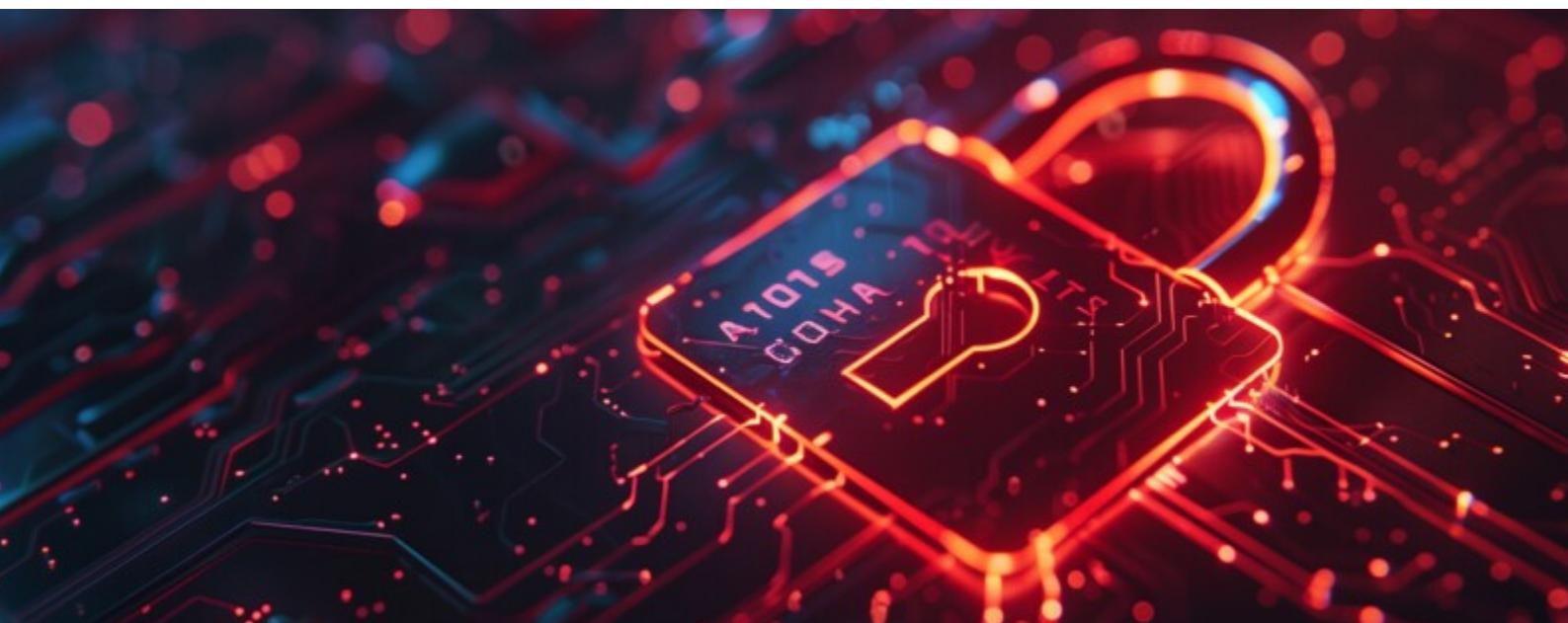


Aplikasi Chat dengan Hybrid Cryptography (Algoritma RSA + AES)

Tugas Proyek Ujian Tengah Semester – Kriptografi A

Kusworo Anindito, S.T.,M.T.



Disusun Oleh:

Arwindo Sedy Pratama
230712555

Program Studi Informatika
Fakultas Teknologi Industri
Universitas Atma Jaya Yogyakarta

Daftar Isi

Lembar Jawaban	3
Nomor 1	3
Nomor 2	4
Nomor 3	5
Nomor 4	6
Nomor 5	7
Nomor 6	8
Nomor 7	8
Nomor 8	9
Informasi Proyek.....	10
Struktur Direktori.....	10
Basis Data	10
Back-end.....	10
DB Connector.....	10
Register.....	10
Get Public Key	11
Save Message.....	11
Read Message	12
Front-end	12
CSS.....	12
Indeks.....	12
Sequence Diagram	15
Pengujian.....	15

Lembar Jawaban

Nomor 1

1. Pada aplikasi yang dibuat, kunci private disimpan di mana?

Kunci private disimpan di Session Storage browser, key privatekey, hanya digunakan selama tab aktif dan hilang saat refresh/tutup tab. Disediakan saat menekan tombol “Daftar” dengan `sessionStorage.setItem("privatekey", rsa.getPrivateKey())`, dan dipakai saat pengguna menekan tombol “Lihat Pesan” dengan `sessionStorage.getItem("privatekey")`.

Private key tidak dikirim ke server, yang dikirim hanya username dan public_key saat registrasi, lalu saat kirim pesan hanya asal, tujuan, kunci (session key RSA), dan pesan (cipher). Jika private key tidak ada, aplikasi tidak dapat mendekripsi.

The screenshot shows a web-based hybrid encryption chat application titled "Hybrid Encryption Chat (RSA + AES)". The interface includes input fields for recipient ("Bob" and "Alice"), a message input field ("Tulis pesan..."), and buttons for "Daftar (ganti kunci RSA)" (register/change RSA key), "Kirim Pesan" (Send Message), and "Lihat Pesan Masuk" (View Incoming Messages). Below the interface, a developer tools storage panel is open, showing the "Session storage" section under "LocalStorage" (highlighted with a red box). A specific entry for the key "privatekey" is highlighted with a red box and circled with number 2. The value of this key is a long string of characters, starting with "-----BEGIN RSA PRIVATE KEY-----". A red box also surrounds the beginning of this string, with circled number 3 above it. The right side of the storage panel displays the raw RSA private key data.

