PENGAMANAN DATA *CLOUD STORAGE* DENGAN MENGGUNAKAN *ADVANCED ENCRYPTION STANDARD DAN ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM* PADA *SECURE SOCKET LAYER* BERBASIS *WEBSITE*

SECURING CLOUD STORAGE DATA USING ADVANCED ENCRYPTION STANDARD AND ELLIPTIC CURVE DIGITAL SIGNATURE ALGORITHM ON SECURE SOCKET LAYER BASED ON WEBSITE

Muhammad Ariq Mahardhika[1], Dr Yudha Purwanto S.T., M.T.[2], Muhammad Faris Ruriawan S.T., M.T[3]
[1,2,3]Prodi S1 Teknik Komputer, Fakultas Teknik Elektro, Universitas Telkom
[1]**mariqmar@student.telkomuniversity.ac.id**, [2]**omyudha@telkomuniversity.co.id**,
[3]**muhammadfaris@telkomuniversity.ac.id**

Abstrak
Cloud Storage adalah media penyimpanan *file* berbasis *online* atau digital yang mengandalkan koneksi internet untuk akses *data. Cloud storage* adalah terobosan baru dalam dunia penyimpanan data yang menonjolkan banyak kelebihan jika dibandingkan dengan media penyimpanan *offline* seperti *hardisk* dan *flashdisk*. Data yang disimpan ke dalam *cloud storage* akan disimpan di sejumlah *server* yang dikelola oleh pihak penyedia layanan atau yang biasa disebut juga dengan *hosting*. *Eliptic Curve Digital Signature Algorithm* adalah implementasi kurva elips dari DSA. Secara fungsional saat RSA dan DSA memerlukan panjang kunci 3072 bit untuk memberikan keamanan 128 bit, ECDSA dapat mencapai hal yang sama hanya dengan kunci 256-bit. Namun, ECDSA bergantung pada tingkat keacakan yang sama seperti DSA.

Kata Kunci: *Cloud storage, server, cloud, Eliptic Curve Digital Signature Algorithm, Owncloud***.**

Abstract
Cloud Storage is an online or digital file storage media that relies on an internet connection for data access. Cloud storage is a new breakthrough in the world of data storage that features many advantages when compared to offline storage media such as hard drives and flash drives, so data stored in cloud storage will be stored on a number of servers managed by service providers or commonly known as hosting. Eliptic Curve Digital Signature Algorithm is an implementation of the elliptic curve of the DSA. Functionally where RSA and DSA require a key length of 3072 bits to provide 128 bit security, ECDSA can achieve the same thing with only 256-bit keys. However ECDSA relies on the same degree of randomness as DSA.

Keywords: *Cloud storage, server, cloud, Eliptic Curve Digital Signature Algorithm, Owncloud.*

1. Pendahuluan

Dengan semakin banyaknya jumlah data yang disimpan di cloud storage, masalah keamanan data dan privasi menjadi semakin penting. Penyedia *cloud storage* dapat dipercaya karena mereka memiliki mekanisme keamanan yang memadai untuk melindungi data pengguna dari seorang peretas atau disebut juga dengan *hacker*. Oleh karena itu, sangat penting bagi pengguna *cloud storage* untuk mengevaluasi keamanan penyedia *cloud storage* [1]. Sangat penting bagi penyedia jasa untuk menjaga situs dengan mengenkripsi melalui SSL atau TLS. *Elliptic curve cryptography* (ECC) adalah salah satu teknologi yang lebih menjanjikan di bidang TLS ini, ECC berkemampuan lebih cepat dan lebih skalabilitas di server ini, dan memberikan keamanan lebih daripada kriptografi biasa yang digunakan di web lain [2]. Keamanan data merupakan masalah penting yang harus segera dipecahkan untuk teknologi penyimpanan *cloud*. Dalam beberapa tahun terakhir, semakin banyak serangan terjadi pada sistem *cloud storage*, dan kebocoran data yang sering terjadi. Advanced Encryption Standard (AES) adalah cipher blok simetris yang dipilih oleh

pemerintah AS untuk melindungi informasi rahasia. AES diimplementasikan dalam perangkat lunak dan perangkat keras di seluruh dunia untuk mengenkripsi data sensitive [3].

