JURNAL ILMU KOMPUTER
KHARISMA .TECH

Jurnal KHARISMA Tech | ISSN:1907-2317 | e-ISSN: 2810-0344
https://jurnal.kharisma.ac.id/kharismatech
published by: Pusat Penelitian STMIK KHARISMA Makassar

Volume: 19 no. 01 – Maret 2024- hlm. 64-74

PEMANFAATAN ENKRIPSI DATA BERBASIS ALGORITMA BLOWFISH PADA APLIKASI PASSWORD MANAGER REMEMBERME!

Oleh:

Christian Rayhan Suryanto[1], Mohammad Fajar[2*], Hasniati[3]
[1,2,3] Teknologi Informatika, STMIK Kharisma Makassar
e-mail: [1] christianrayhan_20@kharisma.ac.id, [2] fajar@kharisma.ac.id,
[3] hasniati@kharisma.ac.id

Abstrak: *Password manager RememberMe merupakan salah satu aplikasi yang dapat digunakan oleh penggunanya untuk mengelola dan melindungi kata sandi dari berbagai aplikasi di internet. Sebagai aplikasi yang mengelola data yang sangat penting, tentunya diperlukan tambahan lapisan keamanan di sisi data pengguna yang tersimpan di basis data. Oleh karena itu penelitian ini bertujuan untuk memanfaatkan algoritma blowfish dalam melakukan enkripsi data pengguna yang akan disimpan di basis data dan meng-evaluasi kinerja aplikasi. Pengumpulan data dilakukan dengan mengevaluasi proses enkripsi dan dekripsi serta observasi kinerja aplikasi password manager RememberMe! terhadap aspek waktu tanggap, ukuran data hasil enkripsi, dan pemakaian bandwitdth. Hasil evaluasi menunjukkan data pengguna khususnya password berhasil dienkrip kemudian disimpan ke basis data serta berhasil didekrip ketika digunakan kembali oleh pengguna. Selain itu pengujian waktu tanggap rata-rata berada dibawah 1 detik, ukuran data enkripsi relatif sebesar 22 byte dan total data bandwidth sent yaitu relatif kurang dari 5 MB.*

Kata kunci: *Enkripsi, Algoritma Blowfish, Password Manager, Aplikasi RememberMe, Firebase.*

Abstract: *RememberMe! Password manager is one of the applications that users can utilize to manage and secure passwords for various internet applications. As an application that handles highly sensitive data, an additional layer of security on the user data stored in the database is essential. Therefore, this research aims to utilize the Blowfish algorithm to encrypt user data that will be stored in the database and evaluate the application's performance. Data collection is conducted by evaluating the encryption and decryption processes and observing the performance of the RememberMe! Password manager application in terms of response time, size of encrypted data, and bandwidth usage. The evaluation results indicate that user data, especially passwords, were successfully encrypted and stored in the database, and they were successfully decrypted when used by the user. Furthermore, the average response time testing was below 1 second, the size of the encrypted data was relatively 22 bytes, and the total data bandwidth sent was relatively less than 5 MB.*

Keywords: *Encryption, Blowfish Algorithm, Password Manager, RememberMe Application, Firebase.*

1. PENDAHULUAN

RememberMe! merupakan aplikasi password manager mobile yang dibuat untuk memastikan pengguna dapat dengan mudah mengelola kata sandi pada berbagai jenis akun yang dimiliki. RememberMe! juga dapat diakses melalui website dari situs (https://byenigma.com/). Data aplikasi RememberMe! disimpan pada basis data firebase

* Corresponding author : Mohammad Fajar (fajar@kharisma.ac.id)

Diterima : *Agustus, 2023* ***Disetujui :*** *September, 2023* ***Dipublikasikan :*** *Maret, 2024*

melalui koneksi internet. Mempertimbangkan pentingnya data yang diolah dan disimpan tersebut, maka diperlukan tambahan lapisan keamanan di sisi penyimpanan data ke basis data. Salah satu mekanisme yang dipertimbangkan yaitu enkripsi dan dekripsi data.

