Mekanisme Rotasi Kunci Berbasis ECDH dan HKDF untuk Komunikasi Antar Layanan Web

Ahmad Rafi Maliki NIM 13522137
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
13522137@std.stei.itb.ac.id, rafimaliki28@gmail.com

***Abstrak*—Arsitektur *web service* telah menjadi standar dalam pengembangan web modern, sehingga memunculkan berbagai permasalahan baru terkait keamanan komunikasi antar layanan. Penelitian ini mengimplementasikan dan membuktikan sebuah mekanisme rotasi kunci berbasis ECDH dan HKDF yang bertujuan untuk meningkatkan keamanan komunikasi antar layanan web, khususnya pada aspek autentikasi, otorisasi, dan nir-penyangkalan. Pendekatan *proof of concept* digunakan untuk memvalidasi mekanisme yang diusulkan melalui sebuah infrastruktur *web service* yang dijalankan pada jaringan Docker lokal.**

***Keywords*—Rotasi Kunci, ECDH, HKDF, *Web Service Security*.**

I. Latar Belakang

Arsitektur *web service* telah menjadi pendekatan yang populer dalam pengembangan aplikasi web modern. Arsitektur ini membagi sistem menjadi komponen-komponen layanan yang terpisah dan saling berkomunikasi melalui antarmuka terstandar [1]. Dalam praktiknya, arsitektur *web service* sering diimplementasikan dalam lingkungan sistem terdistribusi yang melibatkan banyak layanan dan perantara komunikasi, sehingga meningkatkan kompleksitas pengelolaan keamanan dan risiko paparan terhadap serangan siber.

Salah satu isu yang muncul terkait arsitektur *web service* adalah mengenai keamanan komunikasi antar layanan. Terdapat enam faktor keamanan yang perlu diperhatikan yaitu autentikasi, otorisasi, kerahasiaan, integritas, ketersediaan, dan nir penyangkalan [2].

Teknologi WS-Security dan SSL/TLS menyediakan mekanisme keamanan pada tahap transmisi data, baik pada tingkat pesan maupun pada tingkat transport [3]. Namun, mekanisme tersebut belum menangani aspek manajemen kunci secara dinamis, khususnya dalam lingkungan *web service* yang masih mengandalkan penggunaan kunci API sebagai mekanisme kontrol akses [4].

Kunci API umumnya bersifat statis dan tidak dirancang untuk diubah secara berkala, sehingga meningkatkan risiko kompromi kunci dan penyalahgunaan akses apabila kunci tersebut bocor. Kondisi ini menyebabkan sistem menjadi rentan terhadap serangan seperti *replay attack*, yaitu serangan di mana pesan atau permintaan yang sah direkam dan dikirim ulang oleh penyerang untuk memperoleh akses atau memicu tindakan tertentu [5]. Serangan lain yang mungkin terjadi adalah *key reuse attack*, yaitu serangan yang memanfaatkan penggunaan kunci kriptografi yang sama secara berulang sehingga memungkinkan penyerang menganalisis pola komunikasi atau menurunkan tingkat keamanan kriptografi yang digunakan [6].

Diperlukan suatu mekanisme rotasi kunci untuk mendukung keamanan komunikasi antar *web service*. Mekanisme tersebut diharapkan mampu menghasilkan kunci baru secara berkala guna meminimalkan risiko penyalahgunaan kunci akibat kebocoran atau penggunaan ulang kunci. Komputasi yang ringan menjadi penting agar mekanisme rotasi kunci dapat diterapkan pada lingkungan *web service* yang bersifat terdistribusi dan melibatkan banyak layanan, tanpa memberikan beban tambahan yang berarti terhadap performa sistem.

Penelitian ini berupaya memanfaatkan mekanisme Elliptic Curve Diffie–Hellman (ECDH) sebagai metode pertukaran kunci yang ringan secara komputasi [7], serta Hash-based Key Derivation Function (HKDF) sebagai mekanisme penurunan kunci [8], untuk mendukung penerapan rotasi kunci secara aman dan efisien. Pendekatan ini diharapkan mampu menghasilkan kunci kriptografi baru secara berkala tanpa memerlukan pertukaran kunci yang kompleks, sehingga sesuai untuk diterapkan pada lingkungan komunikasi antar *web service*.

II. Studi Literatur

A. Manajemen Kunci Kriptografi

National Institute of Standards and Technology (NIST) mengeluarkan rekomendasi melalui NIST SP 800-57 yang menjadi acuan dalam manajemen kunci kriptografi. NIST membagi manajemen kunci menjadi empat fase sebagai berikut [9].

