

Secure Communications and collaboration Design, Programming and analysis

Module: Computer System Security

Group Members:

Ananya Sahu (10627321)

Sumit Gupta (10634262)

Baran Azak (10636432)

Contents

[Introduction 4](#_Toc141345745)

[Business Requirements 5](#_Toc141345746)

[Functional Requirements 5](#_Toc141345747)

[Secure Messaging: 5](#_Toc141345748)

[View Once Only 5](#_Toc141345749)

[User Authentication via Facebook: 5](#_Toc141345750)

[Message Composing and Sending 5](#_Toc141345751)

[Non-functional Requirements 6](#_Toc141345752)

[Security 6](#_Toc141345753)

[Reliability 6](#_Toc141345754)

[User-Friendly Interface 6](#_Toc141345755)

[Privacy 6](#_Toc141345756)

[Compliance 6](#_Toc141345757)

[Security Requirements 7](#_Toc141345758)

[Data Encryption: 7](#_Toc141345759)

[Secure Authentication: 7](#_Toc141345760)

[Secure Data Transmission: 7](#_Toc141345761)

[Message Expiry and Deletion: 7](#_Toc141345762)

[Session Management: 7](#_Toc141345763)

[Secure Third-Party Integrations: 7](#_Toc141345764)

[Assumptions 8](#_Toc141345765)

[Cryptography 9](#_Toc141345766)

[Key Generation Process 10](#_Toc141345767)

[Encryption/Decryption 12](#_Toc141345768)

[Vulnerability Report for Flash Messenger System 13](#_Toc141345769)

[Assessment Scope 13](#_Toc141345770)

[Vulnerability Findings 13](#_Toc141345771)

[Recommendations 14](#_Toc141345772)

[Conclusion For Vulnerability Report 14](#_Toc141345773)

[Conclusion 15](#_Toc141345774)

[Bibliography 16](#_Toc141345775)

# Introduction

In a world where digital communication has been more prone to data breaches, unauthorised access, and undesired retention of sensitive information are escalating in a society where digital communication has grown pervasive. Flash Messenger tackles these issues front on by delivering a secure messaging solution that employs cutting-edge encryption techniques, giving you peace of mind that your communications are safe from prying eyes.

Flash Messenger is cutting-edge chat software that puts your privacy and security first. When required, Flash Messenger will ensure that communication is smooth, private, and ephemeral. This method allows you to send and receive encrypted messages that can only be read once before disappearing, guaranteeing that your personal chats stay completely secret.

The platform was created with simplicity and use in mind, allowing anybody to effortlessly compose and send messages with just a few clicks. Furthermore, we recognise that seamless authentication is critical for a hassle-free experience. As a result, we provide secure login through Facebook, employing their sophisticated authentication infrastructure to provide a smooth and trustworthy sign-in procedure.

# Business Requirements

Flash Messenger may meet your demands whether you are an individual searching for a safe means to transmit sensitive information or a corporation wishing to retain client confidentiality. Our dedication to privacy, data security, and industry best practices distinguishes us as a global leader in private messaging. The aim of this messenger is to rigorously adhere to data protection legislation, and we are committed to preserving your personal information. You can be certain that your chats on the messenger will stay purely between you and your intended recipients because of our commitment to user-centric design and security.

## Functional Requirements

### Secure Messaging:

* + The system should allow users to securely transmit and receive encrypted communications.
  + To maintain secrecy, messages should be encrypted during transmission and storage.

### View Once Only

* + Messages should be constructed such that the receiver may only read them once.
  + After the recipient views the message, it should instantly evaporate and become unreachable.

### User Authentication via Facebook:

* + Users should be able to log in to the system securely using their Facebook credentials.
  + The system should support OAuth or comparable authentication protocols.

### Message Composing and Sending

* + Users must be able to create new messages.
  + Users should be able manually input their email addresses.
  + User should be able to see the messages received from other users.

## Non-functional Requirements

### Security

* + The system should utilise strong encryption methods to protect communications and user data.
  + It should adhere to best practices in the industry to prevent unauthorised access to messages and user information.