Menurut Putra [1], enkripsi adalah suatu metode yang digunakan untuk mengkodekan data sedemikian rupa sehingga keamanan informasinya terjaga dan tidak dapat dibaca tanpa di dekripsi (kebalikan dari proses enkripsi) dahulu. Secara teknis, enkripsi adalah proses mengubah teks yang dapat dibaca manusia menjadi teks yang tidak dapat dipahami. Teks yang dienkripsi disebut ciphertext. Menurut Muklas [2], dekripsi, kebalikan dari enkripsi, pesan yang dienkripsi kembali ke bentuk awal. Tujuan enkripsi adalah untuk mencegah pihak ketiga mengetahui apa yang sedang dikomunikasikan atau di diskusikan. Saat pihak ketiga yang mencoba menyusup ke dalam sistem yang mereka lihat hanyalah teks acak dan tidak dapat dipahami.

Menurut Hairullah [3], Bruce Schneier merancang algoritma blowfish pada tahun 1993. Sejak dicetus, algoritma ini telah dianalisa terus menerus, dan perlahan diakui sebagai algoritma enkripsi yang handal. Menurut Wahyudi [4], blowfish merupakan algoritma yang tidak dipatenkan, dan tersedia secara gratis untuk berbagai macam kegunaan. Menurut Bruce [5], Algoritma blowfish adalah alternatif enkripsi data yang cepat dan terbuka (open-source). Menurut Annas [6], blowfish dikembangkan guna mencukupi standar desain yang cepat dengan pelaksanaanya untuk kondisi yang ideal dapat mencapai 26 clock cycle per-byte, mudahnya pada algoritma dapat diketahui masalahnya, dan keamanan faktor dari panjang kunci bervariasi (minimnya 32 bit, maksimalnya 448 bit, multiple 8 bit, default 128 bit). Menurut Barneci [7] blowfish pada strategi implementasi yang tepat akan lebih optimal, dapat berjalan pada memori kurang dari 5 KB dan kesederhanaan pada algoritmanya. Menurut Sebastian [8] Banyak kelebihan dari algoritma blowfish seperti kompatibilitas dan efisiensi dalam penerapannya dan tidak ada lisensi yang diperlukan. Menurut Andy [9], Java merupakan salah satu bahasa pemrograman yang wajib untuk dimiliki, karena merupakan pondasi awal untuk dapat memahami dan mempelajari pemrograman tingkat lanjut terutama di bidang pengembangan aplikasi serta desain perangkat lunak. Java adalah Bahasa pemrograman yang digunakan dalam membuat aplikasi password manager RememberMe!.

Perumusan masalah dalam penelitian ini adalah bagaimana memanfaatkan algoritma blowfish untuk proses enkripsi dan dekripsi data yang dikelola oleh aplikasi password manager RememberMe, serta bagaimana kinerja aplikasi tersebut dalam hal *response time*, ukuran data enkripsi, dan jumlah total data bandwidth yang dikirim. Tujuan penelitian ini adalah untuk memanfaatkan algoritma blowfish untuk proses enkripsi dan dekripsi pada data yang dikelola oleh aplikasi password manager RememberMe!, sekaligus mengevaluasi kinerja aplikasi password manager RememberMe! pada aspek *response time*, ukuran data enkripsi dan jumlah total data *bandwidth sent*.

2. METODE PENELITIAN

Penelitian ini menggunakan jenis data kuantitatif. Data kuantitatif yang diperoleh mencakup data response time, ukuran data enkripsi, dan total data bandwidth sent. Dalam penelitian ini, sumber data yaitu primer yang dikumpulkan melalui evaluasi sistem.

Penelitian ini melibatkan penggunaan password manager RememberMe! sebagai objek observasi. Observasi password manager RememberMe! dilakukan untuk mengevaluasi response time, serta database firebase digunakan untuk mengamati dan mengukur ukuran data enkripsi dan total data bandwidth sent.

Dalam penelitian ini, visualisasi data digunakan sebagai metode untuk memvisualisasikan data terkait response time, ukuran data enkripsi, dan jumlah total data bandwidth sent. Visualisasi data disajikan menggunakan grafik, tabel, dan gambar.

