**Secure text transfer using Diffie-Hellman key Exchange Based on Cloud**

**A Project Work Synopsis**

*Submitted in the partial fulfillment for the award of the degree of*

**BACHELOR OF ENGINEERING**

**IN**

**COMPUTER SCIENCE WITH A SPECIALIZATION IN**

**ARTIFICIAL INTELLIGENCE AND MACHINE LEARNING**

**Submitted by:**

20BCS3566 Abhay Tomer

**Under the Supervision of:**

**Er. Namit Chawla**



**CHANDIGARH UNIVERSITY, GHARUAN, MOHALI - 140413,**

**PUNJAB**

**March, 2022**

# Abstract

This project proposes a method of secure text transfer using the Diffie-Hellman key exchange algorithm for authentication. The aim of this project is to ensure the confidentiality and integrity of transferred data while also preventing unauthorized access. The Diffie-Hellman algorithm is used to establish a shared secret key between two parties, which is then used to encrypt and decrypt the text messages. The project also incorporates digital signatures to provide an additional layer of security and ensure message authenticity. The implementation of this method involves the development of a client-server application using Python programming language. The project's success will be evaluated based on its ability to securely transfer text messages while maintaining confidentiality, integrity, and authenticity.

# Table of Contents

|  |  |
| --- | --- |
| Title Page | i |
| Abstract | ii |
| 1.    Introduction |  |
| 1.1  Problem Definition |  |
| 1.2 Project Overview |  |
| 1.3 Hardware Specification |  |
| 1.4 Software Specification |  |
| 2.    Literature Survey |  |
| 2.1 Existing System |  |
| 2.2 Proposed System |  |
| 2.3 Literature Review Summary |  |
| 3.    Problem Formulation |  |
| 4.    Research Objective |  |
| 5.    Methodologies |  |
| 6.    Experimental Setup |  |
| 7.    Conclusion |  |
| 8.    Tentative Plan for the proposed work |  |
| 9.    Reference |  |

# 1. INTRODUCTION

## 1.1 Problem Definition

Cloud computing has become a popular means of storing and accessing data for individuals and businesses alike. However, the security of data in the cloud is a significant concern due to the risk of unauthorized access by malicious actors. One way to address this issue is to use the Diffie-Hellman key exchange algorithm for secure data transfer.

The Diffie-Hellman algorithm is a cryptographic protocol used to establish a shared secret key between two parties over an insecure network. This key can then be used to encrypt and decrypt data, ensuring confidentiality, integrity, and authenticity of the transferred data. By using the Diffie-Hellman algorithm for data transfer in the cloud, sensitive information can be securely transmitted between the client and the cloud server.

One of the main advantages of using the Diffie-Hellman algorithm is that it provides a secure means of key exchange without the need for pre-shared keys or complex public key infrastructure (PKI). This makes it a suitable solution for secure data transfer in the cloud, where there may be multiple users accessing the same data.

## 1.2 Problem Overview

By creating the illusion that nothing out of the ordinary is happening, encryption may be used to hide sensitive or private information. The cypher text won't be understood as containing any hidden information by someone who reads it. Because our senses aren't trained to look for data files, encryption essentially exploits human perception. Users might send text messages that are encrypted and supply a key or password to lock the channel in order for the project to work.

## 1.3 Hardware Specification

* Working system
* Ram over 8gb | Processor over 1.1 Ghz | Storage over 256 gb
* Java compiler
* Web browser and working internet

## 1.4 Software Specification

* React framework
* Google Firebase
* Sass framework
* Web Dev Enviornment

# 2. LITERATURE SURVEY

## 2.1 Existing System

2.1.1 New Directions in Cryptography

There are two different modern developments in cryptography that are studied. The need for new kinds of cryptographic systems that reduce the need for secure key distribution methods and provide the equivalent of a written signature has arisen as a result of the expanding applications of teleprocessing. This essay offers solutions to these unresolved issues. It also examines how long-standing cryptographic problems are starting to be solved with the help of theories of computation and communication.

