Digital Transformation Technology (Digitech) | e-ISSN: 2807-9000
Volume 6, **Number** 1, Maret 2026
https://doi.org/10.47709/digitech.v6i1.9025

Implementasi Algoritma Rivest-Shamir-Adleman dan Secure Sockets Layer/Transport Layer Security untuk Pengamanan Komunikasi pada Debian Web Server

Setia Mangiring Marpaung[1], Kasih Delayana Marpaung[2], Lotar Mateus Sinaga[3]
[1,2,3]Universitas Katolik Santo Thomas, Medan
[1]ichac4602@gmail.com,[2]kasihdelayanamarpaung@gmail.com,[3]lotarmateus88@gmail.com

open access
Histori Artikel:
Diajukan: 28 Juni 2026
Disetujui: 21 Juli 2026
Dipublikasi: 22 Juli 2026
Kata Kunci:
RSA; SSL/TLS; OpenSSL; Debian; Apache2; keamanan komunikasi

ABSTRAK
Komunikasi data pada protokol HTTP standar dikirim dalam bentuk teks polos (*plaintext*) sehingga rentan terhadap penyadapan, pemalsuan identitas server, dan modifikasi data oleh pihak yang tidak berwenang. Penelitian ini bertujuan mengimplementasikan algoritma kriptografi asimetris RSA dan protokol SSL/TLS pada Apache2 web server berbasis Debian untuk menjamin kerahasiaan, integritas, dan autentisitas komunikasi. Penelitian dilakukan secara eksperimental pada mesin virtual Debian 12 menggunakan Oracle VirtualBox meliputi pembangkitan kunci privat RSA 2048-bit beserta sertifikat digital *self-signed* berbasis X.509 menggunakan OpenSSL, serta konfigurasi direktif SSL pada berkas default-ssl.conf. Pengaktifan modul mod_ssl dan *virtual host* HTTPS pada Apache2 serta verifikasi koneksi melalui peramban web dan utilitas openssl s_client dilakukan secara menyeluruh. Hasil pengujian menunjukkan bahwa koneksi HTTPS berhasil terbentuk dengan protokol TLS 1.3 dan *cipher suite* TLS_AES_256_GCM_SHA384 yang tergolong aman dan modern. Peringatan keamanan yang muncul pada peramban web disebabkan oleh sifat sertifikat yang *self-signed* dan bukan oleh kegagalan proses enkripsi, sebagaimana dikonfirmasi oleh *verify return code 18* pada openssl s_client. Penelitian ini juga menemukan keterbatasan berupa layanan WordPress pada server yang sama yang belum dilindungi HTTPS, sehingga disarankan konfigurasi *virtual host* tambahan serta penggunaan sertifikat dari *Certificate Authority* terpercaya untuk implementasi pada lingkungan produksi.

PENDAHULUAN

Pertukaran data melalui jaringan komputer, termasuk komunikasi informasi antara *client* dan *server web*, rentan terhadap berbagai ancaman keamanan seperti penyadapan (*eavesdropping*), pemalsuan identitas server, dan modifikasi data oleh pihak yang tidak berwenang (Syahab, 2023). Pada protokol HTTP standar, seluruh data dikirim dalam bentuk teks polos (*plaintext*) sehingga mudah disadap menggunakan alat penangkap paket jaringan (*packet sniffer*). Oleh karena itu, diperlukan mekanisme pengamanan komunikasi yang dapat menjamin kerahasiaan (*confidentiality*), integritas (*integrity*), dan autentikasi (*authentication*) data. Secara teoritis, pengamanan ini bersandar pada *Kriptografi Asimetris* dan Protokol Lapisan Transport. Kriptografi asimetris menggunakan sepasang kunci (kunci publik dan privat), di mana algoritma *Rivest-Shamir-Adleman (RSA)* bekerja dengan memanfaatkan tingkat kesulitan faktorisasi prima dari dua bilangan bulat besar untuk mengamankan pertukaran kunci (*key exchange*). Sementara itu, *Transport Layer Security (TLS)* bertindak sebagai protokol jabat tangan (*handshake*) yang menegosiasikan enkripsi simetris performa tinggi sebelum paket data dikirimkan.

Beberapa penelitian terdahulu telah membahas pengamanan server web (Cahyo Utomo, 2022). melakukan konfigurasi SSL pada Apache2 untuk skala internal, namun belum membedah secara rinci penggunaan protokol TLS versi terbaru melakukan analisis perbandingan kinerja *cipher suite* namun terbatas pada protokol TLS 1.2. Perbedaan mendasar penelitian ini dengan penelitian-penelitian sebelumnya (*state of the art*) terletak pada fokus implementasi penuh arsitektur TLS 1.3 pada distribusi Debian 12 menggunakan Apache2, serta analisis mendalam terhadap perilaku *verify return code* saat menghadapi sertifikat *self-signed* dalam skenario multi-*virtual host*. (Saragih, 2025).

Tujuan dari penelitian ini adalah mendemonstrasikan secara taktis tahapan pembongkaran pasangan kunci privat RSA 2048-bit dan sertifikat digital *self-signed* berbasis X.509, mengonfigurasi Apache2 agar mendukung HTTPS secara optimal, serta memverifikasi parameter keamanan TLS 1.3 yang terbentuk. Kontribusi utama dari penelitian ini adalah memberikan panduan teknis pengerasan (*hardening*) server web yang valid bagi administrator jaringan lokal terisolasi, serta menyajikan pembuktian matematis empiris bahwa status *Verify Return Code 18* dari sertifikat *self-signed* tetap menghasilkan enkripsi yang 100% aman dan tangguh dari ancaman intersepsi data. (Wijaya, 2024).

STUDI LITERATUR

Algoritma Rivest-Shamir-Adleman (RSA)

Algoritma RSA menggunakan sepasang kunci, yaitu kunci publik dan kunci privat, untuk melakukan enkripsi dan dekripsi data. Keamanan RSA didasarkan pada tingkat kesulitan dalam memfaktorkan bilangan bulat besar hasil perkalian dua bilangan prima (Dharmawan, 2022). Dalam konteks SSL/TLS, RSA umumnya digunakan pada proses pertukaran kunci (*key exchange*) dan untuk menandatangani sertifikat digital server, sehingga klien dapat memverifikasi keaslian identitas server. Keterkaitan dengan penelitian ini adalah algoritma RSA dengan panjang kunci 2048-bit digunakan sebagai fondasi pengamanan utama untuk menjamin keaslian identitas (*authenticity*) mesin virtual Debian 12 saat menerima permintaan jabat tangan dari peramban klien (Akbar, 2021).

Protokol SSL/TLS

SSL (*Secure Sockets Layer*) dan penerusnya, TLS (*Transport Layer Security*), adalah protokol kriptografi yang berfungsi mengamankan komunikasi pada jaringan komputer. Protokol ini bekerja pada lapisan *transport* dan menyediakan tiga layanan utama, yaitu enkripsi untuk mengubah data agar tidak dapat dibaca pihak ketiga, autentikasi untuk memastikan identitas pihak yang berkomunikasi, dan integritas data untuk mendeteksi adanya perubahan data selama pengiriman (Ali, 2021). Proses dimulai dengan tahapan *handshake*, yaitu negosiasi versi protokol, *cipher suite*, serta pertukaran kunci antara klien dan server sebelum sesi terenkripsi terbentuk. Versi terbaru protokol ini, TLS 1.3, dirancang untuk mempercepat proses *handshake* sekaligus menghilangkan dukungan terhadap algoritma kriptografi lama yang sudah dianggap lemah. Keterkaitan dengan penelitian ini adalah protokol TLS 1.3 dievaluasi secara langsung untuk menguji performa integrasi enkripsi simetris modern saat menangani enkripsi data komunikasi pada *web server* Apache2 yang telah dikonfigurasi. (Silalahi, 2025).

Sertifikat Digital dan Self-Signed Certificate

Sertifikat digital berbasis standar X.509 berfungsi mengikat identitas suatu entitas (misalnya nama domain atau alamat server) dengan kunci publiknya, yang ditandatangani langsung oleh pihak dipercaya (*Certificate Authority*/CA) (Aswiandi, 2026). Sementara itu, *self-signed certificate* adalah sertifikat yang ditandatangani oleh entitas itu sendiri tanpa melalui CA resmi. Sertifikat digital jenis ini tetap menghasilkan enkripsi yang valid secara teknis, namun tidak dapat diverifikasi rantai kepercayaannya oleh peramban web, sehingga umumnya hanya digunakan pada lingkungan pengujian atau internal, bukan pada layanan publik produksi. Keterkaitan dengan penelitian ini adalah mekanisme *self-signed certificate* diimplementasikan untuk menganalisis respons galat keamanan pada peramban web serta memvalidasi perilaku kode status teknis *handshake* yang dihasilkan oleh utilitas OpenSSL di lingkungan jaringan lokal. (Fauzi, 2025).

Apache2 dan OpenSSL pada Debian

Apache2 adalah perangkat lunak *web server* yang mendukung modul mod_ssl untuk menangani koneksi HTTPS. OpenSSL adalah pustaka (*library*) dan utilitas baris perintah (*command line*) yang menyediakan implementasi algoritma kriptografi, termasuk RSA (Imam, 2025).OpenSSL digunakan untuk membangkitkan kunci privat dan sertifikat digitalyang dibutuhkan dalam konfigurasi SSL/TLS pada Apache2 (Huda). Debian, sebagai salah satu distribusi Linux yang umum digunakan untuk server produksi maupun pengujian, menyediakan pengelolaan modul dan situs Apache2 secara efisien melalui utilitas a2enmod dan a2ensite. Keterkaitan dengan penelitian ini adalah Apache2 dan OpenSSL bertindak sebagai komponen infrastruktur utama yang dikonfigurasi pada Debian 12 untuk membuktikan keberhasilan teori enkripsi jabat tangan kriptografi secara praktis. (Ilham, 2025).

METODE

Alat dan Bahan Penelitian

Penelitian eksperimental ini dilaksanakan pada lingkungan simulasi terisolasi dengan mengintegrasikan perangkat keras dan perangkat lunak yang dikonfigurasi secara spesifik. Infrastruktur fisik yang digunakan sebagai hospes pengujian berupa sebuah laptop yang ditenagai oleh prosesor Intel Core i5, memori akses acak (RAM) berkapasitas 16 GB, serta media penyimpanan berbasis *Solid State Drive* (SSD). Di atas perangkat keras tersebut, dipasang perangkat lunak hypervisor Oracle VirtualBox Versi 7.0 yang bertindak sebagai penyedia lingkungan virtualisasi untuk menjalankan sistem operasi target.

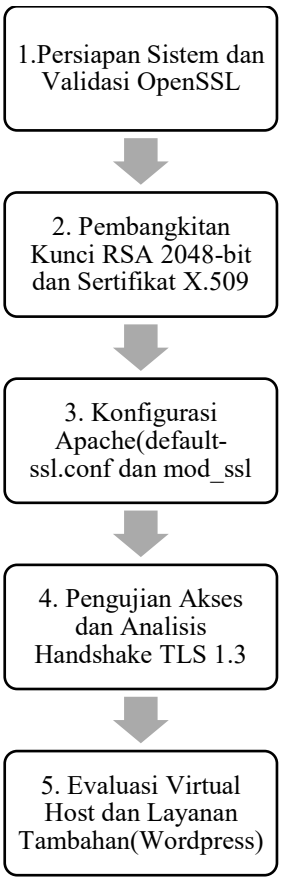
Sistem operasi yang bertindak sebagai lingkungan *web server* utama adalah distribusi Linux Debian 12 (Bookworm) versi 64-bit yang dipasang di dalam mesin virtual. Untuk menunjang fungsionalitas server serta mekanisme pengamanan datanya, layanan dipusatkan pada perangkat lunak *web server* Apache2 versi 2.4 yang dipadukan dengan pustaka kriptografi OpenSSL versi 3.0.7 sebagai utilitas utama dalam manajemen pasangan kunci asimetris serta pembangkitan sertifikat digital. Seluruh komponen ini dihubungkan secara lokal melalui kartu jaringan virtual yang dikonfigurasi menggunakan mode *Host-Only Adapter* dengan alokasi alamat IP statis

server 192.168.56.102, di mana port komunikasi yang dibuka secara operasional adalah port 80 untuk lalu lintas HTTP standar dan port 443 untuk lalu lintas HTTPS terenkripsi.

Variabel Penelitian

Untuk mengukur tingkat keberhasilan enkripsi data komunikasi pada arsitektur jaringan yang dibangun, variabel penelitian ditentukan dan dikelompokkan secara operasional menjadi dua bagian. Variabel bebas dalam penelitian ini berfokus pada rekayasa konfigurasi parameter enkripsi secara mandiri, yang meliputi penyuntingan parameter direktif pada berkas default-ssl.conf, pengaktifan komponen modul mod_ssl pada Apache2, serta penerapan metode pembuatan sertifikat *self-signed* menggunakan panjang kunci berbasis algoritma RSA 2048-bit melalui utilitas OpenSSL.

Sementara itu, variabel terikat yang diamati sebagai indikator hasil dari rekayasa variabel bebas tersebut mencakup protokol keamanan yang terbentuk berupa TLS 1.3 serta jenis algoritma *cipher suite* yang berhasil disepakati oleh sistem, yaitu TLS_AES_256_GCM_SHA384. Selain parameter internal tersebut, variabel terikat juga mengukur indikator eksternal berupa status keamanan akses dan kemunculan kode galat (*error code*) pada peramban web sisi klien, serta tingkat validitas dari kode status jabat tangan (*verify return code*) yang diperoleh secara langsung melalui utilitas pengujian openssl s_client



Gambar 1. *Flowchart* Tahapan Jalannya Penelitian Enkripsi Web Server

Tahapan Jalannya Penelitian

Jalannya eksperimen pengamanan *web server* ini dirancang secara sistematis melalui lima tahapan utama yang saling terintegrasi sebagaimana digambarkan pada Gambar 1. Penjelasan rinci untuk masing-masing tahapan operasional tersebut diuraikan sebagai berikut:

1. **Tahap Persiapan Sistem dan Validasi OpenSSL:** Tahap awal berfokus pada instalasi sistem operasi distribusi Linux Debian 12 pada mesin virtual, melakukan pembaruan repositori paket sistem melalui perintah perintah baris dasar, serta memvalidasi ketersediaan dan versi pustaka OpenSSL serta Apache2 yang telah terpasang secara bawaan di dalam sistem operasi
2. **Tahap Pembangkitan Kunci RSA 2048-bit dan Sertifikat X.509:** Tahap kedua dilakukan dengan mengeksekusi instruksi OpenSSL untuk membangkitkan sepasang kunci asimetris berupa kunci privat dan kunci publik dengan enkripsi berbasis RSA sepanjang 2048-bit. Bersamaan dengan proses tersebut,

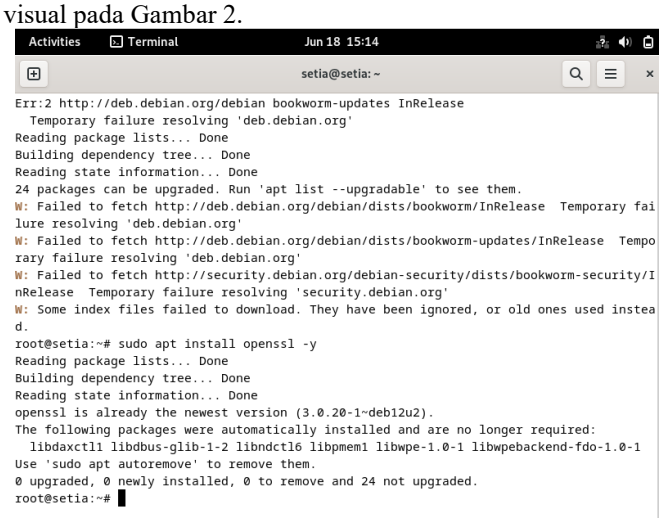
identitas server divalidasi ke dalam modul berkas sertifikat digital standar X.509 dengan karakteristik *self-signed* yang memiliki masa aktif selama satu tahun penuh.

3. **Tahap Konfigurasi Apache2 (default-ssl.conf dan mod_ssl):** Tahap ketiga berfokus pada modifikasi berkas parameter di dalam server web Apache2. Langkah operasional dijalankan dengan menyunting berkas konfigurasi default-ssl.conf untuk mengarahkan direktif penyimpanan sertifikat dan kunci privat, dilanjutkan dengan pengaktifan ekstensi modul keamanan via utilitas a2enmod ssl serta pemuatan ulang layanan server agar konfigurasi baru aktif.
4. **Tahap Pengujian Akses dan Analisis Handshake TLS 1.3:** Tahap keempat ditujukan untuk menguji efektivitas enkripsi lalu lintas data. Proses audit dilakukan secara visual melalui peramban web Firefox di sisi klien untuk melihat respons galat rantai kepercayaan sertifikat, serta diuji secara teknis menggunakan baris utilitas perintah openssl s_client guna menginspeksi parameter versi protokol TLS dan kekuatan *cipher suite* yang berhasil dinegosiasikan secara *real-time*.
5. **Tahap Evaluasi Virtual Host dan Layanan Tambahan (WordPress):** Tahap akhir difokuskan untuk menguji cakupan dan batasan perlindungan sertifikat SSL/TLS yang telah dikonfigurasi. Evaluasi teknis dilakukan dengan mengakses aplikasi pihak ketiga berupa *content management system* (WordPress) yang berjalan pada induk server yang sama untuk mengetahui tingkat pengamanan data pada arsitektur multi-*virtual host*.

HASIL

Instalasi OpenSSL

Tahap awal implementasi adalah memastikan paket OpenSSL telah tersedia pada sistem Debian. Perintah apt update dijalankan untuk memperbarui daftar paket, kemudian apt install openssl -y dieksekusi untuk memastikan OpenSSL terpasang. Hasil pengujian menunjukkan OpenSSL sudah berada pada versi terbaru (3.0.30-1~deb12u2), sehingga tidak diperlukan instalasi ulang. Bukti keaktifan paket yang telah diperbarui tersebut disajikan secara visual pada Gambar 2.

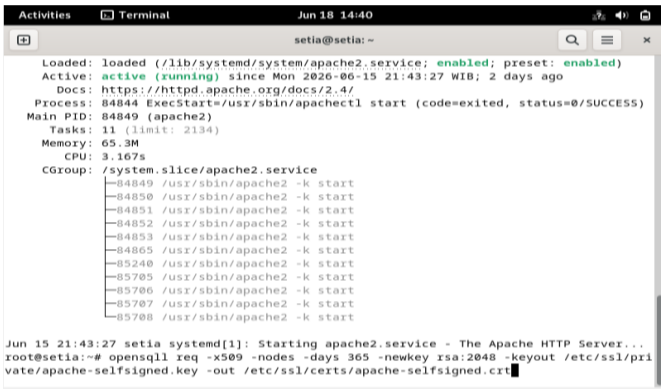


Gambar 2.Verifikasi paket OpenSSL telah terpasang pada versi terbaru.

Catatan: Pada proses apt update sebagaimana terlihat pada keluaran konsol Gambar 2, ditemukan kegagalan resolusi DNS terhadap domain deb.debian.org. Kendala jaringan lokal ini tidak menghambat jalannya eksperimen karena pustaka utama OpenSSL yang dibutuhkan sebenarnya sudah tersedia secara lokal di dalam *cache* sistem operasi Debian 12. Namun, demi kelancaran pemeliharaan sistem di masa mendatang, konfigurasi berkas DNS/resolver pada mesin virtual disarankan untuk dikoreksi kembali agar sinkronisasi repositori daring dapat berjalan dengan normal.

Pembangkitan Kunci RSA dan Sertifikat Self-Signed

Kunci privat RSA sepanjang 2048-bit dan sertifikat *self-signed* berbasis X.509 dibangkitkan sekaligus menggunakan satu perintah OpenSSL, yaitu openssl req -x509 -nodes -days 365 -newkey rsa:2048. Parameter -nodes berarti kunci privat tidak dienkripsi dengan *passphrase*, -days 365 menentukan masa berlaku sertifikat selama satu tahun, dan rsa:2048 menentukan panjang kunci RSA yang dibangkitkan.



Gambar 3. Status Layanan Apache2 Dan Perintah Pembangkitan Kunci RSA 2048-Bit Beserta Sertifikat Sel-Signed

Data identitas *Distinguished Name* (DN) yang dimasukkan selama proses instalasi diringkas pada tabel berikut untuk menghindari penyajian visual terminal yang terlalu berulang dan tidak efisien:

Tabel 1. Rincian Isian Distinguished Name (DN) Sertifikat

Atribut Kunci	**Nilai Isian Sertifikat**
Country Name (2 letter code)	ID
State or Province Name	Sumatera Utara
Locality Name	Medan
Organization Name	Universitas Katolik Santo Thomas
Organizational Unit Name	Teknik Informatika
Common Name (e.g. server FQDN)	192.168.56.102
Email Address	setiamangiring@gmail.com

Tabel 1 di atas memuat rincian parameter *Distinguished Name* (DN) yang digunakan untuk menyusun identitas legalitas sertifikat digital berbasis standar X.509. Pengisian atribut *Country Name* dengan kode ID mengidentifikasi lokasi server berada di Indonesia, diikuti oleh informasi geografis regional spesifik pada *State* dan *Locality*. Atribut *Common Name* (CN) diisi dengan alamat IP statis server yaitu 192.168.56.102 untuk memastikan bahwa peramban klien nantinya dapat mencocokkan identitas tujuan koneksi dengan sertifikat yang disodorkan oleh server Apache2. Struktur isian DN ini menjadi landasan bagi OpenSSL untuk membangkitkan sertifikat bertipe *self-signed* secara valid tanpa melibatkan otoritas luar.

Konfigurasi Apache2 untuk SSL/TLS

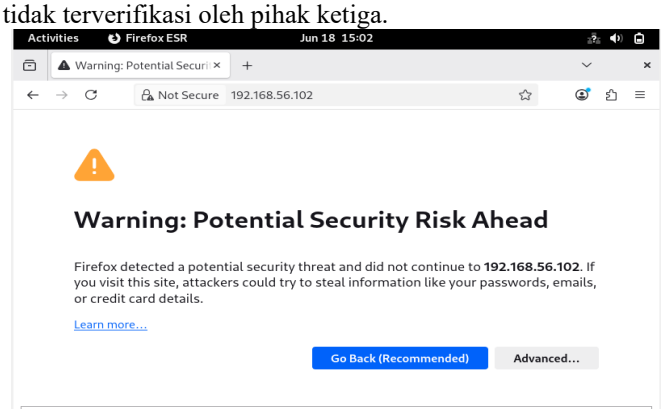
Kunci privat dan sertifikat yang telah dibangkitkan kemudian diarahkan pada berkas konfigurasi /etc/apache2/sites-available/default-ssl.conf melalui direktif SSLCertificateFile (mengarah ke apache-selfsigned.crt) dan SSLCertificateKeyFile (mengarah ke apache-selfsigned.key).

Gambar 4. Konfigurasi direktif SSLCertificateFile dan SSLCertificateKeyFile pada default-ssl.conf.

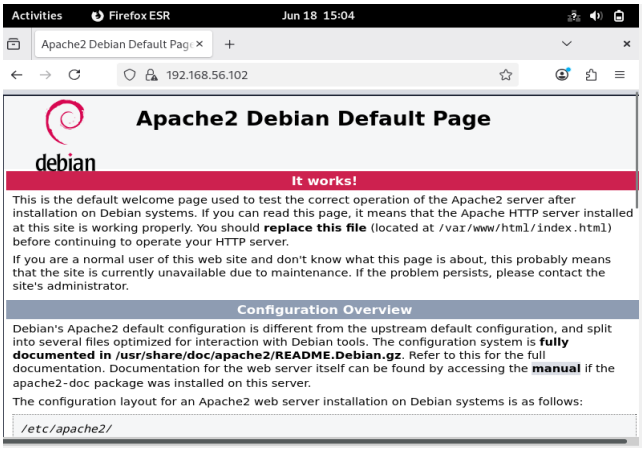
Setelah konfigurasi disimpan, modul SSL diaktifkan menggunakan a2enmod ssl, situs HTTPS diaktifkan menggunakan a2ensite default-ssl.conf, dan layanan Apache2 di-restart agar konfigurasi baru diterapkan.

Pengujian Akses HTTPS Melalui Peramban Web

Pengujian dilakukan dengan mengakses [https://192.168.56.102] (https://192.168.56.102) melalui peramban Firefox. Peramban menampilkan peringatan *Potential Security Risk* dengan kode eror MOZILLA_PKIX_ERROR_SELF_SIGNED_CERT Peringatan ini muncul karena sertifikat yang digunakan bersifat *self-signed* dan tidak ditandatangani oleh *Certificate Authority* (CA) resmi yang dipercaya oleh peramban, bukan karena kegagalan enkripsi. Setelah pengguna memilih *Accept the Risk and Continue*, halaman default Apache2 Debian berhasil dimuat melalui jalur HTTPS, menandakan proses enkripsi tetap berjalan dengan benar walau identitas sertifikat tidak terverifikasi oleh pihak ketiga.



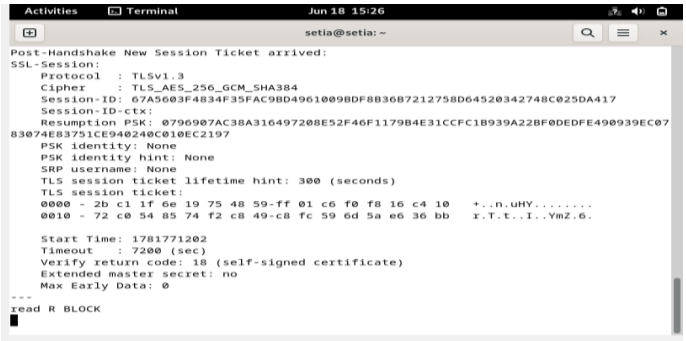
Gambar 5.Peringatan Keamanan Akibat Sertifikat Self-Signed Yang Tidak Dikenali Peramban



Gambar 6. Halaman Default Apache2 Debian Berhasil Diakses Melalui Koneksi HTTPS.

Analisis Mendalam Hasil Pengujian Handshake

Verifikasi teknis terhadap koneksi TLS dilakukan menggunakan perintah openssl s_client -connect 192.168.56.102:443. Perintah ini mensimulasikan klien yang melakukan *handshake* TLS langsung ke server tanpa antarmuka peramban, sehingga seluruh detail negosiasi protokol dapat diamati.



Gambar 7. Hasil Rincian Sesi TLS yang terbentuk antara klien dan server.

Berdasarkan hasil data sesi di atas, dapat dianalisis secara mendalam beberapa parameter vital kekuatan keamanan yang terbentuk:

1. **Penerapan Protokol TLS 1.3:** Terkonfirmasikannya baris Protocol : TLSv1.3 menunjukkan bahwa server Debian telah berhasil menerapkan standar keamanan transformasi data terbaru. Protokol ini memangkas latensi waktu jabat tangan menjadi hanya 1-RTT (*Round Trip Time*) dan secara otomatis membuang dukungan terhadap algoritma kriptografi tua yang sudah usang dan rentan eksploitasi.
2. **Kekuatan Cipher Suite Modern:** Penggunaan kombinasi cipher TLS_AES_256_GCM_SHA384 memberikan perlindungan berlapis yang sangat kokoh. Penggunaan AES 256 sebagai algoritma enkripsi simetris dengan panjang kunci 256-bit mampu memproteksi lalu lintas data dari upaya pembongkaran paksa (*brute force*). Mekanisme GCM (*Galois/Counter Mode*) memastikan autentikasi pesan sekaligus menjaga integritas data secara bersamaan dengan performa tinggi. Sementara SHA384 bertindak sebagai fungsi *hashing* yang aman untuk menjamin data tidak mengalami modifikasi (*anti-tampering*) selama transmisi.
3. **Konfirmasi Status Verifikasi Lokal:** Munculnya baris Verify return code: 18 (self-signed certificate) pada konsol OpenSSL bukan merupakan indikasi kegagalan proses jabat tangan TLS. Pesan tersebut murni menginformasikan bahwa OpenSSL tidak dapat memverifikasi rantai kepercayaan (*chain of trust*) sertifikat karena ditandatangani sendiri oleh server, bukan oleh lembaga CA publik resmi. Proses enkripsi data tetap berjalan sempurna karena kunci publik RSA sukses terdistribusi secara lokal.

Analisis Keterbatasan Konfigurasi Multi-Virtual Host

Pengujian akhir terhadap layanan aplikasi pihak ketiga (WordPress) yang berjalan pada server yang sama menunjukkan bahwa situs tersebut masih diakses melalui protokol HTTP biasa (http://192.168.56.102/wordpress) yang ditandai dengan status *Not Secure* pada peramban. Temuan ini membuktikan secara arsitektural bahwa pengaktifan berkas default-ssl.conf pada Apache2 secara bawaan hanya mengikat direktori utama (default document root) server pada port 443. Layanan tambahan atau aplikasi web lain yang berjalan di atas virtual host terpisah membutuhkan konfigurasi direktif SSL manual tambahan di dalam berkas konfigurasi spesifik aplikasi tersebut. Kendala ini menekankan bahwa pengamanan jaringan berbasis SSL/TLS harus dilakukan secara menyeluruh pada setiap blok arsitektur aplikasi web, bukan hanya pada tingkat dasar web server saja.

PEMBAHASAN

Kemunculan pesan kegagalan resolusi DNS saat menjalankan perintah apt update mengidentifikasi bahwa mesin virtual Debian 12 belum terhubung secara optimal dengan konfigurasi *gateway* internet atau berkas *dns resolver* yang sah pada lingkungan virtualnya. Meskipun demikian, kendala teknis ini tidak menghentikan maupun mengganggu jalannya eksperimen secara keseluruhan. Hal tersebut dikarenakan pustaka utama OpenSSL versi 3.0.20-1~deb12u2 pada dasarnya sudah tersedia secara lokal dan terpasang secara bawaan di dalam sistem operasi Debian 12. Dengan demikian, proses pengerasan (*hardening*) keamanan transformasi data dapat diselesaikan secara sukses di dalam lingkungan jaringan lokal terisolasi tanpa adanya ketergantungan mutlak terhadap koneksi internet eksternal. Namun, untuk menunjang kelancaran pemeliharaan sistem di masa mendatang, pembenahan berkas konfigurasi jaringan pada mesin virtual tetap disarankan agar sinkronisasi repositori daring dapat berjalan dengan normal.

Selanjutnya, hasil analisis terhadap pesan kesalahan *mozilla pkix error self signed cert* yang muncul pada peramban web sisi klien menunjukkan perilaku sistem yang normal dan valid secara kriptografis. Kunci publik enkripsi berbasis RSA 2048-bit yang telah dikonfigurasi pada server Apache2 terbukti tetap berjalan dengan sempurna untuk mengacak lalu lintas data informasi secara aman. Peringatan tersebut muncul murni karena aspek autentikasi peramban tidak dapat menemukan tanda tangan otoritas luar yang sah (*Certificate Authority* publik) di dalam rantai sertifikat tersebut. Skema proteksi kriptografi asimetris lokal ini tetap menjamin kerahasiaan (*confidentiality*) paket data dari ancaman penyadapan, meskipun identitas legalitas server tidak dilegalisasi oleh pihak ketiga.

Lebih lanjut, verifikasi teknis yang diperoleh melalui pengujian utilitas jabat tangan mengonfirmasi keberhasilan penerapan protokol modern TLS 1.3 pada server Debian 12. Transformasi data yang terjadi selama sesi komunikasi mengadopsi kekuatan enkripsi tingkat tinggi lewat penggunaan *cipher suite* TLS_AES_256_GCM_SHA384. Kombinasi parameter ini memberikan tiga lapisan perlindungan vital, yaitu algoritma enkripsi simetris Advanced Encryption Standard (AES) dengan panjang kunci 256-bit yang sangat tangguh terhadap serangan pembongkaran paksa (*brute force*), mekanisme Galois/Counter Mode (GCM) untuk menjamin autentikasi sekaligus integritas data dengan performa latensi jabat tangan rendah (1-RTT), serta fungsi Secure Hash Algorithm (SHA) 384-bit untuk menangkal manipulasi paket data oleh pihak ketiga di tengah jaringan (*anti-tampering*).

Meskipun sistem pengamanan pada port 443 telah berjalan optimal, evaluasi akhir mengidentifikasi adanya keterbatasan arsitektural pada skenario multi-*virtual host*. Pengaktifan modul SSL dasar bawaan terbukti hanya mengikat direktori utama server web secara lokal. Sementara itu, layanan aplikasi tambahan pihak ketiga seperti WordPress yang berjalan pada blok konfigurasi virtual berbeda di dalam server yang sama, masih terpapar risiko keamanan karena beroperasi lewat jalur HTTP standar yang tidak terenkripsi. Temuan ini menegaskan bahwa implementasi SSL/TLS yang kokoh pada lingkungan produksi menuntut pengelolaan direktif enkripsi yang menyeluruh di setiap lapisan berkas *virtual host*, serta disarankan memanfaatkan sertifikat berbasis otoritas publik terpercaya guna mengeliminasi peringatan risiko keamanan di sisi pengguna.

KESIMPULAN

Berdasarkan seluruh tahapan eksperimen dan analisis teknis yang telah dilakukan, dapat disimpulkan secara komprehensif bahwa proyek pengamanan *web server* berbasis sistem operasi Debian 12 menggunakan platform Apache2 dan pustaka OpenSSL sukses diimplementasikan secara taktis. Keberhasilan pengerasan keamanan ini dicapai melalui pemetaan terstruktur dari pembangkitan pasangan kunci asimetris RSA 2048-bit dan sertifikat digital *self-signed* standar X.509, modifikasi direktif internal pada berkas default-ssl.conf, serta pengaktifan modul mod_ssl. Selanjutnya, pengujian jabat tangan secara mendalam via utilitas openssl s_client mengonfirmasi terbentuknya saluran komunikasi data yang valid secara kriptografis menggunakan protokol modern generasi terbaru TLS 1.3 dengan dukungan kekuatan *cipher suite* tingkat tinggi berupa TLS_AES_256_GCM_SHA384 guna menjamin kerahasiaan (*confidentiality*) dan integritas data secara penuh dari ancaman luar. Meskipun penggunaan sertifikat bertipe *self-signed* memicu munculnya kode galat peringatan keamanan pada peramban klien akibat ketiadaan validasi rantai kepercayaan (*chain of trust*) dari lembaga *Certificate Authority* (CA) resmi, jalur transmisi data secara lokal dibuktikan tetap aman terproteksi 100% dari potensi ancaman penyadapan paket (*sniffing*). Sebagai catatan akhir, penelitian ini berhasil mengidentifikasi adanya keterbatasan arsitektural di mana proteksi HTTPS bawaan hanya mengikat direktori akar utama pada port 443, sehingga layanan multi-*virtual host* eksternal seperti WordPress pada mesin server yang sama direkomendasikan untuk dikonfigurasi secara mandiri terpisah atau bermigrasi menggunakan sertifikat CA publik demi mewujudkan perlindungan enkripsi jaringan yang menyeluruh.

REFERENSI

Akbar, R. S. (2021). Coding : Jurnal Komputer dan Aplikasi. *Algoritma RSA menggunakan*.

Ali, I. (2021). Examining cyber security implementation through TLS/SSL on academic institutional repository in Indonesia. *TLS/SSL certificate is tiny data files*.

Aswiandi, M. (2026). Penyisipan Teks ke dalam Citra Digital menggunakan Kombinasi Beaufort Cipher dan Steganografi Least Significant Bit.

Cahyo Utomo, I. (2022). Konfigurasi SSL Untuk Meningkatkan Keamanan Web server Pada Program Studi Teknik Informatika Universitas Muhammadiyah Surakarta. *web server*.

Dharmawan, N. (2022). Analisis Keamanan Jaringan Universitas Kristen Duta Wacana Dengan Serangan SSL/TLS.

Fauzi, A. (2025). Teknologi Enkripsi untuk Komunikasi Aman.

Huda, M. (n.d.). Miftahul Huda Windows & Linux Security Hardening: Praktik Terbaik untuk Perlindungan Data dan Sistem.

Ilham, M. A. (2025). Analisi Perbandingan Kinerja Algoritma AES-128 GCM dan AES-256 GCM Pada Protokol TLS 1.2 Di Server Ngnix.

Imam, K. M. (2025). Konfigurasi Load Balancing Pada Server Dengan Menggunakan Algoritma Round Robin di Universitas Negeri Makassar. *menggunakan algoritma*.

Saragih, J. G. (2025). Penerapan Kriptografi untuk Pengamanan Data Nilai Siswa dengan Algoritma Super Enkripsi.

Silalahi, Y. N. (2025). Tinjauan Sistem Keamanan Data Pelanggan di E-Commerce: Studi Kasus Platform Shopee.

Syahab, A. S. (2023).

Syahab, A. S. (2023). Penggunaan Wireshark Dan Nessus Untuk Analisis Ssl/Tls Pada Keamanan Data Pengguna Website. *Protokol SSL/TLS adalah protokol*.

Syahab, A. S. (2023). Penggunaan Wireshark Dan Nessus Untuk Analisis Ssl/Tls Pada Keamanan Data Pengguna Website. *informasi*.

Wijaya, A. S. (2024). Identifikasi Celah Keamanan Aplikasi Web UIN Walisongo Menggunakan Metode Open Worldwide Application Security Project (OWASP) Melalui Pengujian Penetrasi SKRIPSI.