Tahapan penelitian terbagi menjadi empat tahap yaitu sebagai berikut :

1. Tahap pertama adalah tinjauan literatur, di mana dilakukan studi literatur terkait password manager, algoritma blowfish, firebase, dan metode evaluasi yang berkaitan dengan response time, ukuran data enkripsi, dan total data bandwidth sent. Selanjutnya, tahap penentuan metodologi melibatkan penentuan metode pengumpulan data, seperti menggunakan aplikasi password manager RememberMe!. Penentuan parameter evaluasi yang akan digunakan seperti response time, ukuran data enkripsi, dan total data bandwidth sent. Alat dan teknik yang akan digunakan untuk pengumpulan dan pengolahan data juga ditentukan.
2. Tahap pengumpulan data yaitu mengobservasi password manager RememberMe! dengan mengumpulkan data pengujian kecepatan response time yang dilakukan sebanyak sembilan puluh kali untuk mengevaluasi response time, serta database firebase digunakan untuk mengamati dan mengukur ukuran data enkripsi dan total data bandwidth sent.
3. Setelah data terkumpul, tahap pengolahan data dilakukan dengan memvisualisasikan data dengan menggunakan grafik, tabel, dan gambar.
4. Selanjutnya, tahap evaluasi dan interpretasi dilakukan dengan menafsirkan hasil pengolahan data dan menghubungkannya dengan tujuan penelitian. Kesimpulan tentang kinerja aplikasi password manager dalam hal response time, ukuran data enkripsi, dan total data bandwidth sent juga dibuat dalam tahap ini.

3. HASIL DAN PEMBAHASAN

3.1 Spesifikasi kebutuhan sistem

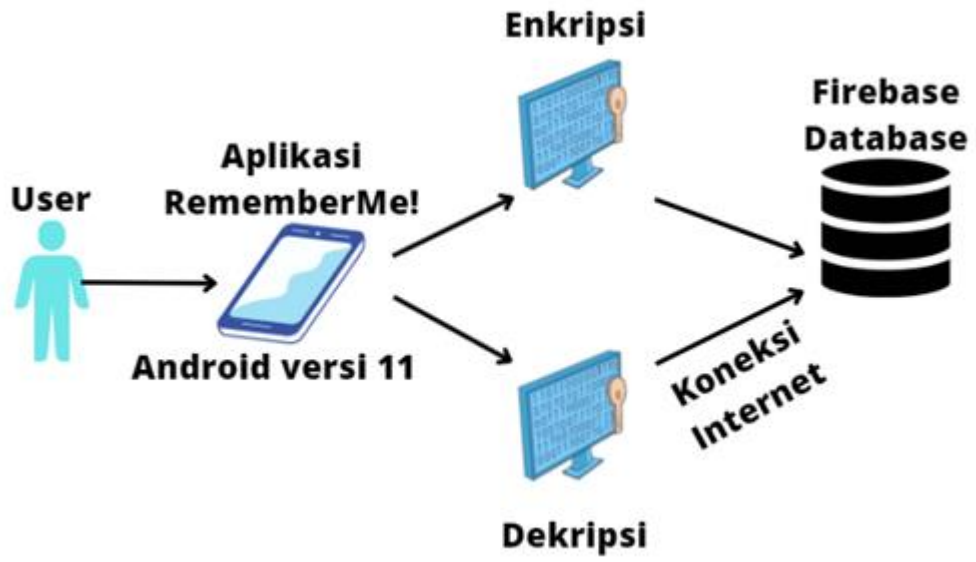
Diagram use case pada Gambar 1 yang memperlihatkan spesifikasi kebutuhan sistem aplikasi password manager RememberMe! yang memanfaatkan mekanisme enkripsi dan dekripsi data. Saat pengguna menginput data, seperti kata sandi dan informasi login lainnya, setelah pengguna menekan tombol simpan data maka aplikasi akan mengenkripsi data tersebut sebelum menyimpannya di Firebase. Ketika pengguna ingin mengakses kembali data yang telah diinput, aplikasi RememberMe! akan melakukan proses dekripsi untuk mengembalikan data asli. Firebase akan membaca data yang telah didekripsi ini dan pengguna dapat mengakses kembali data yang telah diinput.

Gambar 1. Diagram Usecase Password Manager RememberMe!

