Implementasi Kriptografi Visual Temporal Sebagai CAPTCHA

M. Hanief Fatkhan Nashrullah - 13522100
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
hanieffatkhan@gmail.com, 13522100@std.stei.itb.ac.id

***Abstrak*—Penggunaan bot otomatis di internet seringkali merugikan penyedia layanan, sehingga diperlukan CAPTCHA untuk membedakannya dengan manusia. CAPTCHA konvensional saat ini semakin mudah ditembus oleh AI, sedangkan solusi kriptografi visual yang sudah ada sebelumnya cukup sulit digunakan karena pengguna harus menyatukan gambar secara manual. Makalah ini mengusulkan metode CAPTCHA menggunakan kriptografi visual berbasis temporal yang memanfaatkan *persistence of vision*. Dengan menampilkan gambar terenkripsi secara bergantian dengan cepat, pengguna bisa langsung membaca teks tanpa harus menggeser gambar. Sistem ini dibuat menggunakan teknologi web sederhana. Hasil pengujian menunjukkan bahwa sistem ini aman dari serangan screenshot biasa. Meskipun bot bisa membaca teks dengan cara merekam dan mengolah video, cara ini membutuhkan proses komputasi yang lebih rumit dibandingkan menembus CAPTCHA biasa.**

Kata Kunci—CAPTCHA, Integrasi Temporal, Keamanan Web, Kriptografi Visual, Persistence of Vision.

I. Pendahuluan

Pada zaman ini, internet sudah menjadi pusat media distribusi dari informasi. Informasi yang ingin dicari kemungkinan besar sudah ada di internet. Hanya dengan membuka *web browser* dan mengakses *search engine* ataupun LLM, informasi yang ingin dicari dapat ditemukan di internet. Selain sebagai media untuk mendapatkan informasi, internet juga dapat digunakan sebagai media penyebaran informasi. Namun, di balik mudahnya distribusi internet terdapat suatu masalah yaitu *automated task* atau *bot*.

Penggunaan *bot* pada internet dapat menguntungkan pengguna *bot* tersebut. Misalnya, pengguna dapat melakukan penyebaran informasi secara masif dan cepat tanpa perlu mengirim informasi secara manual satu per satu. Selain itu, pengguna *bot* juga dapat mengambil informasi secara otomatis dari internet. Hal ini sekilas tampak menguntungkan, tetapi bagi penyedia layanan hal ini adalah hal yang merugikan. Informasi yang disebarkan oleh *bot* kemungkinan berisi konten yang tak pantas. Selain itu pengambilan informasi secara masif dan terotomasi juga berpotensi menimbulkan masalah pada perlindungan data pribadi. Terakhir *automated bot* juga umumnya menghabiskan *resource* seperti *bandwidth*, CPU, RAM, dan *storage* dari penyedia jasa. Maka dari itu diperlukan sebuah cara untuk mencegah *bot* untuk mengakses *resource* yang ada pada internet.

Saat ini, penyedia layanan di internet umumnya menggunakan CAPTCHA untuk mencegah *resource* mereka diakses oleh *automated bot*. CAPTCHA merupakan sebuah tes untuk menguji pengakses *resource* apakah pengakses adalah manusia atau bukan. Pada umumnya, CAPTCHA yang digunakan adalah CAPTCHA berbasis identifikasi gambar. Pengakses *resource* akan diberikan sebuah tes terkait gambar kemudian pengakses harus bisa melakukan identifikasi dan menyelesaikan tes yang diberikan. Namun, saat ini sudah banyak teknologi yang dapat melakukan identifikasi gambar. Meskipun tidak banyak orang yang memberikan *source code* cara melakukan *bypass* dari CAPTCHA, orang yang berpengalaman dalam pengembangan *software* dapat membuat *bypass* dari CAPTCHA. Sebelumnya terdapat solusi oleh Khandar William pada tahun 2009 yang menggunakan kriptografi visual untuk menyelesaikan permasalahan ini, tetapi diperlukan usaha lebih dari pengguna untuk meletakkan gambar enkripsi pada tempat yang sama. Selain itu *bot* yang dapat memindahkan kedua gambar di satu tempat yang sama akan dengan mudah melakukan *bypass* dari metode ini.

