



## Implementasi Algoritma *OTP* dan *HMAC* untuk *Two-Factor Authentication* Sistem Login Relawan Pemilu

Chairil Anwar\*<sup>1</sup>, Sriani<sup>2</sup>

\*<sup>1,2</sup>Program Studi Ilmu Komputer, Universitas Islam Negeri Sumatera Utara, Medan, Indonesia

\*Email Penulis Korespondensi: [chairil.anwarg70@gmail.com](mailto:chairil.anwarg70@gmail.com)

### Abstrak

Dalam era digital, keamanan informasi pemilihan umum (pemilu) menjadi sangat penting, terutama dalam Sistem Informasi Relawan Pemilu (SIRP) yang menyimpan data sensitif. Penelitian ini mengeksplorasi penggunaan *Two-Factor Authentication* (2FA) dengan menggabungkan algoritma *One-Time Pad* (OTP) dan *HMAC* untuk meningkatkan keamanan login pada SIRP. Sistem 2FA ini menghasilkan kode OTP berupa 6 digit angka yang berganti setiap 15 detik, sehingga meningkatkan keamanan akses dengan memastikan bahwa setiap kode hanya berlaku untuk satu kali penggunaan. Penelitian ini berfokus pada kelemahan sistem password yang ada dan bagaimana 2FA dapat mengatasi masalah tersebut. Metode penelitian melibatkan implementasi OTP untuk menghasilkan kode autentikasi dinamis dan HMAC untuk memastikan integritas data. Pengujian dilakukan dalam lingkungan simulasi untuk menilai efektivitas dan keandalan sistem yang diusulkan. Hasil menunjukkan bahwa 2FA dengan OTP dan HMAC secara signifikan meningkatkan keamanan login dengan mengurangi kerentanan terhadap serangan siber seperti brute force dan pencurian password. Penelitian ini memberikan kontribusi dalam pengembangan metode keamanan autentikasi yang lebih kuat dan memberikan dasar bagi implementasi sistem keamanan yang lebih baik pada SIRP dan sistem informasi lainnya.

**Kata kunci**—Algoritma, *One Time Pad*, *HMAC*, *Two-Factor Authentication*

### Abstract

In the digital age, the security of election information is crucial, particularly within the Election Volunteer Information System (SIRP) that stores sensitive data. This study explores the use of *Two-Factor Authentication* (2FA) by combining the *One-Time Pad* (OTP) algorithm and *HMAC* to enhance login security on SIRP. The 2FA system Generates a 6-digit OTP that changes every 15 seconds, increasing access security by ensuring each code is valid for only one-time use. The research focuses on identifying vulnerabilities in existing password systems and how 2FA can address these issues. The methodology involves implementing OTP to produce dynamic authentication codes and HMAC to ensure data integrity. Testing is conducted in a simulated environment to evaluate the effectiveness and reliability of the proposed system. Results indicate that 2FA with OTP and HMAC significantly improves login security by reducing vulnerability to cyber-attacks such as brute force and password theft. This research contributes to developing

---

*stronger authentication security methods and provides a foundation for better security system implementation on SIRP and other information systems.*

**Keywords**—*Algorithm, One Time Pad, HMAC, Two-Factor Authentication*

## 1. PENDAHULUAN

Pemilihan umum (pemilu) menjadi fondasi utama dalam mempertahankan demokrasi di suatu Negara. Dalam era digital saat ini, teknologi informasi memainkan peran yang semakin penting dalam berbagai aspek kehidupan, termasuk dalam proses pemilu [1].

Dengan semakin kompleksnya proses pemilu yang dipadukan dengan teknologi, keamanan informasi dan data menjadi hal yang sangat vital [2], [3]. Salah satu tantangan utama dalam menjaga keamanan SIRP adalah memastikan akses ke sistem tersebut terlindungi dengan baik, terutama melalui mekanisme login yang aman [4].

Namun, kenyataannya menunjukkan bahwa banyak pengguna masih menggunakan *password* yang lemah atau mudah ditebak. Akibatnya, meskipun sudah ada upaya untuk meningkatkan keamanan dengan menggunakan karakter khusus, kata sandi pada Sistem Informasi Relawan Pemilu (SIRP) sering kali masih belum sepenuhnya aman, sehingga menimbulkan potensi resiko keamanan pada sistem tersebut [5]. Seiring dengan perkembangan serangan siber yang semakin canggih, hal ini menimbulkan risiko besar bagi sistem seperti SIRP. Oleh karena itu, diperlukan solusi yang lebih kuat untuk melindungi akses ke SIRP, salah satunya adalah dengan menggunakan algoritma untuk mengamankan *password* dengan mengubahnya menjadi format yang tidak dapat dipahami tanpa kunci yang sesuai [6], [7].

Algoritma OTP merupakan metode autentikasi yang menghasilkan kode unik yang hanya berlaku untuk satu kali penggunaan [8]. Dengan sifat yang dinamis dan sekali pakai, algoritma ini dapat memberikan perlindungan ekstra terhadap serangan siber. Implementasi OTP dalam SIRP tidak hanya akan meningkatkan keamanan login, tetapi juga memberikan perlindungan tambahan terhadap data sensitif yang disimpan dalam sistem tersebut.