2.1.2 Oblivious Transfer based on Key Exchange

It has not been considered to be possible to perform an oblivious transfer using key-exchange protocols. (OT). In this paper, using the concept of unintentionally exchanging encryption keys, we propose a protocol for the exchange of secrets between parties, 1-out-of-2 OT, and coin flipping. Given the popularity of the Diffie-Hellman scheme, our protocol could be a helpful complement to existing techniques for implementing oblivious transfer as well as a foundation for more complex cryptographic systems.

2.1.3 Advanced Encryption using AES and Diffie Hellman Key Exchange

The project's goal is to examine, put into practice, and research the Diffie- Hellman key exchange protocol's uses, particularly for online key exchange. Make a website for an online company. Before sending confidential information from the client to the server, encrypt it, including customer credit information and SSNs. The concept is straightforward: after a customer accesses a website driven by our straightforward HTTP server, our web server responds with a java servlet. The Diffie- Hellman key exchange protocol will be used by the client and server to exchange data in order to create the shared key that will be used for subsequent communication. After the key exchange, the shared key will be used by the server and client to encrypt and decrypt confidential data, such as the customer’s SSN and credit card numbers.

2.1.4 The Simplest Protocol for Oblivious Transfer

The basic building block of cryptographic protocols is the oblivious transfer (OT). In this article, we present the Diffie-Hellman key-exchange protocol's modification, which results in the simplest and most effective 1-out-of-2 OT protocol to date. The protocol obtains UC security against active corruption in the random oracle model. We also discuss an elliptic curve-based version of the protocol and the various security measures we use to protect our software from active attacks. According to experimental findings, our protocol is at least an order of magnitude faster than earlier work (as a consequence of algorithmic and implementation optimizations).

2.1.5 Text Transfer Using Diffie-Hellman Key Exchange On Cloud

Cloud computing architectures are widely used in enterprises. It offers basic services, great network connectivity, flexibility, and more. However, the use of these features is difficult due to several security concerns. When someone views a text file, they don't get the impression that it contains hidden information. To compose the cipher text, the recipient needs a key. In this way, you can double-check that your private messages are being sent to hackers or crackers without outside influence. When the sender posts this text file, people will receive it without knowing what it is.

## 2.2 Proposed System

The project's goal is to discover a secure channel for sending and receiving text and other types of data. Using Diffie-Hellman and AES (Advanced Encryption Standard) encryption methods, the text message and files are encrypted and sent to the receiver in a safe manner. Before sending the channel and key information to the receiver in the cloud, the system encrypts it using AES.

Security dangers are, as we are all aware, growing daily. There is no assurance that exchanging information with someone else will be done securely and without the risk of hackers or cracking. At the end of this assignment, we will have learned about a technique that uses encryption to conceal the real key and safely transfer the desired information.

# 3. PROBLEM FORMULATION

The Diffie-Hellman algorithm is a public-key cryptographic protocol used to establish a shared secret key between two parties over an insecure network. It was invented by Whitfield Diffie and Martin Hellman in 1976 and is widely used in secure communication protocols, such as SSL/TLS and SSH.

The Diffie-Hellman key exchange is a method used by two parties to generate a shared secret key over an insecure network, without any prior communication or pre-shared secret keys. The process involves the following steps:

1. Each party generates a public-private key pair.
2. Each party sends their public key to the other party.
3. Each party calculates the shared secret key using their own private key and the other party's public key.

The formula used to calculate the shared secret key is:

**shared secret key = (other party's public key)^own private key mod prime number.**

Where:

**"other party's public key" is the public key of the other party.**

**"own private key" is the private key of the party calculating the shared secret key.**

**"prime number" is a large prime number agreed upon by both parties.**

The Diffie-Hellman key exchange algorithm relies on the fact that it is computationally infeasible to determine the private key from the public key. This ensures that the shared secret key generated by both parties is secure and cannot be intercepted by an attacker.

Once the shared secret key is generated, it can be used for symmetric encryption of data between the two parties. This allows for secure communication over an insecure network.

