BAB II
KAJIAN LITERATUR

2.1 Tinjauan Pustaka

Pada bagian ini berisikan landasan teori dan akan dijelaskan tinjauan pustaka yang berkaitan dengan penelitian yang dilakukan.

2.1.1 *Diffie-Hellman* (DH) *Key Exchange*

Diffie-Hellman (DH) adalah metode pertukaran kunci eksponensial yang memungkinkan dua pihak yang tidak saling mengenal untuk bertukar *secret key* secara aman, meskipun melalui saluran komunikasi yang rentan terhadap pengawasan pihak ketiga [9,10]. Diperkenalkan oleh Whitfield Diffie dan Martin Hellman pada tahun 1976, DH menjadi dasar dari banyak protokol pertukaran kunci modern. Protokol ini bergantung pada kesulitan masalah logaritma diskret dalam grup bilangan prima untuk menjaga kerahasiaan kunci [23]. Meskipun melindungi pertukaran data dari serangan pasif, DH tidak menyediakan autentikasi dua arah [11], sehingga rentan terhadap serangan *Man-in-the-Middle* (MITM) di mana penyerang dapat menyisipkan kunci sendiri dalam komunikasi [10]. Untuk mencapai tingkat keamanan yang sebanding dengan standar saat ini, ukuran kunci DH harus diperbesar secara substansial (misalnya minimal 2048-bit), yang berdampak pada peningkatan *overhead* komputasi, waktu proses *handshake* yang lebih lama, serta konsumsi *resource* perangkat yang lebih tinggi [14,15]. Selain itu, dalam beberapa skenario, ada kemungkinan kunci sesi sebelumnya dapat terkompromikan di masa depan jika kunci privat salah satu pihak bocor [17].

Tahapan algoritma DH secara umum dapat dijelaskan sebagai berikut [23]:

1. Dua pihak menyepakati bilangan prima publik (p) dan basis (g).
2. Masing-masing pihak memilih bilangan rahasia a dan b secara acak.
3. Kedua pihak menghitung nilai publik masing-masing:
   a. Pihak A menghitung persamaan (1).
      $A = g^a \bmod p$ .....(1)
   b. Pihak B menghitung persamaan (2).
      $B = g^b \bmod p$ .....(2)
4. Mereka bertukar nilai publik A dan B.
5. Masing-masing pihak menghitung kunci bersama:

a. Pihak A menghitung persamaan (3).

$s = B^a \ mod \ p$ .....(3)

b. Pihak B menghitung persamaan (4).

$s = A^b \ mod \ p$.....(4)

Kunci bersama (s) yang dihasilkan akan identik pada kedua pihak dan dapat digunakan sebagai dasar untuk menghasilkan kunci simetris yang digunakan untuk enkripsi komunikasi [23]. Meskipun sederhana dan elegan, DH murni tidak aman tanpa adanya autentikasi tambahan. Oleh karena itu, berbagai pengembangan seperti *Elliptic Curve Diffie-Hellman* (ECDHE) [17] dan PrivateDH [12] dikembangkan untuk mengatasi kelemahan ini, dengan menerapkan autentikasi tambahan atau struktur kriptografi yang lebih kuat untuk meningkatkan keamanan komunikasi.

2.1.2 *Elliptic Curve Diffie-Hellman Ephemeral* (ECDHE) Curve25519

Elliptic Curve Diffie-Hellman Ephemeral (ECDHE) Curve25519 adalah protokol pertukaran kunci berbasis kriptografi kurva eliptik yang dirancang untuk menyediakan keamanan tingkat tinggi dengan efisiensi komputasi yang lebih baik dibandingkan dengan *Diffie-Hellman* (DH) klasik [24,25]. ECDHE merupakan varian dari *Diffie-Hellman* yang menggunakan pasangan kunci baru untuk setiap sesi komunikasi (*ephemeral*), sehingga mendukung fitur *forward secrecy*, di mana kompromi satu kunci sesi tidak membahayakan sesi komunikasi lainnya [24]. Dengan keunggulan ini, ECDHE menjadi salah satu metode pertukaran kunci yang banyak diadopsi dalam protokol keamanan modern.

