Implementasi Enkripsi Dua Tahap Dengan Algoritma Camellia dan RSA Untuk Memperkuat Keamanan Database

1st Ahmad Jundy Azfarozan
Fakultas Informatika
Universitas Telkom
Purwokerto, Indonesia
azfarozan@student.telkomuniversity.ac.id

2nd Wahyu Adi Prabowo, S.Kom, M.B.A, M.Kom.
Fakultas Informatika
Universitas Telkom
Purwokerto, Indonesia
wahyup@telkomuniversity.ac.id

3rd Cahyo Prihantoro, S.Kom., M.Eng
Fakultas Informatika
Universitas Telkom
Purwokerto, Indonesia
cahyop@telkomuniversity.ac.id

***Abstrak* — Seiring meningkatnya ancaman siber, keamanan data pada sistem berbasis web menjadi prioritas utama, terutama terhadap serangan seperti *SQL Injection*. Penelitian ini bertujuan untuk merancang dan menguji sistem keamanan database yang diperkuat melalui implementasi enkripsi dua tahap. Metode yang digunakan adalah kombinasi algoritma kriptografi simetris Camellia yang efisien dan algoritma asimetris RSA yang kuat. Penelitian ini berfokus pada pengembangan sistem, pengujian fungsionalitas enkripsi-dekripsi, serta analisis ketahanan sistem terhadap simulasi serangan *SQL Injection*. Sistem dibangun menggunakan PHP dengan *library* OpenSSL untuk fungsi kriptografi. Hasil penelitian menunjukkan bahwa data sensitif pasien berhasil dienkripsi dan tidak dapat dibaca saat terjadi kebocoran data melalui eksploitasi celah *SQL Injection*. Meskipun penyerang berhasil mengakses dan melakukan *dump* terhadap tabel, data yang diperoleh tetap dalam bentuk *ciphertext* yang tidak dapat dimengerti. Kesimpulannya, pendekatan enkripsi dua tahap ini terbukti efektif memberikan lapisan pertahanan yang kuat pada level data, menjaga kerahasiaan informasi meskipun sistem aplikasi memiliki kerentanan.**

***Kata kunci*— Camellia, RSA, Data Security**

I. PENDAHULUAN

Pada era digital ini, menjaga keamanan data menjadi sangat penting bagi individu, perusahaan, dan pemerintah. Seiring dengan banyak informasi yang dibagikan dan data yang disimpan secara digital, semakin besar pula risikonya terhadap kerahasiaan dan integritas data. Ancaman seperti peretasan, pencurian data, dan serangan malware semakin parah, sehingga diperlukan solusi keamanan data yang lebih efektif [1]. Dengan meningkatnya nilai informasi sebagai komoditas berharga, penting untuk memastikan bahwa data sensitif hanya dapat diakses oleh individu yang memiliki otorisasi yang tepat. Oleh karena itu, mencegah akses yang tidak sah dan potensi kerusakan bagi pemegang data menjadi prioritas utama [2]. Salah satu bagian terpenting dalam sistem informasi modern adalah basis data. Basis data tidak hanya berfungsi sebagai pusat penyimpanan informasi penting, tetapi juga menjadi target utama dalam berbagai jenis serangan siber karena nilai dan sensitivitas data yang tersimpan didalamnya [3]. Website sebagai antarmuka utama dalam banyak sistem digital memberikan kemudahan akses dan interaksi bagi pengguna, namun tidak jarang menjadi titik lemah dari sisi keamanan sistem. Banyak aplikasi web yang tidak dilengkapi dengan mekanisme keamanan yang memadai, sehingga membuka peluang bagi pihak tidak bertanggung jawab untuk mengeksploitasi celah yang ada [4]. Serangan SQL Injection merupakan salah satu ancaman serius dalam keamanan aplikasi web. Berbagai metode telah dirancang untuk mendeteksi dan mengantisipasi jenis serangan ini, mulai dari teknik analisis statis hingga pendekatan dinamis. Namun demikian, kedua metode tersebut masih memiliki sejumlah kelemahan. Analisis statis kerap kali tidak mampu mengenali kelemahan dalam proses penyaringan input, khususnya pada sistem yang menerapkan konsep pemrograman berorientasi objek. Di sisi lain, pendekatan dinamis cenderung mereduksi definisi serangan SQLi, sehingga berpotensi memblokir permintaan yang sebenarnya valid. Untuk mengatasi kelemahan ini, salah satu solusi yang semakin banyak diterapkan adalah enkripsi data. Dengan mengenkripsi data sensitif, proses transmisi informasi menjadi lebih aman dari upaya penyadapan maupun manipulasi, sekaligus memberikan lapisan perlindungan tambahan terhadap akses illegal [5]. Salah satu pendekatan yang umum digunakan untuk menjaga keamanan data adalah melalui penerapan teknik kriptografi. Kriptografi merupakan cabang ilmu yang mempelajari metode pengamanan informasi agar tetap terlindungi selama proses transmisi dari satu lokasi ke lokasi lainnya. Dengan menerapkan teknik kriptografi yang tepat, data sensitif dapat dienkripsi sehingga hanya pihak-pihak yang memiliki otorisasi yang sah yang dapat mengakses dan memahami informasi tersebut [6]. Melihat pentingnya penerapan kriptografi dalam menjaga kerahasiaan data, muncul kebutuhan untuk mengembangkan pendekatan yang lebih kuat dan berlapis. Pengembangan teknologi enkripsi yang kuat merupakan aspek krusial dalam upaya melindungi informasi penting dari akses pihak-pihak yang tidak

