             <input type="password" name="password" id="password1" placeholder=
Password style="height:30px; width:170px"></input>
           </td>
     </tr>
```

```
            <tr>
            <td>
                <input type="submit" value="Login"  style="height:30px; width:65px" />
            </td>
        </tr>


          </table>
      </center>
     </form>
        <br>



       <div class="cleaner"></div>

          </div><!-- End Of Container -->


    <div id="templatemo_footer">
          <a href="#"></a>
    </div>

</body>
</html>


<%@page import="java.sql.*"%>
<%@page import="frauddetection.Dbconnection"%>
<%@ page session="true" %>
<%

    String username = request.getParameter("username");
    String password = request.getParameter("password");


    try{



        Connection con = Dbconnection.getConnection();
        Statement st =con.createStatement();
```

```
        ResultSet rs = st.executeQuery("select * from server where
username='"+username+"' and password='"+password+"'");

        if(rs.next())
        {

            response.sendRedirect("shome.jsp?m1=success");

        }
       else
        {
            response.sendRedirect("server.jsp?m2=LoginFail");
                }
                }



    catch(Exception e)
    {

    }
%>
```

5.2.3 User.html

```
<!DOCTYPE html PUBLIC "-//W3C//DTD XHTML 1.0 Transitional//EN"
"http://www.w3.org/TR/xhtml1/DTD/xhtml1-transitional.dtd">
<html xmlns="http://www.w3.org/1999/xhtml">
<head>
<meta http-equiv="Content-Type" content="text/html; charset=utf-8" />
<title>HPAKE Honey Password-Authenticated Key Exchange for Fast and Safer
Online Authentication</title>
<meta name="keywords" content="" />
<meta name="description" content="" />
<link href="templatemo_style.css" rel="stylesheet" type="text/css" />
</head>
    <%
            if (request.getParameter("msg") != null) {%>
        <script>alert('User Card Added Successfully');</script>
        <%}
            if (request.getParameter("msg1") != null) {%>
        <script>alert('Login Failed ');</script>
```

```
    <%}
    %>
<body>

  <script>
  function validateform(){

var mobile = document.getElementById('txtphoneno');
 if (mobile.value == "" || mobile.value == null) {
      alert("Please enter your Card No.");
      return false;
    }
    if (mobile.value.length < 16 || mobile.value.length > 16) {
      alert("Card No. is not valid, Please Enter 16 Digit Card No.");
      return false;
    }



  }
  </script>
      <div class="templatemo_container">
      <div id="templatemo_header">
      <div id="templatemo_logo_area">
            <div id="templatemo_logo_left">
         
        </div>
          <h1><font size="4">HPAKE Honey Password-Authenticated Key
Exchange for Fast and Safer Online Authentication</h1>

        <div id="templatemo_logo_right">
         
        </div>
                  </div>
```

```
        </div>
        <div id="templatemo_top_section">
              <div id="templaetmo_top_section_top">
            <div id="templatemo_top_section_glow">
                        <div id="templatemo_menu">
                <div id="templatemo_menu_left">
                </div>
              <ul>
                          <li><a href="userhome.jsp">Home</a></li>
                          <li><a href="user_card.jsp" class="current">Account
Details</a></li>
                          <li><a href="user_products.jsp">View Products</a></li>
                          <li><a href="index.html">Logout</a></li>
               </ul>
                              </div>
             <br><br>
             <h1>Abstract</h1>
             <a href="#"><img src="images/card1.jpg" alt="Free CSS
Templates" border="0" /></a>


                          <div class="cleaner"></div>
                        </div><!-- end of glow -->
            </div>
            <div id="templaetmo_top_section_bottom"></div>

        </div>




                <center>
            <h1>Add HPAKE Honey Password-Authenticated Key Exchange for
Fast and Safer Online Authentication Details</h1>
    <br><br>
    <form name="myform" id="loginForm" action="user_addcardact.jsp"
method="post"  onsubmit="return validateform()" >
             <center>
<table>

        <tr>
            <th><font color="white" size="4">Card Holder Name :</th>
              <th><input type="text" name="cname" placeholder= "Card Holder
Name" style="height:30px; width:170px"></input>
              </th>
```

```
        </tr>

        <tr>
              <th><font color="white" size="4">Card Number: </th>
              <th><input type="text" name="cno" placeholder= "Card Number"
id="txtphoneno" onkeypress="return isNumber(event)"style="height:30px;
width:170px"></input>
              </th>
        </tr>

      <tr>
              <th><font color="white" size="4">CVV: </th>
              <th><input type="text" name="cvv"  placeholder= "CVV"
style="height:30px; width:170px"></input>
              </th>
        </tr>


         <tr>
              <th><font color="white" size="4">Expiry Date :</th>
              <th><input type="text" name="edate" placeholder= "Expiry Date"
style="height:30px; width:170px"></input>
              </th>
        </tr>


              <tr>
              <td>
                    <input type="submit" value="ADD" style="height:30px;
width:65px"/>
             </td>
           </tr>

           </table>
                    </center>
     </form>


        <hr>

         <br><br>


        <div class="cleaner"></div>
```

```
        </div><!-- End Of Container -->


    <div id="templatemo_footer">
        <a href="#"></a>
    </div>

</body>
</html>

<%@page import="java.sql.*"%>
<%@page import="frauddetection.Dbconnection"%>
<%@ page session="true" %>
<%

