

Project Report: SecureMessenger - A Secure Messaging Application

Author: Tejaswi Velaga

UIN: 663404952

1. Introduction

SecureMessenger is a secure, web-based messaging platform built to simulate Telegram's dual-chat architecture. It supports two communication modes: Secret Chats and Cloud Chats. Each mode is built to reflect different security philosophies — one prioritizing end-to-end privacy, and the other offering server-side synchronization for usability. This report explains the design, implementation, and cryptographic principles used in each mode in a clear and beginner-friendly manner.

Key Features

SecureMessenger provides a flexible messaging experience through two primary communication modes: Secret Chats and Cloud Chats. Secret Chats offer true end-to-end encryption where only the sender and receiver can access message content. These chats rely on client-side encryption using Diffie-Hellman key exchange and AES-256-IGE, ensuring that the server has no visibility into the communication.

In contrast, Cloud Chats are designed for convenience and device synchronization. They use transport-level AES encryption, and messages are decrypted and temporarily stored on the server before being delivered. This model mimics Telegram's cloud-based messaging behavior.

The application also incorporates **Forward Secrecy**, a feature that periodically regenerates encryption keys in Secret Chats. This ensures that even if a key is later compromised, past messages remain protected.

Additional functionalities include real-time messaging over WebSockets and an in-memory message queue for offline delivery, all implemented with strict client-side cryptographic processing to maximize security and privacy.

2. Tech Stack and Cryptographic Assurance

Frontend:

- **HTML + Tailwind CSS:** Used to build a clean, responsive UI with minimal dependencies.
- **Vanilla JavaScript:** All client-side logic including WebSocket communication, key generation, and encryption/decryption is implemented using JavaScript.

Backend:

- **Node.js (TypeScript):** The server is built using Node.js for handling WebSocket connections, message routing, session management, and cryptographic functions.

- **WebSocket (ws module):** Real-time communication between clients and the server is handled using WebSockets for both Secret and Cloud Chat modes.

Cryptography:

- **Node Crypto Module:** Core cryptographic operations are powered by the built-in crypto module in Node.js, ensuring performance and security.
 - `crypto.createDiffieHellman()` – secure Diffie-Hellman key exchange
 - `crypto.createHash('sha256')` – for hashing shared secrets and generating `msg_keys`
 - `crypto.createCipheriv()` / `crypto.createDecipheriv()` – AES-256 encryption/decryption
 - `crypto.randomBytes()` – generates salts, IVs, and keys
- **AES-256-IGE Mode:** Simulated using AES-256-CBC logic with additional transformations to replicate IGE chaining, as used in Telegram.
- **KDF (Key Derivation Function):** SHA-256-based derivation of AES keys and IVs from the combination of `msg_key` and the shared key.
- **Client Folder:** Contains HTML/CSS/JS assets and mode-specific UI rendering.
- **Server Folder:** Divided into multiple modules (`storage.ts`, `cloud_handler.ts`, etc.) to manage different chat modes cleanly.

3. Security Concepts

Data Integrity

- **Message Key Verification:** After decryption, the message hash (`msg_key`) is recomputed and matched to detect tampering.
- **Sequence Numbers & Timestamps:** Built into the payload to preserve message order and prevent replay attacks.

Authorization

- **Auth Key (Cloud Chat):** A session-specific 256-bit key ensures message access is restricted to authorized sessions.
- **Auth Key ID:** A 64-bit derived ID enables fast lookup of the auth key.

Confidentiality

- **AES-256 Encryption:** Used in both chat modes.
- **Key Derivation:** SHA-256-based derivation ensures unique AES keys per message.

End-to-End Encryption (Secret Chats)

- Messages are decrypted only on the client.
- Server never has access to keys or plaintext.
- Uses `key_fingerprint` to validate correct key usage.

Forward Secrecy

- To enhance long-term security and prevent retroactive message compromise, SecureMessenger implements a key rotation strategy in its Secret Chat mode. Each key is monitored for usage, and once it has been used to encrypt and decrypt more than 100 messages, or has been active for over one week, a re-keying process is

initiated. This ensures Perfect Forward Secrecy (PFS), meaning that even if a future key is compromised, previous communications remain secure. Old keys are securely discarded and cannot be reconstructed from the newly established keys.

4. Message Payload Construction

Secret Chat Payload Structure:

- Fields:
 - **Length (32-bit)**
 - **Payload Type (32-bit)**
 - **Random Bytes (min 128-bit)**
 - **Layer (32-bit)**
 - **IN_seq_no / OUT_seq_no (32-bit)**
 - **Message Type and Object** (variable length)
 - **Padding (12–1024 bytes)**
- $\text{msg_key} = \text{SHA-256}(\text{payload} + \text{auth fragment})$
- AES-IGE encryption applied using derived keys

Cloud Chat Payload Structure:

- Fields:
 - **Salt (64-bit)**
 - **Session_id (64-bit)**
 - **Payload (variable)**
 - **Padding (12–1024 bytes)**
- Encrypted with AES-IGE using session-based keys
- Sent as: `auth_key_id`, `msg_key`, and encrypted data

5. Secret Chats

5.1 What Are Secret Chats?

Secret Chats use **end-to-end encryption (E2EE)** to ensure that only the sender and receiver can read the message. The server cannot decrypt or see the message content.

5.2 Architecture

Secret Chat is architected with a client-centric security model, where encryption and decryption occur only on user devices. The server has no visibility into the content or keys.

Client Side:

- Each client generates a Diffie-Hellman key pair (public and private).
- Public keys are exchanged via the server.
- A shared secret key is derived from both public-private key pairs.
- Messages are serialized into a structured payload with metadata (layer version, sequence numbers), content, and padding.
- A SHA-256 hash of the payload is used to compute a `msg_key`, from which AES-256 keys and IVs are derived.

- The payload is encrypted using AES-IGE.
- Final message structure includes: `key_fingerprint`, `msg_key`, and encrypted payload.

Server Side:

- Relays public keys and encrypted messages between users.
- Does not decrypt or analyze the content.
- Temporarily queues messages in memory if the recipient is offline.

This design ensures complete end-to-end encryption, making the server a stateless router without any decryption capability.

5.3 Message Flow

1. DH key exchange between clients
2. Payload created and encrypted on sender's side
3. Message sent to and forwarded by server
4. Recipient decrypts using the same shared key

5.4 Security Summary

- Full end-to-end encryption
- Zero server knowledge

6. Cloud Chats

6.1 What Are Cloud Chats?

Cloud Chats prioritize usability and multi-device sync. Messages are encrypted in transit but decrypted and stored on the server.

6.2 Architecture

Cloud Chat architecture reflects a server-centric design where the server temporarily handles decrypted messages to support features like message history, sync across devices, and delivery to offline users.

Client Side:

- After user login, a persistent session key (`auth_key`) is generated through a Diffie-Hellman exchange.
- Each outgoing message is wrapped with metadata such as salt, session ID, and message body.
- A `msg_key` is calculated using SHA-256 over the payload.
- Using the `msg_key` and `auth_key`, the client derives AES-256 encryption keys and IVs via a KDF.
- The message is encrypted using AES-IGE and sent to the server with its `auth_key_id` and `msg_key`.

Server Side:

- The server uses `auth_key_id` to retrieve the correct session key.
- Decrypts the message and verifies its `msg_key` for integrity.
- Stores or relays the message as necessary.