3.2 Arsitektur sistem enkripsi dan dekripsi data

Arsitektur aplikasi RememberMe! pada Gambar 2 menggambarkan langkah-langkah dari penggunaan hingga proses enkripsi dan dekripsi data. Aplikasi ini dapat diakses oleh pengguna melalui perangkat android. Ketika pengguna ingin menyimpan data, aplikasi melakukan proses enkripsi terlebih dahulu, sehingga data terjamin keamanannya sebelum disimpan. Data yang telah dienkripsi kemudian disimpan dalam firebase database menggunakan koneksi internet. Ketika data perlu diambil kembali, aplikasi mengambil data yang telah dienkripsi dari firebase database dan

menjalani proses dekripsi untuk mengembalikannya ke bentuk semula. Dengan demikian, aplikasi RememberMe! dapat memberikan perlindungan keamanan data bagi pengguna melalui enkripsi dan dekripsi yang efisien dan handal.



Gambar 2. Arsitektur Sistem Enkripsi dan Dekripsi Password Manager RememberMe!

3.3 Analisis proses enkripsi dan dekripsi

Proses berikut adalah penjelasan mengenai enkripsi dan dekripsi menggunakan contoh 1 password (mcbn123HJKL). Sebelum menjelaskan proses enkripsi dan dekripsi, password mcbn123HJKL memiliki ukuran 11 byte sedangkan saat melakukan proses enkripsi dan dekripsi ukuran password harus berada dalam kelipatan 8. Oleh karena itu ukuran password diubah dengan melakukan padding:

1. Hitung ukuran padding
   Ukuran padding = block_size - (panjang_data modulo block_size)
   Ukuran padding = 8 - (11 modulo 8)
   Ukuran padding = 8 - 3
   Ukuran padding = 5
2. Ukuran padding dalam bentuk heksadesimal, setiap digit heksadesimal mewakili setengah byte. Oleh karena itu, dua digit heksadesimal mewakili 1 byte data= 0x05.
3. Gabungkan data dengan blok padding:
   Data setelah padding: "mcbn123HJKL" + blok padding
   Hasil: "mcbn123HJKL\x05\x05\x05\x05\x05" (16 byte).

   Langkah-langkah proses enkripsi dijelaskan dibawah ini :
   1. Pertama persiapkan teks asli (plaintext) yang digunakan yaitu “mcbn123HJKL“

2. Jadwal kunci menggunakan kunci enkripsi yang dipersiapkan oleh algoritma blowfish.
3. Bagi teks asli (plaintext) menjadi blok-blok dengan panjang 64 bit (8 byte)
4. Blok plaintext: " mcbn123H " (8 byte)
5. Kemudian lakukan operasi XOR antara blok plaintext dengan blok kunci
6. Blok plaintext: " mcbn123H " (8 byte) - KHARISMA
7. Jalankan 16 putaran enkripsi pada blok plaintext menggunakan jadwal kunci. Jadwal kunci hanya diketahui oleh algoritma blowfish sendiri.
8. Setelah semua blok plaintext dienkripsi, hasil enkripsi menjadi ciphertext.
9. Hasil enkripsi (ciphertext) = " 558rlcgC5f4PAfSTsFFyXQ "

Langkah-langkah proses dekripsi dijelaskan sebagai berikut :
Hasil enkripsi (ciphertext) = " 558rlcgC5f4PAfSTsFFyXQ " (22 byte)
Karena ukuran chipertext bukan dalam kelipatan 8 perlu dilakukan padding

1. Hitung ukuran padding
   Ukuran padding = block_size - (panjang_data modulo block_size)
   Ukuran padding = 8 - (22 modulo 8)
   Ukuran padding = 8 - 6
   Ukuran padding = 2
2. Blok padding dalam bentuk heksadesimal, setiap digit heksadesimal mewakili setengah byte. Oleh karena itu, dua digit heksadesimal mewakili 1 byte data= 0x02.
3. Gabungkan data dengan blok padding:
   Data setelah padding: "558rlcgC5f4PAfSTsFFyXQ " + blok padding
   Hasil: "558rlcgC5f4PAfSTsFFyXQ \x02\x02 " (24 byte).
   Karena ukurannya sudah dalam kelipatan 8 sekarang saatnya dimulai proses dekripsi :
   1. Pertama persiapkan hasil enkripsi (chipertext) yang digunakan yaitu " 558rlcgC5f4PAfSTsFFyXQ \x02\x02 " (24 byte)
   2. Jadwal kunci menggunakan kunci dekripsi yang dipersiapkan oleh algoritma blowfish.
   3. Bagi hasil enkripsi (chipertext) menjadi blok-blok dengan panjang 64 bit (8 byte)
   4. Blok chipertext: " 558rlcgC " (8 byte)
   5. Kemudian lakukan operasi XOR antara blok chipertext dengan blok kunci
   6. Blok chipertext: " 558rlcgC " (8 byte) - KHARISMA
   7. Jalankan 16 putaran dekripsi pada blok chipertext menggunakan jadwal kunci. Jadwal kunci hanya diketahui oleh algoritma blowfish sendiri.
   8. Setelah semua blok chipertext didekripsi, hasil dekripsi menjadi plaintext.
   9. Hasil dekripsi (plaintext) = " mcbn123HJKL "