Curve25519 adalah salah satu kurva eliptik yang dirancang secara khusus untuk keamanan, kecepatan, dan kemudahan implementasi. Kurva ini diperkenalkan oleh Daniel J. Bernstein pada tahun 2005 dan telah menjadi standar *de facto* dalam banyak sistem keamanan modern karena ketahanannya terhadap serangan kriptografi umum dan performa optimumnya [24]. Kurva ini bekerja pada bidang hingga $F_{2^{255}-19}$ dan dioptimalkan untuk operasi *scalar multiplication* yang cepat serta tahan terhadap berbagai serangan implementasi seperti *timing attacks* dan *invalid curve attacks* [24]. Curve25519 menghindari banyak kelemahan implementasi yang ditemukan pada kurva lain dan menawarkan efisiensi tinggi bahkan pada perangkat dengan sumber daya terbatas. Dengan menggunakan kunci sepanjang 256-bit, Curve25519 dapat menyediakan keamanan setara atau lebih baik dibandingkan DH klasik dengan kunci 2048-bit [16], sekaligus secara signifikan mengurangi *overhead* komputasi dan mempercepat proses pertukaran kunci.

Dalam ECDHE Curve25519, setiap pihak dalam komunikasi menghasilkan pasangan kunci privat dan publik baru untuk setiap sesi. Mereka bertukar kunci publik dan menghitung *shared secret* dengan menggunakan operasi pada kurva eliptik. Keamanan algoritma ini bergantung pada kesulitan masalah *Diffie-Hellman* pada kurva eliptik (*Elliptic Curve Diffie-Hellman Problem*, ECDHP), yang secara signifikan lebih sulit dipecahkan dibandingkan dengan DH klasik untuk ukuran kunci yang sama [25]. Dengan demikian, ECDHE Curve25519 menyediakan perlindungan yang kuat terhadap serangan kriptografi konvensional.

Tahapan algoritma ECDHE Curve25519 adalah sebagai berikut:

1. Parameter awal:
   a. Kurva yang digunakan adalah Curve25519 dengan bidang hingga F_p, $p = 2^{255} - 19$.
   b. *Base point* standar memiliki *x-coordinate* = 9.
2. Pembuatan kunci privat:
   a. Setiap pihak menghasilkan bilangan acak sepanjang 32 byte.
   b. Proses *clamping* diterapkan:
      i. Bit 0 dan 1 dari *byte* pertama diatur ke 0.
      ii. Bit 254 diatur ke 1.
      iii. Bit 255 diatur ke 0.
3. Perhitungan kunci publik:
   a. Kunci publik dihitung dengan melakukan operasi *scalar multiplication* antara kunci privat dan *base point*.
   b. Hasil operasi berupa koordinat x yang digunakan sebagai kunci publik.
4. Pertukaran kunci publik:
   Masing-masing pihak bertukar kunci publik yang telah dihitung.
5. Validasi kunci publik:
   Pihak penerima memastikan kunci publik yang diterima valid dan berada dalam domain kurva.
6. Perhitungan *shared secret*:
   a. Setiap pihak melakukan *scalar multiplication* antara kunci privatnya dan kunci publik pihak lawan.
   b. Hasil operasi ini menghasilkan *shared secret* berupa *x-coordinate*.

7. Pengolahan *shared secret*:
   Shared secret dapat digunakan secara langsung atau diproses lebih lanjut menggunakan *Key Derivation Function* (KDF) seperti HKDF dengan *input shared secret* dan *context information*.
8. Penggunaan kunci sesi:
   Kunci sesi yang dihasilkan digunakan untuk mengenkripsi dan mengamankan komunikasi *end-to-end* menggunakan algoritma enkripsi simetris seperti AES.

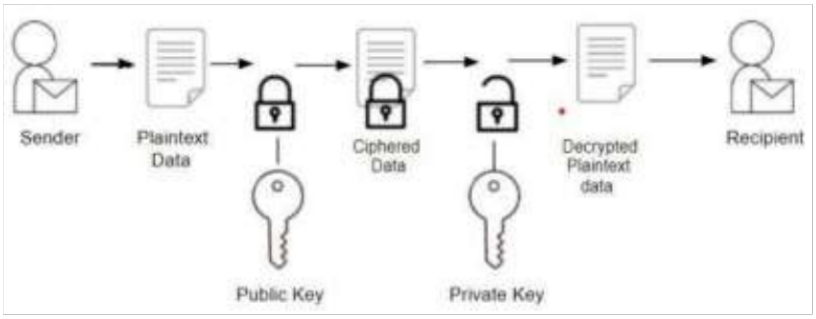
Keunggulan utama ECDHE Curve25519 dibandingkan DH klasik meliputi:

- Tingkat keamanan yang lebih tinggi untuk ukuran kunci yang lebih kecil.
- Efisiensi komputasi yang lebih baik, mengurangi *overhead* dalam proses pertukaran kunci.
- Dukungan *inherent* terhadap *forward secrecy*.
- Tahan terhadap serangan *side-channel* tertentu karena desain kurva yang aman.