   String user = session.getAttribute("user").toString();
   user ="chotu";
    int id=0;
    String cname = request.getParameter("cname");
    String cno = request.getParameter("cno");
    String cvv = request.getParameter("cvv");
    String edate = request.getParameter("edate");
    int money=0;

    try{

        Connection con=Dbconnection.getConnection();
         Statement st = con.createStatement();
        PreparedStatement ps = con.prepareStatement("insert into card
values(?,?,?,?,?,?,?)");
         ps.setInt(1,id);
         ps.setString(2,cname);
         ps.setString(3,cno);
         ps.setString(4,cvv);
         ps.setString(5,edate);
         ps.setString(6,user);
         ps.setInt(7,money);
         ps.executeUpdate();
response.sendRedirect("user_addcard.jsp?msg=success");

}
    catch(Exception e)
```

6. EXPERIMENT SCREENSHOTS

6.1 Home Page

Figure 6.1.1 Home Page

6.2 Admin Page

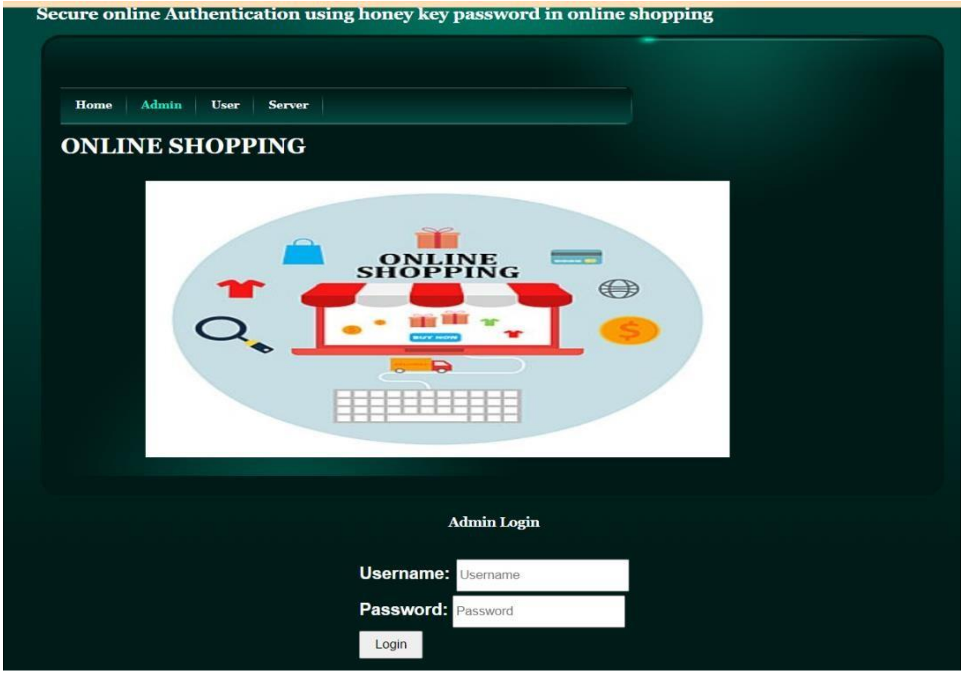


Figure 6.2.1 Admin login

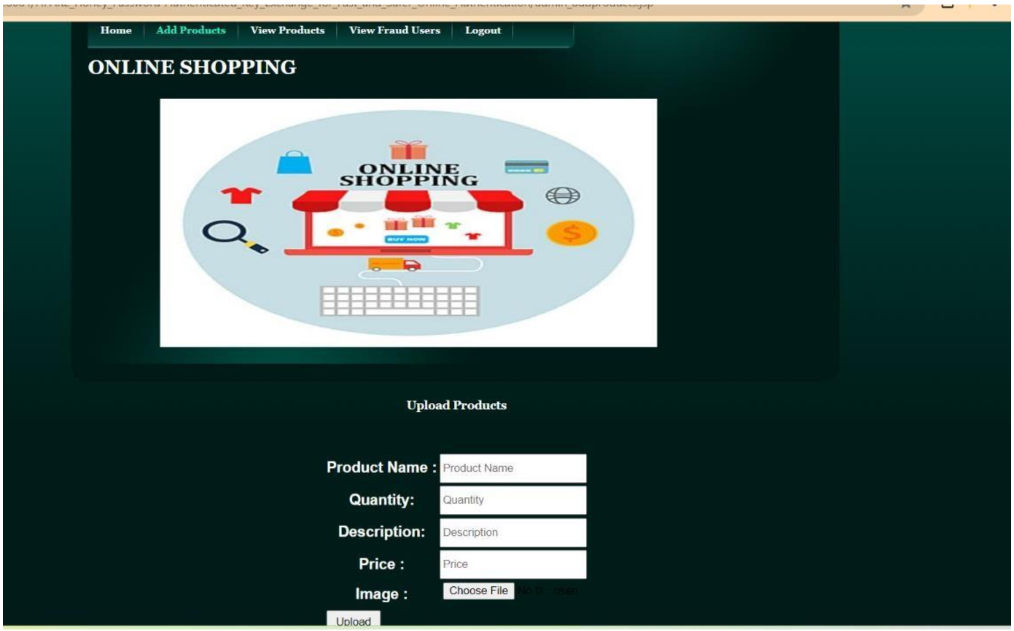


Figure 6.2.2 Add products page

Figure 6.2.3 View products page

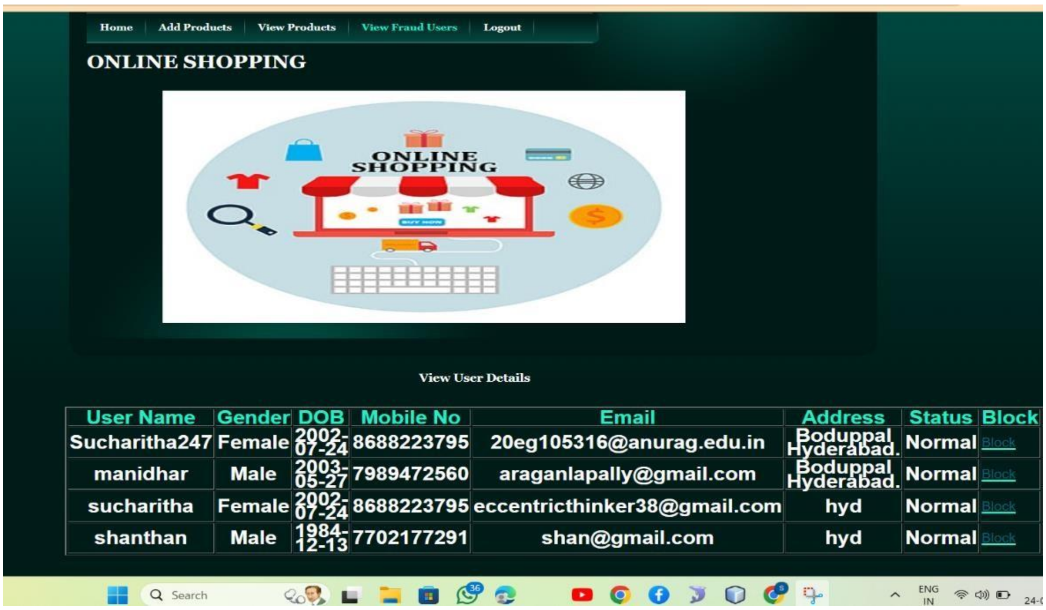


Figure 6.2.4 View users page

6.3 User Page

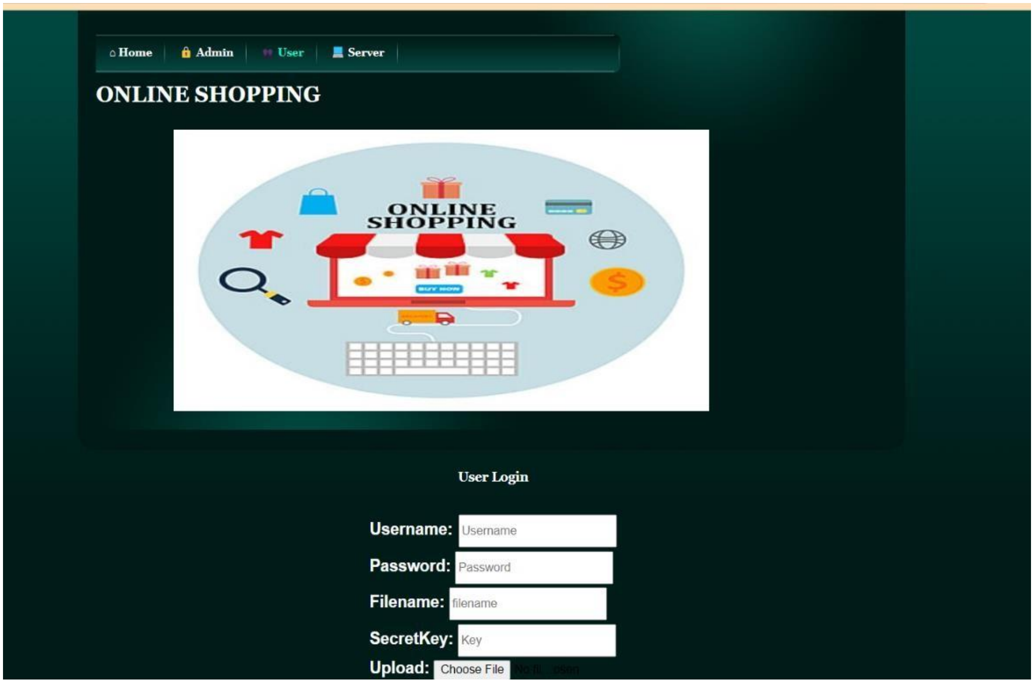


Figure 6.3.1 User Login

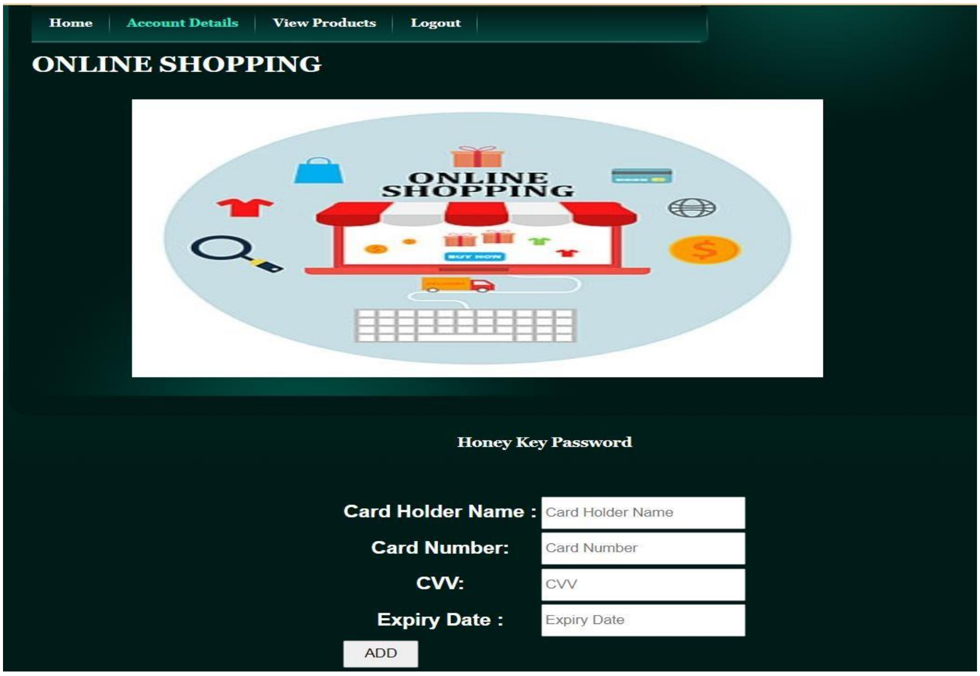


Figure 6.3.2 Account details

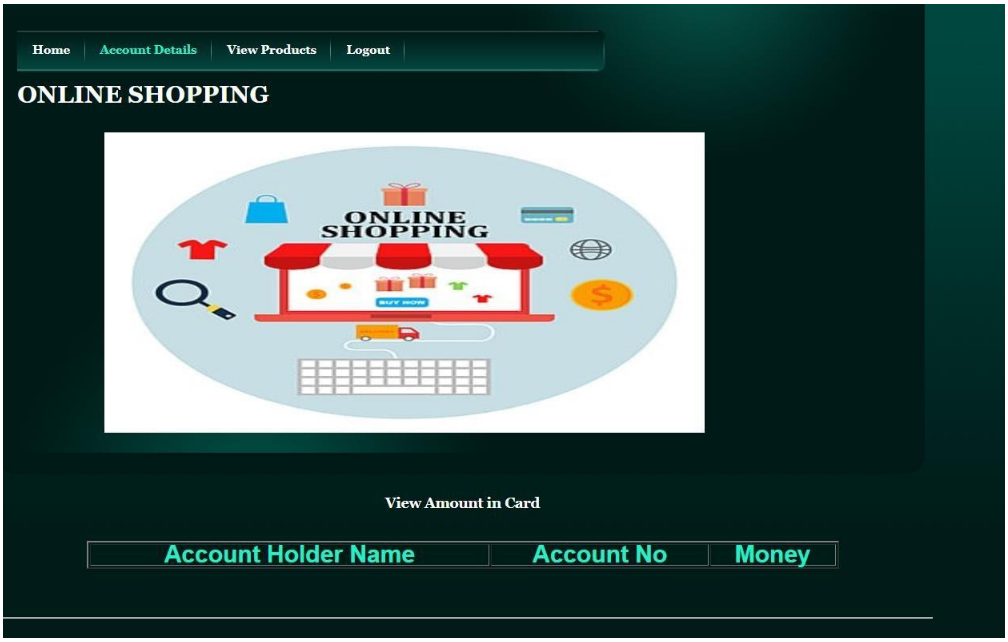


Figure 6.3.3 Account details

6.4 Server Page

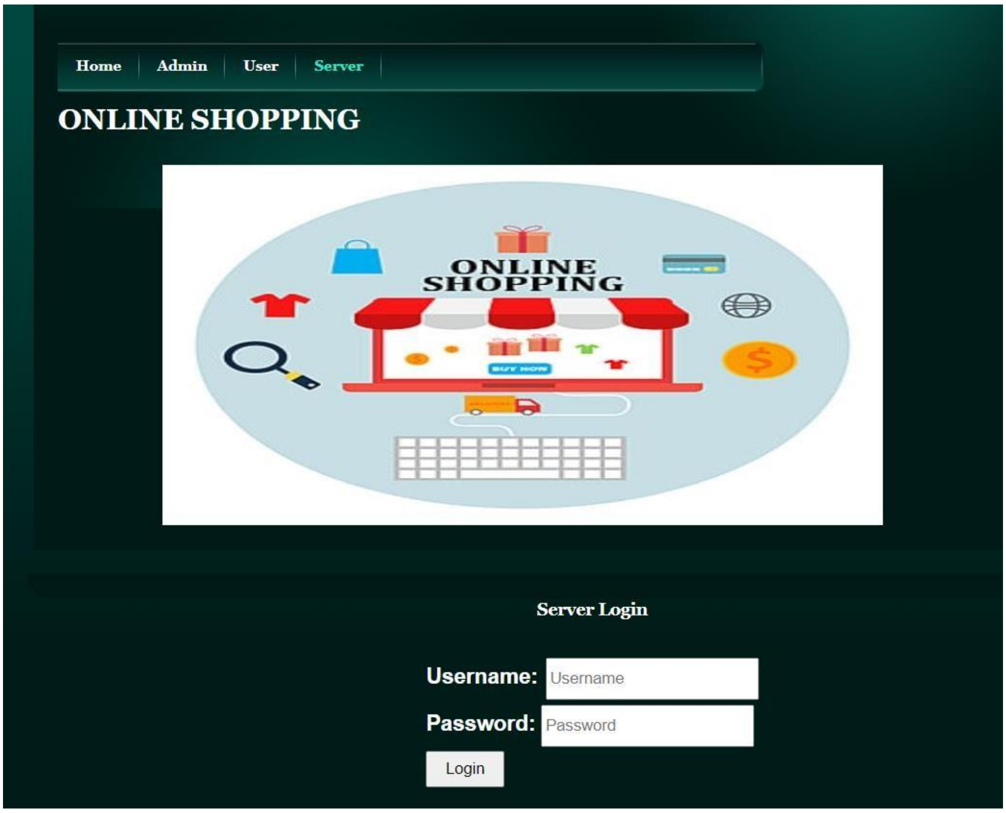


Figure 6.4.1 Server login

View Users

User Name	Email	data	Key	Send
Sucharitha247	20eg105316@anurag.edu.in	hii helloo how are you	7ZIONkmY85NeN6yStHpgJg==	Send
manidhar	araganlapally@gmail.com	hii helloo how are you	71JWC+qMZaQUPazk4on3Hw==	Send
sucharitha	eccentricthinker38@gmail.com	hii helloo how are you	Mqxtd8MFzXBSDXImvjb4iw==	Send
shanthan	shan@gmail.com	hii helloo how are you	dgYhNjdl5a5/eZagxhKZVA==	Send
ewfenfnre	sucharitha2477@gmail.com	hii helloo how are you	8aKQGAtM1FUtDcumo1a0kA==	Send

Figure 6.4.2 View Users

7. OBSERVATIONS

- **Experimental Setup**

Software Requirements:

- Operating system : Windows XP/7/10.
