Artikel ini telah dipresentasikan dalam *Innovative and Creative Information Technology Conference* (ICITech) dengan tema *"E-Transaction and Power Play"* yang diselenggarakan oleh Program Studi Teknik Informatika, Fakultas Teknologi Informasi, Universitas Kristen Satya Wacana – Salatiga, pada 24 November 2016.

Aplikasi Laporan Keuangan Akuntansi Bulog-Jakarta Menggunakan Algoritma MD5 dan RSA

Linda[1], Halim Agung[2]
[1,2]Teknik Informatika, Fakultas Teknologi dan Desain, Universitas Bunda Mulia
Jalan Lodan Raya No. 2, Jakarta 14430
Email : [2)]hagung@bundamulia.ac.id

Abstrak

Data yang bersifat rahasia dan aman merupakan suatu hal penting yang perlu adanya sistem agar dalam menyimpan dan mengirimkannya tidak dimodifikasi oleh pihak yang tidak bertanggung jawab, baik saat data tersebut tersimpan sebagai file di dalam komputer maupun saat dikirim melalui email. Tujuan penelitian ini adalah untuk membuat aplikasi yang dapat menjaga keamanan laporan keuangan yang menerapkan algoritma RSA dan MD5 dimana RSA diterapkan untuk keamanan data laporan keuangan dan MD5 diterapkan untuk mengamankan kunci yang dibentuk oleh algoritma RSA, serta membuktikan algoritma RSA dan MD5 dapat diterapkan kedalam aplikasi. Hasil dari penelitian ini adalah aplikasi laporan keuangan dengan algoritma MD5 dan RSA yang dapat mengamankan data laporan keuangan yang telah diuji dan dinyatakan valid untuk diterapkan.

Keywords: Kriptografi, MD5, RSA, Akuntansi

1. Pendahuluan

Transaksi keuangan merupakan pencatatan dalam sebuah jurnal. Jurnal adalah rincian semua transaksi keuangan dan akun-akun yang mempengaruhi transaksi tersebut. Semua transaksi bisnis akan dicatat dalam jurnal dengan menggunakan metode pembukuan *double-entry* atau *single-entry*. Biasanya, entri jurnal dimasukkan sesuai urutan kronologis atau berurutan atau dicatat sesuai dengan tanggal transaksi, dan saldo debet dimasukkan sebelum saldo kredit. Jurnal transaksi inilah yang kemudian di dasarkan untuk dilakukan proses keamanan data. Untuk menyelesaikan masalah tersebut maka digunakanlah konsep kriptografi. Kriptografi merupakan ilmu yang mempelajari cara mengamankan suatu data atau informasi. Pada kriptografi terdapat 2 tahap yaitu proses enkripsi dan dekripsi. Enkripsi merupakan proses yang dilakukan untuk mengubah pesan asli menjadi *ciphertext* sedangkan proses yang dilakukan untuk mengubah pesan tersembunyi menjadi pesan biasa adalah dekripsi. Pesan biasa atau pesan asli dapat disebut *plaintext* sedangkan pesan yang telah diubah atau disandikan supaya tidak mudah dibaca dapat disebut *ciphertext*. Terdapat banyak algoritma yang digunakan dalam kriptografi salah satunya Affine Chiper dan RC4[1]. Sedangkan sebagai salah satu penyelesaian yang digunakan untuk menjawab permasalahan ini adalah menggunakan algoritma RSA dan MD5 dimana algoritma RSA yang berfungsi untuk mengamankan data laporan keuangan sedangkan algoritma MD5 berfungsi sebagai penyandian pembentukan kunci.

2. Metodologi

Data dan referensi diambil dari beberapa buku yang berhubungan dengan kriptografi dan melakukan studi literatur yang berhubungan dengan metode yang dipakai dalam penelitian yang dilakukan.

2.1 Kriptografi

Kriptografi adalah ilmu yang mempelajari teknik enkripsi dimana data tersebut diacak menggunakan kunci enkripsi diubah menjadi data yang sulit dibaca oleh orang lain yang tidak mempunyai kunci dekripsi[2]. Dekripsi untuk mendapatkan kembali data asli dapat menggunakan kunci dekripsi. Proses enkripsi dapat dilakukan dengan menggunakan suatu algoritma dengan beberapa parameter. Biasanya algoritma yang digunakan tidak dirahasiakan, bahkan enkripsi yang mengandalkan kerahasiaan suatu algoritma dianggap sebagai sesuatu yang tidak baik. Rahasia terletak pada beberapa parameter yang digunakan, jadi kunci yang digunakan ditentukan oleh parameter. Parameter yang harus dirahasiakan adalah parameter yang menentukan kunci dekrpsi. Ilustrasi dari penjelasan diatas dapat dilihat pada gambar 1.