# 4. OBJECTIVES

Creating a secure channel for sending text and other files via the cloud is the project's primary goal. As several users can access the cloud platform simultaneously, it is crucial to create a secure tunnel for sending sensitive data. One of the finest ways to conceal data or information so that unauthorised people cannot access the files is through encryption. We may employ a pair of public and private keys that are only shared by the sender and the recipient to prevent anybody else from accessing the channel. The primary goals of this project may be summed up as the following:

* **Create a web app that can transfer text over the cloud**
* **Secure the channel by using cryptographic techniques.**
* **To access the cloud, use a public key and a private key.**
* **After authentication allow people to send text over cloud.**

# 5. METHODOLOGY

**Steps**

1. **Plan and Design**: Start with planning and designing the chat application. Define the purpose, scope, and requirements of the project. Create user personas and wireframes. Identify the features and functionalities you want to include in the application. Determine the layout, design, and color scheme of the application.
2. **Set up React**: Install React and its dependencies. Create a new React project using a tool like create-react-app. Set up the directory structure, create components, and define their functions. Implement the design using HTML, CSS, and JavaScript.
3. **Set up Firebase:** Create a Firebase account and set up a new project. Enable authentication, real-time database, and hosting. Obtain the Firebase configuration details required for the application.
4. **Connect React and Firebase:** Install Firebase SDK and configure it with the application. Set up Firebase authentication using email and password, Google, Facebook, or other providers. Create a real-time database to store and retrieve chat messages. Define the rules for accessing and modifying the database.
5. **Implement Real-time Messaging:** Implement the real-time messaging functionality using Firebase Realtime Database. Create a chat room or channel where users can join and send messages to each other. Implement features like sending text messages, images, emojis, and notifications. Handle errors and edge cases.
6. **Test and Deploy:** Test the application locally and in various environments. Fix bugs and improve performance. Deploy the application to Firebase hosting or another hosting provider. Set up custom domains, SSL certificates, and other configurations.
7. **Maintain and Update:** Maintain the application by monitoring its performance, security, and user feedback. Update the application with new features, bug fixes, and improvements based on user needs and market trends.

## 8. TENTATIVE PLAN FOR THE PROPOSED WORK

**Check Point - 1**

Literature Phase Deadline: Feb 20-27, 2023

1. Definition of Scope
2. Identifying the problem statement
3. Submit introduction for their research article

**Check Point – 2**

Design Phase Deadline: Mar 20-27, 2023

1. Preliminary Design
2. Methodology Used
3. Analysis of features (What update they have to include in the project)
4. Submit literature survey and proposed system for their research article

**Check Point – 3**

Analysis Phase Deadline: Apr 17-22, 2023

1. Result Analysis
2. Submit results, abstract and conclusion for their research article

**Check Point – 4**

Conclusion Phase Deadline: May 01-06, 2023

1. Submission of research article in IEEE format

**Check Point – 5**

Evaluation Phase – II Deadline: May 15-17, 2023

1. Supervisor Evaluation (Collect research Articles, Project Report, PPT, implementation (Github Link)

**Check Point – 6**

Evaluation Phase – II Final End Term Evaluation

## REFERENCES

1. chromeextension://efaidnbmnnnibpcajpcglclefindmkaj/https://www.irjet.net/archives/V9/i4/IRJET-V9I4232.pdf
2. https://en.wikipedia.org/wiki/Diffie%E2%80%93Hellman\_key\_exchange
3. chrome-extension://efaidnbmnnnibpcajpcglclefindmkaj/https://ijics.com/gallery/58-may-1148.pdf
4. https://portal.aws.amazon.com/billing/signup?nc2=h\_ct&src=header\_signup&redirect\_url=https%3A%2F%2Faws.amazon.com%2Fregistration-confirmation#/account
5. <https://www.chiark.greenend.org.uk/~sgtatham/putty/latest.html>
6. <https://en.wikipedia.org/wiki/Amazon_Web_Services>
7. <https://aws.amazon.com/getting-started/cloud-essentials/?ref=gs&id=m1>
8. <https://intellipaat.com/blog/aws-applications-and-use-cases/>