- Coding Language : Java
- Tool : Netbeans
- Database : MYSQL

Hardware Requirements:

- System : Pentium IV 2.4 GHz.
- Hard Disk : 200 GB.
- Floppy Drive : 1.44 Mb.
- Ram : 1 GB

Parameters with formulas:

Parameter

- Security parameter κ.
- 2HashDH:
  1) m_1 is a primer number, G_1 is a m_1-order cyclic group, and g_1 is a generator of G_1. The length of m_1 is a polynomial function of κ.
  2) H_1, H_1' are two hash functions with ranges $\{0,1\}^{l_1}$ and G_1, respectively. The PRF $F_s(x)$ is $H_1(x, H_1'(x)^s)$. l_1 is a polynomial function of κ.
- HMQV:
  1) m_2 is a primer number, G_2 is a m_2-order cyclic group, g_2 is a generator of G_2. The length of m_2 is a polynomial function of κ.
  2) H_2, H_2' are two hash functions with ranges $\mathbb{Z}_{m_2}$ and $\{0,1\}^{l_2}$. l_2 is the length of the session key, which is a polynomial function of κ.
- A honey encryption scheme $(\mathrm{Enc}, \mathrm{Dec})$ for the message space $\mathbb{Z}_{m_2}$.
- A honeyword generation algorithm Gen.

Initialization (via a secure channel)

- The client C picks $s \leftarrow\!\$\, \mathbb{Z}_{m_1}$ as the secret key of S in 2HashDH, and computes $rw \leftarrow H_1(pw, H_1'(pw)^s)$; computes $k_U \leftarrow\!\$\, \mathbb{Z}_{m_2}, K_U \leftarrow g_2^{k_U}$ to generate the private/public keys (k_U, K_U) for U in HMQV; computes $c \leftarrow \mathrm{Enc}_{rw}(k_U)$ to yield the ciphertext c of k_U using the key rw; sends (U, s, K_U, c) to S.
- Getting (U, s, K_U, c) from the client C, the authentication server S computes $k_S \leftarrow\!\$\, \mathbb{Z}_{m_2}, K_S \leftarrow g_2^{k_S}$ to generate its private/public keys (k_S, K_S) in HMQV; S generates $t-1$ honeywords $hw_i \leftarrow \mathrm{Gen}$ for i from 1 to $t-1$, the corresponding random honeyword $rw_i \leftarrow H_1(pw, H_1'(hw_i)^s)$, and the honey private/public keys $k_{U,i} \leftarrow \mathrm{Dec}_{rw_i}(c)$, $K_{U,i} \leftarrow g_2^{k_{U,i}}$; randomly shuffles $K_{U,i}$ $(1 \le i \le t-1)$ with K_U, and sends the index i_{r} of the real one (with the ID U) to the honeychecker HC; stores s, c with the shuffled $K_{U,i}$ $(1 \le i \le t)$.
