

**Peningkatan Keamanan Transmisi *Data* RFID dengan
Penerapan Enkripsi XOR Sederhana pada Mikrokontroler ESP-
32 untuk Pencegahan Serangan *Eavesdropping***

Tugas Akhir

diajukan untuk memenuhi salah satu syarat

memperoleh gelar sarjana

dari Program Studi S1 Informatika

Fakultas Informatika Universitas

Telkom

1301204195

Dzaky Audizha Patarai



Program Studi Sarjana Informatika

Fakultas Informatika

Universitas Telkom

Bandung

2024

1.1. LEMBAR PENGESAHAN

Peningkatan Keamanan Transmisi *Data* RFID dengan Penerapan Enkripsi XOR Sederhana pada Mikrokontroler ESP-32 untuk Pencegahan Serangan *Eavesdropping*

Enhanced RFID Data Transmission Security by Implementing Simple XOR Encryption on ESP-32 Microcontroller for Eavesdropping Attack Prevention

NIM : 1301204195

Dzaky Audizha Patarai

Tugas akhir ini telah diterima dan disahkan untuk memenuhi sebagian syarat memperoleh gelar pada Program Studi Sarjana Informatika

Fakultas Informatika
Universitas Telkom

Bandung, 9 September 2024

Menyetujui

Pembimbing I,



Dr. Fazmah Arif Yulianto, S.T., M.T.
NIP: 99750034

Pembimbing II,



Febri Dawani, S.T., M.T.
NIP: 20850005

Ketua Program Studi
Sarjana Informatika,



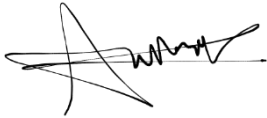
Dr. Erwin Budi Setiawan, S.Si., M.T.
NIP: 99750047

1.2. LEMBAR PERNYATAAN

Dengan ini saya, Dzaky Audizha Patarai, menyatakan sesungguhnya bahwa Tugas Akhir saya dengan judul Peningkatan Keamanan Transmisi *Data* RFID dengan Penerapan Enkripsi XOR sederhana pada Mikrokontroller ESP-32 untuk Pencegahan Serangan *Eavesdropping* beserta dengan seluruh isinya adalah merupakan hasil karya sendiri, dan saya tidak melakukan penjiplakan yang tidak sesuai dengan etika keilmuan yang berlaku dalam masyarakat keilmuan. Saya siap menanggung risiko/sanksi yang diberikan jika di kemudian hari ditemukan pelanggaran terhadap etika keilmuan dalam buku TA atau jika ada klaim dari pihak lain terhadap keaslian karya,

Bandung, 9 September 2024

Yang Menyatakan



Dzaky Audizha Patarai

Daftar Isi

1.1.	LEMBAR PENGESAHAN	1
1.2.	LEMBAR PERNYATAAN	2
	Abstrak	5
	Abstract	6
	BAB 1	7
	Pendahuluan	7
1.1.	Latar Belakang	7
1.2.	Topik dan Batasannya	8
1.3.	Tujuan	8
1.4.	Organisasi Tulisan	8
	BAB 2	9
	Kajian Pustaka	9
2.1.	Studi Terkait	9
2.2.	Enkripsi	9
2.3.	<i>Man In The Middle (MitM)</i>	9
2.4.	<i>Eavesdropping</i>	10
2.5.	<i>Synchronized secret</i>	10
2.6.	<i>Simple XOR encryption</i>	10
	BAB 3	13
	Perancangan Sistem	13
3.1.	Sistem yang Dibangun	13
3.2.	<i>Requirement Sistem</i>	14
3.3.	Skematik Sistem yang Dibangun	15
3.4.	Struktur tabel <i>database</i>	15
3.5.	Desain Sistem	16
3.5.1.	Desain Sistem Registrasi	16
3.5.2.	Desain Sistem <i>Tapping</i>	18
3.5.3.	<i>Flowchart</i> Enkripsi XOR	21
3.5.4.	Validasi enkripsi XOR pada sistem	22
3.6.	Proses Key exchange	23
3.7.	Skenario Pengujian	23
3.7.1.	Pengujian <i>Eavesdropping</i> Jaringan oleh <i>Attacker</i> Tanpa metode Enkripsi XOR sederhana	23
3.7.2.	Pengujian <i>Data</i> RFID Menggunakan Metode Enkripsi XOR sederhana	24
	BAB 4	25
	Evaluasi	25

4.1.	Hasil Pengujian Dan Analisis	25
4.1.1.	Hasil pengujian <i>Eavesdropping</i> Jaringan oleh <i>Attacker</i> Tanpa Metode Enkripsi XOR sederhana	25
4.1.2.	Hasil Pengujian <i>Data</i> RFID Menggunakan Metode Enkripsi XOR sederhana.....	26
4.2.	Hasil analisis pengujian.....	27
BAB 5	29
Kesimpulan dan Saran	29
5.1.	Kesimpulan	29
5.2.	Saran.....	29

Abstrak

Radio Frequency Identification (RFID) telah menjadi populer dalam berbagai aplikasi, seperti sistem pembayaran, pengidentifikasian akses, dan pelacakan inventaris. Kartu RFID memiliki kaitan dengan teknologi *Near Field Communication* (NFC). NFC adalah teknologi nirkabel yang memungkinkan pertukaran *data* antara perangkat elektronik yang berdekatan secara fisik melalui gelombang radio frekuensi rendah. Kartu RFID rentan terhadap risiko keamanan karena informasinya dapat dibaca dengan mudah oleh pembaca RFID di sekitar.

Serangan *eavesdropping* terjadi ketika seseorang mencuri informasi dari kartu RFID dan membuat kartu yang identik dengan aslinya atau di sebut juga *cloning*. Menurut penelitian sebelumnya, sebuah sistem telah diimplementasikan menggunakan metode *synchronized secrets* untuk mendeteksi *cloning* secara otomatis pada kartu RFID. Hal ini memungkinkan pengguna kartu RFID asli mengetahui jika kartunya di *clone*, belum ada solusi yang sepenuhnya efektif dalam menjamin keamanan dan kerahasiaan informasi pada kartu RFID yang digunakan.

Untuk mengatasi masalah ini, solusi yang telah dikembangkan adalah mengimplementasikan sistem enkripsi *Simple XOR* pada ESP-32 pada proses pengiriman *synchronized secrets* ke *server*. Hasil Penelitian ini menunjukkan bahwa sistem Enkripsi *Simple XOR* dapat meningkatkan keamanan transmisi *data* RFID dan mencegah serangan *eavesdropping* pada mikrokontroler ESP-32.[1].

Kata kunci : RFID,*Eavesdropping*,*Synchronized secret*,*Encryption*,*Simple XOR encryption*

Abstract

Radio Frequency Identification (RFID) has become popular in various applications, such as payment systems, access identification, and inventory tracking. RFID cards are related to Near Field Communication (NFC) technology. NFC is a wireless technology that enables *data exchange* between physically adjacent electronic devices via low-frequency radio waves. RFID cards are susceptible to security risks because their information can be read easily by nearby RFID readers.

An *eavesdropping* attack occurs when someone steals information from an RFID card and creates a card that is identical to the original or also called *cloning*. According to previous research, a system has been implemented using the *Synchronized secrets* method to automatically detect *cloning* on RFID cards. This allows the original RFID card user to know if his card is cloned, there is no solution that is fully effective in ensuring the security and confidentiality of the information on the RFID card used.

To overcome this problem, the solution that has been developed is to implement a *Simple XOR encryption* system on the ESP-32 in the process of sending *synchronized secrets* to the server. The results of this study show that the *Simple XOR encryption* system can increase the security of RFID data transmission and prevent *eavesdropping* attacks on the ESP-32 microcontroller.

Keywords: RFID, *eavesdropping*, *synchronized secret*, *encryption*, *simple XOR encryption*

BAB 1

Pendahuluan

Radio Frequency Identification (RFID) telah menjadi teknologi yang sering digunakan dalam berbagai aplikasi seperti pembayaran, akses identifikasi, dan pelacakan inventaris. Pengguna hanya perlu menyentuh atau mendekatkan perangkat ke *tag* NFC atau perangkat lain yang kompatibel untuk melakukan pembayaran digital, membuka pintu, mengunduh informasi, dan berbagi kontak. NFC telah menjadi teknologi yang umum dalam kehidupan sehari-hari, mempermudah interaksi antar perangkat elektronik dengan cepat dan mudah. Namun, informasi dalam kartu RFID dapat dengan mudah dibaca oleh pembaca RFID di sekitarnya, sehingga ada risiko keamanan yang besar [1], [2].

Sistem enkripsi dapat membantu menjaga kerahasiaan informasi dalam kartu RFID sehingga hanya dapat dibaca oleh pembaca RFID yang memiliki kunci enkripsi yang sesuai. Menurut tugas akhir yang diterbitkan oleh Deti Dwi Arisandi, Fazmah Arif Yulianto, dan Andrian Rakhmatsyah pada tahun 2021, sistem dengan metode *Synchronized secrets* dapat mendeteksi *cloning* secara otomatis. Dengan ini, pengguna asli kartu RFID akan mengetahui jika kartunya terindikasi *cloning* dan dapat memblokirnya secara online dengan cepat. Meskipun sudah ada solusi untuk masalah ini, belum ada yang sepenuhnya efektif dalam menjaga keamanan dan kerahasiaan informasi pada kartu RFID [1].

Oleh karena itu, penelitian ini bertujuan mengembangkan teknologi untuk mencegah serangan *eavesdropping* selama pengiriman *Synchronized secrets* antara RFID reader dan server agar masalah keamanan ini dapat diatasi lebih baik. Penelitian ini mengusulkan untuk menggunakan *Simple XOR Encryption*, enkripsi *Simple XOR* ini dipilih dikarenakan metode ini kompatibel dengan *microcontroller ESP-32*. Maka dari itu untuk mengamankan komunikasi antara server dan klien, metode ini memastikan data yang dikirimkan melalui HTTP terenkripsi dan terlindungi dari penyadapan dan serangan. Hal ini memungkinkan server HTTP untuk melindungi informasi sensitif pengguna, seperti data login dan informasi pribadi, dari akses yang tidak sah, sekaligus memenuhi standar keamanan dan regulasi industri yang mewajibkan penggunaan enkripsi yang kuat. Dengan demikian, *simple XOR encryption* memainkan peran penting dalam meningkatkan keamanan server berbasis HTTP dan memastikan komunikasi yang aman.

1.1. Latar Belakang

Radio Frequency Identification (RFID) telah menjadi salah satu teknologi yang populer digunakan dalam banyak aplikasi, termasuk dalam sistem pembayaran, identifikasi akses, dan tracking inventaris [1]. Kartu RFID ini juga mempunyai hubungan dengan *Near Field Communication* (NFC). *Near Field Communication* (NFC) adalah teknologi nirkabel yang memfasilitasi pertukaran data antara perangkat elektronik yang berdekatan secara fisik melalui gelombang radio frekuensi rendah. NFC digunakan dalam berbagai aplikasi seperti pembayaran elektronik, transfer data, akses kebangkitan, dan lainnya [2]. Dengan hanya menyentuh atau mendekatkan perangkat ke *NFC tag* atau perangkat lain yang kompatibel, pengguna dapat melakukan tindakan seperti membayar dengan metode digital, membuka pintu, mengunduh informasi, dan berbagi kontak [3].

Dalam kehidupan sehari-hari, NFC telah menjadi teknologi yang luas digunakan, mempermudah interaksi antara perangkat elektronik dengan cepat dan mudah. Namun, karena informasi yang disimpan dalam kartu RFID dapat dengan mudah dibaca oleh pembaca RFID yang ada di sekitar, terdapat risiko keamanan yang signifikan. Serangan *cloning* pada RFID terjadi ketika pencuri mencuri informasi yang tersimpan pada kartu RFID, dan kemudian membuat kartu RFID yang identik dengan kartu yang asli. Pada akhirnya, *attacker* dapat menggunakan kartu RFID yang diduplikasi ini untuk memperoleh akses tidak sah ke sistem atau aset yang diidentifikasi oleh kartu RFID. Untuk mengatasi masalah ini, banyak solusi telah dikembangkan, salah satunya adalah dengan mengimplementasikan sistem enkripsi pada kartu RFID [2], [3], [4].

Sistem enkripsi ini akan membantu dalam menjaga kerahasiaan informasi yang disimpan dalam kartu RFID sehingga hanya dapat dibaca oleh pembaca RFID yang memiliki kunci enkripsi yang sesuai. Menurut T yang diterbitkan oleh Deti Dwi Arisandi, Fazmah Arif Yulianto dan Andrian Rakhmatsyah 2021, mengimplementasikan sistem menggunakan metode *Synchronized secrets* untuk mendeteksi *cloning* secara otomatis, sehingga 5 pengguna asli kartu RFID mengetahui jika kartunya terindikasi *cloning* serta dapat menerapkan sistem blokir secara online jika kartu RFID terindikasi *cloning* dengan waktu yang efisien. Meskipun telah dikembangkan solusi untuk mengatasi masalah ini, namun belum ada solusi yang sepenuhnya efektif dalam memastikan keamanan dan kerahasiaan informasi pada kartu RFID yang digunakan pada RFID. Oleh karena itu, dalam penelitian ini, kami berupaya untuk mengembangkan teknologi pencegah serangan *eavesdropping* pada proses pengiriman *synchronized secrets* yang dapat mengatasi masalah keamanan ini dengan lebih efektif [1], [4].

