Implementasi *Audio Secret Sharing* dengan Memanfaatkan Skema Shamir

Muhammad Gumilang (13514092)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung
muhammadgumilang11@gmail.com

***Abstrak*—Makalah ini membahas pemanfaatan *secret sharing* menggunakan Skema Shamir pada *file* audio. *File* audio yang digunakan untuk eksperimen hanya *file* WAVE (.wav) saja. Meskipun teknik *secret sharing* dengan skema Shamir berhasil dilakukan pada *file* audio, dimana program dapat memecah *file* dan menggabungkannya kembali ke *file* semula hanya dengan beberapa file, terdapat kekurangan yang dihasilkan pada *file* yang dibagikan hasil *secret sharing*. Kekurangannya adalah pesan masih dapat terdengar pada *file* yang dibagikan meskipun terdapat distorsi suara. Namun, terdapat beberapa konfigurasi yang dapat diubah dan penambahan teknik kriptografi yang dapat dilakukan untuk mengurangi kekurangan dari *secret sharing* pada *file* audio.**

Kata kunci—audio; wav; secret sharing; skema Shamir; share; buffer; byte.

I. PENDAHULUAN

Komunikasi melalui network secara umum dilakukan dengan mengirimkan pesan dalam bentu teks dan gambar. Pada zaman modern ini, komunikasi dapat dilakukan dengan mengirimkan rekaman audio terhadap penerima. Pesan yang dikirim dalam bentuk audio dapat dengan mudah disadap oleh orang ketiga dengan mendengarkan audio yang dikirimkan. Maka dari itu, pengiriman pesan melalui audio juga perlu kemanan dengan memanfaatkan kriptografi seperti yang sudah diterapkan pada media teks dan gambar.

Pengiriman suatu data atau penyembunyian data tertentu dilakukan dengan kriptografi dan steganografi. Kriptografi memlibatkan proses pengenkripsian data untuk menyembunyikan isi dari data tersebut. Untuk melihat konten yang telah disembunyikan, pengguna perlu mendekripsi data dengan kunci yang telah ditentukan oleh pengguna yang melakukan enkripsi. Terdapat banya algoritma kriptografi untuk mengamankan data, namun terkadang hanya menyembunyikan informasi saja tidak cukup. Steganografi adalah seni yang memfokuskan pada penyembunyian kehadiran data daripada isi data itu sendiri. Saat keduanya digabungkan hal ini memberikan kemanan tingkat tinggi.

Kelemahan dari penggabungan kedua teknik adalah data rahasia cenderung tersentralisasi. Seluruh data rahasia tertitik pada lokasi atau pada satu pembawa informasi. Apabila pembawa informasi menjadi *corrupt* atau dimodifikasi, seluruh data rahasia menjadi tidak dapat diproses untuk mendapatkan data aslinya kembali. Memiliki data rahasia yang terfkuskan pada satu lokasi saja sangat rawan terhada ancaman penyerangan keamanan. Jika penyerang mendapatkan pembawa pesan, membuka pesan rahasia menjadi sangat mudah. Pada makalah ini, penggunaan *secret sharing* menjadi skema untuk menutupi kelemahan yang dijelaskan. Isi dari audio yang dipecahkan dengan skema *secret sharing* menambah tingkat keamanan.

Skema *secret sharing* dilakukan untuk memecah media menjadi beberapa file yang berbeda, dimana dengan hanya membaca satu file saja tidak akan cukup untuk mendapatkan informasi yang telah dirahasiakan. Dengan skema secret sharing, untuk mendapatkan informasi yang dirahasiakan diperlukan menggabungkan file-file hasil pecahan dari media asli. Skema secret sharing memungkinkan agar tidak seluruh file hasil pecahan perlu digabung, namun terdapat threshold untuk jumlah file yang perlu digabungkan. Skema tersebut disebut juga dengan Shamir Threshold Scheme yang ditemukan oleh Shamir (1979) [3].

Pengaplikasian skema *secret sharing* telah dilakukan pada penelitian yang sudah ada oleh Kai Chan [1]. Pada penelitian tersebut, skema *secret sharing* digunakan untuk menyembunyikan file pada suatu audio file. File yang disembunyikan sebelumnya dipecahkan menjadi beberapa *share*. *File share* yang dihasilkan dengan masing-masing memiliki ukuran *file* yang sama selanjutnya disembunyikan ke *file* audio yang berbeda-beda dengan jumlah *file* audio sebanyak jumlah *share*.

Pada makalah ini, skema *secret sharing* akan fokus pada pemecahan satu *file* audio menjadi beberapa *file* menggunakan Skema Shamir. Pemecahan dilakukan pada *body* audio saja yang merupakan *byte* yang tersimpan pada *frame-frame* pada *channel*. *File* hasil pemecahan akan memiliki parameter yang sama dengan audio asli, yaitu jumlah *channel*, *framerate*, dan *sample width*. Dalam melakukan proses skema *secret sharing* Shamir untuk data audio, terdapat beberapa parameter yang dapat diubah yang menentukan kinerja dari proses skema *secret sharing* Shamir. Parameter-parameter yang ada seperti jumlah partisipan dan pengenkripsian data sebelum diproses akan diuji untuk melihat durasi dari proses skema *secret sharing* Shamir pada data audio dan kualitas *file* hasil dari proses tersebut dan dianalisis lebih lanjut.

