BAB 3

METODOLOGI PENELITIAN

3.1 Metodologi Penelitian

Terdapat lima tahap pada metodologi penelitian yang digunakan pada penelitian ini, yaitu tahap telaah literatur, perancangan dan implementasi, pengujian, evaluasi, dan dokumentasi.

1. Telaah Literatur

Pada tahap ini, peneliti mengumpulkan dan mempelajari sejumlah literatur yang berhubungan dengan permasalahan yang diangkat pada penelitian ini. Literatur yang digunakan dalam penelitian ini adalah algoritma *post-quantum cryptography* CRYSTALS-Kyber yang akan digunakan sebagai *public-key encryption* dan aplikasi autentikasi produk Oricon yaitu aplikasi tempat diterapkannya algoritma *post-quantum cryptography* CRYSTALS-Kyber.

2. Perancangan dan Implementasi

Perancangan program implementasi algoritma *post-quantum cryptography* CRYSTALS-Kyber akan dijelaskan lebih lanjut menggunakan *flowchart*. Program akan dibuat menggunakan bahasa pemrograman Javascript dengan menggunakan Node.js SDK lalu akan diimplementasikan pada *smart contract* Hyperledger Fabric SDK Node.js 1.4.0.

3. Pengujian

Metode yang digunakan dalam pengujian program adalah *blackbox testing*. *Blackbox testing* dilakukan dengan cara memanggil API melalui *software* Insomnia. Tujuan dari pengujian ini untuk memastikan sistem yang dibuat telah berjalan dengan baik dan sesuai dengan keinginan.

4. Evaluasi

Pada tahap ini akan dilakukan evaluasi performa dari implementasi algoritma *post-quantum cryptography* CRYSTALS-Kyber pada aplikasi autentikasi produk Oricon. Evaluasi akan dilakukan menggunakan Hyperledger Caliper untuk mengukur *throughput* dan *latency*. Selain itu akan digunakan *library* performance-now untuk mengukur waktu algoritma *hybrid encryption*. Data uji yang digunakan dalam proses evaluasi terdiri dari empat ukuran dalam satuan *kilo bytes* (KB), yaitu 25 KB, 50 KB, 75 KB, dan 100 KB. Masing-masing data uji ini akan dienkripsi menggunakan tiga *key size* berbeda, yaitu *key size* 512, *key size* 768, dan *key size* 1024. Hasil dari evaluasi yang telah dilakukan akan digunakan untuk mengetahui *key size* mana yang paling optimal untuk digunakan dalam aplikasi autentikasi produk Oricon.

5. Dokumentasi

Tahap ini dilakukan untuk mendokumentasikan seluruh pengerjaan yang telah dilakukan pada penelitian ini mulai dari telaah literatur, pembuatan program, hingga pemberian kesimpulan dan saran untuk penelitian berikutnya.

