Implementasi Teknologi Blockchain untuk Meningkatkan Keamanan Email

Samuel Eric Yonatan (18221133)
Program Studi Sistem dan Teknologi Informasi
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
18221133@std.stei.itb.ac.id

***Abstract*—Email adalah salah satu media komunikasi yang paling banyak digunakan pada era digital saat ini. Keamanan *email* menjadi salah satu aspek yang sangat krusial mengingat tingginya tingkat ancaman siber seperti *phishing*, *spoofing*, dan serangan *man-in-the-middle*. Meskipun berbagai metode dan protokol telah digunakan untuk meningkatkan keamanan email, *phishing attack* dan ancaman siber lain masih banyak terjadi. Teknologi *blockchain* menawarkan solusi inovatif dengan menyediakan platform terdistribusi yang aman dan transparan untuk komunikasi *email*. Makalah ini membahas penerapan teknologi *blockchain* dalam meningkatkan keamanan *email* dan meminimalkan *phishing attack* dan *spam email*. Pendekatan yang diusulkan mencakup mekanisme enkripsi *end-to-end* yang didukung oleh *blockchain* untuk verifikasi identitas dan otentikasi pesan. Dengan memanfaatkan sifat terdistribusi dari *blockchain*, sistem yang diusulkan tidak hanya meningkatkan keamanan tetapi juga mengurangi ancaman *phishing attack* dan risiko manipulasi data dari pihak ketiga.**

Keywords—blockchain; email; RSA; AES-256; digital signature

I. Pendahuluan

Dewasa ini, teknologi semakin berkembang cepat dan menjadi bagian yang tidak terpisahkan dari kehidupan sehari-hari. Teknologi telah diterapkan dalam berbagai sektor seperti pendidikan, kesehatan, perdagangan, pemerintahan dan mempermudah aktivitas yang dilakukan. Salah satu keuntungan utama dari teknologi adalah mudahnya berkomunikasi dengan orang lain melalui berbagai media seperti *text* (*chat*), telepon, dan hingga *video call*. Dengan teknologi, batasan waktu dan geografis menghilang dan komunikasi dapat dilakukan oleh siapapun dengan mudah.

Salah satu media komunikasi yang muncul dari perkembangan teknologi adalah *electronic mail* atau yang sering dikenal sebagai *email*. Jumlah *email* yang dikirimkan pada 2022 mencapai 333,2 miliar dan diprediksikan akan terus meningkat hingga 392,5 miliar pada 2026 [1]. Penggunaan *email* yang semakin meningkat mengakibatkan *bad actor* melakukan aktivitas yang mengganggu keamanan *email*, salah satunya adalah *phishing attack* dan *spam email*. *Spam email* adalah *email* yang tidak diminta dan tidak diinginkan [2]. Pada tahun 2023, sekitar 45.6% dari seluruh *email* yang dikirimkan merupakan *spam email* [4]. *Spam email* menjadi salah satu metode favorit untuk *hacker* dan *spammer* karena penggunaan *email* dalam jumlah besar di seluruh organisasi dan industri dan untuk keperluan pribadi [2]. *Phishing* adalah praktik mengirim komunikasi penipuan yang tampak berasal dari sumber yang sah dan terkemuka [5]. Sekitar 96% dari *phishing attack* dilakukan melalui *email* [6]. Pada tahun 2023, sekitar 27.32% dari seluruh *email* yang dikirimkan merupakan *phishing attack* yang ditujukan ke institusi keuangan [3].

Phishing attack mengakibatkan kerugian yang besar bagi perusahaan. Berdasarkan penelitian yang dilakukan IBM, *phishing attack* menempati peringkat kedua sebagai penyebab kebocoran data yang paling mahal—pelanggaran yang disebabkan oleh *phishing attack* menghabiskan biaya rata-rata $4,65 juta bagi bisnis [6]. Dampak dari *spam email* dan *phishing attack* yang berhasil pada akhir tahun 2020 adalah 60% dari organisasi kehilangan data, 52% dari organisasi mengalami pembobolan akun atau kredensial, 47% dari organisasi terinfeksi oleh serangan *ransomware*, 29% dari organisasi terinfeksi oleh serangan *malware*, dan 18% dari organisasi mengalami kerugian finansial [6].

Untuk mengatasi masalah tersebut, diperlukan mekanisme perlindungan yang mampu meningkatkan otentikasi pengirim *email* agar jumlah *phishing attack* yang terjadi melalui *email* dapat dikurangi. Mekanisme tersebut juga harus mampu menjaga integritas dan kerahasiaan *email*. Salah satu mekanisme tersebut adalah penggunaan teknologi *blockchain*. Teknologi *blockchain* menyediakan mekanisme yang mampu memastikan integritas data dengan membuat data tidak dapat dimanipulasi setelah ditambahkan ke *blockchain*. Teknologi *blockchain* akan dibantu dengan algoritma RSA sebagai salah satu algoritma kriptografi kunci publik yang kuat dan sulit untuk dipecahkan. Algoritma RSA digunakan agar hanya penerima yang dituju dapat mengakses *email* dengan kunci privat penerima.

Dengan menerapkan teknologi *blockchain* dengan algoritma RSA, diharapkan keamanan *email* dapat ditingkatkan dan serangan siber pada *email* dapat dikurangi. Dengan *blockchain*, *email* dapat disimpan dan tidak dapat diubah sehingga menjaga integritas *email*. Algoritma RSA memastikan hanya penerima *email* yang dapat mengakses dan membaca *email* tersebut. Setiap *email* juga ditandatangani secara digital menggunakan kunci privat pengirim untuk meningkatkan keamanan dalam otentikasi pengirim.

II. METODE

A. Studi Literatur

Langkah awal dalam penyusunan makalah ini adalah melakukan tinjauan pustaka dengan mencari informasi dari berbagai sumber digital seperti *paper*, jurnal, dan materi perkuliahan yang berkaitan dengan topik *blockchain*, *email*, RSA, *digital signature*, dan AES-256. Pada tahap ini, diperoleh informasi mengenai struktur, mekanisme, dan pengetahuan yang relevan dengan topik tersebut sebagai dasar untuk pengembangan lebih lanjut.

B. Implementasi

Setelah itu, penulis mengembangkan sebuah aplikasi berbasis *website* menggunakan React/NextJS dengan *library* yang dibutuhkan untuk mendukung fungsi kriptografi dari aplikasi. Program yang dikembangkan hanya menggunakan *blockchain* sederhana yang disimpan dalam bentuk file JSON. Program ini juga belum menyediakan fungsi-fungsi yang sudah disediakan dengan *email*, seperti melampirkan file biner dan mengirim *email* ke beberapa orang.

III. DASAR TEORI

A. Kriptografi

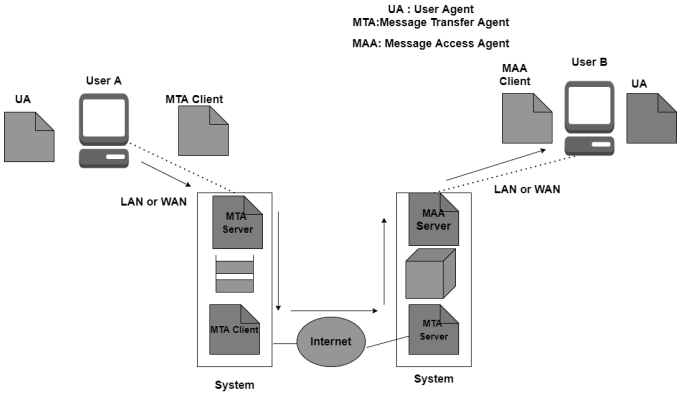
Kata kriptografi berasal dari bahasa Yunani, yaitu *cryptós* yang berarti rahasia atau tersembunyi dan *gráphein* yang berarti tulisan. Menurut Schneier, kriptografi adalah ilmu dan seni untuk menjaga keamanan pesan. Kriptografi menyediakan mekanisme keamanan untuk komunikasi antara kedua pihak. Empat layanan yang disediakan kriptografi untuk keamanan komunikasi antara lain [7]:

- Kerahasiaan pesan (*confidentiality*): memastikan pesan-pesan yang dikirim hanya dapat dibaca oleh pihak yang dituju oleh pesan tersebut.
- Keaslian pesan (*data integrity*): memastikan pesan tidak diubah atau dimanipulasi selama komunikasi.
- Keaslian pengirim dan penerima pesan (*authentication*): memastikan pengirim pesan adalah pihak yang asli dan bukan pihak lain yang menyamar sebagai pihak tersebut.
- Anti penyangkalan *(non-repudiation*): memastikan pengirim pesan tidak dapat menyangkal telah mengirimkan pesan.

B. Email

Email memiliki beberapa komponen yang membentuk layanan yang disediakan *email*, yaitu mengirimkan pesan elektronik dan menerima pesan tersebut. Komponen-komponen tersebut antara lain *user agent* (UA), *message transfer agent* (MTA), *message access agent* (MAA), dan *system/mail server* [8].