- Getting (U, i_{r}) from the authentication server S, the honeychecker HC stores it.

Authentication (via a server-authenticated channel)

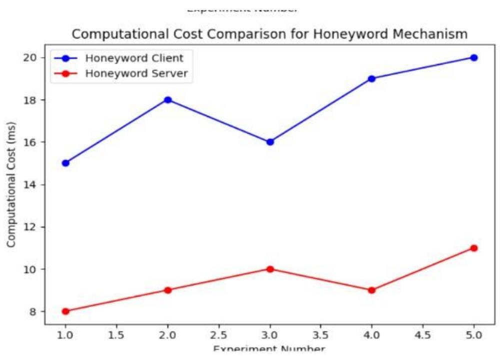
- C picks $r \leftarrow\!\$\, \mathbb{Z}_{m_1}$ and computes $\alpha \leftarrow H_1'(pw)^r$; picks $x \leftarrow\!\$\, \mathbb{Z}_{m_2}$ and computes $X \leftarrow g_2^x$; sends (U, X, α) to S.
- Getting (U, X, α) from C, S picks $y \leftarrow\!\$\, \mathbb{Z}_{m_2}$ and computes $Y \leftarrow g_2^y, \beta \leftarrow \alpha^s$; sends (Y, β, c, K_S) to C; computes $SK_i \leftarrow H_2((X K_{U,i}^{H_2'(X,K_S)})^{y + H_2'(Y,K_{U,i}) k_S})$ for i from 1 to t.
- Getting (Y, β, c, K_S) from S, C computes $rw \leftarrow H_1(pw, \beta^{1/r})$, $k_U \leftarrow \mathrm{Dec}_{rw}(c)$; computes $SK \leftarrow H_2((Y K_S^{H_2'(Y,K_U)})^{x + H_2'(X,K_S) k_U})$ and further uses SK for data transmission (or other purposes).
- S checks if the session key SK used by C is one of $\{SK_i\}_{i=1}^t$:
  1) If it is not, deny this session.
  2) If it is the i-th one, S sends (U, i) to HC (via a secure channel). Then HC checks if the index i is correct for U (i.e., equal to i_{r}):
     a) If it is, allow this session.
     b) Otherwise, raise an alarm of password leakage and take actions according to the pre-defined security policy.

Parameter	Previous methods	Proposed method
Computational Cost (Client)	Baseline	1 additional multi-exponentiation
Computational Cost (Server)	Not specified	t-1 additional multi-exponentiations (for honey session keys, parallelizable)
Communication Cost (Server-Honeychecker)		2 additional rounds
Communication Cost (Client-Server)	5 rounds	2 rounds
Communication Delay (Client-Server)	Dominant factor	Significantly reduced

7.1 Comparison between Existing and Proposed Methods

8. DISCUSSION OF RESULTS

8.1 Computational Cost analysis



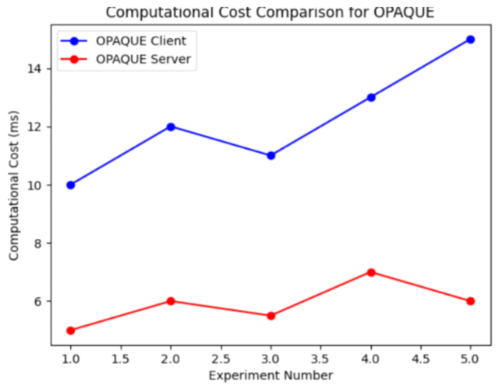


Figure 8.1.1 Computational cost of Honeyword Mechanism

Figure 8.1.2 Computational cost of Opaque

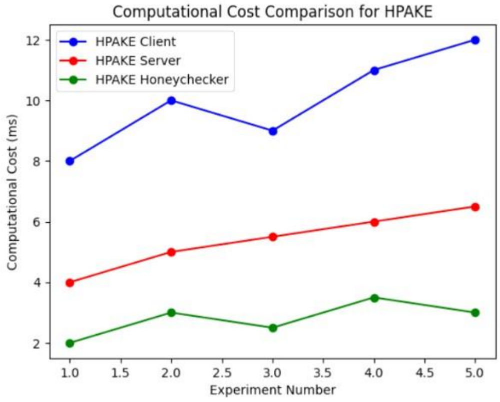


Figure 8.1.3 Computational cost of Hpake