II. DASAR TEORI

A. *Skema Secret Sharing Shamir*

Skema *secret sharing* yang digunakan pada makalah ini berdasarkan pada skema Shamir. Pada dasarnya, ide dari skema Shamir adalah memecahkan suatu data menjadi *n* banyak *file* (*n* > 1) dan terdapat *threshold* atau batas *t* ($0 < t \leq n$) untuk banyaknya jumlah *file* yang diperlukan untuk mengkonstruksi ulang data yang sebelumnya dipecah.

Pada skema *secret sharing* Shamir, terdapat dua bagian penting yang dilakukan, yaitu saat pembagian *share* dan rekonstruksi data. *Share* merupakan data yang dihasilkan dari pemecahan data original. Pemecahan dan rekonstruksi data pada skema *secret sharing* Shamir menggunakan polinom dengan derajat sama dengan *t*-1. Berikut adalah langkah yang dilakukan pada skema *secret sharing* Shamir.

1. Pembagian *Share*

   Untuk memecah data menjadi *n* banyak data, diperlukan polinom yang ditetapkan sebelumnya. Polinom yang ditetapkan memiliki derajat sama dengan *t*-1. Apabila *M* merupakan data yang akan dipecah, maka *M* menjadi koefisien pada polinom.

$$F(x) = a_0 + a_1 x + a_2 x^2 + \cdots + a_{t-1} x^{t-1} \ (mod\ p) \quad (1)$$

   Tentunya, a_0 merupakan *M*. Nilai a_1, a_2, hingga a_{t-1} merupakan bilangan random yang diinisialisasi di awal dengan nilai 1 hingga *p*. Nilai *p* merupakan bilangan prima yang ditentukan juga di awal. Setelah polinom dibentuk, *share* didapatkan dengan memasukkan x dengan nilai lebih dari 0.

$$S_1 = F(1), S_2 = F(2), \ldots, S_n = F(n) \quad (2)$$

   Setiap partisipan akan memiliki nomor *id* sendiri yang kemudian dengan nomor tersebut dimasukkan pada fungsi *F(x)*, sehingga nilai yang dihasilkan menjadi *share* partisipan tersebut. Setiap partisipan juga perlu mengetahui bilangan prima yang digunakan, namun untuk nilai *M* dan a_1, a_2, hingga a_{t-1} dirahasiakan.

2. Rekonstruksi Data dengan *Share*

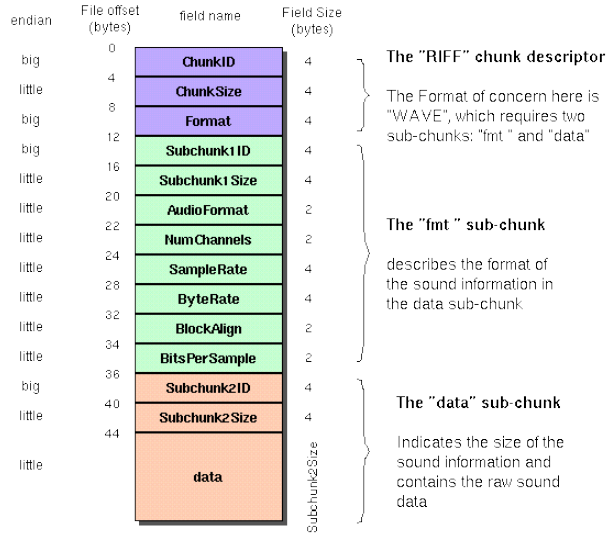
   Setelah *share* dihasilkan dan dibagikan, data dapat direkonstruksi ulang dengan menggunakan *share* sebanyak minimal sebanyak *t* banyak *share*. Saat merekonstruksi data dengan mendapatkan nilai *M* kembali, nilai yang diperlukan adalah nilai *t*, *share*, dan nomor partisipan masing-masing.

$$\begin{pmatrix} 1 & x_1 & \cdots & {x_1}^{t-1} \\ 1 & x_2 & \cdots & {x_2}^{t-1} \\ \vdots & \vdots & \ddots & \vdots \\ 1 & x_t & \ldots & {x_t}^{t-1} \end{pmatrix} \begin{pmatrix} a_0 \\ a_1 \\ \vdots \\ a_{t-1} \end{pmatrix} = \begin{pmatrix} S_1 \\ S_2 \\ \vdots \\ S_t \end{pmatrix} \quad (3)$$

   Untuk mendapatkan nilai a_0, a_1, hingga a_{t-1} dengan menggunakan teknik Gauss-Jordan. Setelah angka a_0, a_1, hingga a_{t-1} didapatkan, hanya a_0 atau *M* yang diambil.

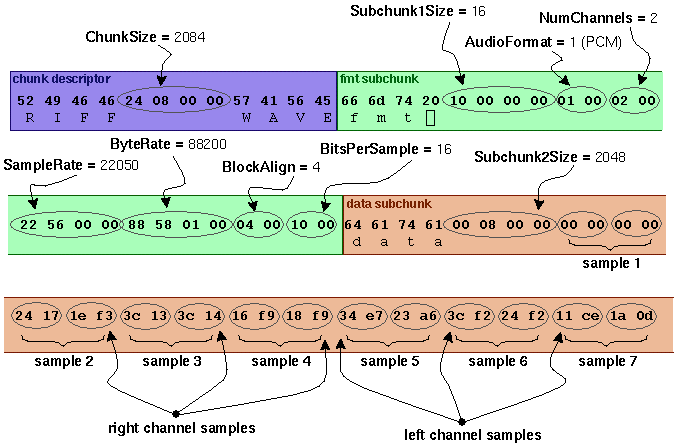
B. *Struktur Data WAVE*

Untuk memproses data audio WAVE, struktur dari tipe data tersebut perlu diketahui. Struktur data WAVE memiliki beberapa *layer chunk* yang terbagi menjadi 3 bagian, yaitu deskripsi *chunk* RIFF, *sub-chunk* 'fmt', dan *sub-chunk* data.



