Implementasi Kriptografi *Hybrid* dalam Pengelolaan Properti Rahasia pada Aplikasi Spring Boot

Christopher Justine William – NIM 13519006
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): 13519006@std.stei.itb.ac.id

***Abstract*—Kriptografi *hybrid* merupakan pendekatan yang menggabungkan algoritma kriptografi simetris dan asimetris untuk meningkatkan keamanan data. Makalah ini membahas tentang implementasi kriptografi *hybrid* dalam pengelolaan properti rahasia pada aplikasi Spring Boot. Makalah ini menjelaskan penggunaan algoritma kriptografi RSA yang digunakan untuk mengenkripsi kunci simetri AES yang digunakan dalam proses enkripsi dan dekripsi properti rahasia.**

Keywords—kriptografi hybrid, properti rahasia, RSA, AES

I. Pendahuluan

Kriptografi memiliki peranan penting dalam proses pengembangan aplikasi untuk menjaga kerahasiaan dan keamanan data. Salah satu pemanfaatan kriptografi dalam proses pengembangan aplikasi adalah untuk mengamankan properti aplikasi yang bersifat rahasia seperti *credential database* dan API *key*. Properti tersebut bersifat rahasia karena hanya pihak yang berwenang saja yang dapat membacanya dan tidak dapat disebarluaskan ke semua pihak. Dengan adanya kriptografi, properti yang bersifat rahasia tersebut dapat terenkripsi sehingga isi dari properti tersebut hanya dapat diketahui oleh pihak yang berwenang yaitu pemilik kunci enkripsi kriptografi yang digunakan.

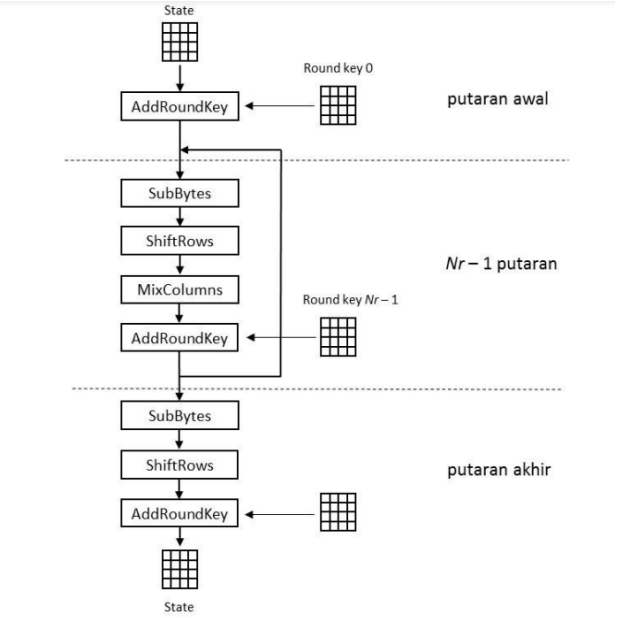
Spring Boot merupakan sebuah *framework* pada bahasa pemrograman Java yang umum digunakan untuk membangun aplikasi *server*. Pengelolaan properti rahasia pada aplikasi Spring Boot merupakan hal yang krusial. Properti pada aplikasi Spring Boot umumnya berada pada *file* konfigurasi application.properties yang dapat dikenali dan dibaca oleh aplikasi. Beberapa properti yang bersifat rahasia seperti *credential* perlu dikelola agar hanya pihak yang berwenang saja yang dapat membacanya. Salah satu pendekatan yang cukup efektif untuk mengamankan properti rahasia tersebut adalah dengan menggunakan algoritma kriptografi *hybrid* yaitu dengan menggabungkan kriptografi simetris dengan asimetris.