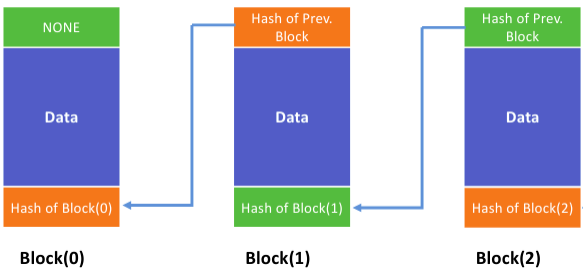
User agent akan digunakan untuk menulis *email* dan menampilkan *email* yang diterima. Ketika mengirimkan *email*, *user agent* berhubungan MTA *client* untuk mengirimkan email ke *mail server*. *Email* tersebut kemudian diterima oleh MTA *server* yang ada di dalam *mail server*. Komunikasi antara MTA *client* dan MTA *server* menggunakan protokol bernama *Simple Mail Transfer Protocol* (SMTP). MTA *client* dan MTA *server* juga berkomunikasi menggunakan SMTP ketika mail server mengirimkan *email* ke *mail server* tujuan melalui *internet*. Penerima dapat mengakses email menggunakan MAA *client* yang berkomunikasi dengan MAA *server* yang ada pada *mail server*. Komunikasi antara MAA client dan MAA server menggunakan protokol POP3 (*Post Office Protocol* 3) atau IMAP4 (*Internet Mail Access Protocol* 4). POP3 dan IMAP4 termasuk ke *pull program* karena *client* menarik *message* dari *server*, sedangkan SMTP termasuk ke dalam *push program* karena client mendorong/mengirimkan *message* ke *server* [8].



Gambar 1. Ilustrasi Arsitektur *Email* (Sumber: [9])

C. Blockchain

Blockchain adalah buku besar atau *ledger* yang terdistribusi untuk menyimpan dan mengelola transaksi secara kolektif dan terdesentralisasi [10]. Sifat *blockchain* yang terdistribusi dan terdesentralisasi ini menyebabkan *blockchain* tidak membutuhkan pihak ketiga yang mengatur setiap transaksi secara terpusat, seperti bank. Teknologi *blockchain* terdiri dari blok-blok yang berisikan data transaksi dan antara blok saling terhubung dengan menyimpan *hash* dari blok sebelumnya [10]. Oleh karena itu, perubahan data transaksi pada salah satu blok akan menyebabkan terputusnya rantai *blockchain* sehingga blok yang sudah ditambahkan ke *blockchain* tidak dapat diubah atau dihapus.



Gambar 2. Ilustrasi Struktur *Blockchain* (Sumber: [10])

Blockchain menggunakan arsitektur *peer-to-peer* untuk menghubungkan lebih dari satu *node* dalam *blockchain* [10]. *Node* adalah komputer yang tergabung dalam jaringan *blockchain* dan dapat berbagi sumber daya tanpa ada *server* yang bertugas sebagai kontrol pusat. Setiap *node* menyimpan catatan *record* yang sama sehingga jika salah satu *node* mencoba melakukan perubahan, *node* lain dapat memeriksa

dan menolak perubahan tersebut [10]. Hal ini mengakibatkan *blockchain* memiliki sifat *immutable*, yang berarti setiap blok data pada *blockchain* tidak dapat diubah oleh siapapun.

Ketika salah satu *node* melakukan *request* penambahan blok pada jaringan *blockchain*, dilakukan metode konsensus untuk menentukan *node* mana yang dapat menambahkan blok. Terdapat beberapa metode konsensus, seperti *proof of work*, *proof of stake*, *proof of elapsed time*, dan lain-lain. *Request* penambahan blok dapat dilakukan melalui *smart contract* yang merupakan program yang berjalan pada *blockchain*.

D. RSA

RSA adalah salah satu algoritma kunci publik yang paling terkenal. RSA dikembangkan oleh tiga peneliti dari MIT, yaitu Ronald Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1976. Algoritma RSA dimulai dengan pembangkitan kunci. Berikut adalah alur pembangkitan kunci pada algoritma RSA [11]:

1. Memilih bilangan prima p dan q. Bilangan p dan q sebaiknya tidak sama dan memiliki panjang lebih dari 100 digit untuk memastikan keamanan algoritma RSA.
2. Menghitung nilai n yang didapatkan dari perkalian antara p dan q.
3. Menghitung nilai Φ(n) yang didapat dari perkalian antara p – 1 dan q – 1.
4. Memilih nilai e sebagai kunci enkripsi (kunci publik). Nilai e yang dipilih harus relatif prima (memiliki PBB atau pembagi bersama terbesar bernilai 1) dengan Φ(n).
5. Menghitung kunci dekripsi d (kunci privat). Nilai d dapat dihitung dengan (1).

$$d \equiv e^{-1}(mod(\phi(n)) \text{ atau } d = \frac{1+k\phi(n)}{e}, \text{ k bilangan bulat} \quad (1)$$

Kunci publik yang didapatkan adalah pasangan (e, n), sedangkan kunci privat adalah pasangan (d, n) [11]. Proses enkripsi dengan algoritma RSA menggunakan kunci publik dengan persamaan (2). Pesan plainteks m dapat diubah menjadi cipherteks c dengan melakukan operasi mod dengan n terhadap plainteks m yang dipangkatkan dengan e. Jika pesan plainteks m berukuran lebih besar dibandingkan n – 1, plainteks m dapat dipecah menjadi beberapa bagian sehingga setiap bagian m memenuhi syarat $0 \leq m < n - 1$.

$$c = m^e \bmod n \quad (2)$$

Untuk mendekripsi cipherteks, digunakan kunci privat. Persamaan dekripsi dituliskan pada (3). Cipherteks dipangkatkan dengan kunci privat d lalu dilakukan operasi mod terhadap n untuk memperoleh pesan plainteks m.

$$m = c^d \bmod n \quad (3)$$

Keamanan algoritma RSA bergantung pada kesulitan memperoleh bilangan p dan q dari n. Oleh karena itu, nilai p dan q yang digunakan untuk RSA disarankan lebih dari 100 digit sehingga nilai n akan berukuran lebih dari 200 digit [11]. Untuk memperoleh nilai p dan q dari n yang berukuran 200 digit, dibutuhkan waktu komputasi selama 4 milyar tahun. Oleh karena itu, nilai n yang saat ini aman adalah yang berukuran di atas 1024 bit [11].

E. Digital Signature

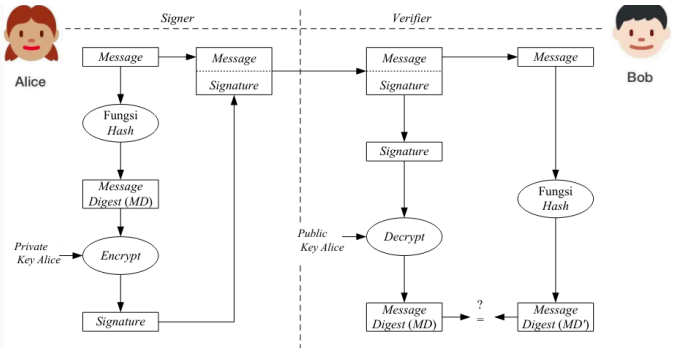
Tanda tangan digital adalah salah satu mekanisme dalam kriptografi untuk membuktikan dan memastikan identitas pengirim. Tanda tangan digital dapat digunakan untuk otentikasi dan anti penyangkalan (*non-repudiation*) [12]. Terdapat dua proses dalam tanda tangan digital, yaitu menandatangani pesan (*signing*) dan memverifikasi tanda tangan. Menandatangani pesan dapat dilakukan dengan dua cara, yaitu mengenkripsi pesan menggunakan kunci privat pengirim dan mengenkripsi *hash* pesan menggunakan kunci privat pengirim.

Tanda tangan digital dengan mengenkripsi pesan menggunakan kunci privat pengirim dilakukan dengan algoritma RSA [12]. Persamaan untuk *signing* dan verifikasi tanda tangan dapat dilihat pada persamaan (4) dan (5).

$$Signing: c = m^d \bmod n \quad (4)$$

$$Verification: m = c^e \bmod n \quad (5)$$

Tanda tangan digital dengan menggunakan fungsi *hash* dilakukan dengan melakukan fungsi *hash* terhadap pesan. *Hash* yang dihasilkan kemudian dienkripsi dengan kunci privat pengirim untuk menghasilkan tanda tangan digital [12]. Untuk memverifikasi tanda tangan digital, dilakukan fungsi *hash* terhadap pesan dan mendekripsi tanda tangan digital dengan kunci publik pengirim. Hasil dekripsi dan hasil *hash* pesan kemudian diperiksa apakah sama atau tidak. Jika sama, maka pesan tersebut berasal dari pengirim yang asli. Berikut ini adalah alur *signing* dan verifikasi tanda tangan digital.