bertanggung jawab [1]. Dalam hal ini, algoritma RSA menjadi salah satu metode kriptografi yang banyak digunakan karena menawarkan sistem kunci publik dan privat yang mampu menjaga kerahasiaan dan integritas data [7]. Khususnya, penerapan RSA dengan panjang kunci 2048-bit memberikan tingkat perlindungan tambahan melalui tingkat kompleksitas perhitungan yang tinggi dan prinsip kriptografi asimetris berbasis kunci publik [8]. Penelitian sebelumnya menunjukkan bahwa penerapan algoritma enkripsi yang kuat, seperti RSA, mampu memberikan tingkat perlindungan yang tinggi dalam menjaga kerahasiaan dan integritas data, khususnya pada sistem berbasis web [9]. RSA merupakan algoritma kriptografi yang menggunakan dua pasang kunci, yaitu kunci publik dan kunci privat, dalam proses enkripsi dan dekripsi. Kunci publik dapat disebarluaskan secara terbuka untuk mengenkripsi data, sementara kunci privat hanya diketahui oleh pihak yang berwenang dan digunakan untuk mendekripsi informasi tersebut [10]. Algoritma Camellia mendukung panjang kunci yang bervariasi, yaitu 128, 192, dan 256 bit. Modifikasi dari Feistel Cipher, algoritma ini menggunakan 18 putaran untuk kunci 128 bit dan 24 putaran untuk kunci 192 dan 256 bit. Meskipun telah dipatenkan, penggunaan algoritma Camellia tidak memerlukan pembayaran royalti asalkan algoritma tersebut tidak dimodifikasi [11]. Penerapan enkripsi dua tahap diyakini mampu memberikan perlindungan data yang lebih kuat dibandingkan pendekatan enkripsi satu tahap, karena mengombinasikan kecepatan enkripsi simetris dan kekuatan autentikasi asimetris dalam satu sistem pengamanan [12]. Pada pendekatan ini, algoritma Camellia digunakan terlebih dahulu untuk mengacak isi data guna menjaga kerahasiaan informasi. Selanjutnya, data yang telah dienkripsi tersebut dienkripsi kembali dengan algoritma RSA, yang memanfaatkan pasangan kunci publik dan privat. Pendekatan berlapis ini tidak hanya melindungi data dari akses yang tidak sah, tetapi juga memberikan perlindungan tambahan apabila terjadi serangan terhadap basis data, termasuk serangan SQL Injection. Sebagai implementasi dari pendekatan ini, penelitian ini merancang sebuah program yang mengenkripsi data sensitif seperti Nomor Identitas, Alamat, dan Nama Lengkap menggunakan kombinasi algoritma Camellia dan RSA. Sebelum proses enkripsi dilakukan, pengguna diharuskan memasukkan password yang diperlukan untuk proses enkripsi dan dekripsi. Setelah terenkripsi, data kemudian disimpan dan diperbarui dalam basis data. Penerapan enkripsi pada data pribadi ini sesuai dengan Undang-Undang Republik Indonesia Nomor 27 Tahun 2022 tentang Perlindungan Data Pribadi, yang mewajibkan pengendali data untuk melindungi kerahasiaan dan keamanan data pribadi dengan cara yang sesuai perkembangan teknologi. Nama lengkap, alamat, dan nomor identitas termasuk kategori data pribadi yang wajib dilindungi untuk mencegah penyalahgunaan atau akses ilegal.

II. KAJIAN TEORI

A. Kriptografi

Istilah *kriptografi* berasal dari kata *crypto* yang berarti rahasia dan *graphy* yang berarti tulisan atau catatan. Dengan demikian, kriptografi dapat diartikan sebagai seni untuk menyembunyikan tulisan agar tidak dapat dipahami oleh pihak yang tidak berwenang. Secara umum, kriptografi merupakan cabang ilmu pengetahuan dan seni yang bertujuan melindungi kerahasiaan data. Bidang ini mempelajari teknik-teknik matematika yang berkaitan dengan pengamanan informasi, termasuk menjaga kerahasiaan, memastikan keaslian, integritas data, serta proses otentikasi. Namun demikian, kriptografi tidak mencakup seluruh aspek dalam keamanan informasi[13]. Kriptografi sendiri dibagi menjadi dua kategori utama yaitu :

1. Kriptografi Simetris

   Enkripsi simetris adalah jenis enkripsi yang umum digunakan di mana kunci yang digunakan untuk mengenkripsi juga digunakan untuk mendekripsi. Artinya pengirim dan penerima harus berbagi kunci yang sama. Jika kunci ini jatuh ke tangan yang tidak berwenang, pihak tersebut dapat mengenkripsi dan mendekripsi [14].

2. Kriptografi Asimetris

   Dalam kriptografi asimetris, berbeda dengan kriptografi simetris, digunakan dua jenis kunci enkripsi yang berbeda untuk proses enkripsi dan dekripsi. Kunci-kunci ini dikenal sebagai kunci publik dan kunci privat [14].

B. Algoritma *Camellia*

Algoritma *Camellia* dikembangkan pada tahun 2000 melalui kolaborasi antara NTT dan Mitsubishi Electric Corporation. Algoritma ini menggabungkan prinsip-prinsip dari algoritma kriptografi E2 (yang dikembangkan oleh NTT) dan MISTY (yang dikembangkan oleh Mitsubishi). Dalam algoritma *Camellia*, setiap blok data memiliki ukuran tetap sebesar 128 bit, dengan panjang kunci yang dapat bervariasi antara 128, 192, atau 256 bit. Algoritma ini merupakan modifikasi dari *Feistel Cipher*, dengan 18 putaran digunakan untuk kunci 128 bit dan 24 putaran untuk kunci 192 atau 256 bit. Meskipun memiliki paten, *Camellia* tersedia untuk digunakan tanpa biaya lisensi, selama tidak ada modifikasi pada algoritma yang telah ditetapkan [15].

C. Algoritma *River, Shamir, Alderman* (RSA)

Algoritma kriptografi RSA menggunakan operasi matematis berdasarkan pada pemfaktoran angka-angka yang sangat besar, dan sampai saat ini dianggap sebagai metode yang aman untuk enkripsi data [16]. RSA adalah algoritma kunci publik yang melibatkan penggunaan dua kunci: kunci publik dan kunci privat. Proses enkripsi dan dekripsi dalam RSA didasarkan pada prinsip bilangan prima dan aritmetika modulo. Kunci enkripsi dan dekripsi keduanya berupa bilangan bulat. Kunci enkripsi, yang dikenal sebagai kunci publik, tersedia untuk umum dan tidak dirahasiakan, sementara kunci dekripsi, yang disebut kunci privat, harus tetap rahasia. Untuk menemukan kunci dekripsi dalam RSA, diperlukan kemampuan untuk memfaktorkan bilangan bulat menjadi faktor-faktor primanya, sebuah tugas yang diketahui sangat sulit karena belum ada algoritma yang efisien untuk memfaktorkan bilangan bulat yang sangat besar dalam waktu singkat. Dengan demikian, keamanan RSA saat ini bergantung pada kompleksitas matematis dari pemfaktoran bilangan bulat besar [17].