Dalam konteks penelitian ini, ECDHE Curve25519 diintegrasikan untuk menggantikan penggunaan DH klasik dalam protokol PrivateDH, dengan tujuan untuk meningkatkan keamanan pertukaran kunci dan mengurangi *overhead* komputasi, sekaligus memperkuat keseluruhan skema komunikasi *end-to-end*.

2.1.3 *Rivest-Shamir-Adleman* (RSA)

Rivest-Shamir-Adleman (RSA) ditemukan oleh Ron Rivest, Adi Shamir, dan Leonard Adleman pada tahun 1977. RSA merupakan kriptografi asimetris pertama yang berhasil diterapkan secara praktis dan kini menjadi yang paling umum digunakan di seluruh dunia, melindungi berbagai aspek keamanan, mulai dari komunikasi melalui ponsel hingga transaksi perbankan online [26]. RSA banyak digunakan dalam keamanan data digital, terutama dalam enkripsi dan *digital signature*. RSA mengandalkan kesulitan faktorisasi sebuah bilangan bulat besar menjadi dua bilangan prima yang besar sebagai dasar keamanannya [27,26]. Algoritma ini mengacu pada penggunaan dua kunci yang berbeda, yakni kunci publik dan kunci privat. Sesuai dengan namanya, kunci publik dapat dibagikan kepada siapa saja yang ingin mengirim pesan secara aman kepada penerima, sementara kunci privat harus dijaga kerahasiaannya oleh penerima [9,26]. Algoritma enkripsi RSA dapat dilihat pada Gambar 2.1 berikut.



Gambar 2.1 Algoritma Enkripsi RSA [9]

Berikut adalah tahapan dalam proses enkripsi RSA:

1. Pembuatan kunci
   a. Pilih dua bilangan prima besar untuk p dan q.
   b. Hitung nilai modulus (N) yang dapat dilihat pada persamaan (5).

      $N = p \times q$ .....(5)

   c. Hitung *Euler's Totient Function* (φ(N)) yang dapat dilihat pada persamaan (6).

      $\varphi(N) = (p-1)(q-1)$ .....(6)

   d. Pilih bilangan bulat eksponen enkripsi (e), yang harus memenuhi persamaan (7) dan e relatif prima terhadap φ(N), yaitu dapat dilihat pada persamaan (8).

      $1 < e < \varphi(N)$ .....(7)

      $\gcd\big(e, \varphi(N)\big) = 1$ .....(8)

   e. Hitung eksponen dekripsi (d), yang harus memenuhi persamaan (9).

      $e \times d \equiv 1\big(mod\ \varphi(N)\big)$ .....(9)

   Hasil akhir pembuatan kunci adalah kunci publik (N, e) yang dikirimkan kepada penerima dan kunci privat (φ(N), d) dijaga kerahasiaannya.
2. Enkripsi

   Pengirim mengenkripsi *plaintext* m menjadi c menggunakan kunci publik penerima, yang dapat dilihat pada persamaan (10).

   $m^e \equiv c(mod\ N)$ .....(10)
3. Dekripsi

   Penerima mendekripsi c menjadi m menggunakan kunci privat d, yang dapat dilihat pada persamaan (11).

   $c^d \equiv (m^e)^d \equiv m^{ed} \equiv m(mod\ N)$ .....(11)

Fungsi RSA m → m^e disebut sebagai *trapdoor one-way function*, yaitu fungsi satu arah dengan pintu jebakan, yang sangat mudah dihitung namun tidak dapat dibalik dalam waktu polinomial tanpa mengetahui pintu jebakannya, yaitu kunci rahasia d. Untuk

mengetahui d, penyerang harus memfaktorkan n menjadi dua bilangan prima besar, yang dikenal sebagai "masalah faktorisasi" (*factoring problem*) [26].