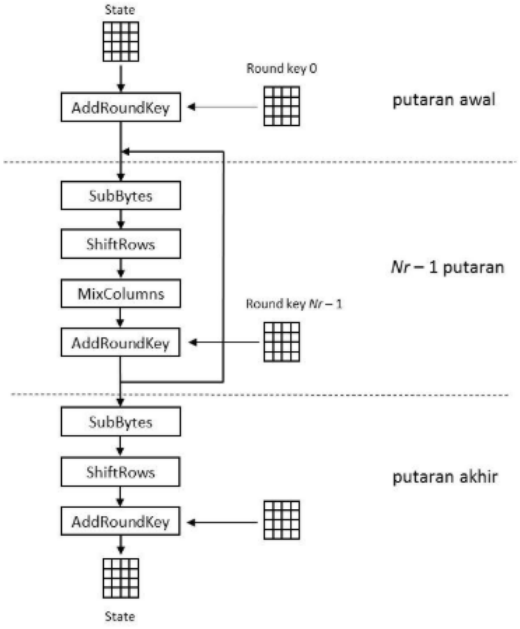


Gambar 3. Ilustrasi alur tanda tangan digital (Sumber: [12])

F. AES-256

Algoritma AES (*Advanced Encryption Standard*) atau algoritma Rijndael adalah algoritma kriptografi simetri yang berbasis *cipher block* [13]. Algoritma ini mendukung beberapa panjang kunci, yaitu 128, 192, dan 256-bit. Karena berbasis *cipher block*, algoritma ini beroperasi pada blok yang berukuran 128-bit. Proses enkripsi dilakukan dalam beberapa putaran, tergantung pada panjang kunci yang digunakan. Pada AES-256 yang menggunakan kunci dengan panjang 256-bit, dilakukan 14 putaran [13]. Kunci AES digunakan untuk membangkitkan kunci internal yang disebut *round key*. Setiap putaran menggunakan *round key* yang berbeda.

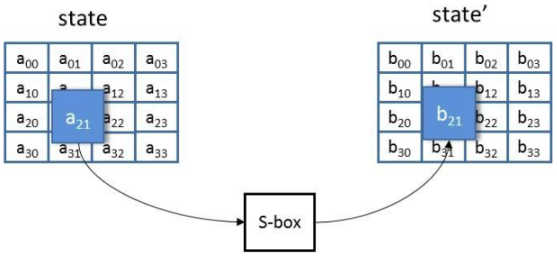
Berikut adalah ilustrasi yang menggambarkan langkah operasi algoritma AES secara garis besar.



Gambar 4. Langkah Operasi AES (Sumber: [13])

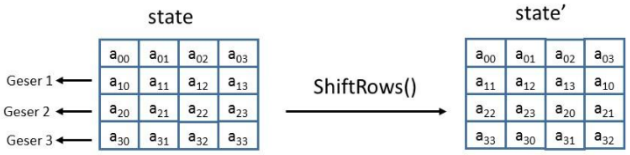
Berikut adalah operasi algoritma AES secara rinci [13].

1. Pada putaran awal, dilakukan operasi *AddRoundKey* yang merupakan operasi XOR antara *state* (*input* plainteks) dengan *round key* yang dihasilkan dari kunci AES.
2. Selama jumlah putaran – 1 (untuk AES-256, berarti 13 putaran), dilakukan operasi sebagai berikut.
    a. *SubBytes*: melakukan operasi substitusi setiap *byte* pada *state* dengan menggunakan *S-box*.



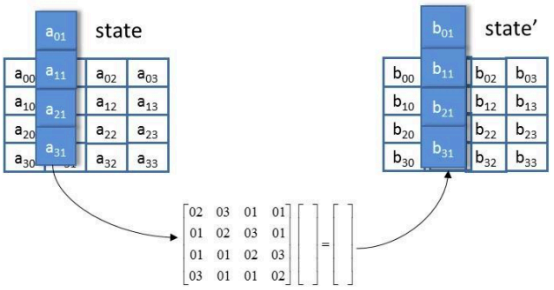
Gambar 5. Operasi *SubBytes* (Sumber: [13])

    b. *ShiftRows*: melakukan pergeseran secara *wrapping* pada 3 baris terakhir *state*. Jumlah pergeseran yang dilakukan tergantung urutan baris.



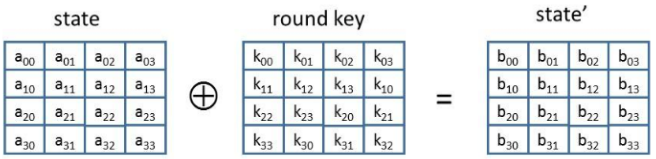
Gambar 6. Operasi *ShiftRows* (Sumber: [13])

    c. *MixColumns*: melakukan perkalian antara *state* dengan matriks tertentu untuk mengacak data.



Gambar 7. Operasi *MixColumns* (Sumber: [13])

    d. *AddRoundKey*: melakukan operasi XOR antara *state* dengan *round key* pada putaran tersebut.



Gambar 8. Operasi *AddRoundKey* (Sumber: [13])

3. Pada putaran terakhir (putaran ke 14 pada AES-256), dilakukan operasi *SubBytes, ShiftRows*, dan *AddRoundKey.* Putaran terakhir ini menghasilkan *state* yang menjadi cipherteks hasil enkripsi.

IV. RANCANGAN

Untuk meningkatkan keamanan *email*, dikembangkan sistem yang menggunakan teknologi *blockchain*, algoritma RSA, dan tanda tangan digital untuk menggantikan protokol *email*. Setiap *email* yang dikirimkan akan ditandatangani dan dienkripsi dengan RSA sebelum ditambahkan ke *blockchain* [14]. Selain itu, digunakan juga algoritma AES untuk mengamankan *database* yang berisi data akun pengguna [14].

A. Format Data Email, Akun, dan Perizinan Pengirim

Data *Email* dan perizinan pengirim disimpan pada *blockchain*, sedangkan data akun disimpan pada *database* [14]. *Blockchain* dan *database* yang digunakan disimpan dalam bentuk JSON (JavaScript Object Notation). JSON dipilih karena kemudahan untuk dimanipulasi sebagai tipe data *object* dalam JavaScript.

Data *email* memiliki lima *field*, yaitu *sender* yang berisi *username* dari pengirim, *receiver* yang berisi username dari penerima, *subject* yang berisi subjek *email* yang dikirimkan, *body*, yang berisi isi *email*, dan *signature* yang berisikan tanda tangan digital dari *field sender, receiver, subject*, dan *body*. Ketika data *email* ditambahkan ke *blockchain*, data *email* dimasukkan ke *field* data dan ditambahkan beberapa *field* lain seperti *timestamp* yang berisi waktu transaksi ditambahkan ke *blockchain*, *prevHash* yang berisi *hash* dari blok sebelumnya, serta *hash* yang berisi *hash* dari *field timestamp*, data, dan

prevHash. Berikut adalah rancangan format data *email* yang ditambahkan ke *blockchain*.

```
{
 "sender": "Samuel",
 "receiver": "Eric",
 "subject": "apa kabar",
 "body": "Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex
ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur.",
 "signature": "iniContohSignature"
}
```

Setelah ditambahkan ke *blockchain*, berikut adalah format data *email*.

```
{
 "timestamp": "1718008566906",
 "data": {
   "sender": "Samuel",
   "receiver": "Eric",
   "subject": "apa kabar",
   "body": "Lorem ipsum dolor sit amet,
consectetur adipiscing elit, sed do eiusmod
tempor incididunt ut labore et dolore magna
aliqua. Ut enim ad minim veniam, quis nostrud
exercitation ullamco laboris nisi ut aliquip ex
ea commodo consequat. Duis aute irure dolor in
reprehenderit in voluptate velit esse cillum
dolore eu fugiat nulla pariatur.",
   "signature": "iniContohSignature"
 },
 "hash": "iniContohHash",
 "prevHash": "iniContohPrevHash"
}
```

Perizinan pengirim memiliki tiga *field*, yaitu *for* yang berisikan *username* pemilik perizinan ini, *approvedSenders* yang merupakan *array* yang berisi *username* pengguna yang boleh mengirimkan *email* ke *username* pada *field for*, dan *signature* yang berisikan tanda tangan digital dari *field for* dan *approvedSenders*. Sama seperti data *email*, ketika ditambahkan ke *blockchain*, terdapat beberapa field yang ditambahkan seperti *timestamp*, *hash*, dan *prevHash*. Berikut adalah rancangan format data perizinan pengirim (*approved senders*)

```
{
 "for": "eric",
 "approvedSenders": ["samuel"],
 "signature":
"Mv7UIohcucN1jOlyGwJqvdkxO8P2bumgb+MgYsXMhQ58pno
OZFxSb87vJSFMHVkRni6RIdHZisehlTX+VqUEGA=="
}
```

Berikut adalah format data *approved senders* ketika ditambahkan ke *blockchain*

```
{
   "timestamp": "1718008470117",
   "data": {
     "for": "eric",
     "approvedSenders": ["samuel"],
     "signature":
"Mv7UIohcucN1jOlyGwJqvdkxO8P2bumgb+MgYsXMhQ58pno
OZFxSb87vJSFMHVkRni6RIdHZisehlTX+VqUEGA=="
   },
   "hash":
"07bad8e41876ebe3061d329582ed4dc6db17c8bb5cebf36
74f0a5bbf978da7ff",
   "prevHash":
"593accd953123b071e53d93f828dfe1fafcdc2f77aa5b57
798b4d2e730e0ed2e"
 }
```

Data akun disimpan dalam *database* dan bukan dalam *blockchain*. Data akun memiliki beberapa *field*, yaitu *username*, *password*, *walletAddress* yang berisikan alamat atau ID pengguna dalam *blockchain* (digunakan untuk menggantikan alamat *email*), *publicKey,* dan *privateKey*. Berikut adalah rancangan format data akun pada *database*.

```
{
 "username": "samuel",
 "password": "test1234",
 "walletAddress": "qwertyuiop",
 "publicKey": "iniContohPublicKey",
 "privateKey": "iniContohPrivateKey"
}
```

B. Rancangan Database dengan Algoritma AES

Database menyimpan data akun yang telah dibahas pada bagian sebelumnya. *Database* diamankan dengan algoritma AES sehingga hanya aplikasi yang memiliki kunci yang tepat yang dapat mengakses dan membaca data akun dari *database* [14]. Hal ini sangat penting karena *database* menyimpan data-data yang sensitif seperti *password* dan kunci privat pengguna.

