Implementasi Algoritma Kriptografi Post-Quantum dalam Sistem Keamanan SSL/TLS

Lie, Kevin Sebastian Sheva Tjahyana
Program Studi Sistem dan Teknologi Informasi
Institut Teknologi Bandung
Bandung, Indonesia
kevinssheva@gmail.com

***Abstract*—Komputasi kuantum mengancam keamanan kriptografi modern dengan kemampuannya untuk memecahkan algoritma tradisional seperti RSA yang digunakan dalam protokol keamanan SSL/TLS. Makalah ini membahas implementasi algoritma kriptografi post-kuantum Kyber sebagai solusi untuk menggantikan RSA dalam mekanisme pertukaran kunci SSL/TLS. Kyber, yang didasarkan pada masalah Learning With Errors (LWE) dan Ring-Learning With Errors (Ring-LWE), menawarkan keamanan yang tahan terhadap serangan kuantum. Implementasi Kyber melibatkan pembangkitan kunci, enkripsi, dan dekripsi yang efisien, meskipun dengan ukuran kunci dan ciphertext yang lebih besar dibandingkan RSA. Studi ini menunjukkan bahwa Kyber tidak hanya meningkatkan keamanan tetapi juga menjaga performa yang optimal dalam komunikasi real-time. Perbandingan antara Kyber dan RSA menunjukkan keunggulan signifikan dalam hal ketahanan terhadap serangan kuantum dan efisiensi operasional, menjadikan Kyber sebagai pilihan ideal untuk protokol SSL/TLS di masa depan. Hasil penelitian ini menegaskan pentingnya mengadopsi algoritma kriptografi post-kuantum untuk memastikan perlindungan jangka panjang terhadap ancaman keamanan digital.*)

Keywords—kyber,post-quantum,cryptography,quantum,computing,ssl,tls,security

I. Pendahuluan

Keamanan informasi merupakan salah satu aspek terpenting dalam era digital saat ini. Penggunaan teknologi internet yang semakin meluas menuntut adanya sistem keamanan yang mampu melindungi data dari berbagai ancaman siber. SSL (Secure Sockets Layer) dan TLS (Transport Layer Security) adalah protokol yang banyak digunakan untuk mengamankan komunikasi melalui jaringan internet. Kedua protokol ini menggunakan algoritma kriptografi untuk memastikan kerahasiaan, integritas, dan autentikasi data yang dikirimkan antara klien dan server.

Saat ini, algoritma kriptografi yang digunakan dalam SSL/TLS seperti RSA, ECC, dan DSA dianggap aman karena mereka bergantung pada kesulitan matematis tertentu, seperti faktorisasi bilangan bulat besar dan logaritma diskrit. Namun, dengan perkembangan pesat dalam bidang komputasi kuantum, algoritma kriptografi klasik ini menjadi rentan terhadap serangan oleh komputer kuantum. Komputer kuantum mampu memecahkan masalah-masalah matematika yang mendasari algoritma kriptografi klasik dengan efisiensi yang jauh lebih tinggi dibandingkan komputer konvensional. Algoritma Shor, misalnya, dapat memfaktorkan bilangan besar dalam waktu yang jauh lebih singkat dibandingkan algoritma terbaik yang berjalan pada komputer klasik.

Mengingat ancaman yang ditimbulkan oleh komputasi kuantum, komunitas kriptografi telah mengembangkan algoritma kriptografi post-quantum yang dirancang untuk tetap aman terhadap serangan komputer kuantum. Algoritma-algoritma ini tidak bergantung pada masalah matematika yang mudah diselesaikan oleh komputer kuantum, melainkan menggunakan pendekatan-pendekatan lain seperti kriptografi berbasis kisi (lattice-based cryptography), kode-kode koreksi kesalahan (code-based cryptography), dan fungsi hash.

Implementasi algoritma kriptografi post-quantum dalam sistem keamanan SSL/TLS menjadi penting untuk memastikan bahwa komunikasi internet tetap aman di era komputasi kuantum. Studi ini akan membahas tentang prinsip-prinsip dasar kriptografi post-quantum, tantangan dalam mengimplementasikannya pada protokol SSL/TLS, serta evaluasi kinerja dan keamanan dari algoritma-algoritma post-quantum yang relevan.

Dengan demikian, penelitian ini bertujuan untuk memberikan wawasan mendalam mengenai potensi penggunaan kriptografi post-quantum dalam meningkatkan keamanan protokol SSL/TLS dan memastikan perlindungan data dalam jangka panjang terhadap ancaman yang ditimbulkan oleh perkembangan teknologi kuantum.

II. SSL/TLS

A. Sejarah dan Mekanisme SSL/TLS

Secure Sockets Layer (SSL) dan *Transport Layer Security (TLS)* adalah protokol kriptografi yang dirancang untuk memfasilitasi saluran komunikasi yang aman antara dua pihak, menyediakan mekanisme untuk pertukaran kunci yang aman, autentikasi, enkripsi, dan integritas data. Tujuan utama SSL/TLS adalah untuk melindungi komunikasi dari serangan man-in-the-middle, penyadapan, serangan replay, dan serangan statistik. Meskipun SSL/TLS mampu mengautentikasi kedua belah pihak dalam komunikasi, dalam praktiknya, biasanya hanya server yang mengautentikasi dirinya kepada klien.

SSL/TLS dirancang tidak hanya untuk keamanan, tetapi juga untuk interoperabilitas, ekstensibilitas, dan efisiensi relatif. Protokol ini terdiri dari dua lapisan utama: lapisan record dan lapisan handshake. Lapisan record bertugas untuk mengambil data dari aplikasi lapisan atas, memecahnya menjadi blok-blok yang lebih mudah dikelola, dan melakukan kompresi, enkripsi kunci simetris, serta menghasilkan digest MAC. Lapisan handshake bertanggung jawab untuk menetapkan sesi dan

negosiasi opsi, menentukan kunci simetris per sesi yang digunakan secara besar-besaran oleh lapisan record.

SSL/TLS berjalan "di atas" Transmission Control Protocol/Internet Protocol (TCP/IP), yang mengatur transportasi dan routing data melalui Internet, dan "di bawah" protokol tingkat lebih tinggi seperti Hypertext Transport Protocol (HTTP), yang menggunakan TCP/IP untuk mendukung tugas aplikasi seperti mengunduh halaman web. SSL memungkinkan server yang mendukung SSL untuk mengautentikasi dirinya kepada klien yang mendukung SSL, dan sebaliknya, serta memungkinkan kedua mesin untuk membentuk koneksi terenkripsi.

B. Versi dan Evolusi SSL/TLS

SSL versi 2.0 diperkenalkan oleh Netscape pada tahun 1994 untuk mentransmisikan data pribadi melalui Internet. SSL versi 2.0 dengan cepat menjadi standar de facto untuk perlindungan kriptografi trafik web. Namun, SSL 2.0 segera terbukti memiliki beberapa kelemahan:

1. Rentan terhadap serangan man-in-the-middle di mana penyerang aktif dapat memaksa klien dan server menggunakan enkripsi 40-bit.
2. Hanya menggunakan fungsi hash MD5 yang memiliki kelemahan keamanan.
3. Menggunakan kode autentikasi pesan (MAC) yang lemah.
4. Memasukkan byte padding ke dalam MAC dalam mode cipher block tanpa mengautentikasi field panjang padding, memungkinkan penyerang aktif menghapus byte dari akhir pesan.
5. Menggunakan kunci yang sama untuk autentikasi dan enkripsi, yang dapat menimbulkan masalah untuk beberapa cipher.