2.1.4 *RSA Signature Scheme with Appendix - Probabilistic Signature Scheme* (RSASSA-PSS)

RSA Signature Scheme with Appendix - Probabilistic Signature Scheme (RSASSA-PSS) adalah skema tanda tangan digital berbasis probabilistik yang dirancang untuk meningkatkan keamanan RSA klasik. Skema ini menggunakan teknik padding acak (*probabilistic padding*) untuk mencegah serangan yang mengeksploitasi struktur deterministik dari pesan terenkripsi [19]. Dalam konteks tanda tangan digital, padding digunakan untuk menambahkan elemen acak pada pesan sebelum dilakukan proses tanda tangan. Tujuannya adalah untuk mencegah pihak musuh mengenali pola atau struktur dalam pesan yang dapat dimanfaatkan dalam serangan seperti *chosen plaintext attack* atau *adaptive chosen ciphertext attack* [26].

Tidak seperti skema tanda tangan deterministik seperti PKCS#1 v1.5, RSASSA-PSS menghasilkan tanda tangan yang berbeda meskipun isi pesan sama, sehingga meningkatkan keamanan terhadap serangan *replay* dan *padding oracle*. Standar ini diadopsi secara luas dalam berbagai protokol kriptografi modern dan direkomendasikan dalam RFC 8017 (PKCS #1 v2.2) karena mendukung keamanan semantik dan ketahanan terhadap *adaptive chosen-message attacks*. Dalam konteks penelitian ini, RSASSA-PSS digunakan untuk memberikan autentikasi digital yang kuat dan membedakan identitas pengirim secara kriptografis, terutama ketika *ciphertext* identik diterima dari sumber berbeda [19,20].

Tahapan algoritma RSASSA-PSS terdiri dari dua proses utama, yaitu:

1. Proses Penandatanganan:
    a. *Hashing*: Pesan asli di-*hash* menggunakan fungsi *hash* kriptografis (misal: SHA-256).
    b. *Salt*: Nilai acak (*salt*) dihasilkan dan ditambahkan.
    c. *EMSA-PSS Encoding*: *Hash* dan *salt* dikombinasikan dan dienkode menggunakan skema EMSA-PSS untuk menghasilkan blok pesan terstruktur.
    d. *Signing*: Blok tersebut dienkripsi menggunakan kunci privat RSA untuk menghasilkan *signature*.
2. Proses Verifikasi:
    a. *Decryption*: *Signature* didekripsi menggunakan kunci publik RSA untuk mendapatkan blok terstruktur.

b. *EMSA-PSS Decoding*: Blok didekode dan nilai *salt* serta *hash* dipisahkan.
c. *Hash Matching*: *Hash* dari pesan diverifikasi ulang terhadap hasil *decoding* untuk memastikan keaslian.

Skema ini mengacu pada spesifikasi resmi dalam RFC 8017 (PKCS #1 v2.2) yang merekomendasikan RSASSA-PSS sebagai standar untuk tanda tangan digital modern karena kemampuannya menangkal serangan *padding oracle* dan *chosen-message* [28].

2.1.5 *Advanced Encryption Standard* (AES)

Advanced Encryption Standard (AES) adalah algoritma kriptografi simetris modern yang dirancang untuk menggantikan *Data Encryption Standard* (DES), dengan tingkat keamanan dan efisiensi komputasi yang lebih tinggi. AES dirancang oleh Joan Daemen dan Vincent Rijmen, dan diadopsi sebagai standar oleh NIST pada tahun 2001 dalam dokumen FIPS PUB 197 [29]. AES menjadi algoritma standar dalam banyak sistem keamanan karena kecepatan, efisiensi, dan ketahanannya terhadap berbagai serangan kriptografi [30]. AES bekerja pada blok data 128-bit dengan panjang kunci 128, 192, atau 256 bit. Proses enkripsi AES terdiri dari sejumlah putaran transformasi (10, 12, atau 14 putaran tergantung panjang kunci), yang mencakup empat tahap utama: *SubBytes*, *ShiftRows*, *MixColumns*, dan *AddRoundKey*. Transformasi ini memastikan difusi dan kompleksitas tinggi pada data terenkripsi. Proses tersebut telah dijelaskan secara teknis dalam buku "*Understanding Cryptography*" oleh Christof Paar dan Jan Pelzl, yang mendeskripsikan bagaimana kombinasi dari operasi substitusi dan permutasi membuat AES sangat tahan terhadap analisis statistik dan serangan diferensial [30].