Gambar 1. Struktur *File* WAVE [6]

Berikut adalah contoh *file* WAVE sebesar 72 *byte*.



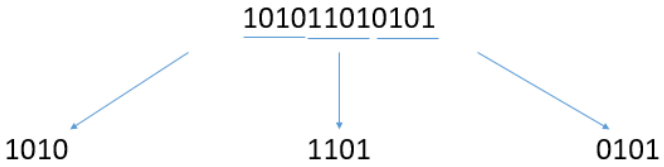
Gambar 2. Contoh Struktur Dalam *File* Audio Berukuran 72 *Byte* [6]

Sebelum membaca data audio, pembacaan atribut seperti jumlah *channel* penting. Jumlah *channel* menentukan apakah data WAVE *stereo* (2) atau *mono* (1). Data juga dibungkus menjadi beberapa *sample*, dimana untuk setiap *sample* terdiri dari beberapa *bit* yang telah ditentukan sebelumnya pada BitsPerSample.

C. *Algoritma Enkripsi Modern Operasi XOR*

Dalam algoritma enkripsi modern, operasi bit *xor* paling banyak digunakan. Algoritma enkripsi modern tetap menggunakan gagasan pada algoritma klasik: substitusi dan

transposisi, namun lebih rumit. Perkembangan kriptografi modern didorong oleh pengguna komputer digital untuk kemanan pesan. Komputer digital merepresentasikan data adlam biner. Dalam memproses dengan algoritma enkripsi modern, pesan dipecah menjadi beberapa blok.

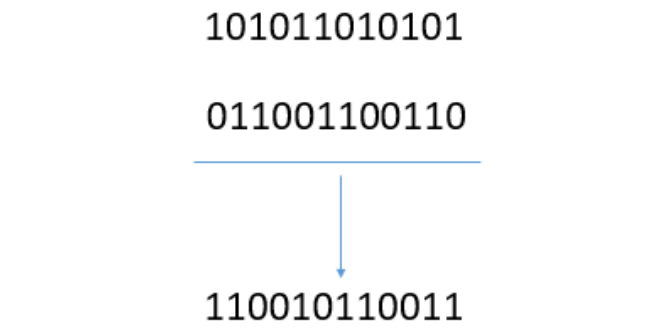


Gambar 3. Contoh Pemecahan Pesan Menjadi Blok 4 Bit

Setelah pesan dipecah kedalam blok-blok bit, pesan dilakukan operasi bit *xor*, dimana operasi bit *xor* ditunjukkan sebagai berikut.

$$0 \; xor \; 0 = 0 \quad (4)$$
$$1 \; xor \; 0 = 1 \quad (5)$$
$$0 \; xor \; 1 = 1 \quad (6)$$
$$1 \; xor \; 1 = 0 \quad (7)$$

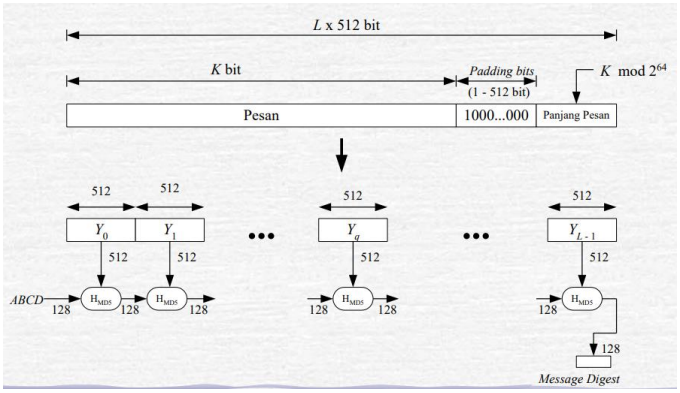
Selanjutnya, ditentukan kunci yang akan digunakan untuk menggunakan blok pesan dengan bit tertentu. Berikut adalah contoh proses enkripsi dengan kunci ‘0110’.



Gambar 4. Contoh Enkripsi Pesan dengan Kunci '0110'

D. Algoritma MD5

MD5 adalah fungsi *hash* satu-arah yang dibuat oleh Ron Rivest. MD5 merupakan perbaikan dari MD4 setelah MD4 ditemukan kolisinya. Algoritma MD5 menerima masukan pesan dengan ukuran sembarang dan menghasilkan *message digest* yang panjangnya 128 bit.