2. Dasar Teori

2.1 *Cloud storage*

Cloud storage adalah metode penyimpanan data di sejumlah server yang dikelola pihak penyedia layanan. Koneksi internet dibutuhkan untuk mengakses data yang tersimpan di server. Metode ini memiliki sejumlah keunggulan dibanding penyimpanan data pada memori fisik konvensional, yakni data dapat diakses menggunakan berbagai perangkat, selama terkoneksi dengan jaringan internet. Pengguna media *cloud storage* dapat membagi data dalam server ke pihak-pihak lain dengan cepat dan mudah, serta keamanan dan keutuhan data terjamin. Sejumlah layanan penyimpanan cloud terkemuka menawarkan layanan masing-masing secara gratis untuk digunakan, apabila membutuhkan kapasitas penyimpanan yang lebih besar terdapat versi berbayar yang bisa digunakan oleh pengguna jasa *cloud storage*.

2.2 *Eliptic Curve Digital Signature Algorithm*

ECDSA adalah salah satu tipe digital signature dari ECC yang memberikan layanan data *origin authentication*, *data integrity* dan *non-repudiation*. ECDSA diperkenalkan pertama kali pada tahun 1992 oleh Scott Vanstone dan pada tahun 1998 mendapatkan standar ISO (*International Standards Organization*) yaitu ISO 14888-3. Tahun 1999 diterima sebagai standar ANSI (*American National Standards Institute*) ANSI X9.62 dan tahun 2000 sebagai standar IEEE (*Institute of Electrical and Electronics Engineers*) IEEE 1363-2000 serta standar NIST (*National Institute of Standards and Technology*) yaitu FIPS 186-2. ECDSA (*Elliptic Curve Digital Signature Algorithm*), atau ECC (*Elliptic Curve Cryptography*) seperti yang terkadang dikenal, adalah penerus algoritma DSA (*Digital Signature Algorithm)*. ECDSA lahir ketika dua matematikawan bernama Neal Koblitz dan Victor S. Miller mengusulkan penggunaan kurva elips dalam kriptografi. Namun, butuh waktu hampir dua dekade untuk algoritma ECDSA menjadi standar. ECDSA adalah algoritma kriptografi asimetris yang dibangun di sekitar kurva elips dan fungsi dasar yang dikenal sebagai *trapdoor function*. Kurva elips mewakili himpunan titik yang memenuhi persamaan matematika ($y^2 = x^3 + \text{ax} + \text{b}$). Seiring berjalannya semua algoritme asimetris, ECDSA bekerja dengan cara yang mudah dihitung dalam satu arah, tetapi sangat sulit untuk dikembalikan. Dalam kasus ECDSA, angka pada kurva dikalikan dengan angka lain sehingga menghasilkan titik pada kurva.

2.3 *Advanced Encryption Standard Algorithm*

Advanced Encryption Standard (AES) adalah cipher blok simetris yang dipilih oleh pemerintah AS untuk melindungi informasi rahasia. AES diimplementasikan dalam perangkat lunak dan perangkat keras di seluruh dunia untuk mengenkripsi data sensitif. Ini penting untuk keamanan komputer pemerintah, *cyber security*, dan perlindungan data elektronik. *National Institute of Standards and Technology* (NIST) memulai pengembangan AES pada tahun 1997 ketika mengumumkan perlunya alternatif untuk *Data Encryption Standard* (DES), yang mulai menjadi rentan terhadap serangan brute force. NIST menyatakan bahwa algoritma enkripsi canggih yang lebih baru tidak akan diklasifikasikan dan harus mampu melindungi informasi sensitif pemerintah hingga abad ke-21. Itu dimaksudkan agar mudah diterapkan di perangkat keras dan perangkat lunak, serta di lingkungan terbatas seperti *smart card* dan menawarkan pertahanan yang layak terhadap berbagai teknik serangan. AES dibuat untuk pemerintah A.S. dengan tambahan sukarela, penggunaan gratis di program publik atau swasta, komersial atau nonkomersial yang menyediakan layanan enkripsi. Namun, organisasi nonpemerintah yang memilih untuk menggunakan AES tunduk pada batasan yang dibuat oleh kontrol ekspor A.S

2.4 *Secure Socket Layer* (SSL)

SSL (*Secure Socket Layer*) adalah lapisan keamanan untuk melindungi transaksi di *website* dengan teknologi enkripsi data yang canggih. *Website* yang menggunakan SSL, alamatnya berubah menjadi https dan muncul tanda *padlock* (gembok) di *address bar browser*, dan bisa di tekan untuk melihat jenis SSL, teknologi enkripsi yang dipakai dan siapa identitas pemilik *website*. SSL memiliki banyak manfaat dalam penggunaannya, salah satunya yaitu mampu mengamankan transaksi yang ada di *website* dengan enkripsi termasuk data kartu kredit serta *password*.

