Pengembangan Sistem *Secret Sharing* Terdistribusi dengan Layanan *Proactive Secret Sharing*

Fauzan Hilmi Ramadhian
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, ITB
Bandung, Indonesia
fauzanhilmir@gmail.com

Rinaldi Munir
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika, ITB
Bandung, Indonesia
rinaldi@informatika.org

***Abstrak*— *Secret sharing* adalah metode penyebaran data ke sekumpulan pihak sehingga data hanya dapat direkonstruksi oleh sebagian dari kumpulan pihak tersebut. Aplikasi-aplikasi *secret sharing* telah dikembangkan namun belum ada yang mampu menangani pengiriman *share* antar komputer dan menangani pembaruan *share* dengan *proactive secret sharing*. Tujuan penelitian ini adalah untuk membangun sistem *secret sharing* terdistribusi yang mendukung operasi penyebaran *share*, rekonstruksi *secret*, dan pembaruan *share* dengan *proactive secret sharing*. Dua poin penting penelitian adalah protokol operasi *secret sharing* dan arsitektur sistem. Protokol operasi *secret sharing* mencakup protokol operasi penyebaran *share*, rekonstruksi *secret*, dan pembaruan *share*. Arsitektur sistem terdiri dari tiga komponen utama yaitu aplikasi *client*, basis data, dan *message queue*. Aplikasi *client* diimplementasikan sebagai *desktop application* pada platform Windows, basis data diimplementasikan dengan MySQL, dan *message queue* diimplementasikan dengan RabbitMQ. Dilakukan proses pengujian aspek fungsionalitas dan aspek keamanan aplikasi setelah aplikasi selesai dikembangkan. Hasilnya, aplikasi mampu melewati seluruh skenario pengujian.**

Kata kunci— *secret sharing*, skema Shamir, pembaruan *share*, *proactive secret sharing*.

I. PENDAHULUAN

Secret sharing adalah sebuah metode penyebaran data ke sekumpulan pihak sehingga data asli hanya dapat dikonstruksi ulang oleh sejumlah bagian dari kumpulan pihak tersebut (Beimel, 2011). Pada *secret sharing*, pemilik data (disebut sebagai *dealer*) membagikan potongan-potongan data (disebut sebagai *share*) ke para pemegang *share* (disebut sebagai *participant*). Data asli (disebut sebagai *secret*) hanya dapat direkonstruksi apabila jumlah *share* melewati nilai ambang rekonstruksi. Nilai ambang rekonstruksi tidak harus sama dengan jumlah seluruh *share*. Dengan demikian, untuk melakukan rekonstruksi, seluruh *participant* tidak perlu menyerahkan *share*-nya.

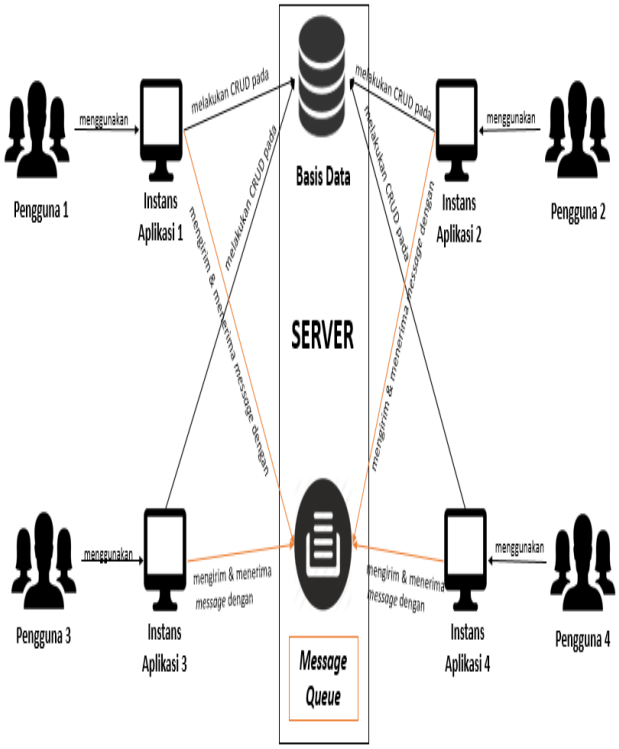
Skema *secret sharing* pertama yang ditemukan adalah Skema Shamir. Skema yang diusulkan oleh Shamir (1979) ini memanfaatkan konsep matematika bahwa polinomial berderajat *k*-1 dapat didefinisikan jika diketahui *k* titik yang dilewati polinomial tersebut. Skema ini dapat dimodifikasi untuk mengakomodasi pembaruan *share*. Pembaruan *share* adalah proses mengubah seluruh *share* tanpa mengubah data asli (*secret)* untuk meminimalisasi kemungkinan bocornya *secret* jika ada *share* yang dicuri. Ide pembaruan *share* pertama kali dicetuskan oleh Ostrovsky & Yung (1991). Pembaruan *share* dilakukan dengan melakukan konstruksi ulang *secret* dan pembagian ulang *share*. Lalu, Herzberg dkk., (1995) mengajukan ide pembaruan *share* yang lebih baik yang diberi nama *proactive secret sharing*. Pada *proactive secret sharing*, pembaruan *share* tidak memerlukan proses rekonstruksi *secret* sehingga keamanan *secret* lebih terjamin.