Untuk mengatasi kelemahan ini, SSL versi 3.0 diperkenalkan pada tahun 1996, yang meningkatkan keamanan dan fungsionalitas SSL 2.0. SSL 3.0 tidak hanya memperbaiki kelemahan keamanan yang disebutkan di atas, tetapi juga mengurangi jumlah perjalanan jaringan, memungkinkan server memilih cipher, mendukung algoritma pertukaran kunci dan cipher yang lebih lengkap, serta menggunakan kunci autentikasi dan enkripsi yang terpisah. Sekitar waktu yang sama, Microsoft memperkenalkan versinya sendiri dari SSL, yaitu Privacy Communication Technology (PCT), tetapi ini tidak pernah sepopuler SSL 3.0.

Pada tahun 1996, Internet Engineering Task Force (IETF) membentuk Transport Layer Security (TLS) Working Group untuk menciptakan versi standar dari SSL dan PCT. Protokol standar yang dinamai TLS versi 1.0 sangat mirip dengan SSL 3.0, hingga TLS 1.0 sering disebut sebagai SSL 3.1. Perbedaan paling penting antara keduanya adalah bahwa TLS menggunakan algoritma Keyed-Hashing for Message Authentication Code (HMAC) yang menghasilkan hash yang lebih aman dibandingkan dengan algoritma MAC SSL. Perbedaan kecil lainnya termasuk pengecualian algoritma Fortezza dan diperbolehkannya sertifikat kembali ke otoritas sertifikat perantara daripada langsung ke otoritas sertifikat root. Karena perbaikan-perbaikan ini, protokol TLS dan SSL 3.0 tidak sepenuhnya dapat dioperasikan, tetapi TLS 1.0 memiliki mode untuk kembali ke SSL 3.0.

C. Penggunaan SSL/TLS

Banyak situs web sekarang menggunakan SSL untuk memperoleh informasi rahasia pengguna, seperti nomor kartu kredit dan nomor jaminan sosial. Semua browser web utama (Mozilla Firefox, Netscape Navigator, Internet Explorer, Safari, dan Opera) mendukung SSL. SSL 3.0 dan TLS diyakini cukup aman jika digunakan dengan benar. Namun, SSL 2.0 memiliki masalah desain fundamental dan tidak boleh digunakan untuk informasi sensitif.

Dengan demikian, SSL/TLS telah berevolusi untuk memenuhi tantangan keamanan yang terus berkembang di internet, namun penting untuk menggunakan versi yang lebih aman seperti TLS 1.0 atau yang lebih baru untuk memastikan perlindungan data yang optimal.

D. Keamanan SSL/TLS

SSL/TLS menggunakan berbagai algoritma kriptografi untuk mencapai tujuan utamanya yaitu memastikan kerahasiaan, integritas, dan autentikasi data yang dikirimkan antara klien dan server. Algoritma ini dapat dibagi menjadi beberapa kategori utama: algoritma pertukaran kunci, algoritma enkripsi simetris, dan algoritma hash untuk integritas pesan. Berikut adalah penjelasan tentang algoritma-algoritma kriptografi yang digunakan oleh SSL/TLS:

1. Algoritma Pertukaran Kunci

   Algoritma pertukaran kunci digunakan selama fase handshake untuk menetapkan kunci enkripsi yang akan digunakan selama sesi komunikasi. Algoritma yang digunakan adalah:

   - RSA (Rivest-Shamir-Adleman)

     RSA digunakan baik untuk pertukaran kunci dan autentikasi. Kunci publik RSA server digunakan oleh klien untuk mengenkripsi data yang akan dikirim ke server, yang kemudian hanya dapat didekripsi oleh server dengan kunci privatnya. RSA juga digunakan untuk menandatangani sertifikat digital yang diverifikasi oleh klien untuk memastikan bahwa server adalah entitas yang sah.

   - Diffie-Hellman (DH)

     DH memungkinkan dua pihak untuk secara aman membuat kunci enkripsi bersama di atas saluran komunikasi yang tidak aman. SSL/TLS menggunakan varian seperti Ephemeral Diffie-Hellman (DHE) untuk memastikan bahwa kunci sesi unik

untuk setiap sesi dan tidak dapat digunakan kembali.

2. Algoritma Enkripsi Simetris

Setelah kunci sesi dibuat selama handshake, algoritma enkripsi simetris digunakan untuk mengenkripsi data yang ditransmisikan antara klien dan server. Beberapa algoritma yang umum digunakan adalah:

- AES (Advanced Encryption Standard)

  AES adalah standar enkripsi simetris yang paling banyak digunakan dalam SSL/TLS. AES menawarkan kekuatan enkripsi yang tinggi dan efisiensi dengan ukuran kunci 128-bit, 192-bit, atau 256-bit.

- ChaCha20

  ChaCha20 adalah algoritma enkripsi stream yang dirancang untuk memberikan keamanan yang kuat dan performa tinggi, terutama pada perangkat yang kurang optimal untuk AES.

3. Algoritma Hash untuk Integritas Pesan

Algoritma hash digunakan dalam SSL/TLS untuk memastikan integritas data. Algoritma yang digunakan adalah SHA (Secure Hash Algorithm), termasuk SHA-1 dan SHA-2. Algoritma ini digunakan untuk membuat MAC yang memastikan integritas data dengan mendeteksi perubahan pada data. SHA-256 dan SHA-384 adalah varian yang paling umum digunakan saat ini karena mereka menyediakan keamanan yang lebih tinggi dibandingkan SHA-1.

4. Algoritma Sertifikat dan Tanda Tangan Digital

Untuk memastikan autentikasi pihak yang terlibat dalam komunikasi, SSL/TLS menggunakan sertifikat digital yang ditandatangani oleh Certificate Authority (CA). Algoritma yang umum digunakan adalah:

Dengan penggunaan berbagai algoritma kriptografi ini, SSL/TLS menyediakan keamanan yang komprehensif untuk komunikasi internet. Namun, dengan perkembangan teknologi dan ancaman baru seperti komputasi kuantum, ada kebutuhan yang terus-menerus untuk memperbaiki dan meningkatkan algoritma-algoritma yang digunakan untuk memastikan bahwa SSL/TLS tetap aman.

III. Quantum Computing

A. Komputer Klasik

Dalam komputer klasik, segala sesuatu yang dapat diwakili sebagai bit, termasuk bilangan bulat, kata, gambar, musik, video, dll. Sama seperti setiap informasi atau data dapat diwakili sebagai bit, semua pemrosesan informasi pada dasarnya dapat dipecah menjadi gerbang logika Boolean. Ada tujuh jenis gerbang logika dalam komputasi klasik yang mengambil satu atau dua bit sebagai input dan menghasilkan bit baru sebagai hasilnya, masing-masing dengan cara yang unik. Setiap komputasi hanyalah pengaturan khusus dari gerbang-gerbang ini. Transistor berfungsi seperti saklar yang dapat mengaktifkan atau menghentikan aliran elektron berdasarkan set tegangan. Transistor ini digunakan untuk membuat gerbang logika.

B. Komputer Quantum

Di dunia kuantum, ada berbagai cara baru untuk merepresentasikan informasi. Ada nol dan satu seperti dalam komputasi klasik, tetapi selain itu, terdapat banyak keadaan antara nol dan satu yang memungkinkan karena superposisi. Bit-bit kuantum ini disebut "qubit". Seperti komputer klasik, program pertama untuk komputer kuantum hanyalah pengaturan dari nol dan satu. Dalam program kuantum, qubit dapat berada dalam berbagai posisi antara nol dan satu. Dalam superposisi, qubit memiliki beberapa probabilitas berada dalam keadaan nol dan beberapa probabilitas berada dalam keadaan satu. Meskipun qubit dapat berada dalam banyak keadaan dibandingkan bit klasik, saat qubit dibaca, superposisinya akan runtuh menjadi nol atau satu. Keadaan atau orientasi bit sebelum runtuh tidak akan pernah diketahui, tetapi probabilitas pengukuran terkait dengan keadaan asli qubit.