Gambar 1 Lokasi penyimpanan private key berada di Session Storage

Daftar

Hybrid Encryption Chat (RSA + AES)

Bob

Daftar (ganti kunci RSA) 1

Alice

Tulis pesan...

Kirim Pesan

Kirim Pesan

Hybrid Encryption Chat (RSA + AES)

Bob

Daftar (ganti kunci RSA)

Alice

Tulis pesan... 3

Kirim Pesan 4

Lihat Pesan Masuk

[2025-10-29 16:51:39] Dari Alice: Hai Bob
[29/10/2025, 16.52.03] Aku ke Alice: Hai Alice
[29/10/2025, 17.54.44] Aku ke Alice: Test

Network

register.php 2

Name Headers Payload Preview Response Initiator Timing

Request Payload View source

username: "Bob",
public_key: "-----BEGIN PUBLIC KEY-----\nMIGeMA0GCSqGSIb3DQEBAQAA4GMAD
username: "Bob"

Payload hanya ada JSON { username, public_key }. Tidak ada privatekey.

Network

register.php 5

Name Headers Payload Preview Response Initiator Timing Cookies

Request Payload View source

asal: "Bob", tujuan: "Alice",
asal: "Bob",
kunci: "W7gwvt35Z89MTQNl3lh5GNK1IASTyHGMj7Xx8Lwtp1WvfrTjRKCQrsRVnJunQz
pesan: "PG9u50zC4YrajIidq4q18s4YehoexAenVSS7009paw"
tujuan: "Alice"

Payload hanya { asal, tujuan, kunci, pesan }. Tidak ada privatekey.

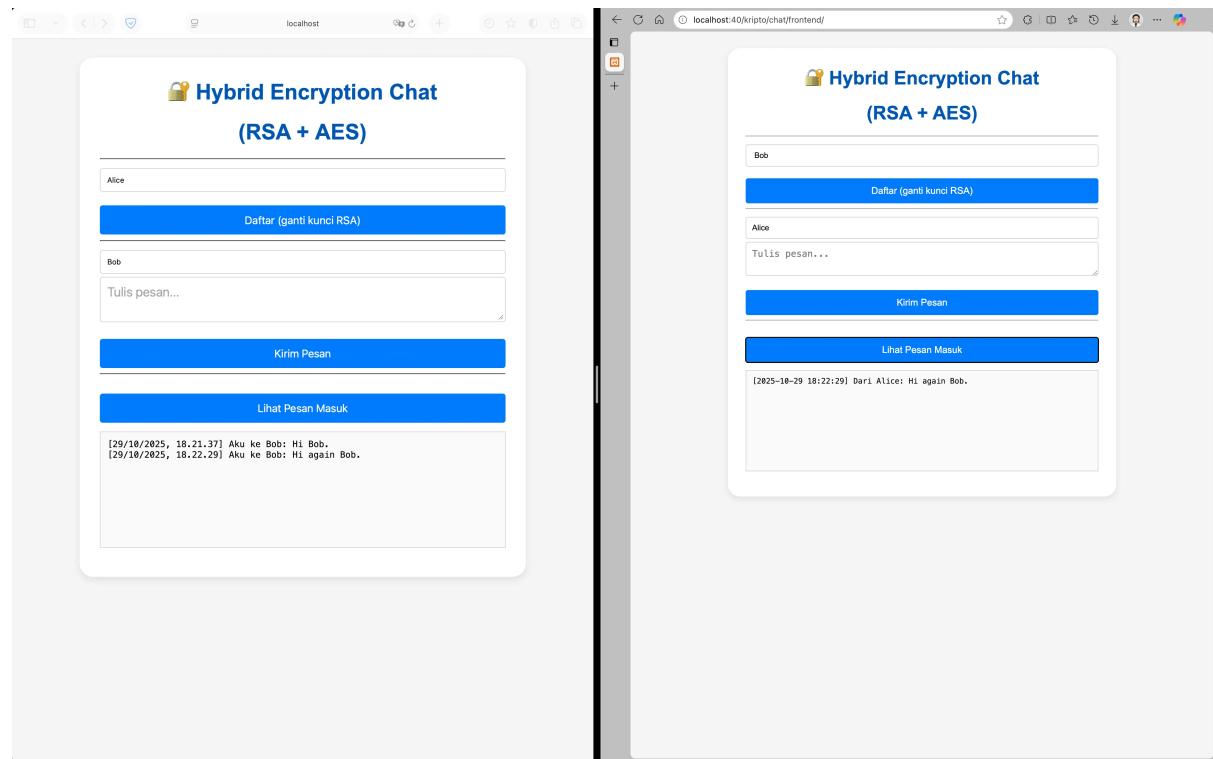
Gambar 2 Private Key tidak dikirim ke Server dilihat dari Payload register.php & simpan_pesan.php

Nomor 2

2. Jika laman di-refresh lalu nama pengirim dan penerima diisi sama seperti sebelum refresh, dan dikirim pesan baru, apakah masih bisa membaca pesan tersebut?

Masih bisa membaca pesan tersebut. Setelah refresh pada tab yang sama, Session Storage tetap ada, jadi private key masih tersedia dan pesan baru tetap bisa dibaca. Pesan hanya akan gagal dibaca jika pengguna menekan tombol “Daftar” lagi karena akan membuat pasangan kunci RSA baru yang tidak cocok dengan pesan lama atau membuka sesi baru seperti tab/jendela baru, yang menyebabkan Session Storage kosong.

Aplikasi menyimpan private key ke sessionStorage saat proses pendaftaran, kemudian mengambilnya kembali saat tombol “Lihat Pesan” ditekan untuk melakukan dekripsi session key RSA dan ciphertext AES menjadi plaintext.