3.2 Perancangan Sistem

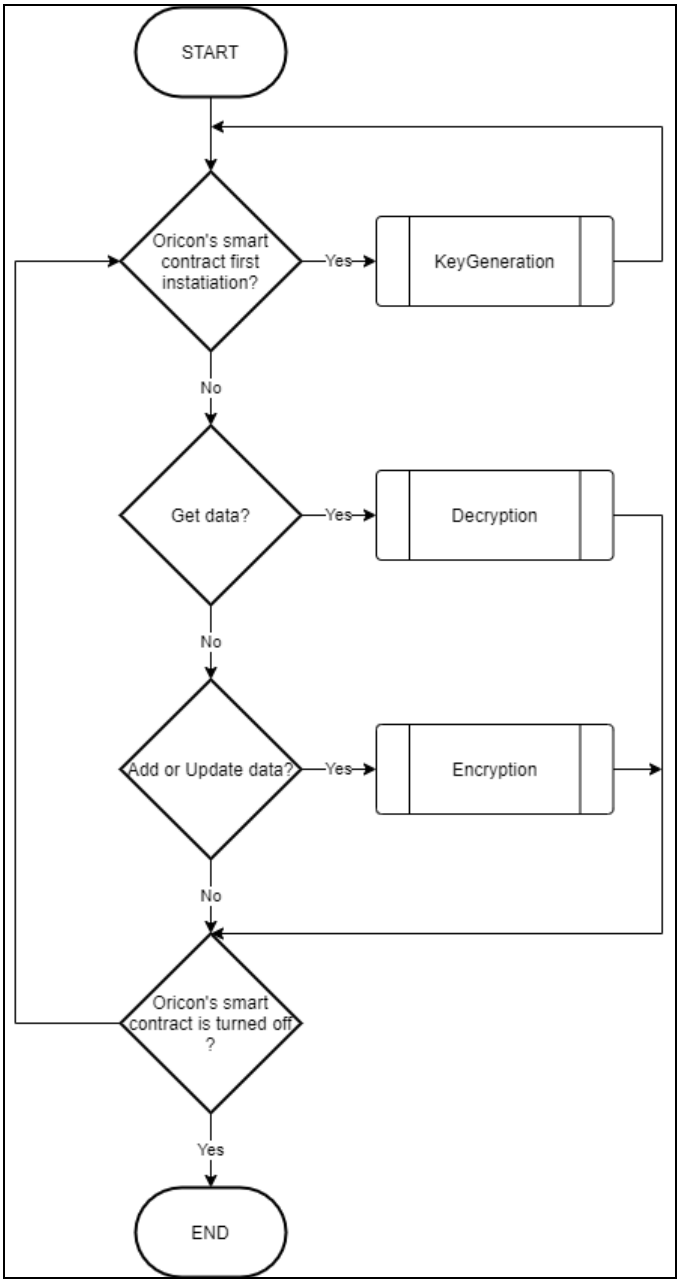
Perancangan sistem pada penelitian ini digambarkan dengan *flowchart* dan rancangan antarmuka *website*. Rancangan antarmuka *website* dibuat dengan tujuan untuk mempermudah *user* dalam mengganti *key size* CRYSTALS-Kyber yang akan digunakan dalam proses enkripsi maupun dekripsi.

3.2.1 Flowchart Perancangan Sistem

Flowchart perancangan sistem digunakan untuk menggambarkan perancangan sistem yang telah dibuat. Terdapat lima buah *flowchart* yang digunakan yaitu *flowchart* utama sistem, *flowchart key generation*, *flowchart* enkripsi data transaksi, *flowchart* dekripsi data transaksi, *flowchart* enkapsulasi *key*, dan *flowchart* dekapsulasi *key*.

A Flowchart Utama Sistem

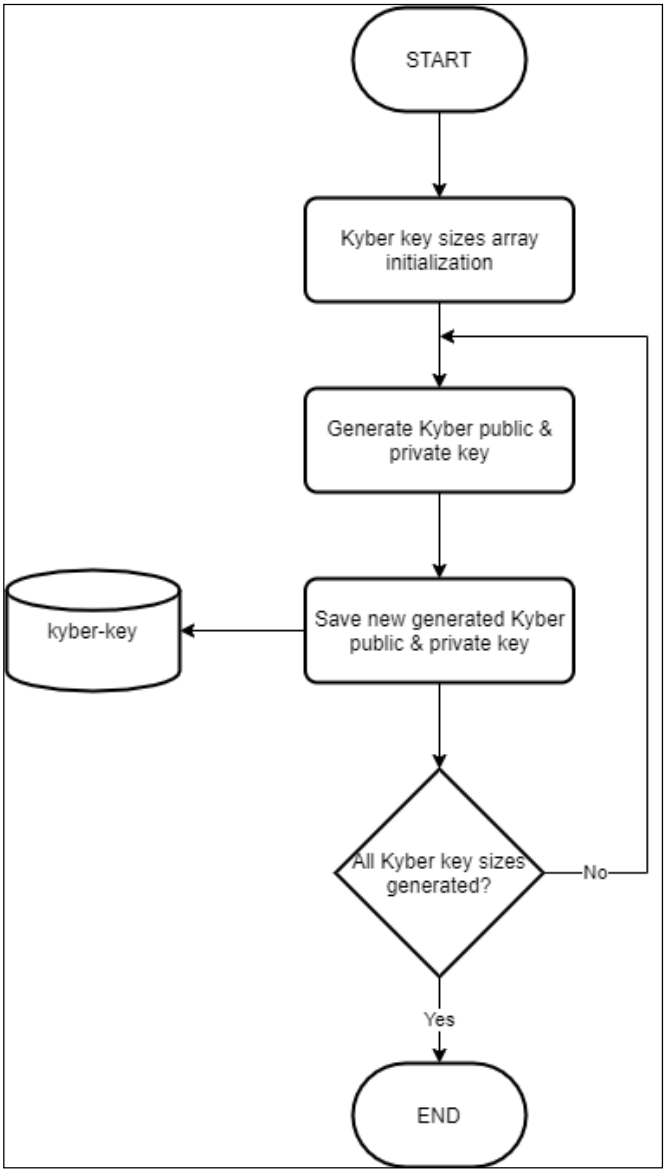
Fungsi KeyGeneration akan dijalankan hanya sekali saat *smart contract* pertama kali dijalankan. Setelah fungsi KeyGeneration dijalankan, *client* dapat menggunakan aplikasi Oricon untuk melakukan proses transaksi seperti mengambil, menambahkan, maupun mengubah data. Fungsi Decryption akan terpanggil saat *client* hendak mengambil sebuah data transaksi. Fungsi Encryption akan terpanggil saat *client* hendak menambahkan maupun mengubah suatu data. Gambar 3.1 di bawah ini merupakan *flowchart* Utama yang menunjukan alur sistem secara garis besar.