Pada pendekatan kriptografi *hybrid* untuk mengamankan properti rahasia, terdapat suatu *secret key* yang dibangkitkan dengan algoritma kriptografi simetri seperti AES. *Secret key* tersebut digunakan untuk mengenkripsi properti rahasia sehingga properti yang diletakkan pada *file* konfigurasi merupakan properti yang sudah terenkripsi dan perlu untuk didekripsi dengan menggunakan *secret key* yang sama sewaktu membaca properti tersebut dari *file* konfigurasi. Untuk meningkatkan keamanan aplikasi, *secret key* kriptografi simetri sebaiknya diletakkan pada tempat penyimpanan yang berbeda dari aplikasi. Kriptografi asimetris atau kunci publik seperti RSA dapat digunakan untuk mengenkripsi *secret key*. Ketika mengambil *secret key* yang telah dienkripsi dengan kunci publik, *secret key* tersebut perlu untuk didekripsi pada aplikasi dengan menggunakan kunci privat. Mekanisme tersebut berperan untuk mengamankan jalur pertukaran *secret key* pada tempat penyimpanan yang terpisah dari aplikasi.

II. Landasan teori

A. Algoritma AES

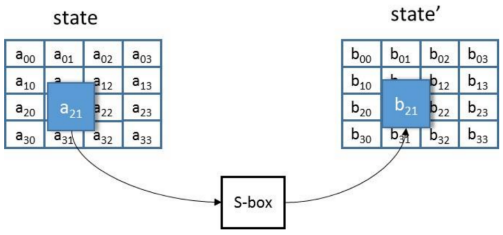
Algoritma AES merupakan salah satu algoritma kriptografi simetris yang banyak digunakan. Algoritma tersebut berasal dari perlombaan yang diadakan oleh *National Institute of Standards and Technology* (NIST) untuk memilih algoritma yang akan digunakan sebagai *Advanced Encryption Standard* (AES). Hasilnya algoritma *Rijndael* ditetapkan sebagai AES.



Gambar 1. Ilustrasi Putaran Blok Algoritma AES [1]

Algoritma AES termasuk ke dalam kelompok algoritma *block cipher* yang dengan panjang kuncinya yang fleksibel mulai dari 128 bit hingga 256 bit dan ukuran blok sebesar 128 bit. Untuk setiap blok AES, dilakukan *enchipering* sejumlah putaran dengan menggunakan kunci *internal* yang berbeda atau biasa disebut sebagai *round key*. Seperti yang bisa dilihat pada Gambar 1, terdapat beberapa operasi yang dilakukan pada setiap putaran, yaitu sebagai berikut:

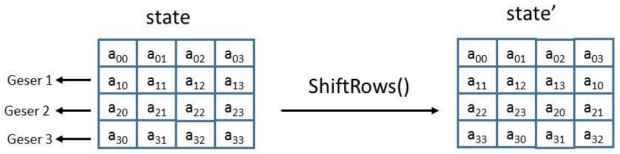
1. *SubBytes*



Gambar 2. Ilustrasi Operasi *SubBytes* [1]

Operasi yang menggantikan setiap *byte* dalam *state matrix* dengan *byte* baru berdasarkan S-box seperti yang dapat dilihat pada Gambar 2. S-box merupakan sebuah tabel substitusi non-linear yang digunakan untuk melakukan substitusi *byte*. Operasi ini bertujuan untuk memperkenalkan non-linearitas dalam enkripsi.

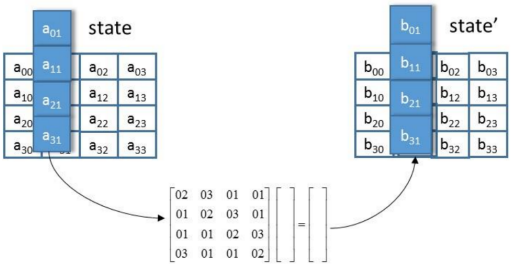
2. *ShiftRows*



Gambar 3. Ilustrasi Operasi *ShiftRows* [1]

Operasi yang menggeser baris-baris dalam *state matrix* seperti yang dapat dilihat pada Gambar 3. Setiap baris dalam *state matrix* digeser ke kiri sejumlah langkah tertentu. Baris pertama tidak berubah, baris kedua digeser satu langkah ke kiri, baris ketiga digeser dua langkah ke kiri, dan baris keempat digeser tiga langkah ke kiri. Operasi ini bertujuan untuk mencampur data antar baris dalam *state matrix* dan memperkenalkan difusi bit.