2.5 FTP (*File Transfer Protocol*)

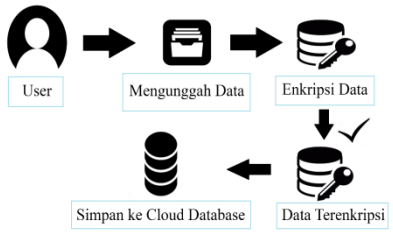
File Transfer Protocol adalah protokol jaringan standar yang digunakan untuk mengirim data komputer antara klien dan server di jaringan komputer. FTP dibangun di atas arsitektur model klien – server menggunakan kontrol terpisah dan koneksi data antara klien dan server, dengan menggunakan FTP maka pengguna dapat mengirim dan mengunduh file. Selain itu FTP digunakan sebagai *remote* (pengendali) *server* yang mengelola web tanpa perlu *log in* (masuk) ke dalam akun hosting atau VPS (*Virtual Private Server*). Pengguna FTP dapat mengesahkan diri dengan protokol *clear-text sign-in* yang biasanya berbentuk nama pengguna dan kata sandi, selain juga bisa terhubung secara anonym jika server konfigurasi mengizinkannya. Demi transmisi yang aman dan melindungi nama pengguna beserta kata sandinya juga mengenkripsi konten, FTP sering diamankan dengan SSL (*Secure Socket Layer*).

2.6 HTTPS (*Hypertext Transfer Protocol Secure)*

Deskripsi: HTTPS memastikan keamanan data melalui jaringan - terutama jaringan publik seperti Wi-Fi. HTTP tidak dienkripsi dan rentan terhadap penyerang yang menguping dan dapat memperoleh akses ke *database* situs *website* dan informasi sensitif. Berdasarkan prinsipnya, enkripsi HTTPS dilakukan secara dua arah, yang berarti bahwa data dienkripsi di sisi klien dan server. Hanya klien yang dapat memecahkan kode informasi yang berasal dari server. Jadi, HTTPS melakukan enkripsi data antara klien dan server, yang melindungi dari penyadapan, pemalsuan informasi, dan gangguan data. Namun, desain ini dapat berbeda untuk browser yang berbeda. Misalnya, pertimbangkan untuk mengunjungi situs web bank, misalnya hdfcbank.com. HTTP tidak aman akan terbuka. Tapi saat kita masuk ke halaman login, kita bisa melihat HTTPS di address bar dengan beberapa desain khusus. Implementasi: HTTPS terutama digunakan oleh situs web yang menangani transaksi moneter atau mentransfer data pribadi pengguna yang mungkin sangat sensitif. Situs web perbankan adalah contoh yang umum. Dalam istilah awam, HTTPS memastikan bahwa pengguna menonton situs web yang ingin mereka tonton. Data yang dipertukarkan antara pengguna dan situs web tidak dibaca, dicuri, atau dirusak oleh pihak ketiga. Tetapi tidak dapat mengenkripsi semuanya, ia juga memiliki beberapa batasan. Misalnya, HTTPS tidak dapat mengenkripsi alamat host dan nomor portal.

3. Perancangan dan Implementasi

3.1 Gambaran Umum Sistem

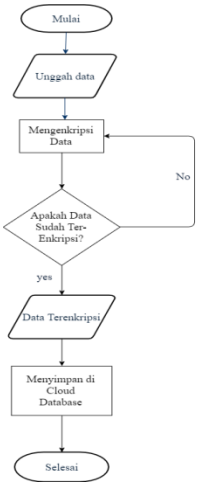


Gambar 3.1 Gambaran umum sistem.

Gambaran umum dari sistem yang digunakan pada aplikasi ini adalah seperti pada Gambar 3.1. Tahap-tahap dari sistem tersebut adalah sebagai berikut :

1. Jaringan diamankan dengan secure socket layer ECDSA.
2. Sistem dimulai dengan proses awal *user* melakukan pengambilan data.
3. Data telah diambil dan akan melakukan proses mengunggah data.
4. Data yang telah diunggah tadi akan dilanjutkan pada tahap enkripsi data menggunakan AES.
5. Setelah data melewati validasi enkripsi AES, selanjutnya kode validasi untuk data tersebut akan tersimpan.