Dalam penelitian ini, AES-256 digunakan sebagai komponen enkripsi simetris utama yang berperan dalam proses pertukaran kunci dan juga untuk mengenkripsi isi pesan selama komunikasi *end-to-end* secara berkelanjutan. AES-256 digunakan karena menawarkan tingkat keamanan tertinggi di antara varian lainnya. AES-256 tahan terhadap serangan *brute-force* dan menghasilkan efek *avalanche* yang signifikan di mana perubahan kecil pada input dapat menghasilkan perubahan besar pada *output*. AES-256 juga memberikan margin keamanan lebih luas dibandingkan AES-128, sehingga sering direkomendasikan untuk skenario dengan kebutuhan keamanan tinggi seperti komunikasi *end-to-end* [21,29]. AES-256 sebagaimana digunakan dalam penelitian ini mengacu pada spesifikasi resmi dalam FIPS PUB 197 sebagai standar enkripsi simetris modern yang banyak diimplementasikan dalam sistem keamanan informasi [29].

Tahapan algoritma AES adalah sebagai berikut:

1. Proses Enkripsi:
    a. *AddRoundKey*: Kunci awal digabungkan dengan blok plainteks menggunakan operasi XOR.
    b. Putaran utama: Dilakukan 9, 11, atau 13 kali tergantung panjang kunci:
        i. *SubBytes*: Operasi substitusi *non-linear* menggunakan *S-Box*.
        ii. *ShiftRows*: Pergeseran posisi baris.
        iii. *MixColumns*: Operasi difusi kolom dengan transformasi matriks.
        iv. *AddRoundKey*: XOR antara hasil sementara dan kunci putaran.
        v. Putaran akhir (tanpa *MixColumns*): *SubBytes* → *ShiftRows* → *AddRoundKey*.
2. Proses Dekripsi:
    a. Lakukan operasi *inverse* dari setiap tahapan di atas secara berurutan.
    b. Gunakan *inverse S-Box*, *InvShiftRows*, *InvMixColumns*, dan *AddRoundKey* untuk mendapatkan plainteks kembali.

Tahapan ini mengacu pada standar FIPS PUB 197 dan penjabaran dalam literatur kriptografi modern [29,30].

2.1.6 PrivateDH

PrivateDH (*Private Diffie-Hellman*) adalah protokol pertukaran kunci yang dikembangkan untuk meningkatkan keamanan komunikasi *end-to-end* dengan mengintegrasikan algoritma RSA dan AES di atas mekanisme dasar *Diffie-Hellman* (DH) [12]. PrivateDH dirancang untuk memberikan kombinasi keamanan dari enkripsi asimetris (RSA) dan enkripsi simetris (AES) dalam satu kerangka kerja protokol, sehingga mampu mengamankan pertukaran kunci rahasia antara dua pihak yang berkomunikasi. Pada prinsip dasarnya, PrivateDH memanfaatkan RSA untuk mengenkripsi kunci simetris yang dihasilkan, sementara AES digunakan untuk mengenkripsi isi pesan. Selain itu, seluruh data yang perlu dibagikan selama pertukaran kunci, termasuk parameter *Diffie-Hellman*, dienkripsi menggunakan algoritma AES untuk mencegah eksposur terhadap pihak ketiga. *Diffie-Hellman* tetap digunakan sebagai dasar untuk menghasilkan *secret key* bersama antara dua pihak. Dengan demikian, PrivateDH berusaha menggabungkan keunggulan dari berbagai teknik kriptografi untuk mencapai tingkat keamanan komunikasi yang lebih tinggi.

Tahapan Algoritma PrivateDH adalah sebagai berikut:

1. Pihak A dan pihak B bertukar kunci publik RSA melalui pihak ketiga terpercaya.
2. Pihak A memilih *initial secret* acak dan mengenkripsinya menggunakan kunci publik RSA pihak B.

3. Pihak A mengirimkan *initial secret* yang telah dienkripsi ke pihak B.
4. Pihak B mendekripsi *initial secret* menggunakan kunci privat RSA miliknya.
5. Berdasarkan *initial secret*, pihak B membangun kunci AES.
6. Pihak B menghitung parameter *Diffie-Hellman* (g^b mod p) dan mengenkripsinya menggunakan kunci AES.
7. Pihak B mengirimkan parameter *Diffie-Hellman* terenkripsi tersebut kepada pihak A.
8. Pihak A mendekripsi parameter *Diffie-Hellman* menggunakan kunci AES.
9. Pihak A dan pihak B masing-masing menghitung *shared secret key* berdasarkan nilai *Diffie-Hellman*.
10. *Shared secret key* ini kemudian digunakan untuk mengenkripsi dan mendekripsi pesan-pesan komunikasi menggunakan AES.