- When delivering to the recipient, the message is re-encrypted using the recipient's session key and forwarded.

This model emphasizes usability by enabling features like chat history and offline message retrieval, though it comes with trade-offs in confidentiality compared to Secret Chats.

6.3 Message Flow

1. Client constructs payload and encrypts
2. Message sent to server
3. Server decrypts and stores
4. Server forwards to recipient
5. Recipient decrypts using own auth key

6.4 Security Summary

- Encrypted transport but server access to plaintext
- Easier multi-device support

7. Secret Chat vs. Cloud Chat – A Comparison

Feature	Secret Chat	Cloud Chat
Encryption Type	End-to-End (AES-IGE)	Transport (AES-256)
Server Decryption	No	Yes
Message Storage	Encrypted blob (in memory)	Plaintext (in memory)
Key Storage	Client only	Server + Client
Message Decryption Location	Client	Server, then Client

8. Design Philosophy and Justification

- **Modular Design:** Separate logic for chat modes avoids conflict.
- **No Persistent Storage:** In-memory message handling for simplicity and privacy.
- **Aligned With MTProto:** Payloads, hashing, and encryption follow Telegram's principles.

9. Output Screenshots

1. **Running the application:**

```
PS C:\Users\chand\OneDrive\Desktop\588-telegram-final\telegram-final\SecureMessenger> npm run dev
> rest-express@1.0.0 dev
> tsx server/index.ts

3:06:25 PM [express] serving on port 5000
█
```

2. User login- 2 users used(Alice and Bob)

```
New WebSocket connection
Received WebSocket message: {"type":"login","username":"alice"}
Parsed message type: login
User alice added to storage. Total users: 1
User logged in: alice with cloud session e3ec3b99
New WebSocket connection
Received WebSocket message: {"type":"login","username":"bob"}
Parsed message type: login
User bob added to storage. Total users: 2
User logged in: bob with cloud session 28f3f821
```

3. Secure connection Establishment between Alice and Bob for Message exchange

Logs on UI:

```
> [5:17:22 PM] Establishing secure WebSocket
connection to ws://localhost:5000/ws
> [5:17:23 PM] WebSocket connection established
> [5:17:23 PM] Initial key pair generated:  $g^a \bmod p = 4ce13da736...$ 
> [5:17:23 PM] Received unknown message type:
session-created
> [5:19:42 PM] This is a secure private connect
with bob
> [5:19:42 PM] Sent key exchange to bob
> [5:19:43 PM] Received key exchange from bob
> [5:19:43 PM] Established shared key with bob
> [5:19:48 PM] This is a secure private connect
with bob
```

Terminal Logs:

```
Recipient "bob" online status for key exchange: true
Forwarding key exchange message to recipient: {
  type: 'key-init',
  senderUsername: 'alice',
  publicKey: '9f0518157169277075987663704556897311579931609658084940446251168323066744400909022623786234609509394304930891273316689730759098274444996551278151620677835909654310012695925497831041355964922
354089215609505285950873509644723642644101266905347725610016905905583702011401507224968484680749928045173364080802082385860348768294356594070408900194270023245022894414560435443889483488025490272805114
90381603250121179694790630636195640364164134487188251620600472601452250163678022391888063548129044407390785927302280818226825915475352936909288998706573276643717434317165271007652555403146747831424258
437029279569772600460805609',
  timestamp: 1746742782828
}
Key exchange: alice -> bob successful
Recipient "alice" online status for key exchange: true
Forwarding key exchange message to recipient: {
  type: 'key-init',
  senderUsername: 'bob',
  publicKey: '290858806247570055902732967020960376268459841775928686686827024204507525974833178738113654421766223177296460207319324530037047040415161427708909883684135800926443217034938930629691111479
3992009171533573173906015008554418927089049790257403570091970437384241795042820761720797646411075908026777950978023553032657169732044414304535780538257274807429625626040402474236081127950412925109933075
474923270994081006084041649608935015105461750503527831404078784867439620814796185170531872008540863613063107640403330167199934045636827794051510187384914895503472305285016117797364708418223308067602722
6005806303409617638171693832',
  timestamp: 1746742783169
}
Key exchange: bob -> alice successful
```

4. When the sender is offline, message is queued until sender comes online:

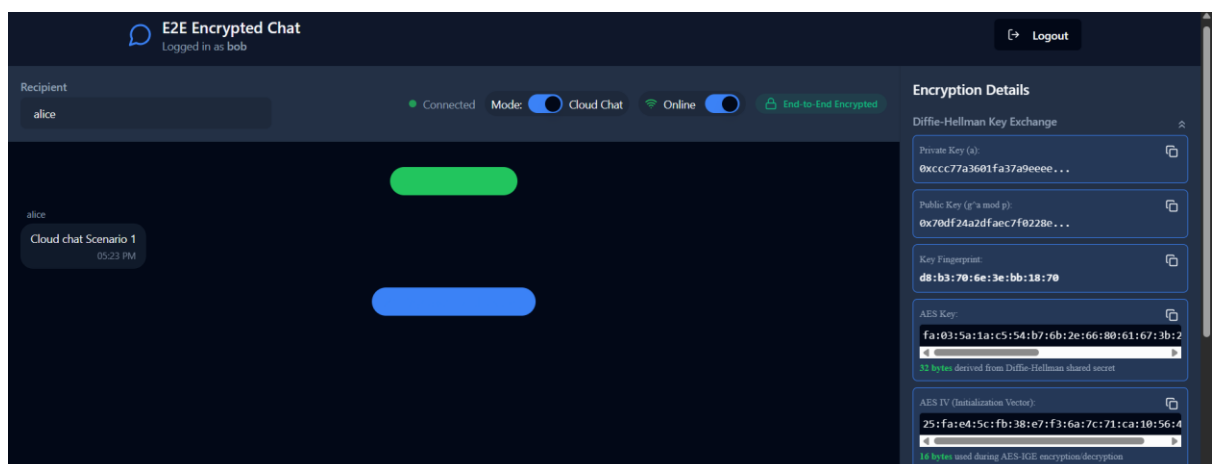
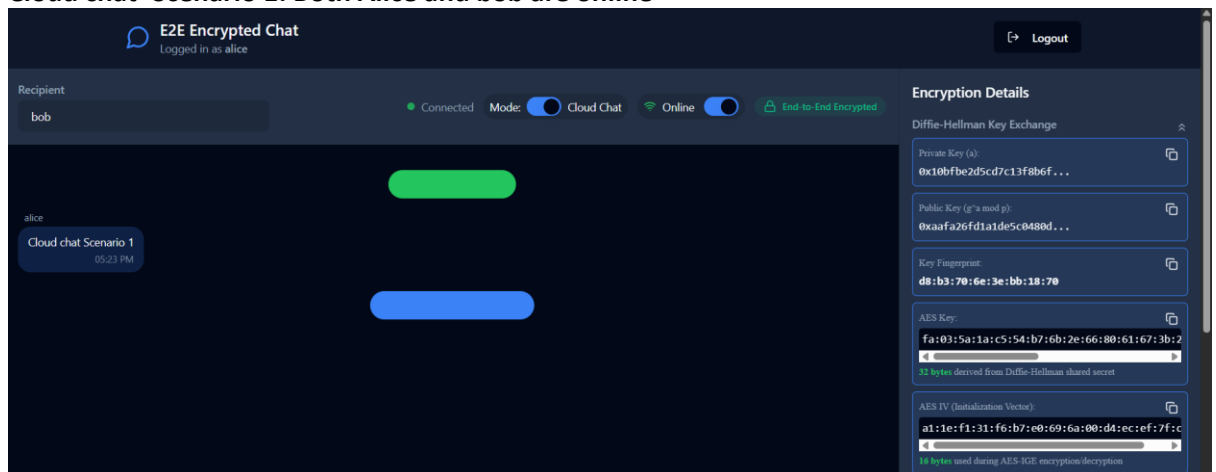
Sender Logs:

```

> [5:57:37 PM] Attempting to send message to bob
> [5:57:37 PM] User offline: Message queued for
later delivery (hii)
> [5:57:38 PM] This is a secure private connect
with bob
> [5:57:41 PM] Setting status to online
> [5:57:41 PM] Attempting to deliver 1 pending
messages
> [5:57:41 PM] Sent queued message: "hii"
> [5:57:41 PM] Status update confirmed: online
> [5:57:41 PM] Received message from alice
> [5:57:41 PM] Received message: "hii"