Gambar 3.1 *Flowchart* Utama Sistem

B Flowchart Key Generation

Key generation merupakan tahap awal yang perlu dilakukan sebelum dapat melakukan enkripsi dan dekripsi. Gambar 3.2 di bawah ini merupakan *flowchart* dari *key generation*.

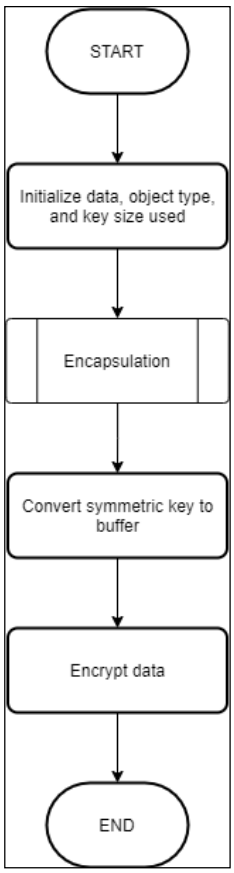


Gambar 3.2 *Flowchart* Key Generation

Key generation diawali dengan inisialisasi sebuah *array* satu dimensi yang menampung semua *key sizes* dari algoritma CRYSTALS-Kyber. Terdapat tiga buah *key sizes*, yaitu *key size* 512, *key size* 768, dan *key size* 1024. Setelah inisialisasi *array* dari *key sizes*, akan dilakukan pembuatan *public key* dan *private key* untuk setiap *key sizes* yang ada dalam *array*. *Public key* dan *private key* yang telah dibuat akan disimpan pada Firestore di *collection* kyber-key. Proses pembuatan *key* akan terus dilakukan hingga semua *key sizes* selesai dibuat.

C Flowchart Enkripsi Data Transaksi

Enkripsi dilakukan untuk mengubah data transaksi dari *plaintext* menjadi *ciphertext*. Enkripsi bertujuan untuk menjamin data transaksi yang disimpan di *database* tidak dapat dilihat maupun diubah isinya oleh pihak tidak berwenang. Gambar 3.3 di bawah ini merupakan *flowchart* dari enkripsi.