1. Fase pra-operasional

   Kunci belum tersedia untuk digunakan dalam operasi kriptografi normal. Kunci mungkin belum dihasilkan atau masih berada pada fase pra-aktivasi. Selain itu, atribut

sistem atau organisasi juga ditetapkan pada fase ini.

2. Fase operasional

Material kunci telah tersedia dan digunakan dalam operasi kriptografi normal. Kunci berada pada status aktif atau ditangguhkan. Kunci dengan status aktif dapat ditetapkan untuk fungsi melindungi data saja, memproses data saja, atau melindungi dan memproses data sekaligus, sedangkan kunci dengan status ditangguhkan hanya dapat digunakan untuk pemrosesan data.

3. Fase pasca-operasional

Material kunci tidak lagi digunakan dalam operasi kriptografi normal, namun akses terhadap material kunci tersebut masih dimungkinkan dan kunci masih dapat digunakan untuk memproses informasi yang telah dilindungi sebelumnya. Pada fase ini, kunci berada pada status dinonaktifkan atau terkompromi. Kunci dalam fase pasca-operasional dapat disimpan dalam arsip

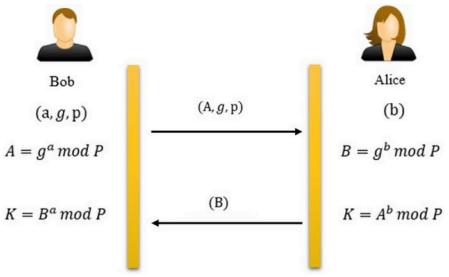
4. Fase penghancuran

Kunci tidak lagi tersedia untuk digunakan. Catatan mengenai keberadaan kunci tersebut dapat dihapus atau tetap disimpan. Kunci berada pada status dimusnahkan. Meskipun kunci itu sendiri telah dimusnahkan, metadata kunci seperti nama kunci, jenis kunci, *cryptoperiod*, dan periode penggunaan, masih dapat dipertahankan untuk keperluan audit atau pencatatan.

B. Elliptic Curve Diffie–Hellman (ECDH)

Elliptic Curve Diffie–Hellman (ECDH) merupakan algoritma pertukaran kunci yang memungkinkan dua pihak untuk menghasilkan kunci bersama bersifat simetris melalui jaringan komunikasi yang tidak aman tanpa melakukan pengiriman kunci secara langsung [7]. ECDH menggunakan konsep Diffie–Hellman, yang merupakan algoritma *key agreement* [10], serta memanfaatkan Elliptic Curve Cryptography (ECC) untuk operasi matematisnya.

Secara sederhana, Diffie–Hellman bekerja dengan memanfaatkan sifat komutatif dari suatu operasi matematika. Pada tahap awal, kedua pihak menyepakati parameter publik yang sama. Selanjutnya, masing-masing pihak melakukan operasi matematika terhadap parameter tersebut menggunakan kunci privatnya. Nilai hasil operasi kemudian dipertukarkan, dan setiap pihak kembali melakukan operasi yang sama menggunakan kunci privatnya sendiri sehingga diperoleh nilai akhir yang identik sebagai kunci bersama [10].



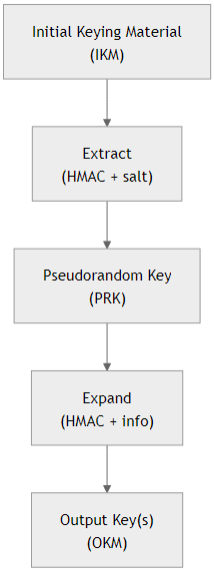
Gambar 1. Cara Kerja Algoritma Diffie–Hellman [12]

Pada Gambar 1 dapat diperhatikan bahwa Alice dan Bob terlebih dahulu menyepakati parameter publik, yaitu sebuah nilai sebagai basis (g) dan sebuah nilai sebagai modulus (P). Selanjutnya, masing-masing pihak melakukan operasi perpangkatan modulo menggunakan kunci privatnya masing-masing, sehingga dihasilkan kunci publik milik Alice ($A = g^a \bmod p$) dan kunci publik milik Bob ($B = g^b \bmod p$). Kedua pihak kemudian melakukan pertukaran kunci publik tersebut dan kembali melakukan operasi yang sama menggunakan kunci privat masing-masing terhadap kunci publik pihak lain. Proses ini menghasilkan nilai akhir (K) yang identik pada kedua belah pihak, yang digunakan sebagai kunci simetris bersama.