### Reliability

* + The system should be extremely dependable and available, with as little downtime and data loss as possible.

### User-Friendly Interface

* + The user interface should be simple and straightforward to use.
  + Users should be able to properly comprehend the message view-once capabilities.

### Privacy

* + The system must follow privacy standards and not keep any superfluous user data.
  + It should properly convey to consumers its data usage policy.

### Compliance

* + The system must adhere to applicable data security and privacy laws and regulations.
  + It must follow Facebook authentication principles and criteria.

## Security Requirements

### Data Encryption:

* + To preserve confidentiality and prevent unauthorised access to the information, all communications and sensitive data transferred and kept inside the system must be encrypted using powerful cryptographic methods (e.g., AES, RSA). To improve security, encryption keys should be securely controlled and cycled on a regular basis.
  + End-to-end encryption should be used in the system to ensure that data is encrypted on the sender's device and decrypted only on the recipient's device, rendering it inaccessible during transmission and storage.
  + Encryption should also be used to protect data stored in databases or other forms of persistent storage, preventing unauthorised access even if the underlying storage medium is compromised.

### Secure Authentication:

* + Integrate with third-party authentication providers like Facebook using the most recent secure protocols (e.g., OAuth 2.0), guaranteeing secure communication during the authentication process.

### Secure Data Transmission:

* + The application should enforce HTTP Strict Transport Security (HSTS) to ensure that all communication with the server takes place over HTTPS, hence lowering the danger of protocol downgrade attacks.

### Message Expiry and Deletion:

* + Create a system that deletes communications from both sender and recipient devices after they have been seen once, in accordance with the "view once and disappear" policy. This guarantees that critical information is not kept on the devices unnecessarily.
  + Use secure deletion mechanisms to ensure that deleted communications cannot be recovered, avoiding potential data breaches if a device is lost or stolen.

### Session Management:

* + Use secure and HTTP-only cookies for managing user sessions, reducing the risk of session hijacking attacks.
  + Implement session timeouts, automatically logging users out after a period of inactivity, to prevent unauthorized access in case a user leaves their session unattended.
  + Enable session management on the server-side to revoke and invalidate session tokens in case of a logout mechanism.

### Secure Third-Party Integrations:

* + Connecting with third-party services or APIs, make sure you have adequate security procedures in place to guard against any security risks that come with those integrations.

# Assumptions

The application security implementation is based on the following assumptions.

**Third-party Authentication**

The user needs the user to sign in using Facebook. It is assumed that the person signing in via Facebook has already been verified by the Facebook database.

The Facebook JavaScript SDK connector is used in the application login process. It was referred to by a URL. It is presumed that the SDK is genuine and does not include any harmful code.

**User Device**

Local storage and session storage are used to store user information for the duration of the session. It must be anticipated that the user will keep the gadget safe.

# Cryptography

RSA

The RSA algorithm is a commonly used asymmetric cryptographic technique that provides secure communication, digital signatures, and data encryption by encrypting and decrypting data with a pair of public and private keys.

Web Cryptography API

online Cryptography API- It is a JavaScript API that offers online applications with a set of cryptographic methods and capabilities for performing different cryptographic activities safely.

Usage

Developers may use the Web Cryptography API to create safe random numbers, conduct encryption and decryption, sign and validate data, and manage keys. It is intended for usage in web browsers and offers a standard interface for cryptographic operations, making it easier for developers to include safe encryption and other cryptographic capabilities in their online applications.

Support for cryptography API

The World Wide Web Consortium (W3C) created and maintains the Web Cryptography API as a web standard. The World Wide Online Consortium (W3C) is a community-driven organisation that develops online standards to promote interoperability and uniformity across various web browsers and systems.

The involvement of the W3C in standardising the Web Cryptography API, as well as key browser makers and security experts, instils faith in the API's architecture, security, and dependability. When developers use the online Cryptography API in online applications, they can be confident that cryptographic operations are handled in a secure and standardised manner.

