

---

# **Jurnal** ***Rekayasa Elektrika***

---

VOLUME 12 NOMOR 1

APRIL 2016

---

**Peningkatan Sistem Keamanan Autentikasi Single Sign On (SSO)  
Menggunakan Algoritma AES dan One-Time Password Studi Kasus: SSO  
Universitas Ubudiyah Indonesia**

21-29

*Zuhar Musliyana, Teuku Yuliar Arif, dan Rizal Munadi*

---

JRE	Vol. 12	No. 1	Hal 1-40	Banda Aceh, April 2016	ISSN. 1412-4785 e-ISSN. 2252-620X
-----	---------	-------	----------	---------------------------	--------------------------------------

# Peningkatan Sistem Keamanan Otentikasi Single Sign On (SSO) Menggunakan Algoritma AES dan One-Time Password Studi Kasus: SSO Universitas Ubudiyah Indonesia

Zuhar Musliyana<sup>1</sup>, Teuku Yuliar Arif<sup>2</sup>, dan Rizal Munadi<sup>2</sup>

<sup>1</sup> Mahasiswa Magister Teknik Elektro, Pasca Sarjana, Universitas Syiah Kuala  
Program Studi Sistem Informasi, Fakultas Ilmu Komputer, Universitas Ubudiyah Indonesia  
Alue Naga-Tibang, Kec. Syiah Kuala, Banda Aceh 23114

<sup>2</sup> Wireless and Networking Research Group (Winner)  
Jurusan Teknik Elektro, Fakultas Teknik, Universitas Syiah Kuala  
Jl. Tengku Syech Abdul Rauf, No. 7, Darussalam, Banda Aceh 23111  
e-mail: zuhar@uui.ac.id

**Abstrak**—*Single Sign On (SSO)* merupakan model otentikasi independen yang diimplementasikan Universitas Ubudiyah Indonesia (UUI) menggunakan *Message-Digest Algorithm 5 (MD5)* dan *web service NuSOAP* berbasis bahasa pemrograman PHP. Sistem ini berjalan pada protokol *Hypertext Transfer Protocol (HTTP)*. Faktanya penggunaan protokol HTTP ini sangat rentan terhadap berbagai jenis serangan karena data dikirim dalam bentuk *plaintext* tanpa ada proses enkripsi dan penerapan algoritma MD5 pada otentikasi login juga rentan terhadap serangan *dictionary attacks* dan *rainbow tables*. Disisi lain, Penggunaan *web service NuSOAP* juga menciptakan celah keamanan karena pengiriman dan penerimaan *payload* tidak dienkripsi. Saat ini diketahui sudah ada beberapa metode yang dapat digunakan untuk meningkatkan pengamanan kerentanan tersebut diantaranya yaitu menggunakan *Hypertext Transfer Protocol Secure (HTTPS)*, *Secure Hypertext Transfer Protocol (SHTTP)* dan *Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA)*. Namun beberapa hasil penelitian terkait memperlihatkan masih terdapat beberapa kelemahan dari penggunaan HTTPS, SHTTP, dan CAPTCHA. Penelitian ini mengusulkan penggunaan algoritma *Advanced Encryption Standard (AES)* dengan pembangkit kunci dinamis dan metode *One-Time Password (OTP)* berbasis sinkronisasi waktu dengan kombinasi salt untuk meningkatkan keamanan pada otentikasi SSO UUI. Hasil pengujian menunjukkan penerapan algoritma AES dan OTP dapat mengamankan proses otentikasi SSO dari serangan *dictionary attack* dan *rainbow table*.

**Kata kunci:** *SSO, kunci dinamis, Salt, OTP, AES*

**Abstract**—SSO is an independent authentication model that implemented by UUI using MD5 and web- service NuSOAP based on PHP programming language. This system runs on the HTTP protocol. In fact, the use of HTTP still vulnerable to various types of attacks because the data sent in plaintext without encryption. Besides MD5 is also vulnerable to dictionary attacks and rainbow tables. On the other hand, using of NuSOAP also create the security holes because of the sending and receiving payload is not encrypted. Currently it has known there have been several methods that can be used to increase the security of these vulnerabilities among which uses HTTPS, SHTTP, and CAPTCHA. However, some related research results show there are still some weakness of the using of HTTPS, SHTTP, and CAPTCHA. This research proposes the use of the AES algorithm with dynamic key generation and development of OTP-based time synchronization with a combination of salt to increase the security of the SSO UUI authentication. The test results demonstrate the implementation of the AES algorithm and OTP managed to secure SSO authentication process of dictionary attack and rainbow table.

**Keywords:** *SSO, dynamic key, Salt, OTP, AES*

## I. PENDAHULUAN

*Single Sign On (SSO)* merupakan salah satu model otentikasi independen yang diimplementasikan kampus Universitas Ubudiyah Indonesia (UUI) dalam mengintegrasikan berbagai layanan aplikasi berbasis

web. Teknologi SSO memungkinkan pengguna dapat mengakses berbagai layanan aplikasi hanya dengan menggunakan satu *account* tunggal [1]. Kemudahan akses melalui single *account* perlu mendapatkan perhatian cermat untuk menjamin bukti-bukti otentikasi agar tidak tersebar dan diketahui pihak lain. Saat ini, sistem SSO UUI

dikembangkan berbasis bahasa pemrograman PHP dengan proses otentikasi *login* menggunakan *Message-Digest Algorithm 5* (MD5) sedangkan pada proses pengecekan sesi *login user* menggunakan *web service* NuSOAP.