3.4 Implementasi

Untuk memanfaatkan mekanisme enkripsi dan dekripsi data menggunakan algoritma blowfish pada penelitian ini, digunakan paket dan class Java yang telah tersedia, diantaranya: class javax.crypto dan class android.util. Berikut ini ditampilkan kode enkripsi dan kode dekripsi. Untuk proses enkripsi dibuatkan dalam metode enkripsi, sementara untuk proses dekripsi dibuatkan dalam metode dekripsi.

3.4.1 Kode Enkripsi :

```
public String enkripsi(String myText, String key) {
    try {
        SecretKeySpec KS = new SecretKeySpec(key.getBytes(), "Blowfish");
        Cipher cipher = Cipher.getInstance("Blowfish");
        cipher.init(Cipher.ENCRYPT_MODE, KS);
        byte[] encrypted = cipher.doFinal(myText.getBytes());
        return Base64.encodeToString(encrypted, Base64.NO_PADDING);
    } catch (Exception e) {
        return "ERROR:" + e.getMessage();
    }
}
```

3.4.2 Kode Dekripsi :

```
public static String dekripsi(String chiperText, String key) {
    try {
        SecretKeySpec KS = new SecretKeySpec(key.getBytes(), "Blowfish");
        Cipher cipher = Cipher.getInstance("Blowfish");
        cipher.init(Cipher.DECRYPT_MODE, KS);
        byte[] decrypted = cipher.doFinal(Base64.decode(chiperText,
Base64.NO_CLOSE));
        return new String(decrypted);
    } catch (Exception e) {
        return "ERROR";
    }
}
```

3.5 Pengujian

Pengujian dilakukan dengan mengumpulkan data pengujian kecepatan response time yang dilakukan sebanyak sembilan puluh kali untuk mengevaluasi response time, lalu database firebase digunakan untuk mengamati dan mengukur ukuran data enkripsi dan total data bandwidth sent, serta mengolah data dilakukan

dengan memvisualisasikan data dengan menggunakan grafik, tabel, dan gambar. Pengujian dilakukan selama dua hari, pengujian dilakukan sebanyak sembilan kali.

3.5.1 Data ukuran data enkripsi

Data pada Tabel 1 mencatat ukuran data yang dihasilkan setelah proses enkripsi pada setiap entri kata sandi dalam aplikasi password manager. Ukuran data enkripsi diukur dalam satuan byte. Berdasarkan pada pengujian data ukuran enkripsi dapat diperoleh bahwa ukuran data chipertext relatif 22 byte. Ukuran chipertext tersebut dapat dikatakan rendah karena ukurannya masih relatif kecil dalam konteks enkripsi, serta menunjukkan efisiensi dalam penggunaan ruang penyimpanan. Hal ini menunjukkan bahwa algoritma blowfish mampu menghasilkan data enkripsi dengan ukuran yang efisien. Sehingga memungkinkan penggunaan ruang penyimpanan yang lebih efektif pada perangkat.