Gambar 3.3 *Flowchart* Enkripsi Data Transaksi

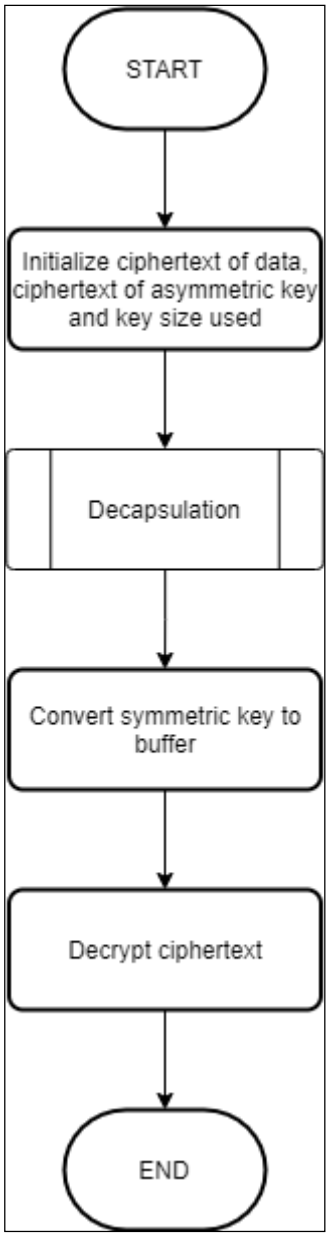
Enkripsi diawali dengan inisialisasi data transaksi yang ingin dienkripsi, tipe data transaksi, dan *key size* yang akan digunakan untuk mengekripsi data transaksi tersebut. Terdapat dua tipe data transaksi yaitu token dan produk. Terdapat tiga buah *key sizes* yang dapat digunakan yaitu *key size* 512, *key size* 768, dan *key size* 1024. Hanya satu *key size* yang dapat digunakan untuk melakukan enkripsi data transaksi dalam satu waktu. Selanjutnya akan dilakukan pemanggilan fungsi enkapsulasi. Keluaran yang dihasilkan dari fungsi enkapsulasi yaitu sebuah *symmetric key* dan *ciphertext* dari *symmetric key* tersebut. Fungsi enkapsulasi menghasilkan *symmetric key* dalam bentuk *array of integer* sehingga harus diubah ke bentuk *buffer* agar dapat digunakan dalam algoritma *symmetric encryption* AES-256. Saat *symmetric key* telah menjadi bentuk *buffer*, data transaksi akan dienkripsi menggunakan *symmetric key* tersebut dengan algoritma *symmetric encryption* AES-256.

D Flowchart Dekripsi Data Transaksi

Dekripsi dilakukan untuk mengubah data dari *ciphertext* menjadi *plaintext*. *Ciphertext* merupakan data transaksi dengan huruf *random* yang sulit untuk dibaca dan dipahami. Oleh karena itu, dekripsi bertujuan untuk mengubah data transaksi dalam bentuk *ciphertext* yang sulit dibaca tersebut menjadi data transaksi dalam bentuk *plaintext* yang dapat dibaca dan dipahami. Gambar 3.4 di bawah ini merupakan *flowchart* dari dekripsi data transaksi.

Dekripsi diawali dengan inisialisasi *ciphertext* dari data transaksi yang ingin didekripsi, *ciphertext* dari *symmetric key*, dan *key size* yang digunakan untuk mendekripsi data transaksi tersebut. Lalu akan dilakukan pemanggilan fungsi dekapsulasi untuk mendapatkan *symmetric key* dari *ciphertext*. *Symmetric key*

yang didapat akan diubah ke bentuk *buffer* lalu digunakan untuk mendekripsi *ciphertext* dari data transaksi menggunakan algoritma *symmetric encryption* AES-256.

