RANCANG BANGUN SISTEM INFORMASI BERBAGI LOKASI BERBASIS ANDROID DENGAN IMPLEMENTASI ALGORITMA SCRYPT PADA PROSES AUTENTIFIKASI

DESIGN OF LOCATION SHARING INFORMATION SYSTEM BASED ON ANDROID USING SCRYPT ALGORITHM IMPLEMENTATION ON AUTHENTICATION PROCESS

Faiz Faidurrahman[1], Gun Gun Gumilar[2]

Program Studi Informtika – Universitas Bakrie[1],
Program Studi Sistem Informasi – Universitas Bakrie[2]
faizfaidurrahman@gmail.com[1], gggumilar@yahoo.com[2]

Abstrak

Penelitian ini bertujuan untuk merancang dan membangun sistem informasi berbagi lokasi dengan mengimplementasikan algoritma scrypt pada proses otentifikasi. Sistem informasi ini dikembangkan menggunakan model pengembangan *prototype* dengan pendekatan *hybrid* pada sistem operasi android. Dengan memanfaatkan Google Maps API untuk memberikan petunjuk arah lokasi yang telah dibagikan dalam sebuah *group*, sistem informasi ini diharapkan dapat membantu pengguna untuk berbagi informasi lokasi kepada pengguna lainnya dalam sebuah *group*. Penerapan algoritma scrypt untuk mengenkripsi *password* pada proses otentifikasi diharapkan dapat meningkatkan keamanan data dari penggunaan sistem informasi oleh orang yang tidak berhak.

Kata Kunci: Berbagi Lokasi, Algoritma Scrypt, Enkripsi Data, *Hybrid* Android, *Prototype Model*

Abstract

This study aims to design and develop location sharing information system by implementing scrypt algorithm on authentication process. The information system was developed using a prototype development model and applying hybrid approach to the android operating system. By utilizing the Google Maps API to provide directions to locations shared in a group, this application is expected to help users to share location information to other users in the group. Implementing scrypt algorithm to encrypt the password in the authentication process is expected to improve the data security of application usage by unauthorized persons.

Keywords: *Sharing Location, Scrypt Algorithm, Data Encryption, Hybrid Android, Prototype Model.*

Tanggal Terima Naskah : 13 Desember 2017
Tanggal Persetujuan Naskah : 31 Januari 2018

1. PENDAHULUAN

Seiring dengan perkembangan teknologi, integrasi teknologi *mobile*, *Geographic Informasiton System* (GIS), dan *Global Positioning System* (GPS) telah memungkinkan dikembangkan sistem informasi *mobile* yang interaktif [1]. Dengan memanfaatkan *Application Programming Interfaces* (API), informasi lokasi yang didapatkan dari sebuah data dapat ditampilkan ke dalam *map*. Informasi lokasi tersebut dapat digunakan untuk melihat lokasi suatu benda, mencari posisi dari sebuah alamat, memberikan petunjuk arah dalam mengemudi, dan melakukan banyak hal lainnya [2].

Berdasarkan survei yang dilakukan dengan jumlah responden sebanyak 78 orang, 91,1% responden menyatakan bahwa mereka ingin berbagi informasi lokasi tempat dan 92,3% responden menyatakan bahwa informasi lokasi tempat tersebut perlu untuk disimpan sehingga dapat digunakan lagi. Sebanyak 94,9% dari responden juga menyebutkan bahwa keamanan atau privasi dalam berbagi informasi juga merupakan faktor penting yang mempengaruhi keputusan mereka dalam berbagi informasi lokasi tempat.

Salah satu cara yang paling populer dalam meningkatkan keamanan dan privasi adalah penggunaan *password* [3] namun *password* juga terkenal mudah untuk diserang karena penggunaan *password* yang lemah [4], ditambah dengan kemampuan komputer yang semakin cepat membuat penyerang dapat melakukan serangan dengan menebak *password*. Dalam beberapa tahun belakangan ini banyak terjadi kasus kebocoran *database* dengan berbagai macam serangan [5]. Hal ini menjadi ancaman besar apabila data kredensial seperti *password* yang terdapat pada *database* disimpan dalam bentuk teks biasa karena penyerang dapat mendapatkannya dengan mudah. Untuk mencegah hal tersebut maka *password* secara khusus disimpan menggunakan *hash* kriptografi [3].