The graph compares the communication costs of the following authentication methods: Honeyword Mechanism,Opaque, Hpake.

Honeyword: 55.3 ms
OPAQUE: 62.2 ms
HPAKE: 58.7 ms

The graph emphasizes that HPake has a computational cost of 58.7 ms, which competitive compared to other methods but not the lowest.

8.2 Communication cost analysis

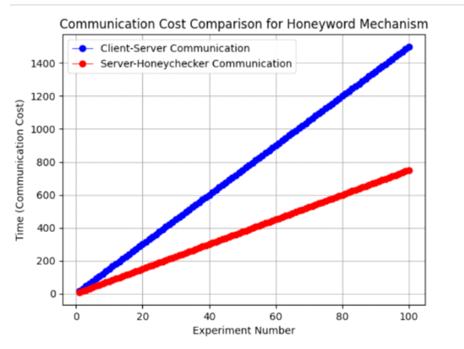


Figure 8.2.1 Communication cost of Honeyword Mechanism

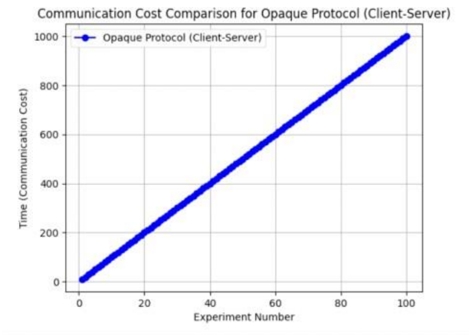


Figure 8.2.2 Communication cost of Opaque

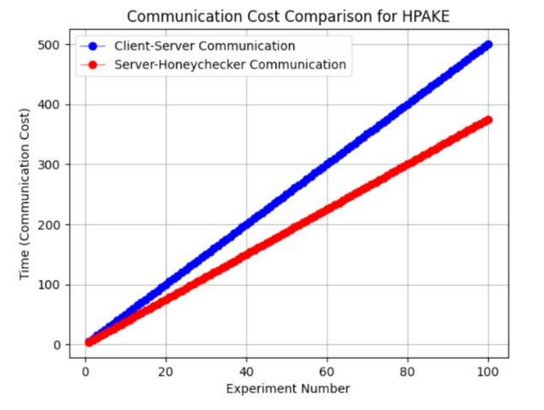


Figure 8.2.3 Communication cost of Hpake

The graph compares the communication costs of the following authentication methods : Honeyword Mechanism,Opaque,Hpake

Communication costs (in milliseconds) for each authentication method

Honeyword: 42.7 ms

OPAQUE: 50.48 ms

HPAKE: 47.01 ms

The graph clearly shows the communication cost comparison between the selected authentication methods. HPAKE has a communication cost of 47.01 ms, which is competitive compared to other methods like aPAKE and OPAQUE.

9. CONCLUSION:

We propose the notion of HPAKE, which is the first of its type, achieving the advantages of the honeyword and aPAKE techniques, i.e., detecting the password leakage caused by external attackers and preventing the insider from getting the password plaintext. Using OPAQUE, honeyword mechanism, and honey encryption, we build a concrete HPAKE construc tion. To analyze the security of our design, we propose a game-based security model and formally prove the security of our design in this model. We implement and deploy the proposed scheme in the real-world environment. The experi mental results show that our design is efficient for the real world applications.

Future Scope:

1. **Integration with UPI Payments**:
   - Implementing a secure integration of HPAKE with UPI (Unified Payments Interface) to facilitate secure transactions between users and merchants.
   - Develop APIs or SDKs to allow seamless integration of HPAKE with UPI payment systems.
   - Ensure compliance with UPI security standards and regulations to safeguard user financial data.

