Pengembangan Modul Autentikasi *Captcha* Berbasis Gambar dengan Algoritma *Flood Fill*

Jessika Wandapranata[1)], Seng Hansun[2)]

[1)2)]*Program Studi Teknik Informatika, Fakultas Teknik dan Informatika,*
Universitas Multimedia Nusantara
Jl. Scientia Boulevard, Gading Serpong, Tangerang, Banten-15811 Indonesia
[1)]jessika.wandapranata@student.umn.ac.id
[2)]hansun@umn.ac.id

Abstrak— ***Captcha* merupakan salah satu cara memberikan akses internet yang aman dari serangan *bots*. Captcha (*Completely Automated Public Turing Test To Tell Computers and Humans Apart*) merupakan sebuah *challenge response test* yang digunakan untuk membedakan manusia dan robot (*bots*). Namun seiring perkembangan waktu, telah berkembang serangan otomatis yang berhasil membobol beberapa jenis *Captcha*. Oleh karena itu, pada penelitian ini dikembangkan *Captcha* berbasis gambar dengan metode autentikasi yang berbeda. Autentikasi *Captcha* dilakukan dengan menerapkan algoritma *Flood Fill*. Dalam implementasinya, dapat disimpulkan bahwa algoritma *Flood Fill* berhasil diimplementasikan pada autentikasi modul *Capthca* berbasis gambar. Telah dilakukan juga pengumpulan sampel data dan didapatkan hasil 77.4% responden merekomendasikan penggunaan *Captcha* yang telah dibangun.**

Kata Kunci— **Autentikasi, *Captcha*, *Captcha* Berbasis Gambar, *Flood Fill***

Abstract— **Captcha is one of many methods to secure internet from bots attack. Captcha (Completely Automated Public Turing Test To Tell Computers and Humans Apart) is a challenge response test which used to differentiate human from bots. However after the times flow, some hackers had successfully made an automatic bots to break captcha. Therefore, in this particular research, formed an imaged based captcha that uses different method of authentication. Flood Fill algorithm will be used as authentication method. In the implementation, Flood Fill algorithm has been successfully implemented as an authentication method in image based captcha. After that, data samples have been collected and the result shows 77.4% respondents agreed to recommend using the Captcha that has been developed.**

Keywords— **Authentication, Captcha, Image-based Captcha, Flood Fill**

Article history:
Received 6 October 2016; Received in revised form 14 November 2016; Accepted 15 December 2016;
Available online 28 April 2017

I. PENDAHULUAN

Banyak serangan yang terjadi dalam internet, salah satu serangan yang cukup marak ditemui adalah serangan *bots* (Saini & Bala, 2013). *bots* merupakan program komputer yang melakukan pekerjaan-pekerjaan otomatis yang bertujuan buruk seperti menghabiskan *resource* yang menyebabkan *user* tidak dapat menggunakannya, memasukkan data yang tidak valid, ataupun melakukan *spamming* (Saini & Bala, 2013) (Datta, Li, & Wang, 2005). Captcha merupakan salah satu solusi untuk mengatasi *bots*. Captcha (*Completely Automated Public Turing Test To Tell Computers and Humans Apart*) merupakan sebuah *challenge response test* yang digunakan untuk membedakan manusia dan robot (*bots*). Captcha dapat diklasifikasikan menjadi *text-based* Captcha, *image-based* Captcha, *audio-based* Captcha, dan *video-based* Captcha (Saini & Bala, 2013).

Image-based Captcha memiliki kelebihan karena *pattern recognition* merupakan sebuah permasalahan yang sulit dipecahkan oleh kecerdasan buatan sehingga sulit membobol *test* yang menggunakan teknik *pattern recoginition* dengan *bots* (Singh & Pal, 2014). Selain itu, manusia memiliki kemampuan yang lebih baik dalam mengenali bentuk dengan berbagai *noise* yang diberikan dibandingkan dengan komputer (Lin, Huang, Bell, & Lee, 2011). Berdasarkan penelitian sebelumnya, serangan otomatis terhadap *text-based* Captcha telah berhasil dilakukan sebesar 20% terhadap Google's CAPTCHA, 30-35% terhadap Microsoft's CAPTCHA, dan 35% terhadap Yahoo! CAPTCHA. Sedangkan serangan terhadap *audio-based* CAPTCHA milik Google bahkan sekitar 90% berhasil dipecahkan (Setiawan, 2012).