Terdapat tiga algoritma yang paling cocok untuk menyimpan *password* dalam bentuk *hash*, yaitu PBKDF2, bcrypt, dan scrypt [6]. Namun, scrypt lebih unggul dalam mengatasi serangan karena harga perangkat keras yang digunakan untuk *crack password* yang menggunakan algoritma scrypt lebih mahal daripada menggunakan PBKDF2 atau bcrypt [7].

Berdasarkan hal tersebut pada penelitian ini dirancang sebuah sistem informasi yang diberi nama GLoSha (*Grouping Location Sharing*) yang dapat membantu orang-orang dalam berbagi informasi lokasi tempat dan mengimplementasikan algoritma scrypt untuk meningkatkan keamanan dalam penyimpanan data. Keamanan dalam penyimpanan data dibutuhkan pada sistem informasi GLoSha karena sistem informasi GLoSha sendiri memberikan layanan penyimpanan data informasi lokasi yang bersifat *private* dimana hanya orang-orang tertentu yang dapat melihat data tersebut. Hal ini dapat dimanfaatkan oleh *user* yang memiliki informasi data lokasi tempat yang penting dan data tersebut butuh untuk disimpan atau dibagi, namun hanya boleh dilihat oleh *user* tersebut atau *user* yang diinginkan oleh si pemilik lokasi.

2. KONSEP DASAR

2.1. *The scrypt Key Derivation Function*

Fungsi *scrypt Key Derivation* awalnya dikembangkan untuk digunakan pada Tarsnap dalam sistem *backup online* dan dirancang untuk menjadi jauh lebih aman terhadap serangan perangkat keras seperti *brute-force* daripada fungsi alternatif lainnya seperti PBKDF2 atau bcrypt. Fungsi scrypt bertujuan untuk mengurangi keuntungan yang didapatkan oleh penyerang dengan menggunakan sirkuit paralel yang dirancang khusus untuk memecahkan *password-based key derivation functions* [8]. Untuk menghasilkan algoritma scrypt *final* dibutuhkan algoritma-algorima berikut:

a. Fungsi Salsa20/8 core

Salsa 20/8 Core adalah pengurangan putaran varian dari Salsa20 core [9] dan merupakan fungsi *hash* yang membaca *input* 64 oktet *string* dan menghasilkan *output* 64 oktet *string*. Salsa20/8 core bukanlah sebuah fungsi *hash* kriptografi karena *input* yang berbeda pada algoritma ini dapat menghasilkan *hash* yang sama [10]. Pada Gambar 2.2 merupakan referensi kode untuk fungsi SALSA20 core.

```
#define R(a,b) (((a) << (b)) | ((a) >> (32 - (b))))
void salsa208_word_specification(uint32 out[16],uint32 in[16])
{
  int i;
  uint32 x[16];
  for (i = 0;i < 16;++i) x[i] = in[i];
  for (i = 8;i > 0;i -= 2) {
    x[ 4] ^= R(x[ 0]+x[12], 7);  x[ 8] ^= R(x[ 4]+x[ 0], 9);
    x[12] ^= R(x[ 8]+x[ 4],13);  x[ 0] ^= R(x[12]+x[ 8],18);
    x[ 9] ^= R(x[ 5]+x[ 1], 7);  x[13] ^= R(x[ 9]+x[ 5], 9);
    x[ 1] ^= R(x[13]+x[ 9],13);  x[ 5] ^= R(x[ 1]+x[13],18);
    x[14] ^= R(x[10]+x[ 6], 7);  x[ 2] ^= R(x[14]+x[10], 9);
    x[ 6] ^= R(x[ 2]+x[14],13);  x[10] ^= R(x[ 6]+x[ 2],18);
    x[ 3] ^= R(x[15]+x[11], 7);  x[ 7] ^= R(x[ 3]+x[15], 9);
    x[11] ^= R(x[ 7]+x[ 3],13);  x[15] ^= R(x[11]+x[ 7],18);
    x[ 1] ^= R(x[ 0]+x[ 3], 7);  x[ 2] ^= R(x[ 1]+x[ 0], 9);
    x[ 3] ^= R(x[ 2]+x[ 1],13);  x[ 0] ^= R(x[ 3]+x[ 2],18);
    x[ 6] ^= R(x[ 5]+x[ 4], 7);  x[ 7] ^= R(x[ 6]+x[ 5], 9);
    x[ 4] ^= R(x[ 7]+x[ 6],13);  x[ 5] ^= R(x[ 4]+x[ 7],18);
    x[11] ^= R(x[10]+x[ 9], 7);  x[ 8] ^= R(x[11]+x[10], 9);
    x[ 9] ^= R(x[ 8]+x[11],13);  x[10] ^= R(x[ 9]+x[ 8],18);
    x[12] ^= R(x[15]+x[14], 7);  x[13] ^= R(x[12]+x[15], 9);
    x[14] ^= R(x[13]+x[12],13);  x[15] ^= R(x[14]+x[13],18);
  }
  for (i = 0;i < 16;++i) out[i] = x[i] + in[i];
}
```