Oleh karena itu, makalah ini mengusulkan pendekatan baru dalam mekanisme CAPTCHA berbasis kriptografi visual yang memanfaatkan komponen temporal pada persepsi manusia. Berbeda dengan pendekatan sebelumnya, pendekatan ini menampilkan gambar terenkripsi yang muncul secara bergantian sehingga membentuk sebuah pola yang tidak dapat diidentifikasi dari satu *frame* saja. Selain itu, pendekatan ini tidak memerlukan pengguna untuk menyamakan beberapa posisi dari gambar terenkripsi.

II. Landasan Teori

A. CAPTCHA

Completely Automated Public Turing Test To Tell Computers and Humans Apart (CAPTCHA), sesuai dengan namanya merupakan sebuah program berupa tes yang digunakan untuk membedakan antara komputer atau

bot dengan manusia. CAPTCHA sendiri biasanya digunakan untuk mengamankan *software* dari serangan spam ataupun *bot*. Salah satu tipe CAPTCHA paling umum adalah identifikasi teks terdistorsi melalui sebuah gambar, dan kemudian menuliskan ulang teks tersebut ke dalam form.

CAPTCHA sering digunakan pada aplikasi berbasis web. Contoh penggunaannya, adalah pencegahan *spam* pada kolom komentar *blog* atau *forum*. Biasanya kolom komentar dapat diisi konten iklan dan dikirim oleh *bot* secara *automated*. Selain mencegah spam, CAPTCHA juga dapat digunakan untuk melindungi halaman login dari *brute force attack*. *Brute force* attack mencari *password* dengan mencoba seluruh kemungkinan biasanya menggunakan *automated bot*. *Anti-scraping* juga dapat menjadi salah satu pemanfaatan CAPTCHA. Selain dari sisi *security*, CAPTCHA juga dapat digunakan untuk melakukan pelatihan AI model untuk melakukan identifikasi gambar.

CAPTCHA menggunakan algoritma yang *automated* dan *reliable*. *Automated* berarti dapat diotomasi dengan sedikit intervensi dari manusia dan *reliable* berarti andal atau konsisten dalam membedakan manusia dengan *bot*. Selain itu, CAPTCHA dengan algoritma publik membuktikan bahwa algoritma CAPTCHA tidak dapat di-*bypass* hanya dengan menggunakan *reverse engineering*.

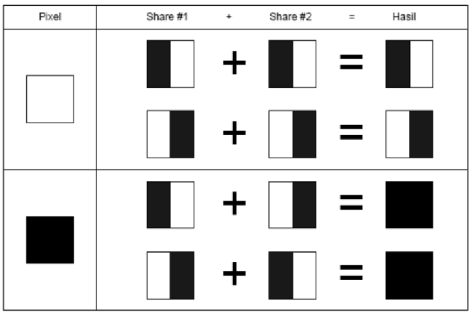
CAPTCHA bergantung pada tiga kemampuan utama manusia dalam mengidentifikasi gambar, yaitu *invariant recognition*, *segmentation*, dan *context recognition*. *Invariant recognition* berarti kemampuan manusia untuk mengenali bentuk objek dengan berbagai variasi. Contohnya, tulisan dengan coretan, distorsi, rotasi dapat dengan mudah dibaca oleh manusia, tetapi sulit dibaca oleh komputer. *Segmentation* berarti kemampuan manusia dalam membedakan objek meskipun dengan posisi yang sangat berdekatan. Terakhir, *context recognition* berarti kemampuan manusia dalam mengenali konteks dari objek atau huruf yang mirip, seperti “u”,”n”, dan “m” hanya dapat diidentifikasi ketika manusia melihatnya dalam sebuah kata yang utuh. Penyelesaian CAPTCHA merupakan sebuah permasalahan yang jauh lebih rumit dibandingkan dengan apa yang terlihat karena kemampuan otak manusia dalam mengenali pola-pola tersebut.

