TUGAS BESAR

MATA KULIAH KRIPTOGRAFI DAN KEAMANAN SIBER

Sistem Absensi Berbasis RFID Dengan Enkripsi DES Untuk Keamanan Data Waktu Kehadiran

Oleh Kelompok 7:

Bambang Istijab	105222007
Ni Putu Merta Bhuana Ningsih	105222008
Jihan Fadila	105222022
Senopati Baruna Pasha	105222023

PROGRAM STUDI ILMU KOMPUTER

FAKULTAS SAINS DAN ILMU KOMPUTER

UNIVERSITAS PERTAMINA

2025

BAB I
PENDAHULUAN

1.1 Latar Belakang

Kejadian kebocoran data di Indonesia tidak menjadi hal yang mengejutkan lagi bagi warga Indonesia karena sering terjadi. Kasus besar seperti peretasan Bjorka pada data Peduli Lindungi, dengan mendapatkan 3,2 miliar data pengguna aplikasi Peduli Lindungi. Pemerintah sampai diminta untuk tidak saling melempar tanggung jawab atas kejadian itu [1].

Fenomena kebocoran data ini menggarisbawahi urgensi keamanan informasi dalam setiap aspek kehidupan digital, termasuk dalam bidang Internet of Things (IoT). Perangkat IoT seringkali mengumpulkan dan mentransmisikan data sensitif, termasuk data terkait waktu, id pengguna, password akun, dan lainnya. Terutama data waktu yang dapat krusial untuk riwayat, aktivitas, dan data log waktu yang dapat disadap dan disalahgunakan. Keamanan data dalam IoT menjadi perhatian utama mengingat potensi ancaman siber seperti penyadapan, modifikasi data, atau penolakan layanan.

Dalam konsep sistem kehadiran atau absensi, informasi mengenai waktu absen sangat penting dan seringkali membutuhkan perlindungan terhadap data pribadi. Contohnya kasus kerahasiaan waktu tap terakhir seorang petinggi pemerintah untuk menghindari kedatangan media, atau seorang yang terlibat dalam dunia bisnis atau politik yang perlu menyembunyikan keberadaannya dari musuh, seperti mafia. Dengan waktu absen yang terenkripsi, informasi mengenai keberadaan seseorang tidak akan terdeteksi secara *real-time*, yang secara signifikan meningkatkan tingkat privasi dan keamanan. Oleh karena itu, modifikasi data waktu secara ilegal dapat mengganggu integritas sistem, menyebabkan kesalahan dalam pencatatan peristiwa, atau bahkan memfasilitasi serangan jika data waktu digunakan sebagai bagian dari mekanisme otentikasi atau integritas pesan.

Mengingat keterbatasan sumber daya komputasi dan energi pada perangkat IoT, pemilihan algoritma keamanan harus mempertimbangkan efisiensi. Proyek ini berfokus pada implementasi keamanan untuk data waktu yang berasal dari perangkat IoT sebelum disimpan atau ditransmisikan. Dengan mengaplikasikan teknik enkripsi, integritas dan kerahasiaan data waktu dapat dijaga, memastikan bahwa informasi krusial ini tetap otentik dan tidak dapat dimanipulasi oleh pihak yang tidak berwenang.

1.2 Rumusan Masalah

Latar belakang di atas ditarik beberapa rumusan masalah, seperti di bawah ini:

1. Bagaimana cara mengimplementasikan proses enkripsi data waktu yang diperoleh dari perangkat IoT (RFID) sebelum data tersebut disimpan ke dalam basis data?

2. Bagaimana sistem dapat mengambil data waktu terenkripsi dari basis data, kemudian mendekripsinya kembali ke format UNIX timestamp asli, dan menampilkannya sebagai objek datetime yang dapat dibaca di antarmuka web?
3. Bagaimana sistem dapat memverifikasi integritas dan keaslian data waktu setelah melalui seluruh proses enkripsi, penyimpanan, pengambilan, dan dekripsi?

1.3 Tujuan Proyek

Dalam proyek terdapat tujuan yang dapat dilihat di bawah ini.

1. Membangun sistem yang mampu mengkonversi data waktu yang diterima dari perangkat IoT (RFID) menjadi UNIX timestamp, mengenkripsi timestamp tersebut menggunakan algoritma DES, dan kemudian mengirimkan data waktu terenkripsi ke basis data.
2. Mengembangkan fungsi yang dapat mengambil data waktu terenkripsi dari basis data, mendekripsinya untuk mengembalikan UNIX timestamp asli, dan selanjutnya menampilkannya sebagai objek datetime yang dapat dibaca pada antarmuka web.
3. Memastikan dan memverifikasi integritas serta keaslian data waktu setelah melalui seluruh alur proses mulai dari enkripsi, penyimpanan di basis data, pengambilan, hingga dekripsi dan penampilan.