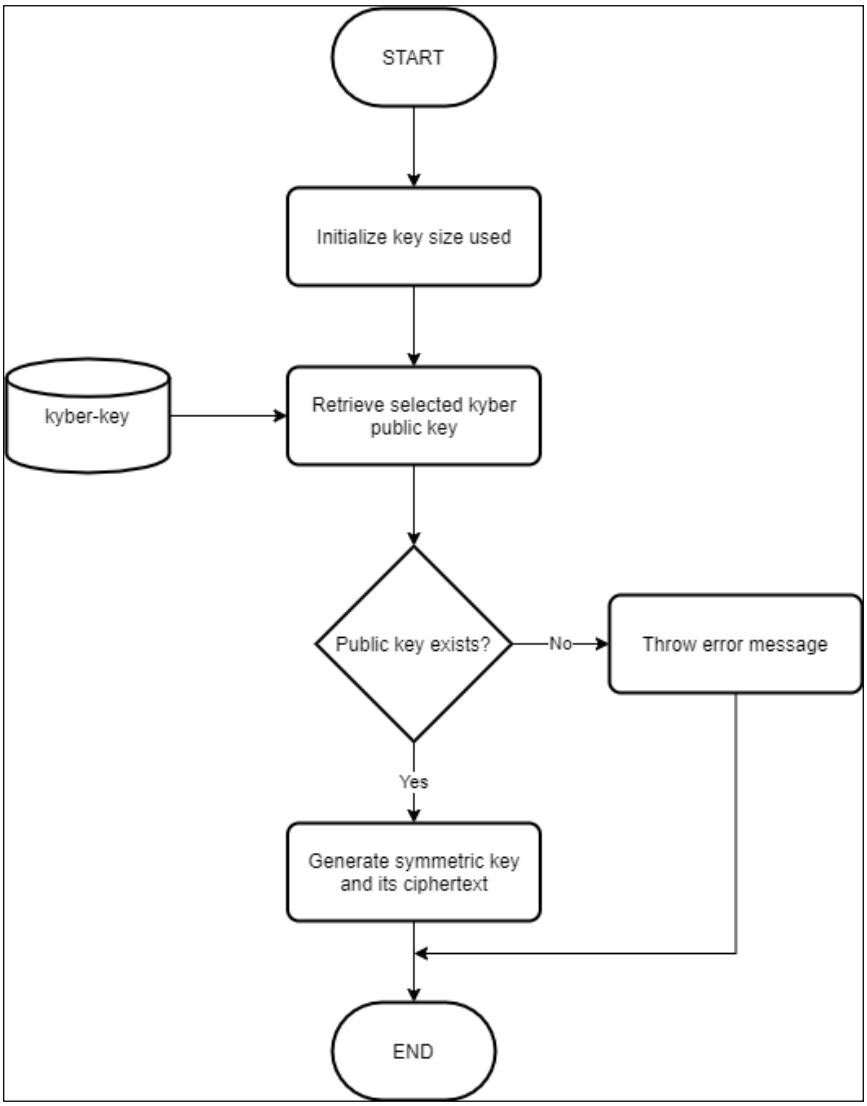


Gambar 3.4 *Flowchart* Dekripsi Data Transaksi

E Flowchart Enkapsulasi Key

Enkapsulasi dilakukan untuk membungkus *symmetric key* agar tidak dapat dilihat dan digunakan oleh pihak yang tidak berwenang. Proses pembungkusan

key ini dilakukan dengan menggunakan algoritma *asymmetric encryption*. Gambar 3.5 di bawah ini merupakan *flowchart* dari enkapsulasi *key*.