Komputasi kuantum tumbuh secara eksponensial dalam ukuran. Dua bit klasik dapat mewakili empat kemungkinan keadaan (00, 01, 10, 11) tetapi hanya bisa berada dalam satu keadaan pada satu waktu. Ini membatasi komputer klasik untuk memproses satu input pada satu waktu. Sebaliknya, dalam kasus kuantum, dua bit dapat mewakili empat keadaan yang sama, tetapi karena superposisi, qubit dapat mewakili keempat keadaan secara bersamaan. Semua pengukuran dalam dunia kuantum bersifat probabilistik. Karena fakta ini, program yang sama dieksekusi beberapa kali untuk mencapai hasil yang relevan. Jadi, untuk menyelesaikan masalah pada komputer kuantum, pernyataan masalah pertama kali dirancang atau disusun sedemikian rupa sehingga setelah dieksekusi di komputer kuantum, probabilitas yang didapatkan memungkinkan untuk menyimpulkan jawaban yang diinginkan. Sebagai contoh pada gambar di bawah ini yang menunjukkan superposisi dengan keadaan $|0\rangle$ memiliki probabilitas 2/3, sedangkan keadaan $|1\rangle$ memiliki probabilitas 1/3. Resultan dari superposisi tersebut memiliki magnitudo 1 dan terletak pada keliling lingkaran dua dimensi.

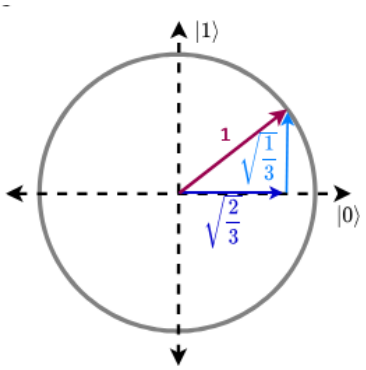


Fig. 1. Representasi grafis dari qubit

Komputer kuantum dapat lebih cepat dibandingkan dengan komputer klasik karena mereka memanfaatkan prinsip-prinsip dasar mekanika kuantum seperti superposisi, keterkaitan (entanglement), dan interferensi kuantum. Dalam komputasi klasik, bit hanya dapat berada dalam salah satu dari dua keadaan, 0 atau 1. Sebaliknya, qubit dalam komputer kuantum dapat berada dalam kombinasi linear dari kedua keadaan ini, memungkinkan superposisi yang membuat qubit dapat mewakili banyak kemungkinan keadaan secara bersamaan. Keterkaitan kuantum memungkinkan dua atau lebih qubit menjadi saling terkait sedemikian rupa sehingga keadaan satu qubit akan langsung mempengaruhi keadaan qubit lainnya, meningkatkan efisiensi dan kecepatan komputasi. Interferensi kuantum memungkinkan algoritma kuantum untuk mengatur dan mengkombinasikan berbagai kemungkinan hasil secara konstruktif atau destruktif, memperkuat jalur komputasi yang benar.

Selain itu, qubit dapat memproses banyak jalur solusi secara paralel, memungkinkan komputer kuantum mengeksplorasi banyak solusi sekaligus. Dengan setiap tambahan qubit, kemampuan komputer kuantum bertambah secara eksponensial, sedangkan komputer klasik bertambah secara linier. Oleh karena itu, komputer kuantum dapat memecahkan masalah tertentu yang sangat sulit atau bahkan tidak mungkin diselesaikan oleh komputer klasik dalam waktu yang wajar. Namun, penting untuk dicatat bahwa kecepatan ini bergantung pada jenis masalah dan algoritma yang digunakan. Tidak semua masalah cocok untuk komputasi kuantum, tetapi untuk beberapa aplikasi spesifik, keunggulan komputasi kuantum sangat signifikan [4].

C. Dampak Komputer Quantum

Komputasi kuantum menghadirkan ancaman signifikan terhadap sistem keamanan informasi modern, terutama algoritma kriptografi konvensional seperti RSA, Diffie-Hellman, dan Elliptic Curve Cryptography (ECC) yang saat ini mendasari protokol keamanan seperti SSL/TLS. Dengan kemampuan komputasi paralel yang jauh lebih kuat, komputer kuantum dapat memecahkan masalah matematika kompleks yang menjadi dasar keamanan algoritma ini dalam waktu singkat menggunakan algoritma kuantum seperti *Shor's algorithm*. Ini berarti data sensitif yang sebelumnya aman dapat menjadi rentan terhadap serangan, mengancam kerahasiaan, integritas, dan autentikasi data dalam komunikasi digital. Untuk mengatasi ancaman ini, diperlukan pengembangan dan adopsi algoritma kriptografi post-kuantum yang dapat menjamin keamanan di era komputasi kuantum.

Algoritma Kriptografi	**Tipe**	**Kegunaan**	**Dampak Quantum Computing**
AES-256	Kunci Simetris	Encryption	Aman
SHA-256, SHA-3	–	Hash Function	Aman
RSA	Kunci Asimetris	Signature, Key Establishment	Tidak Aman
ECDSA, ECDH (Elliptic Curve Cryptography)	Kunci Asimetris	Signature, Key Exchange	Tidak Aman
DSA (Finite Field Cryptography)	Kunci Asimetris	Signature, Key Exchange	Tidak Aman
Diffie-Hellman	Kunci Asimetris	Key Exchange	Tidak Aman

Table 1. Dampak Quantum Computing terhadap Kriptografi Modern

IV. Shor's Algorithm

Shor's Algorithm, yang dikembangkan oleh matematikawan Peter Shor pada tahun 1994, adalah algoritma kuantum yang terkenal karena kemampuannya untuk memecahkan masalah faktorisasi bilangan bulat dan menghitung logaritma diskrit dalam waktu polinomial. Algoritma ini menunjukkan potensi besar dari komputer kuantum dalam mengatasi masalah yang sangat sulit diselesaikan oleh komputer klasik.

A. Prinsip Dasar Shor's Algorithm

Shor's Algorithm memanfaatkan sifat superposisi dan interferensi kuantum untuk mempercepat proses faktorisasi. Algoritma ini dapat dibagi menjadi dua tahap utama: tahap klasik dan tahap kuantum.

1. Tahap Klasik

   Pada tahap ini, algoritma dimulai dengan memilih bilangan acak a yang relatif prima terhadap bilangan yang akan difaktorkan, N. Jika a sudah menjadi faktor dari N, algoritma selesai. Jika tidak, algoritma berlanjut ke tahap kuantum.

2. Tahap Quantum

Pada tahap ini, algoritma menggunakan komputer kuantum untuk menemukan periode dari fungsi eksponensial

$$f(x) = a^x \bmod N$$

Penemuan periode ini adalah kunci untuk faktorisasi bilangan N. Algoritma ini menggunakan transformasi Fourier kuantum untuk menemukan periode dengan efisiensi yang jauh lebih tinggi daripada metode klasik.