Sistem SSO UUI dijalankan menggunakan protokol *Hypertext Transfer Protocol* (HTTP). Penggunaan protokol HTTP ini sangat rentan terhadap berbagai jenis serangan karena data dikirim dalam bentuk *plaintext* tanpa ada proses enkripsi. Berdasarkan data laporan yang diperoleh dari kampus UUI mengenai ancaman serangan yang terjadi terhadap *web server* SSO, selama periode Agustus 2014 hingga Mei 2015 telah terjadi sebanyak 888 ancaman serangan atau rata-rata 88 serangan per bulan [2]. Tipe-tipe serangan yang terjadi diantaranya adalah *SQL Injection*, *Denial of Service* (DOS), *Brute Force*, *Sniffing*, dan *Dictionary attack*. Serangan terhadap sistem SSO UUI tentu tidak dapat dibiarkan karena dapat berdampak sangat merugikan bagi Universitas Ubudiyah. Untuk itu diperlukan suatu metode pengamanan baru yang dapat meningkatkan keamanan sistem SSO UUI.

Dari hasil penelusuran literatur, saat ini diketahui sudah ada beberapa metode yang dapat digunakan untuk meningkatkan pengamanan pengiriman data menggunakan protokol HTTP. Metode pengamanan tersebut diantaranya yaitu menggunakan *Hypertext Transfer Protocol Secure* (HTTPS) [3], *Secure Hypertext Transfer Protocol* (SHTTP) [4], dan *Completely Automated Public Turing test to tell Computers and Humans Apart* (CAPTCHA) [5]. Namun beberapa hasil penelitian lainnya memperlihatkan masih terdapat beberapa kelemahan dari penggunaan HTTPS [6], SHTTP [7], dan CAPTCHA [8].

Pengamanan sebelumnya yang telah dilakukan pada level aplikasi SSO UUI adalah menggunakan algoritma MD5 pada proses otentikasi *login* dan *web service* NuSOAP pada pengecekan sesi *login user* SSO. Namun penelitian terkait juga menunjukkan penggunaan algoritma MD5 dan *web service* NuSOAP juga tidak menjamin keamanan data secara utuh [9][10].

Dari latar belakang dan permasalahan yang dikemukakan di atas, perlu dilakukan penelitian agar dapat meningkatkan sistem keamanan SSO UUI. Penelitian ini dilaksanakan di UUI Banda Aceh. Pada penelitian ini, penggunaan algoritma MD5 pada SSO UUI yang rentan terhadap serangan *dictionary attacks* dan *rainbow tables* akan diganti menggunakan algoritma kriptografi yang lebih kuat. Ada beberapa algoritma yang dapat digunakan, diantaranya *Data Encryption Standard* (DES), AES, *Triple Data Encryption Standard* (3DES), dan RSA. Pada penelitian ini digunakan algoritma *Advanced Encryption Standard* (AES) 128 bits. Pemilihan ini didasari atas pertimbangan AES memiliki keseimbangan antara keamanan dan fleksibilitas dalam berbagai *software* dan hardware. Selain itu AES juga merupakan salah satu jenis algoritma kriptografi kunci simetris yang dijadikan standar algoritma kriptografi sampai saat ini [11].

Kajian mengenai algoritma AES telah banyak dilakukan seperti optimasi *S-Box* untuk AES [12] dan pengembangan keamanan AES dengan menggunakan *Double S-Box* [13].

Penelitian lainnya yaitu pengembangan pembangkitan kunci dinamis AES dengan menggunakan fungsi waktu [14]. Pengembangan ini juga menggabungkan kombinasi nilai *salt* [15] untuk mengacak karakter kunci yang dihasilkan agar semakin kompleks dan menghasilkan kunci AES yang selalu berubah-ubah untuk setiap proses enkripsi dekripsi.

Pada penelitian ini, pembangkit kunci AES [15] akan digunakan untuk meningkatkan keamanan otentikasi *login* SSO UUI terhadap serangan *dictionary attacks* karena kunci AES yang dibangkitkan hanya dapat digunakan untuk satu kali proses enkripsi-dekripsi pada rentan waktu tertentu. Penerapan ini juga akan berdampak terhadap pencegahan serangan *rainbow tables* menggunakan database table *hash* pada kasus algoritma fungsi *hash* MD5 yang menggunakan enkripsi satu arah.

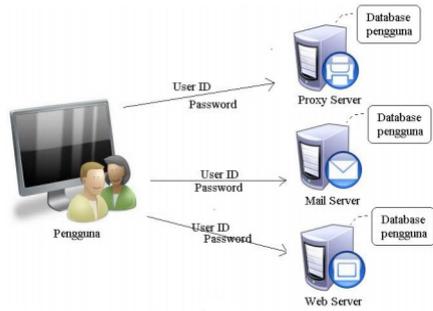
Pada proses pengecekan sesi *login user* SSO UUI yang sebelumnya menggunakan *web service* NuSOAP akan diganti menggunakan metode melalui otentikasi *One-time Password* (OTP) berbasis sinkronisasi waktu [16]. Penggunaan OTP sudah banyak diterapkan terutama pada remote *Virtual Private Network* (VPN), dan keamanan jaringan *Wireless Fidelity* (WiFi) dan juga pada berbagai aplikasi *Electronic Commerce* (*E-commerce*) [17]. Namun pada penelitian ini OTP yang diterapkan dikembangkan dengan kombinasi nilai *salt* untuk memperkuat pola otentikasi OTP. Metode ini akan membangkitkan sesi *login user* SSO secara acak dan hanya dapat digunakan untuk satu kali otentikasi berdasarkan nilai waktu.

Bagian selanjutnya dari paper ini dapat dijelaskan sebagai berikut. Pada bagian II dijelaskan studi pustaka terkait sistem SSO, algoritma AES, OTP, dan *salt*. Pada bagian III dijelaskan metode penelitian yang digunakan dalam paper ini. Pada bagian IV dijelaskan hasil penelitian dan pembahasan. Pada bagian V dijelaskan kesimpulan hasil penelitian.