D. *Cipher Block Chaining* (CBC)

Cipher Block Chaining (CBC) adalah salah satu mode operasi dalam kriptografi blok yang digunakan untuk

meningkatkan keamanan proses enkripsi dengan menambahkan mekanisme umpan balik antar blok. Berbeda dengan Electronic Code Book (ECB) yang mengenkripsi setiap blok plaintext secara terpisah, CBC mengenkripsi blok data dengan menggabungkan hasil enkripsi dari blok sebelumnya melalui operasi XOR, sehingga menciptakan keterkaitan antar blok Ciphertext. Secara umum, proses enkripsi pada mode CBC dilakukan dengan terlebih dahulu melakukan XOR antara blok plaintext saat ini dengan blok Ciphertext dari hasil enkripsi sebelumnya. Hasil dari XOR ini kemudian dienkripsi menggunakan algoritma kriptografi tertentu. Proses ini membuat setiap blok Ciphertext tidak hanya bergantung pada blok plaintext saat itu, tetapi juga pada seluruh blok sebelumnya, sehingga lebih tahan terhadap analisis statik [18].

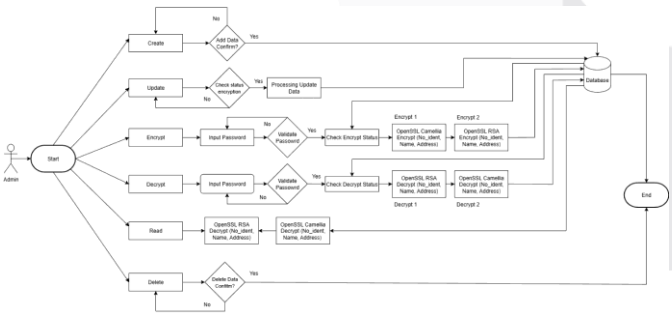
E. *SQL Injection*

SQL Injection merupakan salah satu bentuk kerentanan keamanan yang terjadi ketika penyerang mampu memanipulasi perintah SQL yang dijalankan oleh aplikasi, sehingga dapat mengakses, mengubah, atau merusak data dalam basis data secara tidak sah. Pada umumnya, ini memungkinkan seorang penyerang untuk melihat data spesifik yang seharusnya tidak bisa dia dapatkan. Sebagai contoh, itu bisa mencakup data milik pengguna lain, login, struktur tabel, atau data lainnya yang dapat diakses oleh aplikasi. Dalam banyak kasus, seorang penyerang juga dapat memodifikasi atau menghapus data tersebut, menyebabkan perubahan yang persisten pada konten atau perilaku aplikasi [19]. SQL Injection memungkinkan seseorang untuk masuk ke dalam sistem basis data tanpa memerlukan akun, dengan cara memanfaatkan pengaturan bawaan SQL [20].

III. METODE

Metode penelitian yang digunakan dalam studi ini terdiri dari beberapa tahapan penting, yaitu Perancangan Sistem, Integrasi Sistem, serta Pengujian Sistem.

A. Perancangan Sistem



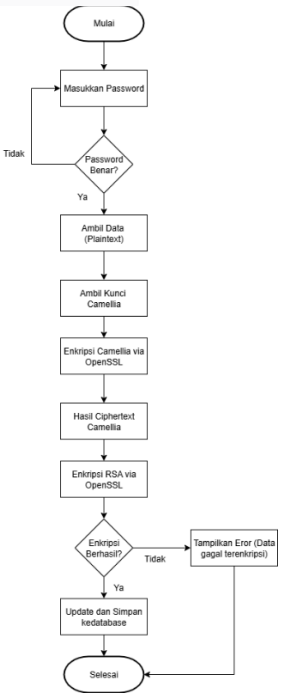
GAMBAR 1

Proses diawali oleh user yang memiliki beberapa opsi tindakan melalui antarmuka web, yaitu membuat data baru, memperbarui data, melakukan enkripsi, dekripsi, membaca data, dan menghapus data. Setiap proses diawali dengan validasi atau konfirmasi tindakan, misalnya pada pembuatan dan penghapusan data yang memerlukan konfirmasi terlebih dahulu dari pengguna untuk mencegah kesalahan input atau penghapusan yang tidak disengaja. Pada tahap enkripsi, user diminta untuk memasukkan password sebagai langkah autentikasi yang berfungsi memastikan hanya pengguna berwenang yang dapat mengakses fitur enkripsi. Sistem kemudian memeriksa status enkripsi pada data yang dipilih. Jika validasi berhasil, data sensitif seperti nomor identitas, nama lengkap, dan alamat akan melalui dua tahap enkripsi. Tahap pertama menggunakan algoritma Camellia sebagai enkripsi simetris yang cepat dan efisien, dengan proses pembagian data ke dalam blok, kemudian dilakukan operasi substitusi dan permutasi sesuai panjang kunci (128, 192, atau 256 bit) hingga dihasilkan ciphertext tahap pertama. Selanjutnya, hasil enkripsi Camellia tersebut dienkripsi kembali menggunakan algoritma RSA, memanfaatkan kunci publik yang telah dibuat untuk menghasilkan ciphertext akhir. Hasil akhir dari proses enkripsi ganda ini kemudian disimpan dalam basis data dalam format terenkripsi.

Proses dekripsi dilakukan secara terbalik dengan autentikasi password terlebih dahulu untuk memvalidasi hak akses pengguna. Sistem memeriksa status enkripsi data, kemudian menggunakan kunci privat RSA untuk mendekripsi hasil enkripsi tahap kedua, menghasilkan ciphertext Camellia. Tahap selanjutnya adalah dekripsi dengan algoritma Camellia menggunakan kunci rahasia yang sama untuk mendapatkan kembali data asli dalam bentuk plaintext. Proses pembacaan data pada sistem hanya dapat dilakukan setelah data berhasil didekripsi sepenuhnya, sehingga memastikan bahwa informasi sensitif tetap terlindungi dari akses tidak sah.

Selain itu, setiap operasi pembuatan data baru maupun pembaruan data pada sistem selalu diarahkan untuk disimpan ke basis data hanya setelah melewati proses validasi atau konfirmasi. Hal ini dirancang untuk mencegah input yang tidak sah atau tidak diinginkan dan memastikan konsistensi data. Dengan alur kerja seperti ini, sistem dikembangkan untuk memberikan tingkat keamanan yang tinggi pada data sensitif pengguna melalui penerapan enkripsi dua tahap Camellia dan RSA, yang mampu mempersulit upaya akses ilegal maupun serangan injeksi SQL (*SQL Injection*), sekaligus mendukung kewajiban perlindungan data pribadi sesuai regulasi yang berlaku.