B. Langkah-langkah Shor's Algorithm

Berikut adalah langkah-langkah rinci dari Shor's Algorithm dalam mencari faktor prima dari N:

1. Pilih bilangan acak a yang relatif prima terhadap N. Jika $gcd(a, N) \neq 1$, maka a adalah faktor dari N.
2. Siapkan dua register kuantum. Register pertama diinisialisasi ke superposisi dari semua kemungkinan nilai input. Register kedua diinisialisasi ke keadaan dasar (ground state).
3. Hitung nilai fungsi $f(x) = a^x \bmod N$ dan simpan hasilnya di register kedua. Ini menciptakan superposisi dari semua kemungkinan pasangan $(x, f(x))$.
4. Terapkan transformasi Fourier kuantum pada register pertama untuk menemukan periode dari fungsi $f(x)$.
5. Ukur register pertama untuk mendapatkan nilai yang terkait dengan periode r. Pengukuran ini memungkinkan algoritma untuk mendapatkan informasi tentang periode dari fungsi eksponensial $f(x) = a^x \bmod N$.
6. Dengan hasil pengukuran, gunakan metode klasik untuk menentukan nilai $r$yang merupakan periode dari fungsi $f(x)$. Ini melibatkan beberapa langkah tambahan dalam domain klasik untuk memverifikasi dan menghitung periode yang tepat.
7. Jika periode r adalah genap, dan $a^{r/2} \neq -1 \bmod N$, maka kita dapat menemukan faktor-faktor dari N sebagai $gcd(a^{r/2} - 1, N)$ dan $gcd(a^{r/2} + 1, N)$. Ini adalah faktor non-trivial dari N.
8. Jika nilai periode yang diperoleh tidak memenuhi kondisi untuk faktorisasi, ulangi algoritma dengan memilih nilai aaa yang berbeda.

C. Keuntungan dan Implikasi Shor's Algorithm

Beberapa keuntungan utama dari penggunaan algoritma Shor adalah :

1. Kecepatan, Shor's Algorithm mampu memecahkan masalah faktorisasi bilangan besar dalam waktu polinomial $O((log\ N)^3)$, yang jauh lebih cepat daripada algoritma klasik terbaik yang diketahui seperti algoritma faktorisasi bilangan bulat umum (General Number Field Sieve) yang memiliki kompleksitas waktu sub-eksponensial.
2. Efisiensi, dengan menggunakan komputer kuantum, Shor's Algorithm dapat menemukan faktor bilangan besar yang mendasari banyak skema kriptografi modern seperti RSA, dalam waktu yang secara praktis tidak dapat dipecahkan oleh komputer klasik.

Di sisi lain, beberapa implikasi algoritma Shor terhadap kriptografi modern adalah :

1. Kerentanan Algoritma RSA, RSA yang memiliki keamanan bergantung pada kesulitan faktorisasi bilangan besar, menjadi rentan terhadap serangan menggunakan Shor's Algorithm. Ini berarti bahwa semua data yang dienkripsi dengan RSA dapat dipecahkan oleh komputer kuantum yang cukup besar.
2. Ancaman terhadap algoritma seperti Diffie-Hellman dan Elliptic Curve Cryptography karena masalah logaritma diskrit yang mendasarinya juga dapat dipecahkan oleh Shor's Algorithm.

Shor's Algorithm menunjukkan potensi luar biasa dari komputer kuantum dalam memecahkan masalah faktorisasi bilangan bulat dan logaritma diskrit, yang menjadi dasar keamanan banyak algoritma kriptografi konvensional. Kemampuan ini menimbulkan ancaman serius terhadap sistem keamanan modern, terutama yang menggunakan RSA dan algoritma serupa. Oleh karena itu, pengembangan dan adopsi algoritma kriptografi post-kuantum menjadi sangat penting untuk menjaga keamanan informasi di era komputasi kuantum. Algoritma ini harus dirancang untuk tetap aman bahkan ketika komputer kuantum dengan kemampuan yang cukup untuk menjalankan Shor's Algorithm tersedia.

V. Post Quantum Cryptography

Post-Quantum Cryptography (PQC) adalah bidang kriptografi yang berfokus pada pengembangan algoritma kriptografi yang aman terhadap serangan dari komputer kuantum. Dengan ancaman nyata yang ditimbulkan oleh algoritma kuantum seperti Shor's Algorithm, yang mampu memecahkan masalah matematika kompleks seperti faktorisasi bilangan bulat dan logaritma diskrit, kebutuhan akan algoritma yang dapat bertahan di era komputasi kuantum menjadi sangat mendesak.

Algoritma post-kuantum dirancang untuk tetap aman bahkan ketika komputer kuantum tersedia. Beberapa karakteristik utama dari algoritma ini meliputi:

1. Ketahanan terhadap serangan kuantum yang dikenal, seperti Shor's Algorithm dan Grover's Algorithm.
2. Algoritma harus dapat diimplementasikan secara efisien pada perangkat keras dan perangkat lunak yang ada, meskipun beberapa mungkin memerlukan lebih banyak sumber daya dibandingkan dengan algoritma konvensional.

3. Algoritma post-kuantum harus dapat diintegrasikan ke dalam protokol dan sistem yang ada tanpa memerlukan perubahan besar.

Pencarian algoritma yang tahan terhadap serangan dari komputer klasik dan kuantum berfokus pada algoritma kunci publik. Beberapa kategori algoritma telah diusulkan untuk pasca-kuantum meliputi :

1. Lattice-based cryptography – Kriptografi ini menggunakan struktur matematika yang kompleks yang disebut kisi untuk mengamankan data. Keunggulannya terletak pada efisiensi dan potensi ketahanan terhadap serangan kuantum, menjadikannya kandidat kuat untuk perlindungan data di era komputasi kuantum.
2. Code-based cryptography – Jenis kriptografi ini memanfaatkan kode koreksi kesalahan, yang awalnya dirancang untuk memperbaiki kesalahan transmisi data. Kode-kode ini diadaptasi untuk membuat skema enkripsi yang sulit dipecahkan, bahkan oleh komputer kuantum.
3. Multivariate polynomial cryptography – Kriptografi ini bergantung pada kesulitan menyelesaikan persamaan polinomial dengan banyak variabel. Meskipun beberapa skema sebelumnya telah dipecahkan, penelitian terus mengembangkan pendekatan baru yang lebih aman.
4. Hash-based signatures – Tanda tangan digital ini dibangun menggunakan fungsi hash, yang merupakan fungsi matematika yang mengubah data menjadi nilai unik yang disebut hash. Keamanan tanda tangan ini bergantung pada keamanan fungsi hash yang mendasarinya.

Post-Quantum Cryptography adalah bidang penting yang bertujuan untuk mengamankan sistem informasi di era komputasi kuantum. Dengan ancaman nyata dari kemampuan komputasi kuantum untuk memecahkan algoritma kriptografi konvensional, pengembangan dan adopsi algoritma post-kuantum menjadi sangat mendesak. Meskipun menghadapi tantangan dalam hal kinerja, ukuran kunci, dan kompatibilitas, upaya global dalam penelitian, pengembangan, dan standarisasi akan memastikan bahwa sistem keamanan informasi tetap kuat dan tahan terhadap ancaman kuantum.

VI. CRYSTALS KYBER

Kyber adalah salah satu algoritma kriptografi post-kuantum yang menjanjikan, dikembangkan sebagai bagian dari upaya global untuk menemukan algoritma yang aman terhadap serangan komputer kuantum. Kyber termasuk dalam kategori lattice-based cryptography, yang didasarkan pada masalah matematika kompleks yang diyakini tahan terhadap serangan kuantum, seperti Learning With Errors (LWE) dan Ring-Learning With Errors (Ring-LWE).

A. Prinsip Dasar Kyber

Kyber menggunakan prinsip-prinsip dari lattice-based cryptography untuk memastikan keamanan yang kuat. Berikut adalah beberapa konsep utama yang mendasari algoritma Kyber:

1. Learning With Errors (LWE)

   Kyber dibangun di atas masalah Learning With Errors (LWE), yang melibatkan penambahan sedikit 'noise' atau kesalahan ke dalam persamaan linier. Masalah ini sangat sulit dipecahkan, bahkan dengan komputer kuantum.

