BAB II

LANDASAN TEORI

2.1. Captcha

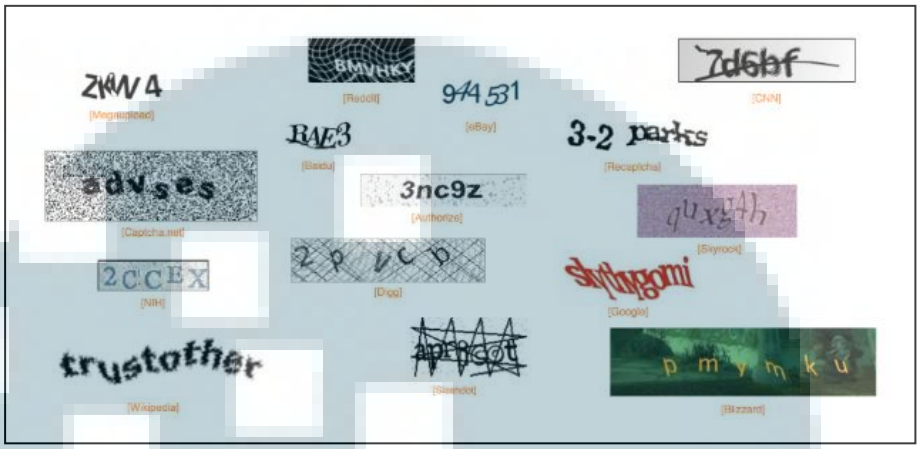
Menurut Saini dan Bala (2013), Captcha (*Completely Automated Public Turing test to tell Computers and Humans Apart*) merupakan suatu teknik yang digunakan untuk membedakan manusia dan komputer (*bots*). Captcha diciptakan oleh Luis Von Ahn, Manuel Blum, Nicholas J. Hooper dan John Langford pada tahun 2000. 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 juga bisa disebut sebagai sistem pertahanan untuk mencegah *web bots* melakukan penyalahgunaan dalam *online service* di internet. Zhu dkk. (2010) menyatakan bahwa tingkat keberhasilan *bot* melakukan penyerangan terhadap Captcha tanpa bantuan manusia tidak boleh melebihi 0.6%.

Captcha diklasifikasikan menjadi beberapa kelompok berdasarkan pada bagian apa dilakukan distorsi, apakah karakter, gambar, suara, atau video. Berikut beberapa klasifikasi Captcha menurut Saini dan Bala (2013).

1. Berbasis *Text*

   Captcha berbasis *text* merupakan yang paling sering digunakan dan cukup mudah untuk diimplementasikan. Captcha berbasis *text* membutuhkan bank soal yang besar. Dalam Captcha berbasis *text*, *user* akan diberikan rangkaian karakter yang dapat terdiri dari huruf maupun angka yang telah

diberikan berbagai efek seperti diubah ukurannya, diberikan garis, dibuat saling berdempetan dengan karakter lainnya, diubah kemiringannya, dan beberapa cara lainnya agar sulit dibaca oleh *bots*.



Gambar 2.1 Contoh Captcha Berbasis *Text*
(Lee, 2011)

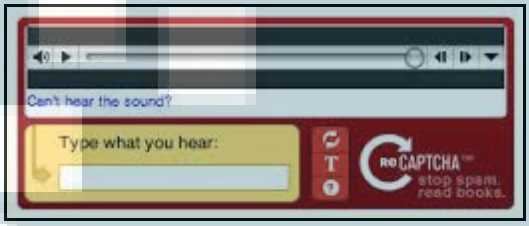
2. Berbasis Gambar

Ada beberapa implementasi dalam Captcha berbasis gambar, salah satunya adalah saat *user* diberikan beberapa gambar dan diharuskan memilih gambar yang sesuai dengan instruksi yang diberikan, misalnya memilih gambar kucing, atau memilih gambar yang kategorinya berbeda dari gambar lain.

Gambar 2.2 Contoh Captcha Berbasis Gambar
(Ivanoff, 2014)

3. Berbasis audio

Captcha berbasis audio dikembangkan untuk *user* yang memiliki disabilitas secara visual (tidak dapat melihat). Untuk menggunakan Captcha ini, *user* akan mendengarkan sebuah *audio-clip* lalu mengetik apa yang didengar pada tempat yang telah tersedia. Untuk menilai apakah *user* benar-benar manusia, digunakan *sound-based system* yang akan memilih rangkaian digit dan huruf secara acak dan me-*render*-nya menjadi *sound clip* lalu melakukan distorsi terhadap *clip* tersebut.