B. Kriptografi Visual

Kriptografi visual merupakan sebuah teknik dalam kriptografi yang melakukan enkripsi informasi dalam media visual. Teknik ini bekerja dengan cara membagi informasi visual dalam beberapa share/bagian yang semi-transparan. Kemudian untuk melakukan dekripsi dari teknik ini cukup dengan menumpuk kedua share dan informasi dapat dipersepsikan secara visual

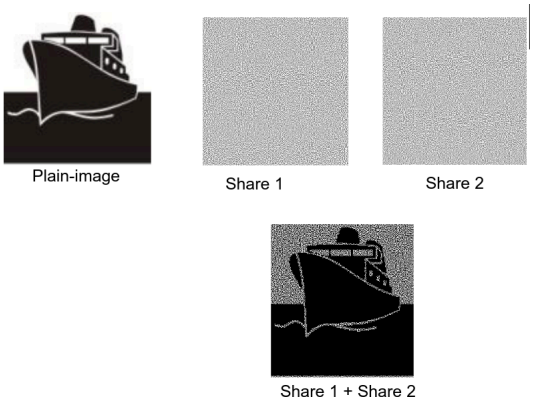
Ide dari kriptografi visual pertama kali dikemukakan oleh Moni Naor dan Adi Shamir pada tahun 1995 di makalahnya yang berjudul *Visual Cryptography*. Pada makalah tersebut, dijelaskan mengenai skema enkripsi dengan pembagian citra biner di mana citra biner tersebut dibagi menjadi n bagian citra biner *semi transparent*. Untuk dapat melakukan dekripsi, diperlukan seluruh n bagian citra terenkripsi yang kemudian ditumpuk untuk mendapatkan kembali informasi terdekripsi.

Skema paling sederhana yang diusulkan oleh Moni Naor dan Adi Shamir adalah skema pembagian (2,2) pada citra biner. Skema ini membagi citra biner menjadi dua bagian. Setiap *pixel* pada citra biner akan dibagi menjadi dua 2x2 *sub-pixel*, dengan setiap bagian memiliki 2x2 *sub-pixel*. Skema ini membagi dengan aturan sederhana, yaitu *pixel* yang memiliki *value* 1 dibagi menjadi dua bagian yang berbeda sedangkan *pixel* dengan *value* 0 dibagi menjadi dua bagian yang sama atau bisa juga sebaliknya. Untuk melakukan dekripsi, tumpuk kedua bagian menjadi satu. Penumpukkan ini dapat dipandang sebagai operasi OR. Untuk lebih jelasnya, dapat dilihat skema pembagian pada Gambar 2.1.



Gambar 2.1 Skema pembagian (2,2)
sumber : Kriptografi Visual, Teori dan Aplikasinya (Bag. 1) - IF4020 - Rinaldi

Citra yang dihasilkan pada hasil dekripsi tidak tepat sama dengan *plain-image*nya. Perbedaan ini disebabkan karena metode penumpukkan yang menggunakan operasi OR tidak akan menghapus sebuah *value* yang ada pada *sub-pixel*. Contoh dari perbedaan antara *plain-image* dengan hasil dekripsi dapat dilihat pada Gambar 2.2.



Gambar 2.2 Perbedaan antara plain-image dengan decrypted image
sumber : Kriptografi Visual, Teori dan Aplikasinya (Bag. 1) - IF4020 - Rinaldi

C. Kriptografi Visual berbasis Integrasi Temporal

Kriptografi visual berbasis integrasi temporal merupakan sebuah pengembangan dari konsep kriptografi visual. Sama seperti kriptografi visual biasa, konsep ini menggunakan skema pembagian dalam enkripsi gambar. Perbedaannya terdapat pada operasi dekripsi. Operasi dekripsi melibatkan komponen waktu untuk dapat menghasilkan citra terdekripsi. Konsep ini memanfaatkan karakteristik *persistence of vision* dari manusia yang membuat manusia mempersepsi cahaya lebih lambat dibandingkan kecepatan perubahan cahaya.