2. **Enhanced Security Measures**:
   - Strengthen security measures within HPAKE to handle financial transactions, including robust encryption algorithms and secure key management practices.
   - Implement multi-factor authentication methods to enhance security during UPI transactions, such as OTP (One-Time Password) verification or biometric authentication.

3. **Real-time Alert System**:

- Develop a real-time alert system that notifies users in case of any interruptions or suspicious activities related to their debit/credit cards.
- Integrate the alert system with HPAKE to provide timely notifications through various channels such as mobile apps, email, or SMS. - Include features to allow usersto verify and take immediate action on alerts, such as reporting unauthorized transactions or blocking compromised cards.

10. REFERENCES / BIBILOGRAPHY

[1] J. Bonneau, C. Herley, P. C. Oorschot, and F. Stajano, “The quest to replace passwords: A framework for comparative evaluation of web authentication schemes,” in Proc. IEEE S&P 2012, pp. 553–567.

[2] N. Huaman, S. Amft, M. Oltrogge, Y. Acar, and S. Fahl, “They would do better if they worked together: The case of interaction problems between password managers and websites,” in Proc. IEEE S&P 2021, pp. 1367–1381.

[3] D. Pasquini, A. Gangwal, G. Ateniese, M. Bernaschi, and M. Conti, “Improving password guessing via representation learning,” in Proc. IEEE S&P 2021, pp. 265–282.

[4] W. Li and J. Zeng, “Leet usage and its effect on password security,” IEEE Trans. Inform. Foren. Secur., vol. 16, pp. 2130–2143, 2021.

[5] “Have i been pwned?” [Online]. Available: https://haveibeenpwned.com

[6] “Yahoo! data breaches.” [Online]. Available: https://en.wikipedia.org/w iki/Yahoo! data breaches

[7] “Yahoo tries to settle 3-billion-account data breach with $118 million payout.” [Online]. Available: https://arstechnica.com/tech-policy/2019/0 4/yahoo-tries-to-settle-3-billion-account-data-breach-with-118-million -payout/