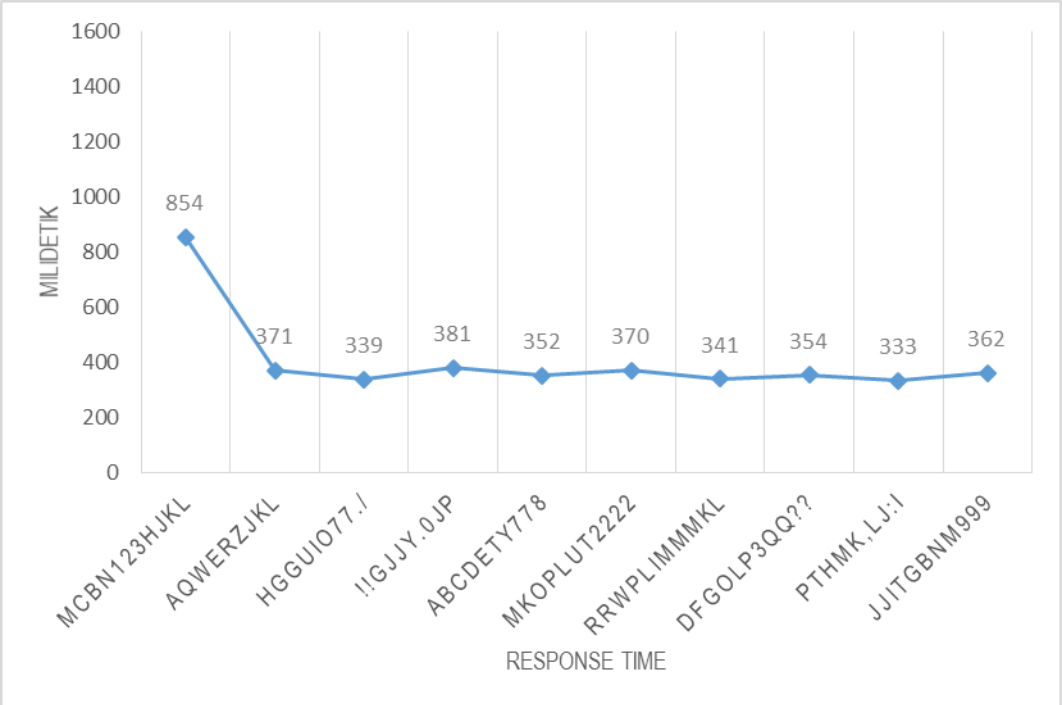
Tabel 1: Tabel Password Sebelum dan Setelah Enkripsi Serta Ukuran Data Enkripsi

No	Sebelum Enkripsi	Setelah Enkripsi
1	Password : mcbn123HJKL	Password : 558rlcgC5f4PAfSTsFFyXQ Size : 22 byte
2	Password : aqwerzjkl	Password : gVoLtgbzqciG2j1TFRYQ5w Size : 22 byte
3	Password : Hgguio77./	Password : hdaZbcf77o61VCQKKKkLpA Size : 22 byte
4	Password : !!gjjy.0jp	Password :7esRkdB8ohcQTWBaHE41/Q Size : 22 byte
5	Password : abcdety778	Password : yie2nijp0LLMkaLF3ROpqA Size : 22 byte
6	Password : mkoplut2222	Password : n/h6qEVXd2g+s6upCqwgEA Size : 22 byte
7	Password : RRwplimmmkl	Password : zF9Nb3v9LJtJBZV4pG6FRg Size : 22 byte
8	Password : dfgolp3qq??	Password : jvQhX6HqyM5mVV8WZXZnRg Size : 22 byte
9	Password : pthmk,lj:i	Password : uihNJKKA67SxZH+c8RdPyg Size : 22 byte
10	Password : jjitgbnm999	Password : oP+MLrBBKbxXbcEy9bPeXQ Size : 22 byte

3.5.2 Response time

Data response time pada Gambar 3 mencatat waktu yang dibutuhkan oleh password manager untuk mengenkripsi kata sandi sampai datanya tersimpan ke firebase. Response time diukur dalam satuan milidetik (ms) dan direkam untuk setiap operasi yang dilakukan. Berdasarkan pada grafik response time dapat diperoleh bahwa pada pertama kali pengujian response time cukup tinggi dan pada pengujian berikutnya menurun menjadi relatif 300 milidetik saat melakukan proses enkripsi kata sandi. Namun, dapat dilihat bahwa dari semua pengujian waktunya cenderung kurang dari 1