Maka dari itu motivasi dari penelitian ini yaitu menjaga kerahasiaan informasi yang tersimpan di RFID card dalam proses transmisi antara *microcontroller* dan server dari referensi tugas akhir sebelumnya, mikrokontroler menawarkan solusi ideal untuk aplikasi RFID tag reader dengan keunggulan ukurannya yang kecil, dan harganya yang terjangkau. Selain itu, fitur konektivitas nirkabel yang kuat (Wi-Fi) dari ESP-32 memungkinkan integrasi

yang mudah dan fleksibel dengan jaringan yang ada, menjadikannya pilihan unggul untuk aplikasi *IoT* yang memerlukan komunikasi *data* yang andal dan *real-time*.

1.2. Topik dan Batasannya

Permasalahan yang dibahas dalam tugas akhir ini bagaimana merancang sistem yang dapat mencegah dari serangan *Eavesdropping* atau *Man-in-the-middle (MitM)*.

Batasan dari penelitian tugas akhir ini adalah RFID digunakan untuk memberi akses ke *user* atau pengguna yang sah, contohnya seperti mahasiswa telkom yang di berikan akses untuk ke parkir dan gedung. Secara umum contoh kasus lainnya seperti, hanya pengguna yang sah dapat mengakses ruangan tertentu atau ruangan yang bersifat rahasia dan hanya bisa dimasuki oleh orang-orang tertentu. Penelitian ini berupaya untuk meningkatkan keamanan transmisi data RFID ke *server*. Transmisi sistem ini menggunakan protokol HTTP, sehingga data yang di transmisikan dapat di lihat oleh *attacker*. Maka proses pengiriman *synchronized secret* dari server ke RFID reader diasumsikan dapat terbaca oleh *attacker* menggunakan teknik *eavesdropping* dikarenakan *attacker* mampu melakukan *Sniffing* atau *Cloning*. Dari teknik ini *attacker Man-in-the-middle (MitM)* dapat melakukan *Sniffing* atau *Cloning* yang bertujuan agar *synchronized secret* yang telah di enkripsi ini dapat di gunakan untuk di berikan akses oleh *server*.

1.3. Tujuan

Tujuan dari penelitian ini adalah dapat mengimplementasikan metode enkripsi untuk mencegah dari serangan *Eavesdropping* melalui proses pengiriman *Synchronized secrets* antara RFID reader dan *server*. Sehingga pada proses pengiriman *Synchronized secrets* dapat dipastikan aman dan tidak dapat terbaca *Man-in-the-Middle (MitM)*. Secara khusus, penelitian ini berfokus pada proses enkripsi antara RFID reader dan *server*, sehingga *attacker* tidak dapat melihat *payload* yang ditransmisikan oleh RFID reader ke *server* maupun sebaliknya.

1.4. Organisasi Tulisan

Penelitian ini disusun dalam lima bab. Bab 1 berisi pendahuluan yang mencakup latar belakang masalah, batasan topik, dan tujuan penelitian, yang berfokus pada peningkatan keamanan transmisi data RFID dengan metode enkripsi XOR sederhana. Bab 2 membahas kajian pustaka yang mencakup konsep-konsep terkait seperti teknologi RFID, serangan *eavesdropping*, enkripsi simetris, dan metode XOR. Bab 3 menjelaskan perancangan sistem, mulai dari kebutuhan perangkat keras dan perangkat lunak, desain sistem, hingga validasi enkripsi XOR yang diterapkan pada sistem. Bab 4 menguraikan hasil evaluasi dan analisis dari pengujian keamanan sistem terhadap serangan *eavesdropping* dengan dan tanpa enkripsi XOR sederhana. Terakhir, Bab 5 menyajikan kesimpulan dari hasil penelitian ini serta saran untuk pengembangan lebih lanjut dalam upaya meningkatkan keamanan sistem RFID.

BAB 2

Kajian Pustaka

2.1. Studi Terkait

Radio Frequency Identification (RFID) kini merupakan teknologi yang banyak digunakan dalam berbagai bidang, seperti sistem pembayaran, akses kontrol, dan pelacakan inventaris. Dengan RFID, pengguna dapat melakukan berbagai aktivitas seperti pembayaran digital, membuka pintu, mengunduh informasi, dan berbagi kontak dengan mudah. Teknologi RFID telah menjadi bagian tak terpisahkan dari kehidupan sehari-hari, karena mempermudah interaksi cepat dan efisien antara perangkat elektronik[3].

Namun, informasi yang disimpan dalam kartu RFID dapat dengan mudah dibaca oleh pembaca RFID di sekitarnya, sehingga menimbulkan risiko keamanan yang signifikan. Penyerang dapat menggunakan kartu RFID yang diduplikasi untuk memperoleh akses tidak sah ke sistem atau aset yang diidentifikasi oleh kartu RFID[1]. Untuk mengatasi masalah ini, berbagai solusi telah dikembangkan, salah satunya adalah dengan mengimplementasikan sistem *synchronized secrets* pada kartu RFID. Menurut tugas akhir yang diterbitkan oleh Deti Dwi Arisandi, Fazmah Arif Yulianto, dan Andrian Rakhmatsyah pada tahun 2021, metode *Synchronized secrets* dapat mendeteksi *cloning* secara otomatis. Dengan demikian, pengguna asli kartu RFID akan mengetahui jika kartunya terindikasi *cloning* dan dapat menerapkan sistem blokir secara online dengan waktu yang efektif[1].

Maka dari itu penelitian ini berupaya untuk mengembangkan sistem yang dapat mencegah serangan yang memungkinkan seperti *eavesdropping*, *Man-in-the-middle*, *Sniffing*, dan *Clone*. Agar informasi yang tertampung di dalam RFID card tersebut tidak dapat terbaca oleh *attacker*.

2.2. Enkripsi

Enkripsi adalah proses mengubah informasi atau *data* menjadi bentuk yang tidak dapat dibaca atau di mengerti oleh pihak yang tidak memiliki kunci untuk membuka informasi tersebut. Tujuan dari enkripsi adalah untuk menjaga keamanan dan kerahasiaan *data*, sehingga hanya pihak yang berwenang yang dapat mengakses atau membaca informasi tersebut[4], [5].

Proses enkripsi dilakukan dengan menggunakan suatu algoritma atau kunci enkripsi yang akan mengubah teks biasa atau *data* menjadi teks yang diacak atau teracak. Ada dua jenis enkripsi yang umum digunakan yaitu enkripsi simetris dan enkripsi asimetris. Enkripsi simetris menggunakan satu kunci yang sama untuk melakukan enkripsi dan dekripsi[6]. Artinya, siapa saja yang memiliki kunci tersebut dapat melakukan enkripsi dan dekripsi pada *data* tersebut. Namun, enkripsi asimetris menggunakan kunci publik dan privat. Kunci publik digunakan untuk mengenkripsi *data*, dan kunci privat digunakan untuk mendekripsinya, sedangkan kunci privat digunakan untuk mendekripsi *data* tersebut[5], [6].

Enkripsi dalam penelitian ini menggunakan metode kriptografi simetris, kriptografi Simetris adalah jenis kriptografi yang menggunakan satu kunci yang sama untuk melakukan proses enkripsi dan dekripsi. Dalam kriptografi simetris, pengirim dan penerima pesan menggunakan kunci yang sama untuk menjaga kerahasiaan pesan[6]. Proses enkripsi mengubah pesan asli menjadi bentuk yang tidak dapat dibaca atau dimengerti, sedangkan proses dekripsi mengembalikan pesan tersebut ke bentuk aslinya [4].

Kelebihan utama dari kriptografi simetris adalah kecepatannya dan efisiensinya. Algoritma enkripsi simetris umumnya lebih cepat daripada algoritma enkripsi asimetris, sehingga cocok untuk mengenkripsi dan mendekripsi *data* dalam jumlah besar. Selain itu, implementasi kriptografi simetris relatif sederhana dan membutuhkan sumber daya yang lebih sedikit dibandingkan dengan kriptografi asimetris[11].

Namun, tantangan dalam kriptografi simetris adalah masalah distribusi kunci yang aman. Kunci enkripsi simetris harus disimpan dengan baik dan hanya diberikan kepada pihak yang berwenang. Jika kunci jatuh ke tangan yang salah, maka kerahasiaan pesan dapat terancam. Oleh karena itu, diperlukan mekanisme yang aman untuk mengelola dan mendistribusikan kunci enkripsi simetris agar hanya pihak yang berwenang yang dapat menggunakannya[4], [11].

2.3. Man In The Middle (MitM)

Man in the Middle (MitM) adalah jenis serangan keamanan siber di mana penyerang secara rahasia memasukkan diri mereka ke dalam komunikasi dua pihak, mengintersepsi dan memodifikasi atau merekam *data* yang ditransmisikan antara mereka tanpa pengetahuan kedua pihak tersebut. Ini sering dilakukan dengan mengambil alih sesi komunikasi, seperti pada jaringan Wi-Fi publik yang tidak aman, dimana penyerang dapat menyamar sebagai titik akses jaringan untuk menangkap dan mengubah informasi yang dikirimkan. Penyerang juga dapat menggunakan teknik *eavesdropping* atau *sniffing* DNS untuk mengarahkan korban ke situs *web* palsu, dimana informasi sensitif seperti kredensial *login* dapat dicuri. MitM sangat berbahaya karena memungkinkan akses langsung ke *data* sensitif dan dapat digunakan untuk berbagai tujuan jahat, termasuk pencurian identitas, penipuan keuangan, atau pengintai korporat. Karena kompleksitasnya, serangan MitM sering sulit dideteksi dan memerlukan langkah keamanan yang kuat, seperti enkripsi *end-to-end* dan autentikasi yang aman, untuk mencegahnya[7].

Pada serangan *Man-in-the-Middle* (MITM), penyerang mengambil alih komunikasi jaringan yang sedang

berlangsung antara dua atau lebih saluran. Serangan ini adalah serangan berbasis jaringan di mana peretas menguping semua lalu lintas antara korban dan *server* dalam *subnet* yang sama. Sebaliknya, korban tidak menyadari bahwa sesi tersebut telah dibajak dan menganggap bahwa saluran komunikasi tersebut aman[8].

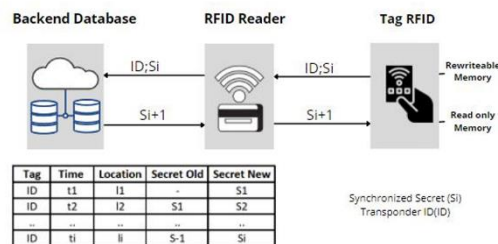
Untuk melancarkan serangan semacam ini, alat seperti Ettercap dan Wireshark sering digunakan. Ettercap adalah alat open-source yang memungkinkan pelaksanaan serangan MitM dengan teknik seperti ARP poisoning dan DNS spoofing, di mana penyerang dapat mencegat, memodifikasi, dan menyisipkan data dalam lalu lintas jaringan. Ettercap sering digunakan dalam pengujian penetrasi untuk menguji kerentanan jaringan terhadap serangan semacam ini, namun harus digunakan dengan etika dan legalitas yang tepat. Di sisi lain, Wireshark adalah alat analisis protokol jaringan yang sangat kuat, digunakan untuk menangkap, menganalisis, dan memecahkan masalah lalu lintas jaringan secara mendalam. Meskipun Wireshark berfokus pada pemantauan dan analisis jaringan, alat ini juga bisa digunakan untuk mendeteksi aktivitas mencurigakan yang mungkin terkait dengan serangan MitM. Kedua alat ini memiliki peran penting dalam memahami dan mengelola keamanan jaringan, serta dalam mengidentifikasi dan mencegah serangan MitM.

2.4. Eavesdropping

Eavesdropping adalah teknik serangan siber dimana seorang penyerang secara rahasia mendengarkan komunikasi pribadi antara dua pihak tanpa sepengetahuan mereka. Biasanya terjadi dalam jaringan komputer, serangan ini melibatkan penyadapan *data* yang ditransmisikan melalui jaringan, seperti *email*, pesan instan, atau percakapan VoIP. Penyerang menggunakan alat khusus untuk menangkap dan menganalisis paket *data* yang bergerak antar jaringan, mencari informasi sensitif seperti kata sandi, detail kartu kredit, atau rahasia perdagangan[9]. Serangan *eavesdropping* sering kali sulit dideteksi karena tidak mengganggu atau mengubah *data* yang ditransmisikan, menjadikannya ancaman yang berbahaya di lingkungan jaringan yang tidak aman, khususnya pada jaringan yang tidak menggunakan enkripsi atau memiliki implementasi keamanan yang lemah[10].

2.5. Synchronized secret

Synchronized secret adalah sebuah protokol komunikasi antara RFID *tag* dan *Server* melalui RFID *reader*.metode protokol ini telah di digunakan oleh Deti dwi arisandi di Tugas Akhir sebelumnya,menurut Deti metode ini dapat di gunakan sebagai pendeteksi adanya *clonning* terhadap kartu RFID[1].