3.2 Perancangan Sistem

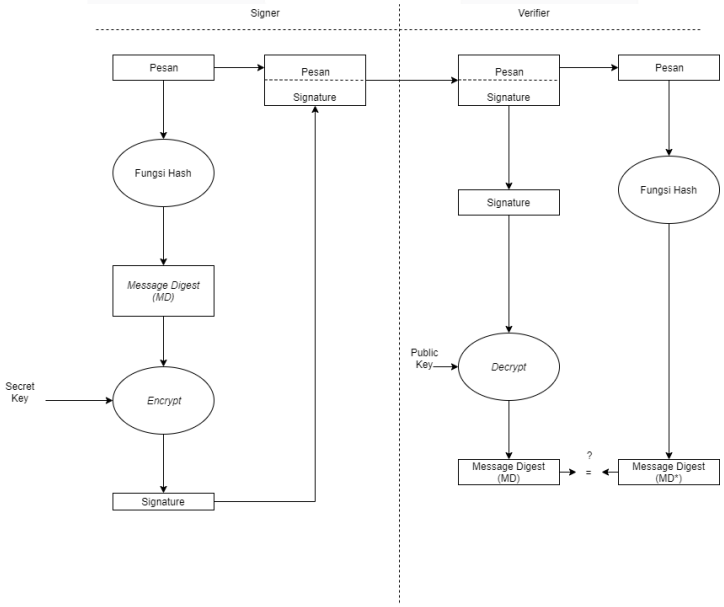


Gambar 3.2 Flowchart pengamanan data di cloud storage.

Perancangan sistem pada gambar di atas sebagai berikut :

1. Memulai dengan memasukkan data ke dalam *cloud storage.*
2. Lalu data dienkripsi dengan menggunakan *Advanced Encryption Standard.*
3. Lalu data dicek oleh sistem sudah terenkripsi dengan benar, kalau belum akan dienkripsi lagi.
4. Apabila data sudah terenkripsi dengan baik akan langsung masuk ke *cloud storage*.

3.3 Klasifikasi dari Algoritma ECDSA



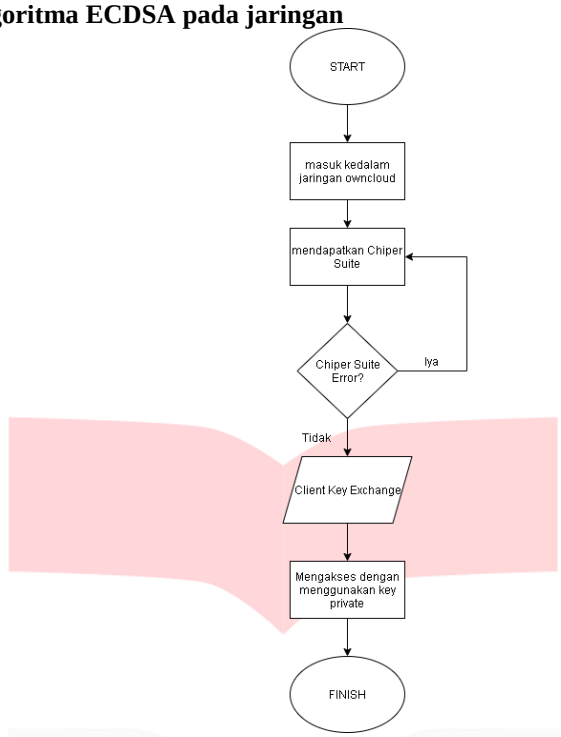
Gambar 3.3 Klasifikasi jaringan dari algoritma ECDSA.

Proses klasifikasi jaringan dari algoritma ECDSA di atas sebagai berikut :

1. Pengirim menghitung nilai hash dari pesan M yang akan dikirim.
2. Pengirim melakukan enkripsi terhadap nilai hash h menggunakan kunci privasi dari ECC.
3. Pengirim mengirimkan pesan M dan hasil enkripsi ECC dari hasil hash pesan ke penerima.
4. Penerima memisahkan antara pesan dan tanda tangan.
5. Penerima melakukan dekripsi tanda tangan menggunakan kunci public pengirim.

6. Penerima melakukan hash terhadap pesan yang diterima.
7. Hasil dari nomor 2 dan 3 dibandingkan, apabila hasilnya sama maka data yang dikirim valid.

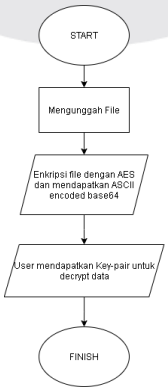
3.4 Implementasi Algoritma ECDSA pada jaringan



Gambar 3.4 Implementasi algoritma ECDSA pada jaringan.