Mode operasi yang dipilih untuk algoritma AES-256 adalah *counter*. Mode *counter* menggunakan *counter* yang digunakan untuk membantu meningkatkan keamanan enkripsi dan dekripsi dari AES. Kunci yang digunakan untuk AES memiliki panjang 256-bit dan dibangkitkan secara internal oleh aplikasi sehingga hanya aplikasi yang dapat mengakses data pada *database* dalam bentuk plainteks.

C. *Rancangan Sistem Email dengan Blockchain dan RSA*

Data *email* disimpan dalam *blockchain*. Sebelum ditambahkan ke *blockchain*, *email* ditandatangan dengan kunci privat pengirim untuk verifikasi identitas pengirim [14]. Hasil tanda tangan disimpan pada *field signature* pada data *email*. Data *email* kemudian dienkripsi menggunakan kunci publik penerima dengan algoritma RSA [14]. Hal ini memastikan hanya penerima yang dapat membaca *email* tersebut. Enkripsi RSA ini hanya dilakukan pada *field subject, body,* dan *signature*. *Field sender* dan *receiver* tidak dienkripsi agar aplikasi dapat mengetahui kunci penerima yang digunakan untuk enkripsi. Jika *field sender* dan *receiver* dienkripsi dengan kunci publik penerima, aplikasi tidak tahu pengguna yang menjadi penerima sehingga tidak mengetahui kunci privat dari pengguna mana yang harus digunakan untuk dekripsi.

Ketika ditambahkan ke *blockchain*, semua *field* termasuk *prevHash* pada blok akan di-*hash* menggunakan algoritma SHA-256. Hasil *hash* akan disimpan pada field *hash*. *Hash* dari setiap blok dan penyimpanan nilai *hash* dari blok sebelumnya membangun rantai antara blok yang akan putus jika data salah satu blok diubah.

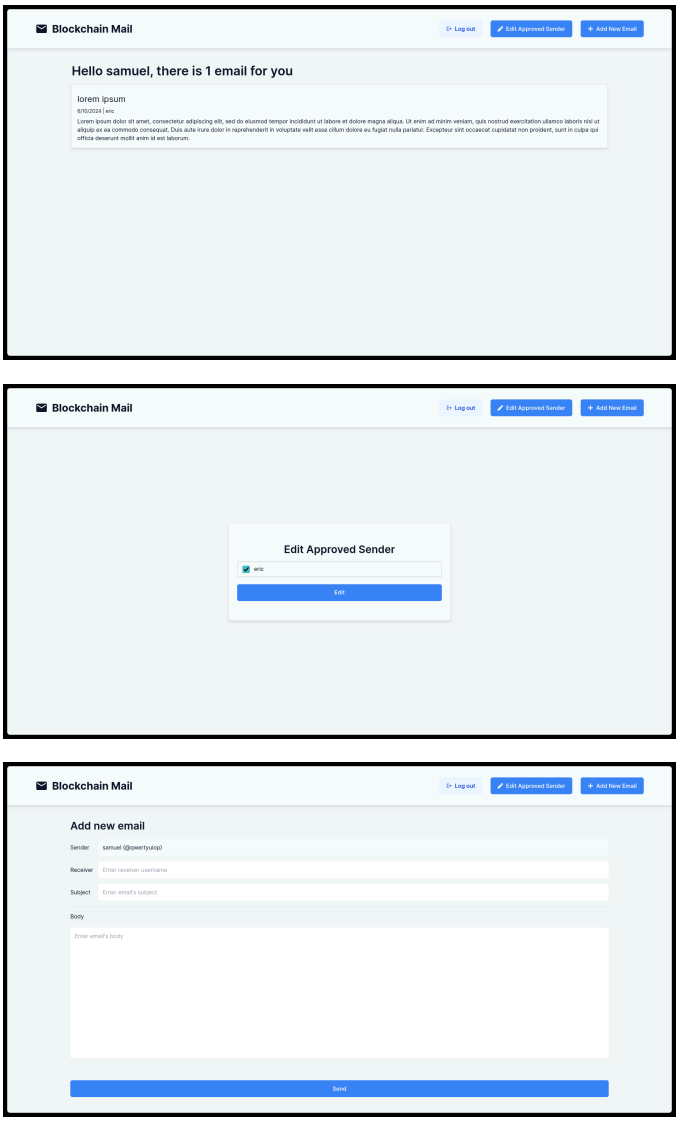
D. *Rancangan Sistem Perizinan Pengirim dengan Blockchain*

Sebelum dapat mengirim *email* ke pengguna lain, pengirim harus ditambahkan ke daftar *approved senders* yang dimiliki pengguna tersebut. Daftar *approved senders* digunakan oleh penerima untuk menentukan pengguna yang diizinkan untuk mengirimkan *email* ke penerima. Jika pengirim tidak ada dalam daftar *approved senders* yang dimiliki penerima, pengirim tidak dapat mengirimkan *email* ke penerima tersebut.

Data *approved senders* disimpan pada *blockchain*. Sebelum ditambahkan ke *blockchain*, *field for* yang berisikan pemilik dari daftar *approved senders* dan *field approved senders* yang berupa *array* yang berisi *username* pengguna yang diperbolehkan mengirim, ditandatangani dengan kunci privat pemilik daftar *approved senders* untuk verifikasi kepemilikan data tersebut. Hasil tanda tangan disimpan pada *field signature* yang ditambahkan ke data *approved senders*. Setelah ditandatangan, data *approved senders* ditambahkan ke *blockchain* tanpa enkripsi. Ketika blok ditambahkan ke *blockchain*, semua field akan di-*hash* dengan algoritma SHA-256. Hasil *hash* ini disimpan di field *hash*.

V. IMPLEMENTASI

Implementasi penggunaan *blockchain* untuk *email* berdasarkan rancangan yang dijelaskan pada bagian sebelumnya dilakukan dengan bahasa pemrograman JavaScript. Untuk *framework*, digunakan NextJS untuk mengembangkan antarmuka dari aplikasi yang dibangun. Bahasa JavaScript dipilih karena menyediakan banyak *library* yang dapat digunakan untuk implementasi. Fungsi kriptografi yang dilakukan aplikasi seperti enkripsi AES dengan *database*, operasi algoritma RSA, dan tanda tangan digital diimplementasi menggunakan *library-library* dari NPM (*Node Package Manager*).



Gambar 9. Tampilan Antarmuka Aplikasi

Proses enkripsi, dekripsi, dan tanda tangan digital antara *blockchain* dan *database* dilakukan pada *server* yang disediakan melalui *route handler* oleh NextJS.

A. *File-file untuk Database*

Database disimpan dalam file db.json. Enkripsi dan dekripsi *database* menggunakan AES-256 dilakukan pada *file route handler* yang berada di *folder app/api/user*.

```
// random generated
const key = [
  85, 74, 19, 10, 28, 51, 25, 35, 88, 44, 34, 27, 33, 95, 59, 20, 93, 50, 39,
  94, 60, 30, 55, 2, 68, 62, 41, 14, 70, 22, 56, 76,
];
```

Gambar 10. Kunci AES yang Digunakan Aplikasi

```
// generate public and private key
const rsaKey = new NodeRSA({ b: 512 });
const privateKey = rsaKey.exportKey('pkcs8-private');
const publicKey = rsaKey.exportKey('pkcs8-public');
// encrypt using aes
const aesCtr = new aesjs.ModeOfOperation.ctr(key);
const encryptedData = {
  username: aesjs.utils.hex.fromBytes(
    aesCtr.encrypt(aesjs.utils.utf8.toBytes(body.username))
  ),
  password: aesjs.utils.hex.fromBytes(
    aesCtr.encrypt(aesjs.utils.utf8.toBytes(body.password))
  ),
  walletAddress: aesjs.utils.hex.fromBytes(
    aesCtr.encrypt(aesjs.utils.utf8.toBytes(body.walletAddress))
  ),
  publicKey: aesjs.utils.hex.fromBytes(
    aesCtr.encrypt(aesjs.utils.utf8.toBytes(publicKey))
  ),
  privateKey: aesjs.utils.hex.fromBytes(
    aesCtr.encrypt(aesjs.utils.utf8.toBytes(privateKey))
  ),
};
```

