Cloud Computing: Secure Cloud Messaging For On-Prem Application

Binoy Kurikaparambil Revi

Independent Researcher, USA
binoyrevi@live.com

Abstract
Computer applications can generate data that can be analyzed to enhance the application itself or generate the data as an output of the application functionality that needs to be securely stored. Local storage tends to be insufficient and insecure to a great extent. Cloud storage is an option to securely store data generated by the application with advanced encryption and secure access using very reliable methods. With the advancement of analytics and big data applications, data collection or uploading from the desktop or on-prem server application has become a system requirement with user permissions nowadays. The cloud messaging software components must be designed to secure the application's existing performance and transmit the message seamlessly with appropriate encryption.

Keywords: **Encryption, Cloud Computing, Messaging Services, data upload**

1. Introduction
Implementing the Secure Cloud Messaging module for applications running on a desktop or on-prem server significantly enhances the application storage capabilities, as it utilizes the Cloud Service Provider's reliable and secure storage services. The application is still detached from the Cloud and must use a Cloud Messaging software component that securely transmits the messages to the cloud. Network libraries provide a mechanism for the backend application to implement and use the REST API[2] services to connect with the cloud and securely transmit the data to the cloud. For secure data transfer, the messages must be sent across the network to the Cloud Server with authentication and encryption.

2. Secure Messaging Module Design
There are two main aspects when designing a secure Cloud Messaging module for the backend system. Network Side Design and Application Interface Design.
The Network-Side Design defines how the application will make a secure connection with the Cloud Server and how the data packet will be encrypted and transmitted to the Cloud Server. An authentication protocol is implemented for the application to make a secure connection with the Cloud Server for data exchange. An authentication protocol that can be used to communicate with the Cloud Server is given in Figure 1. This protocol executes all the communication between the Desktop or On-Prem Application and the Cloud Server using REST API services. The Application uses HTTP Post Request to send the data to the Cloud Server. As JSON is the de facto standard used by modern web servers and web

applications, JSON format is used as the data model to pack the data. All the JSON packages going from the Application to the Cloud Server are Encrypted.