Berdasarkan masalah yang sudah dijelaskan, ingin dikembangkan Captcha berbasis gambar (*image-based* Captcha) dengan mekanisme autentikasi berbeda. Pada Captcha ini *user* akan diminta untuk menghubungkan titik-titik yang diberikan menjadi sebuah bentuk. Setelah itu akan dilakukan pemeriksaan kecocokan bentuk yang telah digambar *user* menggunakan algoritma Flood Fill. Flood Fill digunakan untuk mengisi daerah *pixel* dalam batasan tertentu. Pada penelitian ini, nilai-nilai *pixel* tersebut akan disimpan dalam

e-ISSN 2443-2555
doi: http://dx.doi.org/10.20473/jisebi.3.1.1-7

array untuk dilakukan perbandingan apakah nilai berada dalam batas toleransi benar.

II. TINJAUAN PUSTAKA

A. *Flood Fill*

Algoritma Flood Fill atau yang biasa disebut juga dengan algoritma *seed fill*, merupakan algoritma yang menentukan area yang terhubung dengan suatu *node* dalam *array* multidimensi (Ramadhanus, 2013). Algoritma Flood Fill umumnya diimplementasikan untuk mengisi sebagian area dengan warna tertentu atau mengisi *pixel* dari sebuah gambar dengan *value* tertentu (Harsono, 2015). Flood Fill sering digunakan dalam program pengolah gambar *bitmap* untuk mewarnai suatu daerah terbatas dengan warna tertentu seperti *Adobe Photoshop* dan *Corel Paintshop* (Nosal, 2008). (Khalili, 2006) menyatakan bahwa implementasi algoritma Flood Fill dapat ditemukan pada fitur *bucket tool* dalam piranti lunak pengolahan gambar.

Terdapat berbagai cara implementasi algoritma Flood Fill, tetapi semua menggunakan struktur data *stack* atau *queue*, baik secara eksplisit maupun implisit. Penyebaran dalam algoritma ini dapat dilakukan ke empat arah yaitu ke arah timur, selatan, barat, dan utara, maupun ke delapan arah yaitu ke semua arah mata angin, termasuk sisi diagonal (Ramadhanus, 2013). (Vandevenne, 2004) menggambarkan *pseudocode* dari algoritma Flood Fill dengan metode penyebaran delapan arah menggunakan *stack* pada Gambar 1.

```
void floodFill8Stack(int x, int y, int newColor, int oldColor)
{
    If(newColor == oldColor) return;
    emptyStack();

    static cons int dx[8] = {0, 1, 1, 1, 0, -1, -1, -1};  // relative neighbor x coordinates
    static cons int dy[8] = {-1, -1, 0, 1, 1, 1, 0, -1};  // relative neighbor y coordinates

    if(!push(x, y)) return;
    while(pop(x, y))
    {
        screenBuffer[y][x] = newColor;
        for(int i = 0; i < 8; i++) {
          int nx = x + dx[i];
          int ny = y + dy[i];
          if(nx > 0 && nx < w && ny > 0 && ny < h && screenBuffer[ny][nx] ==
          oldColor) {
            if(!push(nx, ny)) return;
          }
        }
    }
}
```

Gambar 1. Potongan *pseudocode* algoritma *Flood Fill*

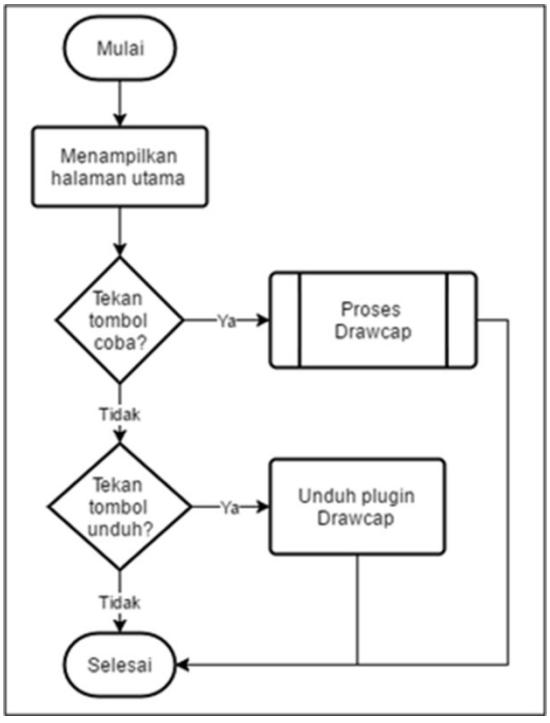
Algoritma Flood Fill menerima *input* berupa *seed pixel* (*pixel* awal dimulainya penyebaran) dan *fill color* (warna pengisi). Algoritma ini akan dimulai dari *seed pixel* lalu dilakukan pengecekan ke *pixel* tetangganya (ke empat arah maupun delapan arah) apakah *pixel* tersebut memiliki warna seperti warna *boundary*. Jika tidak, *pixel* tersebut akan diisi dengan *fill color* yang telah diberikan sebelumnya. Proses ini akan dilakukan terus-menerus secara rekursif sampai ditemukan *pixel* yang memiliki warna sesuai warna *boundary* (Anitha & Evangeline, 2013).

