**CLOUD APPLICATION DEVELOPMENT**

**WEEK 4**

**PROJECT REPORT**

**Name:** Kartik Sengar

**SAPID:** 500083548

**Roll No.:** R214220601

**Batch:** 5, CCVT(Non-Honours)

1. **Problem Statement**

The problem statement chosen for this project is to create and ensure Secure Text Transfer using Diffie Hellman Key Exchange, based on Cloud.

It refers to the need to develop a secure method for transferring text data over the cloud. The focus of the solution is to use the Diffie-Hellman key exchange algorithm to ensure secure communication between two parties.

The Diffie-Hellman algorithm is a cryptographic method for securely exchanging keys over an unsecured channel. It allows two parties to establish a shared secret key that can be used to encrypt and decrypt messages, ensuring that only the intended recipient can read the message.

The solution aims to use the cloud to facilitate this secure communication between the two parties, which may be located in different geographical locations. The challenge is to ensure that the text data is protected from unauthorized access during transmission over the cloud.

Therefore, the problem statement requires designing and implementing a secure text transfer protocol that uses the Diffie-Hellman key exchange algorithm to establish a shared secret key between the sender and the recipient. This protocol should be implemented on the cloud infrastructure and must ensure the confidentiality, integrity, and authenticity of the text data during transfer.

1. **Pert Chart**

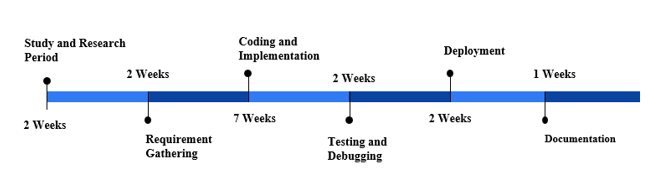
****

Fig 1

1. **Flowchart**

****

Fig 2

1. **Design**
2. Define Requirements: The first step is to define the requirements for the secure text transfer protocol. This includes identifying the key features that the protocol should have, such as confidentiality, integrity, and authenticity.
3. Choose Cloud Infrastructure: The next step is to select a suitable cloud infrastructure to implement the protocol. The cloud infrastructure should be secure and reliable, and capable of handling the required data transfer volume.
4. Develop Protocol: The third step is to develop the protocol using the Diffie-Hellman key exchange algorithm. The protocol should ensure secure communication between two parties and establish a shared secret key.
5. Implement Protocol: The fourth step is to implement the protocol on the chosen cloud infrastructure. This involves setting up the necessary infrastructure and configuring the protocol to run on it.
6. Test and Verify: The fifth step is to test and verify the protocol to ensure that it meets the defined requirements. This involves carrying out various testing scenarios to check for security vulnerabilities and other potential issues.
7. Deploy Protocol: The final step is to deploy the protocol for use. This involves making it available to users and ensuring that they are trained on how to use it properly.
8. **Implementation**

Implementation Steps for deploying on Azure:

1. Analyze the requirements of the secure text transfer protocol to determine if Azure is a suitable cloud infrastructure.
2. Configure the Azure Virtual Network (VNet) to enable secure communication between different virtual machines.
3. Create an Azure Key Vault to securely store and manage the cryptographic keys used by the Diffie-Hellman key exchange algorithm.
4. Use Azure Active Directory (AD) for authentication and authorization of users accessing the secure text transfer protocol.
5. Deploy the protocol on Azure VMs, using Azure Load Balancer to distribute traffic across multiple VMs for high availability and scalability.
6. Use Azure Application Gateway to provide additional security features such as web application firewall (WAF) and SSL/TLS offloading.
7. Configure Azure Monitor to track and analyze protocol performance and security events.
8. Use Azure Security Center to monitor and identify potential security threats and vulnerabilities.
9. Use Azure Backup to protect data in case of accidental deletion or corruption.
10. Ensure compliance with regulatory requirements by using Azure compliance offerings, such as Azure Security and Compliance Blueprint for Payment Card Industry (PCI) DSS.

Implementation of Multithreading:

* The cloud-based application can utilize multithreading techniques to concurrently process expenses and enhance performance.
* This can be achieved by creating multiple threads, with each thread responsible for processing a batch of expenses in parallel.
* To ensure proper synchronization and avoid race conditions, a shared data structure can be implemented as a class that is accessed and updated by multiple threads concurrently.
* The shared resources can be protected by utilizing synchronization mechanisms such as locks, semaphores, or other concurrency control techniques.
* The use of multithreading can enhance scalability and performance by enabling the application to process expenses in a distributed and parallelized manner in the cloud.

1. **Algorithm**
2. Establish a secure communication channel between the two parties over the internet.
3. Generate a public and private key pair for each party.
4. Implement the Diffie-Hellman key exchange algorithm to exchange public keys and generate a shared secret key.
5. Encrypt the text message using the shared secret key and a symmetric encryption algorithm such as AES.
6. Transmit the encrypted message to the receiver.
7. Upon receiving the encrypted message, the receiver decrypts it using the shared secret key and the same encryption algorithm.
8. Securely delete the shared secret key from both parties to ensure data confidentiality.

To deploy this algorithm on the cloud, the following additional steps can be taken:

1. Use cloud infrastructure such as Azure to establish a secure communication channel between the two parties.
2. Use Azure Key Vault to securely store and manage the cryptographic keys used in the Diffie-Hellman key exchange algorithm.
3. Deploy the protocol on Azure VMs, using Azure Load Balancer to distribute traffic across multiple VMs for high availability and scalability.
4. Use Azure Application Gateway to provide additional security features such as web application firewall (WAF) and SSL/TLS offloading.
5. Use Azure Monitor to track and analyze protocol performance and security events.
6. Use Azure Security Center to monitor and identify potential security threats and vulnerabilities.

**7. GitHub Link**

This is my GitHub Link for the same report: <https://github.com/kartiks123/CAD_Week-4>