## II. STUDI PUSTAKA

### A. Single Sign On (SSO)

SSO merupakan sistem otentikasi yang mengizinkan pengguna mengakses berbagai aplikasi dengan menggunakan satu credential tanpa harus *login* di masing-masing aplikasi [1]. SSO memiliki 2 bagian utama yaitu *Single Sign On* (*login* di satu aplikasi, maka aplikasi lain yang didefinisikan ikut dalam SSO otomatis akan dapat diakses) dan *Single Sign Out* (*logout* di satu aplikasi, maka semua aplikasi yang didefinisikan ikut dalam SSO akan *logout* secara otomatis [1]). Sistem ini tidak memerlukan interaksi yang manual, sehingga memungkinkan pengguna melakukan proses sekali *login* untuk mengakses seluruh layanan aplikasi tanpa berulang kali menginputkan *password*-nya. Teknologi ini sangat diminati dalam jaringan yang sangat besar dan bersifat heterogen, di mana sistem operasi serta aplikasi yang digunakan berasal dari banyak vendor, dan pengguna diminta untuk mengisi informasi dirinya ke dalam setiap multi-platform yang



Gambar 1. Sistem sign on [18]

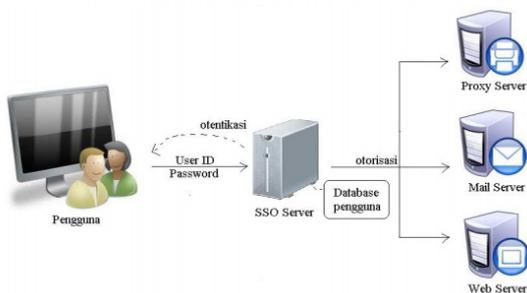
hendak diakses. Gambaran perbedaan sistem SSO dengan sistem login biasa tampak pada Gambar 1 dan Gambar 2.

**B. Algoritma Advanced Encryption Standard (AES)**

AES merupakan algoritma cipher yang cukup aman untuk melindungi data atau informasi yang bersifat rahasia. Pada tahun 2001, AES digunakan sebagai standar algoritma kriptografi terbaru yang dipublikasikan oleh National Institute of Standard and Technology (NIST) sebagai pengganti algoritma Data Encryption Standard (DES) yang sudah berakhir masa penggunaannya [6]. Algoritma AES adalah algoritma kriptografi yang dapat mengenkripsi dan mendekripsi data dengan panjang kunci yang bervariasi, yaitu 128 bit, 192 bit, dan 256 bit [6].

Proses enkripsi algoritma AES terdiri dari 4 jenis transformasi bytes, yaitu SubBytes, ShiftRows, MixColumns, dan AddRoundKey [6]. Awal proses enkripsi, input yang telah disalin ke dalam state akan mengalami transformasi byte AddRoundKey. Setelah itu, state akan mengalami transformasi SubBytes, ShiftRows, MixColumns, dan AddRoundKey secara berulang-ulang sebanyak Nr. Proses ini dalam algoritma AES disebut dengan round function. Round terakhir agak berbeda dengan round sebelumnya dimana pada round terakhir, state tidak mengalami transformasi MixColumns [8].

Pada proses dekripsi AES transformasi cipher dapat dibalikkan dan diimplementasikan dalam arah yang berlawanan untuk menghasilkan inverse cipher yang mudah dipahami untuk algoritma AES. Transformasi byte yang digunakan pada invers cipher adalah InvShiftRows, InvSubBytes, InvMixColumns, dan AddRoundKey.



Gambar 2. Sistem single sign on [18]

Algoritma 1. Algoritma pembangkit kunci dinamis enkripsi AES [14]

```

Algoritma Pembangkit Kunci Dinamis Enkripsi AES
1: Input: U (Username), P (Password), K (Kunci), Salt, WM (waktu_mulai)
2: WM = ambil waktu saat user login (Panjang Kunci 128 bit)
3: Salt = nilai salt (statik/dinamis) (Panjang Kunci 128 bit)
4: K1 = Salt ⊕ WM
5: K2 = Salt ⊕ P
6: K = K1 ⊕ K2
7: Output: CT (Chiphertext/Teks sandi)
8: (Nr, w) ← EkspansiKunci(K) {Nr: Jumlah Ronde, w: Larik Byte Kunci Ronde}
9: CT = P
10: AddRoundKey(CT, w[0..3])
11: for i=1 → Nr do
12:   SubBytes(CT)
13:   ShiftRows(CT)
14:   if i ≠ Nr then
15:     MixColumns(CT)
16:   end if
17:   AddRoundKey(CT, w[(i*4)..(i*4+3)])
18: end for
    
```

**C. Pembangkit Kunci dinamis AES**

Pada jenis algoritma AES kunci merupakan penentu dalam melakukan enkripsi-dekripsi. Selain itu panjang kunci juga menjadi indikator kompleksitas proses enkripsi yang ditawarkan. AES merupakan jenis algoritma enkripsi simetris yaitu algoritma yang menggunakan kunci yang sama untuk proses enkripsi dekripsi. Salah satu faktor penting untuk menjamin keamanan AES adalah kerahasiaan kunci yang digunakan. Agar kerahasiaan kunci dapat terjamin maka diperlukan sebuah konsep pembangkit kunci yang bersifat dinamis, namun kunci ini harus dapat dikenali saat proses enkripsi dekripsi [14].

Pada penelitian ini, sumber program enkripsi AES diambil dari forum developer PHP [19]. Selanjutnya pada pembangkit kunci dinamis AES, kunci enkripsi AES dibangkitkan dengan memanfaatkan pencampuran nilai waktu, salt dan password. Nilai waktu diambil pada saat user melakukan login. Sedangkan nilai salt yang digunakan yaitu nilai salt statis yang ditentukan langsung pada program. Pada proses dekripsi dilakukan sinkronisasi nilai waktu dengan batas toleransi tertentu untuk mendapatkan pasangan kunci yang sama dari nilai waktu yang dibangkitkan pada proses enkripsi. Adapun pseudocode algoritma pembangkit kunci dinamis enkripsi-dekripsi AES dapat dilihat pada Algoritma 1 dan 2.