B. Integrasi Sistem



GAMBAR 2

Proses dimulai dengan pengguna wajib memasukkan password sebagai langkah autentikasi. Tahap ini berfungsi untuk memverifikasi bahwa hanya pengguna yang memiliki otorisasi yang diizinkan menjalankan proses enkripsi pada data sensitif. Jika password yang diberikan tidak valid, sistem secara otomatis akan menolak permintaan tersebut dan mengarahkan kembali ke langkah awal tanpa melanjutkan proses enkripsi. Mekanisme ini dirancang untuk mencegah potensi akses tidak sah atau penyalahgunaan fitur enkripsi oleh pihak yang tidak berwenang. Jika password dinyatakan benar, sistem melanjutkan dengan mengambil data dalam bentuk plaintext yang telah dimasukkan user sebelumnya, seperti nomor identitas, nama lengkap, dan alamat. Tahap berikutnya adalah pengambilan kunci Camellia yang sudah didefinisikan dalam sistem untuk digunakan pada proses enkripsi simetris. Dengan menggunakan pustaka OpenSSL, sistem melakukan enkripsi data plaintext melalui algoritma Camellia. Proses enkripsi ini melibatkan operasi substitusi dan permutasi pada blok data sesuai panjang kunci (128, 192, atau 256 bit) yang telah dipilih, sehingga menghasilkan ciphertext Camellia yang sulit diuraikan tanpa kunci yang tepat. Hasil ciphertext dari tahap Camellia tidak langsung disimpan ke basis data, tetapi melalui proses enkripsi tahap kedua menggunakan algoritma RSA. Pada tahap ini, ciphertext Camellia akan dienkripsi kembali dengan RSA menggunakan kunci publik yang sudah disiapkan dalam sistem. RSA berfungsi sebagai mekanisme enkripsi asimetris untuk meningkatkan keamanan, dengan membagi kunci menjadi kunci publik (untuk enkripsi) dan kunci privat (untuk dekripsi), sehingga hanya pemilik kunci privat yang dapat membuka hasil enkripsi tersebut. Enkripsi RSA dijalankan melalui pustaka OpenSSL yang mendukung operasi enkripsi asimetris berbasis kunci publik-privat dengan panjang kunci, misalnya 2048 bit. Setelah proses enkripsi RSA selesai, sistem melakukan pengecekan apakah proses enkripsi berhasil dilakukan dengan benar. Apabila terjadi kesalahan atau kegagalan pada salah satu tahap, sistem akan menampilkan pesan error yang menjelaskan bahwa data gagal terenkripsi, sehingga tidak ada data yang akan disimpan ke basis data dalam keadaan tidak aman. Namun jika semua proses enkripsi berhasil tanpa kesalahan, sistem melanjutkan ke tahap akhir yaitu *Update dan Simpan ke Database*. Data yang telah terenkripsi secara berlapis kemudian disimpan ke dalam basis data. Dengan desain alur seperti ini, sistem memastikan bahwa tidak ada data sensitif yang tersimpan dalam basis data dalam bentuk plaintext, sehingga mempersulit upaya penyadapan atau akses ilegal langsung melalui query basis data, termasuk serangan seperti *SQL Injection*.

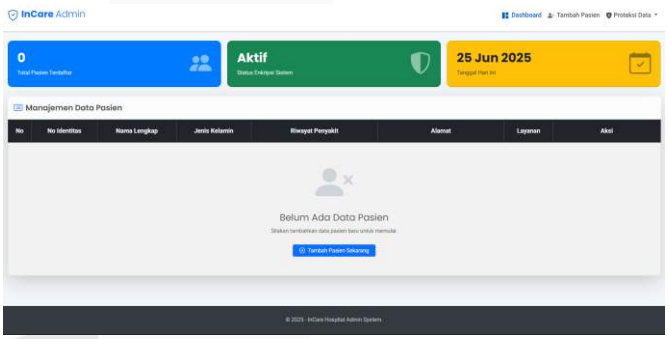
C. Pengujian Sistem

Pengujian sistem dilakukan untuk memastikan bahwa penerapan enkripsi dua tahap menggunakan algoritma Camellia dan RSA pada aplikasi berbasis web dapat berjalan. Pengujian dirancang untuk mengevaluasi fungsi enkripsi dan dekripsi dan menilai keberhasilan perlindungan data terhadap potensi serangan seperti SQL Injection. Proses pengujian diawali dengan simulasi input data dummy seperti nomor identitas, nama lengkap, dan alamat melalui antarmuka web. Setelah data diinput, fitur enkripsi diaktifkan dengan autentikasi password yang valid. Sistem kemudian menjalankan proses enkripsi dua tahap secara berurutan: tahap pertama menggunakan algoritma Camellia untuk enkripsi simetris, kemudian dilanjutkan tahap kedua dengan algoritma RSA untuk enkripsi asimetris menggunakan kunci publik. Hasil pengujian mencatat apakah data berhasil dienkripsi dengan benar dan disimpan ke dalam basis data dalam bentuk yang tidak dapat dibaca secara langsung. Pengujian dekripsi dilakukan dengan mengambil data terenkripsi dari basis data dan menjalankan proses dekripsi dua tahap: RSA dengan kunci privat untuk membuka hasil enkripsi tahap kedua, kemudian Camellia untuk memulihkan data ke bentuk plaintext. Pengujian ini memverifikasi apakah data asli dapat dihasilkan kembali secara utuh dan valid tanpa kehilangan informasi. Selain pengujian fungsi enkripsi dan dekripsi, dilakukan juga pengujian terhadap keamanan sistem dari potensi akses ilegal. Salah satunya adalah simulasi serangan *SQL Injection* pada parameter input web untuk memastikan sistem tidak mengembalikan data asli meskipun terjadi akses query yang tidak valid. Pengujian ini memastikan bahwa data sensitif tetap aman dalam bentuk terenkripsi di basis data dan hanya dapat diakses melalui proses dekripsi yang sah dan terotorisasi.