Skema Shamir telah digunakan oleh beberapa aplikasi penyedia layanan *secret sharing* seperti ssss (Poettering, 2005), *Secret* Sharp (Matthew, 2007), *ShareSecret* (Zargaryan, 2010), dan gf*share* (Silverstone, 2006). Sayangnya, tidak ada satupun aplikasi yang mendukung pembaruan *share* dengan *proactive secret sharing*. Selain itu, tidak ada aplikasi yang dapat menangani proses pengiriman *share*. Keempat aplikasi tersebut hanya bertugas untuk menangani proses pembangkitan *share* dan rekonstruksi *secret*. Pengguna keempat aplikasi tersebut harus mengirimkan *share* secara manual. Pengiriman secara manual berpotensi menimbulkan masalah bocornya *share* jika cara pengiriman yang dipilih tidak aman. Sebaiknya aplikasi menangani proses pengiriman *share* juga sehingga keamanan *share* dapat lebih terjamin. Aplikasi dapat menangani pengiriman *share* jika ia adalah aplikasi terdistribusi, yaitu aplikasi yang berjalan pada suatu jaringan sehingga setiap instansi aplikasi dapat saling berhubungan dengan instansi lain walaupun terletak pada komputer yang berbeda. Oleh karena itu, perlu dikembangkan aplikasi *secret sharing* terdistribusi yang mampu mendukung data berbentuk string karakter dan dokumen serta sanggup menyediakan layanan *proactive secret sharing*.

II. RANCANGAN SISTEM

A. Arsitektur Sistem

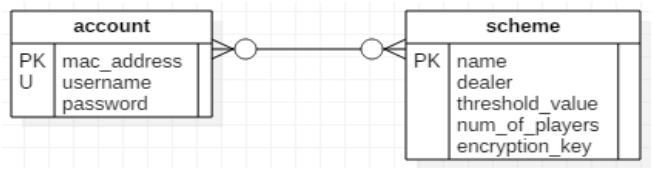
Rancangan arsitektur umum sistem diilustrasikan pada Gambar 1.



Gambar 1. Diagram Arsitektur Sistem

Arsitektur sistem terdiri dari tiga komponen utama yakni aplikasi *client*, basis data akun, dan *message queue*. Aplikasi *client* adalah komponen utama sistem. Komponen ini berfungsi sebagai komponen front-end atau antarmuka sistem dengan pengguna. Aplikasi *client* juga memiliki tugas sebagai komponen back-end pada sistem. Dengan demikian, aplikasi adalah komponen yang melakukan operasi CRUD (*Create, Read, Update, Delete*) pada basis data. Aplikasi juga yang menangani penerimaan dan pengiriman *message* dari dan ke *message queue*.

Komponen kedua dari sistem adalah basis data. Basis data berfungsi untuk menyimpan data yang bersifat tahan lama. Data ini tidak akan hilang meskipun aplikasi telah ditutup. Ada dua data utama yang disimpan basis data, yaitu data akun dan data skema. Data akun berfungsi untuk menyimpan semua informasi dari setiap akun, sedangkan data skema berguna untuk menyimpan informasi dari setiap skema *secret sharing*. Kedua data utama ini masing-masing direpresentasikan sebagai satu relasi pada basis data. Diagram relasional basis data diberikan pada Gambar 2.



Gambar 2. Diagram Relasional Basis Data

Relasi *account* memiliki tiga atribut yakni MAC *address*, *username*, dan *password*. MAC *address* berguna sebagai atribut identifikasi utama *account* karena setiap akun hanya bisa dipakai pada satu komputer saja. Dengan kata lain, setiap akun pasti memiliki MAC *address* unik. MAC *address* didaulat sebagai primary key dari *account*. Selain MAC *address*, *account* memiliki atribut *username* dan *password*. Setiap akun mempunyai *username* unik dan *password* yang berguna untuk proses autentikasi.