Ide ini dikemukakan oleh Wang, dkk. pada tahun 2017 dalam *conference paper* yang berjudul *Temporal Integration Based Visual Cryptography Scheme and Its Application*. Pada *paper tersebut*, dijelaskan bahwa *persistence of vision* dari manusia yang membuat sensasi cahaya bertahan lebih lama dibandingkan sumber cahaya itu sendiri dapat dimanfaatkan dalam kriptografi visual. Konsep ini bekerja dengan cara melakukan menampilkan setiap *share* secara *alternating* dengan *display* frekuensi tinggi. Misalnya dengan dua *share*, manusia tidak akan melihat kedua *share* sebagai citra yang muncul bergantian, tetapi dilihat sebagai rata rata dari kedua *share*. Persepsi ini dapat dinyatakan secara matematis sebagai:

$$L_m = \frac{S_1+S_2}{2} \quad (1)$$

Di mana L_m adalah persepsi mata sedangkan S_1 dan S_2 adalah share yang ditampilkan secara *alternating*.

III. Perancangan dan Implementasi

A. Desain Skema Kriptografi

Desain skema yang digunakan pada makalah ini adalah skema kriptografi visual dengan ukuran (2,2). Berbeda dengan penjelasan pada Bab II, skema pada implementasi ini menggunakan diagonal dan bukan pada salah satu garis pada sumbu x atau y. Untuk *pixel* dengan warna putih, akan dipilih pola yang sama untuk kedua *share*. Pada *pixel* dengan warna hitam, akan dipilih pola yang berbeda untuk kedua *share*. Tidak ada yang berbeda dari sisi enkripsi dalam makalah ini dengan enkripsi pada kriptografi visual pada umumnya.

Pada sisi dekripsi, makalah ini mengikuti metode yang dikemukakan oleh Wang dkk. Proses dekripsi tidak dilakukan dengan menumpukkan kedua gambar menjadi satu kemudian melihat informasi yang didapatkan, tetapi dilakukan dengan menampilkan kedua *share* secara bergantian dengan cepat. Proses ini melakukan dekripsi dengan mengandalkan persepsi dari mata manusia yang akan melihat perubahan cahaya dengan cepat sebagai nilai rata-rata dari cahaya yang tampak. Bukan sebagai warna putih dan hitam yang bergantian, tetapi sebagai warna abu-abu.

B. Alur Sistem

Sistem diimplementasikan menjadi lima bagian, yaitu *string generator*, *preprocessing*, *visual cryptography encryption*, *rendering,* dan *verification*. Setiap bagian memiliki fungsi yang berbeda dan hampir setiap output dari setiap bagian menjadi input dari bagian selanjutnya. Berikut adalah penjelasan mengenai setiap bagian.

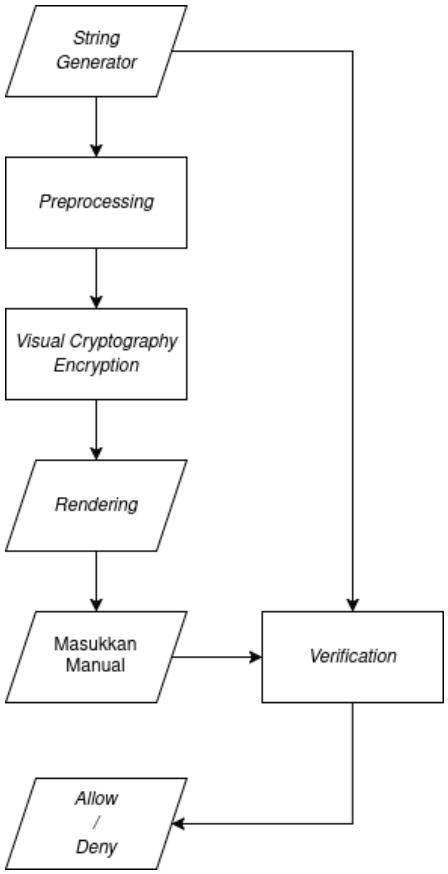
Bagian pertama adalah *string generator*. Bagian ini melakukan *generasi string* secara random. *String generator* ini menerima input opsional berupa panjang string dan memiliki output berupa *string random alphanumeric* dengan panjang *default* sepanjang lima karakter.