# Key Generation Process

code

return window.crypto.subtle.generateKey(

{

name: "RSA-OAEP",

modulusLength: 2048,

publicExponent: new Uint8Array([0x01, 0x00, 0x01]),

hash: { name: "SHA-256" },

},

true,

["encrypt", "decrypt"]

);

Description

**name: "RSA-OAEP"**

"RSA-OAEP" specifies the name of the cryptographic algorithms.The term OAEP (Optimal Asymmetric Encryption Padding) refers to the inclusion of extra padding with real plain text before encryption.

As a result, encrypting the same text many times yields distinct encryption messages.

**modulusLength: 2048**

Sets the size of the private and public keys to 2048 bits.

The most common key sizes are 1024, 2048, 3072, and 4096 bits.The larger the key, the more secure it is, but it necessitates more computer power and memory.

**PublicExponent: new Uint8Array([0x01, 0x00, 0x01]):**

0x01 = 00000001 (8 bits)

0x00 = 00000000 (8 bits)

0x01 = 00000001 (8 bits)

e = [0x01, 0x00, 0x01] = 00000001 00000000 00000001

e = [0x01, 0x00, 0x01] = 000000010000000000000001

e = [0x01, 0x00, 0x01] = 65537

**name: "SHA-256" hash:**

It basically acts as a signature, ensuring that the keys produced are compliant with the algorithm used with the provided setup.

**true:**

Simply said, the key may be retrieved and exported.

**["encrypt", "decrypt"]:**

These are used to specify the permissions that these keys have to encrypt and decrypt.

public key = (n, e) = (product of two large prime number , Exponent) =

e.g (61 \* 53 , 65537) = (3233 , 65537)

(3233 , 65537) converted to ASCII using Base64 encoding

optionally can be convert to JWK(json web key format)

**Public key example:**

{"kty":"RSA","e":"AQAB","n":"wl4WFWEiUxEDDiIwy8xz2VsVHBpo1ezmtCuEFQ2W0krMHAEPKTxUYjxSHgFF2\_drsWb5FwqEdP-q75ITrAQU8DIFbZXwC1V53hMnBN62-itWf8Dw9wdfBXwFlX4FC-fDG0YIOjsremNoaNhD18Sn1ez\_E310cUBRZvT8SooWcw\_wOJ4MfgNtnMWiQS9PMZ0r2a676tjHS1Tnj6EguRZQ9oh4Zs\_ABWVUXlYra0\_5BsQ5437cOzsbKsY-Vmne6Tu0-Lb4aYtrfcZV7ptPhUpepspyWmoO6dClX6Xo1o4YaC7VBuu3L4DYfLS\_lxwn\_GDkUqjXoMLYVn1iisPHyzGjtw"}