Klien melakukan *Handshake* berbasis ECC secara umum, sama dengan Handshake berbasis RSA lewat 2 pesan pertama. Klien dan server bernegosiasi mengenai chiper suite ECC. Pesan *certificate* pada server berisi kunci publik ECDH, server yang ditandatangani oleh otoritas *certificate* yang menggunakan ECDSA. Setelah memvalidasi tandatangan ECDSA, klien mengirim kunci publik ECDH-nya ke *server* pada pesan *Client Key Exchange*. Selanjutnya, baik klien dan *server* menggunakan kunci *private* ECDH masing-masing dan kunci publik yang diterimanya tadi untuk melakukan sebuah operasi ECDH yang menghasilkan *pre-master secret*. Penurunan *master secret* dan kunci-kunci simetris, serta operasi sisanya sama seperti *handshake* pada RSA.

3.5 Implementasi Algoritma AES pada penyimpanan cloud storage



Gambar 3.5 Implementasi Algoritma AES pada cloud storage

Algoritma AES menghasilkan pasangan kunci public atau pribadi yang kuat, sebanyak 4096 bit untuk setiap pengguna. Kunci pribadi dienkripsi dengan kata sandi login pengguna menggunakan AES-256. Lalu file membuat kunci ASCII *encoded base64* 32 *byte* untuk setiap *file*,

lalu enkripsi *file* dengan *File-key* menggunakan AES-256. Dekrip kunci *file* dengan *private-key* dan *share-key* yang cocok, lalu dekrip file menggunakan kunci file AES-256, yang memiliki panjang kunci 256 bit, mendukung ukuran bit terbesar dan secara praktis tidak dapat dipecahkan oleh *brute force* berdasarkan daya komputasi saat ini, menjadikannya standar enkripsi terkuat. Tabel 3.1 menunjukkan bahwa kemungkinan kombinasi tombol meningkat secara eksponensial dengan ukuran kunci.

Key Size	***Possible Combinations***
1 *bit*	2
2 *bits*	4
4 *bits*	16
8 *bits*	256
16 *bits*	65536
32 *bits*	$4.2 \; x \; 10^{9}$
56 *bits* (DES)	$7.2 \; x \; 10^{19}$
64 *bits*	$1.8 \; x \; 10^{19}$
128 *bits* (AES)	$3.4 \; x \; 10^{38}$
192 *bits* (AES)	$6.2 \; x \; 10^{57}$
256 *bits* (AES)	$1.1 \; x \; 10^{77}$

Ukuran kunci yang digunakan untuk sandi AES menentukan jumlah putaran transformasi yang mengubah masukan, yang disebut *plaintext*, menjadi keluaran akhir yang disebut *ciphertext*. Jumlah putarannya 10 *rounds* untuk 128-*bit keys*, 12 *rounds* untuk 192-*bit keys*, 14 *rounds* untuk 256 *bits*. Setiap putaran terdiri dari beberapa langkah pemrosesan, termasuk langkah yang bergantung pada kunci enkripsi itu sendiri. Satu set putaran terbalik diterapkan untuk mengubah ciphertext kembali menjadi plaintext menggunakan kunci enkripsi yang sama.

4 Pengujian dan Analisis

4.1Pengujian

Pengujian yang dilakukan pada sub-bab ini sistem yang terdiri dari pengujian keamanan pada jaringan *website*, pengujian keamanan data pengguna, dan pengujian keamanan data di *cloud storage*.

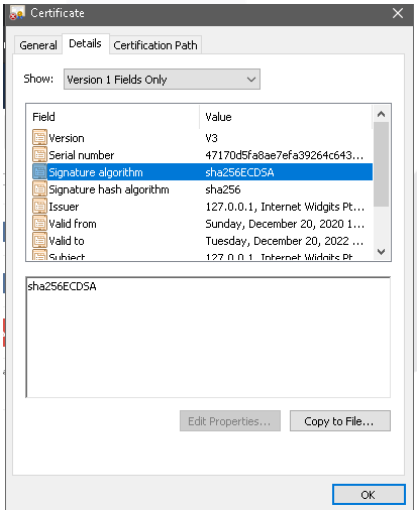
TABEL 4.1 Metode *Black Box*

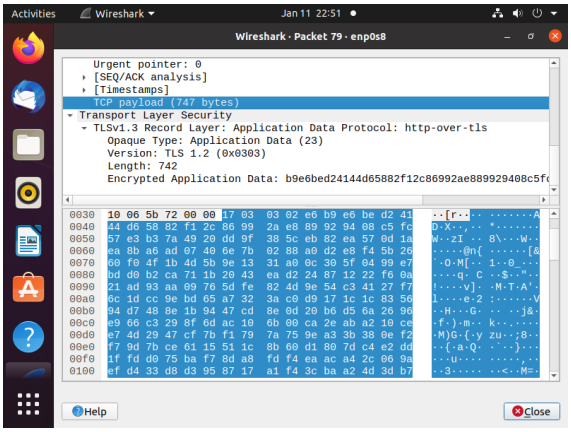
No	Pengujian	Tujuan yang tercapai	Hasil yang didapatkan	Berhasil atau Tidak	Keterangan
1	Login				
	- Username dan password di isi	Dapat masuk kedalam website	Dapat masuk kedalam website	Berhasil	Valid
	- Username saja Password kosong	Tidak dapat masuk dan masukkan username dan password kembali	Tidak dapat Masuk dan masukkan username dan password ulang	Berhasil	Valid
	- Password saja	Tidak dapat	Tidak dapat	Berhasil	Valid