```

5. Cloud chat- Scenario 1: Both Alice and bob are online



Encryption Logs:

```

[5:23:30 PM] [CloudChat] Mode: Cloud Chat
[5:23:30 PM] [CloudChat] Sending plaintext
message: "Cloud chat Scenario 1"
[5:23:30 PM] [CloudChat] Encrypting using
auth_key_id
[5:23:30 PM] [CloudChat] Derived msg_key:
861e80180acb87f4b883aa14c2a01c79
[5:23:30 PM] [CloudChat] AES-256 encryption
complete.
[5:23:30 PM] [CloudChat] Encrypted payload sent
to server.
> [5:23:32 PM] This is a secure private connect
with bob

```

Decryption Logs:

```

with alice
[5:23:30 PM] [CloudChat] Received encrypted
message from user: alice
[5:23:30 PM] [CloudChat] Derived AES key/IV
using msg_key: 861e80180acb87f4b883aa14c2a01c79
[5:23:30 PM] [CloudChat] Decryption successful.
Plaintext: "Cloud chat Scenario 1"
[5:23:30 PM] [CloudChat] Storing message in
cloud memory (to: bob)
> [5:23:31 PM] This is a secure private connect

```

Terminal Logs:

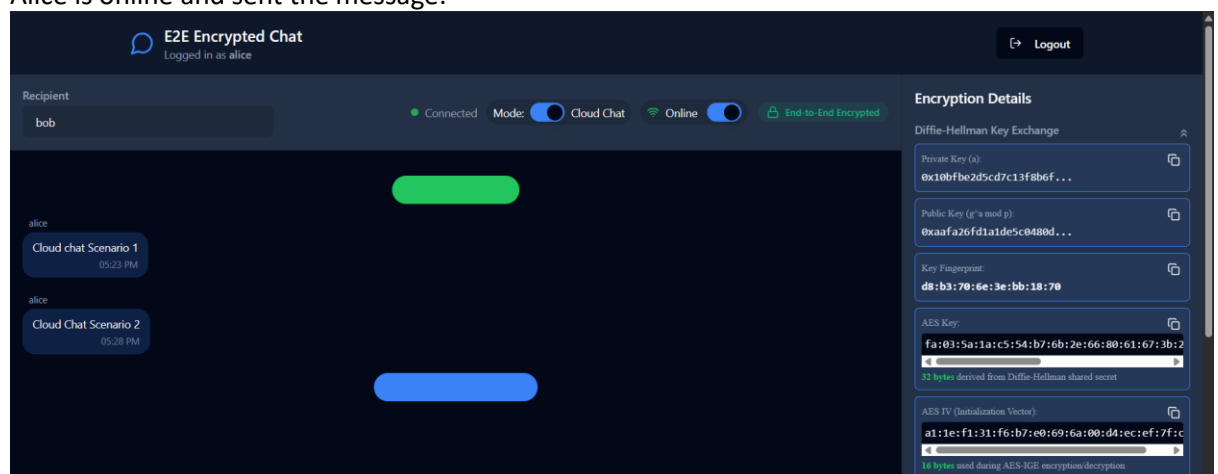
```

Received WebSocket message: {"type":"cloud-message","cloudMessage":{"msg_id":"cloud-1746743010030","seq_no":103920,"auth_key_id":"","msg_key":"861e80180acb87f4b883aa14c2a01c79","from_id":"alice","to_id":"bob","timestamp":1746743010030,"content":"Cloud chat Scenario 1","isCloudMessage":true}}
Parsed message type: cloud-message
[CLOUD CHAT] Processing message on server
Invalid session
[CLOUD CHAT] Message processed and stored on server
[CLOUD CHAT] Delivering message to online recipient bob
Received WebSocket message: {"type":"check-user","username":"bob"}
Parsed message type: check-user
user status check: bob is online

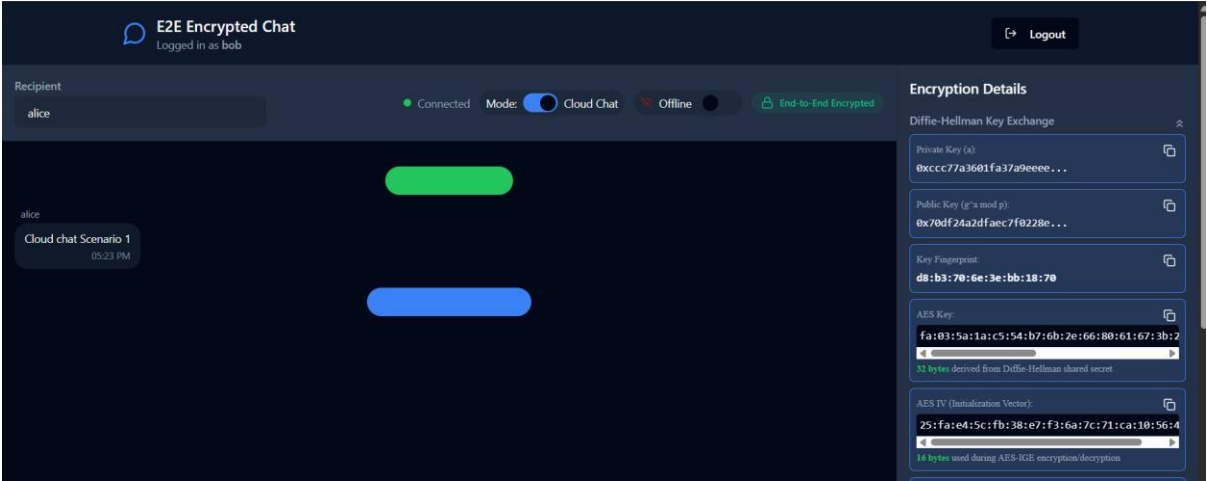
```