**Private key example:**

{"alg":"RSA-OAEP-256","d":"Ouf03Wng0zf2qxPlof5GpfsavLesShlQGg6OKytBKF6O2XXguGiWUmdho8JS\_8VOmCdTGo6Pcj4B-jVVTO63wcwHJR0p8OVo18sKKkmtPshfoLxCyzsqqyjZMGNO-Dq3QM7Ks4s3NBJolE3m0b\_B9e2rqKZ-4cvSX9ejlpufDiOgxd775oEa-xPGLRkHq3Zs6i3HhTbEovKXhpuTCs7HZQ3MeOTIAk-Fc2S2oFcdvFc1kjI9NCOAdpWBFJyP-D0pAOK55LVz\_s62u8WwM40Prppxz-P3aGKgU2l-WVOm2sD3QA2I1KqRwRRiklvaWir1uussfCm-vdvibfraD4PpQQ","dp":"O\_Fui0Kw5TV8qsGG9fZA2zqKnCvhNh\_qAl4aA6UPTjxqrT026ZFi9rC6aIKfNwGO2Gi-dfMZHz3qtYFaZXfeNkQw00WVHs0gW7olgXDRk9YTYFzDu5bEGgDsuxtmRb6IODBw83q0rDs4rLFNMMydztaJr7smAAE0Ys2Farh457E","dq":"DkIOFid1rUgwa9z6zhaArRvXkJPj-NYCHMeoHXEojpeT0qwL9ugCOp5xpqS-IQ8Id-1Aue1AFKOE0UTHR2lZQ5C6PHdJq1OPY48yXbqmwCJGIdwfJvhUSnUHS3L8nth4F3PqMnopucZuIxkkSErvAIr4SqL55e7PajsXNnd0XyE","e":"AQAB","ext":true,"key\_ops":["decrypt"],"kty":"RSA","n":"wl4WFWEiUxEDDiIwy8xz2VsVHBpo1ezmtCuEFQ2W0krMHAEPKTxUYjxSHgFF2\_drsWb5FwqEdP-q75ITrAQU8DIFbZXwC1V53hMnBN62-itWf8Dw9wdfBXwFlX4FC-fDG0YIOjsremNoaNhD18Sn1ez\_E310cUBRZvT8SooWcw\_wOJ4MfgNtnMWiQS9PMZ0r2a676tjHS1Tnj6EguRZQ9oh4Zs\_ABWVUXlYra0\_5BsQ5437cOzsbKsY-Vmne6Tu0-Lb4aYtrfcZV7ptPhUpepspyWmoO6dClX6Xo1o4YaC7VBuu3L4DYfLS\_lxwn\_GDkUqjXoMLYVn1iisPHyzGjtw","p":"8n\_ZhCDO8pLt7QItH6\_1Jrz2n-Wp0Dsqo7cgAwT6HEAb9wXBJm-zQ7FiX8d8bDo-3JiSHt79YdVtaDvr4XE0Lo942UpXTIT69OomG-mLg7lRpHltfC8F9DIrLD-8w0SdHpqhHud6Gsyf62zAMPP31lmj0AlyFJgGoxyTB62hsYc","q":"zTBAUijqHjxbhRTzBK7eljvZkrBRqf2xO1luWrBmhVkRGmnUxL-kZ-wAvFcQLbYbmVZE\_C0VlH7mv3F4pYe9RM5Lnn9jpXOd6ajdFjSkwIkQwmdEMXB\_4CT5nlC6SOo91NPohkKRtDtbfYrQhfiKQf55EELQC0heDIIMdGaxyFE","qi":"xGYtg1tQD56BpMkHDMGKRBRfW6\_QIn-NOSVjufcPNyz\_uBVsSzSmcmqmgokSWX1cAZSHzlgFAjvEMCx8ZELkDD4aJpD\_M9ezOfoHpSIikzDxJkUPMgOc0aksGmXf-8xxU5rS6ilbadLCzVBEW5sCpIXW651pl\_B\_9QUVckBlBoc"}

# Encryption/Decryption

Encryption: Converting to cipher text using this public key

c = m\_numerical^e (mod n)

c= cipher text

m\_numerical = numerical representation of plain text

e = public exponent provided while configuration

n= product of two large prime numbers

await window.crypto.subtle.encrypt(algorith name, publickey, encodedData)

encodedData: Converting data to Binary format

Then binary format can be converted to ASCII.

so basically data --> binary --> base64 .ie ASCII representation(Cipher text)

Decryption: Converting to plain text using the private key

m\_numerical = c^d (mod n)

m\_numerical = numerical representation of plain text

c= cipher text

d= private key exponent generated by using two large numbers provided

n = product of two large prime numbers

await window.crypto.subtle.decrypt(algorith name, private key, encodedEncryptedData)

encodedEncryptedData: Converting ASCII(encrypted text) to binary using private key

decoding: Converting binary data to human-readable text

cipher text--> binary --> Readable text

# Vulnerability Report for Flash Messenger System

This vulnerability report's goal is to discover potential security flaws and vulnerabilities in the Flash Messenger system. The assessment was carried out to guarantee that user data and communications are sufficiently safeguarded against potential threats.

## Assessment Scope

The vulnerability assessment covered various aspects of the Flash Messenger system, including but not limited to:

* + Authentication mechanisms
  + Data encryption and transmission
  + User session management
  + Message expiry and deletion
  + Third-party integrations

## Vulnerability Findings

**Insecure Cryptography:**

The implementation of cryptography in the system revealed possible flaws in key generation. For RSA keys, the system was discovered to employ a comparatively short modulus length of 2048 bits. To improve encryption strength, consider raising the modulus length to 3072 bits or higher.