IV. HASIL DAN PEMBAHASAN

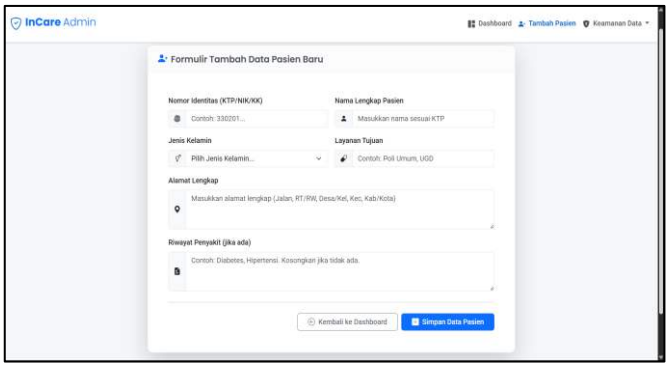
A. Tampilan Sistem

Sistem dibangun berbasis web dengan menggunakan bahasa pemrograman PHP dan Database MySQL. Adapun halaman-halaman yang tersedia mencakup halaman dashboard sebagai tampilan awal, formulir entri data pasien, halaman untuk melakukan pembaruan data, tampilan rincian informasi pasien, serta halaman khusus yang digunakan untuk menjalankan proses enkripsi dua tahap dengan kombinasi algoritma Camellia dan RSA.



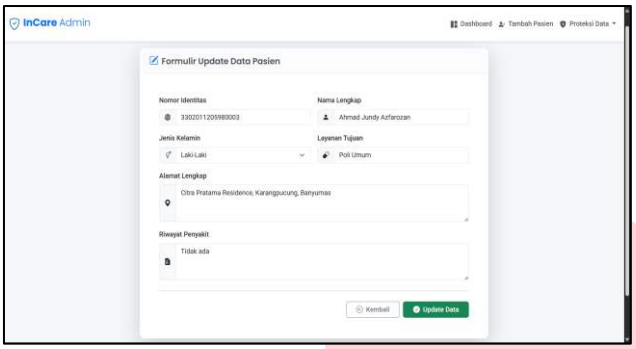
GAMBAR 3

Proses penambahan data pasien dilakukan dengan cara mengisi sejumlah field yang tersedia pada form input, di antaranya mencakup Nomor identitas, nama lengkap, Alamat, dan informasi lainnya yang diperlukan sesuai struktur data yang telah ditentukan dalam sistem.



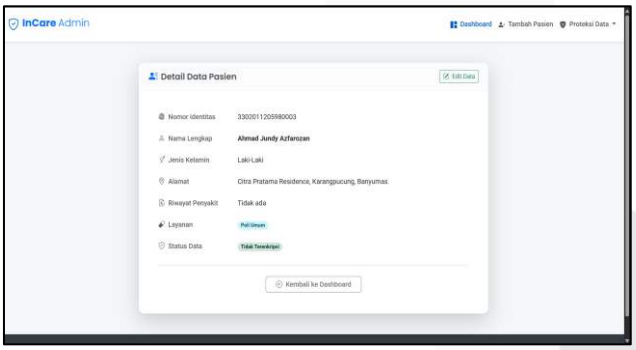
GAMBAR 4

Sedangkan proses pembaruan data dilakukan dengan mengisi kembali field yang sama seperti pada saat input data, sebagaimana terlihat pada Gambar 4.3. Gambar tersebut menampilkan rincian data pasien yang telah tersimpan sebelumnya, mencakup detail seperti nomor identitas, nama lengkap, jenis kelamin, alamat, riwayat penyakit, hingga layanan yang diterima.



GAMBAR 5

Pada halaman tampilan detail data pasien, pengguna dapat meninjau informasi lengkap milik pasien. Informasi yang ditampilkan mencakup data pribadi, alamat, serta keterangan medis dan layanan yang diterima. Selain itu, terdapat pula indikator penting berupa status data yang menunjukkan bahwa data pasien tersebut masih dalam kondisi plaintext dan belum diamankan oleh sistem. Halaman ini juga dilengkapi dengan tombol edit data yang akan mengarahkan pengguna ke formulir pembaruan apabila diperlukan modifikasi, serta tombol kembali ke dashboard untuk memudahkan navigasi ke halaman utama admin.



GAMBAR 6

B. Integrasi Sistem

Integrasi ini dikembangkan untuk memastikan bahwa data sensitif, seperti nomor identitas, nama lengkap, dan alamat pasien, hanya dapat disimpan dan diakses dalam bentuk terenkripsi melalui proses yang aman dan terautentikasi.

```php
function prosesEnkripsi($inputPassword, $realPassword)
{
    if ($inputPassword !== $realPassword) {
        throw new Exception('Password salah!');
    }

    $cek = mysqli_query($GLOBALS['koneksi'], "SELECT COUNT(*) as jumlah_encrypt
        FROM tb_pasien WHERE encrypt = 0");
    $dataCek = mysqli_fetch_assoc($cek);
    if ($dataCek['jumlah_encrypt'] == 0) {
        echo "<script>alert('Data sudah terenkripsi, Harap dekripsi dahulu');
            window.location='index.php';</script>";
        exit;
    }

    $query = "SELECT id, no_ident, nama, jk, alamat, riwayat, layanan FROM
        tb_pasien WHERE encrypt = 0";
    $result = mysqli_query($GLOBALS['koneksi'], $query);

    if ($result) {
        while ($row = mysqli_fetch_assoc($result)) {
            $id = $row['id'];

            $camelliaKey = getCamelliaKey();
            $encrypted_no_ident = camelliaEncrypt($row['no_ident'], $camelliaKey
                );
            $encrypted_nama = camelliaEncrypt($row['nama'], $camelliaKey);
            $encrypted_alamat = camelliaEncrypt($row['alamat'], $camelliaKey);

            $rsa_encrypted_no_ident = rsaEncrypt($encrypted_no_ident,
                $GLOBALS['publicKey']);
            $rsa_encrypted_nama = rsaEncrypt($encrypted_nama,
                $GLOBALS['publicKey']);
            $rsa_encrypted_alamat = rsaEncrypt($encrypted_alamat,
                $GLOBALS['publicKey']);

            $updateQuery = "UPDATE tb_pasien SET
                            no_ident = '$rsa_encrypted_no_ident',
                            nama = '$rsa_encrypted_nama',
                            jk = '{$row['jk']}',
                            alamat = '$rsa_encrypted_alamat',
                            riwayat = '{$row['riwayat']}',
                            layanan = '{$row['layanan']}',
                            encrypt = 1
                            WHERE id = $id";
            mysqli_query($GLOBALS['koneksi'], $updateQuery);
        }
        return "Data berhasil dienkripsi";
    }
```

GAMBAR 7

Pada sisi enkripsi, sistem memulai proses dengan autentikasi password yang dimasukkan oleh pengguna. Langkah ini digunakan untuk memvalidasi hak akses, memastikan hanya pengguna yang berwenang dapat menjalankan fungsi enkripsi. Setelah validasi berhasil, sistem melakukan pemeriksaan terhadap status kolom *encrypt* pada basis data untuk menentukan apakah data sudah terenkripsi. Jika ditemukan data dengan status *encrypt = 0*, sistem mengambil data plaintext dari basis data dan memprosesnya secara berulang untuk setiap entri.