Elliptic Curve Cryptography (ECC) adalah metode kriptografi kunci publik yang menggunakan kurva eliptik sebagai dasar perhitungan matematis untuk menyediakan tingkat keamanan yang tinggi dengan ukuran kunci yang relatif kecil dibandingkan dengan algoritma kunci publik konvensional seperti RSA [13]. Penggunaan ukuran kunci yang lebih kecil membuat ECC lebih efisien secara komputasi dan mengurangi kebutuhan sumber daya pemrosesan serta *overhead* komunikasi. Oleh karena itu, ECC banyak diterapkan pada sistem modern, termasuk mekanisme pertukaran kunci dan komunikasi aman pada lingkungan sistem terdistribusi dan *web service*.

C. HMAC-based Key Derivation Function (HKDF)

Key Derivation Function (KDF) merupakan komponen dasar dan esensial dalam sistem kriptografi yang bertujuan untuk mengambil material kunci awal dan menurunkannya menjadi satu atau lebih kunci rahasia yang kuat secara kriptografis. Salah satu penerapan KDF yang umum digunakan adalah HMAC-based Key Derivation Function (HKDF), yang memanfaatkan fungsi hash kriptografis berbasis HMAC untuk menghasilkan kunci turunan yang aman dan independen dari material kunci awal [8].



Gambar 2. Cara Kerja HKDF

HKDF terdiri dari dua fase utama, yaitu ekstrak dan

ekspan sebagai berikut.

1. Fase Ekstrak

Fase ekstrak bertujuan untuk mengekstraksi dan menormalkan *initial keying material* (IKM) yang mungkin belum memiliki kualitas kriptografis yang baik. Pada fase ini, fungsi HMAC digunakan bersama sebuah nilai *salt* untuk menghasilkan sebuah *pseudo random key* (PRK). PRK bersifat lebih acak dan memiliki entropi yang lebih stabil dibandingkan IKM, sehingga aman digunakan sebagai dasar untuk pembangkitan kunci selanjutnya [8].

2. Fase Ekspan

Fase Ekspan bertujuan untuk menurunkan satu atau lebih *output keying material* (OKM) dari PRK yang dihasilkan pada fase ekstrak. Pada fase ini, fungsi HMAC digunakan secara iteratif dengan masukan berupa PRK, dan nilai *context information* (*info*) untuk menghasilkan kunci dengan panjang yang diinginkan [8]. Parameter *info* berfungsi untuk membedakan konteks penggunaan kunci, misalnya untuk enkripsi, autentikasi, atau rotasi kunci. Dengan mekanisme ini, kunci-kunci yang dihasilkan bersifat independen secara kriptografis dan dapat digunakan secara aman untuk berbagai keperluan kriptografi tanpa saling mempengaruhi [14].

D. Autentikasi Berbasis HMAC

Message Authentication Code (MAC) adalah mekanisme kriptografi yang digunakan untuk memastikan integritas dan keaslian suatu pesan dengan memanfaatkan sebuah kunci rahasia bersama. MAC memungkinkan penerima untuk memverifikasi bahwa pesan tidak mengalami perubahan selama transmisi dan benar-benar berasal dari pihak yang memiliki kunci yang sama [15].

Hash-based Message Authentication Code (HMAC) merupakan salah satu konstruksi MAC yang menggunakan fungsi hash bersama kunci rahasia untuk menghasilkan nilai autentikasi pesan [15].

E. Keamanan pada Web Service

Layanan web sebagai bagian dari sistem terdistribusi memerlukan penerapan mekanisme keamanan yang kuat untuk melindungi pertukaran data antar sistem. Sifat layanan web yang independen terhadap bahasa pemrograman, arsitektur, dan platform memungkinkan interoperabilitas yang luas, namun pada saat yang sama meningkatkan tantangan keamanan. Oleh karena itu, aspek keamanan menjadi faktor krusial yang harus diperhatikan dalam perancangan dan implementasi layanan web sebagai berikut [2].

1. Autentikasi

Proses untuk mengidentifikasi pengguna di mana sistem berupaya memastikan identitas pengguna serta memverifikasi bahwa identitas yang diklaim oleh pengguna tersebut adalah benar.

2. Otorisasi

Proses pemberian izin kepada pengguna untuk melakukan suatu tindakan tertentu. Otorisasi sering dipahami sebagai proses awal penetapan hak akses oleh administrator sistem, serta proses pemeriksaan kembali terhadap izin yang telah ditetapkan ketika pengguna mencoba mengakses sumber daya sistem.

3. Kerahasiaan

Prinsip keamanan informasi yang memastikan data yang dikirimkan antar pihak hanya dapat diakses oleh pihak yang berwenang, sehingga mencegah penyadapan oleh pihak ketiga. Kerahasiaan umumnya dijaga melalui penggunaan enkripsi atau VPN.