Gambar 11. Proses Penambahan Data Akun ke *Database*

Gambar 11 memperlihatkan proses penambahan data akun ke *database*. Ketika *sign up*, pengguna hanya memasukkan *username, password*, dan *wallet address*. *Private key* dan *public key* dari pengguna dikembangkan oleh aplikasi tanpa masukan pengguna. Lalu setiap *field* dienkripsi menggunakan algoritma AES-256 dengan mode *counter* yang disediakan oleh *library aes-js*. Setelah dienkripsi, data ditambahkan ke *file* database.json.

```
const decryptedUsers = [];
encryptedUsers.forEach((encryptedUser) => {
  const aesCtr = new aesjs.ModeOfOperation.ctr(key);
  const decryptedUser = {
    username: aesjs.utils.utf8.fromBytes(
      aesCtr.decrypt(aesjs.utils.hex.toBytes(encryptedUser.username))
    ),
    password: aesjs.utils.utf8.fromBytes(
      aesCtr.decrypt(aesjs.utils.hex.toBytes(encryptedUser.password))
    ),
    walletAddress: aesjs.utils.utf8.fromBytes(
      aesCtr.decrypt(aesjs.utils.hex.toBytes(encryptedUser.walletAddress))
    ),
    publicKey: aesjs.utils.utf8.fromBytes(
      aesCtr.decrypt(aesjs.utils.hex.toBytes(encryptedUser.publicKey))
    ),
    privateKey: aesjs.utils.utf8.fromBytes(
      aesCtr.decrypt(aesjs.utils.hex.toBytes(encryptedUser.privateKey))
    ),
  };
  decryptedUsers.push(decryptedUser);
});
```

Gambar 12. Proses Dekripsi Data Akun

B. *File-file untuk Blockchain*

Blockchain disimpan dalam file blockchain.json. Proses penambahan data ke *blockchain* dan pembacaan data dari *blockchain* dilakukan pada *file-file* di *folder app/api/email* dan *app/api/permission*. Penambahan dan pembacaan data pada *blockchain* juga dibantu oleh file blockchain.js yang ada pada *folder utils*.

```
class Block {
  constructor(data = {}) {
    this.timestamp = Date.now().toString();
    this.data = data;
    this.hash = this.getHash();
    this.prevHash = '';
  }

  getHash() {
    return sha256(this.timestamp + JSON.stringify(this.data) + this.prevHash);
  }
}
```

Gambar 13. *Class Block* pada *File Blockchain.js*

Class Block digunakan oleh *class Blockchain* untuk membentuk blok yang ditambahkan ke *blockchain*. Pada *class Blockchain*, jika *blockchain* tidak memiliki blok, akan ditambahkan sebuah blok dengan data kosong (*genesis block*) sebelum ditambahkan blok lain yang berisikan data *email* atau *approved senders*.

```
class Blockchain {
  constructor() {
    this.chain = [new Block()];
  }

  getLastBlock() {
    return this.chain[this.chain.length - 1];
  }

  addBlock(block) {
    block.prevHash = this.getLastBlock().hash;
    block.hash = block.getHash();

    this.chain.push(block);
  }

  isValid(blockchain = this) {
    for (let i = 1; i < blockchain.chain.length; i++) {
      const currentBlock = blockchain.chain[i];
      const prevBlock = blockchain.chain[i - 1];

      if (
        currentBlock.hash !== currentBlock.getHash() ||
        prevBlock.hash !== currentBlock.prevHash
      ) {
        return false;
      }
    }
    return true;
  }

  saveToJson() {
    console.log(this.chain);
    const json = JSON.stringify(this.chain);
    fs.writeFileSync('data/blockchain.json', json);
  }

  readFromJson() {
    const data = fs.readFileSync('data/blockchain.json', 'utf8');
    const blockchain = JSON.parse(data);
    if (blockchain.length === 0) {
      this.chain = [new Block()];
    } else {
      this.chain = blockchain;
    }
  }
}
```

Gambar 14. *Class Blockchain* pada *File Blockchain.js*

```
// tanda tangan email
const emailBody = {
  sender: body.sender,
  receiver: body.receiver,
  subject: body.subject,
  body: body.body,
};
const rsaSenderKey = new NodeRSA();
rsaSenderKey.importKey(body.senderPrivKey, 'pkcs8-private');
const signature = rsaSenderKey.sign(JSON.stringify(emailBody), 'base64');
// enkripsi email pakai kunci publik penerima
const rsaReceiverKey = new NodeRSA();
rsaReceiverKey.importKey(body.receiverPubKey, 'pkcs8-public');
const encrytedData = {
  sender: body.sender,
  receiver: body.receiver,
  subject: rsaReceiverKey.encrypt(body.subject, 'base64'),
  body: rsaReceiverKey.encrypt(body.body, 'base64'),
  signature: rsaReceiverKey.encrypt(signature, 'base64'),
};
// tambah ke blockchain
emailBlockchain.readFromJson();
emailBlockchain.addBlock(new Block(encrytedData));
emailBlockchain.saveToJson();
```

Gambar 15. Proses Pengiriman *Email*

Gambar 15 memperlihatkan proses pengiriman *email* dan penambahan *email* ke *blockchain*. Data *email* yang terdiri dari *sender, receiver, subject*, dan *body* ditandatangani menggunakan kunci privat pengirim. Setelah itu, *field subject, body*, dan *signature* dienkripsi menggunakan kunci publik penerima sehingga hanya penerima yang dapat membaca isi *email* tersebut. Data *email* yang sudah dienkripsi kemudian ditambahkan ke *blockchain* menggunakan metode *addBlock* dari *class Blockchain*.

```
// baca blockchain
emailBlockchain.readFromJson();
const encryptedReceivedEmails = emailBlockchain.chain
  .filter((email) => {
    return email.data?.receiver === username;
  })
  .map((email) => ({ ...email.data, timestamp: email.timestamp }));
if (encryptedReceivedEmails.length > 0) {
  // dekripsi email pakai kunci privat penerima
  const res = await fetch(`${process.env.URL}/api/user/${username}`);
  const data = await res.json();
  const receiverPrivKey = data.user.privateKey;
  const rsaReceiverKey = new NodeRSA();
  rsaReceiverKey.importKey(receiverPrivKey, 'pkcs8-private');
  const decryptedReceivedEmails = encryptedReceivedEmails.map((email) => {
    return {
      sender: email.sender,
      receiver: email.receiver,
      timestamp: email.timestamp,
      subject: rsaReceiverKey.decrypt(email.subject, 'utf8'),
      body: rsaReceiverKey.decrypt(email.body, 'utf8'),
      signature: rsaReceiverKey.decrypt(email.signature, 'utf8'),
    };
  });
  // verifikasi tanda tangan email
  const receivedEmails = await Promise.all(
    decryptedReceivedEmails.map(async (email) => {
      const res = await fetch(`${process.env.URL}/api/user/${email.sender}`);
      const data = await res.json();
      const senderPubKey = data.user.publicKey;
      const rsaSenderKey = new NodeRSA();
      rsaSenderKey.importKey(senderPubKey, 'pkcs8-public');
      const emailBody = {
        sender: email.sender,
        receiver: email.receiver,
        subject: email.subject,
        body: email.body,
      };
      const isValid = rsaSenderKey.verify(
        JSON.stringify(emailBody),
        email.signature,
        'utf8',
        'base64'
      );
      return { ...email, isValid };
    })
  );
```

Gambar 16. Proses Pembacaan *Email* dari *Blockchain*

Gambar 16 memperlihatkan proses pembacaan data *email* pada *blockchain*. Data *email* milik pengguna dibaca dari *blockchain*. Kemudian data *email* tersebut didekripsi dengan kunci privat penerima. Setelah didekripsi, Setiap *email* diverifikasi tanda tangannya.

```
// tanda tangan pakai kunci privat pengguna
const key = new NodeRSA();
key.importKey(body.privKey, 'pkcs8-private');
const data = {
  for: body.username,
  approvedSenders: body.approvedSenders,
};
const signature = key.sign(JSON.stringify(data), 'base64');
data.signature = signature;
// tambah ke blockchain
emailBlockchain.readFromJson();
emailBlockchain.addBlock(new Block(data));
emailBlockchain.saveToJson();
```

Gambar 17. Proses Penambahan Data *Approved Senders*

Gambar 17 memperlihatkan proses penambahan data *approved senders* pada *blockchain*. Data pengirim yang diizinkan ditandatangani dengan kunci privat pengguna, lalu ditambahkan ke *blockchain*.

```
// baca blockchain
emailBlockchain.readFromJson();
const approvedSenders = emailBlockchain.chain
  .filter((block) => {
    return block.data?.for === username;
  })
  .sort((a, b) => b.timestamp - a.timestamp)
  .map((block) => block?.data);
if (approvedSenders.length !== 0) {
  // verifikasi tanda tangan digital
  const res = await fetch(`${process.env.URL}/api/user/${username}`);
  const data = await res.json();
  const pubKey = data.user.publicKey;
  const validApprovedSender = approvedSenders
    .map((sender) => {
      const key = new NodeRSA();
      key.importKey(pubKey, 'pkcs8-public');
      const senderData = {
        for: sender.for,
        approvedSenders: sender.approvedSenders,
      };
      senderData.isValid = key.verify(
        JSON.stringify(senderData),
        sender.signature,
        'utf8',
        'base64'
      );
      return senderData;
    })
    .filter((sender) => sender.isValid)[0];
```

Gambar 18. Proses Pembacaan Data *Approved Senders*

Gambar 18 memperlihatkan proses pembacaan data *approved senders* pada *blockchain*. Data *approved senders* milik pengguna dibaca dari *blockchain* dan diverifikasi tanda tangan. Perubahan data *approved senders* akan mengakibatkan penambahan data yang baru pada *blockchain*. Oleh karena itu, pembacaan data *approved senders* dari *blockchain* di-*filter* berdasarkan *timestamp* (*timestamp* ditambahkan ketika ditambahkan ke *blockchain*) yang terbaru dan yang dapat diverifikasi tanda tangannya.