Algoritma 2. Algoritma pembangkit kunci dinamis dekripsi AES [14]

```

Algoritma Pembangkit Kunci Dinamis Dekripsi AES
1: Input: CT(Chiphertext), K(Kunci), Salt, WM(waktu_mulai), P (Password), WP(waktu_proses), TW(toleransi_waktu), BW(batas_waktu), use_key(UK)
2: WP = ambil nilai waktu saat proses autentikasi
3: TW = 30 (detik) / optional
4: BW = WM+TW
5: UK = 0
6:
7: Output: P (Plaintext: Username dan Password)
8: while (WM ≤ BW) do
9:   if WM = WP
10:    K1 = Salt ⊕ WP
11:    K2 = Salt ⊕ P
12:    K = K1 ⊕ K2 (Panjang Kunci 128 bit)
13:    if use_key = 0
14:      (Nr, w) ← EkspansiKunci(K) {Nr: Jumlah Ronde, w: Larik Byte Kunci Ronde}
15:      P = CT
16:      AddRoundKey(P, w[(Nr-1)*4..(Nr-1)*4+3])
17:      for i=1 → Nr do
18:        InvSubBytes(P)
19:        InvShiftRows(P)
20:        AddRoundKey(P, w[(Nr-i)*4..(Nr-i)*4+3])
21:        if i ≠ Nr then
22:          InvMixColumns(P)
23:        end if
24:        AddRoundKey(CT, w[(i*4)..(i*4+3)])
25:      end for
26:      use_key++
27:    end if
28:    end if
29:    WM = WM+1
30:  end while
31:
    
```

#### D. Algoritma One-time Password (OTP)

OTP merupakan metode otentikasi yang menggunakan *password* yang selalu berubah setelah setiap kali *login*, atau berubah setiap interval waktu tertentu [16]. OTP dapat dibedakan atas dua kategori yaitu:

##### 1. OTP berbasis algoritma matematika

OTP jenis ini merupakan tipe lainnya dari OTP yang menggunakan algoritma matematika kompleks seperti fungsi *hash* kriptografi untuk membangkitkan *password* baru berdasarkan *password* sebelumnya dan dimulai dari kunci shared rahasia [16]. Contoh algoritma matematika yang digunakan dalam OTP ini adalah algoritma *open source* OATH yang telah distandardkan dan algoritma-algoritma lainnya yang telah dipatenkan. Beberapa produk aplikasi yang menggunakan otentikasi ini adalah:

##### a. CRYPTOCARD

CRYPTOCARD menghasilkan OTP baru setiap kali tombolnya ditekan. Sistem komputer akan menerima beberapa nilai balasan jika tombolnya ditekan lebih dari sekali secara tidak sengaja atau jika *client*-nya gagal mengotentikasi.

##### b. Verisign

*Verisign unified authentication* menggunakan standar dari OATH.

##### c. E-Token Aladdin Knowledge System NG-OTP

*E-token Aladdin knowledge system* NG-OTP merupakan *hybrid* antara USB dan token OTP *E-token Aladdin knowledge system* NG-OTP mengkombinasikan fungsionalitas dari token otentikasi yang berbasis *smart card* dan teknologi otentikasi *user one-time Password* dalam mode terpisah.

##### 2. OTP berbasis sinkronisasi waktu

OTP jenis ini berbasis sinkronisasi waktu yang berubah secara konstan pada setiap satuan interval waktu tertentu [16]. Proses ini memerlukan sinkronisasi antara token milik *client* dengan *server* otentikasi. Pada jenis token yang terpisah (*disconnected token*), sinkronisasi waktu dilakukan sebelum token diberikan kepada *client* [16]. Tipe token lainnya melakukan sinkronisasi saat token dimasukkan dalam suatu alat input. Di dalam token terdapat sebuah jam akurat yang telah disinkronisasikan dengan waktu yang terdapat pada *server* otentikasi. Pada sistem OTP ini, waktu merupakan bagian yang penting dari algoritma *password*, karena pembangkitan *password* baru didasarkan pada waktu saat itu dan bukan pada *password* sebelumnya atau sebuah kunci rahasia.

Pada penelitian terkait [17], OTP jenis ini sudah mulai diimplementasikan terutama pada *remote Virtual Private Network (VPN)*, dan keamanan jaringan Wi-Fi dan juga pada berbagai aplikasi *Electronic Commerce (E-commerce)*. Ukuran standar penggunaan waktu pada algoritma ini adalah 30 detik [17]. Nilai ini dipilih sebagai keseimbangan antara keamanan dan kegunaan.

Pada penelitian ini, OTP yang digunakan berbasis sinkronisasi waktu dengan kombinasi *salt*. OTP ini

#### Algoritma 3. Algoritma OTP UUI dengan kombinasi salt

```

Algoritma One-Time Password SSO UUI dengan Kombinasi Salt
1 Input: Salt, WM (waktu_mula), WP(waktu_proses), BW(batas_waktu), TW(tolransi_waktu), SIK(Sesi_id Kirim), SIP(Sesi_id Proses), Use_id
2 WM = ambil waktu saat user login
3 Salt = nilai salt (statik/dinamis)
4 TW = 30 (detik) / optional
5 Use_id = 0
6 Output: SI (Sesi_id) (Akses Diterima / Ditolak)
7 SIK = Salt + WM
8 WP = ambil nilai waktu saat proses autentikasi
9 BW = WM+TW
10 while (WM ≤ BW) do
11   if WM = WP
12     SIP = Salt + WP
13     if Use_id ≤ 0 and SIP = SI
14       Use_id++
15       out: Sesi_id valid akses diterima
16     end if
17   end if
18   WM = WM+1
19 end while

```

diterapkan pada pengecekan *sesi\_id user* SSO UUI untuk menghasilkan *sesi\_id user* yang selalu berubah berdasarkan nilai waktu. Kombinasi nilai *salt* digunakan untuk memperkuat pola OTP. Selain itu, nilai *salt* juga akan dicampur dengan nilai waktu untuk menghasilkan *sesi\_id* autentikasi yang dinamis sehingga setiap aplikasi yang terhubung dengan SSO akan mendapatkan nilai *sesi\_id* autentikasi yang berbeda-beda. Tidak hanya itu, *sesi\_id* yang dibangkitkan juga memiliki masa aktif tertentu. Setelah *sesi\_id* berhasil diverifikasi pada sisi *client* SSO, *sesi\_id* tersebut tidak dapat digunakan kembali untuk yang kedua kali. Secara lebih jelas, algoritma OTP dengan kombinasi nilai *salt* dapat dilihat pada Algoritma 3.