4. Integritas

Prinsip keamanan informasi yang memastikan bahwa perubahan atau manipulasi terhadap data dapat terdeteksi. Integritas umumnya dijaga menggunakan algoritma matematis seperti fungsi hash.

5. Ketersediaan

Prinsip keamanan informasi yang memastikan bahwa sumber daya dan layanan selalu dapat diakses oleh pihak yang berwenang.

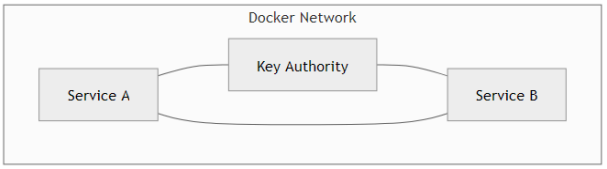
6. Nir-penyangkalan

Prinsip keamanan yang memastikan bahwa pengirim pesan tidak dapat menyangkal telah mengirimkan pesan tersebut.

III. SOLUSI

A. Arsitektur Sistem

Percobaan dilakukan pada sebuah sistem yang terdiri dari tiga buah *service*, yaitu sebuah layanan otoritas kunci yang berfungsi untuk mengelola kunci publik serta *counter* rotasi kunci, dan dua buah *service* lain yang berperan sebagai *service* yang saling berkomunikasi dalam eksperimen. Seluruh *service* dijalankan dalam sebuah jaringan Docker yang saling terhubung, di mana komunikasi antar service dilakukan menggunakan REST API. *Service* target diimplementasikan sebagai server Node.js sederhana yang menyediakan endpoint ping–pong dan memerlukan API *key* sebagai mekanisme autentikasi untuk dapat diakses.

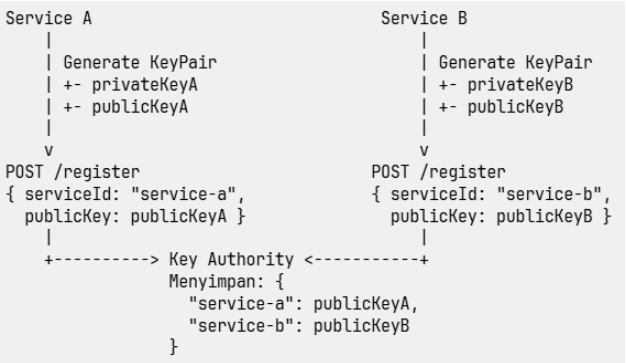


Gambar 3. Arsitektur Sistem

B. Alur Kerja

1. Membangkitkan pasangan kunci ECDH

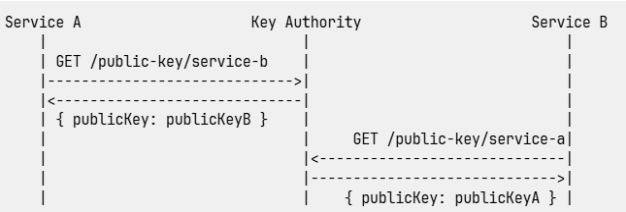
Service A dan *Service* B masing-masing menghasilkan kunci privat dan kunci publik, kemudian mendaftarkan kunci publiknya ke otoritas kunci melalui endpoint POST ‘/register’. *Key authority* menyimpan kunci publik dari setiap *service*, sehingga dapat digunakan pada tahap berikutnya untuk membentuk *shared secret* ECDH.



Gambar 4. Membangkitkan pasangan kunci ECDH

2. Pertukaran kunci publik

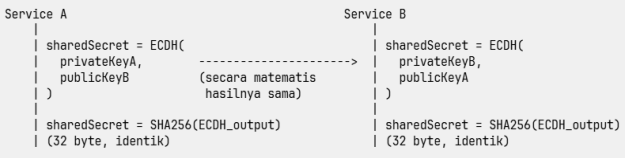
Service A mengambil kunci publik milik *Service* B dengan mengirimkan permintaan GET '/public-key/service-b' ke *Key authority,* sementara Service B mengambil kunci publik milik Service A melalui endpoint GET '/public-key/service-a'.



Gambar 5. Pertukaran Kunci Publik

3. Penurunan *shared secret* dengan ECDH

Service A dan *Service* B masing-masing melakukan perhitungan ECDH dengan mengkombinasikan kunci privat sendiri dan kunci publik pihak lain. Meskipun proses perhitungan dilakukan secara terpisah, secara matematis kedua service menghasilkan nilai ECDH yang sama. Nilai tersebut kemudian diproses menggunakan fungsi hash SHA-256 untuk menghasilkan *shared secret* berukuran 32 byte yang identik.