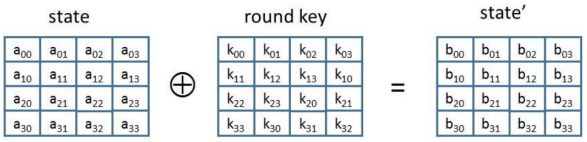
3. *MixColumns*



Gambar 4. Ilustrasi Operasi *MixColumns* [1]

Operasi yang melibatkan perkalian matriks pada setiap kolom dalam *state matrix* seperti yang dapat dilihat pada Gambar 4. Setiap kolom dalam *state matrix* dikalikan dengan menggunakan matriks perkalian tertentu. Operasi ini bertujuan untuk melakukan transformasi linier pada setiap kolom, mencampur data antar kolom, dan meningkatkan difusi bit.

4. *AddRoundKey*



Gambar 5. Ilustrasi Operasi *AddRoundKey* [1]

Operasi yang melibatkan XOR antara *state matrix* dengan kunci putaran seperti yang dapat dilihat pada Gambar 5. Kunci putaran diperoleh dari kunci enkripsi utama dengan menggunakan proses ekspansi kunci. Operasi ini bertujuan untuk meningkatkan difusi bit.

B. Algoritma RSA

Algoritma RSA (Rivest-Shamir-Adleman) merupakan salah satu algoritma kunci asimetris yang banyak digunakan. Algoritma tersebut ditemukan oleh peneliti dari MIT yaitu Ronald Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1976. Algoritma RSA digunakan dalam berbagai aplikasi, termasuk keamanan komunikasi, tanda tangan digital, dan pertukaran kunci rahasia.

Algoritma RSA menggunakan pasangan kunci publik dan kunci privat. Kunci publik digunakan untuk melakukan enkripsi sedangkan kunci privat digunakan untuk melakukan dekripsi. Kunci publik dapat dibagikan kepada semua pihak, sedangkan kunci privat harus tetap dijaga kerahasiaannya. Algoritma tersebut didasarkan pada kompleksitas dalam memfaktorkan bilangan-bilangan besar menjadi faktor-faktor primanya.

1. Pembangkitan Kunci

Pembangkitan kunci RSA dilakukan untuk menghasilkan pasangan kunci publik (e, n) dan kunci privat (d, n). Berikut langkah-langkah proses pembangkitan kunci RSA:

- Pilih dua bilangan prima acak, p dan q ($p \neq q$)
- Hitung modulus $n = p * q$
- Hitung nilai fungsi $\phi(n) = (p - 1) * (q - 1)$
- Pilih bilangan bulat e yang relatif prima dengan $\phi(n)$
- Hitung kunci privat d dengan persamaan berikut:

$$d \equiv e^{-1} (\text{mod}(\phi(n)))$$

2. Enkripsi

Proses enkripsi RSA melibatkan proses pengubahan *plaintext* m yang berukuran besar menjadi *plaintext* yang berukuran lebih kecil seperti m1, m2, m3. *Ciphertext* c dihasilkan dengan melakukan perpangkatan pada m dengan kunci publik berdasarkan persamaan berikut:

$$c_i \equiv m_i^e \bmod n$$

3. Dekripsi

Proses dekripsi algoritma RSA melibatkan pengubahan Kembali ciphertext c menjadi plaintext m berdasarkan persamaan berikut:

$$m_i \equiv c_i^d \bmod n$$

III. RANCANGAN SOLUSI

Rancangan solusi yang diajukan dalam makalah ini adalah pengimplementasian kriptografi *hybrid* dengan algoritma kunci simetris dan asimetris untuk mengamankan properti rahasia pada aplikasi Spring Boot. Algoritma kunci simetris yang digunakan adalah algoritma AES yang didasarkan pada kecepatannya yang sangat cepat dalam melakukan enkripsi dan dekripsi serta keamanannya yang masih terbukti. Algoritma kunci asimetris atau kunci publik yang digunakan adalah RSA yang dapat digunakan untuk mengenkripsi data berukuran pendek seperti kunci simetris.

Langkah pertama yang perlu dilakukan adalah membuat kunci simetris berdasarkan algoritma AES dan kunci publik-privat berdasarkan algoritma RSA. Selanjutnya properti rahasia dienkripsi dengan menggunakan kunci simetris dan diletakkan pada *file* konfigurasi application.properties. Kunci simetris diletakkan pada tempat penyimpanan yang berbeda dari aplikasi dengan mengenkripsinya terlebih dahulu dengan kunci publik. Saat aplikasi dijalankan, aplikasi akan membaca properti rahasia yang terenkripsi pada *file* konfigurasi. Aplikasi perlu untuk mendapatkan kunci simetris dengan mengambilnya dari tempat penyimpanan di luar aplikasi. Kunci simetris yang telah diambil masih terenkripsi dengan kunci publik dan perlu untuk didekripsi dengan kunci privat aplikasi (aplikasi perlu menyimpan kunci privat). Selanjutnya dekripsi properti rahasia dapat dilakukan dengan menggunakan kunci simetri yang telah didekripsi oleh kunci privat aplikasi.

IV. IMPLEMENTASI SOLUSI

Berdasarkan rancangan solusi, telah diimplementasikan aplikasi Spring Boot dengan *database* MySQL. Konfigurasi MySQL di kelas MySqlConfiguration akan membaca properti seperti URL, *username*, dan *password* dari file application.properties seperti yang dapat dilihat pada Gambar 6. Struktur folder aplikasi dapat dilihat pada Gambar 7.

```
@Value("jdbc:mysql://localhost:3306/mydb")
private String url;

@Value("67KmCppVoc9TtOOIIx9tqg==")
@EncryptedValue
private String username;

@Value("dyqUhR9p0ewmJw89P6pMxQ==")
@EncryptedValue
private String password;
```

Gambar 6. *Field* pada Kelas MySqlConfiguration

```
└── java
    ├── application
    │   ├── annotation
    │   │   └── EncryptedValue.java
    │   ├── component
    │   │   └── EncryptedValueBeanProcessor.java
    │   ├── configuration
    │   │   └── MySqlConfiguration.java
    │   ├── util
    │   │   └── EncryptionUtil.java
    │   └── Application.java
    └── resources
        └── application.properties
```

Gambar 7. Struktur Folder Aplikasi

C. *EncryptedValue.java*

kelas yang berfungsi sebagai anotasi. Anotasi ini digunakan untuk menandai properti yang nilainya dienkripsi pada *field* di dalam suatu kelas. Dengan menggunakan anotasi ini, properti rahasia seperti *username*, dan *password* dapat dibedakan untuk didekripsi ketika nilainya dibaca.

D. *EncryptedValueBeanProcessor.java*

Kelas yang berfungsi sebagai *processor* untuk komponen yang memiliki *field* yang ditandai dengan anotasi @EncryptedValue. Kelas ini akan mengambil nilai properti terenkripsi yang ditandai dan melakukan proses dekripsi untuk mendapatkan nilai properti yang sebenarnya.

E. *MySqlConfiguration.java*

Kelas yang berfungsi untuk melakukan konfigurasi *database* MySQL pada aplikasi Spring Boot. Di dalam kelas ini terdapat properti seperti URL, *username*, dan *password* yang ditandai dengan anotasi @EncryptedValue.