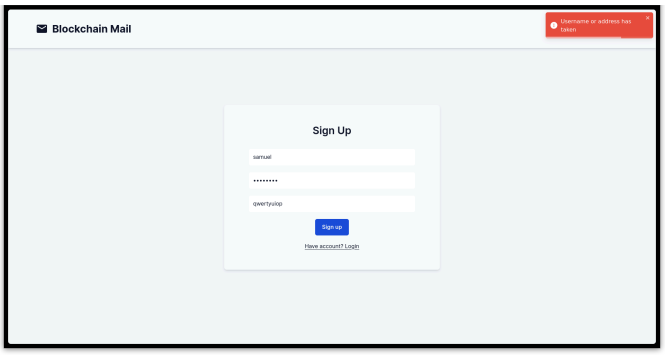
C. File-file untuk Antarmuka

Antarmuka yang dikembangkan berbasis *website* menggunakan React dan terdapat pada *file* yang bertipe jsx.

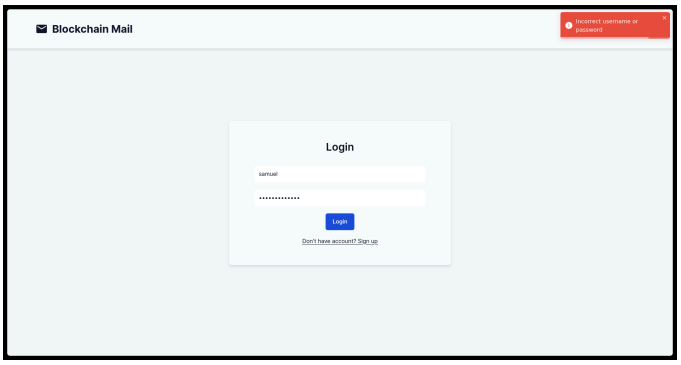
VI. Pengujian dan Diskusi

Setelah implementasi selesai dilakukan, akan dilaksanakan proses pengujian untuk memastikan apakah program telah memenuhi seluruh fungsi yang dibutuhkan. Berikut adalah pengujian program berdasarkan fungsi yang disediakan aplikasi.

A. Pengujian Fungsi Login dan Sign Up



Gambar 19. Tampilan Aplikasi ketika *Sign Up* dengan Username atau Wallet Address yang Sudah Ada Sebelumnya



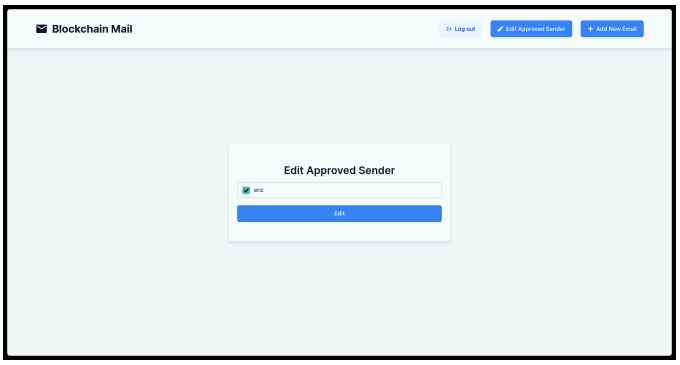
Gambar 20. Tampilan Aplikasi ketika *Login* dengan *Username* atau *Password* yang salah


```
{
  "username": "484bd0211693",
  "password": "33f82447351f7495",
  "walletAddress": "ce2c3e855d8149b3b741",
  "publicKey":
"79609dedfc90517eaa9165cb2da113d1b40a8029e47791fdc85dae331c7fb38c11161052c3567beceadcb1cb1616d59a5b
b91eadb2fe975262ad403801f3c5f03a090ffd3042f25920d4cc876b3120318db2ee6e0a82ff09c6ec7122ba34ffda5673e
aada51c1dec46dc0dbf42b86b311cb70cfba10a6ad2961bd3fa083a90088d6f33056595a08f6df2ff8a90576b13b9357015
b9984c6ea7d3bd27f727f94d3f98ef4605150094ed2a14236f189e58400cdb0559",
  "privateKey":
"1f5c034c8206cc1dc5c6881f6b9338ba35201cbed12e885406b8af861062f3bef33604c4fb15fa979af84c23052dc20561
3c67bee1dc4ac41041f565d50d7728eb6b4687cb92a91dc1e7fb3d173d8c464aa758148efe226684e0257eb006fcba9a888
3a56b6ff036bad7b8cb9101bcf6f0934bb15628a2856c9fa6d1aede5b361b5a31b6eb361f9bdfdff0fe6c434239f594c614
65a5c05744c598e41f9e7e4cdb2f907dfc4e6554b4bd445a3770c45e265c09c0a4b9158f1c5b6881fc252e077a38b9f340c
41a73060afe57ed28c09664b3c6498d0fd3131d09057660b5cb5593c5472a792830dea71223f19dde371af05ab0d711feac
57341091268312333a3c6353067d03a04c88feed8f19ae3da765af64e59df4e5b787805b08fcdb2959dd078d917e473dd06c
19f8c8b4d1b688be1d8637f189fba076e88307cbf27d538c80610d82f109253663fe0df357f07a8f42a2987913b03e5af51
a894fc75c40cf2f1970be95d6c8b7cb01bffc79f051adc2af7981dfae7c8f5063ae3f69052046314cdc56c276600d2f5305
052fb45df1633456523f8cbc0c35defc2e89f2f392bb3104188f369a5729d0fb7c96586771c25512c983632de0713d1e4c5
cb1367257ff362768a1c2ff2d9d0b9a819760703ed5defb85e9811bb890dd7268221974542f7c057dfc508dd99ee4e07a5d
843adb51112f9327f3d087f2097fd8865a2191a3321b70251df99"
},
```


Gambar 21. Struktur Data Akun pada *Database*

Sign up dilakukan dengan mengisi *username, password*, dan *wallet address*. Jika *username* atau *wallet address* sudah digunakan oleh pengguna lain, maka aplikasi akan mengeluarkan *error*. Aplikasi juga akan mengeluarkan *error* ketika melakukan *login* dengan *username* atau *password* yang salah. Data akun disimpan pada *database* dalam bentuk terenkripsi menggunakan algoritma AES-256 sehingga data hanya dapat diakses oleh aplikasi.

B. Pengujian Fungsi Penambahan Approved Senders



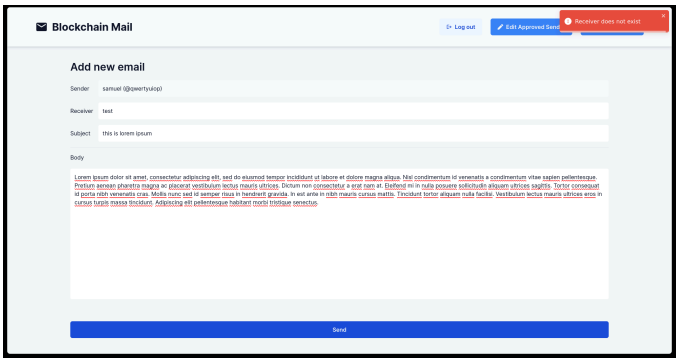
Gambar 22. Tampilan Aplikasi untuk Perubahan *Approved Senders*


```
{
  "timestamp": "1718022305580",
  "data": {
    "for": "samuel",
    "approvedSenders": ["eric"],
    "signature": "VlRLD4Cv5BU8v0IKtmrmlNbNAtPJkJLCB1NOlc+ye18tc/Ffkgm2ca0Y5rDli8PR4kO2e+B634LsadDFCf0nIQ=="
  },
  "hash": "d866931680989985a25fc9ff815473112cc0dd2c51bb4c7d2d20c5c14f590f448",
  "prevHash": "94a4456713d0caab6091e643751c7287e309e5a251cfcfcb93666d229fe4bbc2"
},
```

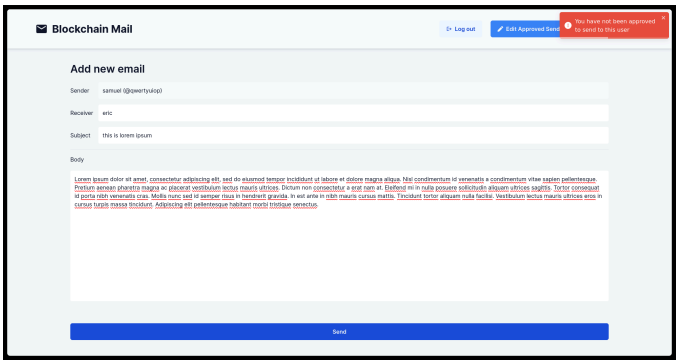

Gambar 23. Struktur Data *Approved Senders* pada *Blockchain*

Data *approved senders* ditandatangan menggunakan kunci privat pengguna lalu ditambahkan ke *blockchain*.