Authentication and Messaging Protocol

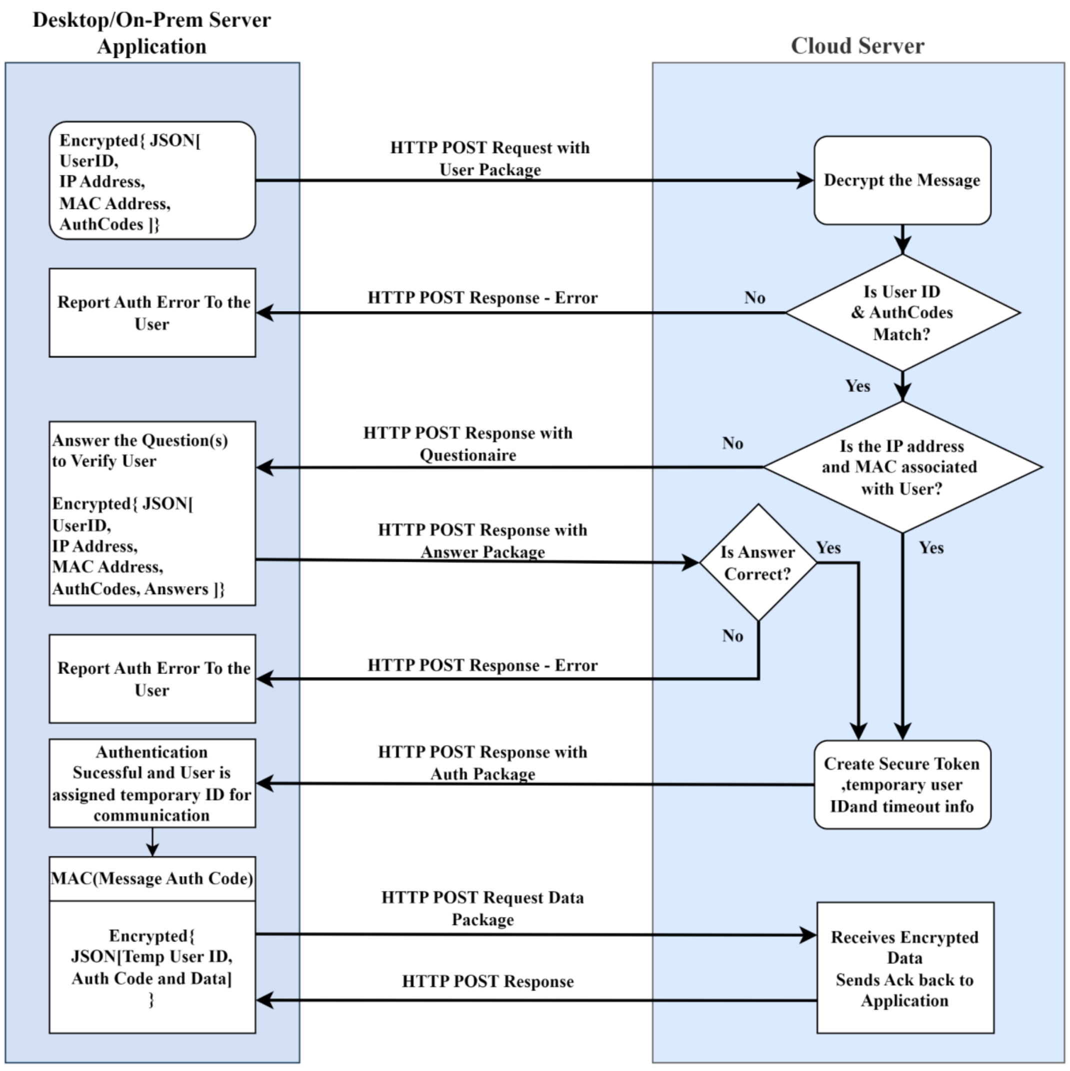
Desktop/On-Prem Server Application
Cloud Server
Encrypted{ JSON[UserID, IP Address, MAC Address, AuthCodes]}
HTTP POST Request with User Package
Decrypt the Message
Is User ID & AuthCodes Match?
No
HTTP POST Response - Error
Report Auth Error To the User
Yes
Is the IP address and MAC associated with User?
No
HTTP POST Response with Questionaire
Answer the Question(s) to Verify User
Encrypted{ JSON[UserID, IP Address, MAC Address, AuthCodes, Answers]}
HTTP POST Response with Answer Package
Is Answer Correct?
Yes
Yes
No
HTTP POST Response - Error
Report Auth Error To the User
Create Secure Token ,temporary user IDand timeout info
HTTP POST Response with Auth Package
Authentication Sucessful and User is assigned temporary ID for communication
MAC(Message Auth Code)
Encrypted{ JSON[Temp User ID, Auth Code and Data] }
HTTP POST Request Data Package
Receives Encrypted Data Sends Ack back to Application
HTTP POST Response

Figure 1: Cloud Messaging Authentication Protocol

The Application is the first to initiate communication by sending the User ID, IP Address, MAC address, and Authentication Codes to the Cloud Server in an Encrypted JSON Package. The user has no control to set any of this. The application reads these data from the system based on the login information and system parameters. Once the Cloud Server receives the encrypted User Package from the application, the Cloud Server decrypts and proceeds with two validation steps. In the first step, the application checks if the User ID and Authentication are valid and associated. If the first step fails, an error response is sent back to the application. If that first step succeeds, the Cloud Server moves to the 2nd validation step. In this step, the Cloud Server checks if the IP address and the MAC address are associated with the user. This step ensures the user is logged in from a known system or server. A questionnaire response will be sent to the application if the IP address or MAC address is not associated with the user. The application is expected to get the answer(s) for the questionnaire and an updated user package with the answer(s) sent to the Cloud Server. The Cloud Server validates the updated user package with the answer(s). If the 2nd validation step fails, an error response is sent to the application. Upon successfully validating the 2nd Step, the Cloud Server creates an Auth Package containing a Secure Token, Temporary User ID, and Timeout info. This auth package is sent to the application as a response. For future data messaging until the package's specified timeout, the application must include an authentication token and temporary user ID(In place of actual user ID) along with the data every time it sends an encrypted package to the Cloud Server.