Bagian selanjutnya adalah *preprocessing*. Pada bagian ini menerima masukkan berupa *string* yang dihasilkan oleh *string generator*. Bagian ini akan membuat sebuah *canvas* yang kemudian akan diisi gambar berisi *string* masukkan. Dari gambar tersebut kemudian dibentuk sebuah *mask* untuk menandai bagian mana yang memiliki *value* gelap. *Preprocessing* ini mengeluarkan *output* berupa *array* biner berisi penanda bagian yang berwarna gelap.

Bagian *visual cryptography encryption* melakukan enkripsi dari *array* biner yang dihasilkan pada *preprocessing*. Bagian ini membagi *array* biner tersebut menjadi dua *share* yang berbeda. *Output* dari bagian ini adalah *array* biner dari kedua *share*.

Bagian selanjutnya adalah *rendering*. Bagian ini menampilkan kedua share yang dihasilkan pada *visual cryptography encryption*. Bagian ini akan melakukan render secara *alternating*. *Share* satu dan *share dua* ditampilkan secara bergantian dengan cepat sehingga seolah-olah terbentuk sekumpulan huruf berwarna abu-abu.

Bagian terakhir adalah bagian *verification*. Bagian ini menerima masukkan dari pengguna dan juga masukkan dari *string generator*. Kedua masukkan tersebut akan dibandingkan untuk memeriksa apakah nilai dari keduanya sama atau tidak. Jika sama, pengguna dapat mengakses *resource* yang diminta. Jika tidak, akan ditampilkan pesan bahwa pencocokkan gagal. Diagram alur dari sistem dapat dilihat pada Gambar 3.1.



Gambar 3.1 Diagram Alur Sistem

C. Implementasi

Implementasi dari sistem ini dibangun dengan menggunakan HTML, CSS, dan JavaScript tanpa menggunakan *library* dari pihak ketiga. Tidak ada *frontend* atau *backend* pada implementasi ini, hanya *static website*. Fokus dari implementasi ini adalah implementasi kriptografi visual berbasis temporal pada CAPTCHA. Implementasi ini mengasumsikan pengaksesan hanya melalui *interface* yang diberikan dan tidak melalui *devtools* dari *browser*. Berikut adalah penjelasan implementasi dari setiap bagian dari alur sistem.

Pada *string generator*, pembentukkan *string* dibentuk dengan membuat sebuah *string template* "ABCDEFGHIJKLMNOPQRSTUVWXYZ123456789". Dari *string template* dipilih indeks *random* yang akan memilih salah satu karakter dari *string template*. Proses ini diulang sebanyak n-kali dengan nilai n *default* adalah lima. Berikut adalah potongan kode implementasi *string generator*.

```
function generateRandomString(length = 5) {
 const chars =
"ABCDEFGHJKLMNPQRSTUVWXYZ23456789";
 const array = new Uint8Array(length);
 crypto.getRandomValues(array);
 let result = "";
 for (let i = 0; i < length; i++) {
   result += chars[array[i] % chars.length];
 }
 return result;
}
```