Gambar 6. Penurunan *Shared Secret*

4. Penurunan kunci dengan HKDF

Nilai *shared secret* yang dihasilkan dari ECDH digunakan sebagai *initial keying material* (IKM) pada HKDF untuk menurunkan kunci-kunci spesifik sesuai arah komunikasi. Parameter *salt* diisi dengan *counter* rotasi untuk mendukung pembaruan kunci secara berkala, sedangkan parameter *info* digunakan sebagai identitas arah komunikasi antara service. Melalui mekanisme ini, kedua service secara independen menghasilkan kunci yang identik untuk komunikasi keluar (*outgoing key*) dan masuk (*incoming key*), sehingga mendukung pemisahan kunci per arah.

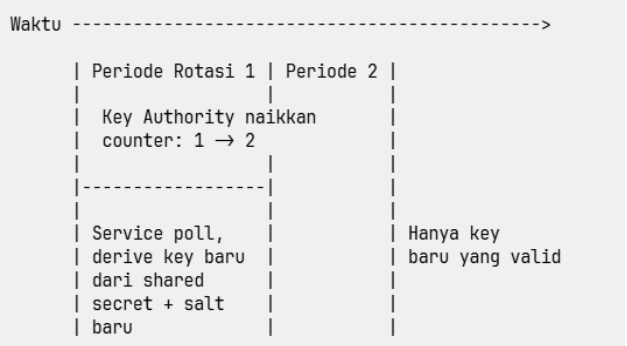
```
outgoingKey = HKDF(
  sharedSecret,              // IKM: dari ECDH (32 byte)
  "rotation-1",              // Salt: counter rotasi
  "service-a→service-b",     // Info: identifier arah
  32                         // Output: key 256-bit
);

incomingKey = HKDF(
  sharedSecret,
  "rotation-1",
  "service-b→service-a",     // Arah sebaliknya
  32
);
```

Gambar 7. Penurunan Kunci Dengan HKDF

5. Rotasi kunci

Setiap periode rotasi, *key authority* menaikkan nilai *counter* rotasi yang digunakan sebagai *salt* dalam proses HKDF. *Service* secara berkala melakukan polling ke *key authority* untuk memperoleh nilai *counter* terbaru, kemudian menurunkan kunci baru dari *shared secret* yang sama dengan *salt* yang diperbarui. Setelah memasuki periode rotasi berikutnya, hanya kunci hasil derivasi terbaru yang dinyatakan valid, sehingga kunci lama tidak lagi digunakan untuk komunikasi.



Gambar 8. Penurunan Kunci Dengan HKDF

IV. Pengujian dan Analisis

A. Metode Pengujian

Pengujian dilakukan dengan menginvokasi *endpoint* pengujian pada masing-masing *service* menggunakan Postman untuk memicu komunikasi antara *service* A dan *service* B. *Endpoint* ini hanya digunakan untuk keperluan pengujian dan tidak disediakan pada implementasi aplikasi nyata. Berikut merupakan endpoint yang digunakan.

Tabel 1
Endpoint Pengujian

Endpoint	Deskripsi
GET /key	Mendapatkan *state* kunci dari sebuah *service*
GET /call-peer=?key=	Memicu pemanggilan *endpoint* ping ke *peer service*

Tabel 1
Endpoint Pengujian (Lanjutan)

Endpoint	Deskripsi
GET /ping	Mensimulasikan komunikasi antar *service*

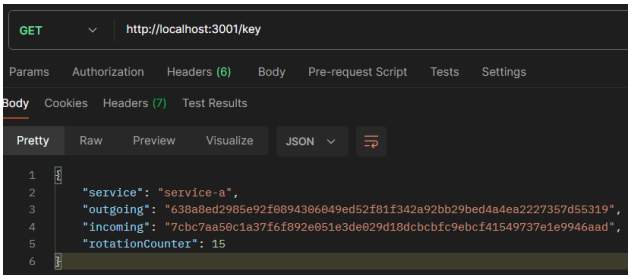
Berikut merupakan URL *service* yang digunakan selama pengujian. URL ini dimanfaatkan untuk melakukan pemanggilan endpoint sebagaimana didefinisikan pada Tabel 1.