6. Cloud chat- Scenario 2: Alice is online and bob is offline

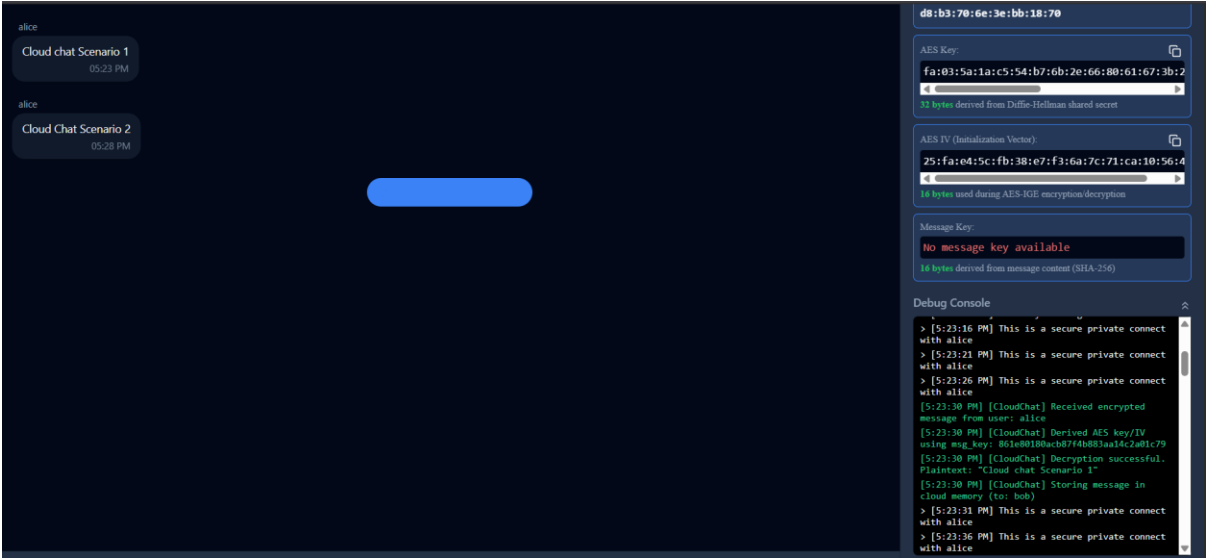
Alice is online and sent the message:



Bob is offline”



Bob comes online and receives the message:



Terminal logs:

```
Forcing both sides to initiate key exchange
Received websocket message: {"type":"cloud-message","cloudMessage":{"msg_id":"cloud-1746743280695","seq_no":480140,"auth_key_id":"","msg_key":"c6009f8a3b722e61e11ffdeac7a6c2c","from_id":"alice","to_id":"bob","timestamp":1746743280695,"content":"Cloud chat Scenario 2","isCloudMessage":true}}
Parsed message type: cloud-message
[Cloud Chat] Processing message on server
Invalid session
[Cloud Chat] Message processed and stored on server
[Cloud Chat] Recipient bob is offline, message stored for later delivery
```

Encryption logs on UI:

```

> [5:28:00 PM] Attempting to send message to bob
[5:28:00 PM] [CloudChat] Mode: Cloud Chat
[5:28:00 PM] [CloudChat] Sending plaintext
message: "Cloud Chat Scenario 2"
[5:28:00 PM] [CloudChat] Encrypting using
auth_key_id
[5:28:00 PM] [CloudChat] Derived msg_key:
c6009f8a3b722e61e111ffdeac7a6c2c
[5:28:00 PM] [CloudChat] AES-256 encryption
complete.
[5:28:00 PM] [CloudChat] Encrypted payload sent
to server.
> [5:28:02 PM] This is a secure private connect
with bob

```

Decryption logs on UI(bob's side):

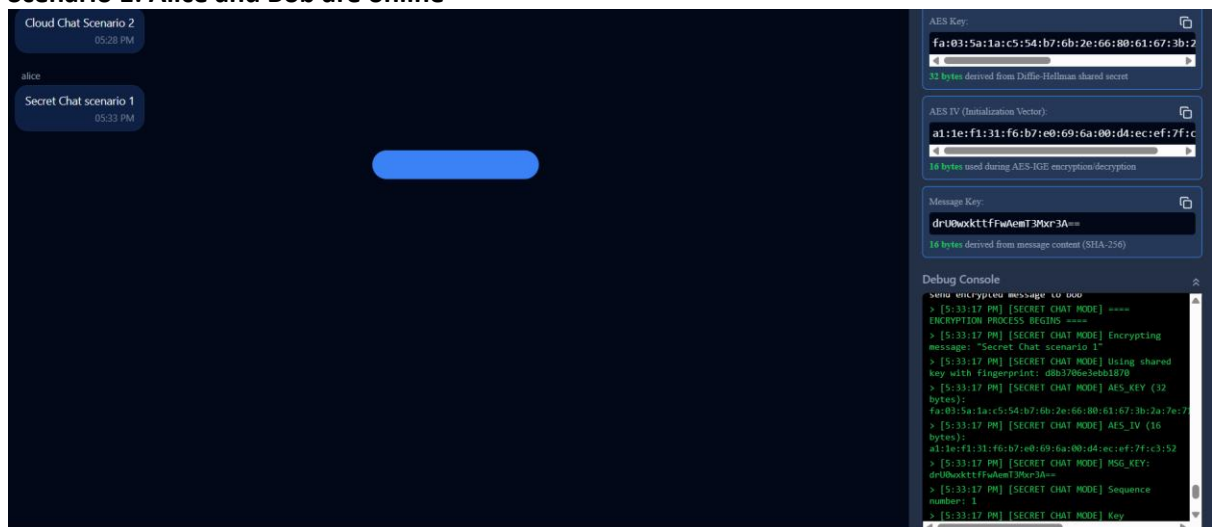
```

[5:28:31 PM] [CloudChat] Storing message in
cloud memory (to: bob)
[5:28:31 PM] [CloudChat] Received encrypted
message from user: alice
[5:28:31 PM] [CloudChat] Derived AES key/IV
using msg_key: c6009f8a3b722e61e111ffdeac7a6c2c
[5:28:31 PM] [CloudChat] Decryption successful.
Plaintext: "Cloud Chat Scenario 2"
[5:28:31 PM] [CloudChat] Storing message in
cloud memory (to: bob)
> [5:28:31 PM] Status update confirmed: online

```

Secret Chat

Scenario 1: Alice and Bob are online



The screenshot displays a chat application interface. On the left, a chat list shows two scenarios: "Cloud Chat Scenario 2" at 05:28 PM and "Secret Chat scenario 1" at 05:33 PM, with "alice" listed as the contact. The main chat window is currently empty. On the right, a debug console provides detailed logs of the encryption process for "Secret Chat scenario 1".