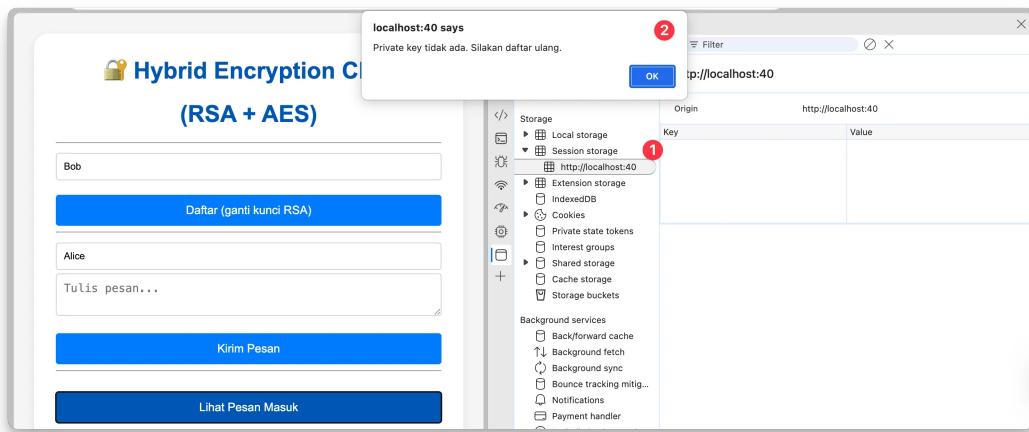


Gambar 3 Pesan yang dikirim Alice masuk ke Bob setelah Bob me-refresh browsernya, karena private key milik Bob masih tersimpan di Session Storage pada tab yang sama.

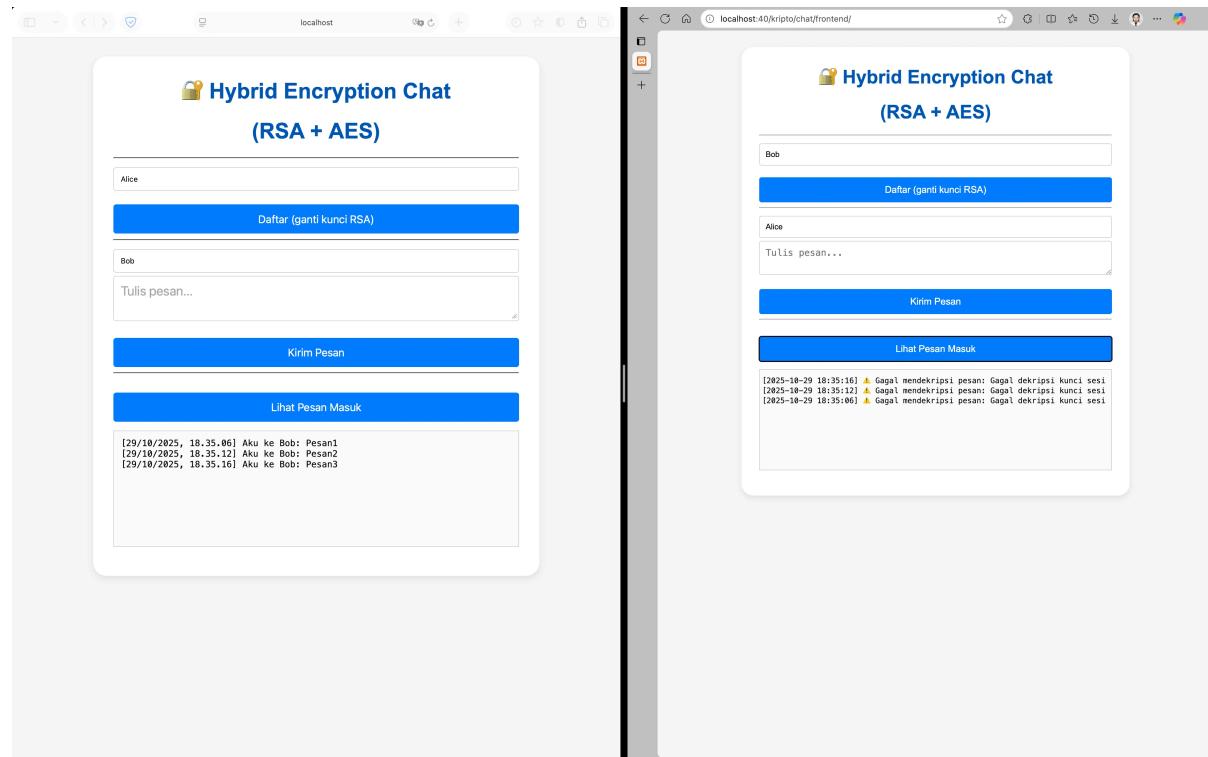
Nomor 3

3. Lakukan seperti Langkah 2 namun tab ditutup, lalu buka lagi.

Tidak bisa membaca sebelum mendaftarkan ulang. Menutup tab sama saja menghapus sessionStorage, jadi private key hilang. Saat tab baru dibuka, tombol “Lihat Pesan” akan gagal karena tidak ada private key. Setelah menekan “Daftar” (ganti kunci RSA) di tab baru, pengguna mendapat pasangan kunci baru dan bisa membaca hanya pesan yang dikirim setelah pengirim mengenkripsi dengan public key baru pengguna. Pesan lama tidak bisa didekripsi karena private key lama sudah hilang.