Relasi *scheme* memiliki lima atribut yakni *name*, *dealer*, *threshold value*, *number of players*, dan *encryption key*. *Name* merepresentasikan nama dari sebuah skema. Atribut ini adalah *primary key* dari relasi. Hal ini mengimplikasikan bahwa tidak akan ada dua atau lebih skema dengan nama yang sama. Lalu, terdapat juga atribut *dealer* yang menyatakan *username* dari akun yang bertindak sebagai *dealer* pada skema tersebut. Atribut *threshold value* dan *number of players* masing-masing mewakilkan nilai ambang (k) dan jumlah participant (n) dari skema. Atribut terakhir yakni *encryption key* merepresentasikan *encryption key* skema yang digunakan pada proses enkripsi dan dekripsi *share* dan sub*share*. *Key* ini dibangkitkan secara otomatis oleh aplikasi saat skema pertama kali dibuat.

Relasi *account* dan *scheme* memiliki hubungan *many-to-many*. Hal ini dikarenakan setiap akun dapat menjadi participant di lebih dari satu skema. Lalu, setiap skema dapat memiliki lebih dari satu akun sebagai participant.

Message queue adalah komponen yang memiliki tugas untuk menjadi perantara *message* yang dikirimkan antar aplikasi *client*. *Message* yang dikirimkan dapat berupa pesan teks yang berisi pemberitahuan mengenai suatu informasi atau permintaan untuk melakukan suatu aksi. *Message* juga dapat berbentuk dokumen atau senarai *byte*. *Message* berjenis ini berguna untuk mengirimkan *share* dan *subshare*. Sebagai perantara *message*, *message queue* akan menerima *message* yang dikirimkan dari pengirim. Lalu, ia akan mengirimkan *message* ke *client* tujuan hanya jika *client* tujuan sedang aktif atau *online*.

B. Protokol Operasi

Terdapat tiga operasi *secret sharing* yang harus dapat didukung oleh sistem, yaitu penyebaran *share*, rekonstruksi *secret*, dan pembaruan *share*. Pada bab ini akan dijelaskan protokol cara kerja dari setiap operasi pada sistem.

1) Protokol Penyebaran *Share*

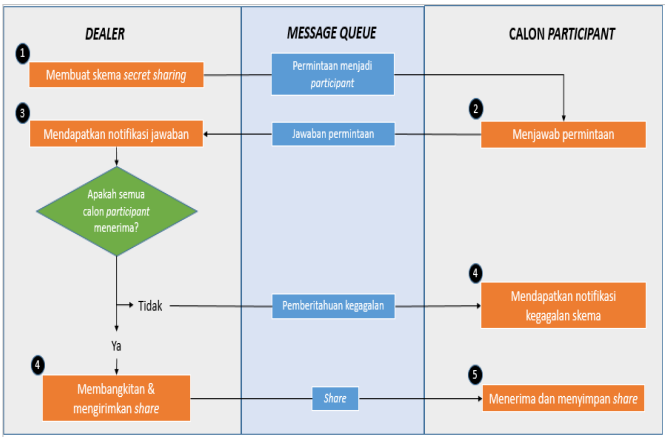
Penyebaran *share* adalah operasi yang wajib dilakukan setelah skema *secret sharing* dibuat oleh dealer. Pembuatan skema dilakukan dengan memberikan nama skema, menentukan nilai ambang (k), jumlah participant (n), dan daftar calon participant. Calon participant adalah akun yang ingin didaftarkan dealer sebagai participant. Calon participant pada akhirnya bisa saja tidak menjadi participant karena terdapat verifikasi persetujuan calon participant sebelum *share* disebarkan. Penyebaran *share* hanya dapat dilakukan apabila semua calon participant setuju untuk menjadi participant.

Proses verifikasi dilakukan dengan mengirimkan permintaan untuk menjadi participant ke setiap calon. Lalu, jawaban dari setiap calon akan dikirimkan ke dealer untuk

dilihat olehnya. Pengiriman permintaan dan jawaban dilakukan melalui *message queue*

Langkah-langkah lengkap operasi penyebaran *share* adalah:

1. Dealer membuat skema *secret sharing* baru dengan memberikan masukan nama skema, nilai ambang (k), jumlah participant (n), dan daftar calon participant. Lalu, proses verifikasi dimulai dengan dikirimkannya permintaan menjadi participant ke seluruh calon.
2. Calon participant menjawab permintaan dari dealer dengan jawaban ya atau tidak berdasarkan kepercayaan calon participant terhadap dealer. Jawaban ini dikirimkan ke dealer.
3. Dealer melihat jawaban dari setiap calon.
4. Langkah ini bergantung pada jawaban para calon participant. Jika terdapat minimal satu calon yang menolak permintaan, operasi penyebaran *share* dianggap gagal. Seluruh calon participant akan mendapatkan pemberitahuan mengenai hal ini dan operasi selesai. Jika seluruh calon menyetujui, mereka resmi menjadi participant. Sekarang, dealer dapat melakukan pembangkitan *share* dengan memberi masukan *secret* yang ia inginkan. Kemudian, aplikasi akan mengirimkan seluruh *share* yang telah dibangkitkan ke para participant.
5. Para participant menerima *share* yang dikirimkan. Mereka dapat menyimpannya di media penyimpanan lokal.



Gambar 3. Protokol Penyebaran *Share*

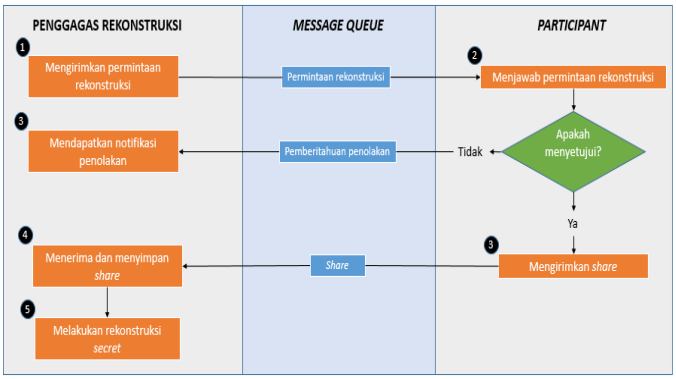
2) Protokol Rekonstruksi *Secret*

Operasi rekonstruksi *secret* dapat dilakukan baik oleh dealer ataupun participant. Disini orang yang ingin melakukan rekonstruksi disebut penggagas rekonstruksi. Untuk melakukan rekonstruksi, penggagas rekonstruksi membutuhkan *share-share* dari para participant hingga jumlahnya minimal mencapai nilai ambang (k). Para participant tidak boleh begitu saja mengirimkan *share* kepadanya karena terdapat proses verifikasi atau persetujuan.

Persetujuan dilakukan dengan pengiriman permintaan *share* dari penggagas ke para participant. Participant dapat menyetujui dengan mengirimkan *share* miliknya atau mengirimkan jawaban tidak. Jika ada participant yang menjawab tidak, operasi akan tetap dilanjutkan karena rekonstruksi tidak membutuhkan semua *share* (jika nilai ambang kurang dari jumlah participant). Konsekuensinya, pesan error dapat muncul saat penggagas melakukan rekonstruksi apabila jumlah *share* yang dimasukkan kurang dari nilai ambang.

Langkah-langkah lengkap operasi rekonstruksi *secret* adalah:

1. Terdapat dealer atau participant yang berperan sebagai penggagas rekonstruksi. Ia mengirimkan permintaan rekonstruksi ke seluruh participant.
2. Participant menerima permintaan rekonstruksi. Ia dapat menyetujui atau menolak permintaan berdasarkan kepercayaannya terhadap penggagas rekonstruksi.
3. Langkah ini bergantung pada aksi participant pada langkah 2. Jika ia menolak, penggagas akan menerima notifikasi penolakan dan tidak akan ada lagi langkah berikutnya. Jika ia menyetujui, ia harus mengunggah *share* miliknya ke aplikasi. Lalu, aplikasi akan mengirimkan *share* ke penggagas rekonstruksi.
4. Penggagas menerima *share* dari participant. Ia dapat menyimpan *share* di media penyimpanan lokal.
5. Penggagas melakukan rekonstruksi *secret* dengan mengunggah *share-share* yang telah ia terima. *Secret* yang dibangkitkan bisa disimpan pada media penyimpanan lokal.



Gambar 4. Protokol Rekontruksi *Secret*

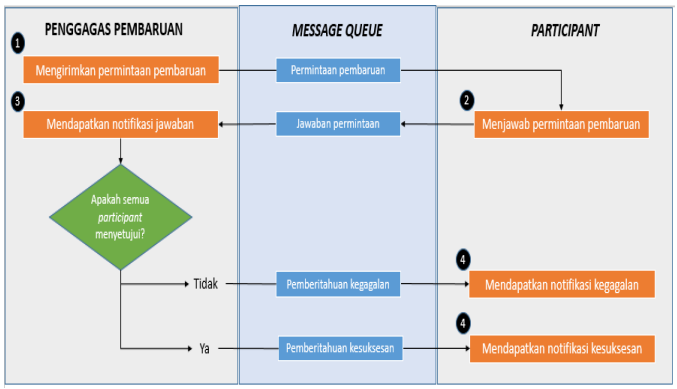
3) Protokol Pembaruan *Share*

Operasi pembaruan *share* diawali dengan adanya motivasi dealer atau salah satu participant untuk memperbarui *share*. Disini dealer atau participant tersebut disebut sebagai penggagas pembaruan. Penggagas akan meminta para participant untuk memperbarui *share* merekea. Para participant tidak boleh begitu saja mengiyakan permintaan penggagas. Hal ini dikarenakan bisa saja participant tidak lagi mempercayai penggagas atau mereka yakin bahwa akun penggagas telah diambil alih oleh pihak lain yang berbahaya. Oleh karena itu, harus ada mekanisme yang dapat memastikan pembaruan *share* hanya akan dilakukan apabila seluruh participant menyetujui. Semua participant harus menyetujui karena pembaruan *share* adalah operasi yang melibatkan seluruh participant. Jika salah satu participant tidak setuju, operasi dianggap gagal.

Verifikasi dapat dilakukan dengan melakukan pengiriman pembaruan *share* dari penggagas ke para participant. Participant dapat menjawab permintaan dengan jawaban ya atau tidak. Jawaban ini akan dikirimkan ke penggagas. Permintaan pembaruan dapat dikirimkan melalui *message queue*

Langkah-langkah lengkap proses verifikasi pada operasi pembaruan *share* adalah:

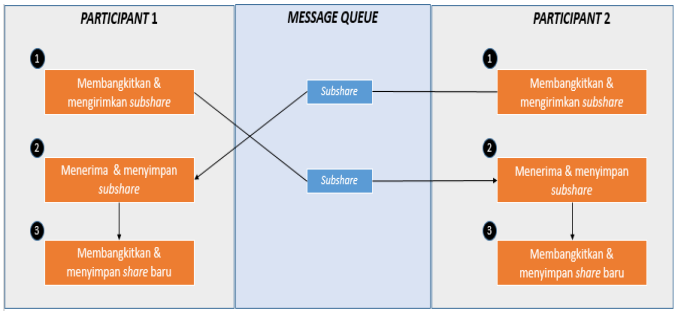
1. Penggagas pembaruan mengirimkan permintaan pembaruan *share* ke seluruh participant.
2. Participant menerima atau menolak permintaan berdasarkan kepercayaannya terhadap penggagas pembaruan. Jawaban participant dikirimkan ke penggagas .
3. Penggagas menerima dan melihat jawaban dari participant.
4. Langkah ini bergantung pada jawaban para participant. Jika ada participant yang menolak, proses verifikasi gagal dan pembaruan *share* tidak akan dilakukan. Jika seluruh participant setuju, pembaruan *share* dapat mulai dilakukan. Seluruh participant akan mendapatkan pemberitahuan mengenai kegagalan atau kesuksesan verifikasi.



Gambar 5. Protokol Proses Verifikasi pada Pembaruan *Share*

Jika proses verifikasi berhasil, para *participant* dapat memulai proses pembaruan *share* dengan *proactive secret sharing*. Langkah-langkah proses pembaruan adalah:

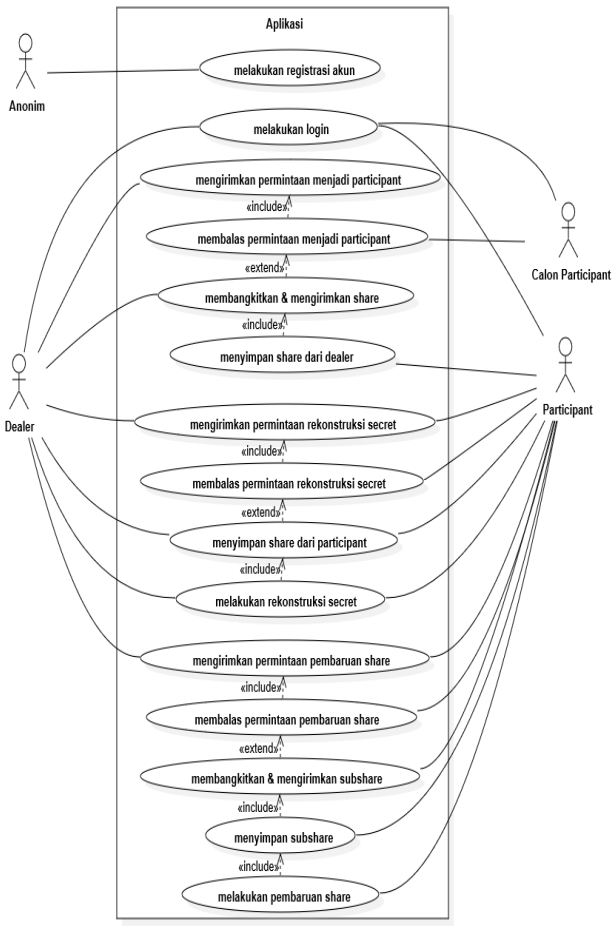
1. Setiap participant membangkitkan sub*share* sejumlah banyaknya participant (n). Sub*share* dibangkitkan secara otomatis oleh aplikasi. Kemudian, aplikasi mengirimkan sub*share*-sub*share* tersebut ke seluruh participant.
2. Participant menerima sub*share*-sub*share* dari para participant lain. Ia dapat menyimpan sub*share* di media penyimpanan lokal.
3. Setelah semua participant telah selesai mengirimkan sub*share*, setiap participant dapat memulai proses pembaruan *share*. Proses ini dilakukan dengan mengunggah *share* lama serta semua sub*share* yang telah diterima ke aplikasi. Aplikasi akan membangkitkan *share* baru yang dapat diunduh participant. Terakhir, participant harus menghapus *share* lamanya secara mandiri.



Gambar 6. Protokol Proses Pembaruan *Share* dengan *Proactive Secret Sharing* pada Pembaruan *Share*

C. Diagram *Use case*

Dari spesifikasi aplikasi, diturunkan 18 kebutuhan fungsional dan 2 fungsi kebutuhan non-fungsional. Berdasarkan kebutuhan fungsional & non-fungsional, ditentukan *use case* sistem yang diberikan pada Gambar 7.



Gambar 7. Diagram *Use case*

Terdapat 4 aktor pada *use case* yakni anonim, *dealer*, calon *participant*, dan *participant*. Anonim adalah Pengguna yang belum terdaftar. *Dealer* adalah pengguna terdaftar yang

membuat skema *secret sharing* dan kemudian menyebarkan *share.* Calon *participant* adalah pengguna terdaftar yang menerima permintaan untuk menjadi *participant. Participant* adalah pengguna terdaftar yang menyimpan *share* dari *dealer*.

III. IMPLEMENTASI

Aplikasi dikembangkan sebagai *desktop application* pada platform Windows. Komputer yang digunakan untuk pengembangan aplikasi memiliki *processor* Intel Core i3-4005U dengan RAM 4 GB. Pada pengembangan, digunakan bahasa pemrograman Visual C# dengan IDE Visual Studio Community.

Selain komputer pengembangan, terdapat komputer lain yang berperan sebagai *server*. Komputer *server* memiliki *processor* Intel Xeon CPU E3-2630L dengan RAM 512 MB. Pada *server*, dipasang kakas basis data dan *message queue*. Kakas basis data yang digunakan adalah MySQL. Kakas *message queue* yang digunakan adalah RabbitMQ.

Pada basis data MySQL, relasi *account* dan *scheme* masing-masing diimplementasikan menjadi tabel *account* dan tabel *scheme*. Lalu, terdapat satu tabel tambahan yakni tabel *players* yang merepresentasikan hubungan *many-to-many* antara *account* dengan *scheme*. Tabel *account send*iri memiliki atribut-atribut yang sama dengan atribut-atribut relasi *account* pada. Tabel *scheme* juga memiliki semua atribut yang ada pada relasi *scheme*. Namun, tabel *scheme* memiliki dua atribut tambahan yaitu *create_date* dan *num_of_confirmations*. *create_date* adalah atribut yang menyimpan tanggal dibuatnya skema. Lalu, *num_of_confirmations* adalah atribut yang bertugas untuk menyimpan jumlah *participant* yang telah memberikan konfirmasi pada operasi penyebaran *share* dan pembaruan *share*.

Message queue diimplementasikan dengan RabbitMQ. RabbitMQ adalah *message broker* yang menggunakan protokol *Advanced Message Queueing Protocol* (AMQP). Pada AMQP, pihak yang mengirim *message* ke *message broker* disebut *publisher* atau *producer*, sedangkan pihak yang menerima *message* disebut *consumer*. Publisher tidak mengirimkan *message* langsung ke *queue* melainkan ke *exchange*. Setelah menerima *message*, *exchange* akan meneruskan *message* ke *queue* yang bersesuaian. Setelah *message* sampai di *queue*, *queue* tidak melakukan pengiriman langsung ke *consumer*. *Consumer*-lah pihak yang aktif untuk menerima *message*. Untuk mendapatkan *message* dari suatu *queue*, *consumer* harus melakukan *subscribe* terhadap *queue* tersebut.

Pada aplikasi ini, setiap pengguna direpresentasikan oleh satu *queue* pada RabbitMQ. *Queue* otomatis dibuat saat pengguna selesai melakukan registrasi. Setiap pengguna melakukan login ke aplikasi, aplikasi melakukan *subscribe* ke *queue* tersebut. Ketika sebuah skema *secret sharing* sedang dibuat, aplikasi akan membuat sebuah *exchange* yang merepresentasikan skema tersebut. Setiap *queue* dari calon participant akan terikat ke *exchange* tersebut. Jika seluruh calon partcipant mau menjadi participant, maka tidak ada perubahan dalam sistem. *Exchange* skema digunakan sebagai "pintu" untuk seluruh *message* yang ditujukan ke participant. Jika ada calon participant yang menolak menjadi participant, maka skema gagal dibuat. Lalu, *exchange* skema akan dihapus dan semua *queue* calon participant dihapus ikatannya dari *exchange*.

IV. PENGUJIAN

Pengujian dilakukan untuk memastikan aplikasi telah memenuhi seluruh kebutuhan fungsional dan non-fungsional. Kebutuhan fungsional telah dinyatakan sebagai *use case.* Kebuutuhan non-fungsional aplikasi adalah aspek keamanan data saat dikirimkan. Data yang dimaksud disini adalah *share* dan *subshare.*

Share dan *subshare* dienkripsi menggunakan *symmetric key encryption* agar kontennya tidak dapat terbaca oleh penyadap. Pengujian dilakukan untuk memastikan hal tersebut. Pengujian dilakukan menggunakan aplikasi *Wireshark* yang berguna untuk melakukan *sniffing* atau penyadapan dari setiap *packet* yang keluar-masuk dari dan ke komputer melalui jaringan. Secara umum, pengujian dilakukan dengan melacak *packet* yang menggunakan protokol AMQP. Lalu, setiap *packet* diperiksa untuk mencari *share* dan *subshare.* Setelah didapatkan *packet* yang berisi *share* atau *subshare*, konten *packet* tersebut dibandingkan dengan *share* atau *subshare* yang asli. Jika isi *packet* berbeda, maka enkripsi berhasil dilakukan dan pengujian sukses. Jika isinya sama, maka pengujian gagal. Pengujian ini dilakukan setiap saat *share* dan *subshare* dikirimkan dari komputer pengguna ke *server* dan sebaliknya.

Pengujian kebutuhan fungsional dan non-funsional dilakukan secara bersamaan dengan teknik *black box testing*. *Black-box testing* adalah pengujian aplikasi tanpa melihat kode aplikasi atau tanpa perlu mengetahui bagaimana aplikasi bekerja. Total terdapat 27 skenario pengujian yang perlu diuji. Seluruh skenario pengujian dilakukan pada skema ambang (2,3) menggunakan 4 buah komputer berbeda. Satu komputer berperan sebagai *dealer* dan 3 komputer lainnya sebagai *participant.*

Setelah dilakukan pengujian, ternyata aplikasi mampu melewati seluruh skenario pengujian dengan baik. Dua puluh tiga skenario pengujian dilakukan dengan hasil sukses. Dengna demikian, aplikasi telah memenuhi kebutuhan fungsional dan kebutuhan non-fungsionalnya.

V. SIMPULAN DAN SARAN

A. Simpulan

Berdasarkan proses implementasi dan pengujian yang telah dilakukan, didapatkan simpulan-simpulan berikut.

1. Tiga protokol untuk operasi penyebaran *share*, rekonstruksi *secret*, dan pembaruan *share* dengan *proactive secret sharing* telah dihasilkan.
2. Arsitektur sistem yang mampu menangani operasi penyebaran *share*, rekonstruksi *secret*, dan pembaruan *share* dengan *proactive secret sharing* telah dihasilkan.
3. Aplikasi *secret sharing* terdistribusi yang mampu menangani operasi penyebaran *share*, rekonstruksi

secret, dan pembaruan *share* dengan *proactive secret sharing* berhasil dibuat.