Debug Console Log:

```

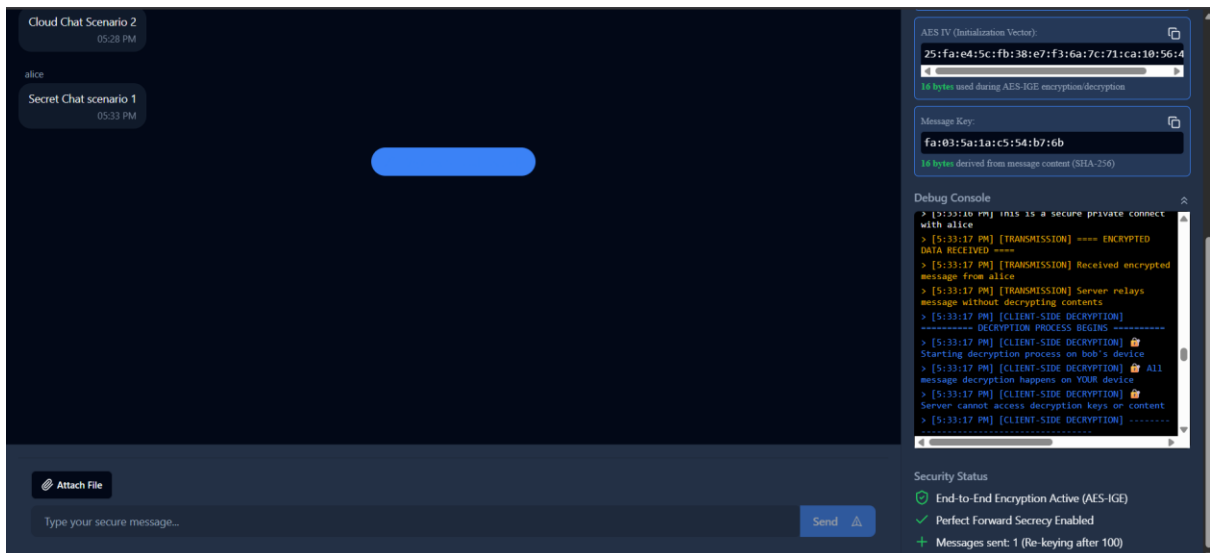
> [5:33:17 PM] [SECRET CHAT MODE] Key
fa:03:5a:1a:c5:54:b7:6b:2e:66:80:61:67:3b:2
32 bytes derived from Diffie-Hellman shared secret

AES IV (Initialization Vector):
a1:1e:f1:31:f6:b7:e0:69:6a:00:d4:ec:ef:7f:c
10 bytes used during AES-IGE encryption/decryption

Message Key:
drU0w0kttfFwAemT3Mr3A==
16 bytes derived from message content (SHA-256)

> [5:33:17 PM] [SECRET CHAT MODE] Key
fa:03:5a:1a:c5:54:b7:6b:2e:66:80:61:67:3b:2a:7e:7
> [5:33:17 PM] [SECRET CHAT MODE] AES_IV (16
bytes):
a1:1e:f1:31:f6:b7:e0:69:6a:00:d4:ec:ef:7f:c3:52
> [5:33:17 PM] [SECRET CHAT MODE] MSG_KEY:
drU0w0kttfFwAemT3Mr3A==
> [5:33:17 PM] [SECRET CHAT MODE] Sequence
number: 1
> [5:33:17 PM] [SECRET CHAT MODE] Key

```



Encryption logs on Alice End:

```
> [5:33:17 PM] [SECRET CHAT MODE] ====
ENCRIPTION PROCESS BEGINS ====
> [5:33:17 PM] [SECRET CHAT MODE] Encrypting
message: "Secret Chat scenario 1"
> [5:33:17 PM] [SECRET CHAT MODE] Using shared
key with fingerprint: d8b3706e3ebb1870
> [5:33:17 PM] [SECRET CHAT MODE] AES_KEY (32
bytes):
fa:03:5a:1a:c5:54:b7:6b:2e:66:80:61:67:3b:2a:7e:71
> [5:33:17 PM] [SECRET CHAT MODE] AES_IV (16
bytes):
a1:1e:f1:31:f6:b7:e0:69:6a:00:d4:ec:ef:7f:c3:52
> [5:33:17 PM] [SECRET CHAT MODE] MSG_KEY:
drU0wxkttfFwAemT3Mxr3A==
> [5:33:17 PM] [SECRET CHAT MODE] Sequence
number: 1
> [5:33:17 PM] [SECRET CHAT MODE] Key
fingerprint: d8b3706e3ebb1870
```

Debug Console

```
a1:1e:f1:31:f6:b/:e0:69:6a:00:d4:ec:ef:/f:c3:52
> [5:33:17 PM] [SECRET CHAT MODE] MSG_KEY:
drU0wxkttfFwAemT3Mxr3A==
> [5:33:17 PM] [SECRET CHAT MODE] Sequence
number: 1
> [5:33:17 PM] [SECRET CHAT MODE] Key
fingerprint: d8b3706e3ebb1870
> [5:33:17 PM] [SECRET CHAT MODE] Message
encrypted successfully
> [5:33:17 PM] [SECRET CHAT MODE] Encrypted
message sent to bob
> [5:33:17 PM] [SECRET CHAT MODE] ====
ENCRYPTION PROCESS COMPLETE ====
> [5:33:17 PM] Received unknown message type:
message-sent
> [5:33:51 PM] This is a secure private connect
with bob
```

Decryption logs on Bob's end:

```
> [5:33:17 PM] [TRANSMISSION] ==== ENCRYPTED
DATA RECEIVED ====
> [5:33:17 PM] [TRANSMISSION] Received encrypted
message from alice
> [5:33:17 PM] [TRANSMISSION] Server relays
message without decrypting contents
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION]
===== DECRYPTION PROCESS BEGINS =====
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 🔒
Starting decryption process on bob's device
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 🔒 All
message decryption happens on YOUR device
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 🔒
Server cannot access decryption keys or content
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] -----
-----
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 1 KEY
```

Debug Console

Server cannot access decryption keys or content

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] -----

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 1 KEY
FINGERPRINT: d8b3706e3ebb1870

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 2
MESSAGE KEY: drU0wxkttfFwAemT3Mxr3A==

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 3
IDENTIFIED SHARED KEY with alice

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 4
DERIVED AES KEY/IV FOR DECRYPTION:

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
AES_KEY (32 bytes):

fa:03:5a:1a:c5:54:b7:6b:2e:66:80:61:67:3b:2a:7e:71

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
AES_IV (16 bytes):

25:fa:e4:5c:fb:38:e7:f3:6a:7c:71:ca:10:56:48:61

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →

===== DECRYPTION PROCESS BEGINS =====

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 🗝️
Decrypting message from alice on bob's device

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION]
DECRYPTION KEY DETAILS:

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
Fingerprint: d8b3706e3ebb1870

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] → AES
Key (32 bytes):

fa:03:5a:1a:c5:54:b7:6b:2e:66:80:61:67:3b:2a:7e:71

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] → AES
IV (16 bytes):

25:fa:e4:5c:fb:38:e7:f3:6a:7c:71:ca:10:56:48:61

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] DECRYPT
CHECKPOINT: Converting encrypted data for alice

> [5:33:17 PM] [CLIENT-SIDE DECRYPTION]
Decryption happens entirely on your device, not
on server

```
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION]
Processing 1520 bytes of encrypted data
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION]
Decrypting with message-specific IV...
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] AES_KEY:
fa:03:5a:1a:c5:54:b7:6b:2e:66:80:61:67:3b:2a:7e:71
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION]
MESSAGE_IV:
99:59:4d:34:03:f3:bb:d9:1d:b9:64:63:a1:a0:2d:54
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] Key
Fingerprint: d8b3706e3ebb1870
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION]
DECRYPTED MESSAGE:
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
From: alice to bob
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
Content: "Secret Chat scenario 1"
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
Content: "Secret Chat scenario 1"
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
Sent: 5/8/2025, 5:33:17 PM
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
Sequence: 1, Padding: 676 bytes
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION]
===== DECRYPTION COMPLETE =====
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 5
DECRYPTED PLAINTEXT:
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
Successfully decrypted message from alice
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
Message content: "Secret Chat scenario 1"
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] 6
PARSED MESSAGE OBJECT:
> [5:33:17 PM] [CLIENT-SIDE DECRYPTION] →
Sender: alice
```