Gambar 4 Membuka laman yang sama di tab baru akan menghapus Session Storage beserta Private Key



Gambar 5 Pesan dari Alice ke Bob gagal didekripsi oleh Bob karena Bob membuka tab baru. Hal ini terjadi karena Session Storage browser dihapus saat tab ditutup, sehingga private key milik Bob tidak lagi tersedia.

Nomor 4

4. Buat tabel perbandingan sederhana untuk membandingkan karakteristik beberapa penyimpanan lokal: **LocalStorage**, **SessionStorage**, **IndexedDB**, dan **Cookie**!

Fitur	LocalStorage	SessionStorage	IndexedDB	Cookie
-------	--------------	----------------	-----------	--------

Model data	Kunci-nilai (string)	Kunci-nilai (string)	Basis data objek, indeks, transaksi	Kunci-nilai (string)
Persistensi	Tetap sampai dihapus	Bertahan selama tab/jendela aktif	Tetap sampai dihapus	Tergantung Expires/Max-Age
Kapasitas tipikal	~5–10 MB per origin	~5–10 MB per tab	Puluhan sampai ratusan MB per origin	≤4 KB per cookie
Akses dari JS	localStorage	sessionStorage	indexedDB	document.cookie (kecuali HttpOnly)
Visibilitas request	Tidak terkirim otomatis	Tidak terkirim otomatis	Tidak terkirim otomatis	Terkirim ke server pada setiap request ke domain/path yang cocok
Isolasi	Per origin	Per tab + origin	Per origin (database bernama)	Domain + subdomain + path + SameSite
Kinerja untuk data besar	Buruk	Buruk	Baik (streaming, cursor)	Sangat buruk
Transaksi/Query	Tidak ada	Tidak ada	Ada (ACID, index)	Tidak ada
Kadaluarsa bawaan	Tidak ada	Saat tab ditutup	Tidak ada	Atur dengan Expires/Max-Age
Keamanan tambahan	Hanya SOP + CSP	Hanya SOP + CSP	Hanya SOP + CSP	Atribut Secure, HttpOnly, SameSite
Cocok untuk	Preferensi kecil klien	State sementara per tab	Cache/offline, data terstruktur	Session token ringan, flag kecil

Nomor 5

5. Jika diinginkan privatekey bisa digunakan selama 7 hari meskipun browser/komputer dimatikan, maka private key sebaiknya disimpan di mana? Mengapa? Bagaimana agar setelah 7 hari private key tidak bisa digunakan lagi?

Private key sebaiknya disimpan di LocalStorage atau IndexedDB browser. Keduanya bersifat persistent, yang artinya data tetap tersimpan walaupun browser atau komputer dimatikan dan dinyalakan kembali. LocalStorage menyimpan data berbasis pasangan kunci nilai yang bertahan sampai dihapus manual atau oleh program dan IndexedDB memberikan penyimpanan terstruktur (database objek) yang mendukung obyek besar dan waktu kadaluwarsa dapat dikontrol melalui metadata. Karena keduanya tidak terhapus otomatis seperti SessionStorage, private key masih tersedia setelah restart, memungkinkan pengguna mendekripsi pesan dalam jangka panjang termasuk 7 hari.

Supaya private key hanya berlaku dalam 7 hari, tambahkan data waktu kadaluwarsa pada saat penyimpanan. Setiap kali kunci disimpan ke LocalStorage atau IndexedDB,

aplikasi juga mencatat waktu penyimpanan dan menghitung batas akhir masa aktif selama tujuh hari. Saat aplikasi dijalankan kembali, sistem akan memeriksa apakah waktu saat ini sudah melewati tanggal kedaluwarsa tersebut. Jika sudah lewat, maka kunci otomatis dihapus dari penyimpanan dan pengguna diminta untuk membuat kunci baru.

Nomor 6

6. Jika private key bisa digunakan dalam waktu yang lama, bagaimana cara mengamankannya? Jika private key dienkrip, kuncinya menggunakan apa?

Private key yang disimpan dalam jangka waktu lama harus diamankan supaya tidak dapat diakses oleh pihak lain secara *unauthorized*. Mekanisme yang dapat digunakan adalah enkripsi sisi klien (client-side encryption) sebelum kunci disimpan, yang artinya, private key tidak disimpan dalam bentuk teks asli, tetapi dalam bentuk terenkripsi yang hanya dapat didekripsi oleh pengguna yang asli.

Kunci enkripsi untuk mengamankan private key biasanya berasal dari faktor autentikasi pengguna, misalnya password, PIN, atau passphrase. Password tersebut tidak digunakan secara langsung, tetapi diproses melalui key-derivation function yang berfungsi mengubah input menjadi kunci enkripsi seperti PBKDF2, Argon2, atau scrypt untuk menghasilkan kunci simetris yang kuat. Hasil derivasi digunakan untuk mengenkripsi private key dengan algoritma simetris seperti AES-256.

Dengan mekanisme ini, jika penyimpanan lokal (seperti LocalStorage atau IndexedDB) dibaca oleh pihak luar, data yang ditemukan hanyalah ciphertext yang tidak berguna. Private key hanya dapat digunakan setelah pengguna memasukkan password yang benar untuk mendekripsinya di sisi klien.

Nomor 7

7. Dalam aplikasi chat yang dibuat, apakah session key (kunci simetris) yang digunakan untuk enkripsi tiap pesan yang dikirimkan sama? Apa tujuannya?

Session key yang digunakan untuk enkripsi tidak sama untuk setiap pesan yang dikirimkan. Hal tersebut dapat dilihat pada kode di index.html:

```
const sessionKey = CryptoJS.lib.WordArray.random(16);  
const iv = CryptoJS.lib.WordArray.random(16);
```

Setiap kali fungsi `kirimpesan()` dijalankan, sistem membuat session key dan IV baru secara acak. Tidak ada penyimpanan atau reuse dari session key sebelumnya, sehingga tiap pesan terenkripsi dengan kunci yang unik.

