Jurnal Informatika Terpadu

https://journal.nurulfikri.ac.id/index.php/JIT
ISSN ONLINE : 2460-8998

ANALISIS KRIPTOGRAFI ALGORITMA BLOWFISH PADA KEAMANAN DATA MENGGUNAKAN DART

Yoga Pratama[1], Tata Sutabri[2]

[1,2]Teknik Informatika, Universitas Bina Darma
Palembang, Sumatera Selatan, Indonesia 30111
yogapratama.sc@gmail.com, tata.sutabri@gmail.com

Abstract

Data security is a vital aspect of information technology to maintain the integrity and authenticity of data from unauthorized access. In supporting this, an approach is taken to protect sensitive information and data. In this research, cryptographic techniques were analyzed and tested to secure data in text, which is commonly used as a password using the Blowfish cryptographic algorithm. The goal is to determine how strong, secure, and fast the blowfish algorithm maintains and executes the text data encoding. The research method used is an in-depth theoretical analysis followed by computational experiments using dart programming. Based on the study and testing carried out, the results of this research show that the use of cryptographic techniques with the blowfish algorithm on text data samples can be used as an alternative for securing data. The results show that the blowfish algorithm pattern has security because it uses various techniques to change and present data, supported by an execution time of less than 10 seconds and minimal memory usage of under 5 MB. Therefore, the blowfish algorithm can be used as a cryptographic choice in developing robust and fast text-based data security.

Keywords: *Algorithm, Blowfish, Cryptography, Dart, Text*

Abstrak

Keamanan data merupakan aspek vital dalam teknologi informasi untuk menjaga integritas dan keaslian data dari sebuah akses yang tidak sah. Dalam menjaga hal tersebut, dilakukan pendekatan untuk melindungi informasi sensitif sebuah data. Pada penelitian ini dilakukan analisis dan pengujian terhadap teknik kriptografi untuk mengamankan data berupa teks yang umum dipakai sebagai *password* dengan kriptografi algoritma *blowfish*. Tujuannya adalah untuk menguji seberapa kuat, aman, dan cepat algoritma blowfish menjaga dan mengeksekusi penyandian data teks tersebut. Metode penelitian yang digunakan adalah analisis teoritis secara mendalam dan dilanjutkan dengan eksperimen komputasi menggunakan pemrograman *dart*. Berdasarkan analisis dan pengujian yang dilakukan, hasil penelitian ini menunjukkan bahwa penggunaan teknik kriptografi dengan algoritma *blowfish* pada sampel data teks dapat dijadikan sebagai alternatif dalam mengamankan data. Didapatkan hasil, pola algoritma *blowfish* memiliki keamanan karena dilakukan berbagai teknik untuk mengubah dan menyajikan data, didukung dengan ukuran waktu eksekusi kurang dari 10 detik serta penggunaan memori yang sangat kecil di bawah 5 MB. Oleh karena itu, algoritma *blowfish* dapat dijadikan sebagai salah pilihan kriptografi dalam mengembangkan keamanan data berbasis teks yang kuat dan cepat.

Kata kunci: Algoritma, *Blowfish*, *Dart*, Kriptography, Teks

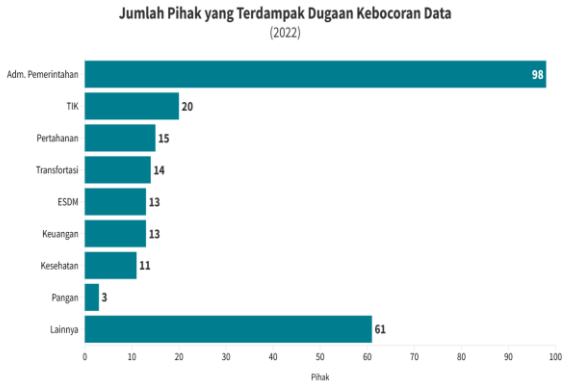
1. PENDAHULUAN

Terbuka dan mudahnya informasi dalam era modern memberikan berbagai akses dan informasi. Teknologi informasi sudah banyak diterapkan di banyak tempat, baik bisnis maupun pelayanan dari berbagai macam institusi maupun instansi lembaga[1]. Seiring dengan itu, pada era digital sekarang ini, isu keamanan data dalam informasi yang tersimpan pada *database* menjadi hal yang penting dan sensitif dalam penerapan sebuah teknologi dan sistem informasi[2]. Laju teknologi yang semakin pesat juga berdampak pada risiko ancaman terhadap keamanan data yang ada di dalam kegiatan terkomputerisasi[3]. Penyebabnya dipicu karena data dapat dijadikan senjata dan disalahgunakan oleh mereka yang tidak memiliki otoritas, wewenang, dan tanggung jawab. Pada banyak kasus yang berkaitan dengan pencurian data yang terjadi, banyak sekali kerugian materiil dan non-materiil.

Pihak yang tidak bertanggung jawab tersebut juga dapat menggunakan data sebagai alat untuk melakukan tindak

Diterima 24 Agustus 2023, Direvisi 11 September 2023, Diterima untuk publikasi 18 September 2023

kriminal yang mengarah pada pemerasan terhadap para korbannya. Pengiriman dan penerimaan dalam proses transfer data juga harus terjaga orisinalitas dan dimodifikasi sedemikian rupa untuk mengantisipasi dan melindungi dari kemungkinan bocornya jalur akses pada pengiriman data atau antisipasi hal paling buruk yakni runtuhnya sebuah sistem penyimpanan data. Data dugaan kebocoran data tersebut dapat kita lihat pada Gambar 1 berikut.



Gambar 1. Data Dugaan Kebocoran Data[4]

Pada teknik pengamanan data umumnya dikenal berbagai macam teknik, salah satunya kriptografi. Ini merupakan sebuah disiplin ilmu komputer yang mempelajari teknik untuk melindungi kerahasiaan, integritas, dan autentikasi data[5]. Kriptografi bertugas menjaga keamanan sebuah data menggunakan metode penyisipan data dengan suatu algoritma khusus dengan tujuan data tersebut terlindungi dari pengguna lain. Teknik ini selain mengubah data, dapat juga dipakai untuk melindungi integritas, keaslian serta nirsangkal[6]. Maka kita dapat menarik pernyataan bahwa kriptografi adalah sesuatu yang vital dan penting karena digunakan sebagai upaya mendukung keamanan dan menjaga rahasia pada informasi[7].

Penerapan teknik kriptografi dapat kita temukan pada banyak penelitian lain yang telah dilakukan sebelumnya, khususnya yang menggunakan algoritma *blowfish*. Misalnya pada penelitian yang dilakukan Simanullang dan Silalahi dalam penelitian yang berjudul Algoritma *Blowfish* Untuk Meningkatkan Keamanan *Database MySql*, yang diketahui bahwa penerapan algoritma *blowfish* pada *database MySql* dapat dilakukan peningkatan. Selain itu pada penelitian lain yang berjudul Algoritma *blowfish* pada *Watermarking* Video Digital oleh Khairani dan Nurwulan dengan menggunakan alat bantu aplikasi *Visual Basic*, didapatkan jika penggunaan metode algoritma *blowfish* dapat diimplementasikan dengan baik untuk *watermarking* video digital dengan tingkat keamanan yang sangat baik. Penelitian berkaitan dengan algoritma ini banyak menggunakan alat bantu aplikasi *Visual Basic*, seperti pada penelitian lain yang berjudul Implementasi Teknik Enkripsi dan Dekripsi di File Video Menggunakan Algoritma *Blowfish* yang dilakukan oleh Fahriani dan Rosyid. Diketahui bahwa dengan memanfaatkan algoritma ini implementasi teknik enkripsi berhasil dilakukan yang mengubah file asli agar tidak dapat dibaca.

Dalam penelitian ini penulis akan menggunakan algoritma *blowfish* yang menjadi salah satu alternatif pada teknik kriptografi untuk melindungi data yang bersifat teks sebagai sampel yang diasumsikan sebagai sebuah data sandi atau *password*. Peneliti akan menggunakan bahasa pemrograman *dart* untuk menguji kinerja algoritma *blowfish.* Selain itu di pilihan bahasa pemrograman *dart* karena jika kita melihat berbagai sumber informasi ilmiah, peneliti belum menemukan algoritma ini diterapkan dan diuji coba dengan basis bahasa *dart.* Ini akan menjadi tantangan tersendiri bagi penulis untuk membaca pola dan memahami alur dari konsep algoritma *blowfish* agar bisa diterapkan pada bahasa *dart.*

Kemudian, selain itu pemilihan terhadap algoritma *blowfish* juga didasari karena jika kita melakukan studi literatur, kita akan menemukan banyak kelebihan dari algoritma *blowfish* ini seperti kompatibilitas, efisiensi baik waktu ataupun sumber daya, dan tidak ada lisensi yang diperlukan untuk menggunakannya sehingga legal untuk dipergunakan serta hanya pemilik yang akan mengetahui isi dari *file* yang dimodifikasi dengan teknik ini. Algoritma *blowfish* menerapkan *simetrics key*, *chipher block*, *fistel network* dan *s-box* pada pemrosesan enkripsi dan dekripsi[8]. Lalu kemudian pada domain dan area ilmu kriptografi, algoritma ini banyak dipergunakan karena algoritma *blowfish* kuat, cepat, serta tidak terbentur terhadap lisensi[9]. Untuk itu, dalam dunia kriptografi algoritma *blowfish* menjadi salah satu pilihan yang banyak dipakai oleh pengembang perangkat lunak yang berkutat dan menginginkan algoritma kriptografi yang bersifat terbuka, kuat, dan cepat[10].

Selain itu secara keamanan, hingga saat ini celah yang muncul pada algoritma *blowfish* ini relatif sulit dipecahkan. Karena kemungkinannya jika terdapat kunci yang lemah, namun untuk memecahkan hal tersebut tetap harus melewati proses lain, yakni mencari pola pembentukan kunci terlebih dahulu. Sehingga pesan atau data misalnya data teks yang ingin dibongkar tanpa mengetahui kuncinya, tentu sangat tidak efisien, dan menghabiskan banyak sumber daya[11]. Secara tujuan, dengan kunci yang kuat, algoritma ini memiliki performansi yang sangat optimal sehingga algoritma ini ditujukan untuk penggunaan pada banyak tempat dan jenis pengembangan[12].

Selanjutnya, dalam penelitian ini juga bertujuan untuk mencoba menerapkan algoritma *blowfish* dalam membantu mengamankan data dan informasi pada sebuah data yang dalam penelitian ini kami menggunakan data berupa teks yang diasumsikan sebagai sebuah sandi. Lalu kami melihat cara kerja algoritma *blowfish* sehingga didapatkan analisis keamanan data dan kemungkinan untuk dikembangkan menjadi lebih kuat dan handal, serta sejauh apa efektivitas sumber daya yang dipakai algoritma *blowfish* dalam bekerja. Selanjutnya kami juga akan menganalisis kekuatan

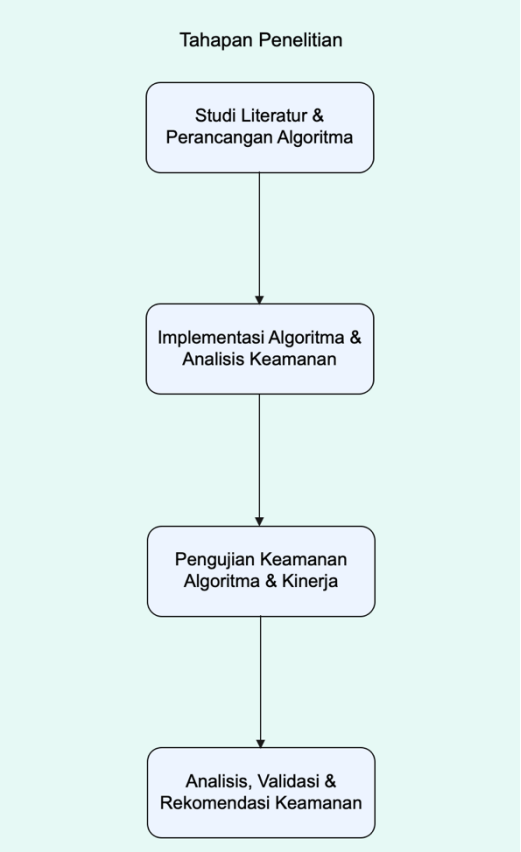
dan kelemahan serta mengukur kinerja algoritma ini terhadap efisiensi waktu eksekusi dan kebutuhan sumber daya pada memori. Sehingga tujuan kami dalam penelitian ini untuk mengukur sejauh apa kinerja algoritma *blowfish* dapat didukung dan ditopang dengan ilmu terapan yang diharapkan dapat memberikan manfaat dan menambah wawasan, dan khazanah pengetahuan. Sehingga ke depan diharapkan terdapat pengembangan lanjutan dalam segala aspek terutama terhadap alat bantu *dart* dan algoritma yang akan dan telah digunakan dalam penelitian ini.

Tentu dilakukan dan diperlukan analisa dan pengujian menggunakan pemrograman. Kami memilih menggunakan bahasa pemrograman *dart. Dart* digunakan oleh penulis karena *dart* merupakan bahasa pemrograman yang dirancang untuk dioptimalkan oleh pengguna, dengan memprioritaskan pengembangan dan pengalaman untuk memproduksi kode berkualitas tinggi dan dapat berjalan pada berbagai *platform*[13]. Selain itu, *dart* adalah bahasa pemrograman yang kuat dan modern serta serbaguna. Jika kemudian kita memiliki pengalaman dengan bahasa pemrograman berorientasi objek, atau bergaya pemrograman C/C++ atau sejenisnya, maka bisa dipastikan *dart* dapat dengan cepat dan mudah untuk dipahami. Karena serbaguna, maka dengan *dart* kita bisa menggunakannya untuk menulis apapun, misalnya baris kode perintah dan *server* untuk *backend* hingga sistem *native* misalnya untuk *android, ios, web, mac os, windows, linux*[14].

2. METODE PENELITIAN

Tahapan penelitian yang penulis gunakan dalam menganalisis algoritma *blowfish* adalah kombinasi antara analisis teoritis dan eksperimen komputasi. Dengan mengumpulkan pengetahuan kriptografi, algoritma dan pemrograman yang dipakai pada algoritma *blowfish*, lalu diikuti dengan melakukan berbagai percobaan susunan algoritma dengan menggunakan data teks sebagai sampel sandi percobaan pada komputasi yang akan diterapkan pada pemrograman *dart* sebagai jawaban dari pengumpulan pengetahuan dan pengujian yang dilakukan. Dalam metode ini, selain dilakukan analisa, yang kami lakukan juga mengukur keakuratan algoritma tersebut dan melihat ukuran waktu eksekusi yang digunakan serta sumber saya yang dihabiskan dalam menjalankan algoritma.

Peneliti memberikan alur cara dan proses kerja dari algoritma *blowfish* ini, dimulai dari inisialisasi kunci, lalu *S-Box* dan *P-Box*, berikutnya data di bagi dan di enkripsi melalui serangkaian berbagai macam putaran, kemudian data diambil dan untuk selanjutnya di dekripsi. Proses ini kami jabarkan secara lengkap pada setiap tahapan ke tahapan berikutnya dari setiap tahapan penelitian yang kami lakukan. Untuk itu, kami gambarkan pada Gambar 2 di bawah ini dari setiap tahapan metode penelitian yang dipakai untuk menganalisis kriptografi algoritma *blowfish* dan menerapkannya pada pemrograman *dart*:



Gambar 2. Tahapan Penelitian

2.1 Studi Literatur dan Perancangan Algoritma

Langkah awal dalam merancang algoritma *blowfish* adalah melakukan studi literatur tentang kriptografi, algoritma *blowfish*, enkripsi, dekripsi dan teknik pengamanan data menggunakan algoritma lainnya. Memahami dasar-dasar kriptografi, prinsip-prinsip enkripsi, dan algoritma kriptografi yang ada yang akan membantu dalam menganalisis algoritma *blowfish*. Melakukan pemilihan terhadap algoritma *blowfish* didukung dengan pengumpulan literasi terkait konsep akan kriteria algoritma itu sendiri.

Secara konsep, algoritma ini mencakup pada ekspansi kunci yang dilakukan sebelum enkripsi dan dekripsi dilakukan, dengan kombinasi pada rentang 32 bit hingga 448 bit[15]. Juga menggunakan struktur *Feistel Network* pada saat enkripsi dan dekripsi yang memanfaatkan *subkey* melalui hasil dari prosedur penggandaan kunci atau *key expansion* denagn 16 kali putaran yang dilakukan[16].

Selain itu algoritma *blowfish* dirancang dengan kriteria dan keunggulan seperti berikut ini:

1. *Fast*
   Algoritma ini dirancang untuk mengenkripsi data pada mikroprosesor 32 bit dengan kecepatan 26 siklus *clock* per bit.
2. *Compact*
   Algoritma ini bisa berjalan pada memori kurang dari 5K
3. *Simple*

Algoritma ini dapat dijalankan dengan metode yaitu penjumlahan, penggeseran, XOR, serta *tables lookup* pada 32 bit.

4. *Variable Secure*
   Algoritma ini sangat aman, dengan keamanan dalam tingkat yang bervariasi yang dapat mencapai hingga 448 bit.

Setelah memahami dasar-dasar kriptografi dan analisis keamanan, serta algoritma *blowfish*, berikutnya tahap perancangan algoritma dimulai. Perancangan algoritma juga harus mempertimbangkan efisiensi dalam penggunaan sumber daya, seperti kecepatan enkripsi dan dekripsi, ukuran kunci, dan kompleksitas algoritma.

Ini melibatkan dan diawali dengan inisialisasi dari pengembangan skema kunci, proses enkripsi, dekripsi, serta struktur algoritma secara keseluruhan. Dimulai dari inisialisasi algoritma menggunakan *P-Box* dan *S-Box*. Ini dilakukan untuk menghitung dan menjadikan algoritma sebagai nilai awal, yang akan disubstitusikan agar diubah dan diatur menjadi kunci yang akan dipakai dalam proses pengubahan data. Di sini kami menggunakan nilai konstan untuk kami masukkan ke dalam larik. *P-Box*. Ini adalah larik yang memiliki 18 elemen dengan 32 bit. Secara visualisasi, larik tersebut akan tampak seperti ini:

P1, P2, P3, P4, P5, P6, P7, P8, P9, P10, P11, P12, P13, P14, P5, P16, P17, P18

Nilai di dalam tersebut lalu diisi dan di-inisialisasikan menggunakan nilai konstanta yang sulit untuk ditebak. Dan setiap elemen tersebut kami lakukan XOR. Proses ini akan terus berulang sehingga semua elemen dalam larik P semuanya terisi. Sebagai gambaran agar Anda mudah memahami proses XOR yang didefinisikan, XOR di sini adalah sebuah operasi berbasis logika yang diterapkan pada nilai dua bit. Dua bit yang kami maksud di sini adalah nilai *boolean.* Misalnya 0 XOR 0 maka nilainya adalah 0. Proses ini dilakukan sebagai bagian dari penggabungan pesan dengan konversi pada enkripsi data yang dimaksudkan sebagai bagian dalam menjaga keamanan data.

Berikutnya adalah inisialisasi untuk larik S-Box. Terdapat larik S-Box berjumlah 4 buah, yang di dalamnya masing-masing berisi nilai konstanta yang juga tidak mudah ditebak dengan isi elemen yang berjumlah 256 elemen. Secara visualisasi larik tersebut akan berbentuk seperti ini:

S1,0, S1,1, S1,2, S1,3, S1,4, …, …, …, …, …., S1,255

S2,0, S2,1, S2,2, S2,3, S2,4, …, …, …, …, …., S2,255

S3,0, S3,1, S3,2, S3,3, S2,4, …, …, …, …, …., S3,255

S4,0, S4,1, S4,2, S4,3, S2,4, …, …, …, …, …., S4,255

2.2 Implementasi Algoritma dan Analisis Keamanan

Setelah merancang algoritma secara teoritis, langkah berikutnya adalah mengimplementasikannya dalam bentuk prototipe pada lingkungan komputasi. Implementasi prototipe memungkinkan peneliti untuk menguji algoritma *blowfish* dalam skenario praktis, dan memastikan bahwa algoritma berfungsi dengan benar dan sesuai dengan desain yang diinginkan. Kami melakukan percobaan dengan skenario praktis dengan mencoba data teks “PASSWORD KU”. Algoritma akan membagi menjadi blok terhadap data yang ada. Blok ini dibagi menjadi blok-blok dengan panjang 64 bit, pada skenario ini misalnya pemisahan blok untuk kata “PASSWORD” dan kata “KU”.

Setiap blok akan diolah pada serangkaian putaran yang di dalamnya terdapat operasi substitusi, XOR, *bitwise*, dan permutasi. Nilai blok akan ditukar dan digunakan untuk menghitung bagian blok lainnya pada putaran berikutnya. Proses ini akan terus berulang sesuai dengan jumlah kunci yang digunakan.

2.3 Pengujian Keamanan dan Kinerja

Langkah berikutnya adalah melakukan pengujian analisis keamanan terhadap algoritma yang akan dirancang. Ini termasuk mengidentifikasi potensi kelemahan dan kerentanannya terhadap berbagai serangan dan potensi celah manipulasi kriptografi dan potensi lainnya yang dapat terjadi dalam kode komputasi. Analisis keamanan akan membantu dalam mengidentifikasi pada area dimana algoritma perlu diperkuat atau dimodifikasi untuk menjaga tingkat keamanan yang optimal.

Setelah implementasi prototipe selesai, algoritma *blowfish* harus diuji untuk mengevaluasi keamanan dan kinerjanya. Pengujian keamanan melibatkan pengujian algoritma dan pengujian kinerja serta pengukuran waktu eksekusi dan penggunaan sumber daya lainnya untuk memastikan algoritma berjalan dengan efisien.

Skenario yang telah dilakukan mulai dari inisialisasi, dan pembagian blok, maka selanjutnya dilakukan proses operasi logika yang melibatkan pemutaran blok teks yang dienkripsi yang melibatkan operasi XOR, *bitwise*, dan permutasi. Geser kiri hingga geser kanan digunakan untuk mengacak data. Misalnya seperti blok A XOR B = “PASSW XOR “ORD KU” = “P!!QR!!! yang menghasilkan bentuk karakter ASCII. Lalu operasi geser kanan W >> 1 = “E$KT” dengan menggeser bit ke kanan dan menggantinya dengan bit baru.

2.4 Analisis, Validasi, dan Rekomendasi

Hasil dari pengujian keamanan dan kinerja dari proses eksekusi algoritma yang didasarkan pada keamanan dan percobaan dalam data berbasis teks memberikan gambaran bahwa dengan panjang kunci yang besar, yang memungkinkan penggunaan kunci dengan panjang hingga 448 bit, maka membuat serangan menjadi lebih sulit karena akan melakukan ruang pencarian yang sangat besar tentunya. Selanjutnya algoritma ini juga menggunakan struktur *feistel* yang akan melakukan cakupan putaran yang

memungkinkan setiap blok yang berisi data akan teracak dan mencegah terjadinya pola yang berulang. Juga didukung dengan substitusi larik membuat algoritma ini sangat aman dalam mencegah serangan.

Lalu secara kecepatan pada setiap operasinya jika dilihat dari operasi yang digunakan algoritma ini menggunakan operasi yang dimiliki XOR dan *bitwise yang* secara umum sangat efisien. Ditopang dengan substitusi larik, maka kebutuhan dan perhitungan selama operasi dilakukan akan sangat berkurang. Dan secara akurasi, setiap putaran yang dilakukan akan mencampur ulang data dengan kunci dan akan memasukkan kembali hasil tersebut ke putaran lain yang berikut-berikutnya. Hal ini menunjukkan bahwa terjadi akurasi enkripsi dengan transformasi data yang variatif.

Ukuran kompleksitas yang terjadi dalam algoritma merupakan parameter kami bahwa algoritma ini sangat baik untuk dipakai dan disisipkan ke dalam pemrograman untuk menjadi perlindungan dalam mengamankan data. Hal itu berdasarkan pada tangkapan penggunaan memori, CPU, dan waktu yang terlampir pada saat algoritma dioperasikan dan dieksekusi. Hal ini yang menjadi ukuran dan parameter yang kami perhatikan yang akan membantu dan mengindikasikan kami dalam memberikan validasi dan rekomendasi bahwa algoritma ini memiliki kecepatan terkait waktu dan memiliki penggunaan sumber daya yang baik saat digunakan dan diterapkan.

3. HASIL DAN PEMBAHASAN

3.1 Implementasi *Key Expansion*, Enkripsi, dan Dekripsi

3.1.1 *Key Expansion*

Kami menggunakan kontstruktor untuk kelas *blowfish* yang digunakan untuk membuat objek dengan menginisialisasi pada tabel berdasarkan kunci yang akan diberikan. Lihat pada Gambar 3(a), konstruktor kelas diinisialisasi. Konstruktor ini akan meng-inisalisasi *_p* dan *_sBoxes* dengan diisi tabel awal lalu kemudian metode *_setupKey* yang terdapat pada Gambar 4(b) akan dipanggil karena akan diteruskan inisialisasinya lebih lanjut dari kunci yang diberikan yang akan dilanjutkan pada *method* yang lain.

```
Blowfish(List<int> key) {
  _p = List.from(_initialP);
  _sBoxes = List.from(_initialS);

  _setupKey(key);
}
```

Gambar 3. Konstruktor Kelas (a)

```
void _setupKey(List<int> key) {
  final keyLength = key.length;
  var dataIndex = 0;
}
```

Gambar 4. Metode Kelas Konstruktor (b)

Algoritma *blowfish* menghasilkan *subkey* dari kunci utama yang diberikan menggunakan proses penggandaan kunci. Kunci utama yang dimasukkan ke dalam algoritma *blowfish*, lalu dipecah menjadi sejumlah *subkey* menggunakan fungsi yang disebut *P-Box*. Pengulangan iteratif dilakukan untuk menghasilkan *subkey* dengan menggunakan mekanisme XOR dan fungsi non-linier khusus. Ini akan mengubah kunci (minimal 32bit, maksimal 448 bit) ke dalam larik *(subkey)* hingga 4168 bit (18x32 bit pada larik P dan 4x256x32 bit pada kotak S, hingga menjadi 33344 bit, dan disimpan dalam larik K. Lalu di-*produce* melalui *subkey* kemudian dihitung. Setelah itu barulah proses berlanjut pada enkripsi dan dekripsi. *Subkey* yang dipakai yakni: Larik P berjumlah 18 (P1, …, P18) dan 32 bit *subkey*. Selanjutnya kotak S berjumlah 4 dan 32 bit *subkey* yang terdiri entri sebanyak 256 mulai dari (S1,0, ..., S1,255, S2,0, ..., S2,255, S3,0, ..., S3,255, S4,0, ..., S4,255). Selanjutnya *subkey* dihitung dengan:

1. Menginisial larik P1 hingga P4 dengan *string* dari awal dan kotak S, berurutan yang memuat digit *hexadecimal*.

   InitialP = [

   0x243F6A88,

   0x85A308D3,

   0x13198A2E,

   0x6E85076A,

   ];

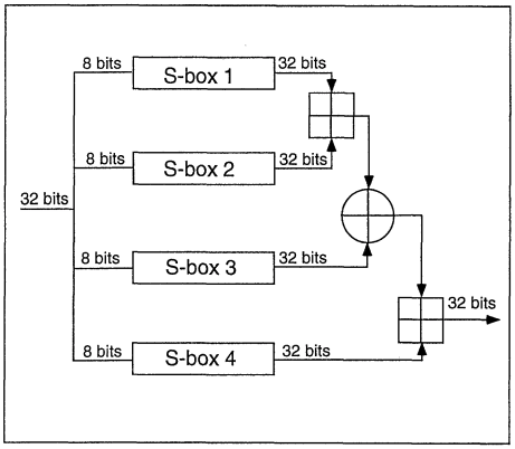
   Proses inisialisasi yang dilakukan pada larik P dan kotak S seperti yang terdapat pada Gambar 5.

```
class Blowfish {
  static const List<int> _initialP = [
    0x243F6A88,
    0x85A308D3,
    0x13198A2E,
    0x6E85076A,
  ];
```

Gambar 5. Inisialisasi P *(InitialP)*

2. Lakukan XOR pada P1 menggunakan 32 bit kunci awal, hingga seterusnya dengan semua kunci. Siklus ini dilakukan berulang dan berurutan hingga semua larik P.
3. Lakukan enkripsi yang seluruh *string* bernilai 0 menggunakan *subkey*.
4. Ubah P3 dan P4 menggunakan hasil dari proses 3
5. Lalu enkripsi dengan *subkey* menggunakan *blowfish*
6. Ubah P3 dan P4 dengan hasil dari proses sebelumnya.
7. Lakukan berurutan untuk mengubah semua elemen larik P dan kotak S.

Pada Gambar 6, terlihat alur langkah dan proses yang dilakukan *key expansion.*



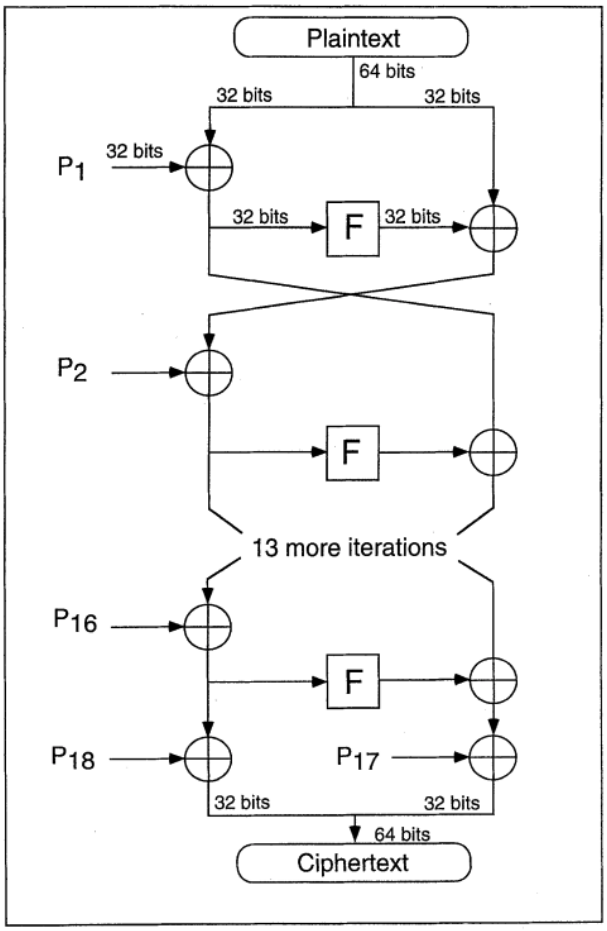
Gambar 6. Proses *Key Expansion*

3.1.2 Enkripsi

Proses enkripsi *blowfish* melibatkan pengulangan 16 putaran dengan menggunakan *subkey* yang dihasilkan dari proses penggandaan kunci. Setiap putaran terdiri dari operasi XOR, substitusi, dan permutasi. Data yang akan dienkripsi dibagi menjadi blok-blok 64-bit. Blok data melewati 16 putaran enkripsi. Dalam setiap putaran enkripsi, blok data dipecah ke dua bagian, yaitu kiri (L) dan kanan (R). X data akan melewati fungsi putaran 16 kali putaran *(Feistel Network)* 64 bit, seperti ini:

1. Buat X ke dalam dua bagian, XL, dan XR. Setiap bagian masing-masing memiliki 32 bit.
2. Selanjutnya membuat putaran sebanyak 16 kali, dan ubah data XL ke XR.
   for i = 1to 16:
   XL = XL XOR Pi
   XR = F(XL) XOR XR
3. Berikutnya putar kembali XL dan XR sehingga pertukaran sebelumnya menjadi dibatalkan.
4. Kemudian, kita lakukan:
   XR = XR XOR P17
   XL = XL XOR P18
5. Dan kembali gabungkan XL dan XR agar didapatkan *chipertext*.

Secara lengkap alur proses enkripsi menggunakan algoritma *blowfish*, dipetakan pada Gambar 7.



Gambar 7. Blok Alur Diagram Enkripsi Algoritma *Blowfish*

Dari gambar tersebut diketahui bahwa *plaintext* yang berupa data masukan yang akan dienkripsi dimasukkan ke dalam algoritma. Lalu akan dibagi menjadi blok-blok data yang biasanya berukuran 64-bit (8 *byte*) masing-masing. Selanjutnya dilakukan Inisialisasi kunci dan subkunci dilakukan menggunakan kunci rahasia yang diberikan. Subkunci ini akan digunakan dalam setiap putaran enkripsi. Proses enkripsi sendiri terdiri atas 16 putaran, di mana setiap putaran melibatkan operasi XOR, substitusi dengan *S-boxes*, dan permutasi dengan *P-boxes*. Setiap putaran mengubah blok data secara berulang. Setelah semua putaran selesai dilakukan, maka blok data hasil enkripsi adalah hasil

gabungan dari bagian kiri dan kanan yang telah diubah selama putaran enkripsi. *Ciphertext* atau data yang terenkripsi merupakan hasil akhir dari proses enkripsi dan dapat kita dikeluarkan dari proses algoritma.

3.1.3 Dekripsi

Proses dekripsi menggunakan algoritma *blowfish* adalah kebalikan dari proses enkripsi. Data terenkripsi dibagi menjadi blok-blok 64-bit. Setiap blok data melewati 16 putaran dekripsi, dengan *subkey* yang digunakan dalam urutan terbalik dari putaran enkripsi. Dalam setiap putaran dekripsi, operasi XOR dilakukan antara bagian kiri (L) dan *subkey* yang sesuai dari putaran dekripsi saat ini. Hasil XOR kemudian melewati fungsi *Feistel*, yang melibatkan operasi substitusi dan permutasi pada bagian kanan. Sebelah kiri (L) dan sebelah kanan (R) diperbarui berdasarkan hasil fungsi *Feistel*. Setelah 16 putaran dekripsi selesai, blok data diubah kembali menjadi bentuk aslinya. Proses dekripsi dilakukan dengan menggunakan *subkey* yang dihasilkan dalam urutan terbalik.

for i = 1to 16:

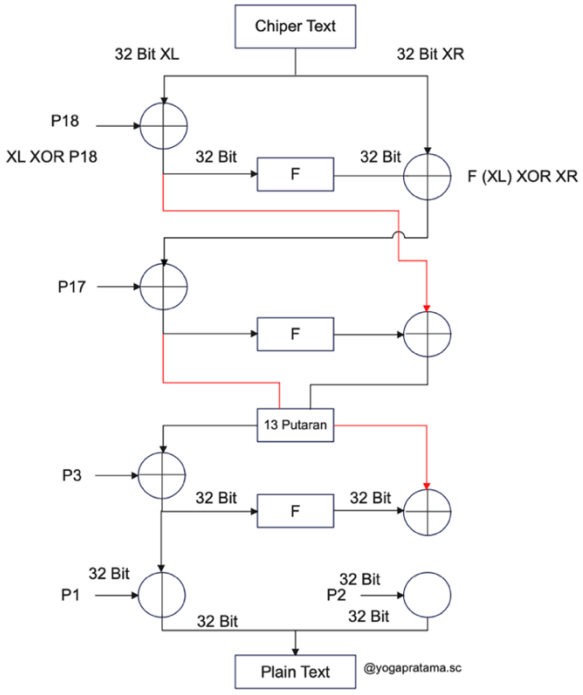
XRi = XLi-1 XOR P19-1

XLi = F[XRi} XOR XRi-1

XL17 = XR16 XOR P1

XR17 = XL16 XOR P2

Secara lengkap alur proses dekripsi menggunakan algoritma *blowfish*, dapat dilihat pada Gambar 8.



Gambar 8. Blok Alur Diagram Dekripsi Algoritma *Blowfish*

Melihat gambar di atas maka diketahui *ciphertext* adalah daya yang terenkripsi yang akan di-dekripsi lalu dimasukkan ke dalam algoritma. Kemudian subkunci yang sama yang digunakan dalam proses enkripsi, tetapi diterapkan secara terbalik, akan digunakan untuk inisialisasi kunci dan subkunci dekripsi. Dalam proses dekripsi juga terdiri atas 16 putaran yang sangat mirip dengan proses enkripsi. Setiap putaran melibatkan operasi XOR, substitusi dengan *S-boxes*, dan dilakukan permutasi dengan *P-boxes*, tetapi dengan subkunci yang diterapkan dilakukan secara terbalik. Kemudian setelah semua putaran dekripsi selesai, blok data hasil dekripsi adalah hasil gabungan dari bagian kiri dan kanan yang telah diubah selama putaran dekripsi. Disini data asli atau *plaintext* merupakan hasil akhir dari proses dekripsi dan dapat dikeluarkan dari algoritma.

3.2 Pengujian dan Pengukuran Kinerja *Blowfish*

3.2.1 Inisialisasi Kunci

Algoritma *blowfish* memulai dengan meng-inisialisasi dua tabel, yaitu P-array *(Plain Array)* dan *S-boxes (Substitution boxes). P-array* adalah larik berisi 18 elemen 32-bit yang akan digunakan selama proses enkripsi dan dekripsi. *S-boxes* terdiri dari 4 larik, masing-masing berisi 256 elemen 32-bit. *S-boxes* digunakan dalam langkah enkripsi yang disebut fungsi F.

3.2.2 Pembentukan *Key Expansion*

Kunci awal yang diberikan oleh pengguna (128-bit) akan digunakan untuk membentuk subkunci yang lebih banyak. Proses ini menggunakan *P-array* dan *S-boxes* yang telah diinisialisasi sebelumnya. Kunci 128-bit akan dipecah menjadi beberapa bagian dan akan di-XOR dengan *P-array* secara berulang untuk membentuk subkunci. Pembentukan *key expansion* pada proses komputasi dapat kita lihat pada Gambar 9 berikut.

```
for (var i = 0; i < _p.length; i += 2) {
  _encryptBlock(dataBlock);
  _p[i] = _bytesToInt(dataBlock, 0);
  _p[i + 1] = _bytesToInt(dataBlock, 4);
}

_sBoxes = List.from(_initialS);

for (var i = 0; i < _sBoxes.length; i += 2) {
  _encryptBlock(dataBlock);
  _sBoxes[i] = _bytesToInt(dataBlock, 0);
  _sBoxes[i + 1] = _bytesToInt(dataBlock, 4);
}
```

Gambar 9. Pembentukan *Key Expansion*

3.2.3 Enkripsi

Data yang ingin dienkripsi akan dibagi menjadi blok-blok 64-bit (8-byte). Data blok akan di-XOR dengan subkunci. Blok data yang di-XOR akan masuk ke fungsi F. Fungsi F mengambil blok 32-bit dan menghasilkan blok 32-bit sebagai *output* dengan melakukan operasi *bitwise* menggunakan *S-boxes* dan operasi matematika lainnya. Hasil dari fungsi F akan di-XOR dengan blok data lain dan kemudian data akan di-*switch* (pertukaran) sebelum masuk ke putaran berikutnya. Semua proses ini akan diulang sebanyak 16 putaran.

Lihat pada Gambar 10 di bawah ini untuk melihat gambaran komputasi dari pembentukan enkripsi yang dilakukan.

```
for (var i = 0; i < 16; i++) {
  left ^= _p[i];
  right ^= _encryptF(left);
  right ^= _p[i + 1];
  left ^= _encryptF(right);
}
```

Gambar 10. Pembentukan Enkripsi

3.2.4 Dekripsi

Proses dekripsi mirip dengan proses enkripsi, namun subkunci diaplikasikan dalam urutan terbalik. Data yang dienkripsi akan di-XOR dengan subkunci terakhir dan kemudian masuk ke fungsi F. Fungsi F yang dilakukan dan dibentuk dapat dilihat pada Gambar 11 berikut.

```
int _encryptF(int x) {
  var name1 = _sBoxes[(x >> .. ) & 0xFF];
  var name2 = _sBoxes[(x >> .. ) & 0xFF];
  var name3 = _sBoxes[(x >> .. ) & 0xFF];
  var name4 = _sBoxes[x & 0xFF];
  return ((name1 << ..) | (name2 << ..) | (name3 << ..) | name4);
}
```

Gambar 11. Pembentukan Fungsi F

Hasil dari fungsi F akan di-XOR dengan subkunci lainnya dan setelah 16 putaran, data akan terdekripsi. Teks data dienkripsi hanya dapat di-dekripsi kembali dengan menggunakan kunci yang sama yang digunakan untuk enkripsi. Pada kode yang diberikan, Anda dapat melihat proses inisialisasi kunci, *key expansion*, enkripsi, dekripsi, diimplementasikan dalam bentuk kelas *blowfish*. Proses enkripsi dan dekripsi terjadi pada fungsi *_encryptBlock* dan *_decryptBlock* yang menggunakan fungsi F *_encryptF* untuk menghasilkan blok data terenkripsi dan terdekripsi. Untuk menggambarkan komputasi dekripsi dapat dilihat pada Gambar 12 berikut.

```
for (var i = 16; i >= 0; i--) {
  left ^= _p[i + 1];
  right ^= _encryptF(left);
  right ^= _p[i];
  left ^= _encryptF(right);
}
```

Gambar 12. Pembentukan Dekripsi pada Komputasi

3.2.5 Hasil Pengujian

Hasil dari algoritma *blowfish* adalah data yang telah dienkripsi dan dapat dikirimkan dengan aman ke pihak lain. Enkripsi yang telah dilakukan bisa terdekripsi kembali melalui *key* yang sama yang digunakan untuk enkripsi. Pada kode kita dapat melihat proses inisialisasi kunci, *key expansion*, enkripsi, dekripsi, diimplementasikan dalam bentuk kelas *blowfish*. Tahapan ini terjadi pada fungsi *_encryptBlock* dan *_decryptBlock* yang menggunakan fungsi F *_encryptF* untuk menghasilkan blok data terenkripsi dan terdekripsi. Secara lengkap hasil pengujian pada komputasi dapat dilihat pada Gambar 13.

```
Launching lib/main_blowfish.dart on macOS in debug mode... main_blowfish.dart:1
Connecting to VM Service at ws://127.0.0.1:53025/sZxQKP7JGRM=/ws
flutter: Plaintext: [78, 111, 119, 32, 105, 115, 32, 116, 104, 101, 32, 116, 1
05, 109, 101, 32, 102, 111, 114, 32, 97, 108, 108, 32, 103, 111, 111, 100, 32,
109, 101, 110]
flutter: Encrypted: [126, 105, 177, 46, 172, 226, 50, 179, 76, 244, 16, 12, 7
9, 74, 223, 191, 236, 94, 134, 143, 134, 79, 120, 10, 48, 191, 104, 40, 185, 2
34, 95, 88]
flutter: Decrypted: [78, 111, 119, 32, 105, 115, 32, 116, 104, 101, 32, 116, 1
05, 109, 101, 32, 102, 111, 114, 32, 97, 108, 108, 32, 103, 111, 111, 100, 32,
109, 101, 110]
```

Gambar 13. Hasil Komputasi

3.2.6 Analisis dan Validasi Metode Algoritma

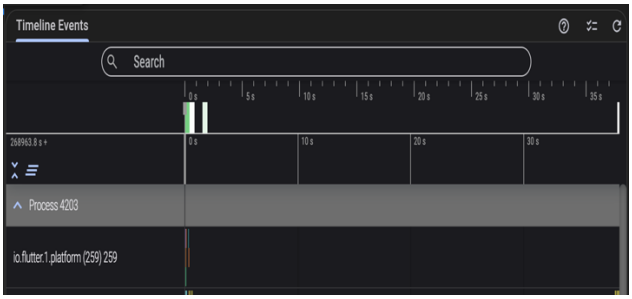
Metode enkripsi dan dekripsi data teks melalui seleksi kunci dalam penggunaan algoritma *blowfish* adalah kunci dalam penggunaan algoritma *blowfish* harus diperhatikan dipilih dengan cermat untuk memastikan kekuatan keamanan yang optimal. Kunci dapat dieksekusi secara acak atau menggunakan teknik pengambilan kunci lain yang aman yang dipilih. Proses enkripsi data teks menggunakan *blowfish* data dalam *database* dienkripsi menggunakan

algoritma *blowfish* dengan menggunakan kunci yang telah dipilih.

Data dibagi menjadi blok-blok 64-bit, dan setiap blok dienkripsi secara independen menggunakan *subkey* yang dihasilkan dari proses penggandaan kunci. Proses dekripsi data dalam *database* menggunakan *blowfish* proses dekripsi data dalam *database* dilakukan dengan menggunakan algoritma *blowfish* yang sama dengan menggunakan *subkey* yang dihasilkan dalam urutan terbalik. Setiap blok data dienkripsi secara independen dan diubah kembali menjadi bentuk aslinya. Mempertahankan integritas dan autentikasi data dalam *database*. Selain mengenkripsi data, teknik kriptografi dengan algoritma *blowfish* juga dapat digunakan untuk mempertahankan integritas dan keaslian data yang dihasilkan dari data terenkripsi.

3.2.7 Analisis dan Validasi Hasil Pengukuran Kinerja

Metode pengujian dan pengukuran terhadap kinerja dilakukan untuk mengukur kinerja teknik kriptografi yang dilakukan menggunakan algoritma *blowfish*, termasuk waktu eksekusi dan kebutuhan sumber daya seperti penggunaan *CPU* dan memori. Pengukuran waktu eksekusi diukur dengan mengamati waktu yang diperlukan pada saat proses enkripsi dan dekripsi data. Kebutuhan sumber daya seperti penggunaan *CPU*, memori, dan penyimpanan juga dievaluasi untuk memahami tingkat efisiensi teknik kriptografi yang diusulkan dalam penggunaan keamanan pada pengiriman data. Selain itu penggunaan memori dan waktu eksekusi yang cukup cepat juga memberikan nilai tambah pada algoritma ini. Secara lengkap dapat dilihat berdasarkan tangkapan layar *debug* mengenai hasil pengujian terhadap waktu eksekusi yang digunakan dapat dilihat pada Gambar 14.



Gambar 14. Tangkapan Waktu Saat Eksekusi

3.2.8 Analisis dan Validasi Keamanan Algoritma

Selanjutnya dari segi keamanan algoritma *blowfish* telah diuji dalam sampel pengujian dan algoritma ini termasuk sebagai algoritma yang kuat. Parameter ini diambil dari proses operasi-operasi logika yang terjadi di dalam algoritma tersebut. Dimana proses tersebut diantaranya operasi geser kanan, geser kiri, *bitwise*, XOR, substitusi blok, maupun logika *boolean.* Namun, masih ada kelemahan seperti rentan terhadap manipulasi dengan kunci yang sudah ada juga masih perlu diperhatikan. Secara eksekusi, efisiensi dan kinerja algoritma *blowfish* dalam memanipulasi data memiliki kecepatan yang tinggi dan efisiensi dalam penggunaan sumber daya. Algoritma ini dapat diimplementasikan dengan baik dalam pengamanan data pada data teks.

4. KESIMPULAN

Berdasarkan hasil penerapan bahwa keamanan dan keefektifan teknik kriptografi algoritma *blowfish* telah dianalisis kekuatannya pada kode yang telah disusun dan diterapkan, lalu dievaluasi serta dikur agar memperoleh rekomendasi dan hasil yang baik dan akurat. Peneliti menggunakan batasan data sampel berupa teks yang dikategorikan sebagai sebuah *password*. Hasil pengukuran didapatkan bahwa penggunaan waktu eksekusi di bawah 10 detik dan penggunaan memori kurang dari 5 MB, menjadi ukuran bahwa waktu eksekusi dan memori yang digunakan cukup efektif.

Lalu secara keamanan algoritma ini adalah algoritma yang kuat, juga memiliki keamanan yang baik dan menawarkan efisiensi yang tinggi dalam penggunaannya. Karena data yang dikirimkan akan melewati berbagai tahapan pengubahan sebelumnya hingga pada akhirnya data sudah dikirim. Hal ini menunjukkan keefektifan teknik kriptografi dalam mengamankan data menggunakan algoritma ini efisien dan andal.

Dengan demikian diharapkan hasil penelitian ini kiranya dapat memberikan implikasi dalam pengembangan keamanan data dengan mempertimbangkan penggunaan teknik kriptografi dengan algoritma *blowfish* dan dapat menjadi salah satu pilihan dalam membangun atau mengembangkan sistem berbasis teknologi informasi, khususnya menggunakan pemrograman *dart*. Diharapkan pada penelitian berikutnya, penelitian lainnya yang membahas keamanan data, baik jenis data ataupun jenis algoritmanya menggunakan pemrograman *dart* dapat terus dilakukan, sebab didukung dengan kinerja yang baik dari algoritma *blowfish* dan ke depannya tentu penelitian lainnya juga dapat mengembangkan kekuatan algoritma yang sudah ada dan meminimalkan celah yang ada pada algoritma ini.

DAFTAR PUSTAKA

[1] Y. Pratama and T. Sutabri, “Service Operation ITIL V3 Pada Analisis dan Evaluasi Layanan Teknologi Informasi,” *Nuansa Inform.*, vol. 17, pp. 169–178, 2023, [Online]. Available: https://journal.uniku.ac.id/index.php/ilkom/article/view/7233%0Ahttps://journal.uniku.ac.id/index.php/ilkom/article/download/7233/3490.

[2] P. P. Santoso *et al.*, “Systematic literature review: Comparison study of symmetric key and asymmetric key algorithm,” *IOP Conf. Ser. Mater. Sci. Eng.*, vol. 420, no. 1, 2018, doi: 10.1088/1757-899X/420/1/012111.

[3] T. Sutabri, *Konsep Sistem Informasi*, vol. 3, no. 1. Yogyakarta, 2012.

[4] R. Mustajab, "BSSN: Ada 311 Kasus Kebocoran Data di Indonesia pada 2022," *dataindonesia.id*, 2023. https://dataindonesia.id/internet/detail/bssn-ada-311-kasus-kebocoran-data-di-indonesia-pada-2022.

[5] S. Suhandinata, R. A. Rizal, D. O. Wijaya, P. Warren, and Srinjiwi, "Analisis Performa Kriptografi Hybrid Algoritma RSA," *Jurteksi*, vol. VI, no. 1, pp. 1–10, 2019.

[6] D. A. Meko, "Perbandingan Algoritma DES, AES, IDEA Dan Blowfish dalam Enkripsi dan Dekripsi Data", j. teknologi terpadu, vol. 4, no. 1, Jul 2018.

[7] I. A. W. Arnawa, P. E. W. Hary, and A. A. G. B. Putra, "Perbandingan Waktu Enkripsi antara Metode Electronic Codebook (ECB) dan Chipher Block Chaining (CBC) dalam Algoritma Blowfish," *J. Ilmu Komput. Indones.*, vol. 5, no. 1, pp. 50–54, 2020, [Online]. Available: https://ejournal-pasca.undiksha.ac.id/index.php/jik/article/download/3056/1723.

[8] N. Fahriani and H. Rosyid, "Implementasi Teknik Enkripsi dan Dekripsi dI File Video Menggunakan Algoritma Blowfish," *J. Teknol. Inf. dan Ilmu Komput.*, vol. 6, no. 6, p. 697, 2019, doi: 10.25126/jtiik.2019661465.

[9] N. Fahriani and I. Kurniawati, "Keamanan Data Pasien dengan Algoritma Blowfish pada HOTSPODT," *J. Comput. Sci. Informatics Eng.*, vol. 5, no. 2, pp. 140–148, 2021, doi: 10.29303/jcosine.v5i2.416.

[10] M. Khairani and N. Nurwulan, "Algoritma Blowfish Pada Watermarking Video Digital," *JURIKOM (Jurnal …*, vol. 5, no. 4, pp. 357–361, 2018, [Online]. Available: http://ejurnal.stmik-budidarma.ac.id/index.php/jurikom/article/view/842.

[11] H. G. Simanullang and A. P. Silalahi, "Algoritma Blowfish Untuk Meningkatkan Keamanan Database Mysql," *Method. J. Tek. Inform. dan Sist. Inf.*, vol. 4, no. 1, pp. 10–14, 2018, doi: 10.46880/mtk.v4i1.58.

[12] S. Retno and N. Hasdyna, "Analisis Kinerja Algoritma Honey Encryption Dan Algoritma Blowfish Pada Proses Enkripsi Dan Dekripsi," *TECHSI - J. Tek. Inform.*, vol. 10, no. 2, p. 82, 2018, doi: 10.29103/techsi.v10i2.858.

[13] Google, "Dart overview | Dart," *Dart Dev*, 2011. https://dart.dev/overview.

[14] J. Sande, *Dart Apprentice: Beyod the Basics*. Kodeco, 2022.

[15] M. H. T. A. Thoriq, A. I. H. Asep, and P. N. S. Puspita, "Data Encryption Pada File Video Menggunakan Algoritma Blowfish Berbasis Android," *Informatics Digit. Expert*, vol. 4, no. 1, pp. 33–39, 2022, doi: 10.36423/index.v4i1.880.

[16] M. Muhathir, "Perbandingan Algoritma Blowfish Dan Twofish Untuk Kriptografi File Gambar," *J. Informatics Telecommun. Eng.*, vol. 2, no. 1, p. 23, 2018, doi: 10.31289/jite.v2i1.1673.