B. *Captcha*

(Saini & Bala, 2013) menyatakan bahwa Captcha (*Completely Automated Public Turing test to tell Computers and Humans Apart*) merupakan suatu teknik yang digunakan untuk membedakan manusia dan komputer (*bots*). Captcha mengikuti *turing test* terbalik (*reverse turing test*), pada *test* tersebut program Captcha bertindak seperti penilai dan partisipan bertindak sebagai *user*. Jika *user* dapat melewati *test* itu, maka *user* adalah manusia, jika tidak maka *user* adalah mesin. Captcha diklasifikasikan menjadi beberapa kelompok berdasarkan pada bagian apa dilakukan distorsi, apakah karakter, gambar, suara, atau video.

C. *Drawing Captcha*

Metode menggambar untuk membedakan manusia dan komputer dikembangkan dalam Rujukan (Shirali-Shahreza & Shirali-Shahreza, 2006). Metode ini bagus untuk digunakan pada perangkat yang memiliki *keyboard* kecil, atau bahkan yang tidak memiliki *keyboard*. Cara kerja metode ini adalah dengan menggambar banyak titik secara *random* di layar, beberapa dari titik tersebut dapat dibedakan dari titik-titik yang lain, misalnya dengan membuat lubang di dalam titik atau menggambar titik dengan bentuk kotak atau berlian. Setelah itu akan ditambahkan beberapa *noise* dalam gambar dan *user* akan diminta untuk menghubungkan titik-titik yang berbeda. Drawing Captcha memiliki beberapa kelebihan, diantaranya adalah tidak membutuhkan *keyboard*, bisa digunakan segala kalangan usia, dan tidak terbatas bahasa tertentu (universal).



Gambar 2. Diagram alir prosedur halaman utama

III. DESKRIPSI DAN PERANCANGAN SISTEM

A. Deskripsi Sistem

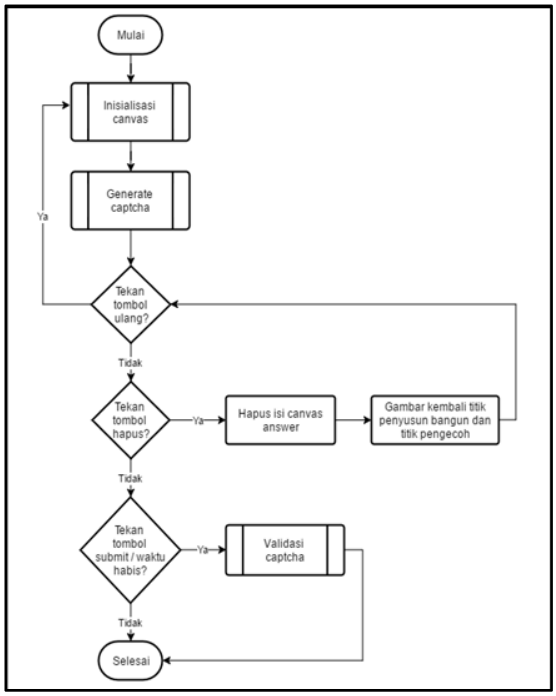
Aplikasi yang dikembangkan diberi nama Drawcap. Pada aplikasi ini *user* diminta untuk menghubungkan titik-titik menjadi sebuah bangun datar yang sesuai dengan contoh yang telah diberikan. Bangun datar yang diberikan terdiri dari bangun yang tersusun atas tiga buah titik (segitiga sembarang, segitiga sama kaki, segitiga siku-siku), empat buah titik (persegi, persegi panjang, belah ketupat, layang-layang, jajar genjang, trapesium, trapesium siku-siku, segi empat sembarang), lima buah titik (segi lima, segi lima sembarang), dan enam buah titik (segi enam, segi enam sembarang) yang dipilih secara acak. Distorsi berupa garis-garis dengan berbagai macam pola juga akan diberikan pada contoh gambar bangun.