[8] Z. Whittaker, “Github says bug exposed some plaintext passwords,” ht tps://www.zdnet.com/article/github-says-bug-exposed-account-passwor ds/, 2018.

[9] S. M. Bellovin and M. Merritt, “Augmented encrypted key exchange: a password-based protocol secure against dictionary attacks and password file compromise,” in Proc. ACM CCS 1993, pp. 244–250.

[10] V. Boyko, P. MacKenzie, and S. Patel, “Provably secure passwordauthenticated key exchange using diffie-hellman,” in Proc. EUROCRYPT 2000. Springer, pp. 156–171.

[11] C. Gentry, P. MacKenzie, and Z. Ramzan, “A method for making password-based key exchange resilient to server compromise,” in Proc. CRYPTO 2006. Springer, pp. 142–159.

[12] S. Jarecki, H. Krawczyk, and J. Xu, “OPAQUE: An asymmetric PAKE protocol secure against pre-computation attacks,” in Proc. EUROCRYPT 2018. Springer, pp. 456–486.

[13] M. Abdalla, M. Barbosa, T. Bradley, S. Jarecki, J. Katz, and J. Xu, “Universally composable relaxed password authenticated key exchange,” in Proc. CRYPTO 2020. Springer, pp. 278–307.

[14] S. Smyshlyaev, N. Sullivan, and A. Melnikov, “[cfrg] results of the PAKE selection process,” 2020. [Online]. Available: https://mailarchive.ietf.org/arch/msg/cfrg/LKbwodpa5yXo6VuNDU66vt Aca8/

[15] A. Juels and R. L. Rivest, “Honeywords: Making password-cracking detectable,” in Proc. ACM CCS 2013, pp. 145–160.

[16] D. Wang, H. Cheng, P. Wang, J. Yan, and X. Huang, “A security analysis of honeywords,” in Proc. NDSS 2018, pp. 1–18.

[17] Akshima, D. Chang, A. Goel, S. Mishra, and S. K. Sanadhya, “Generation of secure and reliable honeywords, preventing false detection,” IEEE Trans. Depend. Secur. Comput., vol. 16, no. 5, pp. 757–769, 2019.

[18] K. C. Wang and M. K. Reiter, "Using amnesia to detect credential database breaches," in Proc. USENIX Secur, 2021, pp. 839-855.

[19] Passwordless Authentication Duo Security. Accessed: Aug. 15, 2021. [Online]. Available: https://duo.com/solutions/passwordless

[20] W. Li, H. Cheng, P. Wang, and K. Liang. "Practical threshold multi- factor authentication," IEEE Trans. Inf. Forensics Security, vol. 16.pp. 3573-3588, 2021.