	Username kosong	masuk dan masukkan username dan password kembali	Masuk dan masukkan username dan password ulang		
2	Upload file				
	Dengan menekan tombol “+”	Data ter upload di storage	Data ter upload di storage	Berhasil	Valid
	- Dengan drag file kedalam website	Data ter upload di storage	Data ter upload di storage	Berhasil	Valid
3	Menghapus File				
	- Dengan menekan sekali pada icon file	File tertandai dan dapat di hapus	File tertandai dan terhapus	Berhasil	Valid
	- Dengan menekan “Delete files”	File tertandai dan dapat di hapus	File tertandai dan terhapus	Berhasil	Valid

4.2 Pengujian Keamanan Pada Jaringan *Website dan di cloud storage*

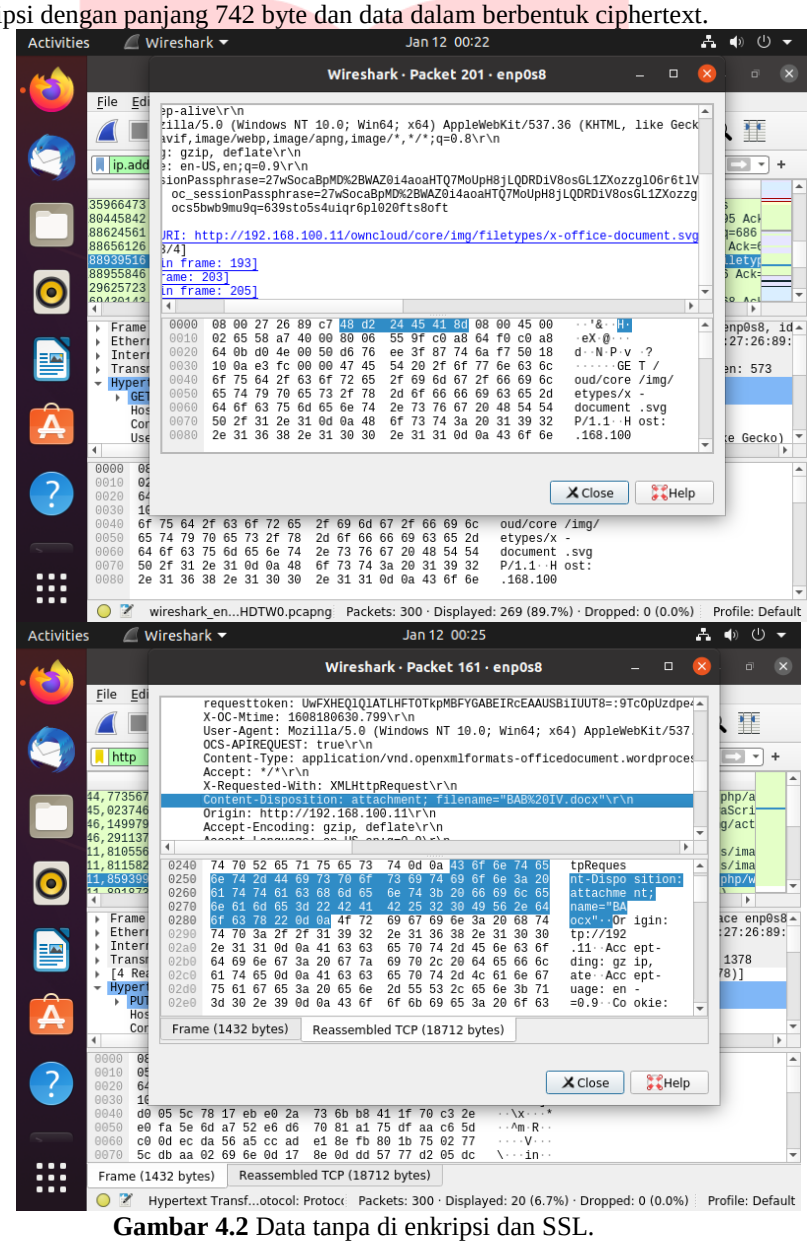
Pengujian ini dilakukan agar mengetahui jalur data website cloud storage sudah terenkripsi jaringannya dengan benar.