   Sebagai ilustrasi, bayangkan sebuah sistem persamaan linear sederhana dengan A dan B sebagai kunci publik, dan s sebagai kunci privat yang merupakan solusi dari persamaan $A \times s = B$. Persamaan ini mudah diselesaikan dengan metode seperti eliminasi Gauss.

$$\begin{bmatrix} 4 & 12 & 3 & 7 \\ 8 & 2 & 14 & 2 \\ 7 & 17 & 12 & 3 \\ 6 & 9 & 1 & 4 \end{bmatrix} \times \begin{bmatrix} s1 \\ s1 \\ s3 \\ s4 \end{bmatrix} = \begin{bmatrix} 214 \\ 170 \\ 294 \\ 184 \end{bmatrix}$$

   Misalnya, solusi dari persamaan tersebut adalah$s = (12, 9, 3, 7)$. Namun, dengan menambah jumlah persamaan dan menambahkan vektor error e yang berisi bilangan bulat kecil, persamaan menjadi $A \times s + e = B$. Ini membuat perhitungan s menjadi jauh lebih rumit. Selain itu, penggunaan aritmatika modular semakin meningkatkan kompleksitas persamaan.

$$A = \begin{bmatrix} 14 & 15 & 5 & 2 \\ 13 & 14 & 14 & 6 \\ 6 & 10 & 13 & 1 \\ 10 & 4 & 12 & 16 \\ 9 & 5 & 9 & 6 \\ 3 & 6 & 4 & 5 \\ 6 & 7 & 16 & 2 \end{bmatrix} \quad s = \begin{bmatrix} 1 \\ -1 \\ 0 \\ -1 \\ 1 \\ 0 \\ 1 \end{bmatrix} \quad e = \begin{bmatrix} 0 \\ 13 \\ 9 \\ 11 \end{bmatrix} \quad q = 17$$

$$B = A \times s + e \ (mod\, q) = \begin{bmatrix} 8 \\ 16 \\ 3 \\ 12 \\ 9 \\ 16 \\ 3 \end{bmatrix}$$

2. Ring-LWE

   Kyber memperluas LWE menjadi Ring-LWE, yang memungkinkan penggunaan struktur matematika yang lebih efisien. Ini memungkinkan kinerja yang lebih baik dalam hal kecepatan dan ukuran kunci, sambil tetap mempertahankan keamanan yang kuat. Ring-LWE adalah permasalahan matematis dalam kriptografi berbasis kisi (Lattice-based), di mana tujuannya adalah menyembunyikan sebuah polinomial rahasia dalam *noisy* data yang

diambil dari *ring* terstruktur. Beberapa hal yang harus diperhatikan adalah sebagai berikut :

1. $a(x)$ adalah sebuah polinomial dari *ring* polinomial $Z_p[X]/(X^n + 1)$ di mana semua koefisiennya berasal dari Z_p
2. $e(x)$ dan $s(x)$ juga merupakan polinomial dari $Z_p[X]/(X^n + 1)$ dengan koefisien kecil
3. Lalu, $b(x) = a(x) . s(x) + e(x)$ di mana $b(x)$ juga merupakan polinomial dari $Z_p[X]/(X^n + 1)$

Perhatikan bahwa di sini semua a, b, s, dan e adalah polinomial, sedangkan dalam LWE A adalah matriks.

Beberapa operasi yang bisa diterapkan dalam *ring* polinomial adalah :

1. Penjumlahan

   Penjumlahan polinomial dalam ring ini mirip dengan cara tradisional menjumlahkan polinomial, kecuali koefisiennya harus berasal dari $\mathbb{Z}_p$. Sebagai contoh, jika cincin didefinisikan modulo $x^3 + 1$, hasil penjumlahan polinomial di atas mungkin akan menjadi :

   $$\left(2x^2 + 3x + 1\right) + \left(x^2 - 4x + 5\right) \equiv 3x^2 - x + 6$$

2. Perkalian

   $$a,\ b\ \in \mathbb{Z}_{13}\left[X\right]/\left(X^4 + 1\right)$$

   $$a = \left(4x^3 + x^2 + 11x + 10\right)$$

   $$b = \left(6x^3 + 9x^2 + 11x + 11\right)$$

   Untuk melakukan perkalian a×ba \times ba×b, kita akan mengonversi polinomial aaa menjadi matriks sirkulan. Seperti yang dapat dilihat pada gambar, setiap kolom dari matriks AAA adalah pergeseran siklik dari kolom sebelumnya di mana elemen terakhirnya dinegasikan saat dipindahkan ke depan.

$$A = \begin{bmatrix} 4 & -10 & -11 & -1 \\ 1 & 4 & -10 & -11 \\ 11 & 1 & 4 & -10 \\ 10 & 11 & 1 & 4 \end{bmatrix} \quad B = \begin{bmatrix} 6 \\ 9 \\ 11 \\ 11 \end{bmatrix}$$

$$A \times B \ mod\, q = \begin{bmatrix} -198 \\ -189 \\ 9 \\ 214 \end{bmatrix} \quad mod\, 13 \ = \begin{bmatrix} 10 \\ 6 \\ 9 \\ 6 \end{bmatrix}$$

Dengan pemahaman ini, Kyber dapat memanfaatkan struktur matematis yang efisien untuk menjaga keamanan sekaligus meningkatkan kinerja kriptografi berbasis kisi.

B. Mekanisme Kerja Kyber

Algorithm	n	k	q
Kyber512	256	2	3329
Kyber768	256	3	3329
Kyber1024	256	4	3329

Tabel yang diberikan menunjukkan nilai n, k, dan q sesuai dengan spesifikasi Kyber. Namun, untuk menjelaskan cara kerja Kyber, pada makalah ini akan menggunakan parameter yang lebih sederhana yakni $k = 2$, $q = 17$, $n = 4$. Kyber dirancang untuk menyediakan keamanan yang kuat dalam komunikasi digital melalui proses berikut:

1. Key Generation (Pembangkitan Kunci)

   Proses dimulai dengan pembangkitan sepasang kunci publik dan kunci privat. Kunci publik digunakan untuk enkripsi, sementara kunci privat digunakan untuk dekripsi. Algoritma menghasilkan vektor-vektor acak dalam ruang kisi yang besar dan menambahkan noise ke dalamnya untuk menghasilkan kunci-kunci ini.

   Kunci privat dari Kyber menggunakan sejumlah k polinomial yang memiliki derajat n (disebut s). Polinomial ini dihasilkan menggunakan koefisien kecil acak.

   $$s = \left(-x^3 - x^2 + x,\ -x^3 - x\right)$$

   Kunci publik Kyber terdiri dari dua elemen:

   1. Sebuah matriks polinomial acak A (ukuran $k \times k$), polinomial pada matriks A dibuat dengan koefisien kurang dari q

$$A = \begin{pmatrix} 6x^3 + 16x^2 + 16x + 11 & 5x^3 + 3x^2 + 10x + 1 \\ 9x^3 + 4x^2 + 6x + 3 & 6x^3 + x^2 + 9x + 15 \end{pmatrix}$$

2. Sebuah vektor polinomial t, untuk menghitung vektor tersebut dibutuhkan vektor error tambahan e yang juga dihasilkan menggunakan koefisien kecil acak. Kemudian kita dapat menghitung $t = A \times s + e$. Perlu dicatat bahwa semua operasi dilakukan di bawah ring polinomial $Z_{17}[X]/(X^4 + 1)$.

$$e = \left(x^2,\, x^2 - x\right)\ t = \left(16x^3 + 15x^2 + 7,\, 10x^3 + 12x^2 + 11x\right.$$

Sekarang, s sudah aman sebagai kunci privat dan $(A,\, t)$ sebagai kunci publik yang ditunjukkan kepada orang lain.