F. *EncryptionUtil.java*

Kelas utilitas yang berisi implementasi algoritma enkripsi dan dekripsi menggunakan RSA dan AES. Kelas ini menyediakan metode-metode untuk menghasilkan kunci, melakukan enkripsi dan dekripsi dengan RSA dan AES. Algoritma RSA dan AES diimplementasikan dengan menggunakan pustaka javax.crypto. AES yang digunakan adalah AES 256 bit.

G. *Application.java*

Kelas utama aplikasi Spring Boot untuk melakukan inisialisasi aplikasi. Terdapat fungsi main() di dalamnya.

H. *application.properties*

File yang berisi konfigurasi seperti pengaturan *database*, konfigurasi log, dan lain-lain. Properti dalam *file* ini akan digunakan untuk konfigurasi MySqlConfiguration.java, properti rahasia seperti *username*, dan *password* akan dienkripsi menggunakan EncryptionUtil sebelum digunakan.

V. Eksperimen

A. *Eksperimen Program*

Eksperimen dimulai dengan pembuatan kunci simetri AES serta kunci publik dan kunci privat RSA. Berikut merupakan kunci yang dihasilkan,

Kunci Simetris
rncd9zF4TaC22NSDvrpiqFYsbEorncGcOPV09ozRFdQ=

Kunci Publik
Sun RSA public key, 2048 bits params: null modulus: 22045263338847062642330817806560002846428114632126752359761307079515762907044047844680951145768689126804299035089671589621736121422932095801685616007919454736219032799162259370758720944234383804924998384470442916845643218824326480836020889167160697878439876622598125799117879896969697702590529284170699676910146523977933502467810583215635962190478179063556097824702758282789744044962807437109538017336133122048365639034018156707020453917405204637778599974742943920955389070849640391809505028343280420703621836189997655755364911550949212140170408257065979293928287990765656642148965772598571479075434533210358009453741
public exponent: 65537

Kunci Privat
SunRsaSign RSA private CRT key, 2048 bits params: null modulus: 22045263338847062642330817806560002846428114632126752359761307079515762907044047844680951145768689126804299035089671589621736121422932095801685616007919454736219032799162259370758720944234383804924998384470442916845643218824326480836020889167160697878439876622598125799117879896969697702590529284170699676910146523977933502467810583215635962190478179063556097824702758282789744044962807437109538017336133122048365639034018156707020453917405204637778599974742943920955389070849640391809505028343280420703621836189997655755364911550949212140170408257065979293928287990765656642148965772598571479075434533210358009453741
private exponent:
84585835775004690053565275274083011364005274965204283815029346448800429385008411599912598610174322410641454985178583363997158324198704621973722706278154723102516715560329916556616689330258909495192640703098060269313600619582299111509922834276427500320620281299396138264586143383005022877052878431993593554110665422356943570676573762383390422054393422131779875005966067873881375001319575857613691162984891187212293115251651791726257051842958314995906986463993047019569517921341362476793079495196476600023813641903632501875685193822690961801772293384043354023503320157621974200081471773474765870194096469453557944547 3

Kunci simetris yang telah dihasilkan digunakan untuk mengenkripsi properti konfigurasi MySQL yang berada di *file* application.properties. Berikut merupakan hasil enkripsi properti rahasia dengan kunci simetris,

Before	After
# MySQL Configuration mysql.url=jdbc:mysql://localhost:3306/mydb mysql.username=root mysql.password=password	# MySQL Configuration mysql.url=jdbc:mysql://localhost:3306/mydb mysql.username=BVwxlijtW+nhmN09TzXc/A== mysql.password=IFQ4BzIAjv1Vgle+koBNSg==

Kunci simetris kemudian akan dienkripsi dengan kunci publik algoritma kunci asimetris. Berikut merupakan kunci simetri terenkripsi,