Penerapan *Two-Factor Authentication* menggunakan kombinasi *OTP* dan *HMAC* menambahkan lapisan keamanan ekstra, dengan *HMAC* menghasilkan kode autentikasi unik yang meningkatkan resistensi terhadap serangan siber [9]. Penelitian sebelumnya juga menekankan pentingnya autentikasi yang lebih kuat untuk mengamankan sistem. Penggunaan *OTP* mempunyai keunggulan dalam mencegah serangan berbasis *password reuse* dan pentingnya penerapan kode yang hanya dapat digunakan sekali dalam sistem kritis [10]. Berdasarkan penelitian oleh [3] telah mengeksplorasi metode-metode untuk menghasilkan *OTP* yang lebih aman, sedangkan penelitian [2] membahas tantangan dalam penggunaan autentikasi berbasis *password* dan kebutuhan untuk meningkatkan keamanan dengan lapisan tambahan.

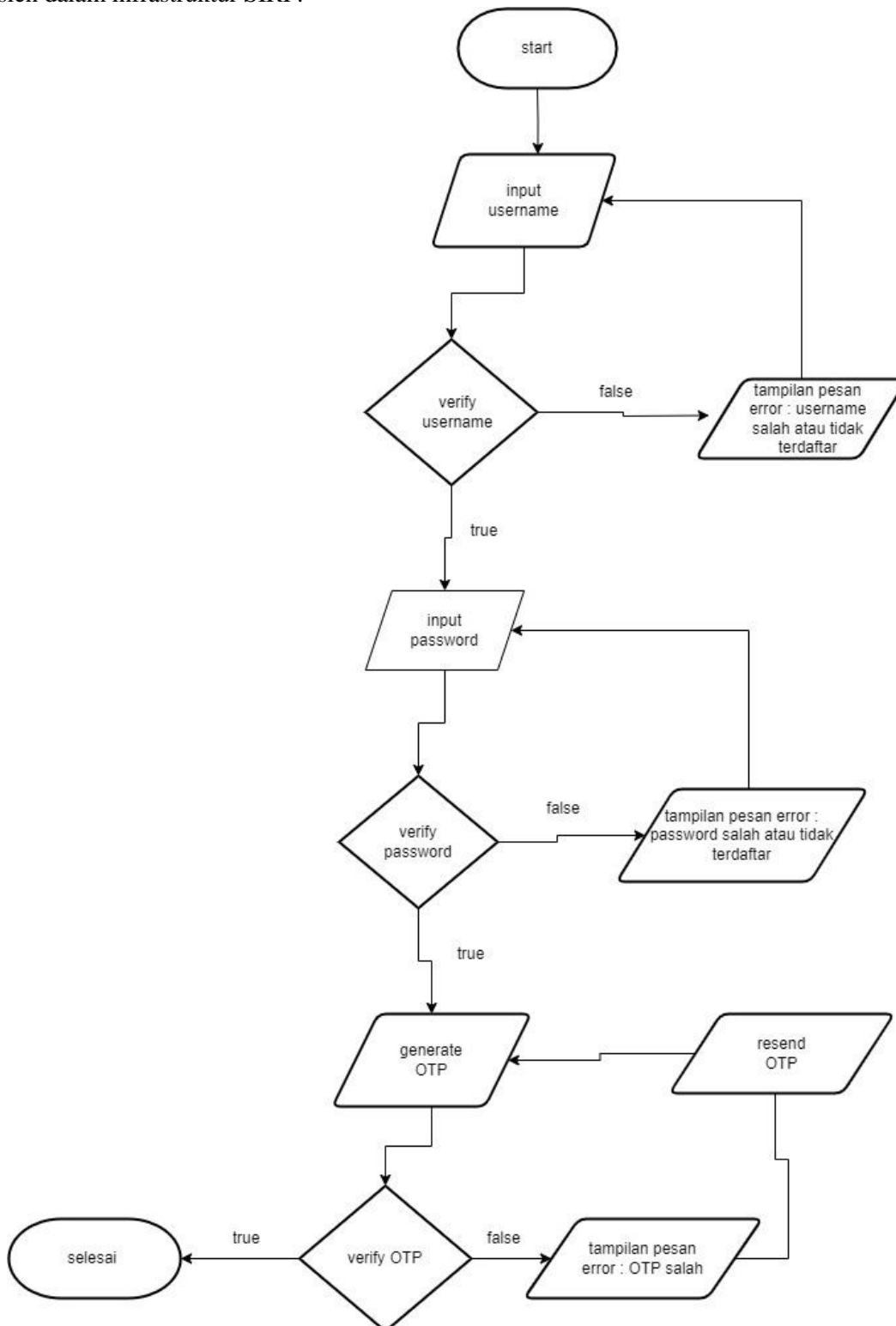
Penelitian ini memilih verifikasi dua langkah *OTP* dan *HMAC* karena memberikan perlindungan lebih kuat dibanding metode lain. *OTP* menghasilkan kode unik yang hanya berlaku dalam waktu singkat, membuatnya sulit untuk ditebak atau dicuri. *HMAC* memastikan integritas dan keaslian data, mencegah pemalsuan selama proses autentikasi. Metode alternatif seperti token fisik atau biometrik kurang efisien atau lebih kompleks dalam implementasi, sehingga kombinasi *OTP* dan *HMAC* dianggap solusi terbaik untuk meningkatkan keamanan sistem.