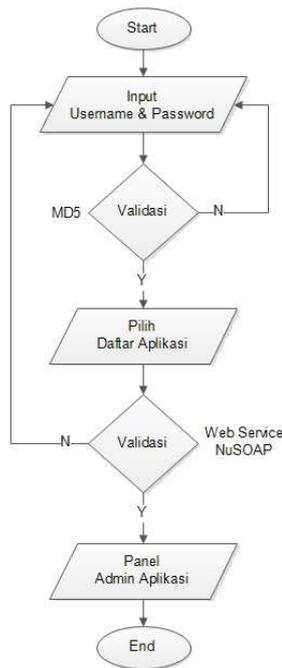
#### E. Salt

*Salt* adalah data atau teks yang dipakai untuk menyulitkan penyerang *password* [20]. Biasanya *salt* digunakan pada proses algoritma *hash* untuk dimasukkan ke dalam proses *hash* sebagai tambahan input. Hal ini menyebabkan nilai *hash* akan berubah jauh dari *hash* sebelumnya tanpa *salt*. *Salt* dapat dipilih tetap atau acak. Dengan *salt* maka penyerangan tidak dapat dilakukan secara paralel dengan *lookup password* dalam satu tabel. Tapi penyerang harus terlebih dahulu membangkitkan tabel untuk tiap-tiap *salt*.

Selain menyulitkan penyerang *password*, penerapan *salt* pada fungsi *hash* juga di klaim dapat menjaga karakter *password* yang disimpan karena akan selalu memiliki panjang karakter yang sama [21]. Namun tentu saja pada kasus ini, *user* tidak akan dapat mengetahui teks asli saat *password*-nya terlupa karena algoritma *hash* yang digunakan bersifat satu arah.

#### F. SSO Universitas Ubudiyah Indonesia

Sebagai salah kampus swasta di Aceh, UUI sangat gencar melakukan berbagai inovasi pemanfaatan media teknologi informasi dalam membantu proses penyelenggaraan pendidikan. UUI yang sebelumnya STIKES dan STMIK Ubudiyah, diresmikan bulan April 2014. Meski tergolong universitas baru, melalui direktorat *Cyber Development Center (CDC)* UUI telah mengembangkan berbagai sistem informasi berbasis web sebagai langkah besarnya dalam mencapai visi misi kampus menjadi *world class cyber university* di tahun 2025.



Gambar 3. Alur sistem SSO UUI

Dalam rangka memudahkan akses terhadap seluruh aplikasi yang telah dibangun, UUI telah membangun sebuah sistem untuk mengintegrasikan user pengguna seluruh aplikasi melalui satu *account* tunggal, sistem ini dinamakan *Single Sign On* (SSO). Sistem ini dibangun berbasis bahasa pemrograman PHP. Pemilihan bahasa pemrograman ini untuk memudahkan integrasi seluruh aplikasi lainnya yang dibangun dengan menggunakan bahasa pemrograman yang sama. Melalui SSO ini, akses ke seluruh aplikasi ditangani melalui satu pintu sehingga pengguna tidak perlu mengingat setiap *account login* untuk masing-masing aplikasi. Alur sistem SSO UUI dapat ditunjukkan pada Gambar 3.

Seperti yang ditunjukkan pada Gambar 3, alur otentikasi SSO UUI menggunakan algoritma MD5 untuk validasi *account login user*. Selanjutnya pada saat user mengakses daftar aplikasi yang telah terhubung dengan SSO, validasi sesi *login user* dilakukan menggunakan *web service* NuSOAP. *Web service* ini akan melakukan pemeriksaan nilai *sesi\_id user* pada server SSO, jika sesi user telah terbentuk maka user mendapatkan hak akses terhadap aplikasi tersebut.

### III. METODE

#### A. Metode Penelitian

Penelitian ini menggunakan metode penelitian kuantitatif dengan menganalisis penggunaan algoritma enkripsi terhadap keamanan otentikasi SSO UUI. Pengamatan yang akan dilakukan adalah menganalisis kelemahan otentikasi *login* UUI terhadap serangan *dictionary attacks* dan *rainbow table*. Pada pengecekan sesi *login user* dilakukan analisis paket data terhadap kemungkinan aksi *intercept* (serangan dimana pihak ketiga

menangkap pesan yang dikirimkan oleh *user A* tetapi pesan tersebut tetap dapat diterima oleh *user B* secara utuh) saat user mengakses aplikasi yang terdapat dalam SSO UUI.

#### B. Bahan Penelitian

Pada penelitian ini digunakan beberapa *software* pendukung diantaranya:

- Wireshark* sebagai *software* ini untuk menganalisis dan memeriksa aliran paket data saat komunikasi SSO berlangsung.
- Notepad ++* sebagai *software* editor programming untuk memudahkan pengetikan kode program.
- Linux Apache MySQL* dan PHP (LAMP) sebagai perangkat lunak untuk menjalankan aplikasi SSO UUI
- Burp suite* sebagai *software* untuk melakukan percobaan serangan *dictionary attack* pada otentikasi *login* SSO UUI.

#### C. Alat Penelitian

Peralatan pendukung yang digunakan pada penelitian ini adalah seperangkat komputer dengan spesifikasi cukup untuk menjalankan *software* LAMP, *Notepad++*, *Wireshark* dan *Burp suite*.