Tahap pertama enkripsi dilakukan menggunakan algoritma Camellia. Pada langkah ini, data seperti nomor identitas, nama lengkap, dan alamat dienkripsi menggunakan kunci Camellia yang dihasilkan dari fungsi *getCamelliaKey()*. Hasil dari enkripsi Camellia berupa ciphertext sementara yang kemudian menjadi input untuk tahap kedua, yaitu enkripsi RSA. RSA digunakan sebagai enkripsi asimetris dengan kunci publik yang dikelola di server. Fungsi *rsaEncrypt()* memproses hasil enkripsi Camellia menggunakan kunci publik, menghasilkan ciphertext akhir yang aman untuk disimpan. Sistem kemudian memperbarui basis data dengan ciphertext hasil enkripsi RSA dan menetapkan nilai *encrypt* menjadi 1 untuk menandai bahwa data sudah dienkripsi.

```php
function prosesDekripsi($password, $storedPassword): string
{
    if ($password !== $storedPassword) {
        throw new Exception(message: 'Password salah!');
    }

    $query = "SELECT id, no_ident, nama, jk, alamat, riwayat, layanan FROM tb_pasien WHERE encrypt = 1";
    $result = mysqli_query(mysql: $GLOBALS['koneksi'], query: $query);
    if ($result) {
        while ($row = mysqli_fetch_assoc(result: $result)) {
            $id = $row['id'];

            $camelliaKey = getCamelliaKey();
            $rsa_decrypted_no_ident = rsaDecrypt(encryptedData: $row['no_ident'], privateKey: $GLOBALS['privateKey']);
            $rsa_decrypted_nama = rsaDecrypt(encryptedData: $row['nama'], privateKey: $GLOBALS['privateKey']);
            $rsa_decrypted_alamat = rsaDecrypt(encryptedData: $row['alamat'], privateKey: $GLOBALS['privateKey']);

            $decrypted_no_ident = camelliaDecrypt(cipherText: $rsa_decrypted_no_ident, key: $camelliaKey);
            $decrypted_nama = camelliaDecrypt(cipherText: $rsa_decrypted_nama, key: $camelliaKey);
            $decrypted_alamat = camelliaDecrypt(cipherText: $rsa_decrypted_alamat, key: $camelliaKey);

            $updateQuery = "UPDATE tb_pasien SET
                            no_ident = '$decrypted_no_ident',
                            nama = '$decrypted_nama',
                            jk = '{$row['jk']}',
                            alamat = '$decrypted_alamat',
                            riwayat = '{$row['riwayat']}',
                            layanan = '{$row['layanan']}',
                            encrypt = 0
                            WHERE id = $id";
            mysqli_query(mysql: $GLOBALS['koneksi'], query: $updateQuery);
        }
        return "Data berhasil didekripsi";
    }
    return "Data gagal didekripsi";
}
```

GAMBAR 8

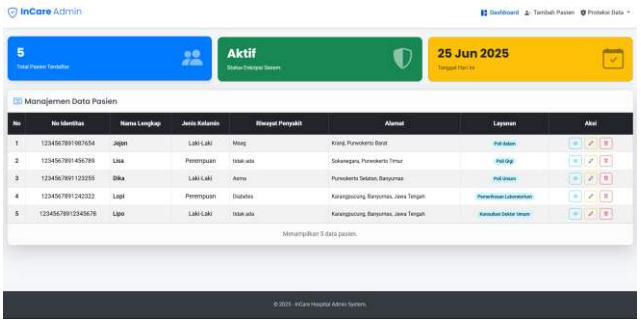
Sebaliknya, pada sisi dekripsi, sistem juga menerapkan validasi password di awal proses untuk mengonfirmasi hak akses pengguna. Setelah berhasil, sistem mengeksekusi query untuk mengambil data dengan status *encrypt = 1*. Proses dekripsi dilakukan dalam dua tahap berurutan. Pertama, fungsi *rsaDecrypt()* digunakan untuk membuka enkripsi RSA dengan kunci privat yang tersimpan aman di server, sehingga memperoleh kembali hasil ciphertext dari enkripsi Camellia. Tahap kedua menggunakan fungsi *camelliaDecrypt()* untuk mendekripsi ciphertext Camellia menggunakan kunci yang sama seperti pada saat enkripsi, sehingga data asli dalam bentuk plaintext dapat diperoleh kembali. Setelah proses dekripsi selesai, sistem memperbarui basis data dengan nilai plaintext hasil dekripsi, dan menetapkan nilai *encrypt* menjadi 0 untuk menunjukkan bahwa data kembali ke bentuk terbaca. Integrasi fungsi enkripsi dan dekripsi dalam kode program dilakukan pada file *functions.php* menggunakan bahasa pemrograman PHP dengan pustaka OpenSSL. Fungsi *prosesEnkripsi()* memproses semua data yang belum terenkripsi secara otomatis dengan autentikasi password, enkripsi dua tahap, dan update basis data. Sementara fungsi *prosesDekripsi()* memproses semua data terenkripsi untuk didekripsi kembali ke bentuk aslinya dengan mekanisme serupa. Model integrasi ini mendukung pengelolaan data secara batch, memastikan konsistensi status enkripsi melalui kolom *encrypt* di basis data, serta menyediakan kontrol akses berbasis password untuk menjaga kerahasiaan data sensitif. Dengan pendekatan ini, sistem memberikan perlindungan ganda melalui enkripsi simetris Camellia yang efisien dan enkripsi asimetris RSA yang aman untuk distribusi kunci, memastikan data tersimpan dalam bentuk terenkripsi yang tidak dapat dibaca langsung bahkan jika terjadi akses ilegal ke basis data.

C. Pengujian Sistem

Tahap pengujian ini dirancang untuk memastikan bahwa data sensitif yang disimpan dalam basis data tetap terlindungi, meskipun sistem diuji dengan serangan seperti *SQL Injection* yang biasanya menargetkan kelemahan pada input query.