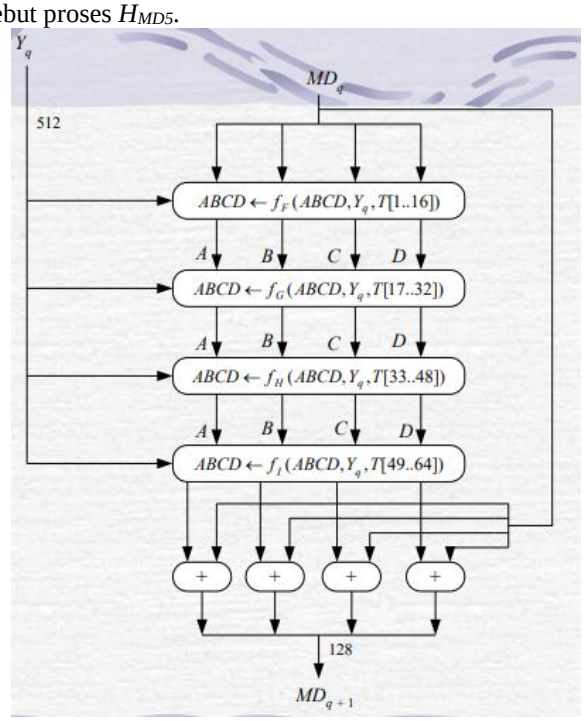


Gambar 5. Gambaran Umum MD5

Pembuatan *message digest* pertama-tama dilakukan dengan penambahan bit-bit pengganjal. Pesan ditambah dengan sejumlah bit pengganjal sedemikian sehingga panjang pesan (dalam satuan bit) kongruen dengan 448 (mod 512). Selanjutnya pesan yang telah diberi bit-bit pengganjal selanjutya ditambah lagi dengan 64 bit yang menyatakan panjang pesan semula. Jika panjang pean > 2^{64} maka yang diambil adalah panjangnya dalam modulo 2^{64}. Dengan kata lain, jika panjang pesan semula adalah *K* bit, maka 64 bit yang ditambahkan menyatakan *K* modulo 2^{64}.

MD5 membutuhkan 4 penyangga (*buffer*) yang masing-masing panjangnya 32 bit. Total panjang penyangga adalah 4 x 32 = 128 bit. Keempat penyengga ini menampung hasil antara dan hasil akhir. Keempat penyangga diinisialisasikan dengan nilai-nilai sebagai berikut dalam notasi HEX: ‘01234567’, ‘89ABCDEF’, ‘FEDCBA98’, dan ‘76543210’.

Pesan selanjutnya dibagi menjadi *L* buah blok yang masing-masing panjangnya 512 bit. Setiap blok 512-bit diproses bersama dengan penyangga *MD* menjadi keluaran 128-bit, dan disebut proses H_{MD5}.

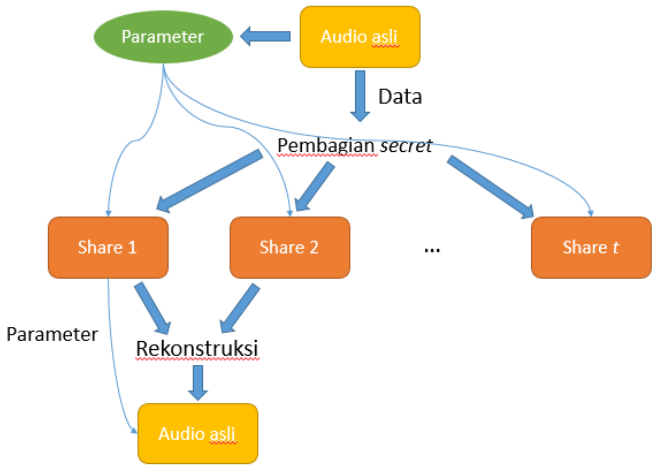


Gambar 6. Proses H_{MD5}

III. Implementasi Audio Secret Sharing

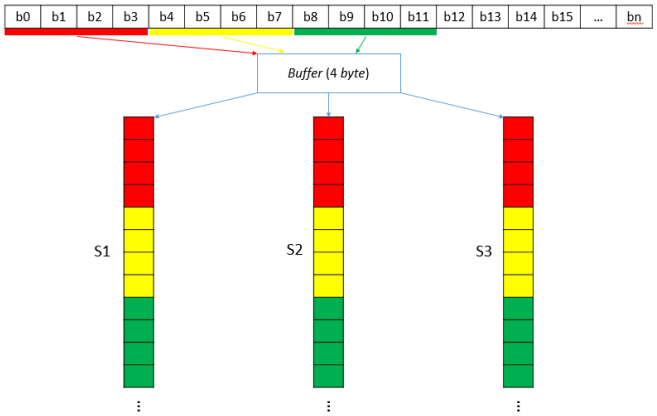
Skema *audio secret sharing* Shamir yang dilakukan diimplementasikan dengan bahasa pemrograman Python. Untuk membaca *file* audio WAVE (.wav), Python memiliki *library* tersendiri yang disebut dengan 'wav'. *Library* 'wav' akan membaca dan menulis *file* audio WAVE dalam bentuk *byte*.

Untuk mengolah melakukan *secret sharing* pada *file* WAVE, selain pembacaan data audio, parameter seperti jumlah *channel* dan *framerate*, akan dicatat dan digunakan kembali saat membentuk audio baru. Berikut adalah skema umum dari implementasi skema *secret sharing* Shamir pada data audio.



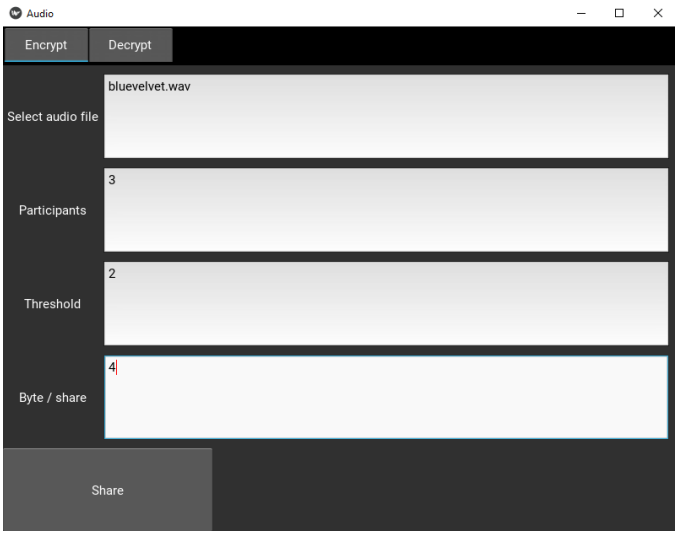
Gambar 7. Gambaran Umum Skema *Secret Sharing* Shamir pada Data Audio

Pengolahan data audio dilakukan dengan mengambil beberapa *byte* dari bagian audio dan mengambil nilai *byte* tersebut, yang kemudian dijadikan sebagai *M* atau a_0 pada polinom skema Shamir. Dengan kata lain, digunakan *buffer* untuk memproses *secret sharing* data. Setelah *share* didapatkan, masing-masing partisipan mengumpulkan *share* yang diperoleh kedalam *list of share* masing-masing.

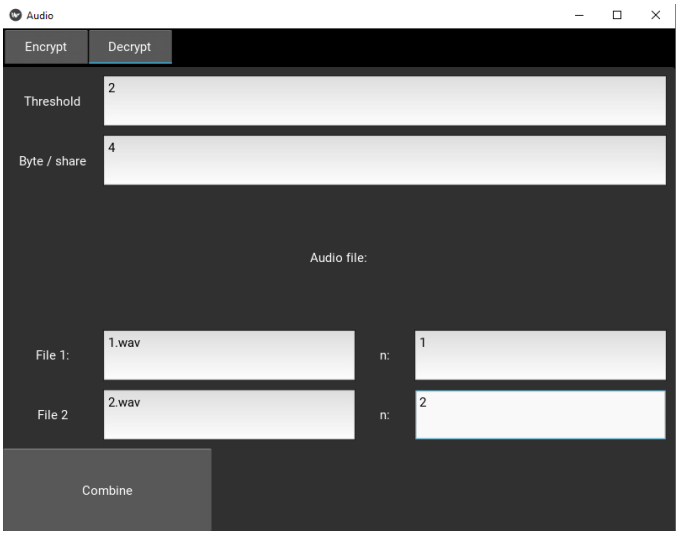


Gambar 8. Skema Pemrosesan Data Audio pada Pembagian *Secret*

Karena data yang akan menjadi *M* akan selalu bernilai antara 0 dan 2 pangkat jumlah *bit* yang diambil dari data audio untuk setiap proses *secret sharing*, nilai prima (*p*) tidak dapat sembarang nilai. Hal ini terjadi karena apabila nilai *p* lebih kecil dari 2 pangkat jumlah *bit* yang diproses atau nilai maksimal *byte* yang digunakan, maka saat suatu nilai yang lebih besar dari *p* menjadi *M*, saat dimasukkan ke polinom, nilainya akan berubah karena nilai tersebut perlu dilakukan operasi modulo dengan *p*. Sedangkan bila, nilai *p* lebih besar, saat melakukan rekonstruksi data audio, apabila nilai yang dihasilkan lebih besar dari nilai maksimal *byte* yang digunakan dan lebih kecil dari nilai *p*, maka nilai tersebut tidak akan cukup untuk dijadikan ke dalam bentuk *byte* kembali dengan ukuran *byte* yang digunakan. Maka dari itu, nilai *p* yang digunakan harus sama dengan nilai maksimal *byte* yang digunakan.



Gambar 9. Tampilan Program Bagian Enkripsi (Pembagian *Secret*)



Gambar 10. Tampilan Program Bagian Dekripsi (Rekonstruksi)

IV. Eksperiman dan Pembahasan Hasil

Eksperimen skema *secret sharing* Shamir pada data audio akan dilakukan dengan menggunakan data audio kecil dan besar yang ditujukkan pada Tabel 1. Pada eksperimen ini, waktu pemecahan *file* dan rekonstruksi data audio akan diuji untuk tipe data, besar *byte* yang diproses, dan jumlah partisipan serta batas yang berbeda-beda. Untuk menguji keabsahan rekonstruksi data

audio (apakah data audio sama dengan data sebelumnya) digunakan fungsi *hash* MD5 untuk membandingkan 2 file yang berbeda. Selain itu, kualitas dari *file* hasil pemecahan data audio original akan dianalisis juga.

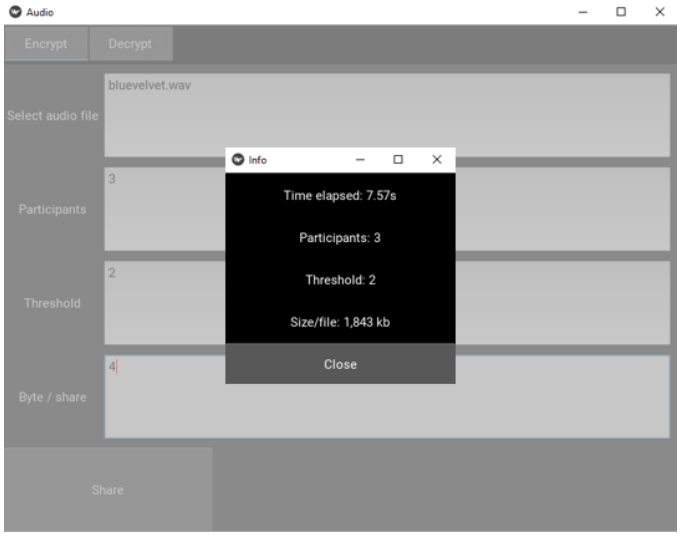
Tabel 1. Data Audio yang Digunakan untuk Eksperimen

Nama *file*	**Ukuran**	**Durasi**
funnyten.wav	1.843 kB	10 detik
bluevelvet.wav	10.353 kB	120 detik

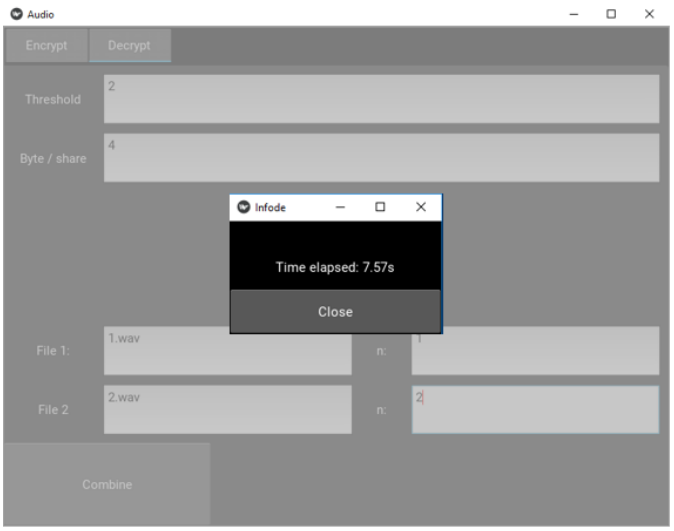
Nilai *byte* yang akan diuji untuk pemrosesan *secret sharing* adalah 1, 2, 4, dan 8. Untuk nilai partisipan dan batas yang digunakan adalah (3,2) dan (8,3). Pengujian dilakukan dengan spesifikasi mesin berupa *Intel Core i5-5200U* (2.7 GHz) dengan RAM 12 GB

A. Uji Durasi Secret Sharing

Uji durasi *secret sharing* dilakukan untuk melihat kecepatan proses dari skema Shamir, sekaligus menguji keabsahan. Ukuran *buffer* yang digunakan adalah 2 *byte* dengan 3 partisipan dan nilai batas 2. Tabel 2 menunjukkan hasil pengujian durasi pada data besar dan data kecil.



Gambar 11. Tampilan *Popup* Saat Program Selesai Membagi *Secret*



Gambar 12. Tampilan *Popup* Saat Program Selesai Rekonstruksi

Tabel 2. Hasil Pengujian Durasi Proses Skema *Secret Sharing* Shamir untuk Data Audio Ukuran Kecil dan Besar

Nama *file*	**Proses**	**Durasi (detik)**
funnyten.wav	Pembagian *secret*	7,57
	Praproses *share*	3,6
	Rekonstruksi	120,67
bluevelvet.wav	Pembagian *secret*	84,03
	Praproses *share*	34,12
	Rekonstruksi	1.355,20

Karena saat melakukan rekonstruksi memerlukan pembacaan beberapa audio *file* dan mengubah data kedalam bentuk *list of integer*, maka terdapat waktu tambahan saat melakukan rekonstruksi. Untuk membuktikan keabsahan program dalam membagi *share* dan rekonstruksi data, dilakukan pengecekan data hasil rekonstruksi dari kombinasi ketiga data dengan data original. Hasil dari kombinasi *file* 1, 2, dan 3, ketiga-tiganya memiliki nilai MD5 yang sama dengan data asli. Namun terlepas dari keberhasilan skema Shamir, pesan asli pada *file* audio yang dihasilkan dari pembagian *share* masih dapat terdengar meskipun terdapat distorsi suara pada *file* audio. Maka dari itu, diperlukan pengujian untuk nilai parameter yang berbeda-beda untuk meningkatkan kualitas dari *share* yang dibagikan.

B. Uji Pengaruh Jumlah Byte Buffer

Uji pengaruh ukuran *buffer* dilakukan untuk melihat apakah kualitas dari *file* hasil pembagian *secret* berubah atau tidak. Selain itu pengujian juga dilakukan untuk melihat apakah pengubahan ukuran *buffer* mempengaruhi durasi dari proses skema Shamir. Nilai yang akan diuji adalah 1, 2, 4, dan 8 *byte*. Angka-angka tersebut dipilih karena dapat membagi habis jumlah *byte* keseluruhan data audio. Tabel III. menunjukkan hasil pembagian *share* dengan ukuran yang berbeda-beda untuk

3 partisipan dan nilai batas 2 pada data ukuran kecil (‘funnyten.wav’).

Tabel 3. Hasil Pengujian Durasi Proses Skema *Secret Sharing* Shamir untuk Ukuran *Buffer* yang Berbeda-beda

Ukuran *Buffer* (*byte*)	**Durasi (detik)**
1	15,52
2	7,57
4	4,00
8	2,04

Dari Tabel 3, dapat dilihat bahwa semakin besar ukuran *buffer* maka proses pembagian *share* akan lebih cepat. Meskipun dengan nilai *buffer* lebih besar komputasi yang dilakukan lebih besar, hal tersebut tidak terlalu mempengaruhi durasi yang diperlukan untuk memproses data.

Untuk menguji kualitas dari akan dilakukan dengan mendengarkan setiap *file share* satu per satu untuk ukuran *buffer* yang berbeda-beda dan memberi nilai kualitas distorsi rendah, sedang, atau tinggi. Selain itu, apakah pesan asli masih dapat terdengar pada *file share* atau tidak juga menentukan kualitasi dari pembagian *share*.

Tabel 4. Hasil Pengujian Kualitas *File Share* untuk Ukuran *Buffer* yang Berbeda-beda

Ukuran *buffer (byte*)	***File share***	**Distorsi suara**	**Pesan asli terdengar**
1	1	Rendah	Ya
	2	Tidak ada	Ya
	3	Sedang	Ya
2	1	Rendah	Ya
	2	Rendah	Ya
	3	Rendah	Ya
4	1	Rendah	Ya
	2	Sedang	Ya
	3	Sedang	Ya
8	1	Tinggi	Ya
	2	Tinggi	Ya
	3	Tinggi	Tidak

Tabel 4 menunjukkan bahwa semakin besar ukuran *buffer* makan akan semakin besar distorsi suara yang dihasilkan. Untuk menentukan apakah pesan asli terdengar atau tidak, digunakan pendapat responden untuk mengisi ‘ya’ atau ‘tidak’. Responden yang mendengarkan *file share* sebelumnya tidak mendengarkan pesan asli. Meskipun distorsi semakin tinggi untuk ukuran *buffer* yang semakin besar, namun pesan asli masih dapat terdengar untuk sebagian besar *file*. Maka dari itu, diperlukan eksperimen lebih lanjut untuk meningkatkan kualitas *file share* sehingga menghilangkan pesan asli.

C. Uji Pengaruh Nilai Partisipan

Uji pengaruh nilai partisipan dilakukan untuk melihat apakah semakin banyak partisipan maka durasi pemrosesan pembagian *secret* lebih besar dan kualitas file hasil pembagian share akan semakin baik atau tidak. Pada pengujian ini, digunakan data kecil ‘funnyten.wav’ dengan ukuran buffer sebesar 8 *byte*. Nilai partisipan yang akan diuji adalah 3, 8, dan 15. Penentuan pesan asli masih terdengar atau tidak dipertimbangkan dengan responden sudah mendengarkan pesan asli sebelumnya.

Tabel 5. Hasil Pengujian Durasi Proses Pembagian *Secret* untuk Jumlah Partisipan yang Berbeda-beda

Jumlah partisipan	**Durasi (detik)**
3	2,04
8	4,39
15	7,11

Tabel 6. Hasil Pengujian Kualitas *File Share* untuk Jumlah Partisipan yang Berbeda-beda

Jumlah partisipan	***File share***	**Distori suara**	**Pesan asli terdengar**
3	1	Tinggi	Ya
	2	Tinggi	Ya
	3	Tinggi	Tidak
8	1	Tinggi	Tidak
	2	Tinggi	Tidak
	3	Sedang	Ya
	4	Sedang	Ya
	5	Tinggi	Ya
	6	Tinggi	Tidak
	7	Rendah	Ya
	8	Tinggi	Ya
15	1	Tinggi	Ya
	2	Tinggi	Ya
	3	Tinggi	Tidak
	4	Tinggi	Tidak
	5	Sedang	Ya
	6	Sedang	Ya
	7	Sedang	Ya
	8	Rendah	Ya
	9	Tinggi	Ya
	10	Tinggi	Ya
	11	Tinggi	Tidak
	12	Tinggi	Ya
	13	Sedang	Ya
	14	Tinggi	Tidak
	15	Tinggi	Tidak

Dari hasil pengujian, Tabel 5 menunjukkan bahwa semakin banyak partisipan maka waktu yang dilakukan untuk memproses pembagian *secret* semakin lama, sedangkan untuk

kualitas *file share* tidak dapat dipastikan lebih baik atau tidak. Hal ini terjadi karena pada nilai jumlah partisipan yang lebih besar, ternyata masih terdapat distorsi suara rendah meskipun telah menggunakan ukuran *buffer* dengan nilai 8 *byte*.

D. Uji Enkripsi Data

Karena kualitas data audio lebih masih belum cukup baik untuk menyembunyikan pesan dari data asli pada *file share*, maka dilakukan enkripsi sedarhana sebagai upaya untuk meningkatkan kualitas skema Shamir pada data audio. Enkripsi yang dilakukan hanya memerlukan perkalian *xor* terhadap data dengan *byte* '10101010'. Parameter pengujian yang dilakukan sama dengan pengujian pengaruh nilai partisipan, yaitu dengan data 'funnyten.wav' dan ukuran *buffer* sebesar 8 *byte*.