Gambar 1. Ilustrasi dari metode *Synchronized secret* [1]

Terlihat Gambar 1 adalah ilustrasi metode *Synchronized secret*. Metode ini diusulkan oleh Mikko Lehtonen, Daniel Ostojic, Alexander Ilic dan Florian Michahelles, untuk menggunakan memori *tag* yang dapat ditulis ulang. angka *random* (*pseudo random*) ditulis ke dalam memori yang berubah setiap kali *tag* dipindai. Angka acak ini hanya diketahui oleh *database* dan *backend* pusat *tag*. Lehtonen menyebut metode ini “*Synchronized secret*”. Selama proses sinkronisasi awal, *tag* dipindai dan UID dibaca oleh RFID *reader* dan dibandingkan dengan *data* di *backend*. Jika ada yang cocok atau *valid*, maka *backend* memberikan nomor acak baru (*pseudo-random number*) di memori *tag*, yang menjadi kunci sinkronisasi antara *tag* dan *backend*[1], [12].

2.6. Simple XOR encryption

Simple XOR Encryption adalah teknik enkripsi simetris dasar yang menggunakan operasi *Exclusive OR* (XOR) untuk mengenkripsi dan mendekripsi *data*. Dalam metode ini, setiap *bit* dari *plaintext* di-XOR dengan *bit* yang sesuai dari kunci untuk menghasilkan *ciphertext*. Kunci yang sama digunakan untuk mendekripsi *Ciphertext* dengan meng-XOR kembali dengan kunci yang sama, sehingga membalikkan operasi[13]. Teknik ini bergantung pada sifat-sifat operasi XOR, di mana bit yang di-XOR dengan dirinya sendiri menghasilkan 0 dan bit yang di-XOR dengan 0 menghasilkan bit itu sendiri. Enkripsi XOR sederhana mudah dan cepat, tetapi keamanannya sangat bergantung pada kerahasiaan dan keacakan kunci, jika kunci pendek atau digunakan kembali enkripsi dapat dengan mudah dipecahkan[14]. Berikut Contoh Tabel kebenaran XOR pada tabel 1 berikut.

Tabel 1. Tabel kebenaran XOR

A	B	\oplus
0	0	0
0	1	1
1	0	1
1	1	0

Pada Tabel 1 operasi logika *biner* yang menghasilkan nilai *true* atau 1 jika dan hanya jika dua operasi nya berbeda, XOR bekerja dengan membandingkan *bit-bit* dari dua angka biner. Berikut Penjelasan Tabel kebenaran XOR.

- 0 XOR 0 = 0: Kedua bit sama (0), hasilnya 0.
- 0 XOR 1 = 1: Kedua bit berbeda, hasilnya 1.
- 1 XOR 0 = 1: Kedua bit berbeda, hasilnya 1.
- 1 XOR 1 = 0: Kedua bit sama (1), hasilnya 0.

Maka dari itu XOR adalah operasi yang dapat dibalik dan *data* asli dapat di *decrypt* dengan kunci yang sama. Pada penelitian ini XOR digunakan untuk melakukan proses enkripsi ataupun dekripsi, berikut adalah contoh proses implementasi metode XOR dalam enkripsi dapat dilihat pada gambar berikut.

Plain Text	=	A	Key	=	F
Decimal	=	65	Decimal	=	70
Binary	=	1000001	Binary	=	1000110

Gambar 2. Contoh *plain text* dan *key encryption*

Gambar 2 menunjukan *plain text* ASCII yang di *input* adalah huruf “A” , sedangkan kunci yang di gunakan adalah huruf “F”. Berikutnya Huruf “A” tersebut akan di enkripsi XOR menggunakan Key “F”, proses ini akan melibatkan *binary* dari *plain text* dan *key encryption* yang digunakan. Proses ini akan di jelaskan di gambar berikut.

Plain Text	=	A
Decimal	=	65
Binary	=	1000001
Binary Key	=	1000110 \oplus F
<hr/>		
Binary XOR'ed	=	0000111
Decimal	=	7
Hex	=	7

Gambar 3. Proses *Encrypt XOR*

Gambar 3 menjelaskan proses enkripsi XOR yang dilakukan setelah mendapatkan *Key encryption* yang di gunakan, yaitu huruf “F”. Proses ini dilakukan ketika *binary* dari *plain text* “A” dan *key encryption* “F” dilakukan XOR, sehingga jadilah “*binary XOR'ed*” yang dimana *binary* nya “0000111” yang di konversi dari desimal ke *hexa* menjadi angka “7”. Ketika angka *biner* telah di XOR, maka selanjut nya akan ada proses dekripsi yang dimana angka *biner* yang telah di XOR akan di XOR kembali menggunakan *key* yang sama, yaitu *key* “F”.

Hex	=	7
Decimal	=	7
Binary XOR'ed	=	0000111
binary Key	=	1000110 \oplus F
Decipher Binary	=	1000001
Hex	=	41
Decimal	=	65
Plain Text	=	A

Gambar 4. Proses *Decrypt XOR*

Terlihat Pada Gambar 4 proses dekripsi pada angka “7” tersebut menggunakan *key* huruf “F”, proses ini dilakukan pada *binary* angka “7” dan *binary key* huruf “F”. setelah itu jadilah *binary key* yang telah di *decipher*, yaitu “1000001” yang di konversi ke Hexa menjadi “41” dan di konversi ke Desimal menjadi angka “65”, yang nanti nya di konversi ke *text ASCII* menjadi huruf “A”. Metode diharapkan dapat diimplementasikan di transmisi *data* antara ESP-32 dan *php server*.

BAB 3

Perancangan Sistem

3.1. Sistem yang Dibangun

Berikut adalah detail dari tabel perancangan sistem yang akan digunakan pada penelitian ini.

Tabel 2. Perancangan Sistem

Kebutuhan Perangkat	Nama Perangkat	Keterangan
Hardware	ESP-32	Sebuah hardware microcontroller yang mendukung MFRC 522 sebagai RFID <i>reader</i> . Microcontroller ini berperan sebagai otak dari RFID <i>reader</i> , maka dari itu semua isi codingan atau fungsi-fungsi akan di tampung di microcontroller ini.
	RFID Reader MFRC 522	RFID <i>reader</i> MFRC 522 adalah sebuah modul yang dapat melakukan Read kartu <i>tag</i> RFID dan juga melakukan Write ke kartu RFID tersebut. Modul ini berperan sebagai sensor untuk mendeteksi adanya RFID <i>Tag</i> dengan melakukan read dan write.
	RFID Tag Card mifare 1K	RFID <i>Tag</i> card mifare 1K ini adalah sebuah kartu yang dapat di write dan di read oleh RFID <i>reader</i> , RFID <i>tag</i> ini akan digunakan oleh <i>user</i> sah maupun <i>attacker</i> .
Software	Mysql	Mysql sebagai <i>database</i> yang akan menyimpan <i>data user</i> sah berupa, <i>email</i> , nama, <i>nim</i> , <i>password</i> , <i>synchronized secret</i> .
	Php Server	Php akan di gunakan di penelitian ini dan berperan sebagai <i>server</i> yang akan melakukan autentikasi <i>data user</i> sah ke dalam <i>database</i> .
	Vmware	Virtual machine ini di gunakan ketika <i>attacker</i> ingin melakukan serangan Man-in-the-middle, virtual machine ini menjadi pilihan untuk penelitian ini di karenakan proses (MitM) <i>attack</i> di butuhkan pihak ke-3 dalam proses komunikasi dua pihak.
	Parrot OS	Parrot OS adalah OS yang di rancang khusus untuk melakukan Tes keamanan. OS ini di gunakan dalam penelitian ini dikarenakan sudah di lengkapi <i>security tools</i> seperti ettercap, wireshark, dan mitmproxy.

	Ettercap/Wireshark	Ettercap dan wireshark adalah metode yang akan di uji sebagai teknik serangan (MitM) ketika RFID <i>reader</i> dan <i>Server</i> melakukan komunikasi. Teknik serangan ini akan digunakan <i>attacker</i> untuk meng- <i>capture traffic</i> atau <i>payload</i> yang di kirimkan antara dua pihak tersebut.
Brainware	<i>User</i> sah	<i>user</i> sah dapat melakukan tapping RFID dan memiliki akses penuh ke dalam sistem.
	<i>Attacker</i>	<i>Attacker</i> yang akan berupaya melakukan <i>cloning</i> dengan cara <i>Eavesdropping</i> untuk mencari informasi pada saat <i>server</i> mengirimkan <i>Synchronized secret</i> ke RFID <i>Reader</i>

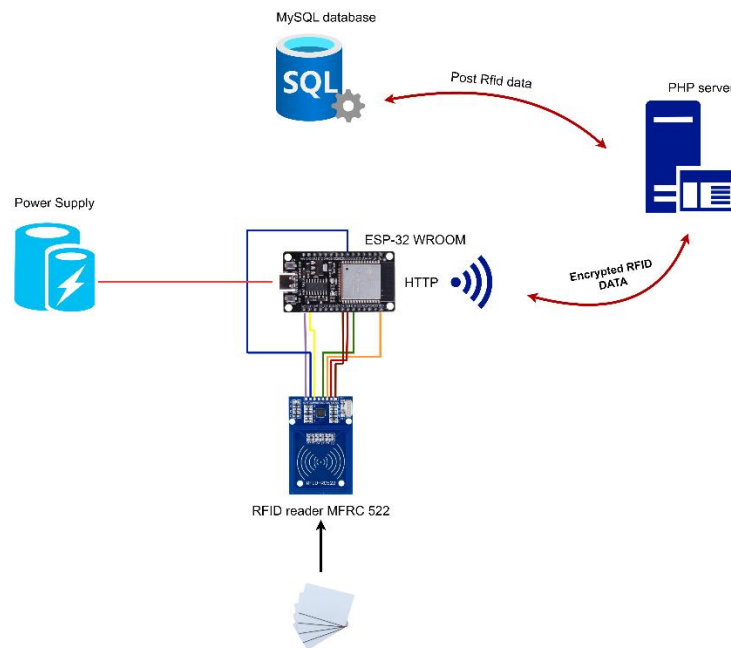
3.2. Requirement Sistem

Pada Tabel 3 Berikut adalah *Requirement System* untuk penelitian ini

Tabel 3.Requirement System

NO	Nama	Spesifikasi
1	Mikrokontroller	ESP-32 WROOM
2	RFID <i>reader</i>	RFID <i>Reader</i> MFRC 522
3	Blank RFID <i>tag</i> mifare	RFID <i>Tag</i> Card mifare 1K
4	<i>Database</i>	MySql
5	<i>Server</i>	Php <i>Server</i>

3.3. Skematik Sistem yang Dibangun

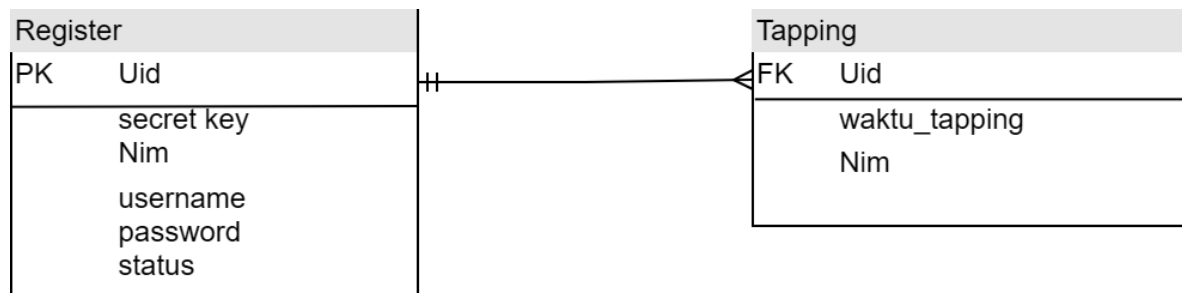


Gambar 5. Skematik Sistem

Tabel 2 dan gambar 5 telah menjelaskan sistem yang diperlukan pada penelitian ini, pada Gambar 6 di atas merupakan skematik sistem dari penelitian ini, pada gambar 5 terlihat RFID tag akan melakukan Tapping ke RFID reader MFRC 522 menggunakan ESP-32 sebagai Mikrokontroler, kemudian ESP-32 melakukan HTTP connect ke PHP server. Setelah mikrokontroler melakukan connect ke server, ESP-32 akan mengirim RFID data yang telah di enkripsi dengan metode *Simple XOR encryption*. Maka dari itu Php server akan melakukan autentikasi menggunakan MySql database untuk mencocokkan RFID data yang telah dikirimkan oleh ESP-32.

3.4. Struktur tabel database

Berikut adalah struktur tabel database SQL yang digunakan di tugas akhir ini.



Gambar 6. Struktur database

Pada gambar 6 diatas adalah struktur database yang digunakan di penelitian ini. Gambar 6 menunjukan tabel register dan tabel tapping mempunyai relasi attribut, yaitu *One to many*.