Terminal Logs:

```

Received websocket message: {"type":"encrypted-message","recipient":"bob","encryptedMessage":{"msgKey":"drU0akdtffwAm13br3A==","encryptedData":"wt3Gzbb4ehy0UzF4ASPSHfFmlZ10TymCdfp0i6ugj1ikPGutL
klUuHfXbW51xqW1YXDSRQld1/W6jehNvbutF1xp2AsySgQj8PT4wq1YVR23ezY8M6PqKtrmKLV4H8gdm3D3V5KYJn8yuhul531DfR1cFQXCh3AW6D4UoCRChXc9dHk0/macqb8+oR28H-21ntub.yjBg4kryOLE4qf75BrWjUn96qrFy1/43boPGw/wa
Tbjc/yoloK5T4W6TVKE8oc2WEDgJntAygyQ7rtJLLPfwUjbp0b55ccjpor5h75jM1V4Ml.u0j0K9inx9Kas1asU7YqpxEv90bc29157V415e8Nz2a1AflL8JezrL8nm28Gzet-x0CjR0ZukB1owaps8X1M2E8BZGZ6+1eFRjM11syc1cstdxz/ZZGdwa8R8SP3dG1
u0eqKL1P3/ykVnCCB1WU/hspUckB385bA1+S6e0/CvVA/8LggADG18zro19pndN1BumSX4H9rVAXqbZqhy5FnsVtDC9P78m6zyFouCwaWv888S2fmgFhW50fGcDMP351Z/17dMERd1/3g5ekaDgh11v0T8Alk6wqVqB1K7ShqatzR4D1ahk1Bx
4yYfQ8389896q2W40m2a1B1mPpHfhwJ05y0K5jha15ZLVay2ZVw0yymH0u55Uyc-C9X8H1Hdc3m+34dshqy9D1c5Hb1CvZ1d1cXK34Kd1J0sh1YbW8esYKqgk1V1p3E-VQgfHb1d1552yU1UQFF7J3b24rk42hUe5fW4P1
C4vqE+1e/jDh1Mq8M4ukUpoe2c0d5L110yWEden1nk0uKd.k0A591JdC3R3/610h1fWfW1q02V0naflKp9Kk0b51M/n0M44TVMUajY5e81CFPXTHCCKLrf9ysjED88+c82v1n2h8fYhJ55fr13J857K8csh1P0y0ydg8t1cGK0
s1fW68435ev9y2h1zgyKcP00X6grf1DYUw97C7Q1d4y61M02f6d10Zv5Z8an200x5t6g1A1EnP1G7f0fWwpyyP3h1Q0Vf3B81st1n2yWlqJh1Pdnc2H0mta015dZa61a9812p2Yg8Zf91RL8so4AZ2h1z8XGCV09f98Qz23bD072200dMD
2Eh1RV1naw5k2wM7P11W5Y9uCG0dP9K4K1L6H8S06BBh1iJ00Qy8uSK/COlGu5+ge1J5A5mpkUpF350r18Zkt3PLVr8Q6CQpbr1nn1/558kpzgsdP1Q57u1/LopVgPnuc3Waz105oV05H9HfV8ajht/uaq10m1BdStfKGrnt5Kcc2XAF6EV8rNX
551dMddEj64b0xbh9rOXV8U7P/pycqa01trsrab0yafm1Hw02K0ga8cK8SSXpQ0bL11PuE1V5t/LR/9tC145nuGrjeshwam1z8RE1NqP7/souLMVteDfug84whs1VDWfQf-aub8e1u22na9Ma1PPYTT0w9a5xyFng13JNYHfAzrwc0arPrdaDnng1VpNE0
kh7CPB5d5Z7KwM1VCL1ha1U07K0m1bV80614m6dHfKCGMLsFELU0Hv5dPUGufoah5Bhr2B8KX/3k07/PC1Q0eedhwaabb1UtnW0R1CPL10qR8/yWotP15x01dUVqg10VdyvdabaGkq55KULN/s51QrHndp0957yc0R8Iv4TqHn/Ygos1QM0R6CDv8g
d4wVdRHHak5aZV0131AKCuAnuej6+ewd5vvcFerjahn3VA7fgC8t5W/Yr88031LW11E3B85ngq0Vf8aTo1MEU5e8/0Vrue/agxj3183e5Kbupw1g812nutoQYfKnAsolvtgD0CUnNo30a6Z1Bo7YyC7HMcCuBL22g==","keyFingerPrint":"d8b3706e3ebb1876
","sequenceNumber":1,"senderUsername":"alice","iv":"mV1NMAPZu9kduRjoatVA=="},"timestamp":1746743597530,"id":"enc-1746743597530-8xmt8rq"}
Parsed message type: encrypted-message
Processing encrypted message: {
  sender: 'alice',
  recipient: 'bob',
  hasEncryptedMessage: true,
  hasMessageKeys: 'msgKey: true, encData: true'
}
Recipient "bob" online status for message: true

```

Scenario 2: Alice is Online and Bob is offline

Alice is online and sends message while bob is offline:

The screenshot shows a chat application interface. On the left, a list of messages from Alice to Bob is visible, including 'Cloud chat Scenario 1', 'Cloud Chat Scenario 2', 'Secret Chat scenario 1', and 'Secret Chat scenario 2'. The main chat area shows a green 'Send' button. On the right, the 'Encryption Details' panel is open, displaying the Diffie-Hellman Key Exchange process, including Private Key (a), Public Key (g^a mod p), Key Fingerprint, AES Key, AES IV (Initialization Vector), and Message Key. The Debug Console at the bottom shows logs for the encryption process.

Bob comes online and receives the message:

The screenshot shows the chat application interface after Bob has come online. The chat list on the left is the same. The main chat area now shows a blue 'Send' button. The 'Encryption Details' panel on the right shows the decryption process, including the AES Key, AES IV, and Message Key. The Debug Console at the bottom shows logs for the decryption process, including 'Starting decryption process on bob's device' and 'All message decryption happens on VOXIR device'.

Encryption logs on Alice end:


```
Debug Console
> [5:37:44 PM] [SECRET CHAT MODE] ====
ENCRIPTION PROCESS BEGINS ====
> [5:37:44 PM] [SECRET CHAT MODE] Encrypting
message: "Secret Chat scenario 2"
> [5:37:44 PM] [SECRET CHAT MODE] Using shared
key with fingerprint: d8b3706e3ebb1870
> [5:37:44 PM] [SECRET CHAT MODE] AES_KEY (32
bytes):
fa:03:5a:1a:c5:54:b7:6b:2e:66:80:61:67:3b:2a:7e:71
> [5:37:44 PM] [SECRET CHAT MODE] AES_IV (16
bytes):
a1:1e:f1:31:f6:b7:e0:69:6a:00:d4:ec:ef:7f:c3:52
> [5:37:44 PM] [SECRET CHAT MODE] MSG_KEY:
RwqtaxHFREYmfSryIaCbTA==
> [5:37:44 PM] [SECRET CHAT MODE] Sequence
number: 2
> [5:37:44 PM] [SECRET CHAT MODE] Key
fingerprint: d8b3706e3ebb1870

> [5:37:44 PM] [SECRET CHAT MODE] Key
fingerprint: d8b3706e3ebb1870
> [5:37:44 PM] [SECRET CHAT MODE] Message
encrypted successfully
> [5:37:44 PM] [SECRET CHAT MODE] Encrypted
message sent to bob
> [5:37:44 PM] [SECRET CHAT MODE] ====
ENCRIPTION PROCESS COMPLETE =====
```