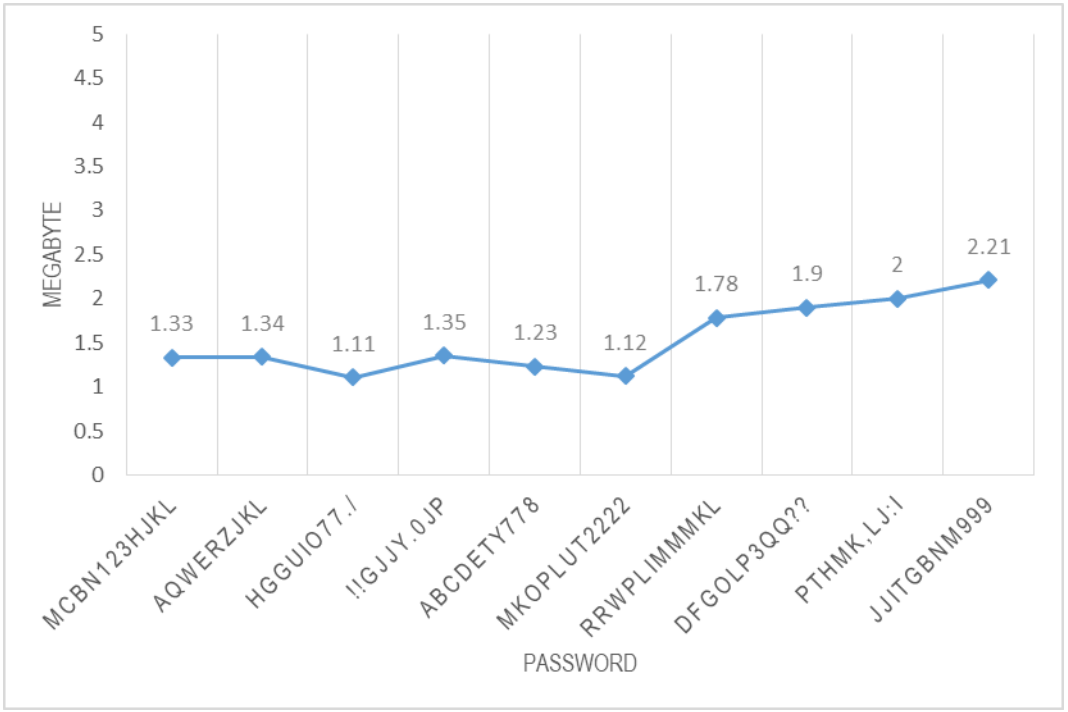
detik, hal ini menunjukkan algoritma blowfish mampu menghasilkan response time yang relatif cepat saat melakukan operasi penyimpanan entri kata sandi.



Gambar 3. Grafik yang Menunjukkan Hasil Pengujian Kecepatan Response Time 10 Kali Dalam Satuan Milidetik

3.5.3 Total data bandwidth sent

Data pada Gambar 4 ini mencakup segala jenis komunikasi antara aplikasi dan firebase, seperti pengiriman notifikasi pada pengguna, dan mengunggah file ke dalam aplikasi. Dalam melakukan hal tersebut data bandwidth sent di firebase dapat bertambah. Berdasarkan pada pengujian total data bandwidth sent dapat diperoleh bahwa, total data bandwidth sent dari 40 kali pengujian adalah 62,26 megabyte (MB) dan sebagian besar pengujian menghasilkan data relatif di bawah 5 MB, jadi bisa diperlihatkan bahwa password manager ini memiliki kebutuhan akan bandwidth yang relatif rendah. Hal ini menunjukkan bahwa algoritma blowfish dapat membantu mengoptimalkan penggunaan bandwidth secara efisien.



Gambar 4. Grafik yang Menunjukkan Total Data Bandwidth Sent dan Terkirim Dalam 10 Kali Percobaan

4. KESIMPULAN

Aplikasi RememberMe! memanfaatkan algoritma blowfish untuk melakukan enkripsi dan dekripsi data password pengguna. Berdasarkan evaluasi kinerja yang dilakukan, disimpulkan bahwa :

1. Pengujian waktu response time cenderung berada dibawah 1 detik.
2. Ukuran data enkripsi relatif sebesar 22 byte.
3. Total data bandwidth sent adalah 62,26 megabyte (MB) dan dari 40 kali pengujian menghasilkan data relatif kurang dari 5 MB.

Hal ini menunjukkan algoritma blowfish mampu menghasilkan response time yang relatif cepat saat melakukan operasi penyimpanan entri kata sandi, data enkripsi dengan ukuran yang efisien memungkinkan penggunaan ruang penyimpanan yang lebih efektif pada perangkat, dapat membantu mengoptimalkan penggunaan bandwidth secara efisien. Selain mengukur response time, ukuran data enkripsi, dan jumlah total data bandwidth sent saat operasi penyimpanan entri kata sandi, disarankan untuk mempertimbangkan variasi skenario pengujian tambahan. Misalnya, menguji kinerja password manager saat proses autentikasi pengguna, pembaruan entri kata sandi, atau operasi pencarian. Hal ini akan memberikan gambaran yang lebih lengkap tentang kinerja password manager dalam berbagai situasi penggunaan.