Pada bagian selanjutnya, yaitu *preprocessing*, dilakukan proses pembentukan *mask*. Proses ini menerima *string* dan membentuk sebuah kanvas di dalam fungsinya. Kanvas tersebut hanya akan digunakan untuk tempat dari teks dan tidak di-*render*. *Pixel* dari kanvas tersebut akan diperiksa satu per satu. *Pixel* yang memiliki warna gelap akan di-*assign* nilai satu dan nol jika sebaliknya. Nilai tersebut akan dimasukkan ke sebuah *array*. Berikut adalah potongan kode implementasi *preprocessing*.

```
function createTextMask(text) {
 const canvas =
document.createElement("canvas");
 canvas.width = WIDTH;
 canvas.height = HEIGHT;
 const ctx = canvas.getContext("2d");

 ctx.fillStyle = "#fff";
 ctx.fillRect(0, 0, WIDTH, HEIGHT);
 ctx.fillStyle = "#000";
 ctx.font = "bold 40px monospace";
 ctx.textAlign = "center";
 ctx.textBaseline = "middle";
 ctx.fillText(text, WIDTH / 2, HEIGHT / 2);

 const imgData = ctx.getImageData(0, 0,
WIDTH, HEIGHT);

 const mask = [];
 for (let y = 0; y < HEIGHT; y++) {
   mask[y] = [];
   for (let x = 0; x < WIDTH; x++) {
     const idx = (y * WIDTH + x) * 4;
     mask[y][x] = imgData.data[idx] < 128 ? 1
: 0;
   }
 }
 return mask;
}
```

Pada bagian *visual cryptography encryption*, dibentuk sebuah *mask* hasil enkripsi dari bagian sebelumnya. *Mask* yang dihasilkan akan berupa *array* biner dengan ukuran empat kali lebih besar karena perlunya representasi *sub-pixel*. Untuk setiap *pixel* dari *mask* akan diperiksa apakah *pixel* berwarna terang atau gelap. Jika berwarna terang, digunakan pola yang sama pada kedua *share*. Jika berwarna gelap, digunakan pola yang berbeda untuk kedua *share*. Berikut adalah potongan kode dari *visual cryptography encryption*.

```
function generateVisualCryptography(mask) {
 const h = mask.length;
 const w = mask[0].length;
 const share1 = [];
 const share2 = [];
 const patterns = [
   [
     [0, 1],
     [1, 0],
   ],
   [
     [1, 0],
     [0, 1],
   ],
 ];

 for (let y = 0; y < h; y++) {
   share1[y] = [];
   share2[y] = [];
   for (let x = 0; x < w; x++) {
     const pixel = mask[y][x];
     const rand = Math.random() < 0.5 ? 0 :
1;
     if (pixel === 1) {
       share1[y][x] = patterns[rand];
       share2[y][x] = patterns[1 - rand];
     } else {
       share1[y][x] = patterns[rand];
       share2[y][x] = patterns[rand];
     }
   }
 }
 return [share1, share2];
}
```

Terakhir, pada tahap *rendering*, dilakukan *render* secara bergantian dan berulang-ulang secara cepat dari kedua *share* yang dihasilkan. Kedua *share* di-*render* pada tempat yang sama agar pengguna dapat melihat bagian mana yang seolah-olah berwarna abu-abu. Berikut adalah potongan kode dari tahap tersebut.

```
function startFlickering(share1, share2) {
 const canvas =
document.getElementById("temporalCanvas");

 const c1 = document.createElement("canvas");
 const c2 = document.createElement("canvas");

 drawShareToCanvas(share1, c1);
 drawShareToCanvas(share2, c2);

 canvas.width = c1.width;
 canvas.height = c1.height;

 const ctx = canvas.getContext("2d");
 let showFirst = true;

 if (flickerInterval)
clearInterval(flickerInterval);

 flickerInterval = setInterval(() => {
   if (showFirst) {
     ctx.drawImage(c1, 0, 0);
   } else {
     ctx.drawImage(c2, 0, 0);
   }
   showFirst = !showFirst;
 }, 50);
}
```