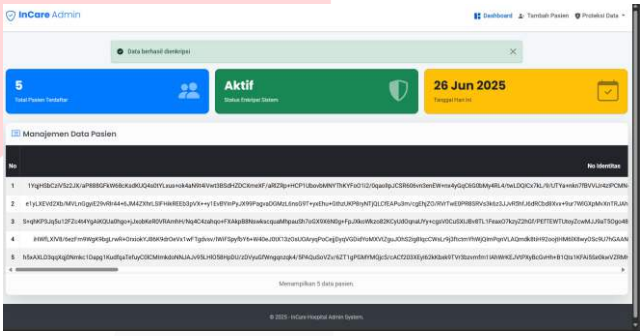
1. Data Pengujian

Data yang ditampilkan pada gambar berikut merupakan data pasien yang dimasukkan melalui form tambah pasien. Informasi tersebut mencakup nomor identitas, nama lengkap, jenis kelamin, alamat, riwayat penyakit, dan jenis layanan. Data dummy ini berfungsi sebagai data pengujian untuk implementasi enkripsi dua tahap menggunakan algoritma Camellia dan RSA. Setelah data tersimpan, proses enkripsi dilakukan secara manual melalui halaman utama.



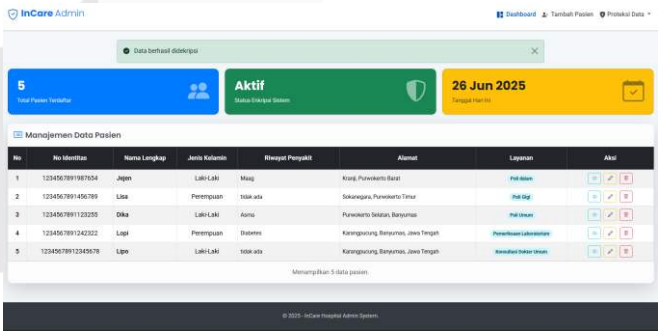
GAMBAR 9

2. Enkripsi dan Dekripsi



GAMBAR 10

Pengujian proses enkripsi dilakukan dengan menggunakan password yang telah terdaftar dalam sistem untuk memastikan bahwa data yang dimasukkan benar-benar melewati tahapan enkripsi dua tahap menggunakan algoritma Camellia dan RSA. Tujuan dari pengujian ini adalah untuk memastikan bahwa data yang telah dienkripsi tidak dapat diakses atau dibaca secara langsung dalam basis data tanpa melalui mekanisme dekripsi yang valid.

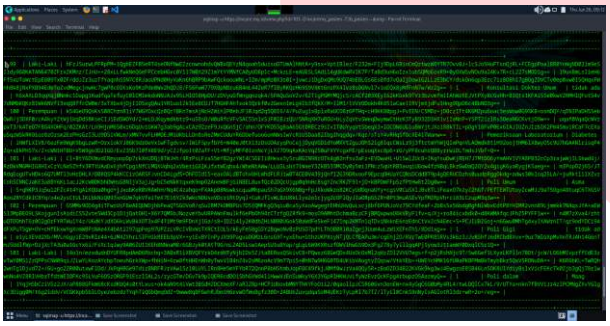


GAMBAR 11

Pengujian dekripsi dilakukan untuk memastikan bahwa data yang telah terenkripsi dapat dikembalikan ke bentuk aslinya secara utuh dan akurat. Proses dekripsi dilakukan secara berurutan, dimulai dari dekripsi menggunakan algoritma RSA untuk membuka lapisan enkripsi asimetris, kemudian dilanjutkan dengan dekripsi menggunakan algoritma Camellia untuk mengembalikan data ke bentuk plaintext.

3. Serangan SQL Injection

Pengujian terhadap serangan *SQL Injection* dilakukan sebagai tahapan untuk menguji seberapa kuat sistem yang telah dikembangkan dalam menghadapi celah keamanan pada input basis data. Pada tahap ini, digunakan aplikasi SQLMap sebagai alat bantu untuk mensimulasikan percobaan ekstraksi data dari tabel pasien secara tidak sah, tanpa melalui proses otorisasi pengguna. Berdasarkan hasil *data dumping* yang disajikan pada Gambar 10, diketahui bahwa kolom yang berisi data penting seperti nomor identitas, nama lengkap, dan alamat, tetap dalam format terenkripsi. Hal ini menunjukkan bahwa skema enkripsi dua tahap yang mengombinasikan algoritma Camellia dan RSA telah berhasil diimplementasikan dengan baik pada sistem penyimpanan data.



GAMBAR 12

Pengujian ini bertujuan untuk mengevaluasi sejauh mana sistem mampu bertahan terhadap salah satu jenis serangan siber yang paling umum terjadi, yaitu *SQL Injection*. Penerapan enkripsi Camellia sebagai enkripsi simetris dan RSA sebagai enkripsi asimetris diuji secara menyeluruh dalam lingkungan yang terkendali, menggunakan pendekatan sistematis dan terstruktur. Temuan dalam tahapan ini menjadi bukti empiris terhadap ketangguhan skema enkripsi dua lapis tersebut. Melalui penyisipan query SQL berbahaya secara terencana dalam sistem yang telah dipersiapkan untuk pengujian, dilakukan pengamatan secara teliti guna menilai efektivitas dari skema enkripsi yang diterapkan. Hasil pengujian menunjukkan bahwa kombinasi enkripsi Camellia dan RSA mampu mencegah kebocoran informasi penting dan melindungi integritas basis data dari upaya eksploitasi. Hal ini mengindikasikan bahwa pendekatan keamanan yang digunakan tidak hanya bersifat teoritis, tetapi juga memiliki nilai penerapan yang tinggi dalam konteks sistem informasi yang rentan terhadap serangan seperti *SQL Injection*.

V. KESIMPULAN

Penelitian ini berhasil mengimplementasikan dan membuktikan keberhasilan metode enkripsi dua tahap menggunakan kombinasi algoritma Camellia dan RSA untuk memperkuat keamanan database pada aplikasi berbasis web. Hasil pengujian menunjukkan bahwa pendekatan ini memberikan lapisan pertahanan yang signifikan, terutama dalam menghadapi ancaman SQL Injection. Berdasarkan hasil *data dumping* terhadap tabel basis data, data sensitif milik pasien seperti nomor identitas, nama lengkap, dan alamat tetap terlindungi dalam bentuk *ciphertext* yang tidak dapat diinterpretasikan secara langsung. Temuan ini menunjukkan bahwa perlindungan pada tingkat data mampu menjaga kerahasiaan informasi secara efektif, meskipun sistem telah berhasil disusupi pada level aplikasi. Dengan kata lain, penerapan enkripsi dua tahap ini tidak hanya mencegah akses ilegal terhadap data, tetapi juga memberikan jaminan terhadap integritas dan kerahasiaan informasi saat terjadi pelanggaran keamanan. Hal ini membuktikan bahwa kombinasi antara algoritma enkripsi simetris yang efisien dan algoritma asimetris yang kokoh merupakan pendekatan yang andal dalam menjaga keamanan sistem yang menangani informasi bersifat sensitif.