C. Pengujian Fungsi Pengiriman Email



Gambar 24. Tampilan Aplikasi jika Tidak Ada *Receiver*



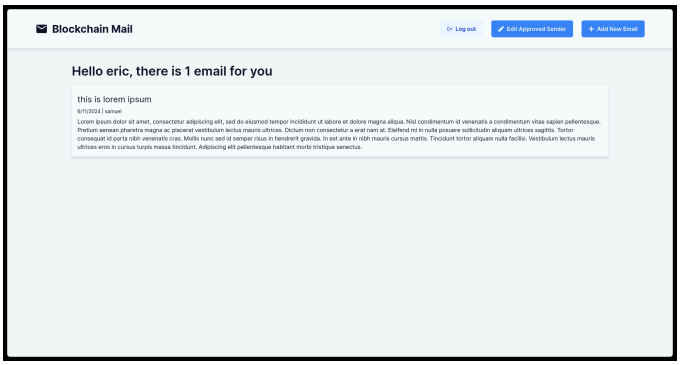
Gambar 25. Tampilan Aplikasi jika *Sender* Belum Dimasukkan ke Daftar *Approved Senders* dari *Receiver*

```
{
  "timestamp": "1710071449438",
  "data": {
    "sender": "samuel",
    "receiver": "eric",
    "subject": "WagI8zbLVo1NH3UgnX0w3iui1FniyyWrFN4TqPb4Y8ke/yTNEY3osAxncZrDZdb/A7+COcYt7wWHdlcvQNWQ==",
    "body": "ETIVAx63+JP501x6815qUreHjTSN5JTis+wNKhN3LRd5WFdlJPxipfJh2jAGDLdlvZz63YfaFrppH9TTmo2HyTk2eDcqSmYl9kaFs7mejdRGk3TQSc/
tugcQQlPFFKTjRHkbZvNO2kFSKFu3aGEs1sSfqJePehxQL0ZEkte0HUETkYK5fEUF2c6apwwiSUWtqUe/
QMwgRyZOCsIfgl1Zd3pS3ta5eNHh2ocb23QTpOxew2fXRJQO6EqYBg4HwBqPveicT835Vuh8NwRF6NUV1c4f2kjbBjO5oSS33yLsWE1hdMsGzpB1CQv8v1SK/
+qcRPJ85kNcXfI5QaslTNOt8ajVFtVpxI2fyshS25OCo/7CvqXkjlH6vGn8JhF5AGxqp36UAD/OhwtEeehe/M2urzREIfbYfJwcZdpmqHxqbqCgLh8CPVIprcxAmlP04rFaKbIIvMQl90
+STnc1ISmJV3v/hQME38kN079FOtAlIWP3Gn4dATfrRsMVujBsBjyNeaaaV6lthAEkSwQ1vAOPlBt4vyxYO7v6tm/
AJOSTzITRgNN4CuZH8WfK9Oc8vQweS6nKQjt1vSvgHJKNk9y8nAYUki3Zb12mWylRbjT5B9Oe2o+1qNE7l61hyCdsQzW/NrCCONPoSUUt/D08Q1eM
+oXsMppJJaR33xbBcgDCv2jQGeAUfEq76Ph3GVvDZCLenEtiKMffY5e76lJnf8QMWq7VW01b1VpAiQwrFbS83Z3SW1KeAm+VKKhUdE+RH1B+sAgm+vkfvf1DyTZT3Czm9aE4dL5ThX/
pjxNFLpipcwWesqFRJyKYA6AkA/
1noo1VOkEWlpN6VFJThe06FBg9UdOq7RKwNdGLpbiPWFyKWppAVOFteyxNPjotoGbX2YjEWIuOMBiOSMhZ1UfOc5KnAUSkORQGhCwSEkDB8n1oauuMpmWcPkoY6c4nNxStigmIgmPyZoRVuG26ibQ2ihdOnhOTYKB6c69oJ6xAxX3eGBElgTL8rCpN9d90tmJ1d5BvOhLLFEDXGzHCoWneZb8IVN5E1S2O1Q2WE5HW6CeLpEpDSACH/yFok
+NxOu0yrMOuf6vguEvu08B8BJiKYZmvy195ZQkZJwrnVQxWQGHrDK3ol9Gi7aDvMVDbYiA7a5ICQiP57NIVucz3NFgSl/
LNBJMokvghqv4upkvf3NNAli3tBOLMqYlIJXSrshopVHxtMDwNCvBUmlKh2iiu/T7O10PfRQmlIxayF
+UkKenb5UVUKcn0kn9l8MOTou2TJYtZl7LXO2SZfpRmkQolRkfPSUis1WN7hlH23ICy9aL/YiqMJlbplqACgQrcXYomZPe8HggKXTdmALnQYgmqCI9YCoAJl2YU2/
lygcFWWl5hJQTqlzYTGFbmk1ETlHqVXrp3+9Yj50NUGL4JTOmaVKjlvXhrqoRQNWKJJi1AUEJ6F5Xs6JHyWf2PAugJl9aQyQBLF3B5Fz7n1Z6YWSD9Sh5bFz/7htLZf8tV2b7oPAx1/
uwrtLMMvQaZFq+mMWfiP5Hu2rRgXeItyLUtAP/DXerd36NRTJT3RzdTIIsYWmvt4wx2Y9tPeMe1jnFG13TuBCI+WqyZNpSfQJol5E4in98j7CGCQVy/WbYBKEXRp4uIXpP
+opv6y1q4i4OvROspB0VesN56XMkaBgSp7eIzWJ7yc4QbCM7MKWfGf5DLh67LPupGDCEV1+P01HDlJK1PODS67IX/mIgCFSlmnOqU+89/r39fskc/
egET9pJKyYd5EQs3FRcZM2Zd2JKtyjGMCIKc1JCAQgYWyYEr81yOwUv/a1mqxIkOn38BqOVd4f5TGqKjwhAhvJlxcraJ5gCa0H8bnJLJaTLnTBmKt2LsH/WGjkOFa5HKVxar7y2qeU/
RvQex0zTiGQQ2Ol4lRaKZf7LVld6jGntQVUNZ5Tyeg/APG6i5rfHUeHR3gyUtbkuBL3injmizR0CCtRXVjwHePO3iq7tnFW2IzKQLKjTV+x/LUaNG40appYbBg9
+XKCFT2jLUHZV80Ed5CAGetcQqRwe9fzthDSc//u0P3v5D56eY2v3z1hkNgbPbtq5oW9xFPOGEsqgfRfZeeVE+d2EjN8jJX3+pzhJroWhnvADFL3oVWouFW/BA7iPYANZ7Peo2ebO94vozm/
YjtbT3ymgehS/elBwqpGrR0gIEXQwkxgXwnOTiJlyFeq3AiZmQZVZ/IrebH/
7tE8ZJ1nPQowKBWhzNREG2B6S2IdmzHB8xICi4Uk7yWvB8O3bqGoiHHIQqfQi4CVsiVZ2derU5gBabvXqBGGzm07fNbPyZaSBfyyA0XppsNcNFnn0YipdL3PJ7WmNnLrN7TLgXNS10895V9kLIM
08bHa3du92/6e5l9v3BaOL83jh2t/Dx4dNN8+nqxMKCZz85lfNFRa0XdDdHe/
e8P589Eldk8Nq2QGd69lOW55xYIayJkTUYBoUn17vffFOGxQJc8cJAyt3ZgamSTeTl9zzqkJUwwTiLeyWVYgrRntURnXpJKmkMeFcytBmNyxb
+2bKVC7jSh7WfQXtuaxz83VumBTJdfaYJoPs1vFiZWkZ1kJm1H6DBCE6swJRIOMI9p8Sdn3QJQcHeDL+AfIeFbnfnMUC/UI7boFMGm0/
t3RH2bphGuS8qc1XiplHnuIZvbiwawWRd8nABQqgAJTkjTXJBAXd4niMiyc8GAjwQrgs+/
"signature": "GKHFaHReRE0w2+6Lvh/f2QOrti7f0HotAoUYgjnZzzfjPwhAeWgZixyRpQguPdCJDe3bLJFCp51trZf3TW3ScLKo/BtSBgSQ7lVmIVLM/
aeqUhaLGvBJTAfpfvl9kLjv9yoZX6mGE+ITKlCdTd5/l18T5X7+xIIQuf8NzZjqN+yCJHCQ1+9AqxN8VpSRQzmR5OvB1qOzypMgIkKlBEfNmkBDKgY9M3uuwuuhNGGJeoCgNpJYqFcMwVXL/
3W0wTS2mopdVV3YWm+4Cgq0RPAgHGNrhilyl2sjlYDS0e++DmNh8l7UXoC9L5JkZ/kWEVTA==",
  },
  "hash": "072c0a9bb23719f216eff980a91044d4c443400e822bc9577d1a527a084ced75",
  "prevHash": "235bd254efbe2591f5a6afd5ea2b18d23aecdd8b604af7724if0e8ce4561dc11"
}
```


Gambar 26. Struktur Data *Email* pada *Blockchain*

Sebelum dapat mengirimkan *email* ke penerima, penerima tersebut harus menambahkan pengguna ke daftar *approved sender* yang dimilikinya. Jika tidak, aplikasi akan memunculkan *error*. Proses pengiriman *email* sama seperti yang dibahas pada bagian implementasi, yaitu tanda tangan menggunakan kunci privat pengirim, enkripsi dengan kunci publik penerima, dan ditambahkan ke *blockchain*.