Gambar 2.3 Contoh Captcha Berbasis Audio
(Timmer, 2008)

4. Berbasis video

Dalam Captcha berbasis video, *user* akan diberikan sebuah video yang di dalamnya terdapat jawaban dari pertanyaan yang akan diberikan. Selain itu, Captcha jenis ini juga bisa dibuat dengan memberikan beberapa kata yang merupakan kategori atau *tags* dari video yang ditampilkan. Jika *user* dapat memilih dengan benar maka *user* adalah manusia.

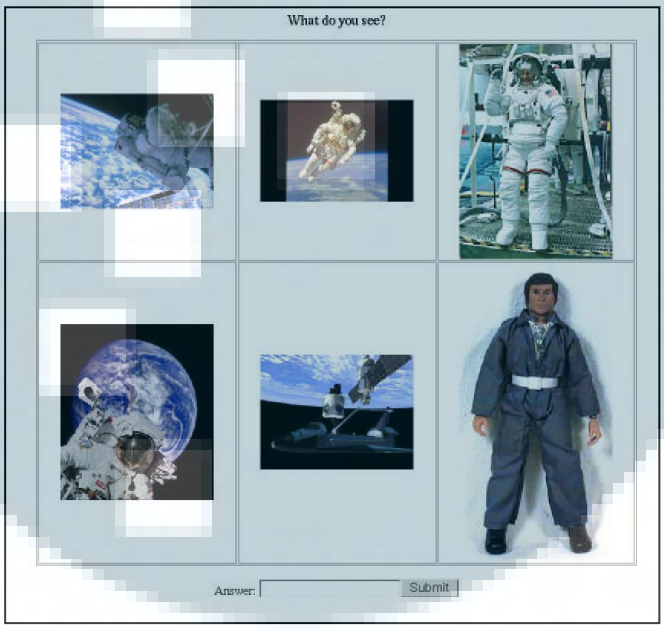
Gambar 2.4 Contoh Capthca Berbasis Video
(McCullagh, 2012)

2.1.1. Image Based Captcha

Pada penelitian sebelumnya yang dilakukan oleh Chew dan Tygar (2004), terdapat beberapa Captcha berbasis gambar yang dibangun menggunakan *image recognition* diantaranya sebagai berikut.

A. Naming Images Captcha

Diberikan enam buah gambar dan *user* akan diminta memasukkan kata yang berhubungan dengan gambar-gambar tersebut.



Gambar 2.5 Contoh *Naming Image* Captcha
(Chew dan Tygar, Image Recognition Captchas, 2004)

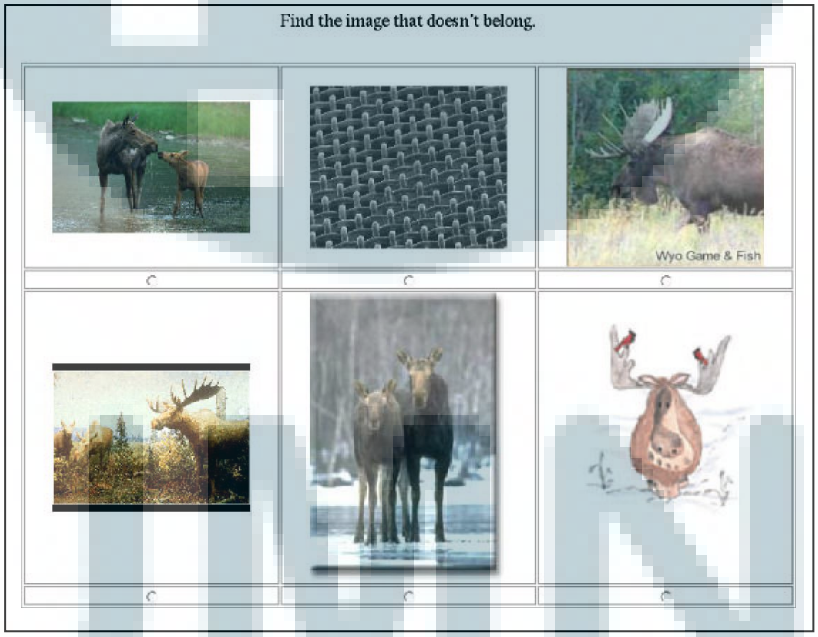
B. Distinguish Images Captcha

Diberikan dua set *image* kepada *user*, masing-masing *image* terdiri dari tiga buah gambar dalam *subject* yang sama. Dengan probabilitas yang sama, kedua set gambar bisa berhubungan ataupun tidak. *User* akan diminta menentukan apakah kedua set gambar tersebut berhubungan atau tidak.