Decryption logs on bob's end(For simplicity I have just added the summary logs):

===== DECRYPTION COMPLETE =====

> [5:37:47 PM] [CLIENT-SIDE DECRYPTION] 5
DECRYPTED PLAINTEXT:

> [5:37:47 PM] [CLIENT-SIDE DECRYPTION] →
Successfully decrypted message from alice

> [5:37:47 PM] [CLIENT-SIDE DECRYPTION] →
Message content: "Secret Chat scenario 2"

> [5:37:47 PM] [CLIENT-SIDE DECRYPTION] 6
PARSED MESSAGE OBJECT:

> [5:37:47 PM] [CLIENT-SIDE DECRYPTION] →
Sender: alice

> [5:37:47 PM] [CLIENT-SIDE DECRYPTION] →
Recipient: bob

> [5:37:47 PM] [CLIENT-SIDE DECRYPTION] →
Timestamp: 5/8/2025, 5:37:44 PM

> [5:37:47 PM] [CLIENT-SIDE DECRYPTION] →
Sequence Number: 2

Terminal Logs:

```
Received WebSocket message: {"type":"encrypted-message","recipient":"bob","encryptedMessage":{"msgKey":"8q1aHfREYfSryIaCbTA==","encrypteddata":"1131swJGcD148K6akd8MwJtQ0D56s4M6z0M1g1YodhECYw94LS79HEoDhmp8B28b1VUC07JP52pS/9w1JfWd+hFRVsykLPin1zsJ9Tsz08n1k1gR+h808wG5Sohm31pxCw9pBPCpzdVcc/aL2vU1A11StsfGjC0387aP0820X81ac67R1p6Sp4y7C741thvQKzESTb0p8Q8SExqE4jD64eVCj7538u2A0hZ1JCFIkw5gAT0/49eQmKsy+21K1NQMBVKUREdnbG1f9s2u80fugU0SYddk6wH7p1MxoZHP6DTJIDmV5m0n6vTgOUATAJOGzGmV/jhsEUqvrGmTK237DFTfBYAGhr+mc1lybJ7HPrQmCawL+CtyCcB3EUqUfHQ01pnp8Lh6/11VouUG6Bur4sajjyASK065Um/VdGn1ufZnCB091g4B1Q031/Xe2p1uEkwhxwUsp1gk=","keyFingerprint":"d8b3706e3ebb1870","sequenceNumber":2,"senderUsername":"alice","iv":"0m+R880dWwVVG9u60tPlw=="},"timestamp":"1746743864097","id":"enc-1746743864097-bq1jp90"}
Parsed message type: encrypted-message
Processing encrypted message: {
  sender: 'alice',
  recipient: 'bob',
  hasEncryptedMessage: true,
  hasMessageKeys: {msgKey: true, encData: true}
}
Recipient "bob" online status for message: false
Recipient bob is offline, storing message for later delivery
```

Scenario 3: Alice is Online Bob is Offline. Alice sends a message to bob and goes Offline. Bob comes online after Alice goes offline.

Alice is Online Bob is Offline. Alice sends a message to bob and goes Offline:

The screenshot shows a chat interface on the left with three messages from 'alice' at 05:33 PM, 05:37 PM, and 05:40 PM, all labeled 'Secret Chat scenario 1', 'Secret Chat scenario 2', and 'Secret Chat scenario 3' respectively. The chat input field is empty. On the right, the 'Debug Console' is open, showing the following logs:

```
> [5:40:58 PM] [SECRET CHAT MODE] ----
ENCRYPTION PROCESS BEGINS ----
> [5:40:58 PM] [SECRET CHAT MODE] Encrypting
message: "Secret Chat scenario 3"
> [5:40:58 PM] [SECRET CHAT MODE] Using shared
key with fingerprint: d8b3706e3ebb1870
> [5:40:58 PM] [SECRET CHAT MODE] AES_KEY (32
bytes):
fa:03:5a:1a:c5:54:b7:6b:2e:66:80:61:67:3b:2a:7e:7
> [5:40:58 PM] [SECRET CHAT MODE] AES_IV (16
bytes):
a1:1e:f1:31:f6:b7:e0:69:6a:00:d4:ec:ef:7f:c
> [5:40:58 PM] [SECRET CHAT MODE] MSG_KEY:
qJlVljmy/x3mnnKUZ++CQ==
> [5:40:58 PM] [SECRET CHAT MODE] Sequence
number: 3
> [5:40:58 PM] [SECRET CHAT MODE] Key
fingerprint: d8b3706e3ebb1870
```

Below the console, the 'Security Status' section shows:

- End-to-End Encryption Active (AES-IGE)
- Perfect Forward Secrecy Enabled
- Messages sent: 3 (Re-keying after 100)

Bob comes online after Alice goes offline and receives the message:

The screenshot shows the same chat interface as before, but now Bob is online. The 'Debug Console' shows the following logs:

```
> [5:41:06 PM] [TRANSMISSION] ---- ENCRYPTED
DATA RECEIVED ----
> [5:41:06 PM] [TRANSMISSION] Received encrypted
message from alice
> [5:41:06 PM] [TRANSMISSION] Server relays
message without decrypting contents
> [5:41:06 PM] [CLIENT-SIDE DECRYPTION]
----- DECRYPTION PROCESS BEGINS -----
> [5:41:06 PM] [CLIENT-SIDE DECRYPTION]
Starting decryption process on bob's device
> [5:41:06 PM] [CLIENT-SIDE DECRYPTION] All
message decryption happens on YOUR device
> [5:41:06 PM] [CLIENT-SIDE DECRYPTION]
Server cannot access decryption keys or content
> [5:41:06 PM] [CLIENT-SIDE DECRYPTION] -----
```

The 'Security Status' section remains the same:

- End-to-End Encryption Active (AES-IGE)
- Perfect Forward Secrecy Enabled
- Messages sent: 3 (Re-keying after 100)