B. Saran

Aplikasi yang dikembangkan masih memiliki beberapa kekurangan dan masih dapat dikembangkan lebih lanjut. Saran-saran yang dapat dilakukan adalah:

1. Protokol dari setiap operasi rekonstruksi *secret* dan pembaruan *share* dapat diperbaiki pada tahap pengiriman permintaan. Setelah permintaan dikirimkan, aplikasi tidak perlu mengirimkan permintaan ke pihak pengirim.
2. Pada tahap permintaan pada ketiga operasi *secret sharing*, aplikasi akan terus menunggu jawaban dari seluruh pengguna. Jika pengguna ternyata sudah tidak aktif menggunakan aplikasi, ia tidak akan pernah menjawab permintaan. Masalah ini dapat diatasi dengan menambahkan timeout atau batas waktu untuk menjawab sebuah permintaan.
3. Tidak aktifknya pengguna dapat mengakibatkan penuhnya basis data dan *message queue* oleh data pengguna tersebut. Sebaiknya aplikasi memiliki fitur penghapusan pengguna untuk menghapus pengguna-pengguna yang sudah tidak aktif.
4. Tampilan aplikasi masih harus perlu diperbaiki. Terdapat beberapa perbaikan yang dapat meningkatkan kualitas visual aplikasi, diantaranya:
   - *Form* untuk *register* sebaiknya dipisah dari *form* untuk login agar tidak terlihat membingungkan.
   - Aplikasi sebaiknya menggunakan skema terakhir yang digunakan sebagai pilihan standar pada dropdown yang berisi skema-skema pada submenu *Reconstruct Secret* dan *Update Share*. Hal ini dikarenakan pengguna biasanya memilih skema yang terakhir kali ia gunakan. Dengan solusi ini, pengguna tidak perlu lagi melakukan *scrolling* untuk memilih skema yang ia pilih terakhir kali.
   - *Participant* tidak dapat mengganti lokasi *share* yang ia unggah saat menyetujui permintaan rekonstruksi *secret*. Hal ini dikarenakan setelah tombol *Upload* ditekan, tombol tersebut hilang diganti oleh tombol *Send*. Masalah ini dapat diatasi dengan memisahkan tombol *Upload* dengan tombol *Send*.

REFERENSI

[1] Bai & Zou (2009). *A Proactive Secret Sharing Scheme in Matrix Projection Method*. International Journal of Security and Networks, Vol. 4, No. 4

[2] Beimel, Amos (2011). *Secret-Sharing Schemes: A Survey*. Coding and Cryptology, Third International Workshop IWCC, Lecture Notes in Computer Science 6639, 11-46.

[3] Benevuto, Christoforus J. (2012). *Galois Field in Cryptography*. Amerika Serikat: University of Washington.

[4] Birkhoff, G. & Mac Lane, S. (1996). *A Survey of Modern Algebra*. New York, Amerika Serikat : Macmillan.

[5] Liu, Chung L. (1968). *Introduction to Combinatorial Mathematics*. Amerika Serikat: Mcgraw-Hill.

[6] Herzberg, dkk. (1995). *Proactive Secret Sharing, or: How to Cope with Perpetual Leakage*. Advances in Cryptology – CRYPTO 1995, 457-469.

[7] Matthew (2007). *Secret Sharp*. https://sourceforge.net/projects/*secret*sharp/. Terakhir diakses pada 21 Juli 2016.

[8] Ostrovsky, Rafail & Yung, Moti (1991). *How to Withstand Mobile Virus Attacks*. ACM Symposium on Principles of Distributed Computing 1991, 51-59.

[9] Pivotal Software (2007). *AMQP 0-9-1 Model Explained*. https://www.rabbitmq.com/tutorials/amqp-concepts.html. Terakhir diakses pada 26 November 2016.

[10] Poettering, B. (2005). *Shamir's Secret Sharing Scheme*. http://point-at-infinity.org/ssss/. Terakhri diakses pada 21 Juli 2016.

[11] Shamir, Adi (1979). *How to Share A Secret*. Communication of the ACM, November 1979 Volume 22 Number 11, 612-613.

[12] Silverstone, Daniel (2006). *Libgfshare – A Secret Sharing Library*. http://www.digital-scurf.org/software/libgf*share*. Terakhir diakses pada 21 Juli 2016.

[13] Tanenbaum & Van Steen (2007). *Distributed Systems: Principles and Paradigms*. Amerika Serikat: Springer.

[14] Zargaryan, Hayk. *Secret Sharing*. https://sourceforge.net/projects/*secretsharing*/. Terakhir diakses pada 21 Juli 2016.