Gambar 4.1 *Certificate* dan *ciphertext* dan data terenkripsi

Jaringan sudah tersertifikasi dan sudah terenkripsi dengan *Transport Layer Security* (TLS) dengan panjang 747 *bytes* dan file yang sudah di *upload* atau *download* dari dalam cloud storagenya sudah melakukan enkripsi dengan panjang 742 byte dan data dalam berbentuk ciphertext.

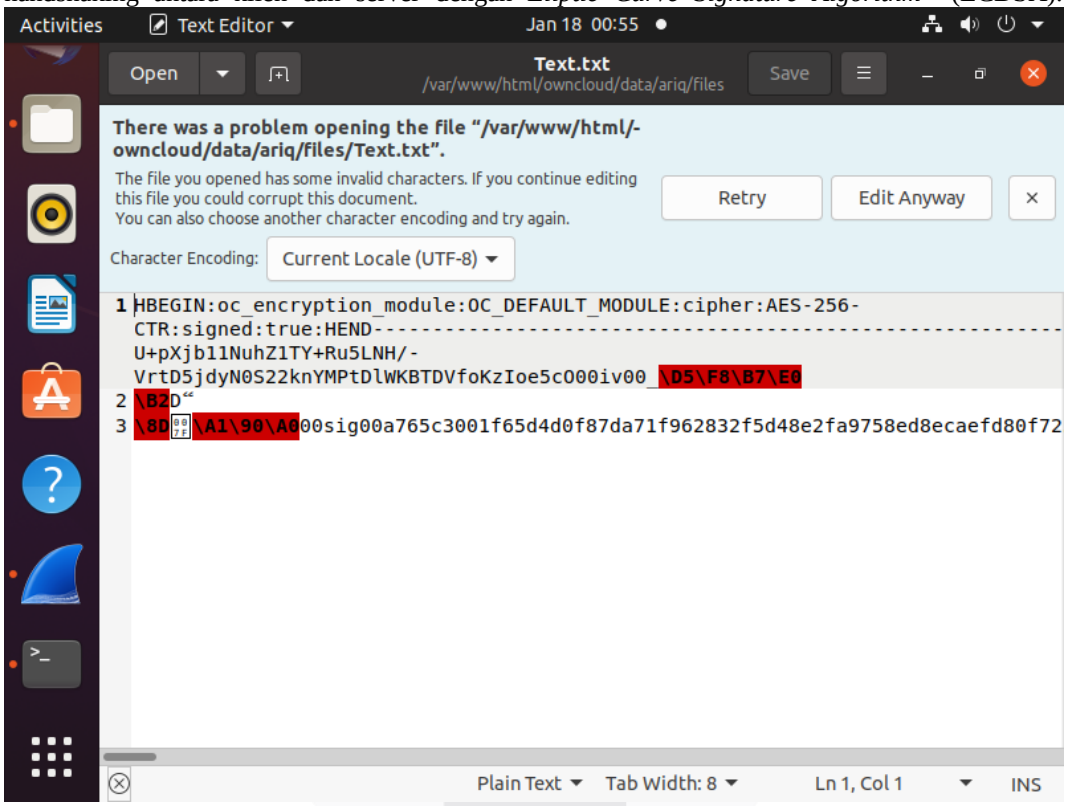


Gambar 4.2 Data tanpa di enkripsi dan SSL.

Ketika jaringan tidak dilindungi oleh *Secure Socket Layer* dan enkripsi dari *cloud storage* jadi data akan terlihat di wireshark dengan berbentuk plaintext, dimana dari nama file dan ukurannya dapat terlihat dan diketahui oleh hacker. Maka dengan itu pengamanan menggunakan SSL dan enkripsi dari cloud storagenya dapat menambahkan keamanan atau privasi dari data tersebut, karena data tersebut hanya user yang tahu.

4.3 Pengujian keamanan Data Yang Tersimpan di *Cloud Storage*

Hasil dari pengujian kemanan data pengguna di *cloud storage* didapat analisa bahwa data yang tersimpan di *cloud storage* tidak bisa diambil dengan mudah begitu saja oleh *hacker*, user lain atau pun admin dari cloud storage, karena data yang tersimpan di *cloud storage* sudah ter-enkripsi *Advanced Encryption Standard* (AES) yang berbasis 256 keys dimana tidak dapat diakses karena file tersebut sudah terenkripsi, dan keamanan jaringannya sudah terenkripsi dan melakukan handshaking antara klien dan server dengan *Eliptic Curve Signature Algorithm* (ECDSA).