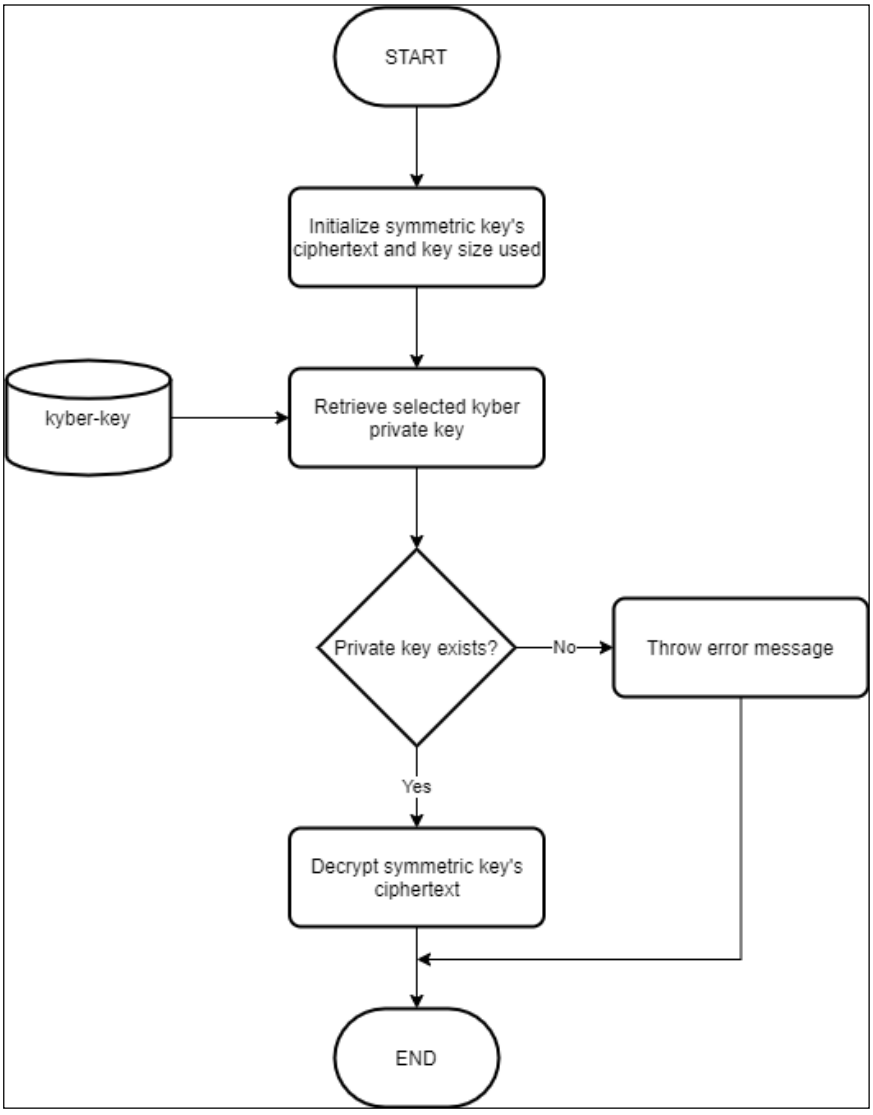


Gambar 3.5 *Flowchart* Enkapsulasi Key

Enkapsulasi diawali dengan inisialisasi *key size* yang akan digunakan dalam enkapsulasi. Selanjutnya *public key user* akan diambil pada Firestore dari *collection* kyber-key. *Public key* yang diambil menyesuaikan dengan *key size* yang dinisialisasi diawal. Dengan menggunakan *public key user*, akan dilakukan pembuatan *symmetric key* secara *random* dan *ciphertext* dari *symmetric key* tersebut.

F Flowchart Dekapsulasi Key

Dekapsulasi dilakukan untuk mengubah *ciphertext* dari *symmetric key* ke bentuk *plaintext*-nya agar dapat digunakan pada proses dekripsi data transaksi. Gambar 3.6 di bawah ini merupakan *flowchart* dari dekapsulasi *key*.



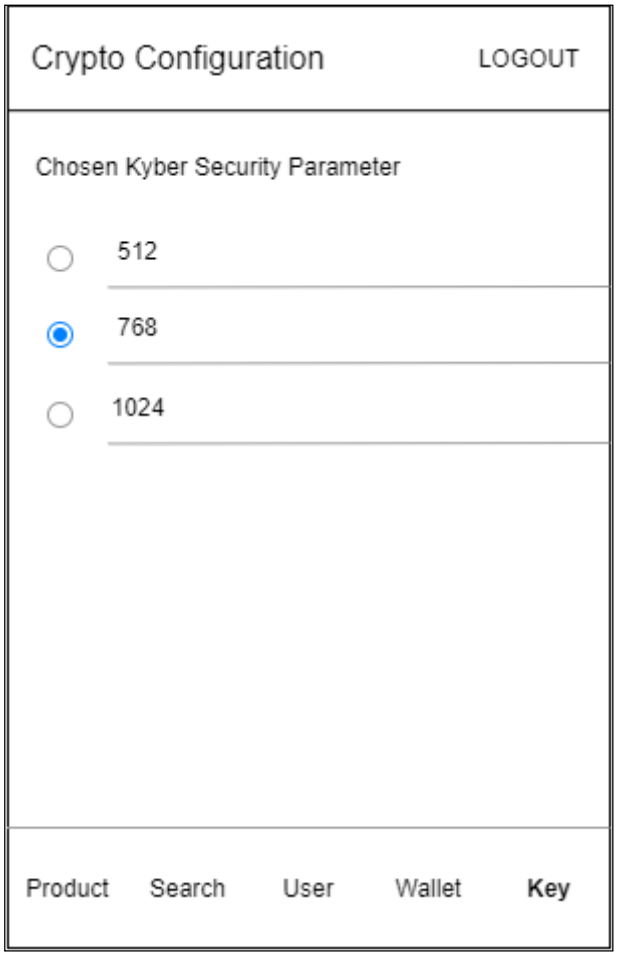
Gambar 3.6 *Flowchart* Dekapsulasi Key

Dekapsulasi diawali dengan inisialisasi *ciphertext* dari *symmetric key* yang hendak didekapsulasi beserta *key size* yang digunakan dalam dekapsulasi. Selanjutnya akan dilakukan pengambilan *private key user* pada Firestore dari *collection* kyber-key. *Private key* yang diambil berdasarkan *key size* yang

diiinisialisasi diawal pemanggilan fungsi dekapsulasi *key*. Lalu akan dilakukan dekapsulasi menggunakan *private key user* tersebut untuk mendapatkan *symmetric key* dari *ciphertext*.

3.2.2 Rancangan Antarmuka Halaman

Sebuah submenu dibuat pada aplikasi autentikasi produk Oricon. Gambar 3.7 di bawah ini merupakan rancangan antarmuka *website* submenu *key*.