## 2. METODE PENELITIAN

Metode penelitian ini dimulai dengan identifikasi masalah keamanan pada Sistem Informasi Relawan Pemilu (SIRP), yang melibatkan analisis ancaman potensial, evaluasi kerentanan dalam otentikasi dan pengelolaan akses, serta penelusuran kebutuhan keamanan spesifik. Pengkajian terhadap insiden keamanan pada *platform* serupa dan konsultasi dengan pakar dilakukan untuk mendapatkan wawasan lebih dalam tentang potensi ancaman.

Selanjutnya, tinjauan literatur dilakukan untuk mengeksplorasi penelitian terdahulu terkait penggunaan algoritma *One-Time Pad* (*OTP*) dan *Hash-based Message Authentication*

*Code* (HMAC) dalam otentikasi, yang kemudian dijadikan dasar dalam perancangan rencana penelitian. Perancangan ini mencakup pengembangan *flowchart* sistem dari input *username* hingga verifikasi OTP, dengan fokus pada pemilihan algoritma yang tepat dan integrasi yang efisien dalam infrastruktur SIRP.



Gambar 1 *Flowchart* Sistem

```

Start Program

Function generateOTP:
  Clear previous interval
  Set secret = "my_secret"
  Set counter = current time / 15 seconds
  Call generateHOTP(secret, counter)
  Display OTP
  Start interval to update OTP every second

Function generateHOTP(secret, counter):
  Convert secret to bytes
  Convert counter to 8-byte array
  Generate HMAC-SHA1 hash using secret and counter
  Extract 6-digit OTP from hash result
  Return OTP

Function startOtpInterval:
  Every 1 second:
    Update OTP
    Update countdown timer

Call generateOTP on start

End Program

```

Gambar 2 Pseudocode Generate OTP

```

Function verifyOTP:
  Get entered OTP
  Get generated OTP

  If entered OTP equals generated OTP:
    Show success message
    Redirect to admin page
  Else:
    Show error message

End Function

```

Gambar 3 Pseudocode Verifikasi OTP

Gambar 1, Gambar 2, dan Gambar 3 menjelaskan bagaimana menggunakan HMAC untuk menghasilkan dan memverifikasi kode OTP. Fungsi *generate* HMAC mengambil *secret* dan *timestamp* sebagai input untuk menghasilkan hash *HMAC-SHA1* dan mengonversinya menjadi kode OTP 6 digit. Fungsi *generate* OTP menggunakan *timestamp* untuk terus memperbarui OTP setiap detik dan menampilkan kode tersebut ke pengguna. *startOtpInterval* memastikan OTP diperbarui secara berkala. Fungsi *verify OTP* membandingkan OTP yang dimasukkan oleh pengguna dengan OTP yang dihasilkan, lalu menampilkan pesan sukses atau gagal.

### 3. HASIL DAN PEMBAHASAN

Penulis memulai perancangan sistem login yang lebih aman dengan menganalisis kelemahan sistem saat ini, terutama terkait serangan *brute force* dan *phishing*. Untuk mengatasi hal tersebut, penulis menetapkan spesifikasi teknis untuk implementasi *Two-Factor Authentication* (2FA) menggunakan OTP dan HMAC, memastikan kompatibilitas dengan sistem yang ada meskipun memerlukan beberapa penyesuaian infrastruktur.

### 3.1 Implementasi One Time Pad (OTP)

*One time pad* (OTP) adalah salah satu algoritma enkripsi yang paling aman karena mengenkripsi setiap karakter *Plaintext* dengan kunci acak yang hanya digunakan sekali [11]. Rumus dasar OTP dapat digunakan untuk mengenkripsi *password* dapat dilihat pada formula (1).

$$\text{Kunci Enkripsi (C)} = \text{Pesan (P)} \text{ XOR Kunci (K)} \quad (1)$$

Keterangan:

C = teks terenkripsi yang dihasilkan.

P = teks asli atau pesan yang ingin dienkripsi.

K = kunci enkripsi yang bersifat dinamis dan hanya digunakan sekali.

XOR adalah operasi logika yang menghasilkan TRUE (1) jika dua digit berbeda dan FALSE (0) jika sama. Panjang pesan (P) dan kunci (K) harus sama, dengan kunci yang benar-benar acak dan tidak boleh digunakan kembali. Teks terenkripsi (C) sangat sulit didekripsi tanpa kunci yang tepat, yang harus dirahasiakan dan hanya diketahui oleh pengirim dan penerima. Prosedur operasional algoritma *One Time Pad* dapat dilihat pada formula (2) dan formula (3) seabgai berikut:

$$1. \text{ Proses enkripsi dilakukan dengan persamaan } C_i = (P_i + K_i) \text{ mod } 26 \quad (2)$$

$$2. \text{ Sementara proses dekripsi dijalankan dengan persamaan } P_i = (C_i - K_i) \text{ mod } 26 \quad (3)$$

Dalam persamaan tersebut,  $P_i$  menunjukkan *Plaintext* sementara  $K_i$  merujuk pada kunci yang digunakan. Sebagai contoh: *Plaintext* (CHAIRIL):

C=2, H=7, A=0, I=8, R=17, I=8, L=11

Kunci acak (misal: XYZABCD):

X=23, Y=24, Z=25, A=0, B=1, C=2, D=3

Informasi mengenai karakter penomoran dijelaskan dalam Tabel 1.