Tabel 2
URL Pengujian (Lanjutan)

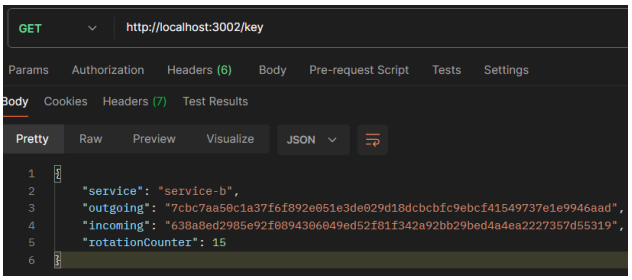
Service	URL
A	http://localhost:3001/
B	http://localhost:3002/

B. Hasil Pengujian

1. Pemerolehan Kunci



Gambar 9. Kunci Milik *Service* A pada Rotasi ke-15

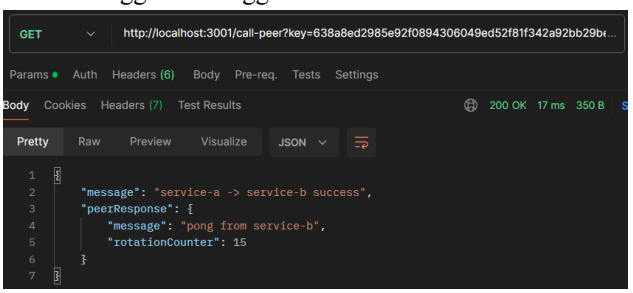


Gambar 10. Kunci Milik *Service* B pada Rotasi ke-15

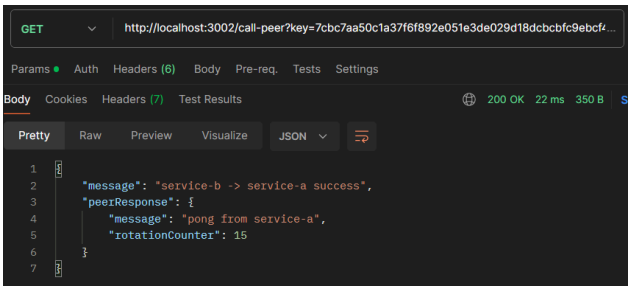
Gambar 9 dan 10 menunjukkan pemerolehan kunci menggunakan *endpoint* 'GET /key' pada Service A dan Service B pada rotasi ke-15, yaitu 15 menit setelah infrastruktur dijalankan (rotasi dilakukan setiap 60 detik untuk keperluan pengujian). Terlihat bahwa nilai *outgoing key* dan *incoming key* pada kedua *service* bersifat saling berkebalikan.

Kunci direpresentasikan dalam bentuk 64 karakter heksadesimal, yang setara dengan ukuran 256 bit. Nilai *key* tersebut akan digunakan sebagai parameter untuk pemanggilan endpoint 'GET /call-peer'

2. Pemanggilan Menggunakan Kunci Valid



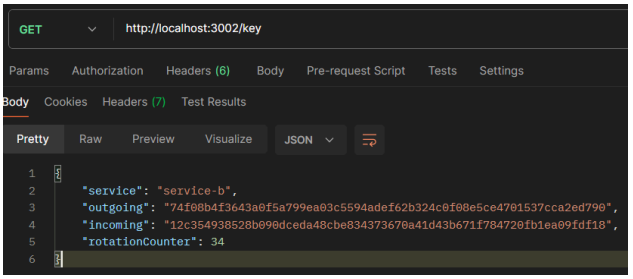
Gambar 11. Pemanggilan Sukses dari A ke B



Gambar 12. Pemanggilan Sukses dari B ke A

Pada Gambar 11 dan 12 terlihat bahwa komunikasi antara kedua service melalui pemanggilan *endpoint* 'GET /call-peer', yang memicu pemanggilan 'GET /ping', berhasil dilakukan karena *outgoing key* yang dikirim oleh *service* pemanggil memiliki nilai yang sama dengan *incoming key* pada service yang dipanggil.

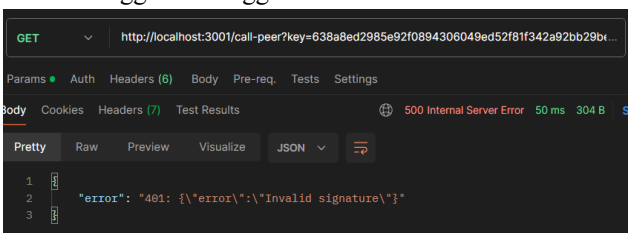
3. Rotasi Kunci



Gambar 13. Rotasi Kunci Pada *Service* B

Pada Gambar 13 terlihat bahwa *state* kunci pada *service* B telah mencapai rotasi ke-34 dan memiliki nilai yang berbeda signifikan dibandingkan dengan Gambar 10 (rotasi ke-15). Hal ini menunjukkan bahwa sistem berhasil melakukan mekanisme rotasi kunci.