Gambar 1. Algoritma Salsa20 core [9]

b. Algoritma scryptBlockMix

Algoritma scryptBlockMix dalam pengaplikasiannya berdasarkan algoritma BlockMix [9] namun menggunakan Salsa20/8 core sebagai fungsi *hash* pada algoritma scryptBlockMix. Salsa (T) pada gambar 2 merupakan fungsi Salsa20/8 core. [10].

```
Algorithm scrypt

Input:
        P       Passphrase, an octet string.
        S       Salt, an octet string.
        N       CPU/Memory cost parameter, must be larger than 1,
                a power of 2 and less than 2^(128 * r / 8).
        r       Block size parameter.
        p       Parallelization parameter, a positive integer
                less than or equal to ((2^32-1) * hLen) / MFLen
                where hLen is 32 and MFlen is 128 * r.
        dkLen   Intended output length in octets of the derived
                key; a positive integer less than or equal to
                (2^32 - 1) * hLen where hLen is 32.

Output:
        DK      Derived key, of length dkLen octets.

Steps:

  1. B[0] || B[1] || ... || B[p - 1] =
       PBKDF2-HMAC-SHA256 (P, S, 1, p * 128 * r)

  2. for i = 0 to p - 1 do
       B[i] = scryptROMix (r, B[i], N)
     end for

  3. DK = PBKDF2-HMAC-SHA256 (P, B[0] || B[1] || ... || B[p - 1],
                              1, dkLen)
```

Gambar 2. Algoritma *scrypt* [10]

c. Algoritma scryptROMix

Algoritma scryptROMix dalam pengaplikasiannya berdasarkan algoritma ROMix [11] namun pada Algoritma *scryptROMix* merupakan rangkaian dari algoritma *scyptBlockMix* yang berjumlah N dan memiliki dua *loop* dimana pada *loop* kedua menggunakan fungsi *integrify* untuk menentukan urutan rangkaian [10].

```
Algorithm scryptROMix

Input:
         r       Block size parameter.
         B       Input octet vector of length 128 * r octets.
         N       CPU/Memory cost parameter, must be larger than 1,
                 a power of 2 and less than 2^(128 * r / 8).

Output:
         B'      Output octet vector of length 128 * r octets.

Steps:

  1. X = B

  2. for i = 0 to N - 1 do
       V[i] = X
       X = scryptBlockMix (X)
     end for

  3. for i = 0 to N - 1 do
       j = Integerify (X) mod N
             where Integerify (B[0] ... B[2 * r - 1]) is defined
             as the result of interpreting B[2 * r - 1] as a
             little-endian integer.
       T = X xor V[j]
       X = scryptBlockMix (T)
     end for

  4. B' = X
```

Gambar 3. Algoritma scryptROMix [10]

d. Algoritma scrypt

Algoritma scrypt yang akan digunakan untuk *encrypt password* pada fungsi *login* dan *signup* dari aplikasi GloSha menggunakan Fungsi PBKDF2-HMAC-SHA-256. Seperti yang ditunjukkan pada Gambar 4, algoritma PBKDF2 dalam algoritma scrypt digunakan dengan HMAC-SHA-256 sebagai *Pseudo Random Function*. Fungsi HMAC-SHA-256 menghasilkan *output* 32 oktet *string* [24]. Selanjutnya hasil dari PBKDF2-HMAC-SHA-256 tersebut diproses oleh *block core* algoritma scryptROMix sebanyak jumlah p dan untuk setiap *block* mempunyai ukuran 128 * r. Setelah diproses menggunakan algoritma scryptROMix semua *block* diproses ulang oleh PBKDF2-HMAC-SHA-256 untuk menghasilkan *output* dengan panjang dkLen.

```
Algorithm scrypt

Input:
        P       Passphrase, an octet string.
        S       Salt, an octet string.
        N       CPU/Memory cost parameter, must be larger than 1,
                a power of 2 and less than 2^(128 * r / 8).
        r       Block size parameter.
        p       Parallelization parameter, a positive integer
                less than or equal to ((2^32-1) * hLen) / MFLen
                where hLen is 32 and MFlen is 128 * r.
        dkLen   Intended output length in octets of the derived
                key; a positive integer less than or equal to
                (2^32 - 1) * hLen where hLen is 32.

Output:
        DK      Derived key, of length dkLen octets.

Steps:

  1. B[0] || B[1] || ... || B[p - 1] =
       PBKDF2-HMAC-SHA256 (P, S, 1, p * 128 * r)

  2. for i = 0 to p - 1 do
       B[i] = scryptROMix (r, B[i], N)
     end for

  3. DK = PBKDF2-HMAC-SHA256 (P, B[0] || B[1] || ... || B[p - 1],
                              1, dkLen)
```