DAFTAR PUSTAKA

[1] Putra R, "IMPLEMENTASI PENGAMANAN BASIS DATA DENGAN TEKNIK ENKRIPSI," Jurnal Sistem Informasi, hlm. 413-419, 2020, [Daring]. Available: https://media.neliti.com/media/publications/455362-none-aacb775c.pdf.

[2] Muklas P, Dadang M, Runi A, dan Mirsandi, "PerancanganAplikasi Enkripsi & Deskripsi pada Dokumen File Dengan Algoritma Triple DES Berbasis Web," Jurnal Pendidikan Sains dan Komputer, vol. 2, no. 1, hlm. 57-69, 2022 https://jurnal.itscience.org/index.php/jpsk/article/view/1354.

[3] Hairullah, Pramartha C, dan Putra I A G S, "Aplikasi Keamanan E-Commerce Berbasis Web Menggunakan Metode Algoritma Blowfish," vol. 01, no. 01, hlm. 79-88, 2022. [Daring]. Tersedia pada: https://www.researchgate.net/publication/367282967

[4] Wahyudi, Laksito W, dan Prabowo I A, "The Application of the Blowfish Algorithm and the Least Significant Bit Method for Securing Student Transcripts," Jurnal Ilmiah SINUS, vol. 20, no. 2, hlm. 87–95, 2022. Tersedia pada: https://p3m.sinus.ac.id/jurnal/index.php/e-jurnal_SINUS/article/download/622/pdf

[5] Marsel F I, Hafiz A A, Afriliansyah, Febrian M A, dan Hasan M A, "PENERAPAN ALGORITMA KRIPTOGRAFI ASIMETRIS DENGAN METODE RSA DAN BLOWFISH UNTUK ENKRIPSI DAN DESKRIPSI GAMBAR MENGGUNAKAN JAVA NETBEANS," vol. 03, no. 02, hlm. 87-96, 2020. Tersedia pada: https://osf.io/srd9a

[6] Rifa A dan Cucu S L, "IMPLEMENTASI KRIPTOGRAFI MENGGUNAKAN METODE BLOWFISH DAN BASE 64 UNTUK MENGAMANKAN DATABASE INFORMASI AKADEMIK PADA KAMPUS AKADEMI TELEKOMUNIKASI BOGOR BERBASIS WEB-BASED," 2019. Diakses: 3 Juli 2023. [Daring]. Tersedia pada: https://www.jurnal.politeknik-kebumen.ac.id/E-KOMTEK/article/download/133/92

[7] Nuboba B H, Putra I G N A C, dan Suhartana I K G, "Rancang Bangun Aplikasi Enkripsi dan Dekripsi Objek 3 Dimensi menggunakan Algoritma Blowfish," Jurnal Elektronik Ilmu Komputer Udayana, vol. 08, no. 03, hlm. 307–316, 2020, Diakses: 7 Juli 2023. [Daring]. Tersedia pada: https://ojs.unud.ac.id/index.php/JLK/article/view/58097/33959

[8] Suhandinata S, Rizal R A, Wijaya D O, Warren P, dan Srinjiwi, "ANALISIS PERFORMA KRIPTOGRAFI HYBRID ALGORITMA BLOWFISH DAN ALGORITMA RSA," JURTEKSI (Jurnal Teknologi dan Sistem Informasi), vol. 6, no. 1, hlm. 1–10, Des 2019. [Daring]. Tersedia pada: https://jurnal.stmikroyal.ac.id/index.php/jurteksi/article/view/395.

[9] Andy D, Jelman N, dan Hari S, "PELATIHAN BAHASA PEMROGRAMAN JAVA TINGKAT LANJUT UNTUK SISWA SMA/SMK DI JAKARTA," vol. 01, no. 01, hlm. 7-12, 2023, [Daring]. Available: https://ejurnal.swadharma.ac.id/index.php/swadimas/article/view/239/195.