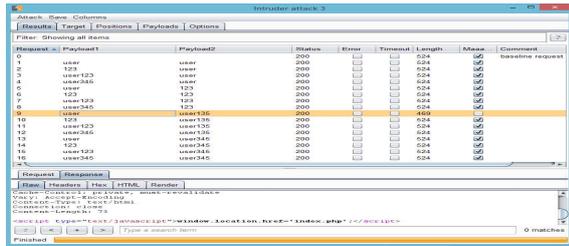
#### D. Prosedur Pengujian

Pada penelitian ini, pengujian dilakukan pada beberapa variabel berikut.

- Pengujian keamanan pada otentikasi *login* SSO terhadap serangan *Dictionary attacks* dan *Rainbow table*.

Pengujian *Dictionary attacks* dilakukan dengan mencoba segala kemungkinan *password* melalui daftar kata dari kamus. Pengujian ini dilakukan pada masing-masing metode otentikasi *login* sebelum dan setelah dilakukan peningkatan keamanan. Pengujian ini menggunakan *software burp suite* yang dijalankan pada komputer *client* SSO dengan memanfaatkan sample *username* dan *password* yang disimpan pada dua file terpisah yaitu *uUser.txt* untuk sample *username* dan *uPassword.txt* untuk sample *password*. Agar memudahkan dalam melakukan pengujian, kedua file tersebut disertakan pasangan *username* dan *password* yang valid. Kecocokan *username* dan *password* pada hasil pengujian ini dapat dilihat dengan mengecek variabel *match string* dan HTTP *response* dari tiap-tiap pasangan *username* dan *password* yang telah diuji.

Pengujian *Rainbow table* dilakukan untuk melihat kerentanan otentikasi algoritma MD5 berdasarkan nilai *hash* dari *account login user*. Pada pengujian ini, digunakan nilai *hash* dari user valid untuk memudahkan melakukan pengujian. Pengujian ini dilakukan untuk mendapatkan *plaintext* dengan mencari nilai *hash* yang telah disimpan dalam sebuah database tabel *hash*. Database tabel *hash* yang digunakan pada pengujian ini memanfaatkan



Gambar 4. Hasil percobaan dictionary attack sebelum penerapan AES database dari <https://crackstation.net> yang memuat sekitar 15 miliar data *lookup hash* MD5.

2. Pengujian keamanan pengecekan sesi *login user*

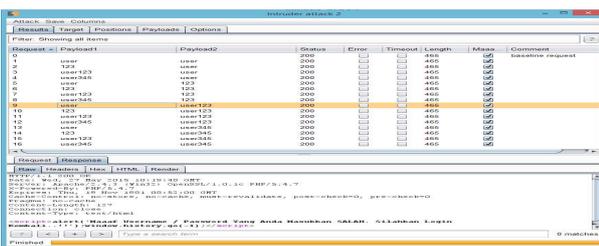
Pengujian ini dilakukan dengan menganalisis paket data pada proses pengecekan sesi *login user* SSO menggunakan *software wireshark* yang dijalankan pada sisi *client* untuk mendapatkan nilai *sesi\_id user*. Selanjutnya dilakukan percobaan *intercept* pada pengecekan sesi *login user* untuk mengetahui celah keamanan dari masing-masing metode sebelum dan setelah dilakukan peningkatan keamanan.

IV. HASIL DAN PEMBAHASAN

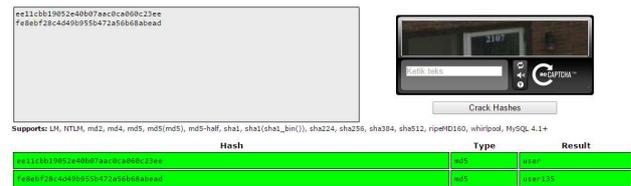
A. Hasil Pengujian Keamanan Otentikasi Login

Pada tahapan ini, pengujian dilakukan terhadap 2 jenis serangan yaitu *dictionary attack* dan *rainbow table*. Pengujian *dictionary attack* dilakukan menggunakan *software Burd Suite*. *Burd Suite* merupakan salah satu perangkat lunak yang paling populer digunakan sebagai *penetration test* melalui berbagai jenis metode penyerangan, salah satunya metode serangan *dictionary attack*. Pengujian ini memanfaatkan sample *username* dan *password* yang disimpan pada dua file terpisah yaitu *uUser.txt* untuk sample *username* dan *uPassword.txt*. Pada kedua file tersebut, disertakan pasangan *username* dan *password* yang valid untuk memudahkan dalam melakukan pengujian. Kecocokan *username* dan *password* pada hasil pengujian ini dapat ditemukan dengan melihat variabel *match string* dan *HTTP response* dari tiap-tiap pasangan *username* dan *password* yang telah diuji. Hasil pengamatan percobaan ini sebelum dan setelah dilakukan peningkatan keamanan diperlihatkan pada Gambar 4.

Gambar tersebut memperlihatkan hasil pengujian *dictionary attack* dengan 16 pola pasangan *username* dan *password*. Hasil pengujian ini menunjukkan serangan *dictionary attack* berhasil dilakukan. Kecocokan *username*



Gambar 5. Hasil percobaan dictionary attack setelah penerapan AES



Gambar 6. Rainbow table pada MD5

dan *password* ditemui pada pencocokan pola ke 16 dengan *username: user135* dan *password: user135*. Kecocokan ini dapat dilihat dari *response HTTP request* yang berbeda pada bagian *variable match string*.

Pada pengujian berikutnya, setelah dilakukan peningkatan keamanan menggunakan enkripsi AES dengan pembangkit kunci dinamis, serangan *Dictionary attack* berhasil digagalkan seperti pada Gambar 5.