Tabel 1 Karakter Penomoran Bilangan

A	B	C	D	E	F	G	H	I	J	K	L	M
0	1	2	3	4	5	6	7	8	9	10	11	12
N	O	P	Q	R	S	T	U	V	W	X	Y	Z
13	14	15	16	17	18	19	20	21	22	23	24	25

Sekarang, mari kita enkripsi teks "CHAIRIL" dengan menggunakan kunci acak yang telah ditentukan:

$$(C+X) \text{ mod } 26 = (2+23) \text{ mod } 26 = 25 = Z$$

$$(H+Y) \text{ mod } 26 = (7+24) \text{ mod } 26 = 5 = F$$

$$(A+Z) \text{ mod } 26 = (0+25) \text{ mod } 26 = 25 = Z$$

$$(I+A) \text{ mod } 26 = (8+0) \text{ mod } 26 = 8 = I$$

$$(R+B) \text{ mod } 26 = (17+1) \text{ mod } 26 = 18 = S$$

$$(I+C) \text{ mod } 26 = (8+2) \text{ mod } 26 = 10 = K$$

$$(L+D) \text{ mod } 26 = (11+3) \text{ mod } 26 = 14 = O$$

Jadi, hasil enkripsi dari teks "CHAIRIL" dengan menggunakan kunci acak "XYZABCD" adalah "ZFIISKO". Untuk mendekripsi pesan, karakter teks *cipher* dikurangi dengan karakter kunci yang sesuai. Dekripsi tidak mungkin dilakukan tanpa kunci *One-Time Pad* yang sama panjang dan benar-benar acak. Alternatif lain adalah menggunakan operasi XOR antara setiap bit *Plaintext* dan kunci untuk mengamankan pesan, memastikan pesan tidak bisa dibaca tanpa kunci yang benar.

*Plaintext*: DIAM

Kunci: TNBU

Proses enkripsi dimulai dengan mengonversi setiap *Plaintext* dan kunci ke dalam representasi biner berdasarkan tabel ASCII seperti dapat dilihat pada Tabel 2 dan Tabel 3.

Tabel 2 Konversi Karakter ke ASCII dan Biner

Karakter	ASCII	Biner
D	68	01000100
I	73	01001001
A	65	01000001
M	77	01001101

Tabel 3 akan menjelaskan konversi bilangan kunci ke bilangan biner.

Tabel 3 Konversi Karakter ke ASCII dan Biner

Karakter	ASCII	Biner
T	84	01010100
N	78	01001110
B	65	01000010
U	77	01010101

Setelah konversi, langkah selanjutnya adalah melakukan operasi XOR antara setiap bit dari *Plaintext* dengan kunci yang sesuai:

D: 0100 0100 (*Plaintext*)  $\oplus$  0101 0100 (Kunci) = 0001 0000 -> Hasil Enkripsi: Q

I: 0100 1001 (*Plaintext*)  $\oplus$  0100 1110 (Kunci) = 0000 0111 -> Hasil Enkripsi: G

A: 0100 0001 (*Plaintext*)  $\oplus$  0100 0010 (Kunci) = 0000 0011 -> Hasil Enkripsi: C

M: 0100 1101 (*Plaintext*)  $\oplus$  0101 0101 (Kunci) = 0001 1000 -> Hasil Enkripsi: H

Oleh karena itu, hasil enkripsi pesan "DIAM" dengan kunci "TNBU" adalah "QGCH". Proses ini menjelaskan bagaimana operasi XOR pada setiap bit pesan dan kunci menghasilkan *ciphertext*, yang merupakan hasil enkripsi *Plaintext* dengan kunci yang sesuai. Hasil enkripsi ini kemudian dapat didekripsi kembali dengan menggunakan kunci yang sama untuk mendapatkan kembali pesan asli.

### 3.2 Implementasi Hash-based Message Authentication Code (HMAC)

Dalam konteks Sistem Informasi Relawan Pemilu (SIRP), penggunaan kode Otentikasi Pesan Berbasis Hash (HMAC) mengatakan bahwa itu dapat menyediakan integritas dan autentikasi, dan itu adalah kode autentikasi mesin yang terkenal yang banyak digunakan dalam berbagai aplikasi keamanan siber[12], [13]. Rumus dasar untuk menghasilkan HMAC dapat dilihat pada formula (4).

$$\text{HMAC}(K,M)=H((K\oplus\text{opad})\parallel H((K\oplus\text{ipad})\parallel M)) \quad (4)$$

Penjelasan:

- H adalah fungsi hash (misalnya, SHA-256).
- K adalah kunci rahasia yang dipakai.
- M adalah pesan asli yang akan di-hash.
- Opad adalah padding luar (*outer padding*).
- Ipad adalah padding dalam (*inner padding*).
- $\parallel$  menunjukkan operasi concatenation atau penggabungan.