**Lack of Rate Limiting:**

The system lacks rate-limiting features for login attempts, rendering it vulnerable to brute-force assaults. Use rate limitation to limit the number of login attempts within a given time range.

**Lack of Input Validation:**

During message composition and transmission, the system lacks sufficient input validation. This vulnerability may allow for injection attacks. To avoid code injection, utilise stringent input validation and sanitise user input.

**Weak Session Management:**

Because the system does not enforce secure session processing, session hijacking is possible. To avoid unauthorised access to user sessions, utilise secure session management strategies such as utilising secure and HttpOnly cookies.

**Insufficient Third-Party Security Assessment:**

The system interfaces with third-party services or APIs without doing a thorough security audit on such integrations. To detect and minimise any security threats, do a complete security review of all third-party integrations.

## Recommendations

The following suggestions are made based on the vulnerabilities discovered during the assessment

1. To improve encryption strength, increase the RSA modulus length to 3072 bits or greater.
2. Use rate restriction to limit the number of login attempts and avoid brute-force assaults.
3. To avoid code injection attacks, perform stringent input validation and sanitise user input.
4. Use secure session management strategies such as secure and HttpOnly cookies to enforce secure session management.
5. Improve the "view once only" message deletion functionality so that message views are verified before deletion.
6. To guarantee security compliance, do a complete security evaluation of all third-party integrations.

## Conclusion For Vulnerability Report

The vulnerability study revealed various issues with the Flash Messenger system. By resolving the aforementioned vulnerabilities and applying the recommended procedures, the system's security posture may be greatly improved, allowing it to better safeguard user data and communications.

To maintain a secure messaging platform and protect user privacy and information, it is critical to undertake frequent security audits and remain up to speed with the newest security best practises.

# Conclusion

Flash Messenger provides a cutting-edge and secure messaging system that prioritises user privacy and data protection. The platform's cutting-edge encryption mechanisms ensure that all communications and sensitive data are secured during transport and storage, ensuring strong secrecy.

The implementation of Facebook or equivalent industry-standard protocols for secure authentication ensures that consumers have a smooth and trustworthy sign-in experience. Flash Messenger safeguards against eavesdropping and man-in-the-middle attacks by using HTTPS for secure data transport, further boosting communication security.

Flash Messenger's "view once only" functionality is one of its distinguishing characteristics. Messages are created in such a manner that the receiver may only view them once before they disappear, ensuring that critical information is not held needlessly and enhancing security, privacy and confidentiality.

The platform also places a premium on dependable session management, which ensures that user sessions are safe and terminate after a reasonable amount of inactivity, lowering the danger of unauthorised access. Third-party integrations are also taken into account, with safe procedures put in place to defend against any security concerns connected with such interfaces. Non-functional criteria, such as a user-friendly interface and adherence to privacy standards, improve the user experience and maintain data security.

Flash Messenger's security needs are well-defined, with strong data encryption utilising RSA and the Web Cryptography API to ensure that communications and user data stay safe. Secure authentication, data transfer, message expiration, and deletion protect user privacy and prevent unauthorised access to sensitive information.

Finally, Flash Messenger is a dependable, secure, and user-friendly messaging platform that prioritises user privacy and data protection while offering people and organisations a smooth and secure communication experience.

GitHub Link: <https://github.com/AnanyaSahu/Computer-SystemSecurity>

Application Link: <https://ananya-python.uksouth.cloudapp.azure.com:8080/>

# Bibliography

*Get Started - Graph API - Documentation* (no date) *Meta for Developers*. Available at: <https://developers.facebook.com/docs/graph-api/get-started/> (Accessed: 27 July 2023).

*Web - Facebook Login - Documentation* (no date) *Meta for Developers*. Available at: <https://developers.facebook.com/docs/facebook-login/web/> (Accessed: 27 July 2023).

*Web Crypto API RSA-OAEP Generate Keys and Perform Encryption/Decryption* (no date). Available at: <https://8gwifi.org/docs/window-crypto-rsaoaep.jsp> (Accessed: 27 July 2023).