2. Encryption (Enkripsi)

Pengirim menggunakan kunci publik penerima untuk mengenkripsi pesan. Proses enkripsi melibatkan penghitungan produk vektor-vektor kisi dan penambahan noise untuk menghasilkan ciphertext.

Untuk mengenkripsi pesan m, kita perlu mengkonversinya menjadi polinomial biner. Kemudian kita perlu mengkalikannya dengan $[q/2]$ (integer terdekat ke $[q/2]$). Mari kita ambil 11 sebagai pesan untuk contoh ini.

$$m_b = x^3 + x + 1$$

$$m = \left[\frac{q}{2}\right] \times m_b \;=\; 9x^3 + 9x + 9$$

Lalu, dibutuhkan 3 polinomial kecil acak $r,\, e_1,\, e_2$

$$r = \left(-x^3 + x^2,\, x^3 + x^2 - 1\right)$$

$$e_1 = \left(x^2 + x,\, x^2\right)$$

$$e_2 = -x^2 - x$$

Kemudian kita mengenkripsi nilai m menggunakan kunci publik $(A,\, t)$. Prosedur enkripsi menghitung dua nilai u dan v.

$$u = A^T r + e_1$$

$$v = t^T r + e_2 + m$$

$$u = \left(11x^3 + 11x^2 + 10x + 3,\, 4x^3 + 4x^2 + 13x + 11\right)$$

$$v = \left(7x^3 + 6x^2 + 8x + 15\right)$$

3. Decryption (Dekripsi)

Dalam mendekripsi, digunakan kunci privat polinomial s untuk mengambil pesan rahasia m dari ciphertext. Perlu dicatat bahwa m masih mengandung noise.

$$m_n = v - s^T u$$

$$m_n = e^T r + e_2 + m + s^T e_1$$

$$m_n = 7x^3 + 14x^2 + 7x + 5$$

Noise dapat dihilangkan dengan membandingkan nilai yang diterima dengan pesan valid terdekat. Dalam hal ini, dengan memeriksa apakah lebih dekat ke $[q/2] = 9$ atau 0 (atau q). Kemudian kita mendapatkan polinomial yang dibulatkan, misalnya $9x^3 + 0x^2 + 9x + 5$, dan membaginya dengan 9 akan menghasilkan m. Sekarang kedua pihak memiliki pesan rahasia m yang dapat mereka gunakan untuk enkripsi asimetris.

C. Keunggulan Kyber

Kyber memiliki beberapa keunggulan utama yang menjadikannya kandidat kuat untuk standarisasi sebagai algoritma kriptografi post-kuantum:

1. Keamanan Tinggi, keamanan Kyber didasarkan pada masalah LWE dan Ring-LWE yang sangat sulit dipecahkan, bahkan oleh komputer kuantum. Ini memberikan tingkat keamanan yang sangat tinggi terhadap serangan kuantum.
2. Efisiensi, kyber dirancang untuk efisiensi, dengan proses enkripsi dan dekripsi yang cepat serta ukuran kunci yang relatif kecil dibandingkan dengan beberapa algoritma post-kuantum lainnya. Ini membuat Kyber cocok untuk berbagai aplikasi, termasuk perangkat dengan sumber daya terbatas.
3. Fleksibilitas, kyber dapat diimplementasikan dengan berbagai parameter keamanan yang dapat disesuaikan dengan kebutuhan spesifik. Ini memungkinkan penyesuaian antara tingkat keamanan dan kinerja.

D. Status Standarisasi

Kyber adalah salah satu dari beberapa algoritma yang sedang dievaluasi oleh National Institute of Standards and Technology (NIST) dalam proses standarisasi kriptografi post-kuantum. Algoritma ini telah menunjukkan kinerja yang sangat baik dalam uji coba dan dianggap sebagai salah satu kandidat utama untuk adopsi global.

E. Perbandingan dengan RSA

VII. Implementasi Kyber dalam Key Exchange

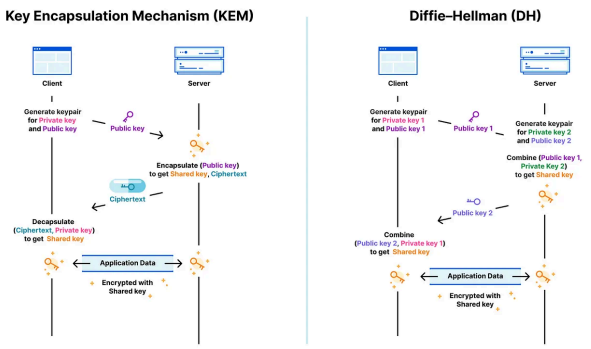
Dengan meningkatnya ancaman dari komputer kuantum yang dapat memecahkan algoritma kriptografi konvensional, seperti RSA dan Diffie-Hellman, penting untuk mengadopsi algoritma kriptografi yang tahan terhadap serangan kuantum. Kyber, sebagai salah satu algoritma kriptografi post-kuantum, menawarkan solusi yang kuat dan efisien untuk menggantikan algoritma key exchange dalam protokol SSL/TLS. SSL (Secure Sockets Layer) dan penerusnya TLS (Transport Layer Security) adalah protokol yang digunakan untuk mengamankan komunikasi di internet. Salah satu komponen penting dalam protokol ini adalah mekanisme key exchange, yang digunakan untuk mendirikan sesi terenkripsi antara klien

dan server. Algoritma key exchange seperti RSA dan Diffie-Hellman saat ini mendominasi, tetapi keduanya rentan terhadap serangan kuantum.

A. Perbandingan Kyber dengan Diffie-Helman

Kyber menggunakan metode Key Encapsulation Mechanism (KEM), sebuah konsep yang cukup mirip dengan algoritma Diffie-Helman. Dalam KEM, proses dimulai dengan client yang menghasilkan sepasang kunci (kunci privat dan kunci publik) dan mengirim kunci publik tersebut ke server. Server kemudian menggunakan kunci publik ini untuk melakukan enkapsulasi, menghasilkan kunci bersama dan ciphertext yang dikirim kembali ke client. Client kemudian mendekapsulasi ciphertext menggunakan kunci privatnya untuk mendapatkan kunci bersama. Dengan kunci bersama ini, kedua belah pihak dapat mengenkripsi data aplikasi.

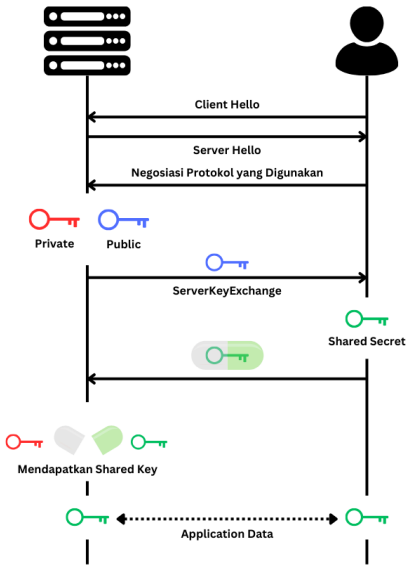
Di sisi lain, metode Diffie-Hellman bekerja dengan client yang menghasilkan sepasang kunci (kunci privat 1 dan kunci publik 1) dan mengirim kunci publik 1 ke server. Server kemudian menghasilkan pasangan kunci kedua (kunci privat 2 dan kunci publik 2), menggunakan kunci publik 1 bersama kunci privat 2 untuk menghasilkan kunci bersama, dan mengirim kunci publik 2 kembali ke client. Client kemudian menggabungkan kunci publik 2 dari server dengan kunci privat 1 untuk mendapatkan kunci bersama. Kunci bersama ini digunakan oleh kedua belah pihak untuk mengenkripsi data aplikasi. KEM melibatkan proses enkapsulasi dan dekapsulasi, sementara Diffie-Hellman melibatkan pertukaran kunci publik dan kombinasi dengan kunci privat masing-masing pihak untuk menghasilkan kunci bersama, memungkinkan berbagi kunci secara aman tanpa mengirimkannya langsung.



Kyber dapat diintegrasikan ke dalam SSL/TLS sebagai algoritma key exchange yang aman terhadap kuantum.

B. Design Key Exchange dengan Kyber

Berikut adalah ilustrasi proses pertukaran kunci dalam protokol TLS (Transport Layer Security) untuk menjamin komunikasi yang aman antara klien dan server.



Berikut adalah penjelasan setiap langkah yang terlibat dalam proses tersebut:

1. Client Hello
   Klien mengirimkan pesan "Client Hello" kepada server untuk memulai koneksi. Pesan ini berisi informasi seperti versi TLS yang didukung, cipher suite (algoritma kriptografi) yang didukung, dan data lainnya yang diperlukan untuk keamanan koneksi.
2. Server Hello
   Server merespons dengan pesan "Server Hello" yang mencakup pilihan cipher suite, versi TLS yang akan digunakan, dan informasi lainnya yang diperlukan untuk melanjutkan negosiasi.
3. Negosiasi Protokol yang Digunakan
   Klien dan server melakukan negosiasi dan menyetujui protokol, versi TLS, dan cipher suite yang akan digunakan selama sesi komunikasi.
4. ServerKeyExchange
   Server mengirimkan kunci publiknya (public key) kepada klien. Ini bisa juga termasuk sertifikat digital yang memverifikasi identitas server.
5. Shared Secret
   Menggunakan kunci publik yang diterima dari server dan kunci privat (private key) milik klien, klien dan server secara independen menghasilkan "shared secret" atau kunci bersama. Shared secret ini digunakan untuk membuat kunci simetris (symmetric key) yang digunakan untuk mengenkripsi data selama sesi.
6. Mendapatkan Shared Key
   Klien dan server menggunakan shared secret untuk menghasilkan shared key yang sama. Shared key ini adalah kunci simetris yang akan digunakan untuk mengenkripsi dan

mendekripsi data yang dikirimkan selama sesi komunikasi.

7. Application Data

Setelah shared key berhasil dihasilkan dan sesi TLS diamankan, klien dan server mulai bertukar data aplikasi yang dienkripsi menggunakan shared key ini. Data aplikasi yang dikirim dan diterima antara klien dan server dijamin kerahasiaannya dan keutuhannya oleh TLS.

C. Implementasi Algoritma Kyber

Kyber dapat diintegrasikan ke dalam SSL/TLS sebagai algoritma key exchange yang aman terhadap kuantum. Berikut adalah langkah-langkah utama dalam implementasi Kyber untuk key exchange pada SSL/TLS :

1. Inisialisasi

Ketika klien menginisiasi koneksi dengan server, klien dan server setuju untuk menggunakan Kyber sebagai algoritma key exchange. Ini dilakukan melalui mekanisme negosiasi cipher suite dalam proses handshake SSL/TLS.

2. Pembangkitan Kunci

Pada server side, server menghasilkan sepasang kunci publik dan kunci privat menggunakan algoritma Kyber. Kunci publik kemudian dikirim ke klien sebagai bagian dari proses handshake. Di sisi client side, klien menerima kunci publik dari server dan menggunakan kunci ini untuk mengenkripsi sesi kunci sementara (temporary session key) yang akan digunakan untuk komunikasi selanjutnya. Berikut adalah kode untuk menghasilkan pasangan kunci menggunakan pustaka pqcrypto untuk Kyber512.

```
from pqcrypto.kem.kyber512 import
generate_keypair

# Generate public and private keys
public_key, private_key =
generate_keypair()

print("Public Key:", public_key)
print("Private Key:", private_key)
```

3. Enkripsi dan Pengiriman Sesi Kunci

Klien mengenkripsi sesi kunci menggunakan kunci publik server yang diterima dan mengirimkan ciphertext kembali ke server. Ciphertext ini memastikan bahwa hanya server yang memiliki kunci privat yang sesuai yang dapat mendekripsi dan memperoleh sesi kunci. Berikut adalah contoh kodenya:

```
from pqcrypto.kem.kyber512 import
encapsulate

# Encrypt session key using server's
public key
ciphertext, shared_secret_client =
encapsulate(public_key)

print("Ciphertext:", ciphertext)
print("Shared Secret (Client):",
shared_secret_client)
```

4. Dekripsi Sesi Kunci

Server menggunakan kunci privatnya untuk mendekripsi ciphertext yang diterima dari klien dan mengekstrak sesi kunci. Sesi kunci ini kemudian digunakan untuk mengenkripsi dan mendekripsi komunikasi antara klien dan server selama sesi tersebut. Berikut adalah contoh kodenya:

```
from pqcrypto.kem.kyber512 import
decapsulate

# Decrypt the session key using private
key
shared_secret_server =
decapsulate(ciphertext, private_key)

print("Shared Secret (Server):",
shared_secret_server)
```

5. Penggunaan Sesi Kunci

Dengan sesi kunci yang sekarang dimiliki oleh kedua belah pihak, klien dan server melanjutkan komunikasi menggunakan kunci simetris ini untuk memastikan keamanan dan integritas data yang ditransmisikan. Berikut adalah contoh menggunakan AES-GCM untuk mengenkripsi dan mendekripsi data:

```
from
cryptography.hazmat.primitives.ciphers.a
ead import AESGCM
import os

# Example session key (shared secret)
key = shared_secret_client  # This would
be the same as shared_secret_server

# Initialize AES-GCM with the session
key
aesgcm = AESGCM(key)

# Encrypt data
nonce = os.urandom(12)  # Generate a
random nonce
```

```
data = b"Secure communication using
Kyber key exchange"
ciphertext = aesgcm.encrypt(nonce, dat
None)

print("Encrypted Data:", ciphertext)

# Decrypt data
decrypted_data = aesgcm.decrypt(nonce,
ciphertext, None)

print("Decrypted Data:", decrypted_dat
```

D. Keuntungan Menggunakan Kyber dalam SSL/TLS

Menggantikan algoritma key exchange tradisional dengan Kyber dalam protokol SSL/TLS menawarkan beberapa keuntungan:

1. Keamanan tahan kuantum

   Kyber dirancang untuk tahan terhadap serangan dari komputer kuantum, menjamin keamanan komunikasi bahkan dengan kemajuan dalam komputasi kuantum.

2. Efisiensi kinerja

   Kyber menunjukkan kinerja yang baik dalam hal kecepatan enkripsi dan dekripsi serta efisiensi penggunaan sumber daya, menjadikannya cocok untuk aplikasi praktis termasuk dalam lingkungan dengan keterbatasan sumber daya.

3. Kompatibilitas dan fleksibilitas

   Kyber dapat diintegrasikan ke dalam protokol SSL/TLS yang ada dengan relatif mudah, memungkinkan transisi yang lebih mulus ke kriptografi post-kuantum tanpa memerlukan perubahan besar pada infrastruktur yang ada.

4. Ukuran kunci dan ciphertext yang relatif kecil

   Dibandingkan dengan beberapa algoritma post-kuantum lainnya, Kyber memiliki ukuran kunci dan ciphertext yang lebih kecil, yang mengurangi overhead dalam komunikasi dan penyimpanan.