Dua tahap *hashing* digunakan dalam prosedur ini untuk menghasilkan HMAC akhir. Pertama, pesan digabungkan dengan *padding* dalam (*ipad*) dan di-*hash*; kemudian, hasilnya digabungkan dengan padding luar (*opad*) dan di-*hash* kembali.

### 3.3 Implementasi HMAC untuk OTP pada Two-Factor Authentication

Dalam sistem *Two-Factor Authentication*, penerapan HMAC (*Hash-based Message Authentication Code*) menjadi unsur penting untuk memastikan bahwa kode OTP (*One-Time Password*) jika kunci tersebut bersifat acak dan hanya digunakan satu kali, maka algoritma tersebut sepenuhnya aman [14].

Dalam penggunaan ini, HMAC dan OTP digunakan untuk membuat kode autentikasi yang hanya sah untuk satu kali penggunaan dan berubah setiap 15 detik. Proses ini memastikan bahwa hanya pengguna yang sah yang dapat mengakses sistem dengan memasukkan kode OTP yang valid. Langkah-langkah Implementasi:

#### 1. Kunci Rahasia (*Secret Key*)

Kunci  $K$  ini harus dijaga kerahasiaannya dan hanya dapat diketahui oleh server dan perangkat pengguna yang sah. Untuk setiap permintaan OTP, kunci rahasia ini digunakan sebagai input dalam proses HMAC. Keamanan kunci ini sangat penting karena seluruh keamanan sistem OTP bergantung padanya.

#### 2. Nilai counter (*timestamp*)

Nilai counter  $C$  yang berubah secara berkala, dalam hal ini setiap 15 detik, memastikan bahwa setiap OTP yang dihasilkan akan berbeda meskipun permintaan OTP terjadi pada waktu yang sama. Counter dihitung dengan membagi waktu saat ini dengan interval waktu, misalnya 15 detik. Metode untuk menghitung jumlah counter dituliskan seperti formula (5).

$$\text{Counter} = \text{Interval} / \text{Time} \quad (5)$$

Keterangan:

- Time = Waktu sekarang dalam detik sejak Epoch
- Interval = 15 detik (atau interval waktu yang ditentukan)

#### 3. Proses HMAC

Fungsi HMAC dibuat dengan menggunakan kunci rahasia  $K$  dan counter  $C$  sebagai input. Nilai hash yang dihasilkan dari fungsi ini kemudian digunakan untuk menghasilkan OTP.

#### 4. Truncation dan Modulo Operasi

Setelah membuat HMAC, nilai hash di-truncate, atau dipotong, menjadi 31 bit terakhir dari byte keempat. Kemudian, untuk menghasilkan 6 digit OTP, dilakukan operasi modulo  $10 \times 6$ .

#### 5. Penggunaan OTP

Dalam proses autentikasi dua faktor, OTP yang dibuat digunakan sebagai faktor kedua. Pengguna harus memasukkan OTP ini dalam waktu yang tepat, yaitu 15 detik, untuk berhasil melakukan login.

Contoh Implementasi:

Misalkan, kita menggunakan kunci rahasia  $K = \text{"my\_secret"}$  dan counter  $C$  yang diperbarui setiap 15 detik. Langkah-langkah perhitungan HMAC dan OTP adalah sebagai dapat dilihat pada formula (6) dan formula (7).

#### 1. HMAC Generation: Menggunakan SHA-1 sebagai fungsi hash:

$$\text{HMAC}(K,C) = H((K \oplus \text{opad}) \parallel H((K \oplus \text{ipad}) \parallel C)) \quad (6)$$

#### 2. Truncation dan Modulo Operasi:

$$\text{OTP} = (\text{HMAC}(K,C)[31 \text{ bit terakhir}]) \bmod 106 \quad (7)$$

Misalnya, jika HMAC menghasilkan *hash* dengan nilai akhir 1004928816, maka OTP adalah:

$$\text{OTP} = 1004928816 \bmod 106 = 928816$$

#### 3. Kode OTP:

OTP ini kemudian digunakan sebagai kode yang harus dimasukkan oleh pengguna dalam waktu 15 detik. Jika berhasil, pengguna akan diberi akses ke sistem.

### 3.4 Tampilan Sistem dan Pembahasan

#### 3.4.1 Desain Sistem 2FA

Penulis merancang arsitektur yang jelas dan terstruktur untuk memastikan sistem 2FA dapat digunakan dengan baik. Penulis menempatkan OTP dan HMAC sebagai bagian penting dari proses autentikasi dalam desain ini, karena keduanya bekerja sama untuk menjamin keamanan login pengguna.