Tujuannya yaitu untuk menjaga kerahasiaan antar pesan (jika satu pesan dapat diretas, pesan lain tetap aman karena memiliki kunci yang berbeda), mencegah analisis

pola enkripsi (penggunaan kunci dan IV baru membuat ciphertext selalu berubah meskipun plaintext-nya sama), dan meningkatkan keamanan jangka panjang karena tidak ada session key yang digunakan kembali yang berarti tidak ada risiko kebocoran global yang berdampak ke seluruh percakapan.

Nomor 8

8. Apa kelemahan pertukaran kunci dengan RSA? Berikan cara yang lebih aman, serta jelaskan konsep kerjanya untuk aplikasi chat!

Kelemahan penggunaan RSA dalam pertukaran kunci terletak pada sifatnya yang statis dan tidak menyediakan kerahasiaan ke depan (forward secrecy) yang artinya, jika suatu saat private key milik salah satu pihak bocor, maka semua session key dan pesan yang pernah dienkripsi menggunakan public key tersebut bisa dibuka kembali. Selain itu, RSA juga memerlukan kunci berukuran besar untuk tetap aman dengan minimal 2048 bit, yang membuat proses enkripsi dan dekripsi lebih lambat dibandingkan algoritma pertukaran kunci yang modern.

Cara yang lebih aman untuk pertukaran kunci adalah menggunakan Diffie Hellman Ephemeral (DHE) atau versi eliptikanya Elliptic Curve Diffie Hellman Ephemeral (ECDHE). Keduanya termasuk metode key exchange yang dinamis. Dalam metode ini, setiap kali dua pengguna akan berkomunikasi, mereka tidak menggunakan kunci tetap, tetapi masing-masing menghasilkan kunci sementara (ephemeral) yang berbeda di setiap sesi.

Konsep kerja secara analogi Alice dan Bob, sebagai berikut:

- Saat Alice dan Bob akan bertukar pesan, masing-masing membuat kunci sementara sendiri.
- Mereka bertukar sebagian data kunci publik sementara tersebut melalui server.
- Dengan rumus matematika tertentu, baik Alice maupun Bob bisa menghitung kunci rahasia bersama (shared secret) tanpa pernah mengirimkannya lewat jaringan.
- Hasil perhitungan shared secret ini menjadi dasar untuk membuat session key AES yang akan digunakan untuk enkripsi pesan.

Keuntungan sistem ini adalah jika suatu kunci jangka panjang bocor, pesan-pesan lama tetap aman karena setiap sesi komunikasi memakai kunci sementara yang berbeda. Untuk aplikasi chat, konsep ECDHE cocok karena ringan, cepat, dan memberikan keamanan berlapis tanpa membebani kinerja browser atau server.

Informasi Proyek

Struktur Direktori

```

kripto
└── chat
    ├── backend
    │   ├── ambil_public_key.php
    │   ├── baca_pesan.php
    │   ├── db.php
    │   ├── register.php
    │   └── simpan_pesan.php
    └── frontend
        ├── index.html
        └── style
            └── stylekoe.css

```

Basis Data

```

CREATE DATABASE uts;
USE uts;

CREATE TABLE `chatkeys` (
  `Nama` varchar(15) NOT NULL,
  `Kuncipublik` varchar(4096) NOT NULL,
  `Timestamp` datetime DEFAULT CURRENT_TIMESTAMP,
  PRIMARY KEY (`Nama`)
);

CREATE TABLE `chatmessages` (
  `Id` int(11) NOT NULL AUTO_INCREMENT PRIMARY KEY,
  `Asal` varchar(15) NOT NULL,
  `Tujuan` varchar(15) NOT NULL,
  `Kunci` varchar(4096) NOT NULL COMMENT 'Session key yang dienkripsi dengan Public Key RSA',
  `Pesel` text NOT NULL COMMENT 'Pesel terenkripsi',
  `Timestamp` datetime DEFAULT CURRENT_TIMESTAMP,
  `Dibaca` tinyint(1) NOT NULL DEFAULT 0
);

```

Back-end

DB Connector

```

<?php
$conn = new mysqli("localhost", "root", "", "uts");

if ($conn->connect_error) {
    die(json_encode(["error" => "Koneksi gagal: " . $conn->connect_error]));
}
?>

```

Register

```

<?php
require 'db.php';

$data = json_decode(file_get_contents("php://input"), true);
$username = $data['username'];
$public_key = $data['public_key'];

// Cek apakah nama sudah ada
$stmt = $conn->prepare("SELECT * FROM chatkeys WHERE Nama = ?");
$stmt->bind_param("s", $username);
$stmt->execute();

```

```

$result = $stmt->get_result();

if ($result->num_rows > 0) {
    // Jika sudah ada > update kunci dan timestamp
    $stmt->close();
    $stmt = $conn->prepare("UPDATE chatkeys SET Kuncipublik = ?, Timestamp = NOW() WHERE Nama = ?");
    $stmt->bind_param("ss", $public_key, $username);
    $stmt->execute();
} else {
    $stmt->close();
    $stmt = $conn->prepare("INSERT INTO chatkeys (Nama, Kuncipublik, Timestamp) VALUES (?, ?, NOW())");
    $stmt->bind_param("ss", $username, $public_key);
    $stmt->execute();
}

echo json_encode(["status" => "success"]);
$stmt->close();
$conn->close();
?>