Gambar 3.7 Rancangan antarmuka Submenu Key

Submenu *key* dapat digunakan *user* untuk mengubah *key size* dari algoritma *post-quantum cryptography* CRYSTALS-Kyber. *Key size* ini digunakan untuk menentukan *key size* dari *public key* yang akan dipakai saat proses

enkapsulasi *symmetric key* menjadi *ciphertext*-nya dan *private key* yang akan dipakai saat proses dekapsulasi *ciphertext* menjadi *symmetric key*-nya.

Saat *user* membuka submenu *key*, *user* akan ditampilkan tiga buah *radio button* yang dapat digunakan untuk memilih *key size* yang akan digunakan. *Key size* yang telah dipilih oleh *user* akan disimpan dalam *local storage*. Saat *user* mengirimkan *request create product* dan *update product* ke *server*, *key size* ini akan diambil dari *local storage* dan diikutsertakan pada JSON *body*. *Key size* akan ditambahkan dalam parameter URL saat *user* mengirimkan *request read product*.

3.3 Perancangan Evaluasi Performa

Pada penelitian ini, terdapat dua buah evaluasi performa yang dilakukan yaitu evaluasi performa waktu kriptografi yang mengukur waktu *key generation*, enkripsi, dan dekripsi serta evaluasi performa *smart contract* dari segi transaksi sukses, transaksi gagal, *latency* dan *throughput*. Evaluasi performa *smart contract* dilakukan hanya pada dua eksperimen saja, yaitu eksperimen 7 dan eksperimen 8 yang menggunakan MaxMessageCount sebesar 20, BatchTimeout sebesar 2 detik, dan jumlah *workers* sebanyak 2 *workers* dan 4 *workers*. Kedua eksperimen ini akan menggunakan *smart contract* yang telah diimplementasikan ketiga *key size* CRYSTALS-Kyber di mana ketiga *key size* tersebut mengikuti ukuran yang telah ditetapkan pada CRYSTALS-Kyber, sehingga eksperimen yang dilakukan pada penelitian ini, yaitu.

Tabel 3.1 Tabel eksperimen Hyperledger Caliper

No	Key Size	MaxMessageCount	BatchTimeout (s)	Number of Workers
1	512	20	2	2
2		20	2	4
3	768	20	2	2

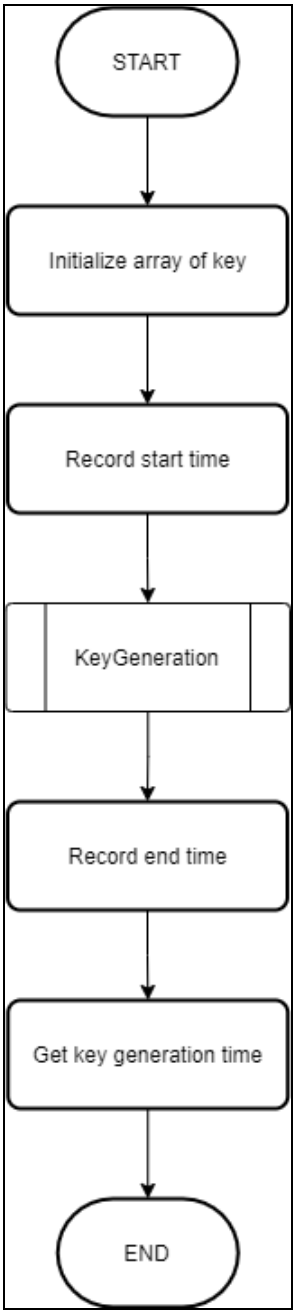
4		20	2	4
5	1024	20	2	2
6		20	2	4

3.3.1 Flowchart Perancangan Evaluasi Performa Waktu

Flowchart perancangan evaluasi performa waktu digunakan untuk menggambarkan perancangan evaluasi performa waktu yang telah dibuat. Terdapat tiga buah *flowchart* yang digunakan yaitu *flowchart testing key generation*, *flowchart testing* enkripsi, dan *flowchart testing* dekripsi. Evaluasi performa waktu CRYSTALS-Kyber menggunakan *library* Performance-Now untuk mengukur waktu *key generation*, enkripsi, dan dekripsi.