PrivateDH memiliki beberapa karakteristik ketahanan dalam desainnya. Jika kunci publik RSA yang digunakan dalam proses pertukaran kunci berhasil dikompromikan, *shared secret key* tetap sulit diakses oleh penyerang tanpa mengetahui nilai rahasia a atau b dari proses *Diffie-Hellman*. Demikian pula, apabila kunci simetris yang digunakan dalam komunikasi berhasil dikompromikan, protokol ini masih mempertahankan mekanisme keamanan dasar serupa dengan *Diffie-Hellman*, menjaga kerahasiaan sebagian besar sesi komunikasi. Karakteristik ini menunjukkan bahwa PrivateDH memiliki tingkat resiliensi tambahan dalam menghadapi serangan terhadap kunci kriptografi.

Namun, PrivateDH dalam implementasinya masih memiliki kelemahan yang signifikan. PrivateDH tidak memiliki mekanisme autentikasi yang kuat terhadap *ciphertext*, sehingga membuka kemungkinan serangan di mana penerima tidak dapat membedakan antara pengirim sah dan pihak penyerang yang mengirimkan pesan palsu. Meskipun PrivateDH memiliki mekanisme deteksi anomali melalui penerimaan dua *ciphertext* identik dari sumber berbeda, mekanisme ini bersifat reaktif dan tidak sepenuhnya mencegah kompromi yang mungkin terjadi [12]. Selain itu, adanya *overhead* komputasi yang muncul dalam proses enkripsi dan dekripsi, khususnya terkait dengan parameter DH. Penggunaan algoritma DH dalam PrivateDH membawa tantangan terkait kebutuhan ukuran kunci yang besar untuk mempertahankan tingkat keamanan modern, yang menyebabkan *overhead* komputasi lebih tinggi dibandingkan pendekatan kriptografi modern seperti ECDHE Curve25519 [13].

Dalam konteks penelitian ini, protokol PrivateDH menjadi fokus pengembangan lebih lanjut dengan mengintegrasikan pendekatan baru seperti penggunaan ECDHE Curve25519 untuk menggantikan DH, dan penerapan RSASSA-PSS untuk autentikasi

digital. Upaya ini bertujuan untuk mengatasi kelemahan PrivateDH yang telah diidentifikasi dan meningkatkan keamanan serta efisiensi dalam komunikasi *end-to-end*.

2.1.7 *Man-in-the-Middle* (MITM) *Attack*

Man-in-the-Middle (MITM) adalah jenis serangan di mana penyerang diam-diam menyisipkan dirinya di antara dua pihak yang berkomunikasi dan berusaha untuk menguping, menyadap, atau bahkan memodifikasi data yang ditransmisikan tanpa sepengetahuan kedua pihak tersebut [4]. Serangan MITM biasanya memanfaatkan ketidakamanan dalam pertukaran kunci atau kurangnya mekanisme autentikasi untuk mendapatkan akses terhadap komunikasi yang seharusnya bersifat rahasia [4]. Dalam konteks protokol pertukaran kunci seperti *Diffie-Hellman* dan PrivateDH, ketidakhadiran autentikasi yang kuat membuka celah bagi penyerang untuk menyisipkan kunci palsu, sehingga kedua pihak berkomunikasi secara tidak sadar melalui saluran yang dikendalikan penyerang.

Serangan MITM sangat berbahaya dalam skenario komunikasi *end-to-end*, karena dapat merusak integritas, kerahasiaan, dan keaslian komunikasi [4]. Penelitian terbaru juga menunjukkan bahwa teknik MITM tetap menjadi metode efektif dalam kriptanalisis terhadap algoritma modern seperti fungsi *hash Streebog*, yang membuktikan bahwa serangan jenis ini terus berkembang dan memerlukan mitigasi yang kuat [31]. Oleh karena itu, mekanisme autentikasi digital, seperti penggunaan tanda tangan digital (misalnya RSASSA-PSS), menjadi sangat penting untuk memastikan bahwa pihak yang berkomunikasi benar-benar sah.