```

Get Public Key

```

<?php
require 'db.php';

$username = $_GET['username'] ?? '';

$stmt = $conn->prepare("SELECT Kuncipublik FROM chatkeys WHERE Nama = ?");
$stmt->bind_param("s", $username);
$stmt->execute();

$result = $stmt->get_result();

if ($result->num_rows === 0) {
    echo json_encode(["error" => "Pengguna tidak ditemukan."]);
} else {
    $row = $result->fetch_assoc();
    echo json_encode([
        "username" => $username,
        "public_key" => $row['Kuncipublik']
    ]);
}

$stmt->close();
$conn->close();
?>

```

Save Message

```

<?php
require 'db.php';

// Ambil data JSON dari body request
$data = json_decode(file_get_contents("php://input"), true);
$asal = $data['asal'] ?? '';
$tujuan = $data['tujuan'] ?? '';
$kunci = $data['kunci'] ?? '';
$pesan = $data['pesan'] ?? '';

// Simpan pesan ke tabel
$stmt = $conn->prepare("INSERT INTO chatmessages (Asal, Tujuan, Kunci, Pesan, Timestamp) VALUES (?, ?, ?, ?, NOW())");
$stmt->bind_param("ssss", $asal, $tujuan, $kunci, $pesan);

if ($stmt->execute()) {
    echo "Pesan berhasil disimpan.";
} else {
    echo "Gagal menyimpan pesan: " . $stmt->error;
}

```

```

}

$stmt->close();
$conn->close();
?>

```

Read Message

```

<?php
require 'db.php';

$username = $_GET['username'] ?? '';

// Ambil semua pesan yang ditujukan ke pengguna
$stmt = $conn->prepare("SELECT Asal, Kunci, Pesan, Timestamp FROM chatmessages WHERE Tujuan = ? AND Dibaca = false ORDER BY Timestamp DESC");
$stmt->bind_param("s", $username);
$stmt->execute();

$result = $stmt->get_result();
$messages = [];
while ($row = $result->fetch_assoc()) {
    $messages[] = [
        "asal" => $row["Asal"],
        "kunci" => $row["Kunci"],
        "pesan" => $row["Pesan"],
        "timestamp" => $row["Timestamp"]
    ];
}

echo json_encode($messages);

// ubah status
$stmt = $conn->prepare("UPDATE chatmessages SET Dibaca = true WHERE Tujuan = ? AND Dibaca = false");
$stmt->bind_param("s", $username);
$stmt->execute();

$stmt->close();
$conn->close();
?>

```

Front-end

CSS

```

body { font-family: Arial, sans-serif; padding: 20px; background-color: #f4f4f4; }
.container { background-color: #fff; padding: 30px; border-radius: 20px; box-shadow: 0 4px 8px rgba(0, 0, 0, 0.1); max-width: 600px; margin: auto; }
h1 { text-align: center; color: #0056b3; margin-top: 0px; margin-bottom: 20px; }
label { display: block; margin-top: 15px; font-weight: bold; }
input, select, textarea { width: 100%; padding: 10px; margin-top: 5px; border: 1px solid #ccc; border-radius: 4px; box-sizing: border-box; }
textarea { resize: vertical; font-size: larger; }
button { background-color: #007bff; color: white; padding: 12px 20px; border: none; border-radius: 4px; cursor: pointer; margin-top: 20px; width: 100%; font-size: 16px; }
button:hover { background-color: #0056b3; }
#chat-box { height: 150px !important; overflow-y: auto; border: 1px solid #ccc; padding: 10px; background: #fafafa; }

```

Indeks

```

<!DOCTYPE html>
<html lang="id">
<head>
    <meta charset="UTF-8" />
    <title>Hybrid Encryption Chat</title>
    <script src="https://cdn.jsdelivr.net/npm/jsencrypt@3.3.2/bin/jsencrypt.min.js"></script>

```

```

<script src="https://cdn.jsdelivr.net/npm/crypto-js@4.1.1/crypto-js.min.js"></script>
<link rel="stylesheet" href="./style/stylekoe.css" />
</head>
<body>
  <div class="container">
    <h1>Hybrid Encryption Chat </h1><h1>(RSA + AES)</h1>
    <hr>
    <input id="username" placeholder="Nama Anda" />
    <button type="button" onclick="daftar()">Daftar (ganti kunci RSA)</button>
    <hr>
    <input id="recipient" placeholder="Kirim ke (nama penerima)" />
    <textarea id="message" placeholder="Tulis pesan..."></textarea>
    <button type="button" onclick="kirimpesan()>Kirim Pesan</button>
    <hr>
    <button type="button" onclick="lihatpesan()>Lihat Pesan Masuk</button>
    <pre id="chat-box"></pre>
  </div>

<script>
// ❶ Registrasi dan buat pasangan kunci
function daftar() {
  const nama = document.getElementById("username").value.trim();
  if (!nama) return alert("Masukkan nama Anda dulu!");

  const rsa = new JSEncrypt({ default_key_size: 1024 });
  rsa.getKey();

  const publicKey = rsa.getPublicKey();
  sessionStorage.setItem("privatekey", rsa.getPrivateKey());

  fetch("../backend/register.php", {
    method: "POST",
    headers: { "Content-Type": "application/json" },
    body: JSON.stringify({ username: nama, public_key: publicKey })
  })
  .then(r => r.text())
  .then(() => alert("✓ Registrasi berhasil!"))
  .catch(() => alert("✗ Gagal registrasi."));
}

// ❷ Kirim pesan terenkripsi (Hybrid AES + RSA)
async function kirimpesan() {
  const pengirim = document.getElementById("username").value.trim();
  const penerima = document.getElementById("recipient").value.trim();
  const teks = document.getElementById("message").value.trim();

  if (!pengirim || !penerima || !teks) return alert("Isi semua kolom.");

  const r = await
fetch(`../backend/ambil_public_key.php?username=${encodeURIComponent(penerima)}`);
  const data = await r.json();
  if (data.error) {
    alert("✗ Terjadi error: " + data.error);
    return;
  }
  if (!data.public_key) return alert("Public key penerima tidak ditemukan.");

  // ❸ Enkripsi pesan dengan session key AES dan IV
  const sessionKey = CryptoJS.lib.WordArray.random(16);
  const iv = CryptoJS.lib.WordArray.random(16);
  const encrypted = CryptoJS.AES.encrypt(teks, sessionKey, { iv });
  const cipherText = iv.concat(encrypted.ciphertext).toString(CryptoJS.enc.Base64);

  // ❹ Enkripsi session key dengan RSA
  const rsa = new JSEncrypt();
  rsa.setPublicKey(data.public_key);
  const encryptedSessionKey = rsa.encrypt(CryptoJS.enc.Base64.stringify(sessionKey));
}

```

```

    const payload = { asal: pengirim, tujuan: penerima, kunci: encryptedSessionKey, pesan: cipherText };

    await fetch("../backend/simpan_pesan.php", {
        method: "POST",
        headers: { "Content-Type": "application/json" },
        body: JSON.stringify(payload)
    });

    const now = new Date();
    const chatBox = document.getElementById("chat-box");
    chatBox.textContent += `[${now.toLocaleString('id-ID', {hour12: false})}] Aku ke ${penerima}: ${teks}\n`;
    chatBox.scrollTop = chatBox.scrollHeight;
    document.getElementById("message").value = "";
}

// ③ Lihat & dekripsi pesan
async function lihatPesan() {
    const nama = document.getElementById("username").value.trim();
    const privateKey = sessionStorage.getItem("privatekey");
    if (!nama) return alert("Masukkan nama Anda dulu!");
    if (!privateKey) return alert("Private key tidak ada. Silakan daftar ulang.");

    const r = await fetch(`../backend/baca_pesan.php?username=${encodeURIComponent(nama)}`);
    const pesanMasuk = await r.json();

    const rsa = new JSEncrypt();
    rsa.setPrivateKey(privateKey);

    let hasil = "";
    for (const p of pesanMasuk) {
        try {
            // 🔒 Dekrip session key
            const sessionKeyBase64 = rsa.decrypt(p.kunci);
            if (!sessionKeyBase64) throw new Error("Gagal dekripsi kunci sesi");
            const sessionKey = CryptoJS.enc.Base64.parse(sessionKeyBase64);

            // 🎨 Pisahkan IV dan ciphertext dari pesan
            const data = CryptoJS.enc.Base64.parse(p.pesan);
            const iv = CryptoJS.lib.WordArray.create(data.words.slice(0, 4), 16);
            const ciphertext = CryptoJS.lib.WordArray.create(data.words.slice(4), data.sigBytes - 16);

            // ✎ Dekrip pesan AES
            const decrypted = CryptoJS.AES.decrypt({ ciphertext: ciphertext }, sessionKey, { iv });
            const plainText = decrypted.toString(CryptoJS.enc.Utf8);

            if (!plainText) throw new Error("Plaintext kosong");

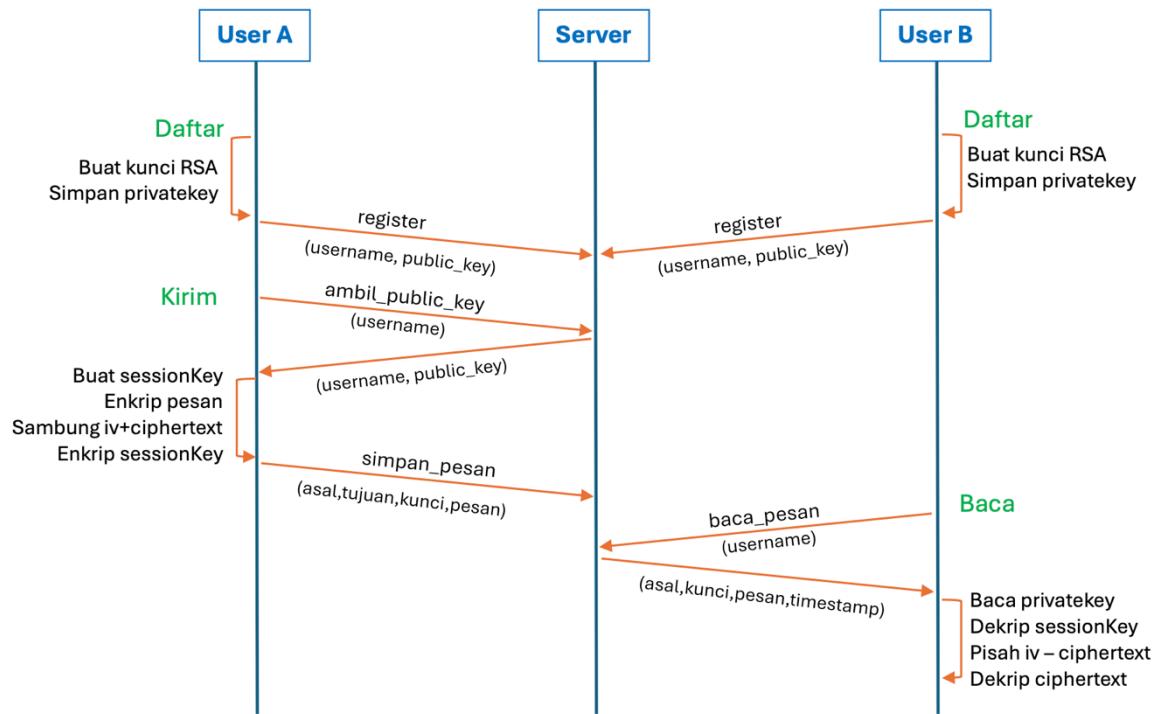
            hasil += `[${p.timestamp}] Dari ${p.asal}: ${plainText}\n`;
        } catch (e) {
            hasil += `[${p.timestamp}] ⚠️ Gagal mendekripsi pesan: ${e.message}\n`;
        }
    }

    const chatBox = document.getElementById("chat-box");
    chatBox.textContent = hasil || "Belum ada pesan.\n";
    chatBox.scrollTop = chatBox.scrollHeight;
}
</script>

</body>
</html>

```

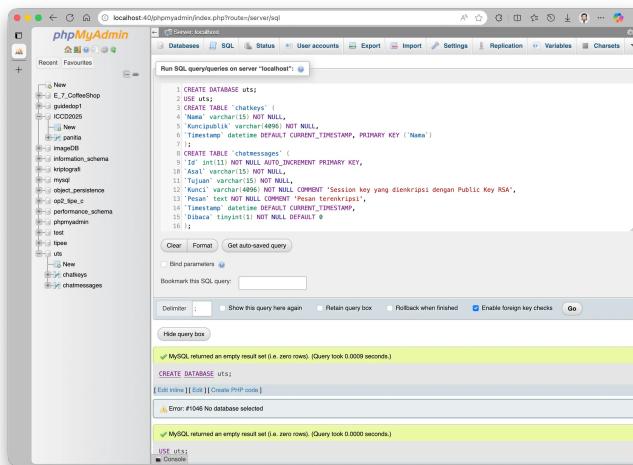
Sequence Diagram



Gambar 6 Sequence diagram dari proyek

Pengujian

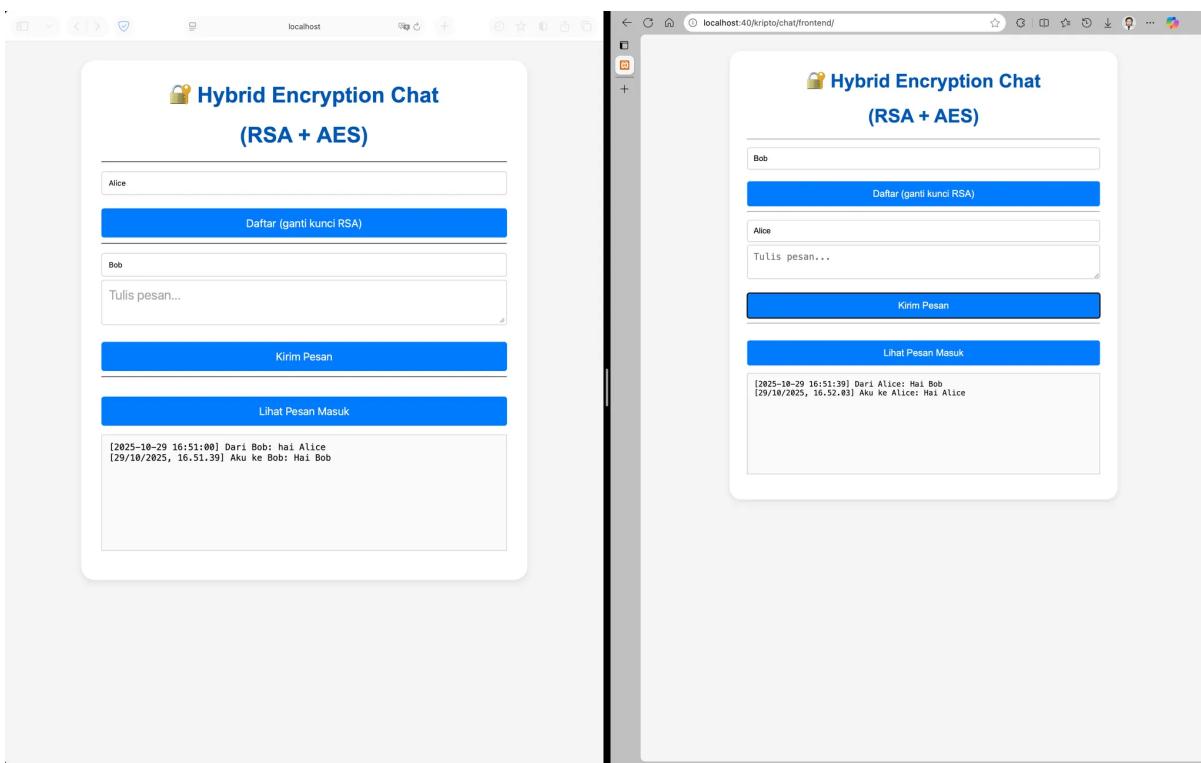
- Pastikan web server (Apache) dan basis data (MySQL/MariaDB) sudah dijalankan, tabel-tabel di basis data sudah ada, dan file-file sudah ditulis dan disimpan di folder yang benar.



Gambar 7 Pembuatan struktur basis data

- Buka web Browser (tab baru), akses: <http://localhost/kripto/chat/frontend>

3. Masukkan nama pengirim: Alice, lalu tekan button "Daftar".
4. Masukkan orang yang diajak komunikasi: Bob.
5. Lakukan Langkah 2-5 dengan pengirim: Bob, Tujuan: Alice (kalau bisa beda web browser).
6. Alice kirim pesan "Hai Bob" ke Bob.
7. Bob menekan tombol "Lihat Pesan Masuk".
8. Bob membalas pesan "hai Alice" ke Alice.
9. Alice menekan tombol "Lihat Pesan Masuk".
10. Pastikan chat berjalan dengan baik.



Gambar 8 Percakapan Alice dan Bob direalisasikan dengan dua browser berbeda