[21] J. Zhang, H. Zhong, J. Cui, Y. Xu, and L. Liu, "SMAKA: Secure many-to-many authentication and key agreement scheme for vehicular networks." IEEE Trans. Inf. Forensics Security, vol. 16, pp. 1810-1824. 2021.

[22] J. Srinivas, A. K. Das, M. Wazid, and N. Kumar, "Anonymous lightweight chaotic map-based authenticated key agreement protocol for industrial Internet of Things." IEEE Trans. Depend. Secure Comput.. vol. 17, no. 6, pp. 1133-1146, Nov. 2020.

[23] J. Srinivas, A. K. Das, N. Kumar, and J. J. P. C. Rodrigues, "Cloud centric authentication for wearable healthcare monitoring system," IEEE Trans. Depend. Secure Comput., vol. 17, no. 5, pp. 942-956, Sep. 2020.

[24] J. Ma, W. Yang, M. Luo, and N. Li, "A study of probabilistic password models," in Proc. IEEE Symp. Secur. Privacy, May 2014, pp. 538-552.

[25] A. Juels and T. Ristenpart, "Honey encryption: Security beyond the brute-force bound," in Proc. EUROCRYPT Cham, Switzerland: Springer, 2014, pp. 293-310.

[26] Github CFRG/Draft-IRTF-CFRG-Opaque/the Opaque Asymmetric Pake Protocol. Accessed: Aug. 15, 2021. [Online]. Available: https://github.com/cfrg/draft-irtf-cfrg-opaque

[27] S. M. Bellovin and M. Merritt, "Encrypted key exchange: Password- based protocols secure against dictionary attacks," in Proc. IEEE Com- put. Soc. Symp. Res. Secur. Privacy, 1992, pp. 72-84

[28] S. Jarecki, A. Kiayias, H. Krawczyk, and J. Xu. "Highly-efficient and composable password-protected secret sharing (or: How to protect your bitcoin wallet online)," in Proc. IEEE Eur. Symp. Secur. Privacy (EuroS&P), Mar. 2016, pp. 276-291.

[29] H. Krawczyk, "HMQV: A high-performance secure Diffie-Hellman protocol," in Proc. CRYPTO, 2005, pp. 546-566

[30] T. Wu, "The secure remote password protocol," in Proc. NDSS, vol. 98. 1998. pp. 97-111.

[31] D. Taylor, T. Wu, N. Mavrogiannopoulos, and T. Perrin, Using the Secure Remote Password (SRP) Protocol for TLS Authentication, document RFC5054, 2007.

[32] H. Cheng. Z. Zheng, W. Li, P. Wang, and C.-H. Chu, "Probability model transforming encoders against encoding attacks," in Proc. USENIX Secur, 2019, pp. 1573-1590.