A Flowchart Testing Key Generation

Flowchart testing key generation yang digunakan untuk melakukan evaluasi performa waktu dari fungsi Key Generation. Evaluasi akan dilakukan dengan mencatat waktu sebelum dan sesudah fungsi Key Generation dipanggil. Waktu yang dibutuhkan untuk melakukan *key generation* diperoleh dengan cara mengurangi nilai waktu sesudah fungsi Key Generation dijalankan dengan nilai waktu sebelum fungsi Key Generation dijalankan. *Flowchart testing key generation* dapat dilihat pada Gambar 3.8 di bawah ini.

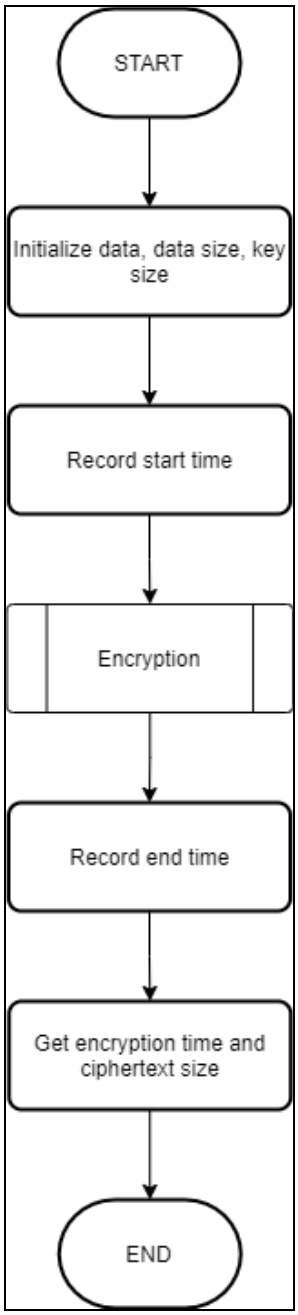


Gambar 3.8 *Flowchart testing key generation*

B Flowchart Testing Enkripsi

Flowchart testing enkripsi yang digunakan untuk melakukan evaluasi performa waktu dari fungsi Enkripsi. Sebelum fungsi Enkripsi dipanggil, akan dilakukan pencatatan waktu dengan bantuan *library* Performance-Now yang menghasilkan akurasi dalam milidetik yang akan dikonversi menjadi detik. Waktu

saat fungsi Enkripsi selesai dijalankan juga akan dicatat lalu dikurangi dengan waktu sebelum fungsi dipanggil. *Flowchart* dari *testing* enkripsi tersebut dapat dilihat pada Gambar 3.9 di bawah ini.

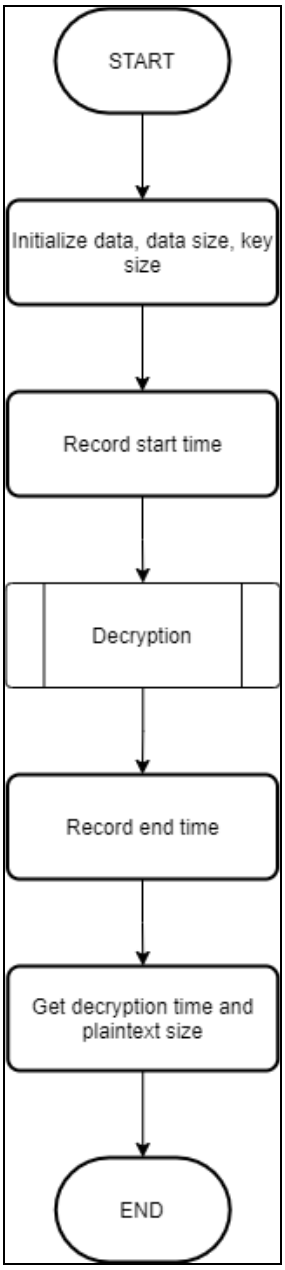


Gambar 3.9 *Flowchart testing* enkripsi

C Flowchart Testing Dekripsi

Gambar 3.10 di bawah ini merupakan *flowchart testing dekripsi* yang

menunjukkan alur evaluasi performa waktu untuk fungsi Dekripsi. *Library* Performance-Now akan digunakan untuk mencatat waktu mulai dan waktu selesai dari pemanggilan fungsi ekripsi. Hasil akhir evaluasi performa waktu didapat dengan cara mengurangi waktu selesai dengan waktu mulai pemanggilan fungsi Dekripsi.



Gambar 3.10 *Flowchart testing* dekripsi