Data Encryption Strategy

As all the communication from the application to the Cloud Server is encrypted, asymmetric encryption is preferred.[1]. The application just needs to have the public key of the Cloud Server. The private key is never shared by the Cloud Server and is secured using cloud infrastructure like a key vault. So, the application always encrypts the data using the public key and sends it to the Cloud Server. Note that the application never receives an encrypted message as a response from Cloud Server. This ensures the application security as the public key can be exposed on the application side and doesn't cause much harm.

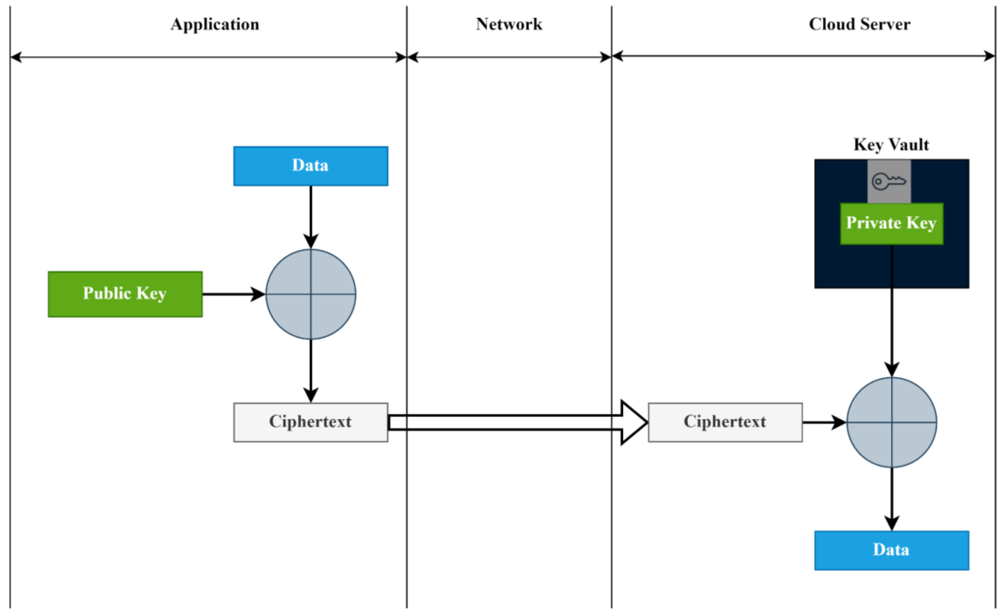

Figure 2: Encryption Strategy for Cloud Messaging

Figure 2 is a simple sketch showing how asymmetric encryption works on the application side, and decryption works on the Cloud Server side.

Cloud Messaging Module Integration

The Clould Messaging Module can be integrated into the application as a thread that processes the application's Message Queue using the FIFO strategy. The module checks the message queue and continues processing messages from it. After successfully completing the authentication steps, the message is packed with the Authentication Code and Temporary User ID using the JSON data model.

The package is encrypted and sent to the Cloud Server. Network Libraries are used to implement the RESTful Service at the application end.