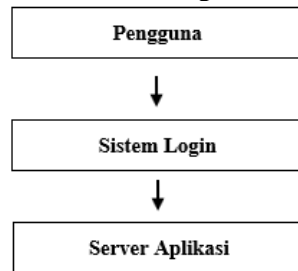
##### A. Pengelolaan Kunci dan Token

Pengelolaan kunci dan token yang digunakan dalam OTP dan HMAC adalah komponen penting dari desain ini. Penulis merancang sistem yang memungkinkan membuat dan menyimpan kunci dengan aman dan memastikan bahwa distribusi token kepada pengguna dilakukan dengan cara yang paling aman dan efisien dengan mengurangi potensi pencurian data [15]. Selain itu, penulis mempertimbangkan masa berlaku token dan kebijakan keamanan terkait untuk menjaga keandalan sistem.

##### B. Desain Antarmuka Pengguna

Penulis mengoptimalkan desain antarmuka 2FA agar mudah digunakan dan dipahami, dengan menyediakan *wireframe* serta instruksi yang jelas. Selain itu, diagram arsitektur dibuat untuk menunjukkan alur kerja, mulai dari input pengguna hingga verifikasi server, menjelaskan komponen utama sistem informasi relawan pemilu.

1. Pengguna, Orang yang akan menggunakan sistem login relawan pemilu.
2. Sistem Login, Memfasilitasi proses autentikasi pengguna melalui 2FA dengan OTP dan HMAC.
3. Server Aplikasi, Ini adalah server yang digunakan untuk menjalankan aplikasi relawan pemilu dan menyimpan data relawan serta informasi login.



Gambar 4 Komponen

Gambar 5 menunjukkan tampilan desain antarmuka sistem yang berhasil dibuat oleh peneliti.

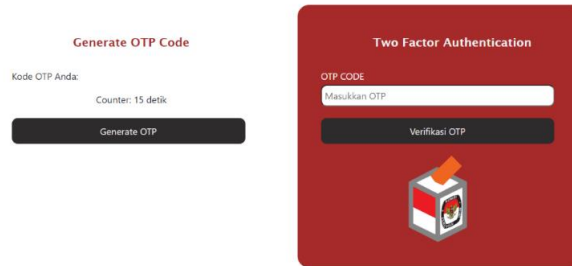


Gambar 5 Tampilan Halaman Login



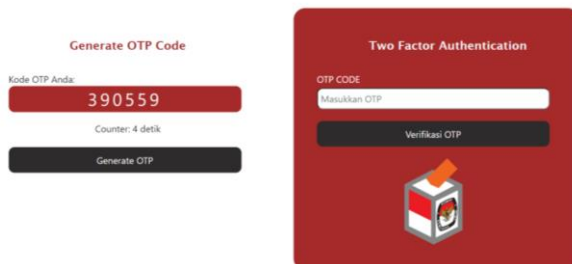
Form login Sistem Informasi Relawan Pemilu 2024 memastikan bahwa pengguna yang mencoba mengakses sistem adalah orang yang sah. Pengguna diminta untuk memasukkan *username* dan *password* mereka, yang kemudian diverifikasi dengan *One-Time Pad* (OTP) untuk meningkatkan keamanan. Setelah validasi, pengguna diarahkan ke halaman verifikasi OTP. Proses autentikasi dimulai dengan pengguna memasukkan *username* sebagai pengenal unik dan password yang dienkripsi menggunakan OTP. Sistem kemudian memverifikasi data tersebut, dan jika valid, pengguna akan dialihkan ke halaman untuk menghasilkan OTP.

Tampilan ini merupakan antarmuka halaman *generate* kunci atau OTP nya, sebelum pengguna mengklik tombol *Generate* OTP.



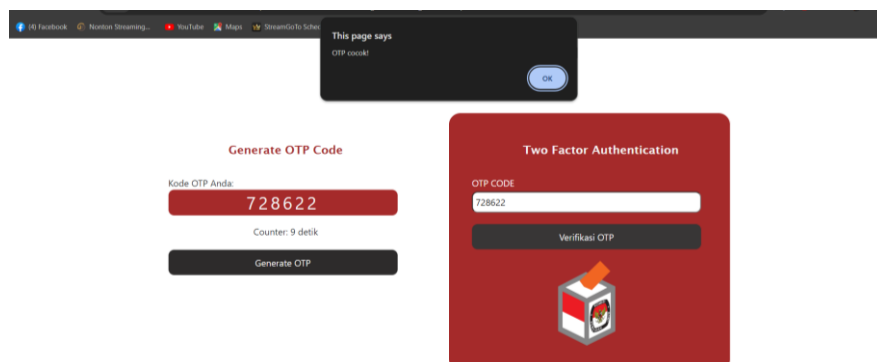
Gambar 6 Halaman *Generate* OTP 1

Gambar 7 adalah tampilan sesudah pengguna mengklik tombol *Generate* OTP, dimana sistem akan melakukan *Generate* otp berupa angka sebanyak 6 digit dengan counter waktu 15 detik.



Gambar 7 Halaman *Generate* OTP 2

Jika kunci atau OTP yang dimasukkan benar, maka sistem akan memberikan akses ke Sistem Informasi Relawan Pemilu.