Gambar 2.6 Contoh *Distinguish Image* Captcha
(Chew dan Tygar, Image Recognition Captchas, 2004)

C. Identifying Anomalies Captcha

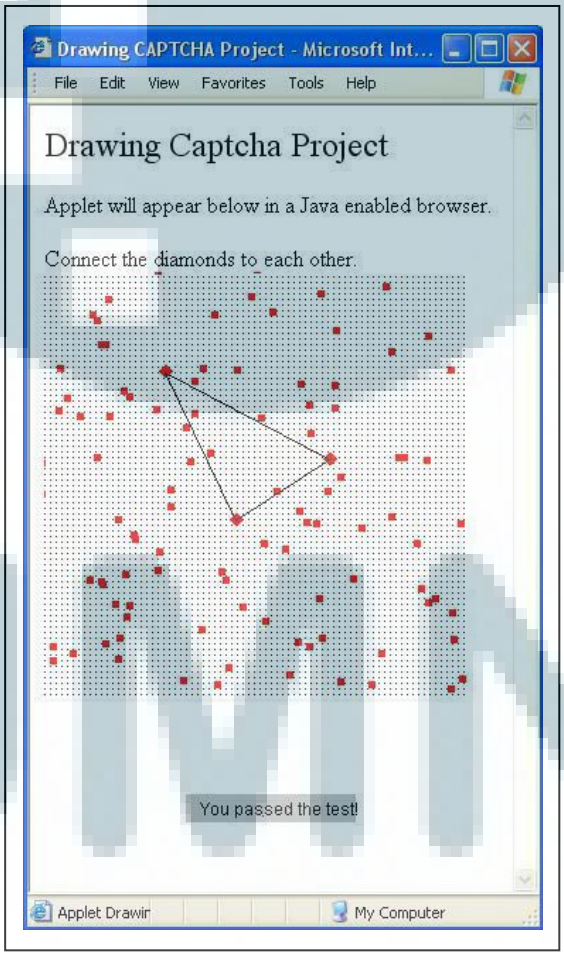
Diberikan enam buah gambar kepada *user* dan *user* diminta memilih salah satu gambar yang tidak berhubungan dengan gambar-gambar lainnya.



Gambar 2.7 Contoh *Identifying Anomalies* Captcha
(Chew dan Tygar, Image Recognition Captchas, 2004)

2.1.2. Drawing Captcha

Mohammad Shirali-Shahreza dan Sajad Shirali-Shahreza (2006) dalam penelitiannya mengembangkan metode menggambar untuk membedakan manusia dan komputer. 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.



Gambar 2.8 Contoh *Drawing* Captcha
(Shirali-Shahreza dan Shirali-Shahreza, Drawing Captcha, 2006)

Menurut Mohammad Shirali-Shahreza dan Sajad Shirali-Shahreza (2006), *Drawing* Captcha memiliki beberapa kelebihan, diantaranya sebagai berikut.

1. Tidak membutuhkan *keyboard*.
2. Bisa digunakan semua kalangan usia, termaksud anak-anak.
3. Dapat digunakan *user* dengan bahasa yang berbeda-beda (tidak terbatas bahasa atau universal).
4. Tidak membutuhkan banyak proses sehingga bisa dijalankan pada perangkat kecil dan mesin dengan *resource* terbatas.
5. Ukuran Captcha kecil (tidak menggunakan gambar dengan resolusi tinggi) sehingga tidak membutuhkan *bandwidth* besar dan waktu yang lama untuk men-*download* Captcha dari *website*.
6. *Client-side program*, sehingga setelah satu kali men-*download* program, tidak butuh *download* ulang atau koneksi ulang ke *server*.

2.2. Algoritma 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. Pada dasarnya Flood Fill merupakan turunan dari algoritma DFS (*Depth First Search*) pada kasus khusus, yaitu DFS pada *array* multidimensi (Ramadhanus, 2013). Menurut Harsono (2015), algoritma Flood Fill umumnya diimplementasikan untuk mengisi sebagian area tertentu dengan warna tertentu atau mengisi *pixel* dari sebuah gambar dengan *value* tertentu.