Kunci Simetris Terenkripsi
JaTIQYU8Lz74THMCuRMbU5ggaj70h17JGO0vkawroJBcqsRn38FnBkimqSPqqErQNbmiiaNgd+LZJS32AYX+1/MZ6lgGiq2f0Q+SYghoG9EmZjMdBT2gtnRRFtkAs5z/kPJ8lgXW0nOMKE+nBCgluMrKyxUiFuORiOx1um34cZf1/Q7aELy8RMtLmmDpLs0OE8ZMvl0yFyC73Pn63SD9Di5CSS0/e6ycWrOiXcogS/htRrCaxOb7posTgEVBWy6LWBLjDB0DhuV36Y54X8+o8enQdp1I4miatOb54bvWCtSn9sAl9YY6WPRInTuUQO0ORBgDU87XpMsR0BV/cSBTsA==

Ketika aplikasi dijalankan, aplikasi akan mengambil kembali kunci simetris terenkripsi, mendekripsi kunci simetris tersebut, lalu menggunakan kunci simetri untuk mendekripsi properti rahasia yang dienkripsi dengan kunci simetri. Ketika properti rahasia tersebut sudah didekripsi, properti tersebut dapat digunakan untuk melakukan konfigurasi. Untuk

memudahkan pengujian kunci simetris terenkripsi diletakkan pada application.properties, tetapi untuk meningkatkan keamanan sebaiknya diletakkan di tempat penyimpanan terpisah yang pada makalah ini tidak diimplementasikan. Kunci privat tidak seharusnya diletakkan pada *file* konfigurasi yang dapat diketahui publik. Kunci private dapat ditambahkan oleh pihak yang berwenang hanya ketika menjalankan aplikasi. Berikut merupakan properti lengkap saat aplikasi dijalankan,

application.properties

```
# Server Configuration
server.port=8090

# Application Configuration
application.private_key=private_key.key
application.encrypted_key=JaTIQYU8Lz74THMCuRMbU
5ggaj70h17JGO0vkawroJBcqsRn38FnBkimqSPqqErQNb
miiaNgd+LZJS32AYX+1/MZ6lgGiq2f0Q+SYghoG9EmZj
MdBT2gtnRRFtkAs5z/kPJ8lgXW0nOMKE+nBCgluMrKy
xUiFuORiOx1um34cZf1/Q7aELy8RMtLmmDpLs0OE8Z
Mvl0yFyC73Pn63SD9Di5CSS0/e6ycWrOiXcogS/htRrCax
Ob7posTgEVBWy6LWBLjDB0DhuV36Y54X8+o8enQdp1
I4miatOb54bvWCtSn9sAl9YY6WPRInTuUQO0ORBgDU
87XpMsR0BV/cSBTsA==

# MySQL Configuration
mysql.url=jdbc:mysql://localhost:3306/mydb
mysql.username=BVwxlijtW+nhmN09TzXc/A==
mysql.password=IFQ4BzIAjv1Vgle+koBNSg==

# Logging Configuration
logging.level.org.springframework=INFO

# Spring Data JPA Configuration
spring.jpa.show-sql=true
spring.jpa.hibernate.ddl-auto=update
spring.jpa.properties.hibernate.dialect=org.hibernate.dialect.
MySQL8Dialect
```

Jika kunci simetris hasil dekripsi dari kunci publik asimetris dapat mendekripsi properti rahasia, aplikasi dapat berjalan dengan semestinya tanpa mengeluarkan pesan kesalahan seperti yang dapat dilihat pada Gambar 8. Jika terdapat kesalahan dalam proses dekripsi, pesan kesalahan akan muncul seperti yang dapat dilihat pada Gambar 9.

B. Analisis Keamanan

Keamanan algoritma RSA didasarkan pada sulitnya memfaktorkan bilangan bulat yang besar menjadi faktor-faktor prima. Keamanan algoritma RSA sangat bergantung pada panjang kunci yang digunakan. Pada makalah ini panjang kunci algoritma RSA yang digunakan adalah sebesar 2048 bit sehingga dapat dikatakan cukup aman dari serangan faktorisasi karena saat ini belum ditemukan algoritma pemfaktoran bilangan yang sangat besar dengan waktu polinomial.