Gambar 5 menunjukkan tidak ditemukannya *response HTTP request* yang berbeda pada bagian variabel *match string*. Padahal salah satu pola *username* dan *password* pada pengujian merupakan pola yang valid. Hal ini menunjukkan penerapan enkripsi AES dengan pembangkit kunci dinamis dapat mengatasi serangan *Dictionary attack*. Serangan *Dictionary attack* tidak berhasil pada kasus ini meskipun yang coba adalah *username* dan *password* yang valid. Hal ini dikarenakan pada sisi server nilai variabel *username* dan *password* yang diterima harus dalam bentuk *chiphertext*. Sehingga, agar serangan ini dapat bekerja *username* dan *password* yang dikirim dalam bentuk *chiphertext* yang tentu saja proses ini memerlukan kunci AES yang valid. Disisi lain, proses ini juga tidak dapat dilakukan karena penerapan enkripsi AES menggunakan pembangkit kunci dinamis artinya kunci AES yang digunakan akan selalu berubah untuk setiap proses enkripsi-dekripsi berdasarkan perubahan nilai waktu.

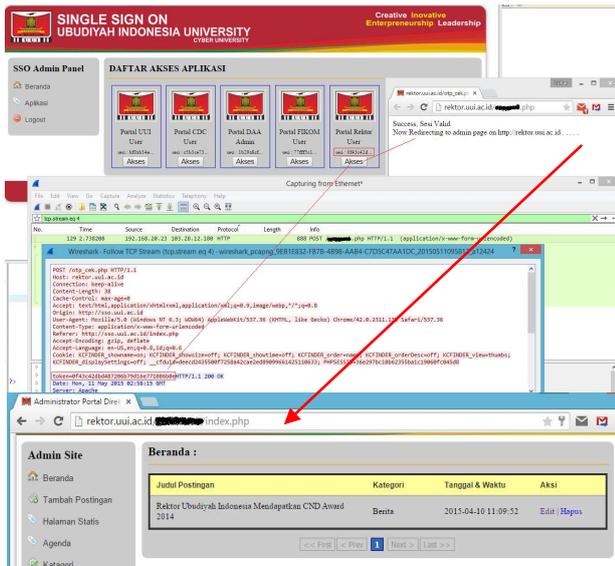
Pengujian selanjutnya dilakukan terhadap kerentanan serangan *rainbow table* pada algoritma MD5 yang digunakan pada otentikasi *login* SSO UII sebelum dilakukan peningkatan keamanan. Adapun *account user* pertama yang digunakan *username: user* dengan nilai *hash: ee11cbb19052e40b07aac0ca060c23ee* dan *password: user135* dengan nilai *hash: fe8ebf28c4d49b955b472a56b68abead*. Hasil percobaan tersebut dapat dilihat pada Gambar 6.

Berdasarkan Gambar 6, algoritma MD5 masih rentan terhadap serangan *rainbow table*, terbukti dengan *lookup table plaintext username* dan *password user* dapat ditemukan. Setelah dilakukan peningkatan keamanan melalui enkripsi AES seperti diperlihatkan pada Gambar 7, serangan *rainbow table* pada algoritma MD5 dapat diproteksi karena AES dengan pembangkit kunci dinamis

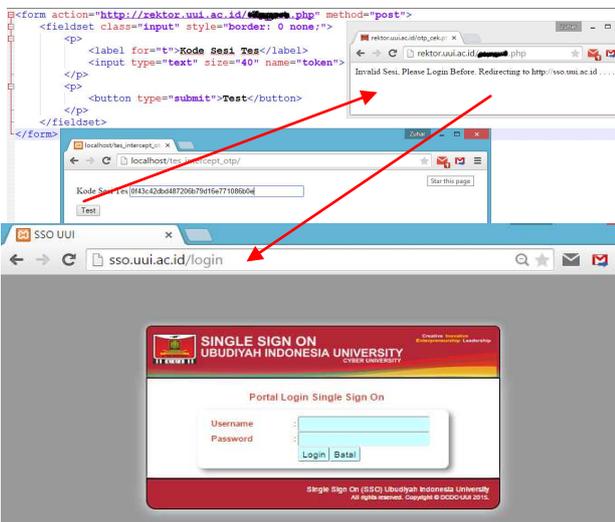


Gambar 7. Rainbow table pada AES





Gambar 11. Monitoring paket data pada metode one-time password



Gambar 12. Pengujian intercept otentikasi SSO setelah penerapan OTP

sistem SSO. Saat *user* yang sah melakukan *logout*, *sesi\_id user* yang sebelumnya telah terbentuk akan dihapus dari database server SSO.

Selanjutnya dilakukan pengujian yang sama setelah dilakukan peningkatan keamanan menggunakan otentikasi *One-time Password*. Hasil pengujian ini dapat ditunjukkan pada Gambar 11.

Berdasarkan Gambar 11, *sesi\_id* yang dikirim melalui metode *one-time Password* bersifat dinamis (masing-masing aplikasi mempunyai kode *sesi\_id* yang berbeda) dengan memanfaatkan kombinasi nilai *salt* dan nilai waktu. Seperti percobaan sebelumnya, dilakukan analisis paket data pada proses pengecekan *sesi\_id* aplikasi menggunakan *software wireshark* saat *user* mengakses portal web rektor melalui SSO. Tahapan berikutnya dilakukan percobaan *intercept* menggunakan *sesi\_id* yang telah didapatkan dari hasil analisis paket data menggunakan *wireshark*. Sama seperti percobaan sebelumnya, percobaan ini dilakukan

dengan mengirim nilai kode *sesi\_id* melalui form ke target action form file otentikasi *one-time Password* pada portal website rektor. Hasil percobaan tampak pada Gambar 12.

Gambar tersebut menunjukkan proses *intercept* gagal dilakukan karena kode sesi pada metode *one-time Password* hanya dapat digunakan untuk satu kali otentikasi.