3.5. Desain Sistem

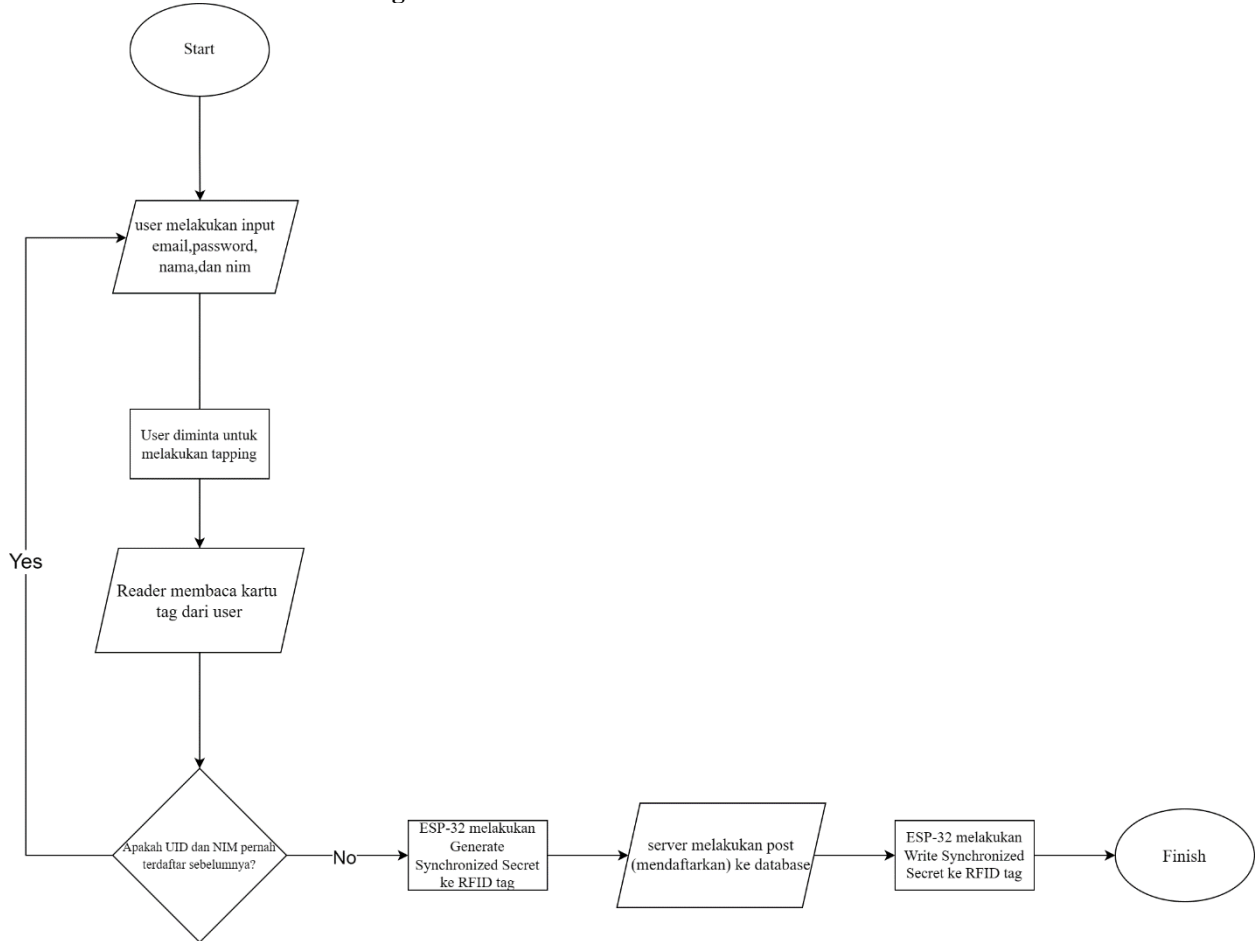
3.5.1. Desain Sistem Registrasi



Gambar 7. Desain Arsitektur Sistem Registrasi

Pada gambar 7 di atas menampilkan proses Sistem registrasi dari penelitian ini, pertama *user* akan diminta untuk meng-input *Email*, *NIM*, dan *password*. Setelah itu ESP-32 akan meminta *user* untuk melakukan tapping *RFID Tag* dan ESP-32 akan men-generate *synchronised secret*. Ketika *synchronised secret* tersebut telah di generate maka ESP-32 melakukan *Write* ke *RFID tag user*. Setelah *RFID tag user* sudah di *write*, ESP-32 akan melakukan *key exchange* dengan *php server* untuk menggunakan kunci yang sama sebelum melakukan enkripsi XOR. Selanjut nya ESP-32 mengirim *Encrypted Data* ke *server*, *server* akan melakukan *decrypt* menggunakan *key* yang telah di sepakati sebelumnya. Setelah didekripsi, *server* akan menyimpan *data* tersebut ke dalam *database Mysql*. ketika *database* tersebut telah berhasil menyimpan *data user*, maka ESP-32 akan men-trigger string “Registrasi telah berhasil” yang menandakan *database* telah merespond request ESP-32.

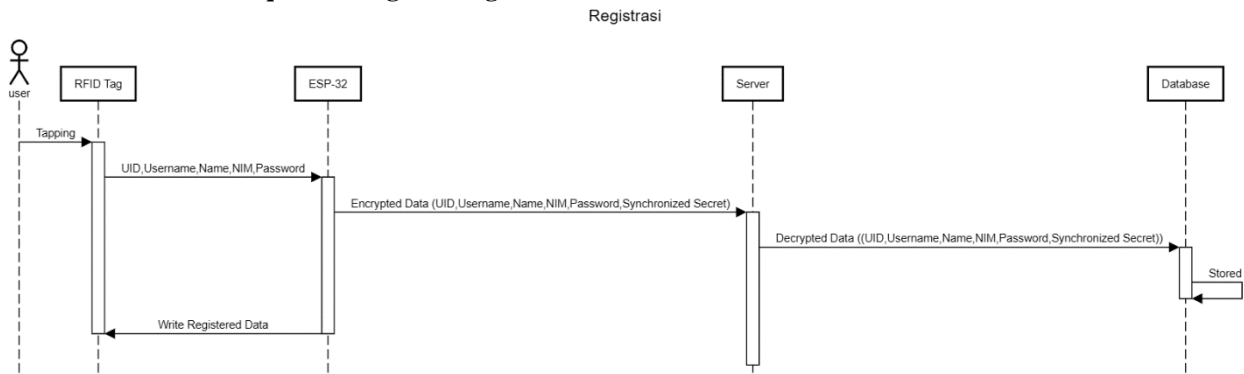
3.5.1.1. Flowchart Registrasi



Gambar 8. Flowchart Proses Registrasi

Flowchart pada gambar 8 menampilkan proses registrasi *user*, yang dimana *user* akan melakukan registrasi dan menginputkan *email*, *password*, nama, dan Nim. Setelah itu *user* akan diminta melakukan tapping untuk memberi akses ESP-32 untuk membaca dan melakukan *write* ke RFID tag. ketika *user* telah melakukan tapping, sistem akan melakukan pengecekan apakah UID dan NIM *user* sudah pernah terdaftar di dalam *database* sebelumnya. jika sudah pernah melakukan registrasi maka sistem akan meminta *user* melakukan *input email*, *password*, nama, dan nim. jika *user* belum pernah melakukan registrasi maka proses selanjutnya *server* akan melakukan post atau menyimpan *data user* ke dalam *database*. ketika *server* telah menyimpan *data user* atau berhasil melakukan registrasi, maka proses selanjutnya ESP-32 akan melakukan *write synchronized secret* dan *data-data user* lainnya ke dalam RFID tag. Setelah ESP-32 melakukan *Write* ke RFID tag *user*, maka *user* dapat melakukan proses tapping seperti yang terlihat di gambar 10.

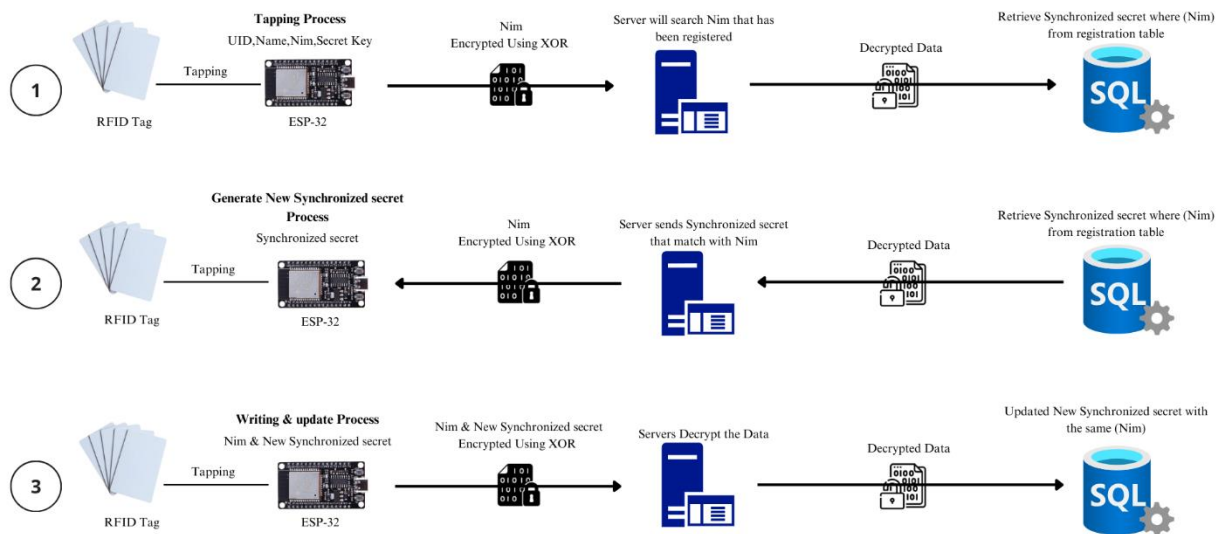
3.5.1.2. Sequence Diagram Registrasi



Gambar 9. Sequence Diagram Registrasi

Pada gambar 9. menjelaskan alur kerja proses registrasi. Pertama *user* akan melakukan Tapping ke RFID Tag, selanjutnya User diminta untuk melakukan *input username*, nama, *nim*, *password*. Ketika User Telah melakukan *input data*, ESP-32 akan mengirimkan *data* tersebut ke *server* dalam bentuk *encrypted data* dan melakukan *Write* ke RFID Tag *user*. *Server* akan melakukan *Decrypt data* dan menyimpan *data user* yang telah di registrasi.

3.5.2. Desain Sistem Tapping



Gambar 10. Desain Arsitektur Sistem Tapping

Gambar 10 merupakan desain arsitektur Sistem *tapping* yang dimana *synchronized secret* tersebut akan *terupdate* setiap kali *user* melakukan tapping. Pada gambar 10 di atas terlihat ada 3 proses yang akan terjadi ketika *user* ingin melakukan tapping, proses 1 (*tapping process*), proses 2 (*Generate new Synchronized secret process*), dan proses 3 (*Writing & update process*).

Proses pertama, *user* akan melakukan tapping RFID tag ke ESP-32, kemudian ESP-32 akan membaca isi *data* dari RFID tag *user* yang berisi UID, name, *nim*, *synchronized secret*. Ketika ESP-32 membaca *data* tersebut, ESP-32 akan mengirimkan *Nim user* ke *php server* untuk melakukan pengecekan atau query ke *database*. Sebelum dikirim *server* ESP-32 akan melakukan *key exchange* agar menyepakati kunci yang digunakan untuk melakukan *encrypt* dan *decrypt*. Setelah kedua pihak sepakat dengan kunci yang di gunakan, ESP-32 akan mengirim *Nim* tersebut dalam bentuk *encrypted data* menggunakan metode *XOR encryption*. Ketika *server* telah melakukan *decrypt data* yang di kirimkan oleh ESP-32, *server* akan melakukan *query* ke *database* untuk membandingkan *Nim* dan *synchronized secret* yang terdaftar sebelumnya.

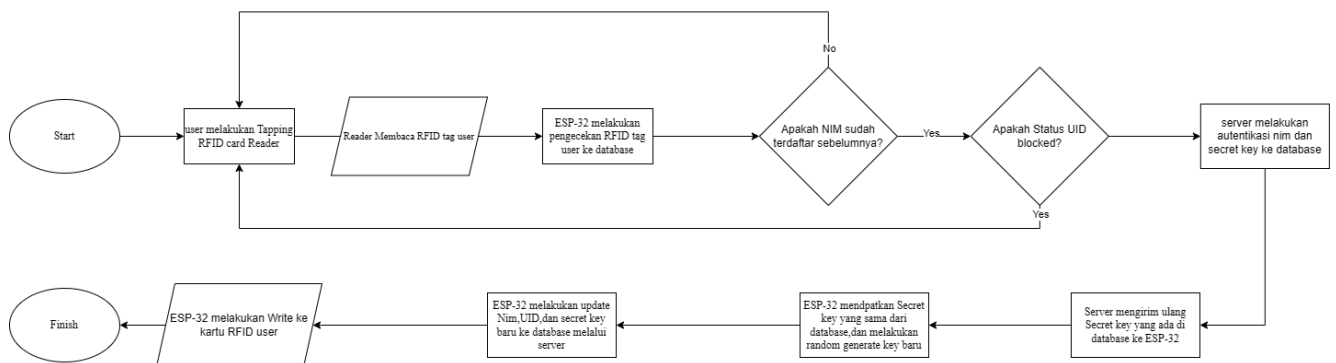
Proses kedua, ketika *server* telah mencocokkan *Nim* dan *synchronized secret* yang terdaftar di *database*, maka *server* akan mengirimkan *synchronized secret* yang telah terdaftar dari *database* dengan *nim* yang sama kembali ke ESP-32, proses ini untuk melakukan konfirmasi kalau *Nim* tersebut sudah terdaftar di *database* dengan

synchronized secret yang dikirimkan. Setelah *synchronized secret* sudah terbaca oleh ESP-32, ESP-32 akan membandingkan *synchronized secret* yang dikirimkan oleh *server* dan *synchronized secret* yang telah di *read* oleh MFRC 522, jika *synchronized secret* tersebut sama dengan yang terbaca oleh ESP-32, maka ESP-32 akan melakukan *generate new synchronized secret*.

Proses ketiga, ESP-32 akan melakukan *writing synchronized secret* baru ke RFID tag *user*. Jika *writing synchronized secret* baru telah berhasil, maka ESP-32 akan mengirimkan *nim* dan *Synchronized secret* yang telah di *generate* baru ke *server*. Ketika *server* telah membaca *new synchronized secret* dari ESP-32, *server* kemudian akan melakukan *update new synchronized secret* ke *database* dengan *nim* yang telah dikirim ke tabel registrasi.

Bisa disimpulkan dari proses-proses diatas ini bahwa terdapat 2 proses autentikasi pada sistem *tapping* ini. Yang pertama adalah proses dimana *server* melakukan pengecekan terhadap *Nim user* dan *status* dari kartu tersebut, apakah “*allowed*” atau “*blocked*”, jika *status* “*allowed*” maka *server* akan mengirimkan *synchronized secret* ke ESP-32, jika *status* “*blocked*” maka *server* tidak akan mengirim *synchronized secret* atau akses tidak diberikan. Selanjutnya proses kedua yang dimana ESP-32 akan melakukan autentikasi atau membandingkan *synchronized secret* yang terdaftar pada *database* dan yang di *tapping* oleh *user*, jika *synchronized secret* tersebut sama seperti yang diberikan oleh *server*, maka ESP-32 melakukan *generate new synchronized secret* dan *update* ke *database*, jika *synchronized secret* tidak sama seperti yang di *tapping* oleh *user* atau *synchronized secret* yang dikirimkan oleh *server*, maka ESP-32 tidak akan melakukan *generate new synchronized secret* dan tidak melakukan *update* ke *server*. Hal ini bisa disimpulkan bahwa proses validitas fungsi autentikasi ini lebih tepatnya ada di *server* dikarenakan *server* akan mengecek apakah *synchronized secret* terdaftar pada *database* dan apakah *status* dari uid tersebut “*allowed*”.

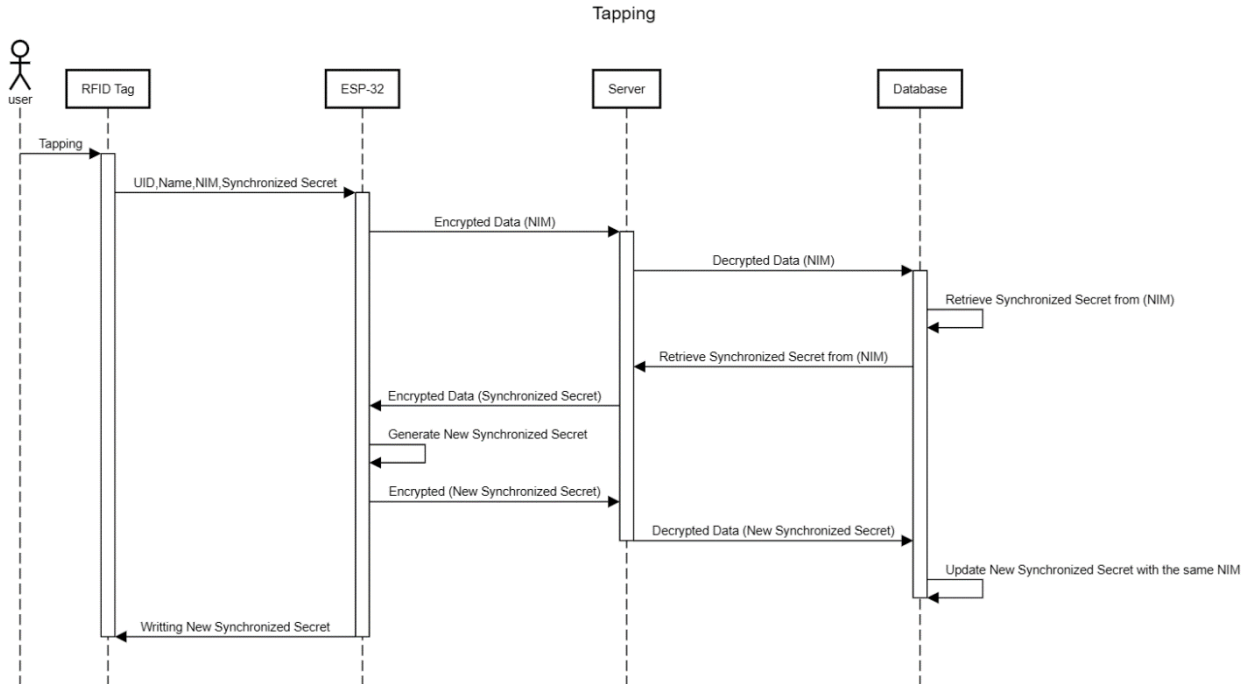
3.5.2.1. Flowchart Tapping



Gambar 11. Flowchart Proses Tapping

Setelah proses registrasi telah berhasil seperti yang terlihat di gambar 7, selanjutnya proses tapping akan dilakukan ketika *user* telah terdaftar di sistem. Pada gambar 10 menampilkan proses tapping yang akan dilakukan *user* menggunakan RFID tag. Ketika MFRC 522 mendeteksi RFID tag *user*, ESP-32 akan melakukan pengecekan data yang sudah terbaca oleh reader. Jika *Nim user* tidak terdaftar di dalam sistem, maka ESP-32 akan meminta *user* untuk melakukan tapping ulang. Akan tetapi jika *user* terdaftar di dalam sistem maka ESP-32 akan melakukan proses selanjutnya, yaitu proses pengecekan status UID, jika UID terblokir maka sistem tidak akan memberi akses *user* tersebut ke proses selanjutnya. Apabila status UID *user* (*allowed*), maka sistem akan melanjutkan proses autentikasi *Synchronized secret*. Proses selanjutnya *server* akan mengirim *Synchronized secret* yang di minta oleh ESP-32 menggunakan *Nim*, sebagai identitas *unique*. Ketika ESP-32 telah mendapatkan *Synchronized secret* dengan *nim* yang sama, ESP-32 akan melakukan *random generate synchronized secret* baru, untuk melakukan proses *Synchronized secret* pada RFID tag *user*. Jika proses *random generate synchronized secret* berhasil, selanjutnya ESP-32 akan melakukan *update data* ke RFID tag, untuk memperbaharui *synchronized secret* baru ke dalam *database* untuk menjadi identitas *unique* ke *tapping* selanjutnya. Saat proses *update* ke *database* berhasil, ESP-32 akan melakukan *write data* ke dalam RFID tag *user*.

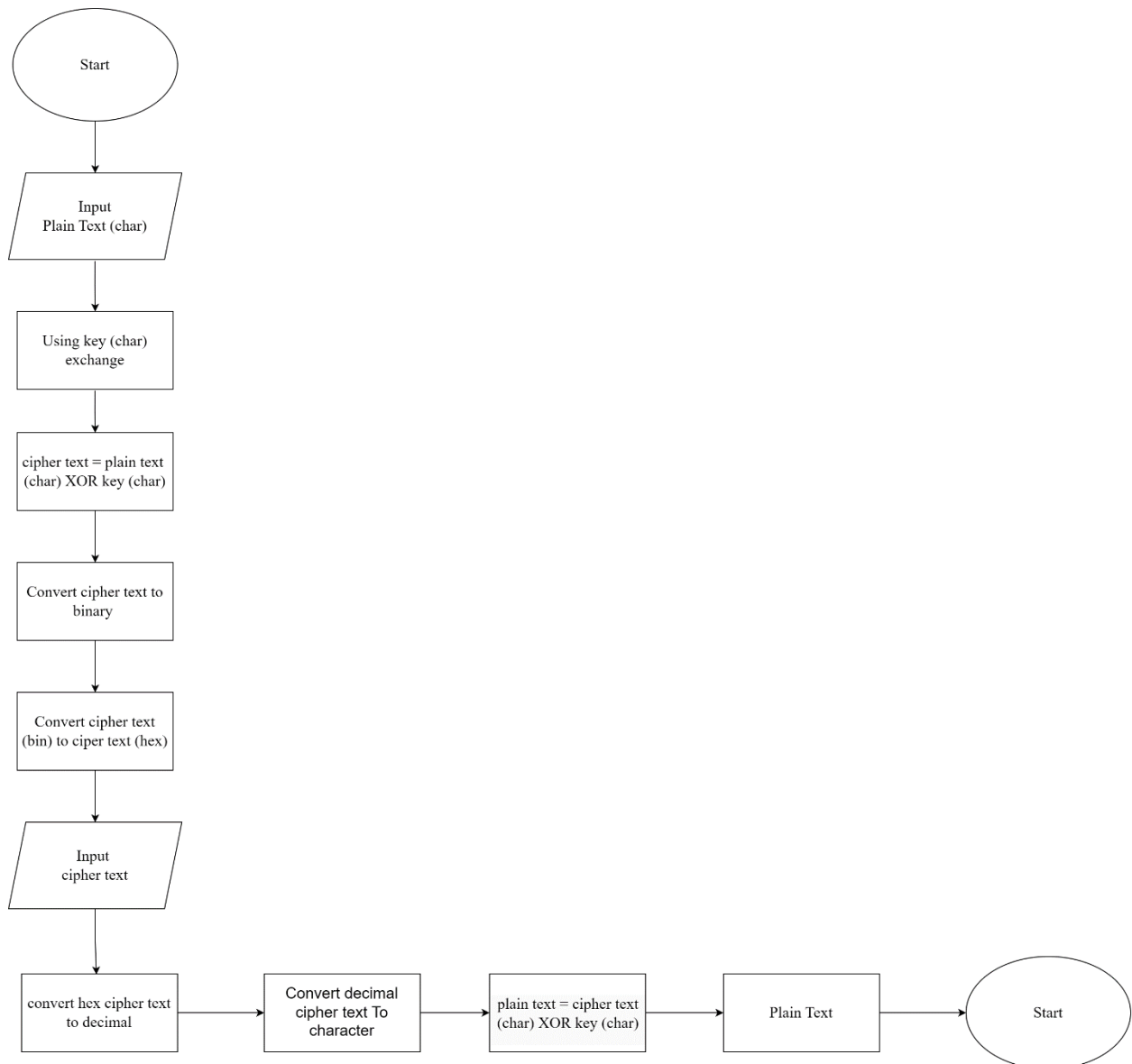
3.5.2.2. Sequence Diagram Tapping



Gambar 12. Sequence Diagram Tapping

Pada Gambar 12. Menjelaskan alur proses *tapping* dari sistem penelitian ini. Pertama *user* akan melakukan *Tapping* menggunakan *RFID Tag* ke *ESP-32*, *reader* akan membaca *UID*, nama, *nim*, dan *synchronized secret*. Setelah itu *ESP-32* akan mengirimkan *Data User* tersebut ke *Server* dengan bentuk *encrypted Data*, dan *server* akan mencari *Nim* yang dikirimkan dari *ESP-32* ke *database*. Saat *server* telah mendapatkan *Synchronized secret* dari *Nim* tersebut, maka *Server* akan mengirim kembali ke *ESP-32*, *ESP-32* akan melakukan membandingkan *synchronized secret* dari *RFID tag* yang telah di *read* dan *synchronized secret* yang di berikan dari *database*. Selanjutnya *ESP-32* akan melakukan *generate new synchronized secret*, di saat yang bersamaan *ESP-32* juga akan melakukan *write* ke *RFID tag user* dan *update new synchronized secret* ke *database*.

3.5.3. Flowchart Enkripsi XOR



Gambar 13. Flowchart proses Enkripsi XOR

Pada gambar 13 menampilkan proses enkripsi XOR sederhana yang ingin diterapkan di penelitian ini. Proses ini dimulai ketika sistem melakukan *inputan* berupa *Plain text* sederhana, setelah itu sistem akan mengambil *key* secara acak untuk melakukan *key exchange* dengan pihak lainnya agar dapat menyetujui *key* yang di gunakan. Saat dua pihak telah setuju menggunakan *key* yang sama, maka sistem akan melakukan proses enkripsi menggunakan Metode *Simple XOR encryption* dengan *key* yang telah disetujui sebelumnya, dengan melakukan XOR *plain text* dan *key* sehingga jadilah *cipher text*.

Ketika *plain text* telah di XOR, maka sistem akan melakukan *convert cipher text* ke *binary*, dan di *convert* lagi dari *binary* ke *hexadecimal*. Selanjutnya sistem akan mengirim ke pihak lainnya, dan melakukan *decrypt* menggunakan *key* yang telah disetujui, setelah itu sistem akan *convert cipher text* yang telah dikirimkan ke *decimal*, dan di *convert* dari *decimal* ke *character plain text*. *Character plain text* tersebut akan di XOR menggunakan *key* dan jadilah *plain text*.

3.5.4. Validasi enkripsi XOR pada sistem

```
encrypt.php
1  <?php
2  function encrypt(string $plainText, string $key) : string {
3      $output = "";
4      $keyPos = 0;
5
6      for ($p = 0; $p < strlen($plainText); $p++) {
7          if ($keyPos > strlen($key) - 1) {
8              $keyPos = 0;
9          }
10         $char = $plainText[$p] ^ $key[$keyPos]; //XOR Key
11         $bin = str_pad(decbin(ord($char)), 8, "0", STR_PAD_LEFT);
12
13         $hex = dechex(bindec($bin));
14         $hex = str_pad($hex, 2, "0", STR_PAD_LEFT);
15         $output .= strtoupper($hex);
16         $keyPos++;
17     }
18     return $output;
19 }
20
21 ?>
```

Gambar 14. Fungsi *code* enkripsi simple XOR pada php

Pada gambar diatas menunjukkan proses fungsi *code* enkripsi *simple XOR* pada php. Subbab ini akan menjelaskan proses enkripsi XOR pada sistem di penelitian ini. Sebelum masuk kedalam penjelasan proses enkripsi ini, di asumsikan *plain text* yang di gunakan adalah huruf "A" dan huruf "F" sebagai *key* nya. Subbab ini akan membandingkan dengan perhitungan *manual simple XOR encryption* pada subbab 2.6 dengan perhitungan *simple XOR encryption* pada sistem.

- Inisialisasi variabel
 - *Plain text* = "A"
 - *key* = "F"
 - *output* = "" (*string* kosong untuk menampung hasil enkripsi)
 - *keyPos* = 0 (posisi awal dari *key*)
- Looping Pertama **for (\$p = 0; \$p < strlen(\$plainText); \$p++) {**
 - Karakter dari *plain text* yang diproses: "A".
 - Karakter dari *key* yang diproses: "F".
 - Pada *loop* pertama, karakter *plain text* "A" (ASCII 65) dan *key* "F" (ASCII 70) akan diambil.
 - `$char = $plainText[$p] ^ $key[$keyPos];`
 - `$plainText[$p]` = "A" (ASCII 65, *biner* 01000001).
 - `$key[$keyPos]` = "F" (ASCII 70, *biner* 01000110).
 - Melakukan operasi XOR antara kedua *biner*:
01000001 (*biner* dari "A")
XOR
01000110 (*biner* dari "F")

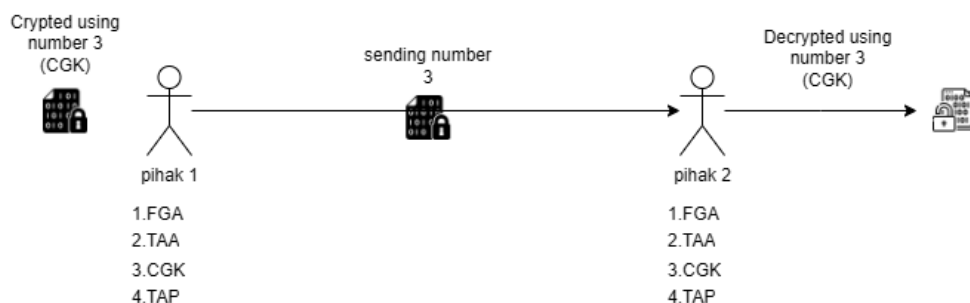
00000111 (*hasil XOR*)
 - Hasil XOR: 00000111 (*biner*) = 7 (*desimal*).
- Konversi XOR ke *Biner* `$bin = str_pad(decbin(ord($char)), 8, "0", STR_PAD_LEFT);`
 - `ord($char)`: Mengambil nilai ASCII dari hasil XOR, yaitu 7.
 - `decbin(7)`: Konversi nilai 7 ke *biner*: 111.
 - `str_pad()`: Tambahkan *padding* sehingga hasil menjadi 8-bit *biner*: 00000111.
- Konversi *Biner* ke Heksadesimal `$hex = dechex(bindec($bin));`
 - `bindec($bin)`: Konversi *biner* 00000111 menjadi *desimal* 7.
 - `dechex(7)`: Konversi *desimal* 7 menjadi heksadesimal 7.
- Menambahkan *padding* `$hex = str_pad($hex, 2, "0", STR_PAD_LEFT);`
 - `tr_pad($hex, 2, "0", STR_PAD_LEFT)`: Tambahkan *padding* agar hasil memiliki 2 digit heksadesimal, sehingga hasil menjadi 07.
- Gabungkan Hasil ke *Output* `$output .= strtoupper($hex);`
 - Hasilnya adalah 07 dalam huruf kapital (dengan `strtoupper()`).
 - `output` = "07".
- Update Posisi *Key* `$keyPos++;`
 - Posisi *key* diubah menjadi 1, tetapi karena *plain text* hanya memiliki satu karakter, *loop* selesai.

```
//Looping sebanyak jumlah karakter yang akan di encrypt
for ($p = 0; $p < strlen($plainText); $p++) {
    if ($keyPos > strlen($key) - 1) {
        $keyPos = 0;
    }
    $char = $plainText[$p] ^ $key[$keyPos]; //XOR Key
}
```

Gambar 15. Code proses *looping encrypt* sebanyak *length* karakter

Pada gambar 15, terlihat *code* dari sistem yang digunakan dari penelitian ini. Fungsi dari *code* ini, yaitu melakukan *looping* sebanyak dari karakter yang telah di *input* oleh *user* atau sistem. Cara kerja proses ini dimulai pada kondisi pengulangan dari “*for*”, isi dari kondisi pengulangan tersebut terdapat “*strlen*” yang artinya *string length*, proses ini akan melakukan *loop* sebanyak panjang dari *plain text* yang di *inputkan*. Setelah itu proses *looping key* yang dimulai pada kondisi pengulangan “*if*” akan melakukan *looping* sebanyak panjang dari *key* tersebut. Selanjutnya ketika proses diatas telah dilakukan, maka proses XOR pada parameter dari karakter (*char*) dilakukan.

3.6. Proses Key exchange



Gambar 16. Proses *Key exchange*

Pada gambar 16 terlihat sebuah proses *key exchange*, yang dimana proses ini terjadi sebelum *data* di enkripsi dikarenakan dua pihak harus menyetujui kunci yang akan di gunakan sebelum melakukan enkripsi dan dekripsi *data* tersebut. Proses ini bisa juga di sebut *PSK(pre-shared key)* yang dimana *key* tersebut telah disimpan oleh kedua pihak. Proses pertama yang dilakukan adalah pihak pertama akan melakukan *random pick number* dari *list*, yang akan dikirim ke pihak dua. Asumsikan pihak pertama menggunakan *random pick number* dari *list*, dan mendapatkan nomor 3. proses selanjutnya pihak pertama mengirimkan nomor 3 tersebut ke pihak ke dua, pihak kedua akan mendapat nomor tersebut dan memilih *PSK* yang akan digunakan dari nomor 3 di *list key* tersebut. Ketika pihak kedua mengetahui *key* dari nomor 3 tersebut, maka pihak kedua dapat melakukan dekripsi *data* yang telah dikirimkan oleh pihak pertama. Penelitian ini hanya menggunakan 3 karakter *key* seperti di gambar 14 sebagai example, akan tetapi metode enkripsi *simple XOR* ini keamanannya tergantung dengan panjang *key* yang digunakan, dikarenakan semakin acak dan panjang *key* tersebut semakin sulit ditebak oleh *attacker*. Adapun kasus lainnya seperti, apakah *PSK server* ke 100 ESP-32 itu sama? Jawabannya tentu saja sama. Namun, dari 100 ESP-32 menggunakan *PSK* yang sama akan meningkatkan kemungkinan yang besar proses transmisi *data* RFID menggunakan *key* yang sama. Solusi ini dari kasus ini yaitu, membuat banyak kemungkinan *key* yang akan digunakan untuk *key exchange*, semisal jika ada 100 ESP-32 maka *server* dan ESP-32 akan menggunakan 200 *PSK*, agar kemungkinan kesamaan *key* yang digunakan tidak besar.

3.7. Skenario Pengujian

Dalam penelitian ini, pengujian dilakukan untuk mengevaluasi efektivitas metode enkripsi XOR sederhana dalam mengamankan transmisi *data* RFID pada jaringan HTTP. Pengujian dilakukan dengan menggunakan perangkat ESP-32 sebagai pengirim *data*, dan *server* PHP sebagai penerima *data*. Vmware dengan sistem operasi ParrotOS digunakan sebagai pihak ketiga yang bertindak sebagai *attacker*, pihak ini mencoba melakukan serangan *Man-in-the-Middle* (MitM) untuk melakukan *eavesdropping data* yang dikirimkan. Tools seperti *Etercap* dan *Wireshark* digunakan oleh *attacker* untuk melakukan *ARP poisoning* dan *capture traffic* pada *payload* yang dikirimkan antara ESP-32 dan *server*.

3.7.1. Pengujian *Eavesdropping* Jaringan oleh *Attacker* Tanpa metode Enkripsi XOR sederhana

Pengujian *Eavesdropping* pada penelitian ini merupakan suatu proses dimana *attacker* akan melakukan *ARP poisoning* yang dimana *attacker* dapat menyadap komunikasi antara dua pihak. Pada proses ini *User* akan melakukan

Registrasi akun dan melakukan *write* ke RFID *tag*, Selanjutnya *attacker* akan menggunakan tools *ettercap* dan *wireshark* untuk melakukan *capture traffic* atau *payload* di transmisi *data* yang sedang di kirimkan oleh ESP-32 dan *Server*. Pada proses *Eavesdropping* ini sistem diasumsikan tidak menggunakan Metode enkripsi XOR sehingga *attacker* dapat melihat isi *traffic* atau *payload* yang dikirim oleh dua pihak tersebut. *Attacker* ini dapat melihat informasi *data* dari RFID *tag* yang telah di *tapping* ke ESP-32, maka dari itu *payload* seperti *username*, *UID*, *nim*, dan *synchronized secret* yang di *write* di dalam RFID *tag* tersebut dapat dilihat oleh *attacker*. Hal ini dapat memungkinkan *attacker* menggunakan informasi tersebut untuk melakukan serangan *cloning*, atau *sniffing*.

3.7.2. Pengujian Data RFID Menggunakan Metode Enkripsi XOR sederhana

Simple XOR encryption adalah metode yang diangkat dari penelitian ini untuk membuktikan dengan penerapan enkripsi pada transmisi *data* RFID di jaringan HTTP dapat diamankan menggunakan metode ini. Pengujian ini *User* akan melakukan *Tapping* ke RFID *reader* yaitu ESP-32, yang nantinya *Attacker* menggunakan *Vmware* dengan OS *parrotOS* sebagai pihak ketiga yang akan melakukan proses attack *Man-in-the-middle*. da juga tools-tools yang akan digunakan dalam pengujian ini seperti *ettercap* dan *Wireshark*. Pada saat ESP-32 melakukan transmisi *data* ke *php server*, *Vmware* ini akan melakukan *Eavesdropping* ke transmisi *data* ESP-32 dan *Php server* dan menggunakan *Ettercap* atau *Wireshark* untuk meng-*capture traffic* dan *payload* yang sedang dikirim oleh kedua pihak tersebut. Ketika metode enkripsi ini dilakukan maka *traffic* atau *payload* yang dikirimkan akan berbentuk *crypted* dan tidak bisa terbaca oleh *attacker*.

BAB 4

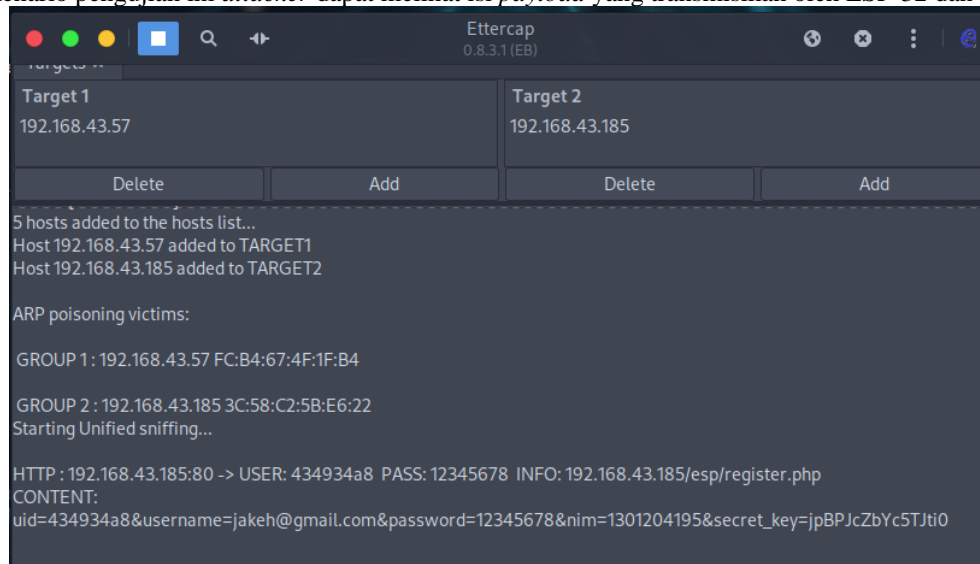
Evaluasi

Pada bab ini akan dilakukan skenario pengujian terhadap jaringan antara ESP-32 dan *Server*. pengujian ini berdasarkan skenario yang telah di jelaskan di bab sebelumnya. Pengujian ini terdiri dari 2 skenario, skenario pertama ESP-32 dan *php server* menggunakan metode enkripsi *Simple XOR* untuk berkomunikasi di jaringan tersebut, sehingga ketika *data* yang di transmisi kan terlindungi oleh *Simple XOR encryption* dan *attacker* tidak dapat melihat *payload* yang dikirim antara dua pihak tersebut. Skenario kedua, pengujian *Eavesdropping* pada jaringan sebagai Man-in-the-middle, yang dimana serangan ini dilakukan menggunakan tools untuk meng-*capture traffic* dan melihat isi *payload* yang dikirimkan *server* dan ESP-32.

4.1. Hasil Pengujian Dan Analisis

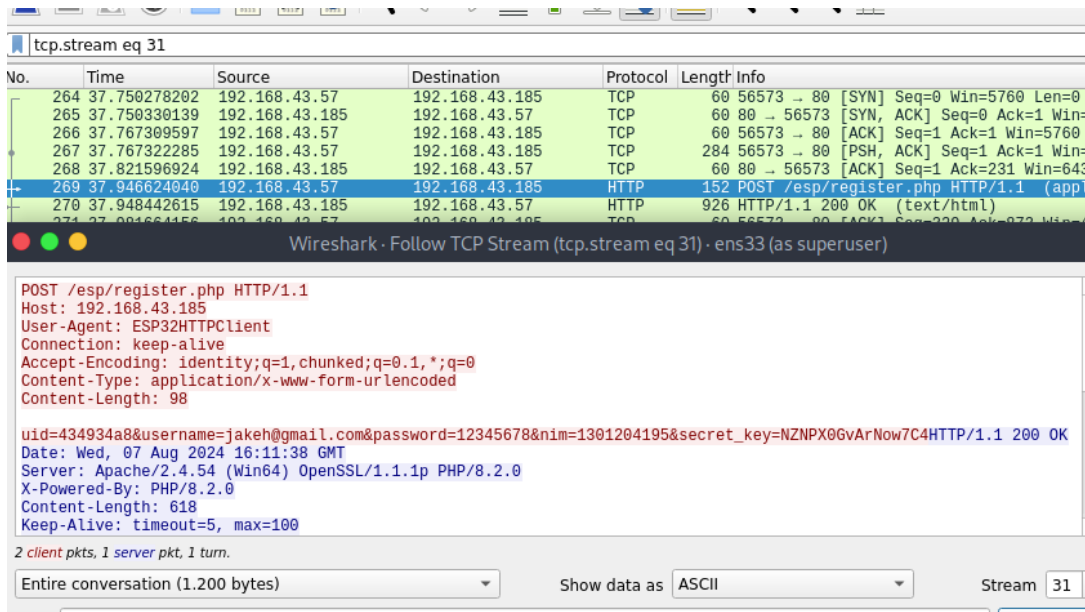
4.1.1. Hasil pengujian *Eavesdropping* Jaringan oleh *Attacker* Tanpa Metode Enkripsi XOR sederhana

Pengujian ini dilakukan dengan Skenario dimana *attacker* menggunakan Vmware untuk menjadi MitM. Proses serangan ini akan melibatkan tools seperti ettercap dan wireshark. *User* akan melakukan registrasi melalui ESP-32, akan tetapi dalam konteks skenario pengujian ini, sistem tidak menggunakan metode enkripsi *Simple XOR*, sehingga skenario pengujian ini *attacker* dapat melihat isi *payload* yang transmisikan oleh ESP-32 dan *server*.



Gambar 17. Proses *Eavesdropping* data tanpa enkripsi

Dilihat pada Gambar 17, proses di atas diasumsikan *user* telah melakukan registrasi di ESP-32, dan *attacker* menggunakan Vmware untuk menjadi MitM. *Attacker* menggunakan Ettercap untuk melakukan *ARP poisoning* ke transmisi ESP-32 dan *server* sebagai MitM untuk melihat *payload* yang telah ditransmisikan. Terlihat dari gambar 14, beberapa *data* yang ter-*capture* oleh Ettercap seperti, *UID*, *username*, *password*, *nim*, dan *synchronized secret*. Hal ini dapat disimpulkan *attacker* telah mendapatkan *data* sensitif berasal dari *RFID tag user*. Selanjutnya proses *attacker* menggunakan Wireshark untuk meng-*capture traffic* jaringan di gambar 18.

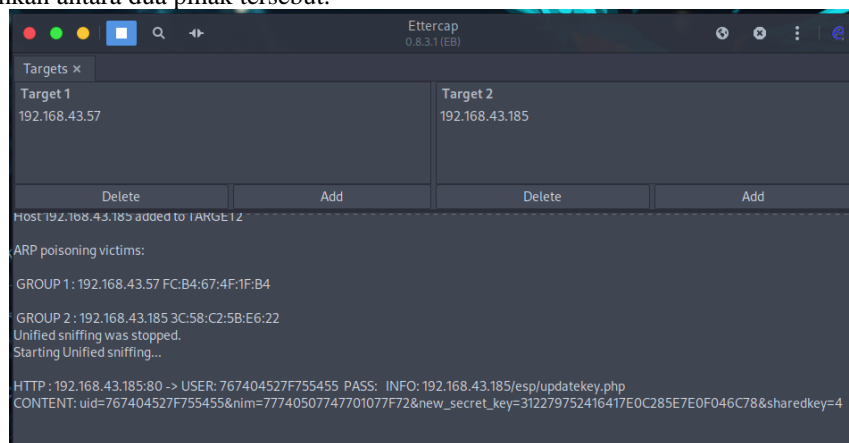


Gambar 18. Proses *Capture traffic* Wireshark tanpa enkripsi

Pada gambar 18, terlihat *Attacker* melakukan *Capture traffic* menggunakan tools Wireshark. Proses ini dimulai ketika *attacker* melakukan “Start capturing packet”. Setelah itu *attacker* melakukan filter IP source yang menggunakan protokol HTTP. ketika *attacker* mendapatkan IP tersebut, selanjutnya *Attacker* akan melakukan “Follow TCP stream”. Terlihat dari gambar 16, beberapa data dapat di lihat seperti UID, username, password, nim, dan *synchronized secret* dari RFID tag user.

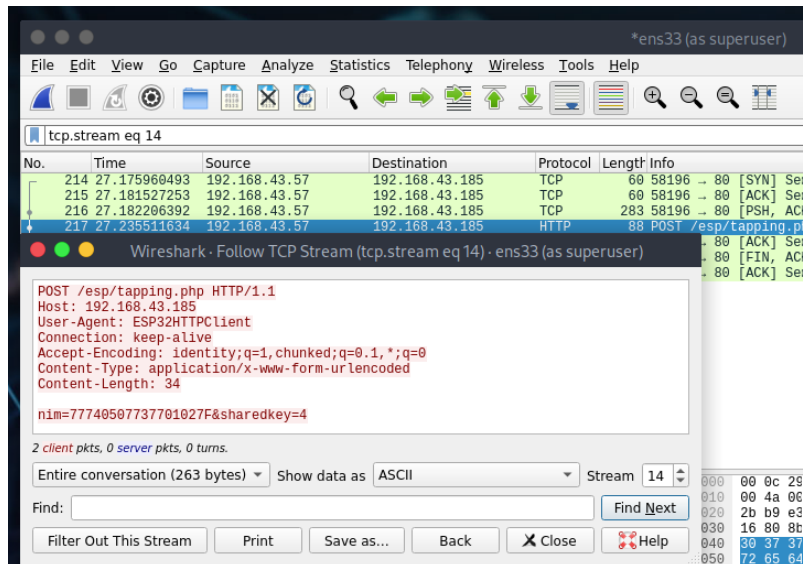
4.1.2. Hasil Pengujian Data RFID Menggunakan Metode Enkripsi XOR sederhana

Pengujian Skenario metode ini dapat dibuktikan melalui *Eavesdropping* yang akan di lakukan oleh *attacker*. proses pertama yaitu user akan melakukan tapping ke RFID reader, dan ESP-32 akan melakukan proses pengiriman transmisi data ke php server. *Attacker* akan melakukan ARP poisoning antara ESP-32 dan server, jika payload atau traffic tersebut telah dienkripsi menggunakan Simple Xor encryption, maka *attacker* tidak bisa melihat payload yang dikirimkan antara dua pihak tersebut.



Gambar 19. Proses *Eavesdropping* MitM oleh *Attacker*

Pada gambar 19 di atas sebuah *attacker* dapat melakukan *Eavesdropping* dan melihat data yang di transmisi kan oleh ESP-32 dan server. Skenario Serangan ini, *attacker* menggunakan VmWare untuk menjadi MitM dan menggunakan Ettercap untuk melakukan ARP poisoning, sehingga transmisi yang dikirim oleh dua pihak tersebut akan melalui MitM dan *attacker* dapat melihat isi payload tersebut. *Attacker* akan melakukan MitM attack dan mencari atau scan host yang ada di jaringan tersebut, ketika *attacker* mendapatkan IP nya, *attacker* akan melakukan set target untuk proses ARP poisoning. Akan tetapi proses transmisi data tersebut telah di enkripsi dengan Simple XOR, sehingga *attacker* tidak dapat mengetahui data yang ditransmisikan tersebut. Selanjutnya proses *attacker* ketika menggunakan Wireshark untuk melakukan *capture traffic* yang ada di jaringan.



Gambar 20. Proses *Capture traffic* Wireshark oleh *Attacker*

Gambar 20 menampilkan proses ketika *attacker* ingin melakukan *capture traffic* menggunakan wireshark. Proses ini memerlukan sebuah *filter* Target IP yang ingin di *capture*. Pada gambar 17, terlihat source IP menggunakan protokol HTTP yang di *capture* oleh *attacker*, namun ketika *attacker* ini melakukan “*Follow TCP Stream*”, *payload* tersebut telah menjadi *data crypted*, sehingga *attacker* tidak dapat memahami *data* yang di kirimkan oleh ESP-32 dan *server*. Dari hasil pengujian di atas telah terbukti, sistem dapat merahasiakan transmisi *data*.

4.2. Hasil analisis pengujian

Dari skenario pengujian sebelumnya, telah disimpulkan bahwa *simple XOR encryption* dapat diimplementasikan menggunakan mikrokontroler ESP-32. Hasil dari skenario pengujian tersebut telah membuktikan bahwa *payload* yang ditransmisikan dapat di *encrypt* dan tidak dapat dibaca langsung. Terlihat pada gambar 14 dan 15, adalah proses dimana serangan MitM melakukan *eavesdropping* pada transmisi data ESP-32 dan *server*. Pada gambar 14, menjelaskan bagaimana proses *tools* ettercap melakukan MitM dengan melakukan ARP *poisoning* dengan kedua pihak tersebut, pihak pertama yaitu ESP-32 dengan IP 192.168.43.57 dan IP *server* 192.168.43.185. Ketika *target* IP telah di pilih oleh *attacker*, maka ettercap akan melakukan *sniffing*. Ketika user melakukan *tapping* atau registrasi, maka ettercap akan meng-*capture* *payload* tersebut, begitu pula dengan proses *capture traffic* pada wireshark.

Pada gambar 15 menunjukkan proses *capture traffic* dengan melakukan “*follow TCP stream*” pada source IP ESP-32 yaitu 192.168.43.57 ke destinasi IP *server* 192.168.43.185 dengan protokol HTTP. Ketika melakukan “*follow TCP stream*”, terlihat pada *tab* tersebut isi dari *payload* yang di transmisikan antara dua pihak. Berikut tabel analisis dari pengujian transmisi data tanpa menggunakan enkripsi *simple XOR*.

Tabel 4. Hasil Analisis Pengujian transmisi *data* tanpa enkripsi XOR

No	Payload	Data Content
1	UID	434934a8
2	Email	Jakehh@gmail.com
3	Password	12345678
4	Nim	1301204195
5	Synchronized secret	NZNPXX0GvArNow7C4

Hasil pengujian ini memperlihatkan bahwa *data* yang di transmisi kan dari ESP-32 ke *server*, dapat terlihat oleh *Attacker* yang menggunakan ettercap dan wireshark. Pada tabel 4. Terdapat 5 jenis *payload* yang dikirimkan oleh ESP-32 ke *server*, yaitu UID, email, password, nim, dan *synchronized secret*. Hal ini dapat memungkinkan *attacker* dapat menggunakan *data* atau informasi sensitif tersebut dapat digunakan untuk *cloning* RFID card.

Selanjutnya, skenario dimana *attacker* melakukan *eavesdropping* dan sistem tidak menggunakan metode enkripsi. Ketika *attacker* melakukan *eavesdropping* menggunakan *tools* ettercap dan wireshark, *attacker* tetap bisa melihat isi *payload* yang ditransmisikan oleh ESP-32 dan *server*, namun *attacker* tidak dapat memahami apa isi dari *payload* tersebut dikarenakan konten dari *data* nya telah berubah menjadi *cipher text*. Skenario ini membuktikan bahwa sistem tersebut dapat mengimplementasikan metode *simple XOR encryption* di ESP-32 dan *server*. Berikut adalah tabel analisis dari pengujian transmisi data menggunakan enkripsi *simple XOR*.

Tabel 5. Hasil Analisis Pengujian Transmisi *Data* menggunakan Enkripsi XOR

No	<i>Payload</i>	<i>Data Content</i>	<i>Encrypted</i>
1	UID	434934a8	767404527F755455
2	Nim	1301204195	77740507747701077F72
3	<i>Synchronized secret</i>	NZNPXX0GvArNow7C4	312279752416417E0C285E7E0F046C78
4	<i>Shared key</i>	4	-

Pada tabel 5, terlihat *payload* yang dikirimkan oleh ESP-32 ke *server* telah di enkripsi menggunakan metode *Simple XOR*, sehingga isi *data* tersebut menjadi *cipher text*.

BAB 5

Kesimpulan dan Saran

5.1. Kesimpulan

Penelitian ini berhasil mengimplementasikan metode enkripsi XOR sederhana pada mikrokontroler ESP-32 untuk meningkatkan keamanan transmisi *data* RFID dan mencegah serangan *eavesdropping*. Berdasarkan pengujian yang dilakukan, metode enkripsi XOR terbukti dapat diimplementasikan dan mampu menyembunyikan *payload data* dari serangan Man-in-the-Middle (MitM), dimana *attacker* tidak dapat membaca isi *payload* yang telah dienkripsi selama proses transmisi antara ESP-32 dan *server* PHP. Penelitian ini juga menunjukkan bahwa metode enkripsi XOR dapat melindungi *data* sensitif dalam kartu RFID dari serangan *eavesdropping*, yang merupakan ancaman umum dalam aplikasi RFID.