REFERENSI

[1] N. S. Nainggolan and I. P. Nasution, “Pentingnya Keamanan Big Data Dalam Lembaga Pemerintahan Di Era Digital,” *J. Sains dan Teknol.*, vol. 3, no. 2, pp. 253–257, 2023.

[2] S. M. T. Situmeang, “PENYALAHGUNAAN DATA PRIBADI SEBAGAI BENTUK KEJAHATAN SEMPURNA DALAM PERSPEKTIF HUKUM SIBER,” *SASI*, vol. 27, no. 1, p. 38, 2021.

[3] E. Budi, D. Wira, and A. Infantono, “Strategi Penguatan Cyber Security Guna Mewujudkan Keamanan Nasional di Era Society 5.0,” *Pros. Semin. Nas. Sains Teknol. dan Inov. Indones.*, vol. 3, no. November, pp. 223–234, 2021.

[4] I. Riadi, R. Umar, and W. Sukarno, “Analisis Forensik Serangan Sql Injection Meanggunakan Metode Statis Forensik,” *Pros. Interdiscip. Postgrad. Student Conf. 1st*, vol. I, no. I, pp. 102–103, 2016.

[5] R. Jahanshahi, A. Doupé, and M. Egele, “You shall not pass: Mitigating SQL Injection Attacks on Legacy Web Applications,” *Proc. 15th ACM Asia Conf. Comput. Commun. Secur. ASIA CCS 2020*, pp. 445–457, 2020.

[6] A. Rahman, “Perancangan Aplikasi Pengamanan File Pada Memory Card Handphone Menggunakan Algoritma Kunci Asimetris Elgamal,” *JURIKOM (Jurnal Ris. Komputer)*, vol. 6, no. 5, pp. 531–537, 2019.

[7] A. H. Kridalaksana, A. Y. Rangan, and A. Ansharie, “Enkripsi Data Audio Menggunakan Metode Kriptografi Rsa,” *Sebatik*, vol. 17, no. 1, pp. 6–10, 2017.

[8] S. Ikhwan and R. F. Christianti, “Penerapan Keamanan WSN Berbasis Algoritma RSA 2048 dan SHA-3 pada Pemantauan Suhu,” *J. Nas. Teknol. dan Sist. Inf.*, vol. 6, no. 3, pp. 150–157, 2021.

[9] R. Pamungkas and F. W. Z. Zaney, "Penerapan Hashing SHA1 dan Algoritma Asimetris RSA untuk Keamanan Data pada Sistem Informasi berbasis Web," *Res. J. Comput. Inf. Syst. Technol. Manag.*, vol. 4, no. 1, p. 84, 2021.

[10] M. W. Saputra, A. Sapitri, and M. A. Putri, "Penerapan Kriptosistem Hybrid Untuk Mengenkripsi Pesan Menggunakan Algoritma Rsa Cipher," *J. JOCOTIS-Journal Sci. Inform. Robot. E-ISSN xxxx-xxxx*, vol. 1, no. 1, pp. 10–21, 2023.

[11] L. N. Hidayati, G. F. Fitriana, and I. F. Adam, "Perbandingan Keacakan Citra Enkripsi Algoritma AES dan Camelia Uji NPCR dan UACI," *JURIKOM (Jurnal Ris. Komputer)*, vol. 8, no. 6, p. 274, 2021.

[12] D. G. Ryandika and W. A. Prabowo, "Two-Stage Encryption for Strengthening Data Security in Web-Based Databases: AES-256 and RSA Integration," *Proceeding - COMNETSAT 2023 IEEE Int. Conf. Commun. Networks Satell.*, pp. 486–492, 2023.

[13] E. Tarigan and D. H. S. Maha, "Kombinasi Vigenere Cipher Dan Polyalphabetic Cipher Pada Pengamanan File Textfile:///C:/Users/HP/Downloads/j.jisa.2021.102778.pdf," *Publ. Ilm. ...*, pp. 71–77, 2018.

[14] Z. Arif and A. Nurokhman, "Analisis Perbandingan Algoritma Kriptografi Simetris Dan Asimetris Dalam Meningkatkan Keamanan Sistem Informasi," *Jtsi*, vol. 4, no. 2, pp. 394–405, 2023.

[15] H. Zuhri and H. Kristian Siburian, "Implementasi Metode Camellia Dalam Keamanan Data File Berekstensi Txt Dan Doc," *J. Pelita Inform.*, vol. 6, no. 2, pp. 223–232, 2017.

[16] M. S. Asih, P. Studi, T. Informatika, and U. H. Medan, "Implementasi Algoritma Kriptografi RSA Dalam Aplikasi Sistem Informasi Perpustakaaan," pp. 214–223, 2023.

[17] D. Email, A. Ginting, R. R. Isnanto, and I. P. Windasari, "Implementasi Algoritma Kriptografi RSA untuk Enkripsi dan Dekripsi Email," vol. 3, no. 2, pp. 253–258, 2015.

[18] S. Siregar, "Implementasi Mode Operasi Cipher Block Chaining (CBC) Untuk Mengoptimalkan Algoritma Affine Cipher Dalam Pengamanan Data," *Bull. Inf. Syst. Res.*, vol. 1, no. 3, pp. 99–109, 2023.

[19] I. S. Crespo-Martínez, A. Campazas-Vega, Á. M. Guerrero-Higueras, V. Riego-DelCastillo, C. Álvarez-Aparicio, and C. Fernández-Llamas, "SQL injection attack detection in network flow data," *Comput. Secur.*, vol. 127, 2023, doi: 10.1016/j.cose.2023.103093.

[20] M. A. Saputra, H. H. Kusuma, and A. Ibrahim, "Mengatasi Keamanan di dalam SQL Injection dan Cara Pencegahannya," *Pros. Annu. Res. Semin. 2017 Comput. Sci. ICT ISBN*, vol. 3, no. 1, pp. 105–108, 2017.