Gambar 8. Tampilan Sukses Menjalankan Aplikasi

Gambar 9. Tampilan Gagal Menjalankan Aplikasi

Algoritma AES yang digunakan dalam makalah ini menggunakan panjang kunci sebesar 256. Untuk memecahkan kunci tersebut secara *brute force* dibutuhkan jumlah kemungkinan sebesar 2^256 kemungkinan. Algoritma tersebut dapat dikatakan aman karena jumlah kombinasi kunci yang memungkinkan sangat besar dan butuh waktu yang lama untuk menyelesaikannya dengan kemampuan komputasi saat ini.

Secara keseluruhan keamanan kriptografi *hybrid* didasarkan pada keamanan masing-masing algoritma yang digunakan. Algoritma RSA dapat dengan baik melakukan enkripsi kunci simetri dalam proses pertukaran kunci simetri yang digunakan untuk enkripsi dan dekripsi properti rahasia. Adapun kelemahan dari implementasi solusi dalam makalah ini terletak pada manajemen kunci privat aplikasi. Jika pihak luar dapat mengetahui kunci privat aplikasi, pihak tersebut dapat mendekripsi kunci simetri untuk kemudian mendapatkan nilai dari properti rahasia yang dienkripsi.

VI. Kesimpulan dan Saran

Algoritma kriptografi *hybrid* mengkombinasikan algoritma kunci publik dengan algoritma kunci simetris. kunci publik digunakan untuk mengkenkripsi kunci simetri sehingga kunci simetri tersebut dapat dipertukarkan melalui suatu saluran komunikasi jarak jauh. Algoritma kriptografi *hybrid* dapat digunakan untuk mengamankan properti rahasia aplikasi.

Pada makalah ini telah diimplementasikan program aplikasi server Spring Boot yang memerlukan *credential* untuk mengakses *database* MySQL. Aplikasi tersebut memanfaatkan algoritma kunci simetris AES untuk mengenkripsi properti rahasia dan algoritma kunci publik RSA untuk mengenkripsi kunci simetris sehingga dapat meningkatkan keamanan properti rahasia aplikasi. Untuk saran implementasi lebih lanjut, kunci simetris sebaiknya diletakkan pada tempat penyimpanan yang berbeda dengan aplikasi sehingga dapat memanfaatkan kekuatan penuh dari algoritma kunci publik dalam melakukan pertukaran kunci. Selain itu diperlukan juga manajemen kunci privat aplikasi sehingga hanya pihak yang berwenang saja yang dapat mengetahuinya.

Pranala Kode Program

https://github.com/cjustinw/makalah-kripto

Ucapan Terima Kasih

Penulis mengucapkan puji syukur kepada Tuhan Yang Maha Esa atas berkat dan rahmatnya sehingga penulis dapat menyelesaikan makalah ini dengan baik. Penulis juga mengucapkan terima kasih kepada Bapak Dr. Ir. Rinaldi, M.T. selaku pengampu mata kuliah IF4020 Kriptografi atas ajaran serta ilmu yang telah disampaikan. Selain itu, penulis juga mengucapkan terima kasih kepada semua pihak lain yang telah mendukung penulis dalam penulisan makalah ini.

References

[1] Munir. Rinaldi 2023. Review Beberapa Block Cipher (Bagian 3: Advanced Encryption Standrad (AES). https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/17-Beberapa-block-cipher-bagian3-2023.pdf diakses pada 20 Mei 2023.

[2] Munir. Rinaldi 2023. Algoritma RSA. https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2022-2023/19-Algoritma-RSA-2023.pdf diakses pada 20 Mei 2023.

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, 22 Mei 2023

Christopher Justine William