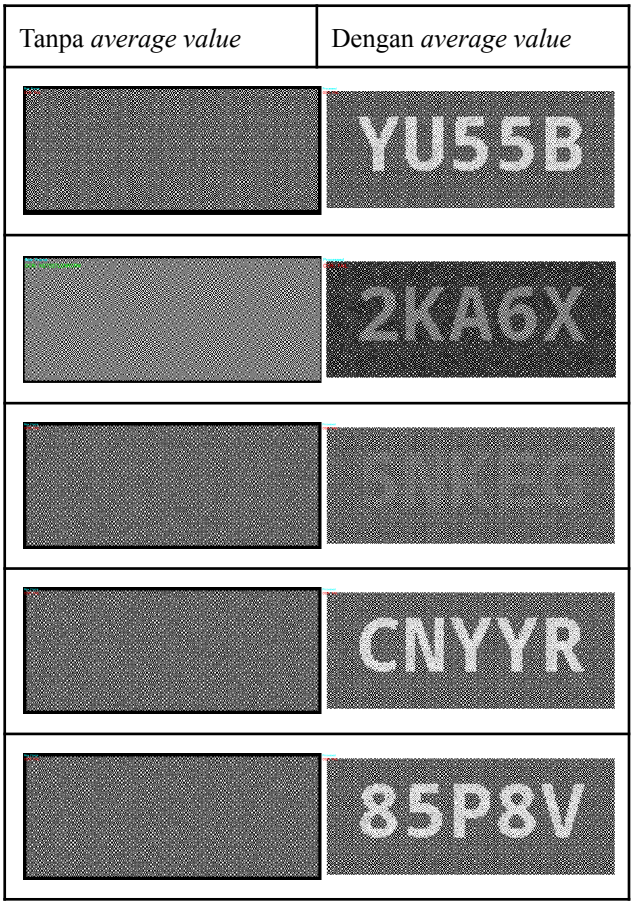
IV. Eksperimen dan Analisis

Saat ini, masih belum banyak aplikasi CAPTCHA *solver* yang tersedia secara gratis sehingga eksperimen dan uji coba pada makalah ini dilakukan dengan terbatas. Uji coba yang dilakukan pada makalah ini meliputi pengujiian kecepatan *flicker* secara manual dan pengujian deteksi *generated string* dengan OCR. Pengujian dengan OCR akan dibagi menjadi dua pengujian yaitu dengan penggunaan *average value* antar *frame* dan tanpa penggunaan *average value*.

Pengujian pertama dengan menguji seberapa mudah atau sulit *string* yang digenerate pada implementasi CAPTCHA terhadap perubahan kecepatan *flicker* antar *share*. Pengujian dilakukan oleh penulis dengan memberikan penilaian seberapa mudah *string* dibaca. Pengujian dilakukan secara terurut dari 2000ms dan menurun dengan faktor pembagi 2 sampai 125ms. Berikut adalah hasil dari pengujian secara manual.

Durasi setiap *share* (ms)	Kesulitan Identifikasi Teks
2000	Sangat sulit
1000	Sulit
500	Sulit
250	Mudah
125	Sangat mudah

Pada pengujian dengan OCR, dilakukan perekaman dari *string* yang di-*render*. Rekaman tersebut kemudian dimasukkan ke deteksi huruf menggunakan OCR. Pengujian ini dilakukan dengan menggunakan Pytesseract dan dijalankan pada Google Colab. Pengujian dilakukan dengan dua metode yaitu pengujian dengan *frame sampling* biasa, dan dengan melakukan *average* antara dua *frame* yang berbeda.



Berdasarkan hasil pengujian, penggunaan durasi *flicker* yang cepat dapat mempermudah manusia untuk mengidentifikasi *string* apa yang tampil pada CAPTCHA ini. Hal ini sesuai dengan karakteristik *persistence of vision*. Pada perubahan cahaya yang cepat, mata manusia tidak mampu mengimbangi kecepatan tersebut sehingga terdapat persepsi munculnya tulisan berwarna abu-abu. Sedangkan pada durasi *flicker* yang lambat, mata manusia hanya melihat pola tulisan tersebut dengan durasi yang sangat sebentar dengan interval yang lama. Hal ini mempersulit manusia dalam mengidentifikasi tulisan pada CAPTCHA.