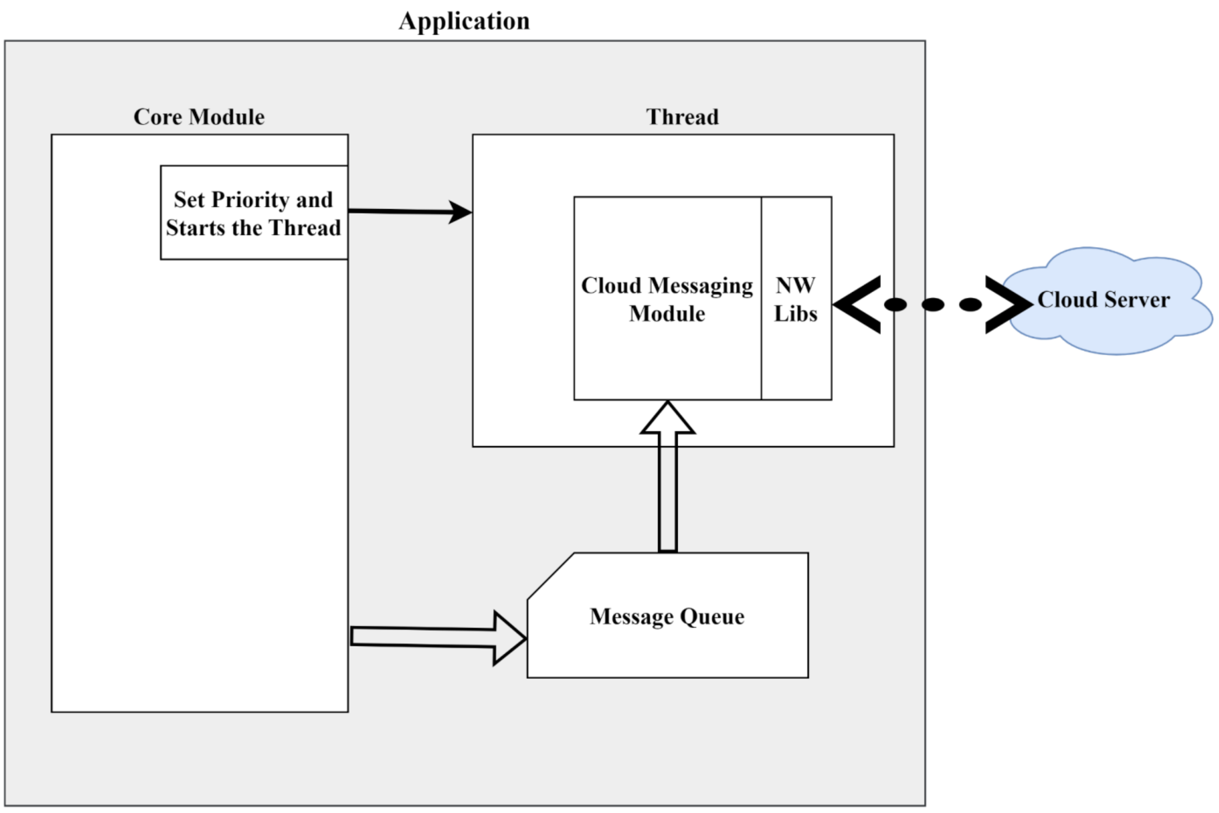

Figure 3: High-level Application Design

Figure 3 provides a typical design for implementing and integrating the Cloud Messaging module with the application. Running the module in the thread[3] allows running the thread in the preferred priority mode. The thread shares memory resources with the main program, so messages can be easily handled using the message queue. This also ensures that the Cloud Messaging module does not block the main program execution.

References

1. Q. Zhang, "An Overview and Analysis of Hybrid Encryption: The Combination of Symmetric Encryption and Asymmetric Encryption," 2021 2nd International Conference on Computing and Data Science (CDS), Stanford, CA, USA, 2021, pp. 616-622, doi: 10.1109/CDS52072.2021.00111.
2. Neumann, N. Laranjeiro and J. Bernardino, "An Analysis of Public REST Web Service APIs," in IEEE Transactions on Services Computing, vol. 14, no. 4, pp. 957-970, 1 July-Aug. 2021, doi: 10.1109/TSC.2018.2847344.
3. Ian Foster, Carl Kesselman, Steven Tuecke, The Nexus Approach to Integrating Multithreading and Communication, Journal of Parallel and Distributed Computing, Volume 37, Issue 1, 1996, Pages 70-82, ISSN 0743-7315, https://doi.org/10.1006/jpdc.1996.0108.