Gambar 4.3 File terenkripsi dengan modul AES.

File yang sudah diunggah ke dalam *cloud storage* terenkripsi dengan module AES 256 keys, data tersebut ada di dalam host, akan tetapi data tidak dapat dibaca bila sudah terenkripsi. AES-256, yang memiliki panjang kunci 256 bit, mendukung ukuran bit terbesar dan secara praktis tidak dapat dipecahkan oleh *brute force* berdasarkan daya komputasi saat ini.

5 Kesimpulan dan Saran

5.1 Kesimpulan

Dari penelitian yang dilakukan didapat kesimpulan seperti berikut :

1. Pengujian menggunakan algoritma AES pada keamanan cloud storage membuat data yang telah diunggah ke dalam storage terenkripsi dengan baik dan tidak mudah untuk melakukan *brute force*.
2. Pengujian menggunakan Secure Socket Layer ECDSA pada jaringan data, saat melakukan handshaking *client* dan *host* lalu *host* dan *client*, membuat suatu enkripsi terhadap jaringan data, untuk tidak bisa dilakukan *sniffing* oleh pihak lain.

5.2 Saran

Saran untuk pengembangan keamanan pada sistem kedepannya adalah :

1. Dapat menambah metode algoritma dari cloud storage ini.
2. Dapat diuji dengan tools lain dan metode lain pada cloud storage dan jaringan.

Reference

[1] Wu, Yuzhao ; Lyu, Yongqiang ; Shi, Yuanchun (December 2019). "*Cloud Storage Security Assessment Through Equilibrium Analysis*" (PDF). *Tsinghua Science and Technology* (Volume: 24, Issue 6).

[2] *Smart, Nigel (February 19, 2008).* "Dr Clifford Cocks CB". Bristol University. *Retrieved* August 14, *2011*.

[3] *Rivest, R.; Shamir, A.; Adleman, L. (February 1978).* "A Method for Obtaining Digital Signatures and Public-Key Cryptosystems" *(PDF).* Communications of the ACM.

[4] Diffie, W.; Hellman, M.E. (November 1976). "New directions in cryptography". IEEE Transactions on Information Theory. 22 (6): 644–654.

[5] Jesus Luna, Ahmed Taha, Ruben Trapero, and Neeraj Suri, "Quantitative Reasoning About Cloud Security Using Service Level Agreements". (October 2014) https://ieeexplore.ieee.org/document/7226823

[6] Vijay Varadharajan and Udaya Tupakula, "Security as a Service Model for Cloud Environment" (March, 1 2014) https://ieeexplore.ieee.org/document/6805344/.

[7] Understanding Denial-of-Service Attacks. US-CERT. 6 February 2013. Retrieved 26 May 2016. Prince, Matthew (25 April 2016). "Empty DDoS Threats: Meet the Armada Collective". CloudFlare. Retrieved 18 May 2016.

[8] Cocks, C.C. *(20 November 1973).* "A Note on Non-Secret Encryption" *(PDF). www.gchq.gov.uk. Retrieved* 2017-05-30.

[9] Jim Sauerberg "From Private to Public Key Ciphers in Three Easy Steps".

[10] Brand.com President Mike Zammuto Reveals Blackmail Attempt. 5 March 2014. Archived from the original on 11 March 2014..

[11] Alasdair McAndrew. "Introduction to Cryptography with Open-Source Software". p. 12.

[12] Surender R. Chiluka. "Public key Cryptography"**.**

[13] Victor Chang and Muthu Ramachandran, "Towards achieving Data Security with the Cloud Computing Adoption Framework" (2015) https://ieeexplore.ieee.org/document/7299312

[14] Cryptography and Network Security, W. Stallings.

[15] Distributed Denial of Service Attacks - The Internet Protocol Journal - Volume 7, Number 4. Cisco. Retrieved 2019-08-26. Smith, Steve. "5 Famous Botnets that held the internet hostage". tqaweekly. Retrieved November 20, 2014.

[16] Jack Doerner, Yashvanth Kondi, Eysa Lee, and Abhi Shelat "Secure Two-party Threshold ECDSA from ECDSA Assumptions. Y. Lindell, Fast Secure Two-Party ECDSA Signing", 2017.

[17] Sirikarn Pukkawanna and Youki Kadobayashi , "Classification of SSL Servers based on their SSL Handshake for Automated Security Assessment." 2014

[18] Andy Triwinarko, Elliptic Curve Digital Signature Algorithm (ECDSA), Makalah TA, Departemen Teknik Informatika ITB.