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) mengatakan bahwa

implementasi algoritma Flood Fill dapat ditemukan pada fitur *bucket tool* dalam piranti lunak pengolahan gambar.

Menurut Ramadhanus (2013), 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).

Gambar 2.9 Penyebaran *Node* dalam Algoritma Flood Fill
(Anitha dan Evangeline, 2013)

Berikut adalah *pseudocode* dari algoritma Flood Fill dengan metode penyebaran delapan arah menggunakan *stack* menurut Vandevenne dalam http://lodev.org/cgtutor/floodfill.html.

```
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 2.10 Potongan *Pseudocode* Algoritma Flood Fill (Vandevenne, 2004)

Menurut Anitha dan Evangeline (2013), 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*.

2.3. HTML5

HTML5 merupakan sebuah *markup language* yang digunakan untuk menata dan menyajikan konten *World Wide Web* (www). Selain itu HTML5 juga merupakan teknologi inti internet yang awalnya diusulkan oleh Opera Software

(Sharma dkk., 2012). Menurut Trivedi dan Harjena (2014), HTML5 bukan merupakan sebuah bahasa atau *development tool* baru, tetapi adalah HTML dengan tambahan *layer* yang berisi *tags* terstandardisasi dan atribut untuk grafis dan efek visual yang mengurangi kebutuhan akan *special plug-ins*.

HTML5 merupakan kerja sama antara World Wide Web Consortium (W3C) dan Web Hypertext Application Technology Working Group (WHATWG). WHATWG mengerjakan *web forms* dan aplikasi, sedangkan W3C mengerjakan XHTML 2.0 (Sharma dkk., 2012). Seenivasan (2012) menyatakan bahwa HTML5 diminati karena banyaknya fitur-fitur baru yang disediakan, beberapa di antaranya adalah *native control* untuk video dan audio menggunakan elemen <video> dan <audio>, *2D Canvas Animation API* yang menyediakan kemampuan menggambar dua dimensi, *web storage* yang menyediakan penyimpanan lokal pada komputer, dan *Offline Access* (*AppCache*) yang menyediakan cara baru untuk menggunakan *web site* atau *web application* tanpa koneksi internet. Pada penelitian ini, akan digunakan fitur *canvas* sebagai tempat menggambar Captcha.

2.4. JavaScript

Menurut Jensen dkk. (2009), JavaScript adalah *main scripting language* untuk *web browser* dan penting untuk aplikasi *web* modern. Berdasarkan Kereki (2015), JavaScript awalnya dikembangkan oleh Netscape pada tahun 1995. Netscape mengumumkan bahwa JavaScript merupakan sebuah *object scripting language* yang mudah digunakan dan didesain untuk membuat *live online application* yang menghubungkan *objects* dan *resources* antara *client* dan *server*. JavaScript awalnya bernama Mocha, lalu berubah menjadi LiveScript. LiveScript

berubah nama menjadi JavaScript setelah dikelola oleh Mozilla Foundation. JavaScript merupakan *object oriented language* yang menggunakan objek-objek *prototype* untuk memodelkan pewarisan sifat (*inheritance*).

2.5. jQuery

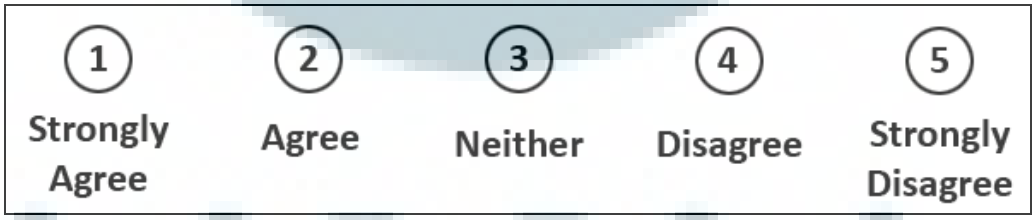
jQuery adalah sebuah *library* JavaScript yang memiliki prinsip "*write less, do more*". jQuery dapat mengubah suatu proses yang awalnya terlihat sulit dan memakan waktu menggunakan JavaScript menjadi mudah dan sederhana (Kumar, 2012). Beighley (2010) dalam bukunya yang berjudul jQuery For Dummies mengatakan bahwa jQuery adalah sebuah *add-on library* untuk JavaScript. jQuery dapat dianggap sebagai kode JavaScript yang telah dituliskan dan dapat digunakan dengan memasukkan satu atau dua baris kode untuk memanggil fungsi yang diinginkan.