2.1.8 *End-to-End Encryption* (E2EE)

End-to-End Encryption (E2EE) adalah metode komunikasi di mana hanya pihak yang berkomunikasi yang dapat membaca pesan yang dikirim. Dalam E2EE, data dienkripsi di sisi pengirim dan hanya dapat didekripsi oleh penerima yang dituju, sehingga mencegah pihak ketiga, termasuk penyedia layanan komunikasi, dari mengakses isi pesan [6]. Kebutuhan akan E2EE semakin meningkat seiring dengan maraknya penyadapan data oleh server layanan komunikasi itu sendiri, di mana server dapat mengakses pesan karena data didekripsi sementara sebelum dikirim ke penerima [6]. Implementasi E2EE mengatasi masalah ini dengan memastikan bahwa kunci enkripsi hanya dimiliki oleh pengguna akhir, sehingga bahkan server tidak dapat membaca pesan yang dikirimkan [6].

Konsep E2EE penting dalam melindungi data sensitif dari berbagai ancaman, termasuk serangan MITM dan penyadapan oleh pihak ketiga [9]. Selain itu, kombinasi algoritma RSA dan AES-256 juga telah banyak diimplementasikan dalam skema E2EE berbasis *Diffie-Hellman* untuk meningkatkan keamanan komunikasi [9]. Analisis terhadap penerapan E2EE pada platform nyata seperti Zoom menunjukkan bahwa meskipun E2EE menawarkan perlindungan terhadap penyadapan oleh pihak ketiga, desain dan implementasinya tetap harus dievaluasi secara ketat untuk mencegah celah keamanan seperti *impersonation* dan serangan *insider* [8]. Dalam konteks penelitian ini, penguatan mekanisme pertukaran kunci dan autentikasi menjadi bagian integral dalam mendukung prinsip E2EE.

2.2 Tinjauan Objek Penelitian

Objek penelitian dalam tesis ini adalah protokol PrivateDH (*Private Diffie-Hellman*) yang diusulkan oleh Patgiri [12], sebuah protokol pertukaran kunci rahasia yang dirancang untuk mendukung komunikasi *end-to-end* melalui integrasi RSA dan AES di atas mekanisme dasar *Diffie-Hellman*. PrivateDH bertujuan untuk menggabungkan keamanan enkripsi asimetris dan simetris guna menghasilkan komunikasi rahasia yang efisien. Protokol ini digunakan sebagai titik awal dalam penelitian ini sebelum dilakukan peningkatan.

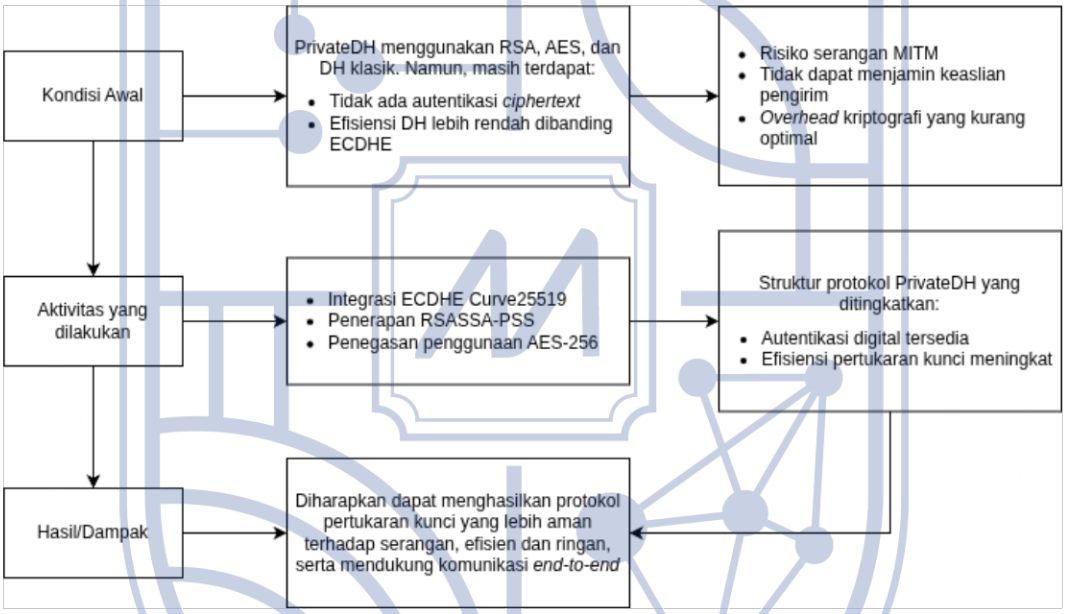
PrivateDH pada implementasi dasarnya masih memiliki kelemahan, seperti ketidakmampuan membedakan antara pengguna sah dan penyerang ketika menerima *ciphertext* identik, dan adanya *overhead* komputasi yang muncul dalam proses enkripsi dan dekripsi, khususnya terkait dengan parameter DH. Penggunaan algoritma *Diffie-Hellman* dapat menimbulkan *overhead* komputasi akibat kebutuhan ukuran kunci yang besar untuk mempertahankan tingkat keamanan modern. Kelemahan-kelemahan tersebut dapat mengekspos komunikasi terhadap risiko serangan *Man-in-the-Middle* (MITM), kegagalan autentikasi, serta berdampak pada efisiensi protokol secara keseluruhan.