4. Pemanggilan Menggunakan Kunci Kedaluwarsa

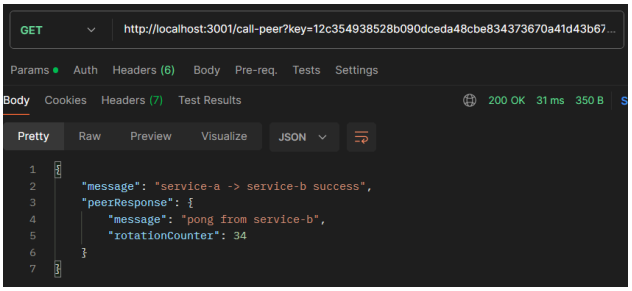


Gambar 14. Pemanggilan Gagal dari A ke B

Gambar 14 menunjukkan pemanggilan dari *service* A ke *service* B menggunakan kunci yang sama seperti pada Gambar 11. Pemanggilan tersebut gagal karena kunci

telah mengalami rotasi dan permintaan dilakukan menggunakan kunci yang sudah kedaluwarsa.

5. Pemanggilan Menggunakan Kunci Baru



Gambar 15. Pemanggilan Sukses dari A ke B Menggunakan Kunci Baru

Gambar 15 menunjukkan pemanggilan dari *service* A ke *service* B menggunakan kunci baru hasil rotasi, yang berhasil dilakukan.

A. Analisis Hasil Pengujian

Pengujian 1 berhasil membuktikan bahwa sistem mampu menghasilkan kunci yang identik dengan ukuran 256-bit. Hasil ini menunjukkan bahwa sistem telah menerapkan panjang kunci yang aman dan sesuai dengan rekomendasi standar NIST [16].

Selanjutnya, pada pengujian 2, berhasil dibuktikan bahwa setiap *service* dapat saling berkomunikasi secara aman dengan menggunakan kunci yang valid sesuai dengan periode rotasinya.

Keberhasilan mekanisme rotasi kunci dibuktikan pada pengujian 3, yang menunjukkan bahwa *state* kunci pada *service* B mengalami perubahan seiring dengan meningkatnya nilai *counter* rotasi. Kunci yang dihasilkan pada rotasi terbaru memiliki nilai yang berbeda secara signifikan dibandingkan dengan kunci pada rotasi sebelumnya, yang merupakan salah satu faktor penting dalam meningkatkan tingkat keamanan sistem.

Pengujian 4 menunjukkan bahwa pemanggilan *service* menggunakan kunci yang telah kedaluwarsa menghasilkan respons ‘error: *invalid signature’*. Hasil ini membuktikan bahwa sistem menerapkan mekanisme keamanan dengan menginvalidasi kunci dari rotasi sebelumnya, sehingga mencegah penggunaan kunci lama dalam proses komunikasi.

Pengujian 5 menunjukkan bahwa pemanggilan *service* menggunakan kunci baru yang sesuai dengan periode rotasinya berhasil dilakukan. Hasil ini membuktikan bahwa sistem mampu menghasilkan dan memverifikasi kunci hasil rotasi, sehingga komunikasi antar service dapat kembali berlangsung normal setelah proses rotasi kunci dilakukan.

Seluruh pengujian yang dilakukan membuktikan bahwa sistem memiliki tingkat keamanan yang baik terhadap serangan *brute force* kunci, karena waktu yang tersedia bagi penyerang untuk menebak kunci hanya 60 detik, sementara kunci yang digunakan berukuran 256-bit, sehingga secara komputasional sangat sulit untuk dipecahkan. Selain itu, mekanisme rotasi kunci secara berkala juga meningkatkan ketahanan sistem terhadap serangan *replay attack*, di mana penyerang berpotensi menangkap paket komunikasi dan mengirimkannya kembali ke *service*. Dengan adanya rotasi kunci, paket yang dikirim menggunakan kunci lama akan menjadi tidak valid.

Meskipun pada pengujian hanya melibatkan dua *service*, mekanisme ini dapat diaplikasikan untuk skenario dengan lebih banyak *service*. Penggunaan kunci *incoming* dan *outgoing* yang spesifik untuk setiap pasangan *service* menunjukkan bahwa sistem ini tidak hanya berfungsi sebagai mekanisme pengamanan komunikasi, tetapi juga dapat berperan sebagai mekanisme autentikasi dan nir-penyangkalan antar *service*, sehingga menambah lapisan keamanan secara keseluruhan pada sistem.

Terlepas dari hasil yang diperoleh, terdapat beberapa pengembangan yang dapat dilakukan ke depannya, salah satunya dengan mengimplementasikan mekanisme komunikasi berbasis gRPC guna membuat komunikasi antar service menjadi lebih ringan secara *overhead* dan memiliki latensi yang lebih rendah [17].