D. Pengujian Fungsi Penerimaan Email



Gambar 27. Tampilan Aplikasi *Email* dari Sisi Penerima

Email yang diterima dibaca dari *blockchain*, kemudian di dekripsi dengan kunci privat penerima, dan diverifikasi tanda tangan digital pada *email*. *Email* kemudian ditampilkan pada aplikasi.

E. Evaluasi

Penggunaan *blockchain* membantu menyelesaikan risiko keamanan yang dihadapi *email* saat ini. Berikut adalah setiap risiko keamanan yang dihadapi oleh *email* saat ini dan bagaimana penggunaan *blockchain* dapat membantu mengurangi risiko keamanan tersebut.

1. Tidak adanya autentikasi

   Dengan penggunaan *blockchain*, setiap pengguna harus memiliki akun pada *blockchain* untuk memperoleh *address* atau alamat akun yang digunakan sebagai pengganti alamat *email*. Setiap *email* yang dikirimkan juga ditandatangani dengan kunci privat yang hanya dimiliki pengirim. Hal ini mengurangi risiko terjadinya *spoofing* atau *identity theft* pada sistem *email* [14].

2. Tidak adanya enkripsi

   Pada sistem email yang dibantu dengan *blockchain*, setiap *email* dienkripsi menggunakan kunci publik penerima. Hal ini dikarenakan *blockchain* bersifat publik sehingga setiap transaksi dapat dilihat oleh semua orang. Dengan adanya enkripsi, setiap *email* yang ditambahkan ke *blockchain* hanya dapat didekripsi dengan kunci privat penerima sehingga mengurangi risiko *eavesdropping*.

3. Arsitektur yang terpusat

   Email disimpan pada *blockchain* yang terdistribusi dan ketika telah tersimpan, *email* tidak dapat lagi diubah (*immutable*). Dengan *blockchain*, *email* tidak lagi memiliki *single point of failure* yang dapat menjadi sasaran serangan *hacker*.

4. Akuntabilitas yang terbatas

   Setiap *email* ditandatangani dengan kunci privat yang dimiliki pengirim sehingga menyediakan cara untuk melacak sumber/asal *email*. Hal ini mempermudah *bad actor* bertanggung jawab atas tindakan mereka dan menyediakan *non-repudiation* [14] (pengirim tidak dapat menyangkal telah mengirim *email* karena *email* ditandatangani dengan kunci privat yang hanya dapat diakses oleh pengirim).

Terdapat beberapa kelemahan dari penggunaan *blockchain* untuk *email*. Penggunaan *blockchain* untuk menggantikan *email* membutuhkan waktu yang cukup lama untuk memperoleh popularitas [15]. Hal ini dapat diselesaikan dengan solusi yang masih menggunakan arsitektur *email* saat ini dan ditambahkan *blockchain* untuk meningkatkan keamanan, dibandingkan menggantikan protokol *email* dengan *blockchain* [15]. Pada awalnya, *blockchain* dapat ditambahkan hanya untuk keamanan *email* dan sebagai langkah *anti-phishing* dibandingkan menyimpan seluruh data *email* pada *blockchain* [15]. Penggunaan *blockchain* untuk *email* juga menghadapi masalah skalabilitas. Hal ini dapat diselesaikan dengan menggunakan *sidechain* yang mengurangi beban komputasi pada *blockchain* utama sehingga kapasitas lebih mudah ditingkatkan [15].

VII. KESIMPULAN

A. Kesimpulan

Spam email dan *phishing attack* merupakan ancaman besar bagi keamanan data dan keuangan organisasi. Implementasi teknologi *blockchain* dapat mengurangi risiko-risiko yang dihadapi sistem *email* saat ini, seperti tidak adanya autentikasi, tidak adanya enkripsi, arsitektur terpusat, dan akuntabilitas yang terbatas. Dengan memanfaatkan teknologi *blockchain*, *email* dapat ditandatangani dengan kunci privat pengirim, dienkripsi dengan kunci publik penerima, dan disimpan dalam *blockchain* yang terdistribusi dalam bentuk yang tidak dapat diubah (*immutable*). Meskipun begitu, ada beberapa kelemahan dalam penerapan blockchain untuk *email*, seperti kebutuhan waktu yang lama untuk adopsi dan masalah skalabilitas yang dapat diatasi dengan solusi untuk menggabungkan arsitektur *email* saat ini dengan *blockchain* untuk meningkatkan keamanan.

B. *Saran*

Penelitian pada bidang ini dapat dikembangkan lebih lanjut dengan merancang dan mengembangkan sistem *email* yang menggunakan *blockchain* hanya untuk aspek keamanannya dan kompatibel dengan sistem *email* saat ini. Hal ini memungkinkan integrasi teknologi *blockchain* ke dalam infrastruktur *email* yang ada tanpa mengganggu operasional sehari-hari, serta memberikan tingkat keamanan tambahan tanpa memerlukan perubahan besar dalam cara pengguna berinteraksi dengan *email* mereka. Penggunaan *blockchain* secara selektif untuk komponen keamanan dapat membantu menciptakan solusi yang efisien dan efektif dalam meningkatkan perlindungan terhadap ancaman siber.

Video Link at Youtube

Berikut adalah tautan video penjelasan makalah. https://youtu.be/O8SdROX4J_Y.

Source Code

Berikut adalah tautan *source code* implementasi *blockchain* pada *email*. https://github.com/samuel-eric/blockchain-mail.

Referensi

[1] Statista, "Daily Number of e-mails Worldwide 2023 | Statista," Statista, 2017. https://www.statista.com/statistics/456500/daily-number-of-e-mails-worldwide/.

[2] S. Choudhari and S. Das, "Spam E-mail Identification Using Blockchain Technology," IEEE Xplore, Jun. 01, 2021. https://ieeexplore.ieee.org/document/9485018 (accessed Apr. 19, 2022).

[3] "Share of financial phishing attacks worldwide 2021," Statista. https://www.statista.com/statistics/1319867/share-of-financial-phishing-attacks/.

[4] A. Petrosyan, "Spam e-mail traffic share 2019," Statista, Mar. 07, 2024. https://www.statista.com/statistics/420400/spam-email-traffic-share-annual/.

[5] Cisco, "What Is Phishing? Phishing Attack Examples and Definition," Cisco, Nov. 2019. https://www.cisco.com/c/en/us/products/security/email-security/what-is-phishing.html.

[6] M. Rosenthal, "Must-Know Phishing Statistics: Updated 2020," Tessian, Jan. 12, 2022. https://www.tessian.com/blog/phishing-statistics-2020/.

[7] R. Munir, "Pengantar Kriptografi STI." https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/01-Pengantar-Kriptografi-(2024).pdf.

[8] G. S. Chhabra and D. S. Bajwa, "Review of e-mail system, security protocols and email forensics," International Journal of Computer Science & Communication Networks, vol. 5, no. 3, pp. 201-211, Jan. 2015 .

[9] "Electronic Mail - Studytonight," www.studytonight.com. https://www.studytonight.com/computer-networks/electronic-mail

[10] R. Munir, "Penggunaan Kriptografi di dalam Blockchain." https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2022-2023/21-Penggunaan-kriptografi-di-dalam-blockchain.pdf.

[11] R. Munir, "Algoritma RSA" https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/11-Algoritma-RSA-2024.pdf.

[12] R. Munir, "Tanda-tangan Digital" https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/18-Tanda-tangan-digital-2024.pdf.

[13] R. Munir, "Advanced Encryption Standard (AES)" https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi-dan-Koding/2023-2024/08-AES-2024.

[14] J. C. González, V. García-Díaz, E. R. Núñez-Valdez, A. G. Gómez, and R. G. Crespo, "Replacing email protocols with blockchain-based smart contracts," Cluster Computing, May 2020, doi: https://doi.org/10.1007/s10586-020-03128-9.

[15] J. N. Al-Karaki, Amjad Gawanmeh, and C. Fachkha, "Blockchain for Email Security: A Perspective on Existing and Potential Solutions for Phishing Attacks," Oct. 2023, doi: https://doi.org/10.1109/bcca58897.2023.10338865.

PERNYATAAN

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 12 Juni 2024

Samuel Eric Yonatan (18221133)