Menurut Kumar (2012), jQuery adalah suatu *library* yang cepat dan ringkas yang dapat digunakan untuk HTML *document traversing*, *event handling*, *animating*, dan Ajax *interactions*. Fitur-fitur dari jQuery di antaranya adalah DOM *element selector*, *traversal*, *and modification*, *event handling* (*Browser events*, *Mouse events*, *Form events*), manipulasi tampilan, efek dan animasi, integrasi Ajax, *extensibility*, *utilities*, dan JavaScript *Plugins*. Semua fitur yang dimiliki membuat jQuery sangat populer dan menjadi teknologi yang dipelajari banyak pengembang aplikasi. Sudah menjadi sebuah standar industri bahwa jQuery digunakan untuk membuat aplikasi *web* yang kaya dan hampir semua perusahaan aplikasi *web* utama seperti Google, Yahoo, Digg, WordPress dan Drupal menggunakan jQuery dalam aplikasinya.

2.6. Skala Likert

Menurut Bertram (2013), Skala Likert merupakan sebuah skala respon psikometri (*psychometric response scale*) yang banyak digunakan dalam kuesioner untuk mendapatkan preferensi atau tingkat persetujuan responden terhadap sebuah atau sekumpulan pernyataan. Skala Likert merupakan sebuah teknik pengukuran non-komparatif dan bersifat *undimensional* (hanya mengukur satu sifat). Responden diminta untuk mengindikasikan tingkat persetujuan mereka terhadap pernyataan yang diberikan dalam skala ordinal.

Budiaji (2013) menyatakan bahwa kategori respon pada Skala Likert mempunyai tingkatan, tetapi jarak antar kategori tidak dapat dianggap sama, sehingga Skala Likert adalah kelas skala ordinal. Menurut Bertram (2013), Skala Likert memiliki beberapa kelebihan, di antaranya adalah mudah dibangun dan mudah dibaca serta diselesaikan oleh para partisipan. Contoh sampel skala yang digunakan dalam Skala Likert dapat dilihat pada Gambar 2.11 di bawah ini.



Gambar 2.11 Sampel Skala yang digunakan dalam Skala Likert (Bertram, 2013)

Contoh penerapan perhitungan secara manual dan analisa hasil kuesioner menggunakan Skala Likert diberikan oleh Handayani (2014). Contoh diberikan berdasarkan penelitian mengenai analisa kualitas perangkat lunak aplikasi pendataan pendidikan dasar menggunakan ISO/IEC 9126. Dalam penelitiannya Handayani menggunakan 4 titik respon yaitu sangat tidak setuju (1), tidak setuju

(2), setuju (3) dan sangat setuju (4). Wawancara dilakukan terhadap 13 responden dan diperoleh hasil pengamatan sebagai berikut.

Tabel 2.1 Data Hasil Pengamatan
(Handayani, 2014)

Alternatif Jawaban	Jumlah	Persentase
Sangat Setuju	2	15,40%
Setuju	4	30,80%
Tidak Setuju	7	53,80%
Sangat Tidak Setuju	0	0%
Total	13 orang	100%

Handayani (2014) berpendapat bahwa perhitungan hasil wawancara dapat dilakukan secara manual maupun menggunakan aplikasi. Berikut diberikan cara perhitungan skor secara manual menggunakan Skala Likert.

1. Terdapat 2 orang menjawab Sangat Setuju (4) : 2 * 4 = 8
2. Terdapat 4 orang menjawab Setuju (3) : 4 * 3 = 12
3. Terdapat 7 orang menjawab Tidak Setuju (2) : 7 * 2 = 14
4. Tidak ada orang yang menjawab Sangat Tidak Setuju (1) : 0 * 1 = 0
5. Didapat jumlah skor 34

Berikut merupakan jumlah skor ideal untuk pertanyaan yang diajukan kepada responden.

1. Skor tertinggi : 13 * 4 = 52 (Sangat Setuju)
2. Skor terendah : 13 * 1 = 13 (Sangat Tidak Setuju)

Berdasarkan perhitungan di atas, didapatkan interpretasi skor hasil pengamatan sebesar 65,38% yang didapat dari perhitungan (34/52) * 100%.