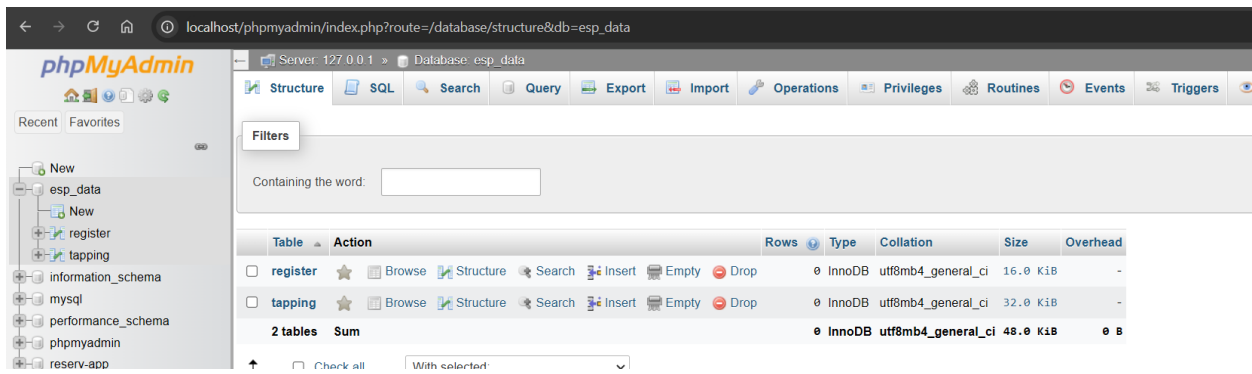
5.2. Saran

Penelitian ini memberikan solusi hemat biaya dan efisien untuk meningkatkan keamanan sistem RFID dengan menggunakan mikrokontroler ESP-32. Mikrokontroler ini menawarkan keunggulan dalam hal ukuran yang kecil, biaya yang terjangkau, dan fitur konektivitas nirkabel yang kuat, sehingga sangat cocok untuk aplikasi IoT yang memerlukan komunikasi *data* yang andal dan *real-time*. Secara keseluruhan, hasil penelitian ini dapat dijadikan dasar untuk pengembangan lebih lanjut dalam bidang keamanan transmisi *data* RFID dan dapat diterapkan dalam berbagai aplikasi yang memerlukan perlindungan *data* yang kuat dari serangan *eavesdropping*. Untuk pekerjaan kedepannya, metode dari penelitian ini menggunakan *key distributon* yang dimana memerlukan distribusi *key* yang komulatif dan *ter-update*, sehingga *operator* atau *admin* tidak perlu melakukan *input key* secara *manual* di dalam sistem. Metode *Simple XOR* ini juga bisa dikatakan kompatibel dan dapat melindungi *payload data* yang dikirimkan, namun, keamanannya sangat bergantung pada kerahasiaan, keacakan kunci yang digunakan dan panjang kunci yang digunakan.

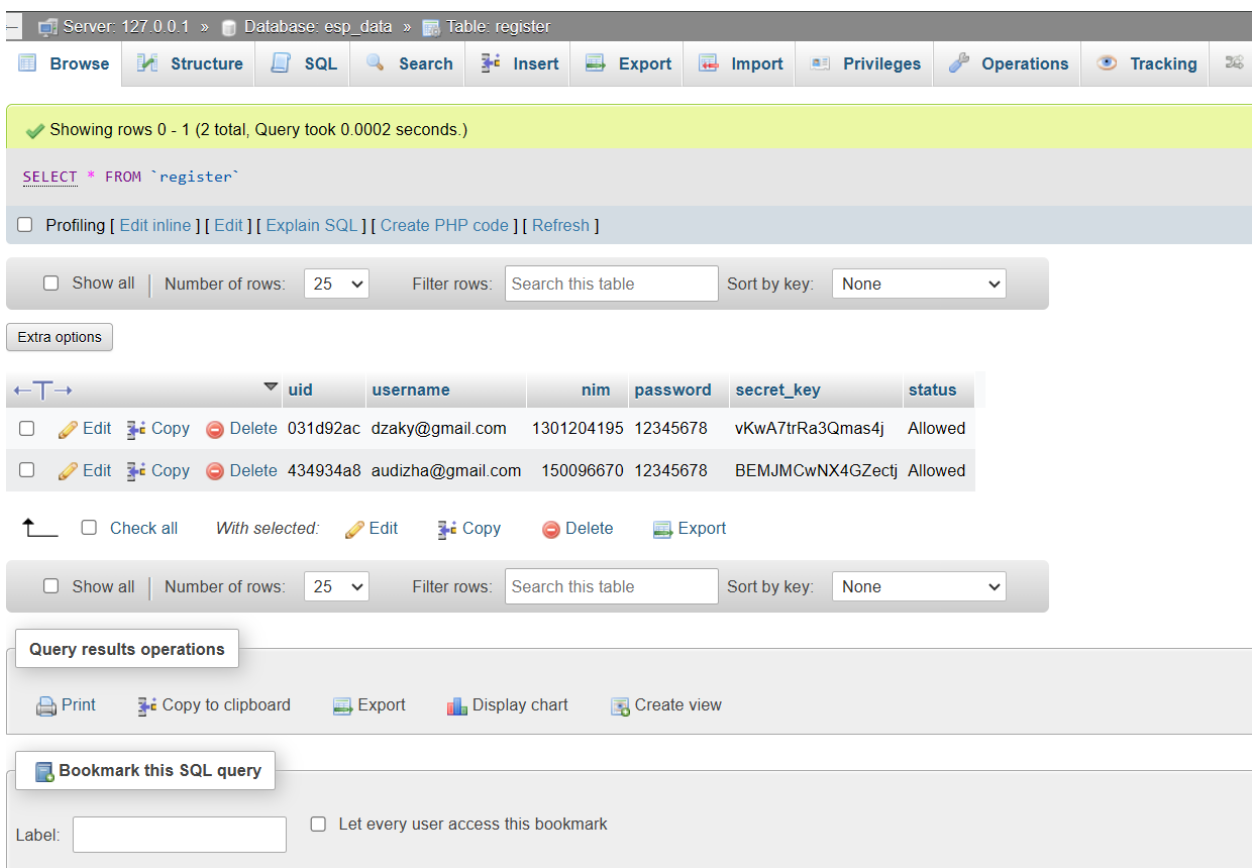
Daftar Pustaka

- [1] J. Tugas, A. Fakultas Informatika, D. D. Arisandi, F. A. Yulianto, and A. Rakhmatsyah, “Deteksi Serangan *Cloning* pada RFID Mifare Menggunakan Metode *Synchronized secret*,” 2021.
- [2] G. Chavira, S. W. Nava, R. Hervás, J. Bravo, and C. Sánchez, “Combining RFID and NFC technologies in an AmI conference scenario,” in *Proceedings of the Mexican International Conference on Computer Science*, 2007, pp. 165–172. doi: 10.1109/ENC.2007.30.
- [3] A. Pratama, “Eksplorasi RFID Menggunakan NFC dengan Teknik Cloning pada Studi Kasus KTM.”
- [4] M. S. Widura, Y. Purwanto, and S. M. Nasution, “Enkripsi Data pada Kartu RFID Menggunakan Algoritma AES-128 untuk Angkutan Umum di Kabupaten Bandung.”
- [5] A. Hermawan, E. Iman, and H. Ujjianto, “Implementasi Enkripsi *Data* Menggunakan Kombinasi AES dan RSA,” vol. 5, no. 2, 2021, doi: 10.30743/infotekjar.v5i2.3585.
- [6] W. : Www, J. Thakur, and N. Kumar, “International Journal of Emerging Technology and Advanced Engineering DES, AES and Blowfish: Symmetric Key Cryptography Algorithms Simulation Based Performance Analysis,” 2011. [Online]. Available: www.tropsoft.com
- [7] S. Akter, S. Chellappan, T. Chakraborty, T. A. Khan, A. Rahman, and A. B. M. Alim Al Islam, “Man-in-the-Middle Attack on Contactless Payment over NFC Communications: Design, Implementation, Experiments and Detection,” *IEEE Trans Dependable Secure Comput*, vol. 18, no. 6, pp. 3012–3023, 2021, doi: 10.1109/TDSC.2020.3030213.
- [8] Oakland University, IEEE Region 4, and Institute of Electrical and Electronics Engineers, *Man-in-the-Middle (MITM) Attack Based Hijacking of HTTP traffic using open source tools*.
- [9] B. Q. Zhao, H. M. Wang, and P. Liu, “Safeguarding RFID Wireless Communication against Proactive *Eavesdropping*,” *IEEE Internet Things J*, vol. 7, no. 12, pp. 11587–11600, Dec. 2020, doi: 10.1109/IIOT.2020.2998789.
- [10] S. Chakravarty, G. Portokalidis, M. Polychronakis, and A. D. Keromytis, “Detection and analysis of *eavesdropping* in anonymous communication networks,” *Int J Inf Secur*, vol. 14, no. 3, pp. 205–220, Jun. 2015, doi: 10.1007/s10207-014-0256-7.
- [11] C. H. Hsu, S. Wang, D. Zhang, H. C. Chu, and N. Lu, “Efficient identity authentication and *encryption* technique for high throughput RFID system,” *Security and Communication Networks*, vol. 9, no. 15, pp. 2581–2591, Oct. 2016, doi: 10.1002/sec.1488.
- [12] A. Ilic, M. Lehtonen, F. Michahelles, and E. Fleisch, “*Synchronized secrets* Approach for RFID-enabled Anti-Counterfeiting.”
- [13] D. Rachmawati, M. Andri Budiman, and I. Aulia, “Super-Encryption Implementation Using Monoalphabetic Algorithm and XOR Algorithm for *Data Security*,” in *Journal of Physics: Conference Series*, Institute of Physics Publishing, Mar. 2018. doi: 10.1088/1742-6596/979/1/012033.
- [14] L. Chong Han and N. Muzlifah Mahyuddin, “An Implementation of Caesar Cipher and XOR *Encryption* Technique in a Secure Wireless Communication.”

Lampiran



Gambar 21. Tampilan *database esp_data*



Gambar 22. Tampilan tabel *register* ketika user melakukan registrasi

✓ Showing rows 0 - 3 (4 total, Query took 0.0002 seconds.)

```
SELECT * FROM `tapping`
```

☐ Profiling [[Edit inline](#)] [[Edit](#)] [[Explain SQL](#)] [[Create PHP code](#)] [[Refresh](#)]

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

Extra options

waktu_tapping	uid	nim
2024-09-07 10:40:38	031d92ac	1301204195
2024-09-07 10:40:44	434934a8	150096670
2024-09-07 10:41:12	031d92ac	1301204195
2024-09-07 10:41:18	434934a8	150096670

☐ Show all | Number of rows: 25 | Filter rows: Search this table | Sort by key: None

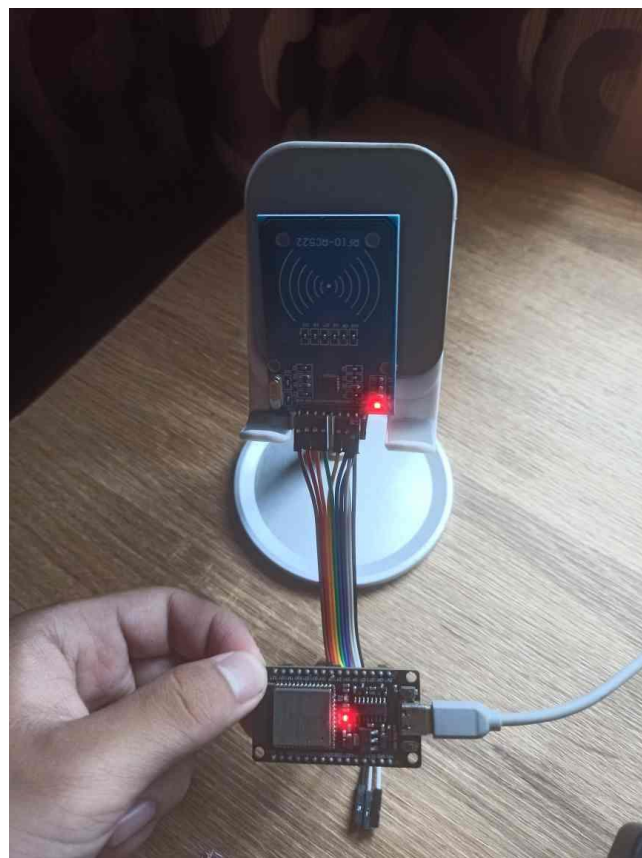
Query results operations

[Print](#) [Copy to clipboard](#) [Export](#) [Display chart](#) [Create view](#)

[Bookmark this SQL query](#)

Label: ☐ Let every user access this bookmark

Gambar 23. Tampilan tabel *tapping* ketika user melakukan *tapping*



Gambar 24. Perangkat Hardware yang digunakan

Video Demo aplikasi:

https://drive.google.com/drive/folders/14vtjOU8qdEu7jL_2RIT6x6ecjBeZxNA0?usp=sharing