Gambar 4. Algoritma *scrypt* [24]

Berikut adalah perbandingan algoritma PBKDF2, bcrypt, dan scrypt, serta perkiraan biaya *hardware* untuk melakukan *crack* sebuah *password*

Tabel 1. Perbandingan Algoritma PBKDF2, bcrypt, dan scrypt

Author	**Nama Algoritma**	**Parameter Algoritma**	**Tujuan**	**Perkiraan Biaya *Hardware* untuk *Crack Password* dalam 1 tahun** [8]			
				8 *Char* ≤ 100ms	**10 *Char* ≤ 100ms**	**8 *Char* ≤ 5s**	**10 *Char* ≤ 5s**
[12]	PBKDF2	PBKDF2 (HMAC, *Password*, *Salt*, c, dklen)	Untuk memperlambat serangan *dictionary* atau *brute force* dengan meningkatkan waktu yang dibutuhkan untuk mencoba setiap *password*	$18k	$160M	$920k	$8.3B
[13]	bcrypt	Bcrypt (Cost, Salt, Pwd)	Dapat beradaptasi terhadap perkembangan *hardware*.	$130k	$1.2B	$4.3M	$39B
[10]	scrypt	PBKDF2-HMAC-SHA-256 (P, S, N, r, p, dkLen)	Memberikan proteksi tambahan melawan serangan yang menggunakan *custom hardware*	$4.8M	$43B	$19B	$175T

3. METODE PENELITIAN

Berikut adalah tahapan - tahapan pengembangan pada penelitian ini:

3.1. *Communication*

Pada tahapan ini dilakukan identifikasi terhadap tujuan dari sistem informasi GLoSha yang akan dibangun, mengidentifikasi target pengguna, dan menentukan jenis *platform* yang akan digunakan. Beberapa hal tersebut diperoleh dengan cara melakukan survei terhadap *potential user* dari sistem informasi GLoSha. Langkah selanjutnya adalah memahami teknologi yang akan digunakan dan menentukan informasi yang diletakkan di dalam sistem informasi.

3.2. *Quick Plan*

Pada tahapan ini seluruh informasi yang diperoleh pada tahapan sebelumnya digabungkan, kemudian dilakukan analisis terhadap kebutuhan fungsional dari sistem, kebutuhan data masukan dan data keluaran.

3.3. Pemodelan *Quick Design*

Pada tahapan ini dirancang desain model *Object Oriented Programming* (OOP) dari sistem informasi GLoSha dan merepresentasikan ke dalam desain *logical* dan *physical* yang akan dibangun pada tahapan *Construction of Prototype*.

3.4. *Construction of Prototype*

Pada tahapan ini sistem informasi GLoSha dibangun berdasarkan hasil rancangan dari fase *Quick Pl*an dan Pemodelan *Quick Design*.

3.5. *Deployment Delivery and Feedback*

Setelah sistem informasi selesai dibangun, dilakukan pengujian terhadap sistem informasi. Tahapan pengujian terbagi dua, yaitu:

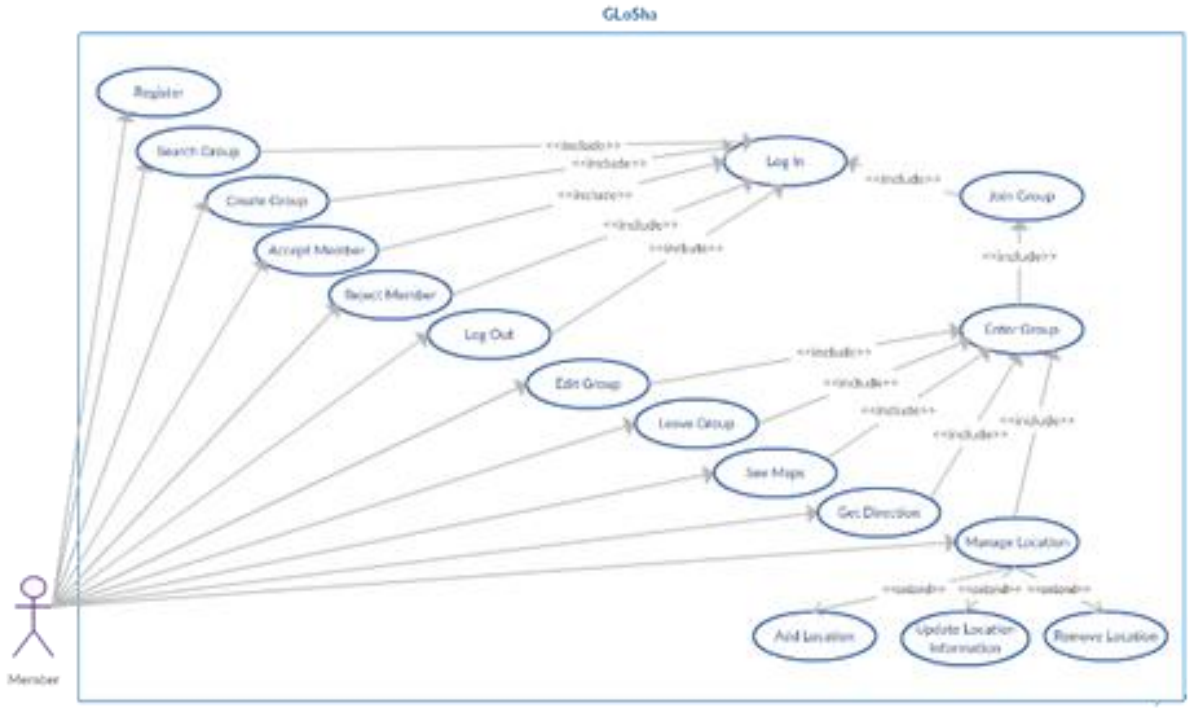
a. *White Box Testing*
   Pengujian ini dilakukan untuk mengetahui apakah tiap-tiap modul berjalan dan mengasilkan *output* yang diharapkan dengan melihat ke dalam modul untuk meneliti dan menganalisis kode-kode program yang ada.
b. *Black Box Testing*
   Pengujian ini dilakukan untuk mengetahui apakah fungsi dari sistem informasi berjalan sesuai yang diharapkan tanpa harus melihat ke dalam modul. Pengujian dilakukan dengan melakukan pengujian sistem informasi oleh penguji. Penguji adalah orang-orang yang dipilih untuk melakukan pengujian sistem informasi sesuai skenario yang sudah ditentukan.

4. HASIL DAN PEMBAHASAN

Pada penelitian ini data yang dibutuhkan adalah data *user*, data *group*, data anggota *group*, dan data informasi lokasi tempat. Data *user* digunakan untuk membuat hak akses *user* terhadap sistem. Data *group* digunakan untuk melihat *list group* yang ada di dalam sistem. Data anggota *group* adalah data *user* yang memiliki relasi dengan data *group*. Data informasi lokasi tempat adalah data yang dibuat oleh data anggota *group*. Setelah semua data didapatkan dilanjutkan dengan perancangan sistem informasi.

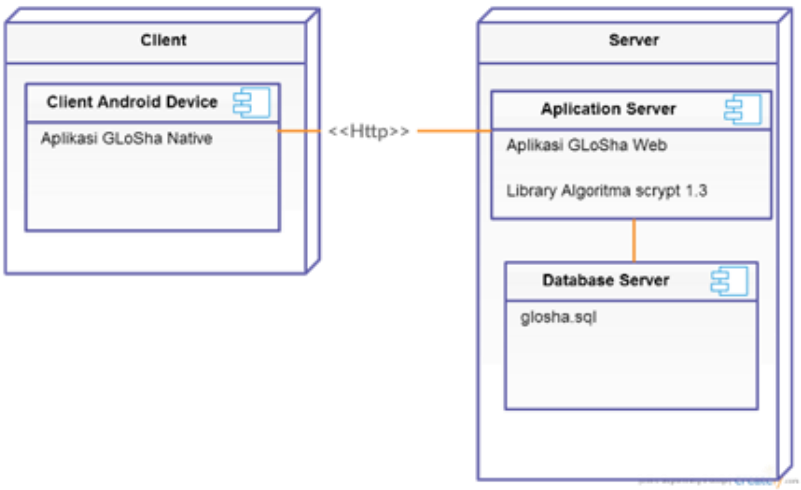
4.1. Perancangan Sistem

Berikut merupakan gambaran *use case* dari sistem informasi GloSha. Dari *use case* tersebut terlihat fungsionalitas yang bisa didapatkan.



Gambar 5. *Use Case Diagram*

Berikut adalah gambaran *deployment diagram* dari sistem informasi GloSha

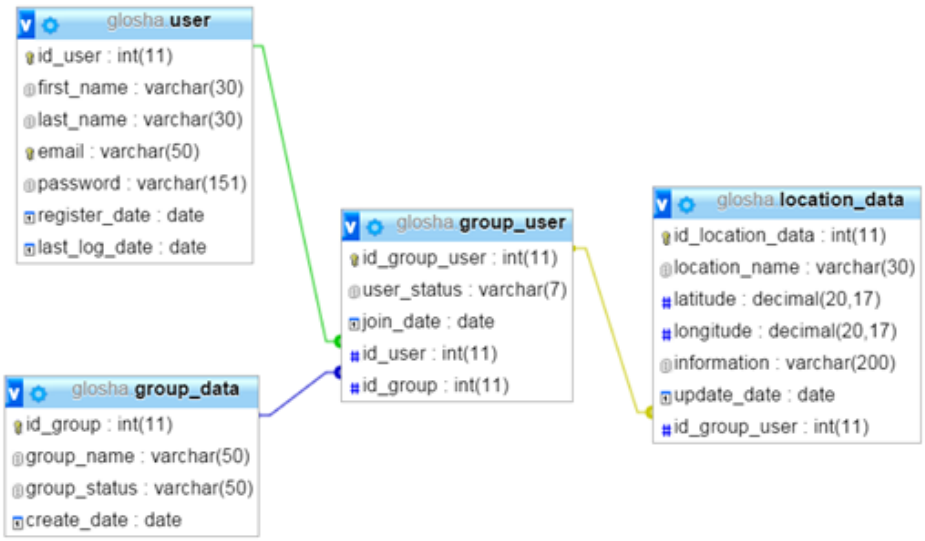


Gambar 6. *Deployment Diagram* Sistem informasi GloSha

Diagram pada Gambar 6 menjelaskan *package* rancangan Sistem informasi GLoSha yang terdiri dari *client* dan *server*. *Client* berisi sistem informasi GloSha *native* yang terhubung dengan *server*, sedangkan *server* berisi sistem informasi GloSha *web Library Algoritma* scrypt 1.3 yang terhubung dengan *database server*.

4.2. Perancangan *Database*

Rancangan *database* sistem informasi GloSha dapat dilihat pada gambar 7.

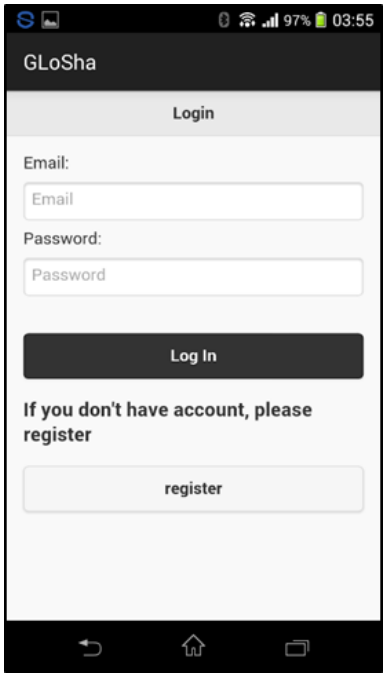


Gambar 7. Rancangan *Database*

Pada gambar rancangan *database* menggunakan 4 tabel. Tabel user untuk menyimpan data *user*, tabel group_data untuk menyimpan data *group*, tabel group_user untuk menyimpan data *user* yang berada di dalam *group*, dan tabel location_data untuk menyimpan data lokasi yang telah dibagi *user* di dalam *group*.

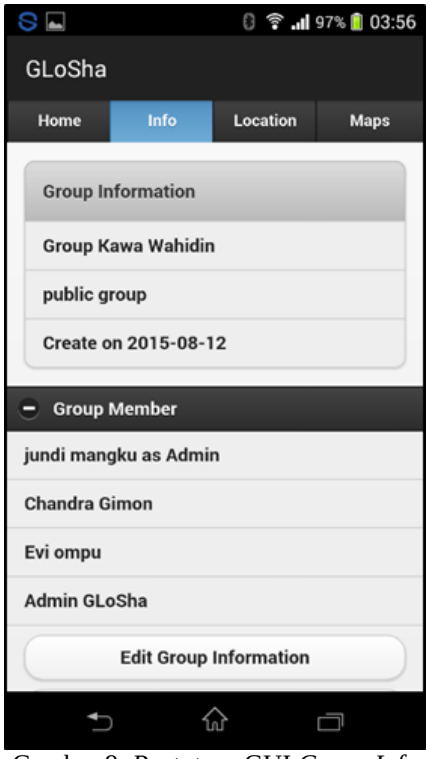
4.3. Tampilan Sistem informasi

Berikut merupakan hasil implementasi rancangan sistem informasi GloSha. Gambar 8 merupakan tampilan halaman *login* sistem informasi GloSha.



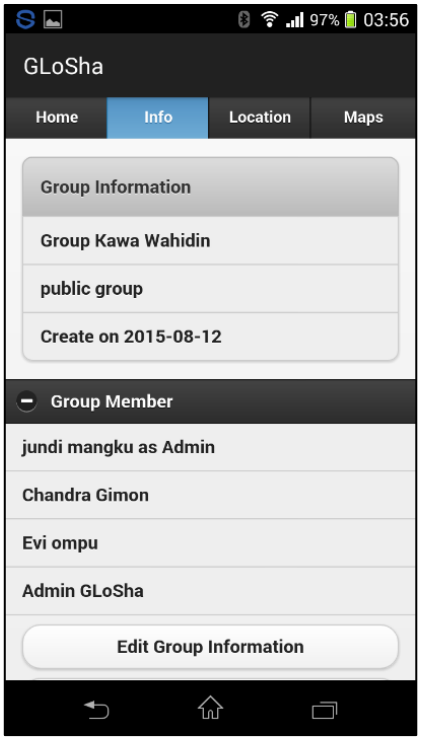
Gambar 8. *Prototype* GUI *Login*

Berikut merupakan halaman *home* dari sistem informasi.



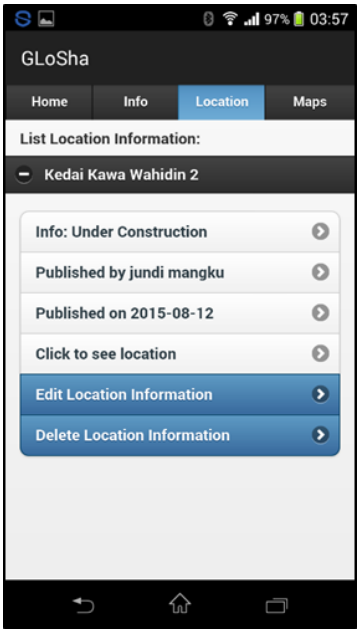
Gambar 9. *Prototype* GUI *Group Info*

Berikut merupakan halaman *group* dari sistem informasi.



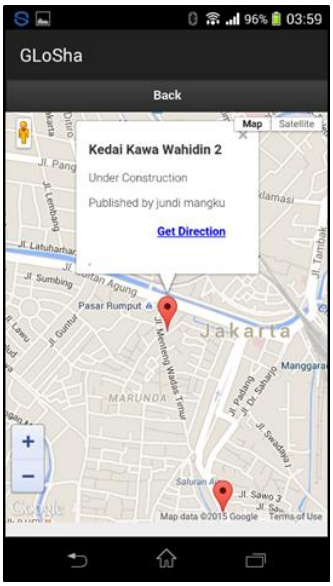
Gambar 10. *Prototype* GUI *Group Info*

Berikut merupakan halaman *list* lokasi yang telah dibagi *user* pada sistem informasi.



Gambar 11. *Prototype* GUI *List Location*

Berikut merupakan halaman lokasi dalam bentuk *map* di dalam sistem informasi.



Gambar 12. *Map View See All Location*

4.4. Pengujian Sistem informasi

Pengujian sistem informasi menggunakan teknik *integration testing* dimana komponen *software*, komponen perangkat keras, atau keduanya digabungkan dan diuji untuk mengevaluasi interaksi antara mereka. Dengan menggunakan *white box* dan *black box testing*, penguji memastikan bahwa unit bekerja bersama ketika dihubungkan ke dalam *code base* yang lebih besar [14].

Dari hasil pengujian *white box* pada algoritma sistem informasi dengan jumlah pengujian sebanyak tujuh *test case* didapatkan bahwa alogoritma barjalan sesuai dengan yang diharapkan. Pada pengujian *white box* fungsi sistem informasi dengan jumlah *test case* sebanyak 83 buah juga didapatkan bahwa fungsi sistem informasi berjalan dengan benar.

Selanjutnya dilakukan pengujian *black box* yang dilakukan oleh lima *potensial user* dimana masing-masing melakukan 45 *test case*, didapatkan bahwa fungsi sistem informasi juga telah berjalan dengan benar.

4.5. *Aplication Implementation and Maintenance*

Setelah sistem informasi GLoSha selesai dibangun dan diuji, maka sistem informasi GloSha native android akan disebarkan kepada *potential user*. Namun untuk sistem informasi GloSha bagian *web* nya masih menggunakan *hosting local* sebagai *server* sehingga jangkauan sistem informasi masih terbatas. Setelah itu diperlukan pemeliharaan dan *update* sistem untuk memastikan bahwa sistem dapat dijalankan sesuai dengan kebutuhan.

5. KESIMPULAN

Berdasarkan penelitian yang telah dilakukan, dapat disimpulkan sebagai berikut:

a. Dari hasil pengujian menunjukkan bahwa seluruh kode program dan kebutuhan fungsional sistem berjalan dengan benar.
b. Pada *module register*, *password* akan diolah menggunakan algoritma scrypt dan kemudian hasil olahan tersebut akan disimpan ke dalam *database* dalam bentuk parameter algoritma dan hasil encrypt.
c. Pada modul *login password* yang dimasukkan akan diproses dengan algoritma scrypt kemudian dicocokan dengan data *password* yang tersimpan dalam *database* untuk otentifikasi *user.*

REFERENSI

[1] D. J. Bernstein. 2005. "*The Salsa20 core*." [Online]. Available: http://cr.yp.to/salsa20.html (Diakses 9 Juni 2015).

[2] United Nations. “Preface,” dalam *Global Navigation Satellite Systems*, New York, United Nations Office, 2012, p. iii.

[3] H. Krawczyk, M. Bellare dan R. Canetti. “HMAC: Keyed-Hashing for Message Authentication,” February 1997. [Online]. Available: https://tools.ietf.org/html/rfc2104 (Diakses 6 July 2015).

[4] M. Meier. 2014. “T*he Importance of Hashing Passwords, Part 4: The Hardware Threat*”. [Online]. Available: http://www.pointsoftware.ch/en/the-importance-of-hashing-passwords-part-4-the-hardware-threat/ (Diakses 14 June 2015).

[5] R. S. Pressman. 2010. “S*oftware Engineering*,” dalam *A Practicioner's Approach*, 7th penyunt., New York: McGraw-Hill, pp. 43-44.

[6] G. Svennerberg, 2010. *Beginning Google Maps API 3*. New York: Paul Manning, 2010, p. 1.

[7] L. Williams. 2006. *Testing Overview And Black-Box Testing Techniques*. *williams2006testing.*

[8] M. Durmuth dan T. Kranz. 2014. *On Password Guessing with GPUs and FPGAs*. *Horst Gortz Institute for IT-Security:* pp. 1-2.

[9] Tarsnap. 2009. “The scrypt Key Derivation Function”. [Online]. Available: http://www.tarsnap.com/scrypt.html (Diakses 4 Juni 2015).

[10] S. S. Saputro. 2013. Perancangan Aplikasi GIS Pencarian Rute Terpendek Peta Wisata di Kota Manado Berbasis Mobile Web dengan Algoritma DIJKSTRA. Manado:pp. 1-15,

[11] D. Cosgrove. 2013. *SmartPark. Cal Poly:* pp. 4-19.

[12] C. Raj dan P. Chandra, "Global Navigation Satellite Systems (GNSS)," *International Journal of Research (IJR),* vol. 1, pp. 1440-1441, Oktober 2014.

[13] Z. Queal. 2014. *Necessary Implementation of Adjustable Work Factor Chipers in Modern Cryptographic Algorithms as it Relates to HeartBleed and OpenSSL*. *American Public University:*pp. 1-3.

[14] PECL, "HOME: What is PECL?," 6 Maret 2006. [Online]. Available: https://pecl.php.net/ (Diakses 8 Agustus 2015).