Pada pengujian menggunakan OCR, dapat terlihat bahwa pengujian dengan *frame sampling* sederhana tidak mampu mengidentifikasi tulisan yang ada. Metode tersebut tidak efektif karena satu *frame* tidak membawa informasi yang berarti. Pada pengujian dengan *average value*, dapat terlihat bahwa terdapat tulisan yang terbentuk. Tulisan tersebut dapat dengan mudah diidentifikasi dan dibaca. Namun pada pengujian tersebut OCR dengan Pytesseract masih kesulitan untuk mengidentifikasi huruf apa saja yang ada pada gambar. Pengujian dengan memasukkannya dengan model yang lebih *advanced* seperti Google Gemini dapat memberikan hasil identifikasi yang akurat.

V. Kesimpulan

Berdasarkan perancangan dan pengujian yang telah dilakukan, makalah ini berhasil mengimplementasikan skema CAPTCHA berbasis kriptografi visual temporal yang memanfaatkan fenomena *persistence of vision*. Pendekatan ini menawarkan keseimbangan antara keamanan dan *usability*, di mana pengguna manusia dapat mengidentifikasi teks tanpa perlu melakukan penyatuan gambar secara manual. Hasil eksperimen menunjukkan bahwa sistem efektif mencegah bot berbasis pengambilan *screenshot* karena informasi rahasia tersebar dalam domain waktu dan tidak dapat direkonstruksi dari satu frame. Meskipun serangan berbasis penggabungan frame video secara teknis dimungkinkan, metode ini menuntut sumber daya komputasi yang jauh lebih besar dari penyerang dibandingkan dengan metode bypass CAPTCHA konvensional.

Referensi

[1] K. William, “Mekanisme CAPTCHA menggunakan kriptografi visual,” Program Studi Teknik Informatika, Institut Teknologi Bandung, Bandung, Indonesia, Makalah IF3058 Kriptografi, Mei 2009.

[2] A. Stec. “What is CAPTCHA and how does it work?” Baeldung.com. https://www.baeldung.com/cs/captcha-intro (diakses Des. 26, 2025).

[3] L. von Ahn, M. Blum, N. J. Hopper, dan J. Langford, “CAPTCHA: Using hard AI problems for security,” dalam *Advances in Cryptology—EUROCRYPT 2003* (Lecture Notes in Computer Science), vol. 2656. Berlin, Heidelberg: Springer, 2003, hlm. 294–311.

[4] M. Naor dan A. Shamir, “Visual cryptography,” dalam *Advances in Cryptology — EUROCRYPT ’94*, vol. 950, A. De Santis, Ed. Berlin, Heidelberg: Springer, 1995, hlm. 1–12.

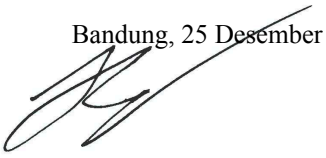
[5] [5] R. Munir, “Kriptografi visual (bagian 1),” IF4020 Kriptografi, Program Studi Teknik Informatika, ITB, Bandung, Indonesia, Bahan Kuliah, 2025. [Daring]. Tersedia: https://informatika.stei.itb.ac.id/~rinaldi.munir/Kriptografi/2025-2026/38-Kriptografi-Visual-Bagian1-2025.pdf.

[6] W. Wang, F. Liu, T. Guo, dan Y. Ren, “Temporal integration based visual cryptography scheme and its application,” dalam *Digital Forensics and Watermarking (IWDW 2017)*, vol. 10431, C. Kraetzer, Y. Q. Shi, J. Dittmann, dan H. Kim, Eds. Cham: Springer, 2017.

Pernyataan

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Bandung, 25 Desember 2025



M. Hanief Fatkhan Nashrullah - 13522100

LAMPIRAN

Source code :
https://github.com/hannoobz/temporal-visual-cryptography-captcha
Demo:
https://hannoobz.github.io/temporal-visual-cryptography-captcha/
Video Youtube:
https://youtu.be/FA7OfJSYirI?si=Z66OguziClgirnqC

Tangkapan layar hasil implementasi: