BAB II
KAJIAN LITERATUR

2.1 Keamanan Data

Perkembangan zaman membuat teknologi informasi dan komunikasi semakin maju. Proses bertukar pesan atau informasi menjadi semakin mudah dilakukan. Dalam proses bertukar pesan sangat penting menjaga keamanan pesan atau informasi agar pesan tersebut tidak dapat dimengerti oleh pihak lain maupun pihak yang tidak berwenang. Dengan pesatnya perkembangan teknologi di dunia internet, masyarakat dapat mengumpulkan informasi, bertukar informasi dan berita dengan mudah, cepat dan bebas. Bebasnya akses informasi di internet juga dapat menimbulkan dampak negatif khususnya *cybercrime* seperti akses tidak sah terhadap data berita, penyalahgunaan informasi untuk keuntungan pribadi yang merugikan pengguna internet. Itulah mengapa penting untuk melindungi dan mengamankan pesan. Keberadaan pengamanan pesan bertujuan untuk melindungi pesan dari berbagai kejahatan dunia maya [10].

Dalam era digital saat ini, keamanan informasi menjadi aspek yang sangat penting dalam mengamankan komunikasi dan pertukaran data melalui jaringan [11]. Masalah keamanan merupakan salah satu aspek terpenting dari sebuah sistem informasi. Sementara itu masalah keamanan untuk saat ini sering tidak dipedulikan, padahal hal tersebut merupakan hal yang sangat penting dalam pengamanan data/informasi yang bersifat rahasia untuk mengurangi dampak kerugian materil dan nonmateril Keamanan data merupakan hal yang sangat penting dalam menjaga kerahasiaan informasi terutama yang berisi informasi sensitif yang hanya boleh diketahui isinya oleh pihak yang berhak saja, apalagi jika pengirimannya dilakukan melalui jaringan publik, apabila data tersebut tidak diamankan terlebih dahulu, akan sangat mudah disadap dan diketahui isi informasinya oleh pihak-pihak yang tidak bertanggung jawab [10].

Keamanan dapat didefinisikan sebagai proses, prosedur, dan kebijakan yang dipakai untuk menutup akses yang tidak sah. Dalam hal ini perlu diperhatikan bahwa keamanan sistem informasi harus dijaga kerahasiaannya agar informasi siap diterima oleh penerima dan dapat dikirim dengan cara yang dirahasiakan oleh orang lain yang tidak berwenang. Ini hanya boleh dibaca oleh penerima dan dapat dirahasiakan oleh pengirim [12].

2.2 Kriptografi

Kriptografi adalah studi tentang metode komunikasi yang aman antara dua belah pihak. Biasanya ada dua pihak yang saling mengirim pesan, tetapi mereka ingin menghindari kemungkinan pihak ketiga memahami isi dari pesan mereka jatuh kepada pihak yang salah. Kriptografi adalah sebuah seni yang berfokus pada menyembunyikan dan mengirimkan pesan secara diam-diam. Banyak sandi yang digunakan sepanjang sejarah, banyak diantaranya sekarang dianggap tidak aman menurut standar modern. Nyatanya, baru pada pertengahan abad ke-20 kriptografi berubah dari seni menjadi sebuah sains [13].

Kriptografi merupakan ilmu dan seni untuk mengamankan informasi sehingga hanya pihak yang berwenang yang dapat mengakses dan memahami informasi tersebut. Dalam konteks keamanan informasi, kriptografi berperan penting dalam melindungi data dari akses yang tidak sah, pengubahan, dan penghapusan. Kriptografi telah menjadi teknik utama dalam mengatasi ancaman siber [14].

Kriptografi telah digunakan selama ribuan tahun untuk membantu menyediakan rahasia komunikasi antara pihak-pihak yang saling percaya. Dalam bentuknya yang paling dasar, dua orang, sering dilambangkan sebagai Alice dan Bob, telah menyepakati kunci rahasia tertentu. Di lain waktu, Alice mungkin ingin mengirim pesan rahasia ke Bob (atau Bob mungkin ingin mengirim pesan ke Alice). Kunci digunakan untuk mengubah pesan asli (yang biasanya kita sebut dengan plaintext) menjadi bentuk acak yang tidak dapat dipahami kepada siapa saja yang tidak memiliki kunci. Proses ini disebut enkripsi, dan pesan yang diacak disebut ciphertext. Ketika Bob menerima ciphertext, dia dapat menggunakan kunci untuk mengubah ciphertext kembali menjadi plaintext atau teks asli, ini disebut dengan proses dekripsi [10].

Kata kriptografi (*cryptography*) berasal dari bahasa Yunani yaitu Kryptos (tersembunyi atau rahasia) dan Graphein (menulis atau tulisan). Sehingga dapat dijabarkan bahwa secara harfiah makna kriptografi adalah menulis secara tersembunyi untuk menyampaikan pesan-pesan yang perlu dijaga kerahasiaannya. Kriptografi memiliki arti lain, yaitu suatu ilmu yang mempelajari tentang teknik-teknik matematika yang berkaitan dengan aspek keamanan informasi data atau keamanan pesan dengan menggunakan dua proses dasar kriptografi yaitu enkripsi dan dekripsi. Enkripsi dapat diartikan sebagai cipher atau kode, merupakan proses penyembunyian sebuah data pesan, dengan cara mengubah plaintext (pesan yang bisa dibaca) menjadi ciphertext (pesan acak yang tidak bisa dibaca). Sedangkan dekripsi merupakan kebalikan dari enkripsi yaitu mengubah kembali bentuk tersamar tersebut menjadi informasi awal [15].

Pesan asli, disebut *plaintext*, disandikan menjadi pesan terenkripsi yang disebut dengan *ciphertext* melalui proses enkripsi, dan *ciphertext* diubah kembali menjadi *plaintext* melalui proses dekripsi. Kriptografi memiliki beberapa algoritma yang banyak digunakan untuk mengamankan informasi. Salah satu algoritma kriptografi yang umum digunakan dalam keamanan adalah algoritma XOR. Algoritma XOR adalah algoritma yang sering digunakan dalam sandi yang menggunakan operasi bit demi bit dan termasuk dalam kriptografi klasik. Algoritma XOR juga merupakan algoritma sederhana yang menggunakan prinsip logika XOR. Untuk proses dimana proses enkripsi dilakukan dengan kunci XOR pada *plaintext* unntuk mendapatkan cipherteks. Pada proses dekripsi, *ciphertext* dikodekan dengan XOR dengan kunci untuk mendapatkan teks aslinya (*plaintext*). Proses enkripsi dan dekripsi tidak sulit dan mudah untuk diimplementasikan [10].

Menurut catatan sejarah kuno, aplikasi kriptografi pertama (yang telah ditemukan) adalah hieroglyphics yang diterapkan oleh orang Mesir kuno pada awal 3000 tahun SM. Mulai ada di masa kejayaan Yunani oleh spartan di Yunani sekitar 400 SM. Pada saat itu alat yang digunakan untuk membuat pesan tersembunyi disebut Scytale. Scytale memiliki bentuk batang silinder dengan kombinasi 18 huruf dan pita panjang yang terbuat dari bahan papirus, cara membaca pesannya adalah dengan menggulung pita pada batang silinder. Orang Cina dan Jepang mulai mengenali kriptografi pada abad ke-15 M. Di bawah pemerintahan Julius Caesar (zaman Romawi), teknik caesar cipher diciptakan untuk mengirim pesan rahasia kepada anggota militer yang berada di tengah-tengah perang. Penggunaan kriptografi juga semakin intens karena pertimbangan stabilitas negara. Meskipun teknik yang digunakan tidak serumit orang Yunani, untuk memahami pesan kriptografi dari periode Romawi cukup sulit untuk dikerjakan [15].

Kriptografi modern dipicu oleh perkembangan peralatan komputer digital. Dengan komputer digital, cipher yang lebih kompleks menjadi sangat mungkin untuk dihasilkan. Tidak seperti kriptografi klasik yang mengenkripsi karakter per karakter (dengan menggunakan alfabet tradisional), kriptografi modern beroperasi pada string biner. Cipher yang kompleks seperti DES (Data Encryption Standard) dan penemuan algoritma RSA adalah algoritma kriptografi modern yang paling dikenal di dalam sejarah kriptografi modern. Di Indonesia kriptografi dikenal atau bisa juga disebut dengan kriptologi ataupun sandisastra. Salah satu tujuan dari kriptografi adalah melakukan berbagai upaya komunikasi antar individu ataupun kelompok secara aman tanpa kehadiran pihak-pihak yang tidak diinginkan, serta menganalisis komunikasi yang sulit dipahami [15].

Kriptografi berfungsi mengamankan data, baik yang ditransfer melalui jaringan komputer maupun tidak, dimana hal tersebut sangat berguna untuk melindungi privasi, integritas data, *authentication*, dan *non-repudiation* [15].

1. *Confidentiality* (Kerahasiaan)

   Informasi yang dilindungi tidak akan bisa diakses oleh siapa pun yang tak memiliki wewenang.

2. *Integrity Data* (Integritas Data)

   Data yang akan diterima dan dikirim tidak dapat diubah tanpa sepengetahuan kedua belah pihak.

3. *Authentication* (Autentikasi)

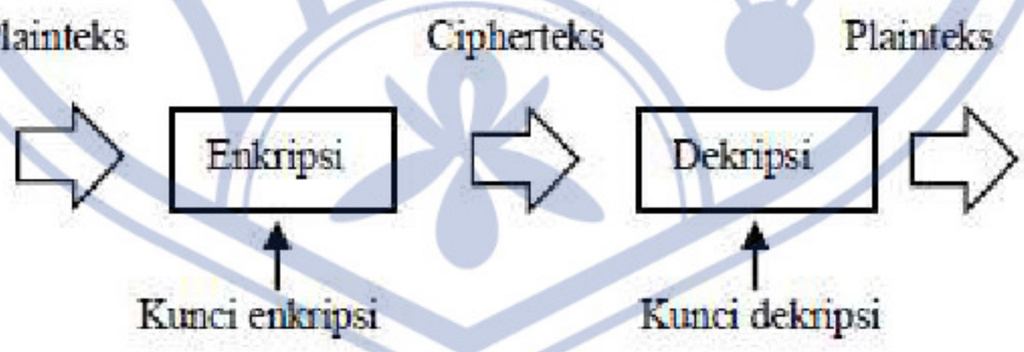
   Pihak penerima dan pengirim dapat mengetahui identitas masing-masing serta sumber data yang sedang mereka gunakan.

4. *Non-Repudiation*

   Pihak penerima atau pengirim tidak akan bisa menyangkal tujuannya menciptakan atau mengubah suatu data.

Kriptografi modern merupakan kriptografi yang cukup rumit. Dibutuhkan pengetahuan matematika untuk menguasainya. Oleh karena itu kriptografi modern berkembang bersamaan dengan berkembangnya komputer hingga jaman sekarang. Kriptografi modern terdiri dari 3 bagian, yaitu kriptografi simetris, kriptografi asimetris dan kriptografi hibrid [15].

Skema enkripsi dan dekripsi dalam kriptografi dapat digambarkan seperti terlihat pada Gambar 2.1.



Gambar 2.1 Skema Enkripsi dan Dekripsi *[16]*

Jika P adalah plaintext dan C adalah ciphertext, maka fungsi enkripsi E yang memetakan P ke cipher C adalah [16]:

$$E(P)=C \tag{1}$$

Fungsi dekripsi D yang memetakan cipher C ke plaintext P adalah

$$D(C)=P \tag{2}$$

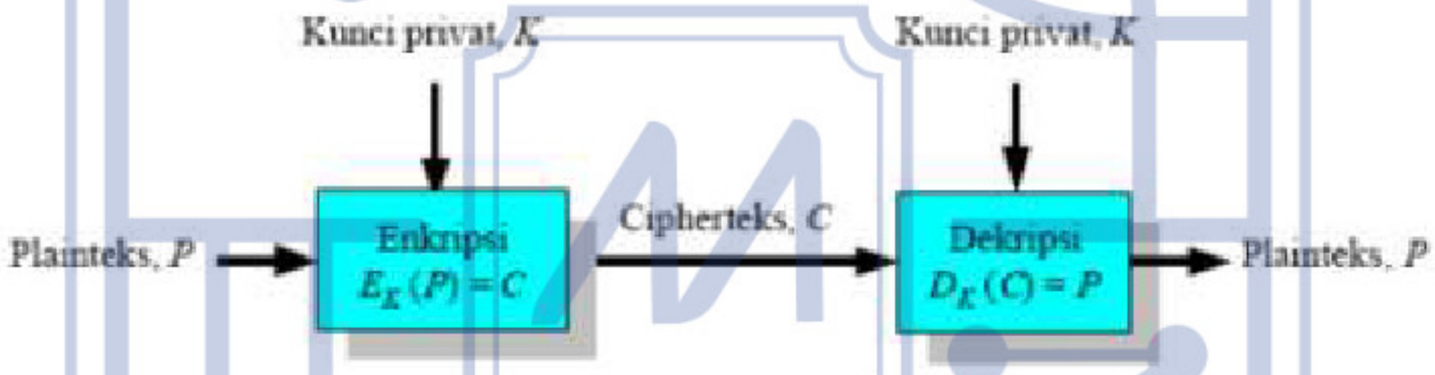
Proses enkripsi menjadi dekripsi yaitu mengebalikan pesan ke pesan asal adalah

$$D(E(P))=P \tag{3}$$

2.2.1 Kriptografi Simetris

Kriptografi simetris menggunakan kunci yang sama untuk enkripsi dan dekripsinya. Algoritma kriprografi simetris sering disebut algoritma kunci rahasia, algoritma kunci tunggal, atau algoritma satu kunci, dan mengharuskan pengirim dan penerima menyetujui suatu kunci tertentu [15],

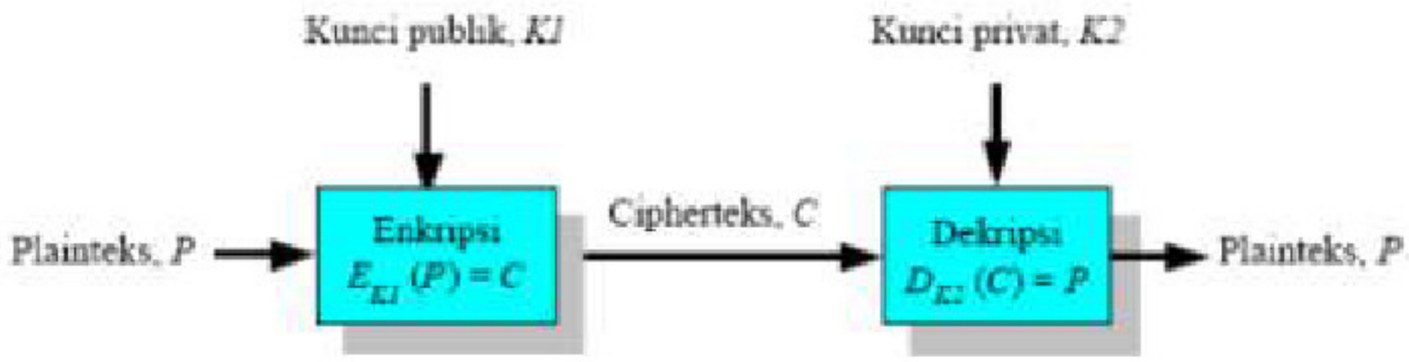
Algoritma kriptografi simetris terdiri dari 2 kategori yaitu algoritma aliran (stream ciphers) dan algoritma blok (block ciphers). Algoritma aliran proses penyandiannya berorientasi pada satu bit (satu byte data). Sedangkan algoritma blok, proses penyandiaannya berorientasi pada sekumpulan bit (per blok) [16].



Gambar 2.2 Skema Algoritma Simetris *[17]*

2.2.2 Kriptografi Asimetris

Pada kriptografi asimetris, pasangan kunci kriptografi yang salah satunya digunakan sebagai proses enkripsi dan dekripsi. Semua orang yang mendapatkan kunci publik dapat menggunakannya guna mengenkripsi suatu pesan, dan hanya satu orang saja yang memiliki rahasia itu dimana dalam hal ini, kunci rahasia digunakan untuk melakukan pembongkaran terhadap kode yang dikirim untuknya [15]. Algoritma kriptografi asimetrik adalah algoritma yang menggunakan kunci yang berbeda untuk proses enkripsi dan dekripsinya. Algoritma ini disebut juga algoritma umum (public key algorithm) atau dapat diketahui oleh setiap orang. Untuk dekripsi hanya diketahui oleh yang berwenang mengetahui data yang disandikan atau sering disebut kunci pribadi (private key). Contoh algoritma asimetris adalah RSA dan ECC [16].



Gambar 2.3 Skema Algoritma Asimetris *[17]*

2.2.3 Kriptografi Hibrid

Kriptografi hibrid memanfaatkan dua tingkatan kunci, yaitu kunci rahasia (simetri) yang disebut juga *session key* (kunci sesi) untuk enkripsi data dan pasangan kunci rahasia – kunci publik untuk pemberian tanda tangan digital serta melindungi kunci simetri [15]. Kriptografi hybrid merupakan pendekatan yang menggabungkan algoritma kriptografi simetris dan asimetris untuk meningkatkan keamanan data [18]. Kelemahan dari algoritma kriptografi simetris terletak pada mekanisme distribusi kunci. Jika kunci rahasia berhasil diketahui oleh pihak yang tidak berwenang, seluruh komunikasi yang menggunakan kunci tersebut dapat didekripsi. Oleh karena itu, dalam aplikasi praktis, kriptografi simetris umumnya dipadukan dengan mekanisme pertukaran kunci yang aman untuk mendistribusikan kunci secara terlindungi. Sementara itu, meskipun memiliki keunggulan dalam distribusi kunci, kriptografi kunci asimetris memiliki kompleksitas komputasi yang lebih tinggi dibandingkan kriptografi simetris. Hal ini disebabkan oleh penggunaan operasi matematika yang lebih kompleks seperti eksponensiasi modular atau operasi pada kurva eliptik. Oleh karena itu, dalam praktiknya algoritma asimetris lebih sering digunakan untuk proses pertukaran kunci atau autentikasi, bukan untuk enkripsi data dalam jumlah besar. Dengan adanya kelemahan dan kelebihan dari kriptografi simetris dan asimetris, kedua metode bisa dioptimalkan dengan mengkombinasikan kedua metode kriptografi tersebut yaitu *Hybrid Cryptography*. Algoritma kriptografi *hybrid* dapat digunakan untuk mengamankan properti rahasia aplikasi [19].

Pada pendekatan kriptografi hybrid untuk mengamankan properti rahasia, terdapat suatu *secret key* yang dibangkitkan dengan algoritma kriptografi simetri seperti AES. *Secret key* tersebut digunakan untuk mengenkripsi properti rahasia sehingga properti yang diletakkan pada file konfigurasi merupakan properti yang sudah terenkripsi dan perlu untuk didekripsi dengan menggunakan *secret key* yang sama sewaktu membaca properti tersebut dari file konfigurasi. Untuk meningkatkan keamanan aplikasi, *secret key* kriptografi simetri sebaiknya diletakkan pada tempat penyimpanan yang berbeda dari aplikasi. Kriptografi

asimetris atau kunci publik seperti RSA dapat digunakan untuk mengenkripsi *secret key*. Ketika mengambil *secret key* yang telah dienkripsi dengan kunci publik, *secret key* tersebut perlu untuk didekripsi pada aplikasi dengan menggunakan kunci privat. Mekanisme tersebut berperan untuk mengamankan jalur pertukaran *secret key* pada tempat penyimpanan yang terpisah dari aplikasi [18].

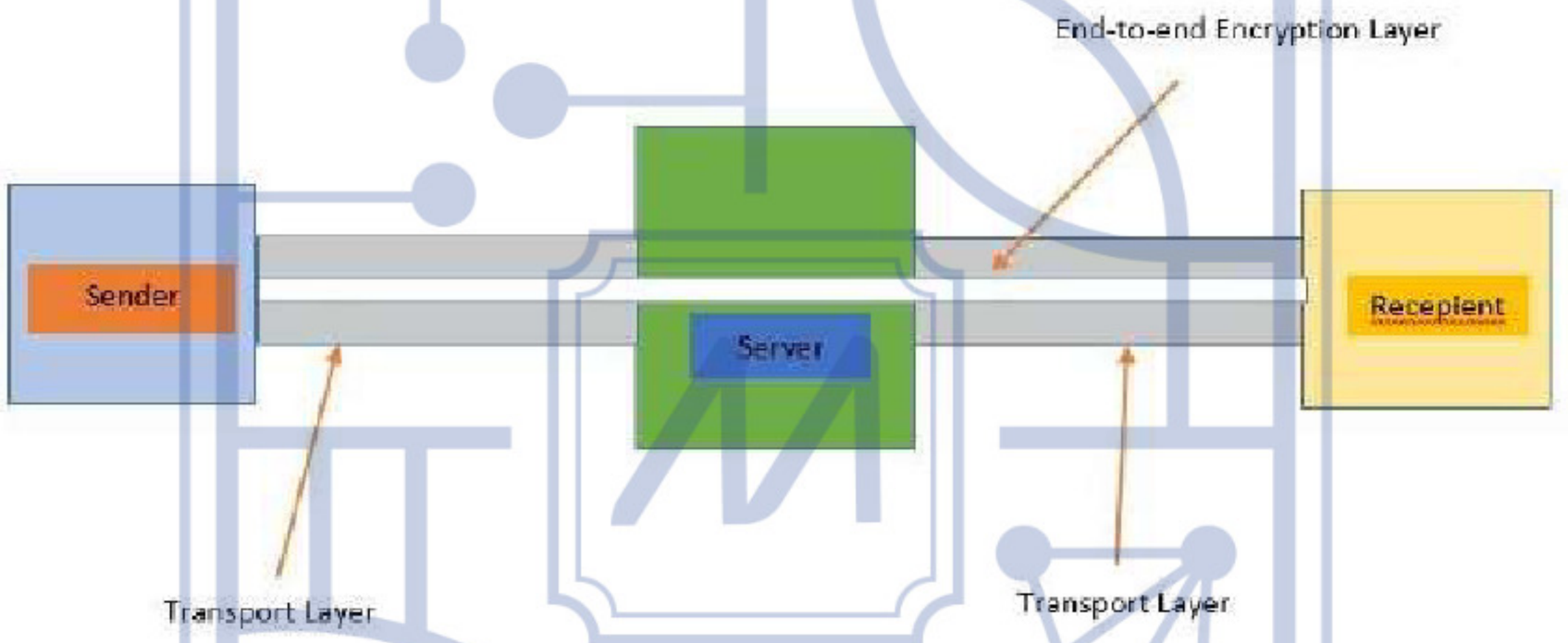
2.3 End-to-End Encryption (E2EE)

Enkripsi End-to-End (E2EE) muncul sebagai mekanisme perlindungan yang memungkinkan hanya penerima dan pengirim yang bisa dapat mengakses konten komunikasi dari server, sedangkan pengguna ketiga tidak bisa mengakses data tersebut [20]. Enkripsi End-to-End (E2EE) dikembangkan untuk mengatasi keterbatasan mekanisme keamanan cloud konvensional. Dalam skema ini, data dienkripsi di sisi pengirim dan hanya dapat didekripsi oleh penerima yang memiliki kunci yang sah, sehingga server perantara tidak dapatmengakses isi komunikasi. Penelitian terdahulu menunjukkan bahwa E2EE secara konseptual mampu mengurangi risiko kebocoran data akibat pelanggaran keamanan pada server, karena data tetap berada dalam kondisi terenkripsi sepanjang proses transmisi dan penyimpanan. Namun, efektivitas E2EE tidak hanya ditentukan oleh kekuatan algoritma kriptografi, melainkan juga oleh manajemen kunci, mekanisme autentikasi, serta keamanan perangkat pengguna sebagai endpoint. Dengan demikian, E2EE perlu dipahami sebagai pendekatan sistemik, bukan sekadar teknik enkripsi [20].

Sejumlah studi empiris melaporkan bahwa penerapan E2EE dapat menurunkan tingkat eksposur data terhadap serangan berbasis jaringan, seperti penyadapan dan serangan man-in-the-middle. Penelitian komparatif menunjukkan bahwa layanan komunikasi yang menerapkan E2EE memiliki tingkat perlindungan yang lebih tinggi terhadap akses tidak sah dibandingkan dengan layanan yang menggunakan *server-side encryption*. Meskipun demikian, beberapa penelitian juga menegaskan bahwa E2EE tidak sepenuhnya menghilangkan risiko kebocoran data. Kerentanan masih dapat terjadi pada sisi endpoint, termasuk pencurian kunci kriptografi, infeksi malware, serta kelemahan dalam proses verifikasi identitas pengguna. Selain itu, implementasi E2EE yang bersifat tertutup dan minim dokumentasi berpotensi menghambat proses evaluasi dan audit keamanan secara independen. Dibandingkan dengan mekanisme keamanan lain seperti SSL/TLS dan server-side encryption, E2EE menawarkan perlindungan yang lebih komprehensif terhadap kebocoran data. SSL/TLS berfokus pada pengamanan data selama proses transmisi, namun

tidak mencegah akses data oleh server dalam bentuk plaintext. Server-side encryption juga masih menyisakan risiko apabila infrastruktur cloud mengalami kompromi keamanan [20].

Namun, beberapa penelitian mencatat bahwa penerapan E2EE dapat membatasi fungsionalitas layanan cloud, terutama dalam pemrosesan data berbasis server, seperti pencarian isi pesan dan analisis data. Hal ini menunjukkan adanya trade-off antara peningkatan keamanan dan kemampuan operasional layanan, yang perlu dipertimbangkan dalam evaluasi efektivitas E2EE [20]. Fitur *end-to-end encryption* pada dasarnya melindungi data pengguna pada saat dikirim dan dibuka pada saat diterima. Untuk mempermudah pemahaman konsep tersebut dapat dilihat pada gambar 2.4 berikut ini [21]:



Gambar 2.4 Konsep Dasar Enkripsi *End to End [21]*

Pada gambar diatas ditunjukkan bahwa fitur enkripsi end-to-end seolah-olah memiliki jalur sendiri untuk berkomunikasi. Tetapi jalur tersebut hanya perumpaan saja dikarenakan jalur tersebut sebagai pembungkusan data pada komunikasi pengguna. Pada jurnal ini akan dijelaskan lebih detail mengenai proses yang terjadi pada pembungkusan data serta algoritma yang dipakai oleh fitur keamanan pada Whatsapp *end-to-end encryption*. Sistem ini menjamin pesan yang dikirim antara dua pihak, tidak dapat diintip oleh orang lain yang mencegat jalur komunikasi. Jalur komunikasi pesan terenkripsi didukung oleh *Transport Layer Security* yang menghubungkan *web client* dan *software web server*. Namun, enkripsi biasa saja tidak cukup, itu sebabnya *client software* melapisi pesan dengan enkripsi sebelum dikirim ke *web client*. Pesan tersebut hanya bisa dibuka oleh penerima [21].

2.4 Metode ChaCha20

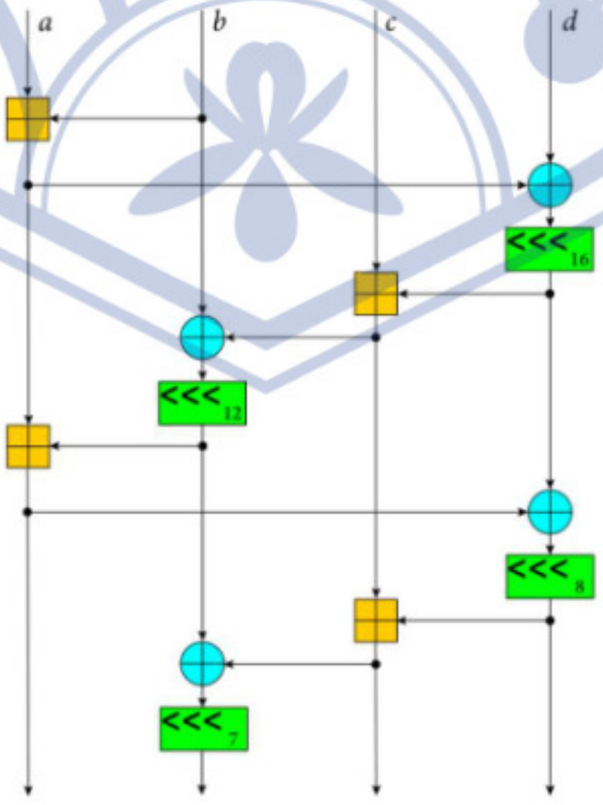
ChaCha20 adalah algoritma kriptografi simetris berbasis stream cipher yang dikembangkan oleh Daniel J. Bernstein pada 2008, sebagai pengembangan dari Salsa20.

Algoritma ChaCha20 menunjukkan fleksibilitas dan efektivitasnya dalam berbagai aplikasi keamanan, mulai dari sistem tertanam, keamanan jaringan sensor nirkabel hingga pengamanan mesin virtual pusat data, termasuk penggunaannya dalam protokol SSL/TLS oleh perusahaan seperti Google dan Cloudflare. Algoritma ini meningkatkan difusi per putaran dengan tetap mempertahankan kinerja tinggi. ChaCha20 memanfaatkan fungsi pseudorandom berbasis operasi Add-Rotate-XOR dan menggunakan struktur initial state berupa matriks 4x4 yang terdiri dari 16 word berukuran 32-bit (32-bit unsigned integers), dengan total ukuran 512-bit (64 byte), untuk menghasilkan keystream. Masukan utamanya mencakup kunci 256-bit, konstanta 128-bit, nonce 96-bit, dan counter 32-bit. Algoritma ini mendukung kunci 128-bit atau 256-bit, dengan konstanta "expand 32-byte k" dan nonce sebagai bilangan acak sekali pakai. Dalam ChaCha20, word disimpan dalam format *little-endian*. Adapun susunan matriks-nya disusun seperti pada tabel 2.1 berikut [22]:

Tabel 2.1 Susunan Matriks Metode ChaCha20 [22]

Cons	Cons	Cons	Cons
Key	Key	Key	Key
Key	Key	Key	Key
Counter	Nonce	Nonce	Nonce

ChaCha20 memiliki sebuah fungsi inti bernama Quarter Round. Ini beroperasi berdasarkan prinsip ARX (Add-Rotate-XOR), yaitu operasi bitwise XOR (⊕), 32-bit penambahan modulus 2^32 ⊞, dan operasi rotasi jarak konstan ke kiri, yang dinyatakan sebagai "<<< n," di mana n merepresentasikan jumlah bit yang diputar ke kiri (menuju bit bernilai tinggi) [22].



Gambar 2.5 Fungsi *Quarter-Round ChaCha20 [22]*

Dari Gambar 2.5 diatas, fungsi Quarter Round dapat didefinisikan sebagai QR(a, b, c, d), dimana :

$$a \mathrel{+}= b;\ d \mathrel{\hat{}}= a;\ d \lll= 16; \tag{4}$$

$$c \mathrel{+}= d;\ b \mathrel{\hat{}}= c;\ b \lll= 12; \tag{5}$$

$$a \mathrel{+}= b;\ d \mathrel{\hat{}}= a;\ d \lll= 8; \tag{6}$$

$$c \mathrel{+}= d;\ b \mathrel{\hat{}}= c;\ b \lll= 7; \tag{7}$$

dengan a, b, c, d mewakili posisi elemen dalam matriks (word) pada baris dan kolom tertentu. Setiap nilai pada matriks di urutan tertentu per Quarter Round akan dilakukan operasi penambahan (+), XOR (^) dan rotasi jarak konstan (<<<) [22].

ChaCha20 menggunakan fungsi Quarter Round untuk mengacak masukan Initial State menjadi keystream. Prosesnya dimulai dengan mengubah masukan pada initial state ke format little-endian sebelum dikirim ke generator pseudo-random melalui fungsi Quarter Round. Blok matriks initial state 512-bit akan diacak melalui 80 kali aplikasi fungsi Quarter Round. Dalam setiap Quarter Round, 4 word 32-bit (128-bit) diacak, dan proses ini diulang hingga seluruh blok teracak. ChaCha20 menjalankan 20 putaran, yang terdiri dari 10 double round. Seperti terlihat pada Gambar 2.6, setiap double round mencakup 2 putaran: putaran ganjil yang terdiri dari 4 quarter round kolom dan putaran genap dengan 4 *quarter round* diagonal. ChaCha20 melakukan 10 double round, setara dengan 20 putaran atau 80 aplikasi *quarter round* [22].

(a). Satu round ganjil (kolom) (b). Satu round genap (diagonal)

Gambar 2.6 *Double Round [22]*

Hasil putaran terakhir dijumlahkan dengan initial state asli dalam format little-endian melalui penambahan. Pada tahap akhir, hasil penjumlahan disimpan kembali dalam format little-endian sebagai keystream yang siap digunakan. Mirip Salsa20, setiap blok keystream ChaCha20 bersifat independen, memungkinkan akses acak dan komputasi paralel untuk beberapa blok secara simultan. Jika plaintext panjangnya melebihi 512-bit (64-byte), ChaCha20 menghasilkan keystream tambahan dengan meng-increment nilai block tabel

untuk setiap blok yang diperlukan. Setiap blok keystream di-XOR dengan bagian plaintext yang sesuai untuk menghasilkan ciphertext. ChaCha20 menggunakan keystream untuk mengenkripsi dan mendekripsi data. Dalam enkripsi, plaintext di-XOR dengan keystream yang dihasilkan dari fungsi pseudorandom berbasis kunci dan nonce, sementara dalam dekripsi, ciphertext di-XOR dengan keystream yang sama untuk memperoleh plaintext. Hubungan ini dijelaskan dengan persamaan 5 dan 6 dibawah ini [22].

$$ciphertext = plaintext \oplus keystream\ (key, nonce) \quad (8)$$

$$plaintext = ciphertext \oplus keystream\ (key, nonce) \quad (9)$$

ChaCha20 merupakan algoritma *stream cipher* yang dikenal memiliki performa tinggi dan efisiensi yang baik, terutama dalam lingkungan perangkat bergerak. Proses algoritmik ChaCha20 terdiri dari serangkaian tahapan kriptografi yang menghasilkan keystream untuk melakukan operasi XOR dengan pesan asli. Tahapan-tahapan inti algoritma ChaCha20 dijelaskan sebagai berikut [23]:

1. *Setup Stage*

   ChaCha20 menghasilkan keystream 512-bit dari matriks awal 4 × 4 yang terdiri dari 16 unit operasi dasar 32-bit. Fungsi ChaCha20 menerima input berupa 4 konstanta {c0, c1, c2, c3}, 8 kunci {k0, k1,...k7}, 1 counter {t0}, dan 3 nonce {n0, n1, n2} yang diatur sebagai berikut:

$$X[16] = \begin{bmatrix} x_0 & x_1 & x_2 & x_3 \\ x_4 & x_5 & x_6 & x_7 \\ x_8 & x_9 & x_{10} & x_{11} \\ x_{12} & x_{13} & x_{14} & x_{15} \end{bmatrix} = \begin{bmatrix} c_0 & c_1 & c_2 & c_3 \\ k_0 & k_1 & k_2 & k_3 \\ k_4 & k_5 & k_6 & k_7 \\ t_0 & n_0 & n_1 & n_2 \end{bmatrix}$$

2. *Quarter Round*

   Quarter Round (QR) merupakan inti utama dalam proses pembaruan matriks state pada setiap putaran algoritma ChaCha20. Proses ini dilakukan dalam dua tahap, yaitu penerapan QR pada kolom-kolom matriks terlebih dahulu, kemudian dilanjutkan pada diagonal-diagonalnya.

Pemilihan ChaCha20 sebagai algoritma simetris pada penelitian ini didasarkan pada beberapa keunggulan: pertama, ChaCha20 hanya menggunakan operasi aritmatika sederhana seperti addition, rotation, dan XOR (ARX), sehingga tidak bergantung pada akselerasi perangkat keras (*hardware acceleration*) tertentu; kedua, algoritma ini telah menunjukkan performa yang kompetitif dibandingkan dengan AES dan algoritma simetris lainnya, terutama pada lingkungan perangkat lunak; dan ketiga, ChaCha20 sangat sesuai untuk perangkat *mobile* yang memiliki keterbatasan sumber daya. ChaCha20 banyak

diadopsi dalam protokol komunikasi modern karena mampu memberikan performa stabil pada perangkat tanpa dukungan *hardware acceleration*, sehingga sangat sesuai untuk implementasi aplikasi berbasis *web* dan *mobile*. Hal ini menjadikan ChaCha20 relevan untuk penelitian ini yang menguji sistem dalam lingkungan simulasi berbasis *web*.

2.5 Metode Elliptic Curve Diffie-Hellman Ephemeral

Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) merupakan mekanisme pertukaran kunci berbasis kriptografi kurva eliptik yang menghasilkan kunci sesi sementara (*ephemeral session key*) pada setiap sesi komunikasi. Penggunaan kunci yang bersifat sementara ini mendukung konsep *forward secrecy*, yaitu kebocoran kunci pada satu sesi tidak dapat digunakan untuk mendekripsi sesi komunikasi lainnya [24].

Dalam skema kriptografi hibrid yang digunakan pada penelitian ini, ECDHE bertanggung jawab untuk membentuk kunci sesi yang aman, sementara ChaCha20 mengenkripsi isi pesan. Kombinasi ini memastikan bahwa setiap sesi komunikasi memiliki kunci yang unik dan sementara, mencegah penggunaan kunci untuk dekripsi sesi lain. Penggunaan kurva eliptik dalam ECDHE memberikan tingkat keamanan yang setara dengan algoritma berbasis faktorisasi bilangan besar, namun dengan ukuran kunci yang lebih kecil dan efisiensi komputasi yang lebih tinggi. Implementasi ECDHE berbasis Curve25519 mampu memberikan performa yang optimal sekaligus mempertahankan tingkat keamanan tinggi, terutama melalui dukungan *forward secrecy* yang mencegah kompromi antar sesi komunikasi [25]. Oleh karena itu, pemilihan ECDHE dalam penelitian ini didasarkan pada pertimbangan efisiensi dan keamanan yang seimbang untuk aplikasi *chatting* berbasis *web*.

2.5.1 Metode Elliptic Curve Diffie-Hellman

Elliptic Curve Cryptography (ECC) dikembangkan secara independen oleh Victor Miller pada tahun 1986 dan oleh Neil Koblitz pada tahun 1987. Sejak itu, ECC telah dievaluasi oleh matematikawan dan pakar komputer di seluruh dunia, dan ECC banyak digunakan untuk keamanan *cryptocurrency*. Karena panjang kunci yang pendek, kecepatan tinggi, dan konsumsi daya yang rendah, *Elliptic Curve Cryptography* (ECC) diterapkan secara luas pada perangkat dengan ruang penyimpanan dan bandwidth yang terbatas [26].

Elliptic-Curve Diffie-Hellman (ECDH) membangun rahasia bersama yang digunakan sebagai kunci antara dua pihak dengan membuat protokol perjanjian kunci publik dan privat berdasar kurva eliptik pada saluran yang tidak aman. Kunci tersebut kemudian dapat digunakan untuk mengenkripsi komunikasi yang kemudian menggunakan sandi kunci

simetris. Ini adalah varian dari protokol Diffie-Hellman yang menggunakan kriptografi kurva eliptik. Pertukaran kunci yang diterapkan dalam bidang kriptografi untuk enkripsi P_e disebut kunci publik yang dapat didistribusikan secara bebas di saluran yang tidak aman, sedangkan kunci dekripsi P_d disebut kunci privat yang bersifat rahasia dan harus dijaga kerahasiaannya [27].

Kriptografi kurva eliptik menawarkan tingkat keamanan yang sama dengan algoritme kriptografi kunci publik konvensional, tetapi dengan ukuran kunci yang lebih pendek dan menunjukkan daya tarik yang tersembunyi. Perbandingan *eliptic curve cryptography* (ECC) dengan RSA, panjang kunci ECC lebih pendek dari RSA, misalnya kunci ECC 160 bit memberikan keamanan yang sama dengan RSA 1024 bit kunci. Operasi aritmatika pada kriptografi kriptografi yang berdasarkan kurva eilptic tidak menggunakan bilangan real, tetapi kriptografi beroperasi pada ranah bilangan bulat. Dalam kriptografi teks biasa, teks tersandi, dan kunci dinyatakan sebagai bilangan bulat. Oleh karena itu, untuk kurva eliptik yang akan digunakan dalam sistem keamanan data, kurva eliptik didefinisikan dalam bidang hingga atau Galois Field $GF(p)$ atau $GF(2^m)$. Bentuk umum dari kurva eliptik di $GF(p)$ atau $GF(2^m)$ adalah $y^2=x^3+ax+b$ mod p dengan p adalah bidang berhingga dan unsur-unsur dalam bidang galois adalah {0, 1,2, ..., p-1} dimana operasi penjumlahan dan perkalian dilakukan dengan modulus p [27].

Metode yang digunakan dalam pertukaran kunci bersama Diffie-Hellman untuk kriptografi kurva eliptik yang dianalisis dan digunakan dalam penelitian ini diambil dari https://asecuritysite.com/encryption/js08. Sebelum dijelaskan lebih lanjut, misalkan Alice ingin membuat menyampaikan kunci sama Bob pada saluran yang tidak aman maka langkah-langkahnya sebagai berikut [27]:

Pilih parameter domain yang kuat secara kriptografi yaitu, , (p,a,b,G,n,h) dalam kasus utama case $(m,f\ (x),a,b,G,n,h)$ dalam kasus biner harus disepakati. Parameter sistem harus dipertukarkan secara otentik antara pihak-pihak yang terlibat dalam komunikasi.

A. Kesepakatan kunci

Perjanjian kunci juga harus diamankan dengan otentikasi yang kuat. dengan prosedur sebagai berikut:

1. Setiap pihak harus memiliki pasangan kunci yang sesuai untuk kriptografi kurva eliptik, yang terdiri dari kunci pribadi d (bilangan bulat dipilih secara acak dalam interval $[1,n-1]$ dan kunci publik yang diwakili oleh titik Q (di mana $Q = d.Q$), yaitu, hasil penambahan G ke waktu itu sendiri).

2. Izinkan pasangan kunci Alice (dA,QA) pasangan kunci Bob menjadi (dB,QB) di mana setiap pihak harus mengetahui kunci publik pihak lain sebelum menjalankan protocol.
3. Alice menghitung poin (xk,yk) = $dA.QB$ dan Bob menghitung poin points (xk,yk) = $dB.QA$ dimana rahasia bersama adalah xk (koordinat x poin). Kebanyakan protokol standar berdasarkan ECDH berasal dari kunci x_ksymmetric menggunakan beberapa fungsi derivasi kunci berbasis hash.
4. Rahasia bersama yang dihitung oleh kedua belah pihak adalah sama, karena $dAQB$=$dA.dB.G$=$dB.dA.G$=$dBQA$.

B. Algoritme penghasil kunci ECDH

Domain parameter kurva eliptik di atas F_p didefinisikan sebagai persamaan $T(p,a,b,G,n,h)$, di mana p adalah bidang yang didefinisikan kurva, , a,b koefisien persamaan kurva elips, G generator titik adalah elemen pembangun kelompok, n adalah orde utama dari G yaitu bilangan bulat positif terkecil adalah nG=0, dan h kofaktor, banyaknya titik dalam kelompok eliptik $Ep(a,b)$ dibagi n.

Algoritma 1	Algoritma penghasil kunci ECDH
Input:	Parameter domain (p, a, b, G, n, h)
Output:	Kunci private: d_A, d_B dan Kunci publik: Q_A, Q_B
1.	Pilih bilangan bulat $d_A, d_B \in [1, n-1]$
2.	User A menghitung $Q_A = d_A.G$ send to User B
3.	User B menghitung $Q_B = d_B.G$ send to User A
4.	User A menghitung $K = d_A.Q_B = d_A(d_B.G)$
5.	User B menghitung $K' = d_B.Q_A = d_B(d_A.G)$

Kemudian proses selanjutnya adalah proses enkripsi pesan dengan ECDH yang merupakan varian dari algoritma Diffie-Hellman untuk kurva elips. Masalah yang dia selesaikan adalah sebagai berikut: dua pihak (biasanya Alice dan Bob) ingin bertukar informasi dengan aman, sehingga pihak ketiga tidak dapat menguraikan kode mereka. Selanjutnya adalah proses enkripsi ECDH.

Algoritma 2	Algorimta enkripsi ECDH
Input:	Parameter domain (p, a, b, G, n, h)
	Kunci privat: x, y dan kunci publik: P_A, P_B, plaintext M
Output:	Chipertext: C
1.	Hitung $S = yP_A = x.P_B$
2.	Hitung C= $M + S$

Dalam desain sistem dekripsi ini, akan mengembalikan pesan terenkripsi ke pesan asli lagi. Proses dekripsi kemudian dilakukan dengan cara mereduksi pesan rahasia.

Algoritma 3	Algorimta dekrispsi ECDH
Input:	Parameter (p, a, b, G, n, h), kunci rahasia bersama S, chipertext C
Output:	Plaintext M
1.	Hitung $M = C - S$
2.	Plaintext $M = C - S$

2.5.2 Prosedur Kerja dari Metode Elliptic Curve Diffie-Hellman Ephemeral

Pada ECDHE, masing-masing pihak membangkitkan pasangan kunci privat dan publik secara sementara untuk setiap sesi. Setelah pertukaran kunci publik, kedua pihak menghitung *shared secret* menggunakan kunci privat masing-masing dan kunci publik lawan. Nilai *shared secret* yang dihasilkan kemudian digunakan untuk menghasilkan kunci sesi pada algoritma kriptografi simetris [25].

Penggunaan kurva eliptik Curve25519 pada ECDHE dipilih karena efisiensi komputasi yang tinggi dan tingkat keamanan yang kuat [25]. Keamanan mekanisme pembentukan shared secret pada ECDHE bergantung pada kesulitan permasalahan matematika yang dikenal sebagai *Elliptic Curve Discrete Logarithm Problem* (ECDLP). Permasalahan ini menyatakan bahwa, meskipun kunci publik diketahui, sangat sulit secara komputasional untuk menentukan kunci privat yang bersesuaian. Penggunaan kurva eliptik modern seperti Curve25519 memberikan tingkat keamanan yang tinggi dengan ukuran kunci yang relatif kecil dibandingkan algoritma berbasis faktorisasi bilangan besar [25].

Dengan demikian, selama parameter kurva dipilih dengan benar dan implementasi dilakukan secara aman, shared secret yang dihasilkan melalui ECDHE tidak dapat direkonstruksi oleh pihak ketiga yang hanya mengamati pertukaran kunci publik.

Kurva yang disajikan dalam penelitian ini adalah Curve25519. Kurva tersebut didefinisikan dalam *field* prima dengan bilangan prima p yang mendekati pangkat 2 untuk mengoptimalkan efisiensi komputasi modulo. Parameter yang digunakan dalam Curve25519 dapat dilihat pada tabel 2.2.

Tabel 2.2 Parameter Curve25519 *[24]*

Parameter	Value
P	$2^{255} - 19$
a	486662
u(S)	09
v(S)	14781619447589544791020593568409986887264606134606134616475288964881837755586237401
n	2^{252} + 0x14def9dea2f79cd65812631a5cf5d3ed
h	08

ECDHE adalah metode yang digunakan selama pertukaran kunci antara *server* dan klien sehingga hasilnya adalah rahasia pra-master yang diketahui oleh kedua belah pihak. Selama jabat tangan, setelah *server* dan klien menyepakati rangkaian *cipher* dan kurva mana yang akan digunakan, keduanya mengetahui domain parameter, termasuk nilai-nilai berikut:

- p: bilangan prima p yang didefinisikan untuk bidang Fp.
- a: parameter persamaan kurva.
- S(u, v): koordinat titik dasar (generator).
- n: orde S, didefinisikan sebagai bilangan bulat terkecil yang memenuhi nS = 0.
- h: kofaktor.

Proses perjanjian rahasia pra-master adalah sebagai berikut:

1. *Server* memilih bilangan acak k ($0 < k < n$). Kemudian, *server* menghitung kS dan mengirimkannya ke klien.
2. Klien memilih bilangan acak k' ($0 < k' < n$). Klien menghitung k'S dan mengirimkannya ke *server*.
3. *Server* dan klien, setelah memperoleh data sementara k atau k' dari pihak lain, dapat menghitung rahasia pra-master P = kk'S = k'kS.

Penggunaan sifat *ephemeral* pada ECDHE memastikan bahwa setiap sesi komunikasi memiliki kunci yang berbeda, sehingga memberikan perlindungan *forward secrecy* apabila kunci jangka panjang sistem berhasil dikompromikan. *Forward secrecy* memastikan bahwa kebocoran kunci jangka panjang atau data lainnya di masa depan tidak dapat digunakan untuk mendekripsi komunikasi yang telah terjadi sebelumnya. Karena setiap sesi komunikasi menggunakan kunci yang berbeda dan bersifat sementara, ECDHE memberikan perlindungan yang kuat terhadap analisis lalu lintas komunikasi jangka panjang. Dengan demikian, meskipun kunci privat jangka panjang (misalnya, kunci identitas pengguna) berhasil dikompromikan di masa depan, pesan-pesan yang telah dikirimkan tetap aman karena menggunakan kunci sesi yang telah dihapus [25].

Konsep *forward secrecy* menjadi komponen penting dalam protokol komunikasi modern karena mampu membatasi dampak kebocoran kunci jangka panjang. Dengan penggunaan ECDHE yang bersifat ephemeral, setiap sesi komunikasi pada penelitian ini menghasilkan kunci yang unik dan tidak dapat digunakan kembali. Hal ini memastikan bahwa meskipun terjadi kompromi terhadap salah satu sesi, sesi lainnya tetap terlindungi dan tidak terpengaruh.

2.6 Metode SHA-256

Kemajuan dalam kriptanalisis fungsi hash kriptografi cukup lambat hingga baru-baru ini, komunitas kriptografi dikejutkan dengan kemajuan kriptanalisis fungsi hash, seperti serangan terhadap MD5 untuk menemukan tabrakan (collision) dan serangan dengan strategi baru pada SHA- 0 dan serangan untuk menemukan multi-tabrakan. Namun, teknik ini tidak berlaku untuk SHA-256 karena jadwal pesan dan fungsi putarannya yang lebih kompleks [9].

Fungsi hash SHA merupakan fungsi hash lain yang sering digunakan untuk keperluan kriptografi. Jika pada MD5, nilai hash yang dihasilkan memiliki panjang 128-bit, pada SHA-1 nilai yang dihasilkan memiliki ukuran 160 bits. Apabila seseorang mencoba untuk memecahkan kode enkripsi pada fungsi hash SHA-1, maka membutuhkan jumlah percobaan sebanyak 2160 setara dengan 1.461.501.637.330.902.918.203.684. 832.716.283.019.655.932.542.976. Selain SHA-1, adapula SHA-256 yang sering digunakan. Lebih dari SHA-1, SHA256 memiliki ukuran 256 bits. SHA-256 adalah fungsi hash kriptografi yang diusulkan pada tahun 2000 sebagai generasi baru fungsi SHA dan diadopsi sebagai standar FIPS pada tahun 2002. SHA-256 dibangun dari konstruksi MD (Merkle-Damg ard) dan mode Davis-Meyer. Fungsi kompresi SHA-256 memiliki 64 putaran, dua jenis fungsi non-linier, rotasi siklik, dan konstanta yang bergantung pada putaran. SHA256 lebih sering digunakan dibandingkan SHA-1 karena kemampuan enkripsinya yang lebih baik karena menghasilkan nilai hash yang lebih rumit untuk dipecahkan. Untuk memecahkan nilai hash dengan fungsi SHA256, dibutuhkan jumlah percobaan sebanyak 2^{256} atau sekitar 115.792.089.237.316.195.423.570.985.008.687.907.853.269.984.665.640.564.039.457.584 .007.913.129.639.936 [9].

2.6.1 Fungsi Hash

Hashing merupakan sebuah metode yang digunakan untuk mentransformasi suatu string menjadi karakter yang nilainya merepresentasikan string aslinya dengan operasi aritmatika. Dari segi Bahasa, hashing berasal dari kata hash yang berarti memenggal lalu menggabungkan. Dalam penggunaannya, metode hashing dapat digunakan sebagai metode penyimapanan data dalam sebuah larik (array) agar penyimpanan, pencarian, penambahan, dan penghapusan data dapat dilakukan dengan cepat. Dasar proses dalam metode hashing adalah dengan menghitung posisi record yang dicari di dalam array dan bukan membandingkan record denga nisi pada array. Hal ini yang membuat proses hashing dapat

mempercepat dalam proses pencarian data. Fungsi yang mengembalikan nilai atau kunci disebuah fungsi hash (hash function) dan array yang digunakan disebut tabel hash (hash table) [9].

Fungsi hash menyimpan nilai asli atau kunci pada alamat yang sama dengan nilai hashnya. Pada pencarian suatu nilai pada tabel hash, yang pertama dilakukan adalah menghitung nilai hash dari kunci atau nilai aslinya, kemudian membandingkan kunci atuau nilai asli dengan isi pada memori yang beralamat nomor hashnya. Dengan cara ini, pencarian suatu nilai dapat dilakukan dengan cepat tanpa harus memeriksa seluruh isi tabel satu per satu. Secara teori, kompleksitas waktu (T(n)) dari fungsi hash yang ideal adalah O(1). Untuk mencapai itu setiap record membutuhkan suatu kunci yang unik. Fungsi Hash (dilambangkan dengan h(k)) bertugas untuk mengubah k (key) menjadi suatu nilai dalam interval [0....X], dimana "X" adalah jumlah maksimum dari record-record yang dapat ditampung dalam tabel. Jumlah maksimum ini bergantung pada ruang memori yang tersedia. Fungsi Hash yang ideal adalah mudah dihitung dan bersifat random, agar dapat menyebarkan semua key. Dengan key yang tersebar, berarti data dapat terdistribusi secara seragam bentrokan dapat dicegah. Sehingga kompleksitas waktu model Hash dapat mencapai O(1), di mana kompleksitas tersebut tidak ditemukan pada struktur model lain [9].

Selain digunakan pada penyimpanan data, fungsi hash juga digunakan pada algoritma enkripsi sidik jari digital (fingerprint) untuk mengautentifikasi pengirim dan penerima pesan. Sidik jari digital diperoleh dengan fungsi hash, kemudian nilai hash dan tanda pesan yang asli dikirim kepada penerima pesan. Dengan menggunakan fungsi hash yang sama dengan pengirim pesan, penerima pesan mentransformasikan pesan yang diterima. Nilai hash yang diperoleh oleh penerima pesan kemudian dibandingkan dengan nilai hash yang dikirim pengirim pesan. Kedua nilai hash harus sama, jika tidak, pasti ada masalah [9].

Kriptografi juga tak lepas dari prosesnya dalam memanfaatkan fungsi hash. Fungsi hash Kriptografis adalah fungsi hash yang memiliki beberapa sifat keamanan tambahan sehingga dapat dipakai untuk tujuan keamanan data. Umumnya digunakan untuk keperluan autentikasi dan integritas data. Fungsi hash adalah fungsi yang secara efisien mengubah string input dengan panjang berhingga menjadi string output dengan panjang tetap yang disebut nilai hash [9].

Berdasarkan sifatnya, hash kriptografi memiliki sifat sebagai berikut [9]:

1. Tahan preimej (Preimage resistant): bila diketahui nilai hash h maka sulit (secara komputasi tidak layak) untuk mendapatkan m dimana h = hash(m).

2. Tahan preimej kedua (Second preimage resistant): bila diketahui input m1 maka sulit mencari input m2 (tidak sama dengan m1) yang menyebabkan hash(m1) = hash(m2).
3. Tahan tumbukan (Collision-resistant): sulit mencari dua input berbeda m1 dan m2 yang menyebabkan hash(m1) = hash(m2).

2.6.2 Algoritma SHA-256

SHA (Secure hash Algorithm) adalah salah satu algoritma hash yang relatif masih baru. Algoritma ini dirancang oleh The National Institute of Standards and Technology (NIST) pada tahun 2002. SHA – 256 menghasilkan message digest dengan panjang 256 bits. SHA – 256 tergolong aman karena didesain sedemikian rupa sehingga tidak memungkinkan mendapatkan pesan yang berhubungan dengan message digest yang sama. Seperti pada contoh kasus dokumen citra digital hasil pemindaian dari ijazah dan transkrip nilai. Proses untuk menghasilkan message digest pada algoritma ini meliputi lima tahapan [8].

1. Message Padding

   Input pesan pada algoritma SHA-256 akan dibagi menjadi blok-blok yang masing-masing panjangnya adalah 512 bit. Akibat dari pembagian ini maka jumlah blok terakhir akan lebih kecil atau sama dengan 512 bit. Blok terakhir tersebut akan mengalami message padding. Langkah-langkah message padding adalah sebagai berikut:

   a. Diawali dengan masuknya input pesan yang memiliki kode American Standard Code for Information Interchange (ASCII) dan kemudian diubah ke dalam bentuk biner rangkaian bit yang akan dihitung panjangnya.
   b. Rangkaian bit tersebut dibagi menjadi blok-blok yang masing-masing mempunyai panjang 512 bit. Hasil pembagian akan menyebabkan jumlah blok terakhir lebih kecil satu atau sama dengan 512 bit.
   c. Lakukan penambahan bit-bit isian (padding) pada blok terakhir pesan tersebut. Bit-bit yang digunakan sebagai bit isian adalah bit ‘1’ diikuti sejumlah bit ‘0’ sesuai dengan kebutuhan, dengan ketentuan sebagai berikut:
      1) Jika panjang bit blok pesan terakhir lebih kecil dari 448 bit, maka ditambahkan bit ‘1’ pada posisi bit paling akhir, diikuti dengan beberapa bit ‘0’ sedemikian sehingga total panjang bit setelah proses tersebut adalah adalah 448 bit.
      2) Jika panjang bit blok pesan terakhir lebih besar atau sama dengan 448 bit, maka ditambahkan bit ‘1’ pada posisi bit paling terakhir, diikuti dengan beberapa bit ‘0’ sedemikian sehingga total panjang bit setelah proses tersebut 512 bit. Kemudian dibuat 448 bit baru yang isi bitnya ‘0’.

d. Jika panjang bit blok pesan terakhir sama dengan 512 bit, maka harus dibuat blok baru untuk menampung proses message padding. Bit pertama dari blok baru diisi bit'1', sedangkan bit-bit berikutnya sampai dengan panjang bit 448 diisi oleh bit '0'. Jumlah total bit isian yang ditambahkan adalah 448 bit.

2. Penambahan Panjang Bit

Setelah proses *message padding*, jumlah bit pada blok terakhir adalah 448 bit. Representasikan M ke dalam bilangan biner untuk memperoleh 64 bit terakhirnya agar total panjang blok terakhir 512 bit.

a. Urutan byte paling kanan dari nilai representasi panjang pesan (M) dijadikan low order.

b. Tambahkan representasi M tersebut pada 448 bit terakhir, sehingga jumlah panjang blok terakhir adalah 512 bit.

3. Inisialisasi Nilai Hash Awal

Pada SHA-256 untuk menyimpan nilai inisialisasi awal dan nilai output sementara digunakan buffer H0, H1, H2, H3, H4, H5, H6, H7. Di sisi lain, untuk penyimpanan proses sementatat digunakan buffer a, b, c, d, e, f, g, h. Nilai H0, H1, H2, H3, H4, H5, H6, H7 untuk inisialisasi awal dalam notasi heksadesimal.

Tabel 2.3 Inisialisasi Awal dalam Notasi Heksadesimal *[8]*

H0	6a09e667
H1	bb67ae85
H2	3c6ef372
H3	a54ff53a
H4	510e527f
H5	9b0588c
H6	1f83d9ab
H7	5be0cd19

4. Pemrosesan

Pemrosesan merupakan bagian inti yang terdiri atas 1 round mempunyai 64 operasi. Untuk memproses setiap satu blok pesan 512 bit diperlukan 64 operasi. Setiap blok pesan M(1), M(2), ... , M(n) dengan N adalah jumlah blok pesan. Untuk setiap blok pesan akan dilakukan langkah-langkah [28]:

a. Persiapkan penjadwalan pesan.

SHA-256 menggunakan jadwal pesan enam puluh empat kata 32 bit, kata-kata dari jadwal pesan diberi label W0, W1, ..., W63.

$$Wt = \begin{cases} M_t^{(t)} & 0 \le t \le 15 \\ \sigma_1^{(256)}(W_{i-2}) + W_{i-7} + \sigma_0^{(256)}(W_{i-15}) + W_{i-16}, & 16 \le t \le 63 \end{cases}$$

Dimana:

$$\sigma_1^{(256)}(W_{i-2}) = ((W_{i-2})ROTR\ 17) \oplus ((W_{i-2})ROTR\ 19) \oplus ((W_{i-2})SHR10)$$

$$\sigma_0^{(256)}(W_{i-15}) = ((W_{i-15})ROTR\ 7) \oplus ((W_{i-15})ROTR\ 18) \oplus ((W_{i-15})SHR3)$$

Wt = Blok pesan yang baru

Mt = Blok pesan yang lama

Wi-2 = Blok pesan dari *W* ke i-2

Wi-15 = Blok pesan dari *W* ke i-15

ROTR = Rotate Right

SHR = Shift Right

$\oplus$ = Operator *XOR*

b. Inisialisasi working variabel a, b, c, d, e, f, g dan h, untuk M(1) dengan nilai hash awal

   a = H0(I-1)

   b = H1(i-1)

   c = H2(i-1)

   d = H3(i-1)

   e = H4(i-1)

   f = H5(i-1)

   g = H6(i-1)

   h = H7(i-1)

c. Hitung masing-masing penjadwalan.

For t= 0 *to* 63

{

$$T_1 = h + \sum_1^{(256)}(e) + Ch(e,f,g) + K_1^{(256)} + W_t$$

$$T_2 = \sum_0^{(256)}(a) + Maj(a,b,c)$$

$$h = g$$
$$g = f$$
$$f = e$$
$$e = d + T_1$$
$$d = c$$
$$c = b$$
$$b = a$$
$$a = T_1 + T_2$$

}

Dimana:

$$\sum_1^{(256)}(e) = (e\ ROTR\ 6) \oplus (e\ ROTR\ 11) \oplus (e\ ROTR\ 25)$$

$$\sum_0^{(256)}(a) = (a\ ROTR\ 2) \oplus (a\ ROTR\ 13) \oplus (a\ ROTR\ 22)$$

$$Ch(e,f,g) = (e\ \hat{}\ f) \oplus (\sim e\ \hat{}\ g)$$

$$Maj(a,b,c) = (a\ \hat{}\ b) \oplus (a\ \hat{}\ c) \oplus (b\ \hat{}\ c)$$

a, b, c, d, e, f, g, h = Variabel yang berisi pesan heksadesimal
$K_j^{(256)}$ = Konstansta SHA-256
ROTR = *Rotate Right*
$\oplus$ = Operator *XOR*
^ = Operator *AND*

Adapun nilai konstanta SHA-256 dapat dilihat pada tabel berikut:

Tabel 2.4 Konstanta SHA-256 [28]

428A2F98	71374491	B5C0FBCF	E9B5DBA5	3956C25B	59F111F1	923F82A4	AB1C5ED5
D807AA98	12835B01	243185BE	550C7DC3	72BE5D74	80DEB1FE	9BDC06A7	C19BF174
E49B69C1	EFBE4786	0FC19DC6	240CA1CC	2DE92C6F	4A7484AA	5CB0A9DC	76F988DA
983E5152	A831C66D	B00327C8	BF597FC7	C6E00BF3	D5A79147	06CA6351	14292967
27B70A85	2E1B2138	4D2C6DFC	53380D13	650A7354	766A0ABB	81C2C92E	92722C85
A2BFE8A1	A81A664B	C24B8B70	C76C51A3	D192E819	D6990624	F40E3585	106AA070
19A4C116	1E376C08	2748774C	34B0BCB5	391C0CB3	4ED8AA4A	5B9CCA4F	682E6FF3
748F82EE	78A5636F	84C87814	8CC70208	90BEFFFA	A4506CEB	BEF9A3F7	C67178F2

d. Menghitung nilai hash perantara untuk masing-masing blok pesan.

Menjumlahkan hasil akhir a, b, c, d, e, f, g dengan inisial *hash value* $H^{(i)}$.

$H_0^{(i)} = a + H_0^{(i)}$

$H_1^{(i)} = b + H_1^{(i)}$

$H_2^{(i)} = c + H_2^{(i)}$

$H_3^{(i)} = d + H_3^{(i)}$

$H_4^{(i)} = e + H_4^{(i)}$

$H_5^{(i)} = f + H_5^{(i)}$

$H_6^{(i)} = g + H_6^{(i)}$

$H_7^{(i)} = h + H_7^{(i)}$

5. Output

Setelah mengulangi langkah (4) sebanyak N kali, fungsi *hash* yang dihasilkan adalah sebagai berikut:

$H_0^{(N)}\ H_1^{(N)}\ H_2^{(N)}\ H_3^{(N)}\ H_4^{(N)}\ H_5^{(N)}\ H_6^{(N)}\ H_7^{(N)}$

Output diperoleh setelah semua blok M(N) 512 bit diproses. Setelah semua langkah pemprosesan dilakukan sejumlah N kali, maka akan didapat 256 bit message digest.

2.7 Aplikasi *Chatting*

Chatting merupakan pesan yang dikirim secara langsung melalui internet tanpa harus menunggu terlebih dahulu dimana pesan itu dikirim melalui sebuah media pembicaraan melalui tulisan. *Chatting* bukan hanya dikenal oleh anak muda saja namun saat ini, sudah mulai merambah ke orang dewasa maupun orang tua juga sudah masuk kedalam ruang lingkup kerja [29].

Aplikasi *chatting* merupakan media yang digunakan untuk menghubungkan dua orang atau lebih melalui *platform* media sosial. Berdasarkan hasil laporan *survey* Asosiasi Penyedia Jaringan Internet Indonesia (APJII) dengan 8.510 jiwa responden yang tersebar di berbagai provinsi yang ada di Indonesia diperoleh 5 aplikasi *chatting* yang sering digunakan sebagai media komunikasi, diantaranya yaitu aplikasi WhatsApp, Facebook, Telegram, Instagram dan Line [30].

2.8 Eavesdropping

Penyadapan sinyal (*Eavesdropping*) adalah tindakan mengintersep atau "menguping" data yang sedang ditransmisikan melalui jaringan. *Eavesdropping* merupakan jenis serangan pasif di mana penyerang berusaha memperoleh informasi dengan cara menyadap lalu lintas komunikasi tanpa mengubah isi pesan atau mengungkapkan kehadirannya. Pada jaringan publik, serangan ini dapat dilakukan dengan menangkap paket data yang ditransmisikan antara pengirim dan penerima menggunakan berbagai teknik *network sniffing*. Penyadapan Sinyal (*Eavesdropping*), yang sering kali diwujudkan melalui serangan *Packet Sniffing*, mereka dapat mencuri kata sandi, email, atau percakapan. Cara kerjanya memanfaatkan jaringan yang tidak terenkripsi untuk secara diam-diam menangkap paket data yang lewat, memungkinkan pelaku mencuri informasi sensitif seperti *username* dan *password* [31].

Dalam konteks komunikasi terenkripsi, sistem dinilai berhasil mencegah *eavesdropping* apabila data yang berhasil disadap hanya berupa *ciphertext* yang tidak dapat didekripsi tanpa memiliki kunci sesi yang sah. Serangan *eavesdropping* menjadi semakin relevan dalam era komunikasi berbasis internet karena banyaknya penggunaan jaringan publik dan Wi-Fi terbuka. Weichbroth dan Łysik [2]. menegaskan bahwa komunikasi yang tidak dilindungi oleh enkripsi yang kuat sangat rentan terhadap penyadapan, bahkan dengan peralatan sederhana. Oleh sebab itu, implementasi kriptografi modern seperti E2EE yang didukung oleh mekanisme pertukaran kunci aman menjadi kebutuhan mendasar dalam sistem komunikasi digital saat ini.

2.8.1 Model Serangan *Eavesdropping*

Pada model serangan *eavesdropping*, penyerang diasumsikan memiliki kemampuan untuk mengamati seluruh lalu lintas jaringan, baik pada jaringan publik maupun infrastruktur perantara seperti router dan server, namun tidak memiliki akses terhadap kunci kriptografi yang digunakan. Penyerang tidak memodifikasi paket data sehingga fokus utama serangan adalah upaya memperoleh informasi bermakna dari *ciphertext* yang berhasil disadap, baik melalui analisis lalu lintas maupun upaya pemecahan kriptografi.

Dalam literatur keamanan jaringan, *eavesdropping* dikategorikan sebagai serangan pasif karena penyerang tidak melakukan modifikasi terhadap data yang ditransmisikan, melainkan hanya mengamati dan mengumpulkan informasi. Serangan ini sering dilakukan menggunakan teknik packet sniffing pada jaringan publik atau jaringan Wi-Fi yang tidak terenkripsi [2].

Karena sifatnya yang tidak terdeteksi, *eavesdropping* menjadi ancaman serius bagi komunikasi digital. Oleh sebab itu, sistem keamanan yang dirancang dalam penelitian ini difokuskan pada perlindungan kerahasiaan pesan sehingga data yang berhasil disadap tidak dapat diinterpretasikan tanpa kunci sesi yang sah.

2.8.2 Pencegahan Eavesdropping dengan Kriptografi Hibrid

Pencegahan *eavesdropping* dapat dilakukan secara efektif dengan mengombinasikan kriptografi hibrid, yaitu ECDHE untuk pembentukan kunci sesi dan ChaCha20 untuk enkripsi pesan. Dengan desain ini, setiap sesi komunikasi memiliki kunci yang unik dan bersifat sementara yang tidak dapat digunakan untuk sesi komunikasi lainnya.

Akibatnya, meskipun penyerang berhasil menangkap dan menyimpan paket data komunikasi, isi pesan tetap tidak dapat dibaca tanpa kunci sesi yang telah dihapus setelah sesi berakhir, dan kunci tersebut tidak dapat digunakan untuk membuka sesi komunikasi lainnya. Penerapan End-to-End Encryption juga memastikan bahwa server aplikasi tidak memiliki akses terhadap kunci enkripsi, sehingga tidak dapat dijadikan titik lemah untuk memperoleh isi komunikasi pengguna bahkan jika server berhasil dikompromikan. Kombinasi ketiga mekanisme ini—ECDHE, ChaCha20, dan E2EE—menciptakan lapisan pertahanan berlapis yang memperkuat keamanan sistem terhadap serangan *eavesdropping*.

Dalam pendekatan keamanan berlapis (*layered security*), kombinasi mekanisme pertukaran kunci yang aman dan algoritma enkripsi yang kuat menjadi fondasi utama perlindungan komunikasi digital [25]. Dengan demikian, integrasi ECDHE dan ChaCha20

pada penelitian ini tidak hanya bertujuan mengenkripsi pesan, tetapi juga membangun model pertahanan sistem yang komprehensif terhadap ancaman penyadapan.

2.9 Model Keamanan Sistem

Model keamanan sistem pada penelitian ini dirancang untuk melindungi kerahasiaan data komunikasi antara pengguna dalam lingkungan jaringan publik yang berpotensi tidak aman. Sistem keamanan yang diusulkan mengadopsi mekanisme *end-to-end encryption* (E2EE) dengan memanfaatkan kombinasi algoritma kriptografi modern, yaitu ChaCha20 sebagai algoritma enkripsi simetris dan *Elliptic Curve Diffie-Hellman Ephemeral* (ECDHE) sebagai mekanisme pertukaran kunci.

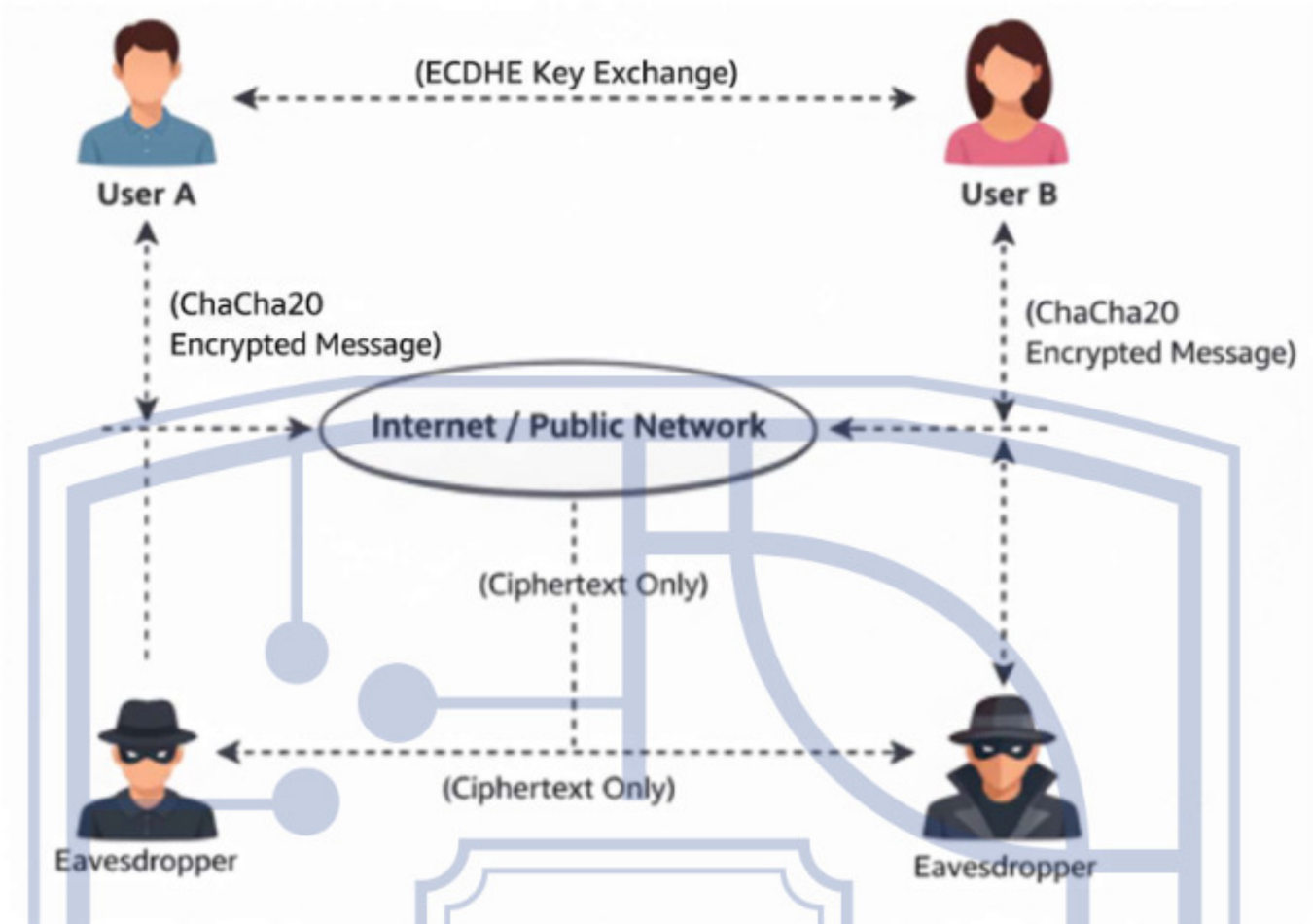
Dalam penelitian ini, model keamanan sistem dibangun dengan asumsi bahwa penyerang memiliki kemampuan untuk mengamati dan menyadap lalu lintas komunikasi pada jaringan publik maupun infrastruktur perantara, termasuk server aplikasi. Namun demikian, penyerang diasumsikan tidak memiliki akses terhadap kunci kriptografi yang disimpan pada perangkat pengguna dan tidak melakukan modifikasi aktif terhadap isi pesan. Oleh karena itu, ancaman yang dibahas dalam penelitian ini difokuskan pada serangan pasif berupa *eavesdropping*.

Untuk mengatasi ancaman tersebut, proses komunikasi diawali dengan mekanisme pertukaran kunci menggunakan ECDHE. Melalui mekanisme ini, dua pihak yang berkomunikasi menghasilkan kunci sesi secara dinamis tanpa pernah mengirimkan kunci tersebut secara langsung melalui jaringan. Penggunaan sifat *ephemeral* pada ECDHE memastikan bahwa setiap sesi komunikasi memiliki kunci yang berbeda, sehingga memberikan perlindungan *forward secrecy* apabila kunci jangka panjang sistem berhasil dikompromikan.

Setelah kunci sesi terbentuk, algoritma ChaCha20 digunakan untuk mengenkripsi pesan yang dikirimkan antar pengguna. ChaCha20 dipilih karena memiliki performa yang efisien pada berbagai perangkat dan tidak bergantung pada akselerasi perangkat keras tertentu. Seluruh data yang dikirimkan melalui jaringan berada dalam bentuk *ciphertext*, sehingga pihak ketiga, termasuk server perantara, tidak dapat mengetahui isi pesan yang dikomunikasikan.

Dengan model keamanan sistem ini, kerahasiaan data komunikasi dapat tetap terjaga meskipun terjadi penyadapan jaringan. Kombinasi antara ECDHE dan ChaCha20 memastikan bahwa sistem mampu memberikan perlindungan terhadap ancaman pasif yang

diasumsikan dalam penelitian ini, sekaligus menjaga keamanan kunci dan pesan selama proses komunikasi berlangsung.



Gambar 2.7 Model Keamanan Komunikasi End-to-End Menggunakan ChaCha20 dan ECDHE

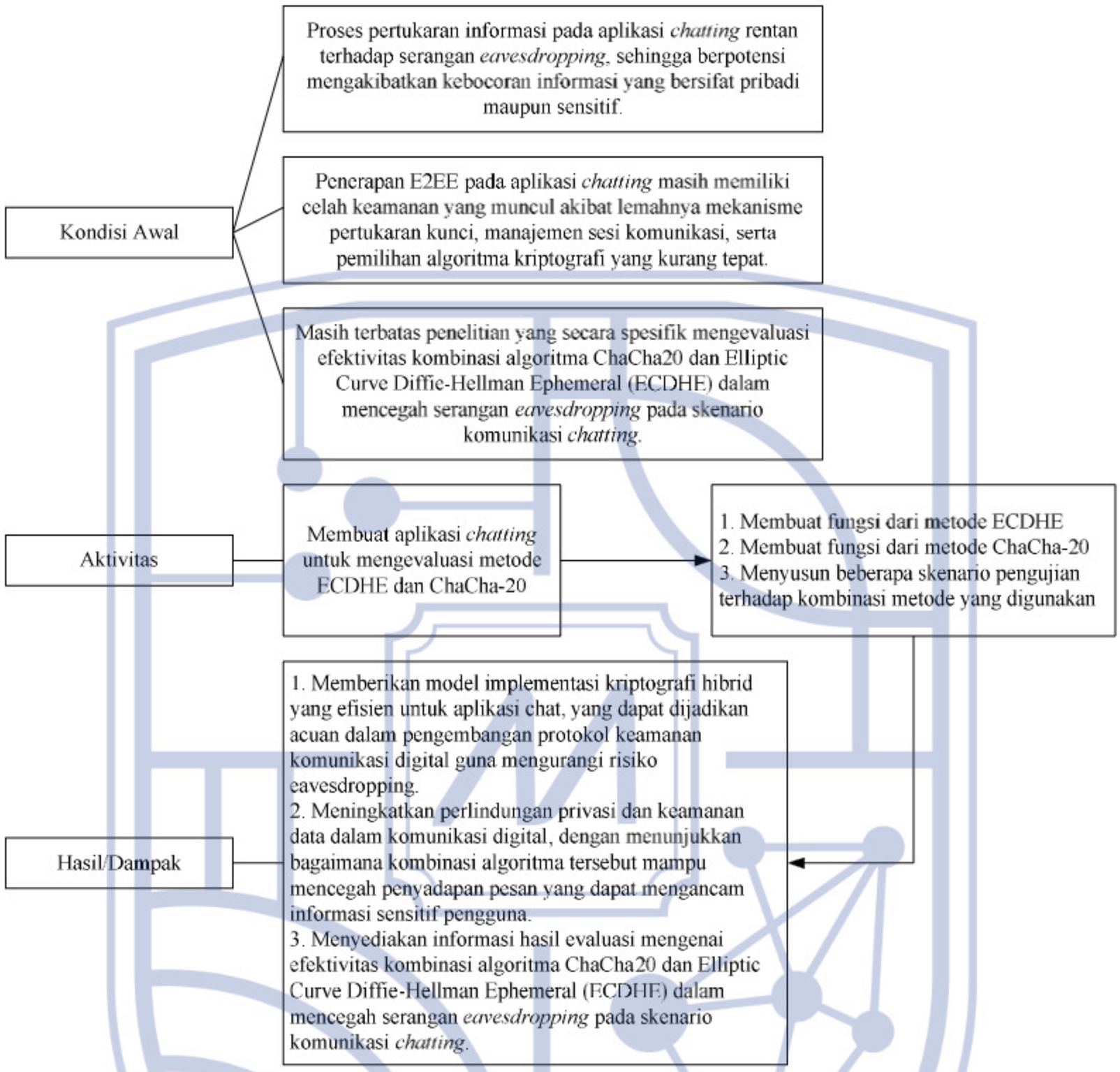
Gambar 2.7 menunjukkan model keamanan komunikasi end-to-end pada aplikasi chatting berbasis web yang menggunakan kombinasi algoritma ChaCha20 dan Elliptic Curve Diffie-Hellman Ephemeral (ECDHE). Proses komunikasi diawali dengan pertukaran kunci menggunakan ECDHE antara dua pengguna untuk menghasilkan kunci sesi yang bersifat sementara (*ephemeral*). Kunci sesi ini tidak pernah dikirimkan secara langsung melalui jaringan publik, melainkan dihitung secara independen oleh masing-masing pihak yang berkomunikasi.

Setelah kunci sesi terbentuk, algoritma ChaCha20 digunakan untuk mengenkripsi pesan sebelum ditransmisikan melalui jaringan publik. Seluruh data yang melewati jaringan berada dalam bentuk *ciphertext*, sehingga pihak ketiga yang melakukan penyadapan hanya dapat menangkap data terenkripsi tanpa memiliki kemampuan untuk mendekripsi isi pesan. Dengan mekanisme ini, meskipun terjadi serangan eavesdropping, kerahasiaan komunikasi tetap terjaga karena penyerang tidak memiliki akses terhadap kunci sesi yang digunakan.

2.10 Kerangka Pemikiran

Kerangka kerja adalah suatu struktur konsep mendasar yang dapat digunakan untuk memecahkan suatu permasalahan yang rumit. Kerangka kerja berisikan urutan dari

pelaksanaan kerja dan identifikasi masalah dalam menyelesaikan penelitian ini. Rancangan kerangka pikir dari penelitian yang akan dilakukan ini dapat digambarkan sebagai berikut:



Gambar 2.8 Kerangka Penelitian

2.11 Penelitian Terdahulu

Beberapa penelitian terdahulu yang pernah dilakukan sebelumnya dapat dirincikan sebagai berikut:

Tabel 2.5 Penelitian Terdahulu

No	Nama Pengarang (Tahun)	Judul	Keterangan
1	Dandi Darmansyah dan Abdul Halim Hasugian (2025)	Enkripsi Pesan Chat Menggunakan Algoritma Chacha20 pada Aplikasi	Penelitian ini bertujuan untuk mengimplementasikan algoritma ChaCha20 dalam mengenkripsi pesan teks pada aplikasi komunikasi

		Komunikasi Real-Time [32]	berbasis Android menggunakan framework Flutter dan layanan *backend* Firebase. Hasil pengujian menunjukkan adanya perbedaan performa signifikan dibanding algoritma AES-CBC, di mana ChaCha20 memiliki waktu enkripsi dan dekripsi yang lebih cepat serta *throughput* yang lebih tinggi. Berdasarkan temuan ini, disarankan penggunaan ChaCha20 pada aplikasi yang memerlukan kecepatan dan efisiensi tinggi dalam pertukaran data. Dampak penelitian ini diharapkan dapat meningkatkan kesadaran dan pemahaman terhadap pentingnya pemilihan algoritma kriptografi yang tepat dalam membangun sistem komunikasi aman. Selain itu, hasil penelitian ini diharapkan memberikan kontribusi nyata dalam pengembangan aplikasi komunikasi yang tangguh terhadap ancaman keamanan siber, khususnya pada perangkat *mobile*.
2	Anggi Susanti, Bayu Ade Prasetiya, Oktafiyan Dyah Pangesti, Liling Daru Suryawati dan Indrawan Ady Saputro (2024)	Perbandingan Kinerja dan Keamanan Algoritma Kriptografi Modern AES-GCM dengan Chacha20-Poly1305 [33]	AES-GCM dan ChaCha20-Poly1305 adalah dua algoritma populer yang digunakan, namun keduanya memiliki trade-off antara kinerja dan keamanan. Penelitian ini bertujuan mengevaluasi kedua algoritma melalui pengujian kinerja dan simulasi serangan timing.

			Pengujian kinerja dilakukan dengan mengukur waktu enkripsi dan dekripsi pada berbagai ukuran data, sedangkan simulasi serangan timing menilai kerentanan terhadap serangan tersebut. Hasil penelitian menunjukkan bahwa AES-GCM lebih lambat dibandingkan ChaCha20-Poly1305, terutama pada data berukuran besar, tetapi menawarkan ketahanan lebih baik terhadap serangan timing. Sebaliknya, ChaCha20-Poly1305 unggul dalam kecepatan, menjadikannya pilihan ideal untuk aplikasi dengan kebutuhan performa tinggi tanpa mengorbankan keamanan dasar.