E. Tantangan Implementasi Kyber dalam SSL/TLS

Meskipun Kyber menawarkan banyak keuntungan, ada beberapa tantangan yang perlu diatasi dalam implementasinya :

1. Standarisasi dan adopsi

   Proses standar dan adopsi oleh komunitas global memerlukan waktu dan kolaborasi. Meskipun Kyber adalah kandidat kuat, perlu ada dukungan dari berbagai pihak untuk adopsi yang luas.

2. Kompatibilitas dengan perangkat lama

   Beberapa perangkat dan sistem yang lebih tua mungkin tidak mendukung algoritma baru dengan segera. Diperlukan strategi untuk memastikan kompatibilitas mundur atau migrasi yang efisien.

3. Penyesuaian protokol dan perangkat lunak

   Mengintegrasikan Kyber memerlukan penyesuaian pada perangkat lunak SSL/TLS yang ada, yang mungkin memerlukan pembaruan dan pengujian yang ekstensif untuk memastikan kinerja dan keamanan yang optimal.

Mengintegrasikan Kyber dalam mekanisme pertukaran kunci SSL/TLS adalah langkah maju yang penting dalam meningkatkan keamanan komunikasi digital. Dengan ketahanan terhadap serangan kuantum dan efisiensi yang tinggi, Kyber memastikan bahwa protokol SSL/TLS tetap relevan dan aman di era komputasi kuantum. Upaya ini sangat penting untuk menjaga integritas dan kerahasiaan data dalam komunikasi internet, melindungi pengguna dari potensi ancaman keamanan di masa depan.

VIII. KESIMPULAN

Makalah ini mengeksplorasi implementasi algoritma kriptografi post-kuantum Kyber dalam mekanisme pertukaran kunci pada protokol SSL/TLS untuk mengatasi ancaman yang ditimbulkan oleh komputer kuantum. Komputasi kuantum memiliki potensi untuk meruntuhkan keamanan algoritma kriptografi tradisional seperti RSA, yang saat ini digunakan secara luas dalam mengamankan komunikasi digital. Oleh karena itu, diperlukan algoritma baru yang dapat bertahan terhadap serangan kuantum.

Algoritma Kyber, yang didasarkan pada masalah Learning With Errors (LWE) dan Ring-Learning With Errors (Ring-LWE), menawarkan solusi yang aman dan efisien untuk kebutuhan ini. Studi ini menunjukkan bahwa Kyber memiliki beberapa keunggulan utama dibandingkan RSA, termasuk ketahanan terhadap serangan kuantum, efisiensi dalam pembangkitan kunci, enkripsi, dan dekripsi, serta ukuran kunci dan ciphertext yang dapat dikelola.

Implementasi Kyber dalam SSL/TLS melibatkan langkah-langkah pembangkitan kunci publik dan privat oleh server, enkripsi sesi kunci oleh klien menggunakan kunci publik server, dan dekripsi oleh server untuk mendapatkan sesi kunci yang sama. Penggunaan sesi kunci ini kemudian dilanjutkan dengan enkripsi komunikasi menggunakan algoritma simetris seperti AES-GCM.

Hasil dari implementasi ini menunjukkan bahwa Kyber dapat meningkatkan keamanan komunikasi digital tanpa mengorbankan kinerja, menjadikannya pilihan yang ideal untuk menghadapi tantangan kriptografi di era komputasi kuantum.

IX. SARAN

Berdasarkan temuan dalam penelitian ini, beberapa saran untuk penelitian dan implementasi lebih lanjut adalah sebagai berikut:

1. Pengujian dan Validasi Lebih Lanjut

   Implementasi Kyber dalam berbagai skenario komunikasi digital perlu diuji lebih lanjut untuk memastikan keandalannya di berbagai kondisi dan aplikasi nyata. Pengujian ini harus mencakup berbagai ukuran data, tingkat kompleksitas jaringan, dan berbagai jenis serangan potensial.

2. Optimalisasi Kinerja

   Meskipun Kyber menunjukkan efisiensi yang baik, penelitian lebih lanjut dapat dilakukan untuk mengoptimalkan kinerja algoritma ini dalam lingkungan komputasi yang berbeda, termasuk perangkat dengan sumber daya terbatas seperti IoT.

3. Standarisasi dan Adopsi

   Upaya perlu dilakukan untuk mendorong standarisasi Kyber dan algoritma post-kuantum lainnya dalam protokol keamanan internasional. Hal ini melibatkan kolaborasi dengan badan standarisasi seperti NIST dan IETF.

4. Implementasi Multiplatform

   Pengembangan implementasi Kyber yang kompatibel dengan berbagai platform dan perangkat lunak keamanan yang ada akan membantu dalam mempercepat adopsi algoritma ini di industri.

Dengan menerapkan saran-saran ini, diharapkan Kyber dan algoritma post-kuantum lainnya dapat diadopsi secara luas, memastikan bahwa komunikasi digital tetap aman di tengah ancaman yang berkembang dari komputer kuantum.

Daftar Pustaka

[1] Lee, H. K., Malkin, T., & Nahum, E. (2007). Cryptographic strength of ssl/tls servers. Proceedings of the 7th ACM SIGCOMM Conference on Internet Measurement - IMC '07. doi:10.1145/1298306.1298318

[2] - secure socket layer/transport layer security (SSL/TLS) Protocols for Transport Layer Security. (2016). Introduction to Computer Networks and Cybersecurity, 1056–1099. https://doi.org/10.1201/9781466572140-32

[3] Mavroeidis, V., Vishi, K., D., M., & Jøsang, A. (2018). The impact of quantum computing on present cryptography. International Journal of Advanced Computer Science and Applications, 9(3). https://doi.org/10.14569/ijacsa.2018.090354

[4] Chen, L., Jordan, S., Liu, Y.-K., Moody, D., Peralta, R., Perlner, R., & Smith-Tone, D. (2016). Report on Post-Quantum Cryptography. https://doi.org/10.6028/nist.ir.8105

[5] Bernstein, D. J., & Lange, T. (2017). Post-quantum cryptography. Nature, 549(7671), 188–194. doi:10.1038/nature23461

[6] Federal Information Processing Standards Publication. (2023). Module-Lattice-Based Key-Encapsulation Mechanism Standard. https://doi.org/10.6028/nist.fips.203.ipd

[7] Celi, G. T., Tamvada, G., Celi, S., Westerbaan, B., Meunier, T., Rubin, C. D., Faz-Hernández, A., Tholoniat, P., Kozlov, D., & Wang, M. (2022, February 23). Deep dive into a post-quantum key encapsulation algorithm. The Cloudflare Blog. https://blog.cloudflare.com/post-quantum-key-encapsulation/

[8] Rubin, B. W. D., Westerbaan, B., Rubin, C. D., Kozlov, D., Wang, M., Evans, W., Patton, C., Wu, P., Gonçalves, V., Mandele, A. van der, Ghedini, A., Wood, C., Mehra, R., & Galicer, M. (2024b, January 9). Defending against future threats: Cloudflare goes post-quantum. The Cloudflare Blog. https://blog.cloudflare.com/post-quantum-for-all/

[9] Pathum, U. (2024, April 29). Crystals Kyber : The key to Post-Quantum Encryption. Medium. https://medium.com/@hwupathum/crystals-kyber-the-key-to-post-quantum-encryption-3154b305e7bd

Dengan ini saya menyatakan bahwa makalah yang saya tulis ini adalah tulisan saya sendiri, bukan saduran, atau terjemahan dari makalah orang lain, dan bukan plagiasi.

Lie, Kevin Sebastian Sheva Tjahyana