Kegagalan komunikasi juga berpotensi terjadi apabila pesan dikirim tepat sebelum proses rotasi kunci berlangsung, sehingga pada saat pesan diterima, kunci yang digunakan sudah tidak lagi valid. Untuk mengatasi permasalahan ini, salah satu solusi yang dapat diterapkan adalah dengan tetap menyimpan dan menerima kunci dari satu periode rotasi sebelumnya, sehingga toleransi terhadap perbedaan waktu pengiriman dan penerimaan pesan dapat ditingkatkan tanpa mengurangi keamanan sistem secara signifikan.

V. KESIMPULAN

Telah berhasil dirancang dan diimplementasikan sebuah mekanisme rotasi kunci berbasis ECDH dan HKDF yang mampu meningkatkan keamanan layanan web, khususnya dalam aspek autentikasi, otorisasi, dan nir-penyangkalan. Mekanisme ini terbukti efektif dalam meningkatkan ketahanan sistem terhadap serangan brute force kunci serta replay attack, sehingga mendukung komunikasi antar layanan web yang lebih aman.

REFERENSI

[1] H. M. Kienle and D. Distante, “Evolution of Web Systems,” Evolving Software Systems. Springer Berlin Heidelberg, pp. 201–228, Dec. 12, 2013. doi: 10.1007/978-3-642-45398-4_7.

[2] Aruna S, “Security in Web Services- Issues and Challenges,” IJERT, vol. V5, no. 09, Sept. 2016, doi: 10.17577/ijertv5is090245.

[3] X. F. Zhang, Y. Hou, and J. L. Ma, “Survey on the Web Services Security Specifications,” AMR, vol. 655–657, pp. 1809–1814, Jan. 2013, doi: 10.4028/www.scientific.net/amr.655-657.1809.

[4] F. Tanveer, F. Iradat, W. Iqbal, and A. Ahmad, “Towards Secure APIs: A Survey on RESTful API Vulnerability Detection,” CMC, vol. 84, no. 3, pp. 4223–4257, 2025, doi: 10.32604/cmc.2025.067536.

[5] M. Nadeem, A. Arshad, S. Riaz, S. W. Zahra, A. K. Dutta, and S. Almotairi, “A Secure Architecture to Protect the Network from Replay Attacks during Client-to-Client Data Transmission,” Applied Sciences, vol. 12, no. 16, p. 8143, Aug. 2022, doi: 10.3390/app12168143.

[6] N. Bindel, D. Stebila, and S. Veitch, “Improved attacks against key reuse in learning with errors key exchange,” *Cryptology ePrint Archive*, Report 2020/1288, 2020.
[7] V. Tanksale, “Efficient Elliptic Curve Diffie–Hellman Key Exchange for Resource-Constrained IoT Devices,” Electronics, vol. 13, no. 18, p. 3631, Sept. 2024, doi: 10.3390/electronics13183631.
[8] H. Krawczyk and P. Eronen, *HMAC-based Extract-and-Expand Key Derivation Function (HKDF)*, RFC 5869.
[9] E. Barker, “Recommendation for key management:,” National Institute of Standards and Technology, May 2020. doi: 10.6028/nist.sp.800-57pt1r5.
[10] B. Kaliski, *Diffie-Hellman Key Agreement Method*, RFC 2631, Internet Engineering Task Force (IETF), Jun. 1999.
[11] D. McGrew, M. Brown, dan S. Turner, *Fundamental Elliptic Curve Cryptography Algorithms*, RFC 6090, Internet Engineering Task Force (IETF), Feb. 2011.
[12] Y. salami, Y. Ebazadeh, and V. Khajehvand, “CE-SKE: cost-effective secure key exchange scheme in Fog Federation,” Iran J Comput Sci, vol. 4, no. 4, pp. 305–317, Apr. 2021, doi: 10.1007/s42044-021-00086-2.
[13] [1] J. Bao, “Research on the Security of Elliptic Curve Cryptography,” Advances in Economics, Business and Management Research, vol. 652. Atlantis Press, 2022. doi: 10.2991/aebmr.k.220405.164.
[14] H. Krawczyk, “Cryptographic Extraction and Key Derivation: The HKDF Scheme,” IACR Cryptology ePrint Archive, Report 2010/264, 2010.
[15] M. Bellare, R. Canetti, and H. Krawczyk, “Keying Hash Functions for Message Authentication,” RFC 2104, 1997.
[16] NIST, “Secure Hash Standard (SHS),” FIPS PUB 180-4, National Institute of Standards and Technology, Aug. 2015.
[17] M. Niswar et al., “Performance Evaluation of Microservices Communication with REST, GraphQL, and gRPC,” Int. J. Electron. Telecommun., vol. 70, no. 2, pp. –, Jun. 2024.

Pernyataan

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.

Bandung, 26 Desember 2025

Ahmad Rafi Maliki
13522137