Tabel 7. Hasil Pengujian Kualitas *File Share* untuk Jumlah Partisipan yang Berbeda-beda dengan Enkripsi Data Sebelumnya

Jumlah partisipan	***File share***	**Distori suara**	**Pesan asli terdengar**
3	1	Tinggi	Tidak
	2	Tinggi	Tidak
	3	Tinggi	Tidak
8	1	Tinggi	Tidak
	2	Tinggi	Tidak
	3	Tinggi	Tidak
	4	Tinggi	Ya
	5	Tinggi	Ya
	6	Tinggi	Tidak
	7	Tinggi	Ya
	8	Tinggi	Tidak
15	1	Tinggi	Tidak
	2	Tinggi	Ya
	3	Tinggi	Tidak
	4	Tinggi	Tidak
	5	Tinggi	Tidak
	6	Tinggi	Ya
	7	Sedang	Ya
	8	Tinggi	Tidak
	9	Tinggi	Ya
	10	Tinggi	Tidak
	11	Tinggi	Tidak
	12	Tinggi	Tidak
	13	Tinggi	Ya
	14	Tinggi	Tidak
	15	Tinggi	Tidak

Meskipun masih dengan enkripsi sederhana kualitas *file share* yang dihasilkan tidak seluruhnya sempurna, namun hasil dari pengujian dengan enkripsi sudah jauh lebih baik dari pengujian tanpa enkripsi. Hal ini dapat dilihat dari distori suara yang hamper seluruhnya tinggi dan sebagian besar dari *file share* tidak dapat didengar pesan aslinya. Untuk menguji keabsahan dari enkripsi pada data, dilakukan juga pengujian rekonstruksi untuk kunci enkripsi yang berbeda dan sama dengan menggunakan jumlah partisipan 3 dan nilai batas 2 yang ditunjukkan pada Tabel 8.

Tabel 8. Hasil Pengujian Kecocokan MD5 Data Rekonstruksi dengan Data Asli menggunakan Kunci Enkripsi

Kombinasi *file*	**Kunci**	**MD5 sama dengan *file* original**
1 dan 2	10101010	Sama
1 dan 2	00000000	Tidak
2 dan 3	10101010	Sama
2 dan 3	10101011	Tidak

V. SIMPULAN DAN SARAN

Skema *secret sharing* Shamir pada data audio nilai *p* yang digunakan tidak dapat bernilai semabarang nilai prima, namun perlu sama dengan nilai ukuran *buffer*. Dari pengecekan MD5 untuk data asli dan data hasil rekonstruksi, skema *secret sharing* Shamir berhasil dilakukan untuk data audio WAVE. Dari pengujian yang telah dilakukan, dapat dilihat bahwa semakin ukuran besar *buffer* yang memproses data, maka kinerja *secret sharing* dengan skema Shamir lebih baik. hal ini ditandai dengan lebih cepatnya proses dan distori suara yang dihasilkan pada *file share* lebih tinggi. Selain itu, menggunakan enkripsi sederhana pada data sebelum dilakukan proses skema Shamir dapat meningkatkan distorsi pada *file share* dan jumlah *file share* yang tidak dapat terdengar pesan aslinya meningkat, meskipun belum sempurna.

Saran yang dapat diberikan dari eksperimen yang telah dilakukan lebih difokuskan pada kualitas *file share* hasil pembagian *secret* dari skema Shamir. Angka *byte* yang lebih besar lagi untuk ukuran *buffer* perlu diuji untuk lebih menyempurnakan kualitas *file share*. Selain itu, enkripsi yang dilakukan juga dapat berupa enkripsi yang lebih rumit dibandingkan dengan hany melakukan operasi *xor* dengan *byte* '10101010' saja. Selain teknik pemrosesan, data yang dipecah sebaiknya perlu diuji dengan melibatkan parameter *file* WAVE, seperti jumlah *channel* dan *framerate*, untuk diproses *secret sharing*. Ada baiknya juga, teknik *secret sharing* yang digunakan menggunakan teknik Interpolasi Langange.

UCAPAN TERIMA KASIH

Penulis mengucapkan terima kasih kepada Bapak Dr. Ir. Rinal Munir selaku dosen mata kuliah Kriptografi karena atas bimbingannya penulis mendapatkan bekal ilmu yang cukup untuk menulis makalah ini. Terakhir, penulis juga mengucapkan terima kasih kepada pihak-pihak yang secara langsung maupun tidak langsung telah membantu menyelesaikan makalah ini.

REFERENSI

[1] Chan, K.F.P. "Secret Sharing in Audio Steganography". 2015. International Journal of Science and Research.

[2] Patil, S., Chavan, T., Sangwan, P., Shashtri, P., Sunthwal, A. "Contemplating Audio Secret Sharing". 2014. International Journal of Advanced Research in Computer and Communication Engineering.

[3] Shamir, A. "How to share a secret". 1979. *Communications of the ACM*, vol. 22, no. 11, pp. 612-613.

[4] Munir, R. "Skema Pembagian data Rahasia (Secret Sharing Scheme)". 2017. Slide Kuliah IF4020 Kriptografi.

[5] Munir, R. "Algoritma Kriptografi Modern". 2017. Slide Kuliah IF4020 Kriptografi.

[6] Munir, R. “Algoritma MD5”. 2017. Slide Kuliah IF4020 Kriptografi.

[7] “Wave PCM Soundfile Format” http://soundfile.sapp.org/doc/WaveFormat/ diakses tanggal 16 Mei 2018.

PERNYATAAN

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

Bandung, 17 Mei 2018

Muhammad Gumilang (13514092)