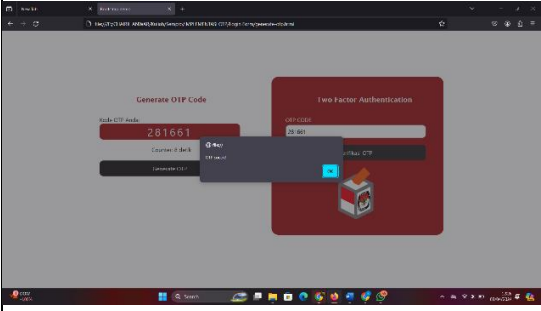
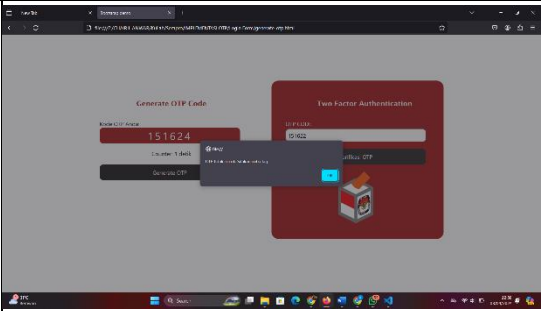
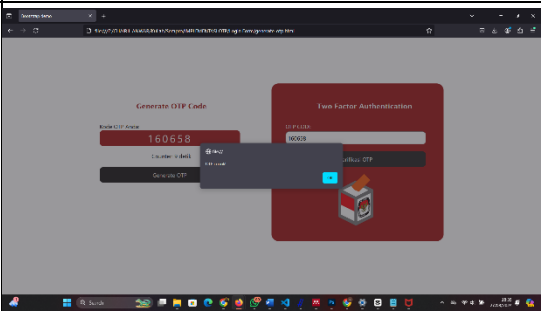
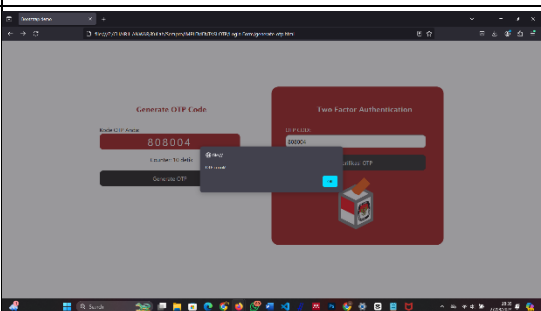


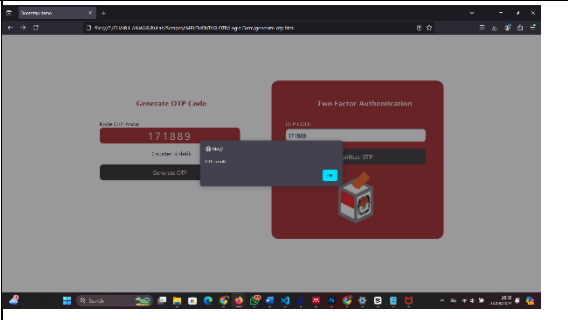
Gambar 8 Halaman *Generate* OTP 3

### 3.4.2 Hasil Pengujian

Tabel 4 adalah hasil pengujian implementasi *One-Time Pad (OTP)* dan *Hash-based Message Authentication Code (HMAC)* dalam konteks *Two-Factor Authentication*.

Tabel 4 Hasil Pengujian

No.	OTP yang Dhasilkan	Proses Pengujian	Hasil
1.	281661		OTP Cocok
2.	151624		OTP Cocok
3.	160658		OTP Cocok
4.	808004		OTP Cocok

5.	171889		OTP Cocok
----	--------	--	-----------

### 3.4.3 Keunggulan dan Keterbatasan Sistem

Sistem OTP yang menggabungkan *One Time Pad* (OTP) dan HMAC menawarkan keamanan tinggi dengan enkripsi satu kali pakai yang sulit diretas dan perlindungan ekstra melalui *Two-Factor Authentication*. Dengan masa berlaku OTP singkat, sekitar 15 detik, sistem ini juga aman dari serangan replay. Meskipun verifikasi cepat dan mudah, tantangannya terletak pada implementasi yang kompleks dan ketergantungan pada sinkronisasi waktu yang tepat, di mana kesalahan sinkronisasi bisa membuat OTP tidak valid.

## 4. KESIMPULAN

Kesimpulan dari penelitian ini menunjukkan bahwa penerapan *Two-Factor Authentication* (2FA) berbasis OTP dan HMAC berhasil diuji, menghasilkan OTP secara konsisten setiap 15 detik. Sistem ini efektif dalam memperkuat keamanan akses pada Sistem Informasi Relawan Pemilu (SIRP) dan siap diimplementasikan untuk meningkatkan perlindungan terhadap ancaman siber.

## 5. SARAN

Penelitian lanjutan dapat mengeksplorasi metode autentikasi lebih canggih, seperti biometrik, security token atau autentikasi berbasis perilaku, untuk meningkatkan keamanan *two factor authentication*. Selain itu, penerapan kecerdasan buatan dapat membantu mendeteksi pola mencurigakan dan menambah lapisan keamanan, menjadikan sistem lebih adaptif terhadap ancaman siber yang terus berkembang.

## UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Tim Redaksi Jurnal Teknik Politeknik Negeri Sriwijaya yang telah memberi kesempatan, sehingga artikel ilmiah ini dapat diterbitkan.

## DAFTAR PUSTAKA

- [1] A. E. B. Arieska and F. S. Mukti, "Pemanfaatan One-Time Password dan Algoritma Advanced Encryption Standard dalam Sistem Login Internet Kampus," *G-Tech J. Teknol. Terap.*, vol. 6, no. 2, pp. 295–305, 2022.
- [2] N. Cristy and F. Riandari, "Implementasi Metode Advanced Encryption Standard (AES 128 Bit) Untuk Mengamankan Data Keuangan," *JIKOMSI [Jurnal Ilmu Komput. dan Sist. Informasi]*, vol. 4, no. 2, pp. 75–85, 2021, [Online]. Available: <https://ejournal.sisfokomtek.org/index.php/jikom/article/view/181%0A>
- [3] M. Furqan, Sriani, and D. Vita, "Application Of RSA-CRT Cryptographic Methods And LSB-LCG Steganography Method In Message Security Systems With Video Media," *J. Infokum*, vol. 10, no. 3, pp. 93–103, 2022.
- [4] I. Permana, M. Hardjianto, and K. Ahmad Baihaqi, "Securing the Website Login System

- with the SHA256 Generating Method and Time-based One-time Password (TOTP),” *Systematics*, vol. 2, no. 2, pp. 65–71, 2020, doi: 10.35706/sys.v2i2.3756.
- [5] D. He, Z. Liu, S. Zhu, S. Chan, and M. Guizani, “Special Characters Usage and Its Effect on Password Security,” *IEEE Internet Things J.*, vol. 11, no. 11, pp. 19440–19453, 2024, doi: 10.1109/JIOT.2024.3367323.
- [6] Y. Ramadhan, S. Suhardi, and Y. Aditama, “Data security using low bit encoding algorithm and rsa algorithm,” *J. Mantik*, vol. 8, no. 1, 2024, [Online]. Available: <http://iocscience.org/ejournal/index.php/mantik/article/view/4945%0Ahttps://iocscience.org/ejournal/index.php/mantik/article/download/4945/3471>
- [7] A. Iqbal and S. Zafar, “SHA-3 Algorithm-based Authentication Scheme for Vehicular Ad-hoc Networks,” *2021 Int. Conf. Cyber Warf. Secur. ICCWS 2021 - Proc.*, pp. 16–23, 2021, doi: 10.1109/ICCWS53234.2021.9703025.
- [8] N. Sarah Hapsari, Y. Fatman, and E. Penulis Korespondensi, “JURNAL MEDIA INFORMATIKA BUDIDARMA Implementasi Metode One Time Password pada Sistem Pemesanan Online,” *J. Media Inform. Budidarma*, vol. 4, pp. 930–939, 2020, doi: 10.30865/mib.v4i4.2195.
- [9] S. M. S. Hussain, S. M. Farooq, and T. S. Ustun, “Analysis and implementation of message authentication code (MAC) algorithms for GOOSE message security,” *IEEE Access*, vol. 7, pp. 80980–80984, 2019, doi: 10.1109/ACCESS.2019.2923728.
- [10] M. I. Batubara, “Implementasi Algoritma *One time pad* (OTP) untuk Pengamanan Pesan Short Message Service (SMS),” vol. 4, 2019.
- [11] Z. Indra and R. Cyra Nabila, “Implementation of the RSA Algorithm and the *One time pad* Algorithm for Text Message Security,” *Formosa J. Sci. Technol.*, vol. 2, no. 1, p. 379, 2023, [Online]. Available: <https://journal.formosapublisher.org/index.php/fjst>
- [12] C. E. Castellon, S. Roy, O. P. Kreidl, A. Dutta, and L. Boloni, “Towards an Energy-Efficient Hash-based Message Authentication Code (HMAC),” *2022 IEEE 13th Int. Green Sustain. Comput. Conf. IGSC 2022*, pp. 1–7, 2022, doi: 10.1109/IGSC55832.2022.9969377.
- [13] D. Upadhyay, N. Gaikwad, M. Zaman, and S. Sampalli, “Investigating the Avalanche Effect of Various Cryptographically Secure Hash Functions and Hash-Based Applications,” *IEEE Access*, vol. 10, no. August, pp. 112472–112486, 2022, doi: 10.1109/ACCESS.2022.3215778.
- [14] F. Ramadhani, U. Ramadhani, and L. Basit, “Combination of Hybrid Cryptography In *One time pad* (OTP) Algorithm And Keyed-Hash Message Authentication Code (HMAC) In Securing The Whatsapp Communication Application,” *J. Comput. Sci. Inf. Technol. Telecommun. Eng.*, vol. 1, no. 1, pp. 31–36, 2020, doi: 10.30596/jcositte.v1i1.4359.
- [15] I. Setiawan *et al.*, “Desain Kontrol Keamanan Pada Content Management System Wordpress Berdasar Aspek Aplikasi Dengan Panduan OWASP,” vol. 19, no. x, pp. 25–35, 2024.