Saat menjawab Captcha, *user* akan diberikan tambahan beberapa titik di luar titik-titik penyusun bangun. *User* harus menentukan mana titik yang harus dihubungkan untuk membentuk bangun seperti pada contoh yang diberikan pada bagian pertanyaan. Akan diberikan batasan waktu pengerjaan Captcha sesuai dengan jumlah titik penyusun bangun. Secara garis besar, aplikasi terdiri dari dua buah bagian utama, yaitu *generate* Captcha dan validasi Captcha.

B. Perancangan Sistem

Prosedur sistem akan digambarkan dalam diagram alir. Berikut diberikan beberapa diagram alir dari prosedur pengembangan sistem. Prosedur sistem dimulai dengan menampilkan halaman utama. Pada halaman ini *user* dapat mencoba mengerjakan Drawcap dengan menekan tombol Coba. Setelah menekan tombol Coba *user* akan diarahkan ke halaman yang berisi contoh formulir registrasi dan contoh penggunaan Drawcap. Jika ingin mengunduh *plugin* Drawcap, *user* dapat menekan tombol unduh pada halaman utama. Diagram alir prosedur ini dapat dilihat pada Gambar 2.

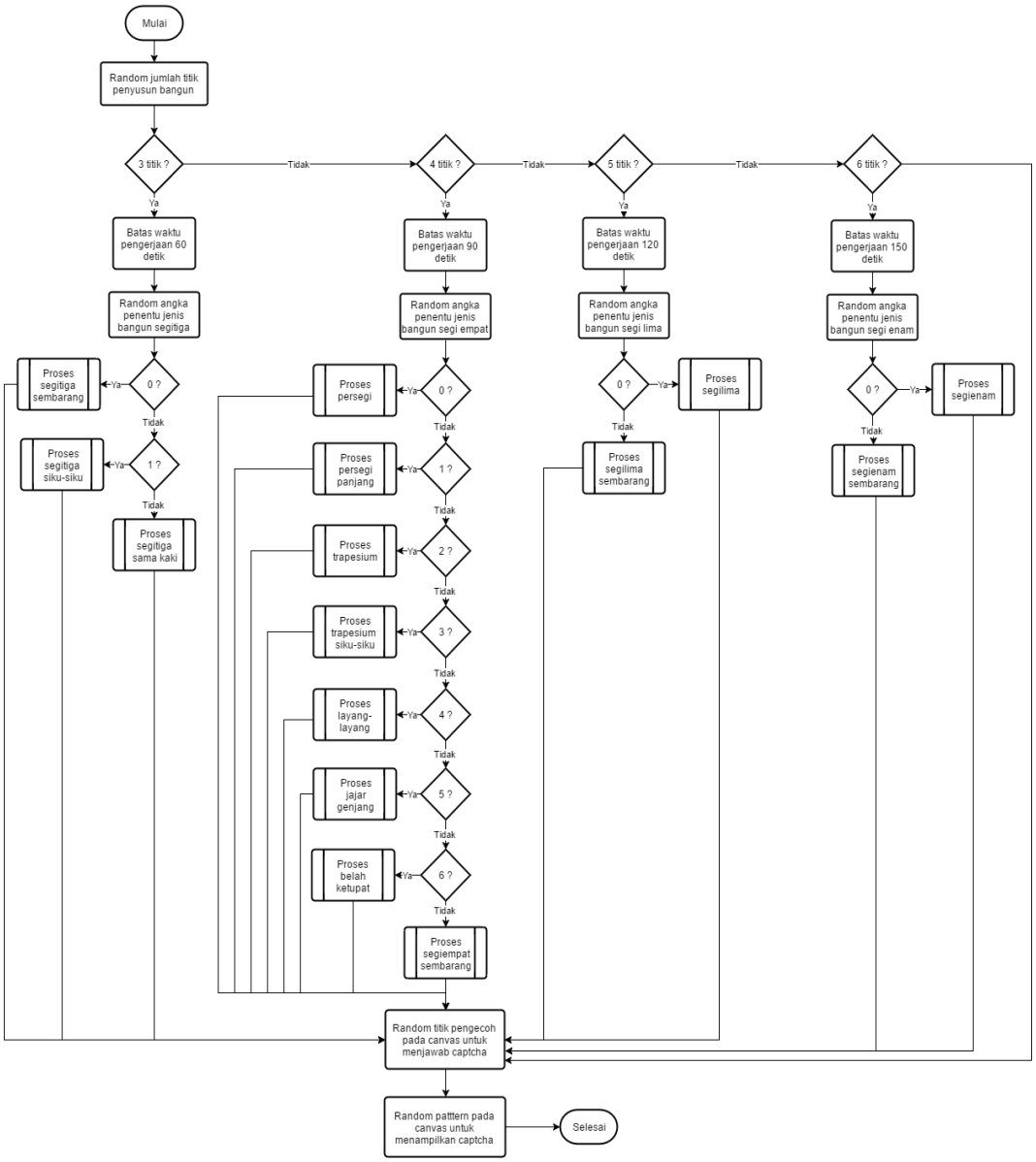
1) *Prosedur Proses Drawcap :* Prosedur proses Drawcap dimulai dari inisialisasi canvas tempat menampilkan dan menjawab Captcha. Setelah inisialisasi selesai dilakukan, akan dilanjutkan dengan proses generate Captcha secara acak. Jika user ingin mengganti Captcha, user dapat menekan tombol Ulang dan inisialisasi canvas serta proses generate Captcha akan diulang kembali untuk menghasilkan Captcha yang berbeda. Jika user mengalami kesalahan dalam menghubungkan titik, user dapat menghapus garis yang telah digambar dengan menekan tombol Hapus. Saat tombol Hapus ditekan, seluruh isi canvas untuk menjawab Captcha akan dihapus dan titik-titik penyusun bangun serta titik-titik tambahan akan digambar kembali. Jika user menekan tombol Submit atau waktu yang disediakan telah habis, akan dilakukan proses validasi Captcha. Diagram alir prosedur proses Drawcap dapat dilihat pada Gambar 3.



Gambar 3. Diagram alir prosedur proses Drawcap

2) Prosedur Generate Captcha : Prosedur generate Captcha dimulai dengan memilih secara acak sebuah angka yang digunakan untuk menentukan jumlah titik penyusun bangun Captcha. Jumlah titik penyusun bangun menentukan jumlah waktu yang diberikan untuk menyelesaikan Captcha. Setelah itu dilakukan pemilihan angka secara acak kembali untuk menentukan jenis bangun berdasarkan jumlah titik penyusun bangun. Contohnya, jika pada proses pemilihan angka penyusun bangun didapatkan angka tiga maka dilakukan pemilihan angka secara acak untuk menentukan jenis bangun segitiga. Jika didapat angka nol maka dibangun segitiga sembarang, jika didapat angka satu maka dibangun segitiga siku-siku, dan jika didapat angka dua maka dibangun segitiga sama kaki. Selanjutnya dilakukan pemilihan secara acak jumlah titik yang ditambahkan pada canvas jawaban berikut posisinya. Terakhir, dipilih garis dengan pola seperti apa yang akan ditambahkan pada canvas soal sebagai distorsi. Diagram alir prosedur generate Captcha dapat dilihat pada Gambar 4.

Prosedur Flood Fill : Prosedur Flood Fill digunakan dalam perhitungan akurasi Captcha yang dikerjakan user. Pada prosedur ini dilakukan pengecekan warna suatu pixel dalam canvas. Jika warna sama dengan warna awal, maka warna pixel



Gambar 4. Diagram alir prosedur *generate* Captcha

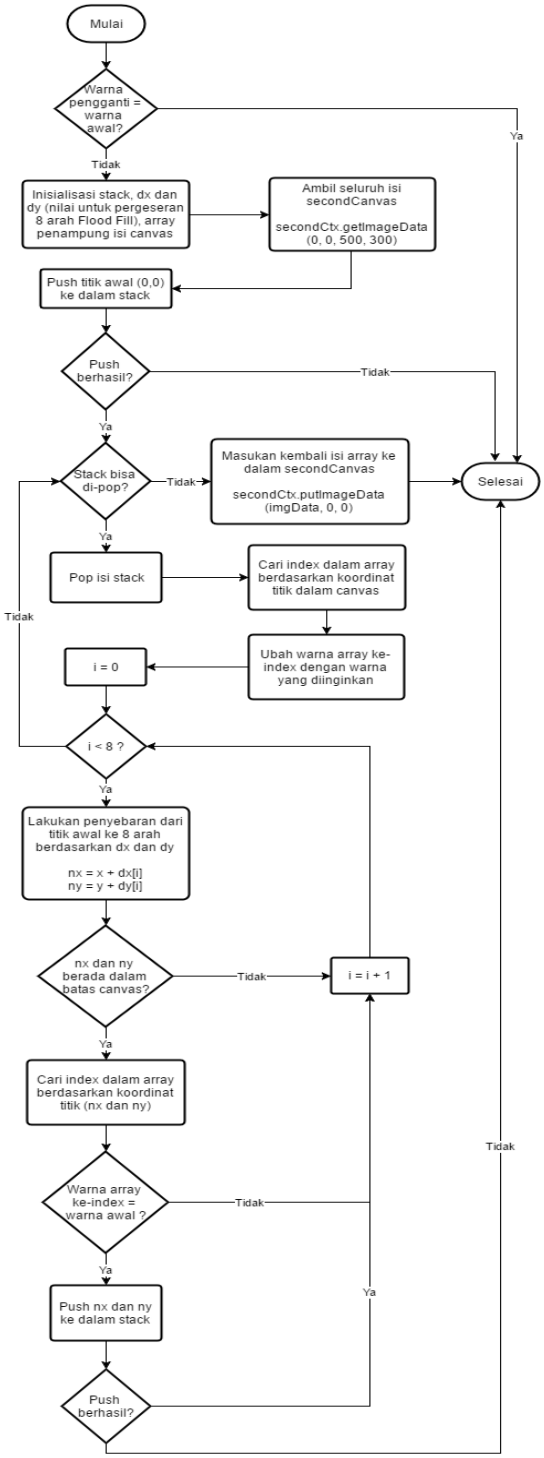
tersebut akan diganti menjadi warna pengganti yang telah ditentukan. Setelah itu, dilakukan penyebaran ke delapan arah sekitar pixel. Selanjutnya dilakukan pengecekan warna yang dimiliki daerah sekitar pixel, jika warna pada daerah tersebut berbeda dengan warna awal, langsung dilakukan pengecekan warna ke daerah sekitar lainnya. Namun jika warna sama, koordinat pixel tersebut akan di-push ke dalam stack untuk selanjutnya dilakukan perubahan warna dan penyebaran ke delapan arah sekitar pixel tersebut. Proses ini akan terus diulangi sampai isi stack kosong. Diagram alir prosedur Flood Fill dapat dilihat pada Gambar 5.

IV. IMPLEMENTASI DAN PEMBAHASAN

A. Hasil Implementasi

Modul autentikasi Captcha berbasis gambar selanjutnya dikembangkan sesuai perancangan yang telah dijelaskan pada bab sebelumnya. Gambar 6 memperlihatkan antarmuka utama aplikasi *web* dimana pengembangan modul autentikasi Captcha ini diterapkan.

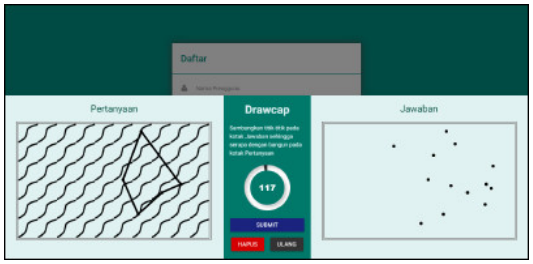
Pada halaman Coba Captcha terdapat sebuah *form* registrasi yang digunakan sebagai contoh penerapan Drawcap. Selain itu terdapat sebuah tombol yang dapat digunakan sebagai *trigger* untuk memanggil *modal* berisi Drawcap yang harus diisi. Jika tombol mulai Captcha ditekan, akan muncul sebuah *modal* yang berisi Drawcap yang harus dikerjakan. *Modal* terdiri dari tiga bagian, bagian tengah berisi instruksi pengerjaan



Gambar 5. Diagram alir prosedur Flood Fill

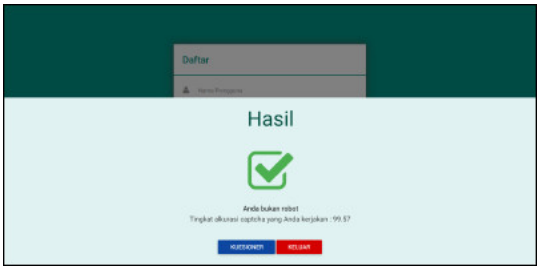
Drawcap beserta *timer* dan tiga buah tombol yaitu tombol *Submit* yang digunakan untuk memvalidasi Drawcap yang telah dikerjakan, tombol Hapus yang dapat digunakan jika *user* melakukan kesalahan dalam menggambar garis untuk menjawab Drawcap, dan tombol Ulang yang dapat digunakan untuk me-*refresh* Drawcap atau menampilkan Drawcap dengan soal berbeda. Gambar 7 menunjukkan bagian *modal* Drawcap.

Gambar 6. Laman utama Drawcap

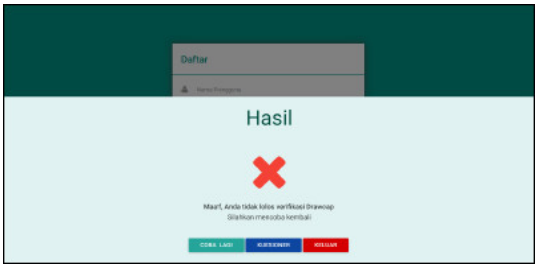


Gambar 7. Modal Drawcap pada laman coba Captcha

Setelah *user* menekan tombol *Submit* atau waktu pada *timer* habis akan dilakukan validasi Drawcap yang dikerjakan. Gambar 8 menunjukkan tampilan *modal* saat *user* berhasil lolos validasi Drawcap, sedangkan *modal* seperti Gambar 9 akan muncul jika *user* belum berhasil lolos validasi Drawcap.



Gambar 8. Tampilan saat *user* berhasil lolos Drawcap



Gambar 9. Tampilan saat *user* gagal Drawcap

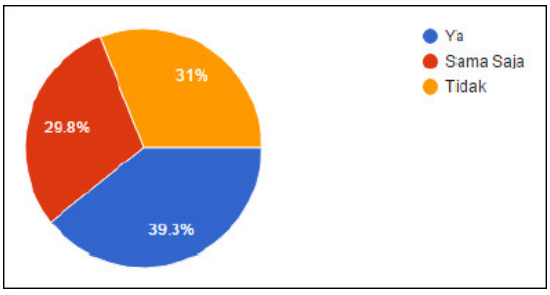
B. Pengujian Threshold dan Evaluasi Sistem

Setelah aplikasi selesai dibangun, dilakukan pengujian untuk mendapatkan *threshold* (nilai batas kesalahan) yang digunakan sebagai *threshold* validasi Captcha saat aplikasi disebarkan untuk mengumpulkan sampel data penelitian. Pengujian dilakukan dengan meminta

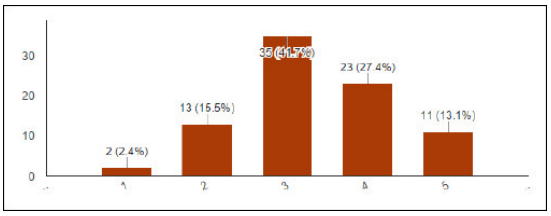
beberapa orang mengerjakan Captcha yang telah dibangun secara berulang-ulang. Captcha dikerjakan sebanyak 150 kali dan dicatat persentase perbedaan gambar *user* dengan gambar yang harus diikuti pada bagian pertanyaan. Setelah itu, dicari rata-rata persentase kesalahan dari data yang telah dikumpulkan dan didapatkan nilai 0.6 sebagai *threshold* validasi Captcha.

Setelah didapatkan nilai *threshold* untuk validasi, dilakukan pengumpulan data untuk analisis lebih lanjut dengan mengunggah aplikasi ke jaringan internet sehingga para relawan dapat melakukan pengujian aplikasi dengan lebih mudah. Analisis yang dilakukan pada sampel data bertujuan untuk mengetahui tingkat kesulitan Drawcap dan kompatibilitas atau penerimaan Drawcap sebagai Captcha berbasis gambar.

Berdasarkan pengumpulan sampel data yang telah dilakukan, didapat 84 data hasil pengisian kuesioner. Berdasarkan data 84 responden, terdapat 33 responden (39.3%) menyatakan Drawcap lebih mudah diselesaikan dibanding Captcha lain yang pernah mereka kerjakan dan perhitungan Skala Likert menghasilkan nilai 66.67% sebagai tingkat kesulitan rata-rata Drawcap. Gambar 10 dan 11 memperlihatkan diagram lingkaran tingkat kesulitan Drawcap dibanding Captcha lainnya dan diagram batang tingkat kesulitan rata-rata Drawcap secara umum berdasarkan responden.

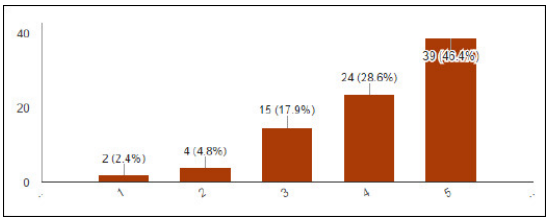


Gambar 10. Tingkat kesulitan Drawcap dibanding Captcha lainnya

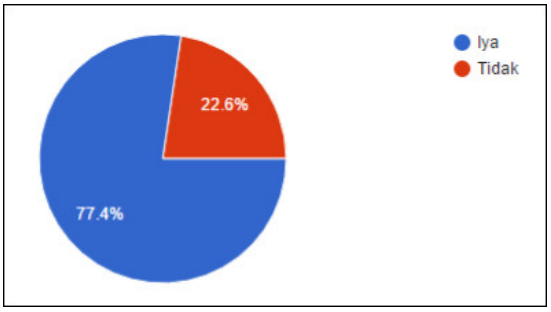


Gambar 11. Tingkat Kesulitan Drawcap

Walaupun Drawcap cukup sulit untuk diselesaikan, perhitungan Skala Likert menghasilkan nilai 82.38% sebagai tingkat keamanan rata-rata Drawcap menurut responden, yang ditunjukkan oleh Gambar 12. Selain itu, 77.4% responden merekomendasikan penggunaan Drawcap dibanding Captcha lainnya sebagaimana yang diperlihatkan oleh Gambar 13.



Gambar 12. Tingkat Keamanan Drawcap



Gambar 13. Rekomendasi Drawcap

V. KESIMPULAN

Implementasi algoritma Flood Fill pada autentikasi modul Captcha berbasis gambar telah berhasil dilakukan. Pada implementasi ini, *user* dapat mencoba mengerjakan Captcha yang telah dibangun dan menggunakannya sebagai *plugin*. Hasil analisis data yang telah dilakukan terhadap 84 responden menunjukkan bahwa Drawcap memiliki tingkat kesulitan rata-rata 66.67% dan memiliki tingkat keamanan rata-rata sebesar 82,38%. Namun demikian, Drawcap direkomendasikan penggunaannya oleh 77.4% responden mengingat fitur yang ditawarkannya. Pengembangan selanjutnya dapat dilakukan dengan menambah variasi bangun Captcha, meningkatkan cara menghubungkan titik menjadi garis pada perangkat *touch screen*, dan mengembangkan cara mengenali bentuk bangun, seperti menggunakan deteksi tepi atau fungsi pengolahan gambar lainnya.

DAFTAR PUSTAKA

Anitha, S., & Evangeline, D. (2013). An Efficient Fence Fill Algorithm using Inside-Outside Test. *International Journal of Advanced Research in Computer science and Software Engineering , 5* (3), 605-609.

Datta, R., Li, J., & Wang, J. (2005). IMAGINATION: A Robust Image-base CAPTCHA Generation System. *Proceedings of the ACM Multimedia Conference* (pp. 331-334). Singapore: ACM, The Pennsylvania State University.

Harsono, E. (2015). *Implementasi Algoritma Flood Fill pada Aplikasi Ishihara Plate Image Generator.* Skripsi, Universitas Multimedia Nusantara, Teknik Informatika.

Khalili, B. (2006). *A hybrid format for storing raster images.* University of Colorado Boulder, Computer Science.

Lin, R., Huang, S., Bell, G., & Lee, Y. (2011). A New CAPTCHA Interface Design for Mobile Device. *AUIC '11 Proceedings of the Twelfth Australasian User Interface Conference* (pp. 3-8). Perth: Australian Computer Society.

Nosal, E. (2008). Flood-fill algorithms used for passive account detection and tracking. *2008 New Trends for Environmental Monitoring Using Passive Systems* (pp. 1-5). Hyeres: IEEE.

Ramadhanus, A. (2013). *Penerapan Algoritma Flood Fill untuk Mengurangi Ruang Pencarian pada Pencarian Solui Puzzle Pentomino.* Makalah, Institut Teknologi Bandung, Program Studi Teknik Informatika.

Saini, B., & Bala, A. (2013). A Review of Bot Protection using CAPTCHA for Web Security. *IOSR Journal of Computer Engineering , 8* (6), 36-42.

Setiawan, E. (2012). Optimalisasi Keamanan Website Menggunakan Captcha – Ad Video. *Jurnal Ilmiah Komputer dan Infomatika , 1* (1), 1-6.

Shirali-Shahreza, M., & Shirali-Shahreza, S. (2006). Drawing Captcha. *28th International Conference on Information Technology Interfaces* (pp. 475-480). Cavtat: IEEE.

Singh, V., & Pal, P. (2014). Survey of Different Types of CAPTCHA. *International Journal of Computer Science and Information Technologies , 5* (2), 2242-2245.

Vandevenne, L. (2004). *Lode's Computer Graphics Tutorial : Flood Fill*. Retrieved February 27, 2016, from http://lodev.org/cgtutor/floodfill.html