Penelitian ini berfokus pada pengembangan peningkatan protokol PrivateDH dengan mengintegrasikan beberapa pendekatan kriptografi modern, yaitu ECDHE Curve25519 untuk menghasilkan kunci sesi yang efisien dan aman, RSASSA-PSS untuk mendukung autentikasi digital, serta AES-256 untuk enkripsi simetris yang lebih kuat. Integrasi ini bertujuan untuk mengatasi kelemahan utama PrivateDH terkait autentikasi data, serta meningkatkan efisiensi dan ketahanan protokol.

Pengujian dan evaluasi protokol yang dikembangkan dilakukan dalam lingkungan simulasi terkontrol untuk mengamati performa keamanan dan fungsionalitas. Penelitian ini tidak mencakup implementasi aplikasi komunikasi *end-to-end* penuh, melainkan fokus pada perbaikan dan pengujian efektivitas protokol pertukaran kunci secara mandiri.

2.3 Kerangka Konseptual

Gambar 2.2 berikut menggambarkan kerangka konseptual dari penelitian ini, dimulai dari kondisi awal protokol PrivateDH, masalah yang timbul, solusi yang diimplementasikan, hingga kondisi baru dan hasil akhir yang diharapkan.



Gambar 2.2 Kerangka Konsep Pemecahan Masalah

Penelitian ini diawali dengan kondisi awal berupa protokol PrivateDH yang menggabungkan algoritma RSA, AES, dan *Diffie-Hellman* (DH) klasik. Meskipun protokol ini telah dirancang untuk mendukung komunikasi *end-to-end*, masih terdapat kelemahan yang signifikan. Kelemahan tersebut mencakup tidak adanya autentikasi digital terhadap *ciphertext*, serta efisiensi kriptografi DH yang relatif lebih rendah dibandingkan pendekatan modern seperti ECDHE. Kondisi ini menyebabkan PrivateDH rentan terhadap serangan *Man-in-the-Middle* (MITM), tidak mampu menjamin keaslian pengirim pesan, dan berpotensi menimbulkan *overhead* komputasi yang kurang optimal.

Untuk menjawab permasalahan tersebut, penelitian ini mengusulkan beberapa tindakan solusi. Pertama, algoritma ECDHE Curve25519 diintegrasikan untuk menggantikan DH klasik guna meningkatkan keamanan dan efisiensi pertukaran kunci. Kedua, RSASSA-PSS diterapkan sebagai mekanisme autentikasi digital untuk memastikan keaslian pengirim dan integritas data. Ketiga, penggunaan AES dengan panjang kunci 256-bit (AES-256) dipertegas untuk menjamin kerahasiaan isi pesan dalam komunikasi. Kombinasi dari ECDHE Curve25519, RSASSA-PSS, dan AES-256 dipilih tidak hanya berdasarkan kecocokannya terhadap kelemahan PrivateDH, tetapi juga karena masing-masing memberikan keunggulan spesifik yang saling melengkapi dalam satu skema protokol.

Sebagai hasil dari integrasi pendekatan tersebut, struktur protokol PrivateDH yang ditingkatkan memiliki karakteristik utama, yaitu autentikasi digital tersedia, dan efisiensi pertukaran kunci meningkat. Dengan perubahan ini, protokol hasil pengembangan diharapkan mampu menjadi sistem pertukaran kunci yang lebih aman terhadap berbagai serangan, lebih efisien secara komputasi, serta mendukung komunikasi *end-to-end* yang kuat dan terpercaya.