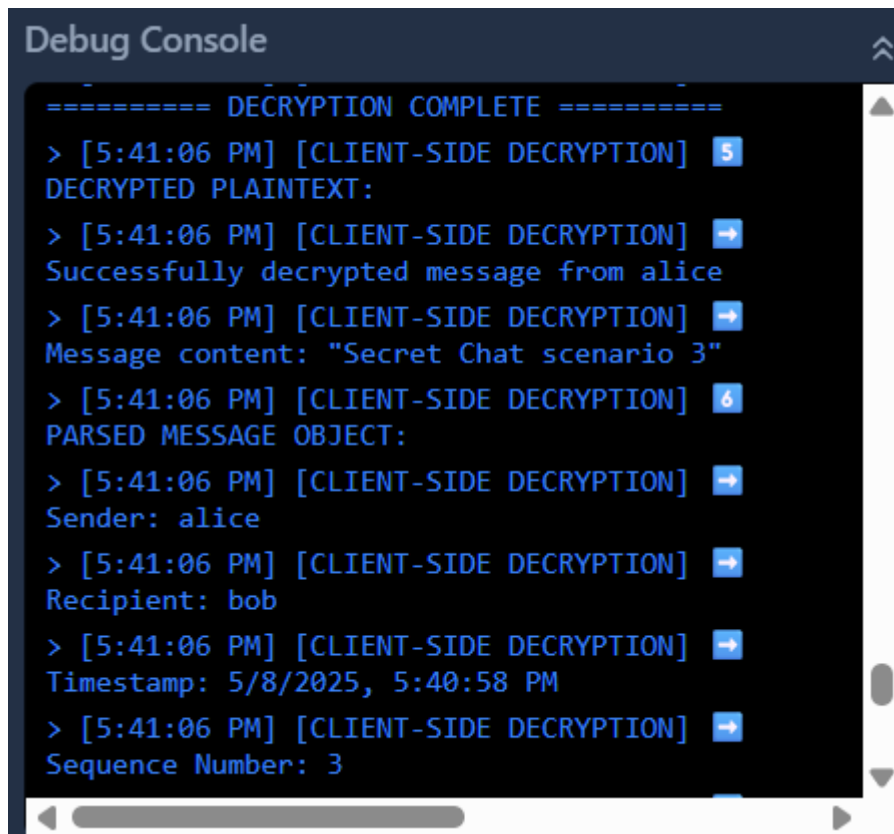
Encryption logs on Alice end:

```
> [5:40:58 PM] [SECRET CHAT MODE] ====
ENCRYPTION PROCESS BEGINS ====
> [5:40:58 PM] [SECRET CHAT MODE] Encrypting
message: "Secret Chat scenario 3"
> [5:40:58 PM] [SECRET CHAT MODE] Using shared
key with fingerprint: d8b3706e3ebb1870
> [5:40:58 PM] [SECRET CHAT MODE] AES_KEY (32
bytes):
fa:03:5a:1a:c5:54:b7:6b:2e:66:80:61:67:3b:2a:7e:71
> [5:40:58 PM] [SECRET CHAT MODE] AES_IV (16
bytes):
a1:1e:f1:31:f6:b7:e0:69:6a:00:d4:ec:ef:7f:c3:52
> [5:40:58 PM] [SECRET CHAT MODE] MSG_KEY:
qJIv1jvmy/x3nrnKUZ++CQ==
> [5:40:58 PM] [SECRET CHAT MODE] Sequence
number: 3
> [5:40:58 PM] [SECRET CHAT MODE] Key
fingerprint: d8b3706e3ebb1870
```

Debug Console

```
qJIv1jvmy/x3nrnKUZ++CQ==
> [5:40:58 PM] [SECRET CHAT MODE] Sequence
number: 3
> [5:40:58 PM] [SECRET CHAT MODE] Key
fingerprint: d8b3706e3ebb1870
> [5:40:58 PM] [SECRET CHAT MODE] Message
encrypted successfully
> [5:40:58 PM] [SECRET CHAT MODE] Encrypted
message sent to bob
> [5:40:58 PM] [SECRET CHAT MODE] ====
ENCRYPTION PROCESS COMPLETE ====
> [5:40:58 PM] Received unknown message type:
message-sent
> [5:41:01 PM] This is a secure private connect
with bob
> [5:41:04 PM] Setting status to offline
> [5:41:04 PM] Status update confirmed: offline
```

Decryption logs on Bob's end:



Terminal Logs:

```
Notifying alice that bob is looking for them  
Forcing both sides to initiate key exchange  
  
Received WebSocket message: {  
  "type": "encrypted-message",  
  "recipient": "bob",  
  "encryptedMessage": {"keyId": "e1c7089f-3d61-f3c1-a2cc-dcf0", "cryptorData": {"PKID:bobMfcAlcaUkGgagMTtVWfrzYQJSPKqLwFKyPjvOtoUx,ykuuUlnHlhmS4MdsRGSQpD3ZGclmFns1nN96ew37IUVXELAAJ3FsTofBwfht5,  
LbmhcCqqrYAwz2ZF6p2oZvA9p2b2u2bfrraf92IS12JsoR9M2MH2M1dRheezS2ozv4dcGvD9Hil98rMaL2X99gucdxd/yAS6Db20ee9NmKqULdE6mAd1569tp25vmGciQ117zhbehrcege8bStDAK2RH9hyerPa2p6ll1/RS9Et3VSBLDc3z2v30rt/3ODcyPhcAW6214wJdz24mg6  
LbzL53612E4adPPgAs5/dsvD9MeAdvPtAt97buOVADTFPMsfurteAZ35+OHKK++TW3Jjad35MtL2CserdfCh7vtH+cUbpqNYq4dCc3KhC6QLcd3Ln2dAJDMYUwdFlcuOXGP7Qy8kbOp/pdpZ99K7zfge3y48SN/FHMzPgSTALZLi2laIKounhwang29KHrc3RCRB85XP100/qnoee+/yoCUhs4SV9SL-jbMLw  
L3jdmp2ocgw4wD0uCD0/zYHF4HQ3ysSm113BGGLlyLCRxfKyrs9Nh/vKgQRDT7zrkMKPlpJD1fbRzyJylUwaVsDK2Am+", "keyIdentifier": "08b37064eb3db1870", "sequenceNumber": "3", "senderUsername": "alice", "iv": "/pJEUE5SiDJmWCKTDIA=r", "timestamp": "174674464958549+-enc":"174674464958549-es4"}  
}  
  
Parsed message type: encrypted-message  
Processing encrypted message: {  
  sender: 'alice',  
  recipient: 'bob',  
  hasEncryptedMessage: true,  
  hashMessageKeys: {keyId: true, endData: true}}  
}  
  
Recipient "bob" is offline, storing message for later delivery  
Added pending messages for bob, total pending: 1  
Received WebSocket message: ("type":"check-user","username":"alice")  
Parsed message type: check-user  
User status checks: alice is online  
Sending status response: { type: 'user-status', username: 'alice', online: true }  
Notifying alice that bob is looking for them  
Forcing both sides to initiate key exchange  
  
Received WebSocket message: ("type":"check-user","username":"bob")  
Parsed message type: check-user  
User status checks: bob is offline  
Sending status response: { type: 'user-status', username: 'bob', online: false }  
Notifying bob that alice is looking for them  
Forcing both sides to initiate key exchange  
  
Received WebSocket message: ("type":"status-update","online":false)  
Parsed message type: status-update  
Use user status update: offline  
Use alice status changed to: offline  
Received WebSocket message: ("type":"check-user","username":"alice")  
Parsed message type: check-user  
User status checks: alice is offline  
Sending status response: { type: 'user-status', username: 'alice', online: false }  
Notifying alice that bob is looking for them  
Forcing both sides to initiate key exchange  
  
Received WebSocket message: ("type":"status-update","online":true)  
Parsed message type: status-update  
Use bob status update: online  
Use bob status changed to: online  
Delivering 1 pending messages to bob  
Delivering 1 pending messages to bob  
Delivered pending unexpired message from alice to bob (recipient online)
```

Forward Secrecy:

To enhance long-term security and prevent retroactive message compromise, SecureMessenger implements a key rotation strategy in its Secret Chat mode. Each key is monitored for usage, and once it has been used to encrypt and decrypt more than 100 messages, or has been active for over one week, a re-keying process is done.

Attach File

Type your secure message...

Send

Security Status

✓ End-to-End Encryption Active (AES-IGE)

✓ Perfect Forward Secrecy Enabled

+ Messages sent: 3 (Re-keying after 100)