## V. KESIMPULAN

Hasil penelitian menunjukkan penerapan algoritma AES dengan pembangkit kunci dinamis pada otentikasi SSO UII dapat mencegah serangan *dictionary attacks* dan *rainbow tables*. Di sisi lain, penerapan algoritma *One-time Password* dengan kombinasi *Salt* pada pengecekan *sesi\_id user* SSO dapat menanggulangi kemungkinan aksi *intercept* terhadap penyalahgunaan hak akses *user* SSO karena kode *sesi\_id user* yang dikirim bersifat dinamis dan hanya dapat digunakan satu kali otentikasi dengan batas waktu tertentu.

## REFERENSI

- [1] K.D. Lewis, "Web Single Sign-On Authentication using SAML," International Journal of Computer Science Issues (IJCSI), Vol. 2, Aug. 2009.
- [2] F. Heriadi, "Laporan Ancaman Serangan Web server Universitas Ubudiyah Indonesia (UII) 2014-2015," DCDC UII, Lap. 08.2015, 2015.
- [3] E. Rescorla, A. Schiffman. (1999, Ags.). The Secure Hypertext Transfer Protocol. The Internet Engineering Task Force (IETF), California, USA. [Online]. Available: <https://www.ietf.org/rfc/rfc2660.txt>
- [4] E. Rescorla, A. Schiffman. (1999, Ags.). The Secure Hypertext Transfer Protocol. The Internet Engineering Task Force (IETF), California, USA. [Online]. Available: <https://www.ietf.org/rfc/rfc2660.txt>
- [5] H. Tschofenig, E. Leppanen, S. Niccolini, M. Arumaiturai (2008, Feb.). Completely Automated Public Turing test to tell Computers and Humans Apart (CAPTCHA) based Robot Challenges for SIP. The Internet Engineering Task Force (IETF), California, USA. [Online]. Available: <https://tools.ietf.org/id/draft-tschofenig-shipping-captcha-01.txt>
- [6] J. Kirk (2007, Mei.). Researcher: RSA 1024-bit Encryption Not Enough. IDG Consumer & SMB, San Francisco, USA. [Online]. Available: <http://www.pcworld.com/article/132184/article.html>
- [7] Adam S., (1999, Mei.). An Overview of SHTTP. National Academy of Sciences Cryptography. USA. [Online]. Available: <http://www.homeport.org/~adam/shttp.html>
- [8] Ahmad B. A, "Bypassing Captcha by Machine-A Proof for Passing the Turing Test," European Scientific Journal, Vol. 10 No.15, Mei. 2014
- [9] M.C. Ah Kioon, Z. Wang and S.D. Das, "Security Analysis of MD5 algorithm in Password Storage," Proceedings of the 2nd International Symposium on Computer, Communication, Control and Automation (ISCCCA-13), 2013
- [10] R. T. A. Fadlan, "Implementasi Algoritma Rijndael pada NuSOAP," Skripsi, Lab. Komputer, STEI ITB, Bandung, Indonesia, 2011.
- [11] R. Sadikin, Kriptografi untuk keamanan jaringan, 1st Ed.

- Yogyakarta, Indonesia: Penerbit Andi Offset, 2012
- [12] Sahoo, O.B.; Kole, D.K.; Rahaman, H., "An Optimized S-Box for Advanced Encryption Standard (AES) Design," International Conference of Advances in Computing and Communications (ICACC), 11 Ags. 2012.
- [13] A. Kumar et al, "AES Security Enhancement by Using Double S-Box," International Journal of Computer Science and Information Technologies (IJCSIT), Vol. 3, Mei. 2012
- [14] Z. Musliyana, T.Y. Arif, R. Munadi, "Security Enhancement of Advanced Encryption Standard (AES) using Time-Based Dynamic Key Generation," ARPN Journal of Engineering and Applied Sciences, Vol. 10, No. 18, Oct. 2015
- [15] Vishwakarma, D.; Madhavan, C.E.V., "Efficient dictionary for salted password analysis," in Electronics, Computing and Communication Technologies (IEEE CONECCT), 2014 IEEE International Conference on, Jan. 2014
- [16] Hyun-Chul Kim; Lee, H.-W.; Young-Gu Lee; Moon-Seog Jun, "A Design of *One-time Password* Mechanism Using Public Key Infrastructure," Fourth International of Networked Computing and Advanced Information Management, Sep. 2008
- [17] D. M'Raihi, S. Machani, M. Pei, J. Rydell. (2011, Mei). TOTP: Time-Based *One-time Password* Algorithm. The Internet Engineering Task Force (IETF), California, USA. [Online]. Available: <https://tools.ietf.org/html/rfc6238>
- [18] G. Ramadhan, "Analisis teknologi Single Sign On (SSO) dengan penerapan Central Authentication Service (CAS) pada Universitas Bina Darma," Skripsi, Lab. Komputer, UBD, Palembang, Indonesia, 2012.
- [19] J. M. Busto. (2007, Mei). AES 128: A pure PHP AES 128 encryption implementation. Icontem, USA. [Online]. Available: <http://www.phpclasses.org/package/3650-PHP-A-pure-PHP-AES-128-encryption-implementation.html>
- [20] Gauravaram, P., "Security Analysis of salt || password Hashes," International Conference Advanced Computer Science Applications and Technologies (ACSAT), 26-28 Nov. 2012.
- [21] P. Ducklin. (2013, Nov.). Anatomy of a password disaster - Adobe's giant-sized cryptographic blunder. Sophos Ltd, Boston, USA. [Online]. Available: <https://nakedsecurity.sophos.com/2013/11/04/anatomy-of-a-password-disaster-adobes-giant-sized-cryptographic-blunder/>

**Penerbit:**

Jurusan Teknik Elektro, Fakultas Teknik, Universitas Syiah Kuala

Jl. Tgk. Syech Abdurrauf No. 7, Banda Aceh 23111

website: <http://jurnal.unsyiah.ac.id/JRE>

email: [rekayasa.elektrika@unsyiah.net](mailto:rekayasa.elektrika@unsyiah.net)

Telp/Fax: (0651) 7554336