BAB II
TINJAUAN PUSTAKA

2.1 Radio Frequency Identification (RFID)

Teknologi Radio Frequency Identification (RFID) telah berkembang pesat dalam beberapa tahun terakhir, bertransformasi dari alat khusus menjadi solusi yang digunakan secara luas di berbagai sektor. Awalnya dikembangkan untuk keperluan militer dan keamanan, RFID kini banyak digunakan di sektor ritel, logistik, dan perawatan kesehatan. Keunggulan utama RFID dibandingkan dengan sistem kode batang tradisional adalah kemampuannya untuk membaca data tanpa memerlukan garis pandang langsung, sehingga memungkinkan pelacakan inventaris yang lebih cepat, akurat, dan mengurangi kesalahan manusia. Sistem RFID modern, terutama tag pasif, terjangkau, kecil, dan efisien, serta menawarkan solusi dengan perawatan rendah dan masa operasional yang lebih lama. Mereka bekerja dengan menggunakan gelombang radio untuk mentransmisikan data antara pembaca dan tag, yang tidak hanya menyimpan data identifikasi tetapi juga informasi tambahan seperti spesifikasi produk dan kondisi lingkungan [2].

Meskipun memiliki banyak keuntungan, teknologi RFID menghadapi tantangan terkait privasi dan keamanan. Karena tag RFID tidak memerlukan garis pandang untuk dipindai, ada kekhawatiran tentang akses tidak sah terhadap informasi sensitif, karena pengguna dapat menangkap data dari tag tanpa sepengetahuan pengguna. Untuk mengurangi risiko ini, protokol enkripsi dan privasi yang lebih canggih telah dikembangkan, termasuk saklar penghentian yang menonaktifkan tag setelah pembelian dan enkripsi untuk transmisi data. Selain itu, integrasi RFID dengan teknologi lain seperti jaringan sensor nirkabel dan perangkat seluler membuka kemungkinan aplikasi baru, seperti pelacakan kondisi lingkungan atau meningkatkan manajemen rantai pasokan. Namun demikian, potensi penyalahgunaan teknologi ini dalam melacak individu atau untuk penyadapan data tetap menjadi kekhawatiran utama yang perlu diatasi untuk adopsi yang lebih luas [3].

2.2 Mel Frequency Cepstral Coefficients (MFCC)

Mel Frequency Cepstral Coefficients (MFCC) merupakan salah satu teknik paling umum dan efisien dalam pemrosesan sinyal, terutama digunakan dalam pengenalan suara dan identifikasi pembicara. Teknik ini didasarkan pada cara manusia mendengar dan memahami frekuensi suara, yakni dengan memetakan spektrum frekuensi ke dalam skala Mel yang lebih sesuai dengan persepsi pendengaran manusia. Proses penghitungan MFCC dimulai dengan mengubah sinyal input menjadi sinyal satu dimensi (1D), diikuti dengan framing dan blocking untuk memecah sinyal menjadi bagian-bagian kecil yang lebih mudah dianalisis. Setelah itu, dilakukan proses windowing, diikuti dengan transformasi Fourier cepat (FFT) untuk mengubah sinyal dari domain waktu ke domain frekuensi. Selanjutnya, filter bank Mel digunakan untuk mengestimasi energi pada setiap frekuensi, dan akhirnya, koefisien MFCC dihitung menggunakan Transformasi Kosinus Diskrit (DCT) [4].

MFCC telah digunakan dalam berbagai aplikasi, terutama di bidang pengenalan suara, karena kemampuannya yang efektif dalam mengekstraksi fitur dari sinyal suara yang bervariasi. Teknik ini tidak hanya terbatas pada pengenalan suara, tetapi juga mulai diterapkan dalam pengolahan gambar, seperti pengenalan gerakan tangan dan identifikasi citra satelit. Dalam konteks pengenalan gerakan tangan, misalnya, gambar tangan diubah menjadi sinyal 1D, kemudian fitur diekstraksi menggunakan MFCC sebelum dilakukan proses klasifikasi menggunakan metode seperti Support Vector Machine (SVM). Meskipun teknik ini telah menunjukkan akurasi yang baik, penelitian lebih lanjut diperlukan untuk mengurangi tingkat kesalahan klasifikasi, serta mengintegrasikan MFCC dengan teknik lain guna meningkatkan performa dalam berbagai aplikasi pengolahan citra [5].

2.3 Algoritma DES

Data Encryption Standard (DES) adalah algoritma kriptografi simetris berbasis blok yang mengenkripsi data dalam blok 64-bit. Meskipun panjang kunci DES adalah 64-bit, hanya 56 bit yang digunakan secara efektif, karena setiap bit kedelapan digunakan untuk pemeriksaan paritas. Proses enkripsi dilakukan dalam 16 putaran (rounds), masing-masing menggunakan subkunci 48-bit yang dihasilkan dari kunci utama melalui transformasi tertentu. DES mengadopsi struktur Feistel, yang menggabungkan operasi substitusi dan transposisi untuk menciptakan kebingungan dan penyebaran data. Meskipun DES pernah menjadi standar enkripsi yang luas digunakan, kelemahan utamanya terletak pada panjang kunci yang relatif pendek, membuatnya rentan terhadap serangan brute-force. Oleh karena itu, DES telah digantikan oleh algoritma enkripsi yang lebih kuat seperti Advanced Encryption Standard (AES) [6].

2.4 MySQL

MySQL dapat dipandang sebagai kumpulan perangkat lunak klien dan server yang ekstensif dan seringkali kompleks untuk menyimpan serta mengambil data. Bagi pengguna baru MySQL atau bahkan SQL secara keseluruhan, kedalaman dan banyaknya opsi yang terlibat dalam tugas basis data yang tampak sederhana dapat terasa membebani. Oleh karena itu, pemahaman konsep dasar yang sering ditemui oleh pengembang maupun administrator sangat diperlukan untuk membantu pengguna merasa nyaman dengan MySQL dan dapat melakukan tugas-tugas harian yang umum [7].

BAB III
HASIL DAN PEMBAHASAN

3.1 Gambaran Umum Sistem

Sistem absensi berbasis IoT ini dirancang untuk mencatat kehadiran seseorang disuatu instansi secara otomatis menggunakan UID (Unique Identifier) yang terenkripsi. Sistem ini terdiri dari dua komponen utama: modul enkripsi/dekripsi DES (Data Encryption Standard) yang diimplementasikan dalam des.py, dan aplikasi web Flask sebagai antarmuka pengguna serta pengelola data yang diimplementasikan dalam app2.py. Data absensi disimpan dalam database MySQL dan juga dicatat ke dalam file CSV untuk redundansi.

3.2 Implementasi Algoritma DES

Modul des.py mengimplementasikan algoritma kriptografi simetris DES. Ini adalah inti dari keamanan data absensi, memastikan bahwa *timestamp* kehadiran seseorang disuatu instansidisimpan dalam bentuk terenkripsi. Source code dapat diakses pada link berikut ini : Link.

3.2.1 Fungsi Dasar Kriptogarfi DES

Beberapa fungsi dasar yang digunakan dalam implementasi DES meliputi:

- permute(bits, table): Fungsi ini melakukan permutasi bit berdasarkan tabel permutasi yang diberikan.
- left_shift(bits, n): Fungsi ini melakukan *left circular shift* pada urutan bit sebanyak n posisi.

- xor(a, b): Fungsi ini melakukan operasi XOR bit-demi-bit antara dua urutan bit.

3.2.2 Tabel dan Konstanta DES

Algoritma DES sangat bergantung pada serangkaian tabel dan konstanta yang telah ditentukan. Tabel-tabel ini didefinisikan dalam des.py dan digunakan dalam berbagai tahapan enkripsi dan dekripsi:

- PC1 (Permuted Choice 1): Digunakan untuk memilih 56 bit dari kunci 64-bit awal dan melakukan permutasi.
- PC2 (Permuted Choice 2): Digunakan untuk memilih 48 bit dari 56 bit setelah pergeseran untuk menghasilkan *subkey*.
- IP (Initial Permutation): Permutasi awal yang diterapkan pada *plaintext* 64-bit.
- IP_INV (Inverse Initial Permutation): Permutasi invers dari IP, diterapkan pada akhir proses enkripsi/dekripsi.
- E_BOX (Expansion Permutation): Mengembangkan blok 32-bit menjadi 48-bit untuk operasi XOR dengan *subkey*.

- P_BOX (Permutation Box): Melakukan permutasi pada output dari S-Box.
- ROTATION_SCHEDULE: Menentukan jumlah *left shifts* untuk setiap putaran dalam pembuatan *subkey*.
- S_BOXES (Substitution Boxes): Kumpulan 8 S-Box yang melakukan substitusi non-linear, inti dari keamanan DES.

3.2.3 Pembangkitan Subkunci

Fungsi generate_subkeys(key_64bit) bertanggung jawab untuk menghasilkan 16 *subkey* 48-bit dari kunci 64-bit utama. Proses ini melibatkan permutasi awal menggunakan PC1, kemudian membagi kunci menjadi dua bagian (C dan D). Bagian C dan D kemudian di-*left shift* sesuai dengan ROTATION_SCHEDULE di setiap putaran, dan digabungkan kembali sebelum permutasi menggunakan PC2 untuk menghasilkan *subkey*.

3.2.4 Fungsi Feistel

Fungsi feistel(R, K) mengimplementasikan inti dari struktur Feistel dalam DES. Ini menerima blok kanan (R) dari data dan *subkey* (K) untuk putaran saat ini. Langkah-langkahnya meliputi:

1. Ekspansi R menjadi 48 bit menggunakan E_BOX.
2. Hasil ekspansi di-XOR dengan *subkey* K.
3. Hasil XOR kemudian dilewatkan melalui 8 S-Box untuk substitusi non-linear.
4. Output dari S-Box kemudian di-permutasi menggunakan P_BOX.

3.2.5 Proses Enkripsi dan Dekripsi

- **Enkripsi**: Fungsi des_encrypt(plain_64bit, subkeys) mengenkripsi *plaintext* 64-bit. Ini dimulai dengan IP, membagi data menjadi L dan R, kemudian melakukan 16 putaran Feistel. Dalam setiap putaran, L menjadi R baru, dan R baru dihitung dengan XOR L lama dengan output fungsi Feistel dari R lama. Pada putaran terakhir, L dan R tidak ditukar. Hasil akhirnya di-*inverse permute* dengan IP_INV.
- **Dekripsi**: Fungsi des_decrypt(ciphertext_bin, subkeys) mendekripsi *ciphertext* biner. Prosesnya serupa dengan enkripsi, tetapi *subkeys* diterapkan dalam urutan terbalik (subkeys[15 - i]) untuk membalikkan operasi enkripsi.

3.3 Implementasi Aplikasi Web Flask

Aplikasi app2.py menyediakan antarmuka web untuk sistem absensi dan mengelola interaksi dengan database serta operasi enkripsi/dekripsi.

3.3.1 Inisialisasi dan Konfigurasi

- Aplikasi Flask diinisialisasi.
- Kunci DES (key = '0011000100110010001101000011010100110110001101110011100 0') didefinisikan dan *subkeys* dihasilkan saat aplikasi dimulai. Kunci ini adalah representasi biner dari string "12345678".
- Koneksi ke database MySQL (sistem_absensi_iot) diatur.

3.3.2 Rute Utama (/)

Rute “/” menampilkan *dashboard* absensi seseorang disuatu instansi. Halaman ini dibangun menggunakan render_template_string dan menyertakan JavaScript untuk:

- Mengambil data absensi secara berkala (setiap 1 detik) dari /data.
- Memperbarui tabel absensi dan statistik (total seseorang disuatu instansi, total masuk, total keluar).
- Menyediakan fungsionalitas pencarian nama seseorang disuatu instansi.
- Menampilkan *alert* SweetAlert2 jika ada pesan dari *backend* (misalnya, "Masih jam sekolah" atau "Silakan tunggu sebentar sebelum tap lagi").

3.3.3 Rute Absensi (/absen)

Rute /absen adalah *endpoint* POST yang menerima data absensi dari perangkat IoT.

- Menerima uid dan nama seseorang disuatu instansi dari *request* JSON.
- Mengambil *timestamp* saat ini dan mengubahnya menjadi format biner 64-bit.
- *Timestamp* biner ini kemudian dienkripsi menggunakan fungsi des_encrypt.
- Pengecekan Tap Ganda: Sebelum menyimpan data, sistem memeriksa apakah UID yang sama telah melakukan *tap* dalam 60 detik terakhir. Ini dilakukan dengan mengambil data *tap* terakhir dari database, mendekripsi *timestamp*-nya, dan membandingkan waktu. Jika *tap* terlalu cepat, *alert* akan dikirim kembali ke *frontend*.
- Data absensi (UID, nama, *timestamp* terenkripsi) kemudian disimpan ke tabel kripto di database MySQL.
- Sebagai tambahan, data absensi juga ditulis ke file kripto.csv.
- Sistem memiliki logika untuk menentukan status absensi (masuk, terlambat, pulang) berdasarkan waktu saat ini, namun pada implementasi /absen yang diberikan, status ini belum secara eksplisit disimpan ke database atau CSV. Variabel jam_masuk_awal, jam_masuk_batas, dan jam_pulang didefinisikan, namun tidak digunakan dalam logika penyimpanan data saat ini.

3.3.4 Rute Data (/data)

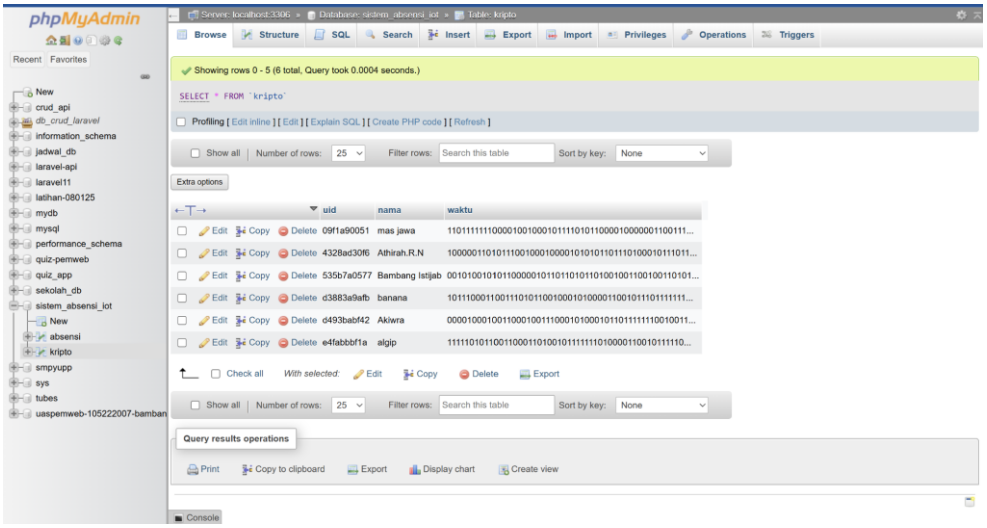
Rute /data adalah *endpoint* GET yang menyediakan data absensi kepada *frontend*.

- Mengambil semua data absensi (waktu terenkripsi, UID, nama) dari tabel kripto di database MySQL.
- Untuk setiap baris data yang diambil:
    - *Timestamp* terenkripsi didekripsi menggunakan des_decrypt.
    - *Timestamp* biner yang telah didekripsi dikonversi kembali ke *timestamp* Unix integer, kemudian menjadi objek datetime yang dapat dibaca manusia.
- Data yang telah didekripsi dan diformat dikirimkan sebagai respons JSON ke *frontend*.

3.4 Output

a. Database

Pada database, waktu tapping yang disimpan merupakan hasil enkripsi, setelah melalui semua tahapan dalam rute absensi, dengan mengenkripsi timestamp terlebih dahulu sebelum diinputkan ke database.



Gambar 3.4.1. Output di database

Adapun hasil tahapan enkripsi dijabarkan sebagai berikut.
timestamp yang diinputkan sistem diubah dalam bentuk unix timestmap, misalnya timestamp: 2025-06-24 12:07:23
timestamp kemudian diubah menjadi unix timestamp: 1750741643
kemudian unix timestamp diubah menjadi binary sebagai input: 00000000000000000000000000000000011010000101101000110010100010

Sebelum enkripsi dimulai, hal pertama yang dilakukan adalah melakukan generasi subkey sebanyak 16 subkey, dengan kunci yang digunakan berupa:

0001001100110100010101110111100110011011101111001101111111110001

key sepanjang 64-bit yang kemudian diproses melalui tahapan:

1. PC-1 Permutation (64-bit menjadi 56-bit), kunci menjadi:
   00010010011010010101101111001001101101111011011111111000
2. Pembagian kunci (56-bit menjadi 28-bit C dan D), kunci menjadi:
   C: 0001001001101001010110111001
   D: 0010011011011110110111111111000
3. Rotasi dan generasi 16 subkey (56-bit menjadi 48-bit), Dari tahapan terakhir, diperoleh subkey berupa:
   Subkey 1: 000110110000001011101111111111000111000001110010
   ....
   Subkey 16: 110010110011110110001011000011100001011111110101

Selanjutnya masuk ke tahap enkripsi,berupa:

1. Initial Permutation (64-bit menjadi 32-bit L dan R), dengan tabel IP:

   IP = [58,50,42,34,26,18,10,2,60,52,44,36,28,20,12,4,62,54,46,38,30,22,14,6,
   64,56,48,40,32,24,16,8,57,49,41,33, 25,17,9,1,59,51,43,35,27,19,11,3,
   61,53,45,37,29,21,13,5,63,55,47,39, 31,23,15,7]

   setelah permutasi dengan tabel IP, menghasilkan input:
   0101010000110000011000001010010010110001101001011001011001100000

   L_0 = 01010100001100000110000010100100
   R_0 = 10110001101001011001011001100000

2. 16 Round Fietsel, dengan tahapan berupa:
   a. Expansion E-Box (32-bit menjadi 48-bit), dengan properti:

      Input R_0 = 10110001101001011001011001100000

      E_BOX = [32,1,2,3,4,5,4,5,6,7,8,9,
      8,9,10,11,12,13,12,13,14,15,16,17,
      16,17,18,19,20,21,20,21,22,23,24,25,
      24,25,26,27,28,29,28,29,30,31,32,1]

      Menghasilkan E(R_0):
      010000000000001010100001010110100001011100000001

   b. XOR E(R_0) dengan subkey 1 (karena sekarang round 1)

      Subkey 1:
      000110110000001011101111111111000111000001110010

      XOR Operation:

      E(R_0): 010000000000001010100001010110100001011100000001
      K_1 : 000110110000001011101111111111000111000001110010
      ———

XOR : 010110110000000001001110101001100110011101110011

c. S-Box Subtitution (48-bit menjadi 32-bit) dengan membagi XOR menjadi 8 blok (per 6 bit), dengan S-Box tabel sebanyak 8 tabel

   Blok 1: 010110

   · Bit pola: 0-1011-0

   - Row = 0 (bit 1&6) = 0_{10}
   - Col = 1011 (bit 2-5) = 11_{10}

   S_1 = [[14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7],
   [0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8],
   [4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0],
   [15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13]]

   S_1[0][11] => baris 0 kolom 11 =>12_{10} = 1100_2

   Dilanjutkan sampai Blok 16 menggunakan S16, diperoleh hasil S-Box Subtitution: 11000101110110100001010110001100

d. P-Box permutation menggunakan tabel P-Box berikut

   P_BOX = [16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,2,8,24, 14,32,27,3,9,19,13,30,6,22,11,4,25]

   Menghasilkan P-Box (f(R_0, K_1)):
   00101100110000011110000101111001

e. Update L_1 dan R_1

   L_1 = R_0 = 10000000010100001011000011100000
   R_1 = L_0 ⊕ f(R_0, K_{16})
   = 01010100001100000110000010100100 ⊕
   00101100110000011110000101111001
   = 00011100101000011110000111111001

Proses diatas dilanjutkan sampai 16 kali, setelah 16 tahap fietsel diperoleh hasil berupa:
L_{16} = 10101100010010000111110100011100
R_{16} = 10110110111100100010011111010100

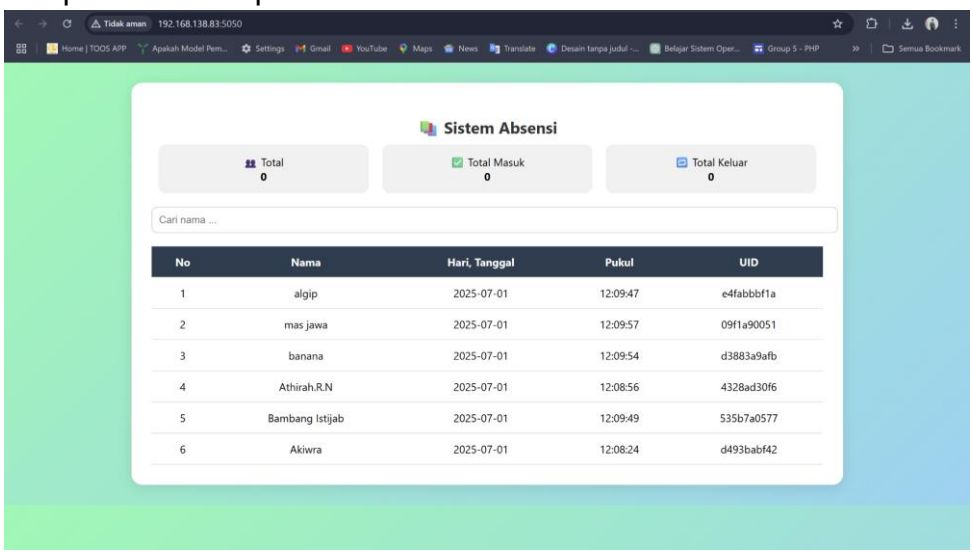
3. Final Permutation (IP^{-1}), dengan menggabung R_{16} + L_{16} (swap posisi):
   combined = R_{16} + L_{16} =
   10110110111100100010011111010100101011000100100001111101000111 00
   Kemudian mengaplikasikan tabel IP^{-1}:

IP_INV = [40,8,48,16,56,24,64,32,39,7,47,15,55,23,63,31,38,6,46,14,54,22,
62,30,37,5,45,13,53,21,61,29,36,4,44,12,52,20,60,28,35,3,43,11,51,19,59,27,
34,2,42,10,50,18,58,26,33,1,41,9, 49,17,57,25]

sehingga menghasilkan ciphertext:
0000110001010100110011111010101001011011110111000011100111010001

b. Dashboard Lokal Instansi

Sementara pada dashboard lokal instansi, waktu yang digunakan sudah memiliki tipe time date, setelah melewati serangkaian data parsing di rute data, dengan mendepkripsi waktu di lokal dan mengembalikasi hasil dekripsi berupa timestamp untuk ditampilkan ke interface.



Gambar 3.4.2. Output di dashboard lokal instansi

Pada tahap dekripsi, proses yang dilakukan hampir sama seperti pada tahap enkrispi, dengan perbedaan berupa urutan penggunaan subkey. Berikut tahapan proses enkripsi pada sistem

1. Initial Permutation (64-bit menjadi 32-bit L dan R), dengan tabel IP:

   IP = [58,50,42,34,26,18,10,2,60,52,44,36,28,20,12,4,62,54,46,38,30,22,14,6,
   64,56,48,40,32,24,16,8,57,49,41,33, 25,17,9,1,59,51,43,35,27,19,11,3,
   61,53,45,37,29,21,13,5,63,55,47,39, 31,23,15,7]

   input ciphertext:
   0000110001010100110011111010101001011011110111000011100111010001

   setelah permutasi dengan tabel IP, menghasilkan input:
   1100101010011010000111010110011011011110010100110001111000100110

   L_0 = 11001010100110100001110101100110
   R_0 = 11011110010100110001111000100110

2. 16 Round Fietsel, dengan tahapan berupa:
    a. Expansion E-Box (32-bit menjadi 48-bit), dengan properti:

        Input R_0 = 11011110010100110001111000100110

        E_BOX = [32,1,2,3,4,5,4,5,6,7,8,9,
        8,9,10,11,12,13,12,13,14,15,16,17,
        16,17,18,19,20,21,20,21,22,23,24,25,
        24,25,26,27,28,29,28,29,30,31,32,1]

        Menghasilkan E(R_0):
        010101011000001001010000001111111010100011111001

    b. XOR E(R_0) dengan subkey 16 (kebalikan dari enkripsi)

        Subkey
        16:110010110011110110001011000011100001011111110101

        XOR Operation:

        E(R_0): 010101011000001001010000001111111010100011111001
        K_{16} : 110010110011110110001011000011100001011111110101
        ———

        XOR : 100111101011111111011011001100011011111100001100

    c. S-Box Subtitution (48-bit menjadi 32-bit) dengan membagi XOR menjadi 8 blok (per 6 bit), dengan S-Box tabel sebanyak 8 tabel

        Blok 1: 100111

        · Bit pola: 1-0011-1

        - Row = 11 (bit 1&6) = 3_{10}
        - Col = 0011 (bit 2-5) = 3_{10}

        S_1 = [[14,4,13,1,2,15,11,8,3,10,6,12,5,9,0,7],
        [0,15,7,4,14,2,13,1,10,6,12,11,9,5,3,8],
        [4,1,14,8,13,6,2,11,15,12,9,7,3,10,5,0],
        [15,12,8,2,4,9,1,7,5,11,3,14,10,0,6,13]]

        S_1[3][3] => baris 3 kolom 3 =>2_{10} = 0010_2

        Dilanjutkan sampai Blok 16 menggunakan S16, diperoleh hasil S-Box Subtitution: 00101111110010101011101110011011

    d. P-Box permutation menggunakan tabel P-Box berikut

        P_BOX = [16,7,20,21,29,12,28,17,1,15,23,26,5,18,31,10,2,8,24,
        14,32,27,3,9,19,13,30,6,22,11,4,25]

Menghasilkan P-Box (f(R_0, K_1)):
01111011011010110110101111010001

e. Update L_1 dan R_1

$L_1 = R_0$ = 11011110010100110001111000100110
$R_1 = L_0 \oplus f(R_0, K_{16})$
= 11001010100110100001110101100110 $\oplus$
01111011011010110110101111010001
= 00011100101000011110000111111001

Proses diatas dilanjutkan sampai 16 kali, setelah 16 tahap fietsel diperoleh hasil berupa:
L_{16} = 10000000010100001011000011100000
R_{16} = 00110000011000000000000010000000

3. Final Permutation (IP^{-1}), dengan menggabung R_{16} + L_{16} (swap posisi):
combined = R_{16} + L_{16} =
00110000011000000000000010000000100000001010000101100001110000 00
Kemudian mengaplikasikan tabel IP^{-1}:

IP_INV = [40,8,48,16,56,24,64,32,39,7,47,15,55,23,63,31,38,6,46,14,54,22, 62,30,37,5,45,13,53,21,61,29,36,4,44,12,52,20,60,28,35,3,43,11,51,19,59,27, 34,2,42,10,50,18,58,26,33,1,41,9, 49,17,57,25]

sehingga menghasilkan plaintext:
00000000000000000000000000000000011010000101101000110010100010 11

Kemudian plaintext diubah kedalam bentuk integer menghasilkan unix timestamp 1750741643, yang berupa timestamp dengan format datetime untuk 2025-06-24 12:07:23.

BAB IV
PENUTUP

4.1 Kesimpulan

Berdasarkan pembahasan di atas, proyek ini telah berhasil mengembangkan sebuah sistem absensi berbasis RFID yang mampu mengamankan data waktu kehadiran melalui enkripsi DES. Sistem ini terbukti dapat mengonversi data waktu dari perangkat IoT menjadi UNIX timestamp, mengenkripsinya menggunakan algoritma DES, dan menyimpannya secara aman ke dalam basis data MySQL. Selain itu, aplikasi web yang dibangun menggunakan Flask juga berhasil mengambil data terenkripsi tersebut, mendekripsinya untuk mengembalikan format waktu asli, dan menampilkannya pada antarmuka pengguna (web) dan mudah dibaca. Integritas dan keaslian data telah berhasil divalidasi di sepanjang alur proses dari enkripsi hingga dekripsi, yang membuktikan bahwa tujuan utama proyek untuk mengamankan dan memverifikasi data waktu telah tercapai.

4.2 Saran

Sistem absensi ini telah berjalan dengan baik, namun untuk penerapan di dunia nyata, terdapat beberapa pengembangan yang harus dilakukan di masa depan. Dimulai dari sisi keamanan, direkomendasikan untuk meningkatkan algoritma enkripsi ke standar yang lebih modern seperti *Advanced Encryption Standard* (AES), mengingat DES memiliki kelemahan pada panjang kuncinya yang pendek dan rentan terhadap serangan *brute-force*.

Dari sisi estetika, untuk implementasi di lingkungan nyata seperti perkantoran, penampilan fisik perangkat perlu diperhatikan. Perangkat pembaca RFID yang saat ini masih berupa rangkaian komponen terbuka dapat dikembangkan lebih lanjut dengan merancang sebuah casing atau selubung khusus yang ringkas, modern, dan kokoh. Penting untuk menyembunyikan dan merapikan instalasi kabel agar tidak terlihat berantakan dan lebih aman bagi pengguna. Dengan demikian, perangkat absensi tidak hanya unggul secara fungsional, tetapi juga dapat terintegrasi dengan baik secara visual dengan interior lingkungan sekitarnya.

DAFTAR PUSTAKA

[1] A. Dirgantara, D. Prabowo, and Tim Redaksi, “Data PeduliLindungi bocor, pemerintah diminta tak saling lempar tanggung jawab,” Kompas.com, 18 Nov. 2022. [Online].Available:https://nasional.kompas.com/read/2022/11/18/05230361/data-pedulilindungi-bocor-pemerintah-diminta-tak-saling-lempar-tanggung. [Accessed: 1 Jul. 2025].

[2] Want, R. (2006). An introduction to RFID technology. *IEEE pervasive computing*, *5*(1), 25-33. Available at: https://sci-hub.se/10.1109/MPRV.2006.2 [Accesed: 1 Juli 2025].

[3] Ahuja, S., & Potti, P. (2010). An introduction to RFID technology. *Communications and Network*, *2*(3), 183-186. Available at: https://www.scirp.org/pdf/CN20100300005_47213461.pdf [Accesed: 1 Juli 2025].

[4] Singh, P. P., & Rani, P. (2014). An approach to extract feature using MFCC. *IOSR Journal of Engineering*, *4*(8), 21-25. Available at: https://iosrjen.org/Papers/vol4_issue8%20(part-1)/D04812125.pdf [Accesed: 1 Juli 2025].

[5] Gupta, S., Jaafar, J., Ahmad, W. W., & Bansal, A. (2013). Feature extraction using MFCC. *Signal & Image Processing: An International Journal*, *4*(4), 101-108. Available at: https://www.researchgate.net/publication/276200395_Feature_Extraction_Using_Mfcc [Accesed: 1 Juli 2025].

[6] Data Encryption Standard (DES) | Set. Available at: https://www.geeksforgeeks.org/computer-networks/data-encryption-standard-des-set-1/ [Accesed: 1 Juli 2025]

[7] A. B., *MySQL*, Jan 2001. [Online]. Available: http://box.cs.istu.ru/public/docs/other/_New/Books/Data/DB/MySQL/MySQL%20Bible.pdf. [Accessed: 1 Juli 2025].

LAMPIRAN

1. Source Code

Source Code lengkap dari sistem absensi ini dapat diakses melalui repositori online pada tautan berikut: [Link]