Gambar 1. Enkripsi secara umum

2.2 *Message Digest* 5 (MD5)

Message Digest 5 (MD5) adalah salah satu alat untuk memberi garansi bahwa pesan yang dikirim akan sama dengan pesan yang diterima, hal ini dengan membandingkan ‘sidik jari’ atau ‘intisari pesan’ kedua pesan tersebut. MD5 merupakan pengembangan dari MD4 dimana terjadi penambahan satu ronde. MD5 memproses teks masukan ke dalam blok-blok bit sebanyak 512 bit, kemudian dibagi ke dalam 32 bit sub blok sebanyak 16 buah. Keluaran dari MD5 berupa 4 buah blok yang masing – masing 32 bit yang mana akan menjadi 128 bit yang biasa disebut nilai *hash*[3].

Setiap pesan yang akan dienkripsi, terlebih dahulu dicari berapa banyak bit yang terdapat pada pesan. Kita anggap sebanyak b bit. Di sini b adalah bit non negatif *integer*, b bisa saja nol dan tidak harus selalu kelipatan delapan. Langkah – langkah untuk pembuatan *message digest* secara umum adalah penambahan bit-bit penyangga atau *padding bits*, penambahan nilai panjang pesan semula, inisialisasi penyangga atau *buffer message digest* dan pengolahan pesan dalam blok berukuran 512 bit.

2.3 *Rivest Shamir Adleman* (RSA)

Algoritma RSA merupakan salah satu algoritma *public key* yang populer dipakai dan bahkan masih dipakai hingga saat ini. Kekuatan algoritma ini terletak pada proses eksponensial, dan pemfaktoran bilangan menjadi 2 bilangan prima yang hingga kini perlu waktu yang lama untuk melakukan pemfaktorannya[4].

Tahun 1978, Len Adleman, Ron Rivest dan Adi Shamir mempublikasikan sistem RSA. Semula sistem ini dipatenkan di Amerika Serikat dan seharusnya masa paten habis tahun 2003, akan tetapi RSA *Security* melepaskan hak paten setelah 20 September 2000. Sebetulnya sistem serupa telah dilaporkan oleh Clifford Cocks tahun 1973 meskipun informasi mengenai ini baru dipublikasi tahun 1997 karena merupakan hasil riset yang diklasifikasikan sangat rahasia oleh pemerintah Britania Raya (Clifford Cocks bekerja untuk GCHQ, suatu badan di Britania Raya yang fungsinya serupa dengan fungsi NSA di Amerika Serikat), jadi validitas paten patut dipertanyakan karena adanya *prior art*[5]. RSA terbagi menjadi tiga proses, yaitu pembangkitan kunci, enkripsi, dan dekripsi. Dasar proses enkripsi dan dekripsi pada algoritma RSA yaitu konsep bilangan prima aritmatika modulo. Kunci enkripsi tidak dirahasiakan dan diberikan kepada umum atau disebut *public key*, sedangkan kunci untuk dekripsi bersifat rahasia atau disebut *private key*[6].

Skema pada RSA mengadopsi skema block chiper, dimana sebelum enkripsi dilakukan, plainteks yang ada dibagi dalam blok dengan ukuran panjang yang sama, dimana plainteks dan cipherteks nya berupa integer atau bilangan bulat antara 1 sampai n, dimana n biasanya berukuran 1024 bit, dan panjang bloknya sendiri berukuran lebih kecil atau sama dengan log(n) +1 dengan basis 2.

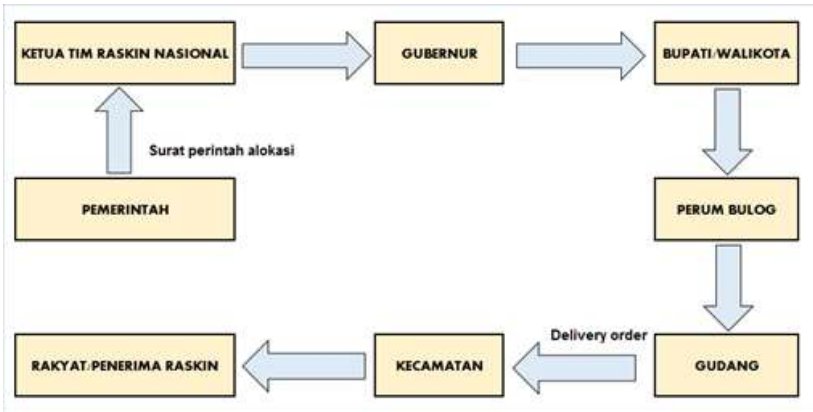
2.4 Penelitian Terdahulu

Pada penelitian yang dilakukan oleh Halim Agung dan Ferry dalam papernya membahas mengenai penerapan hybrid cryptosystem yang mengadopsi algoritma RSA pada penelitiannya mengungkapkan bahwa semua file yang sudah dienkripsi dapat dikembalikan ke file aslinya dalam proses dekripsi dan tidak mengalami perubahan dengan tingkat keberhasilan 100% serta ukuran file yang semakin besar dapat mempengaruhi lama proses enkripsi dan dekripsi[7].

3. Hasil dan Pembahasan

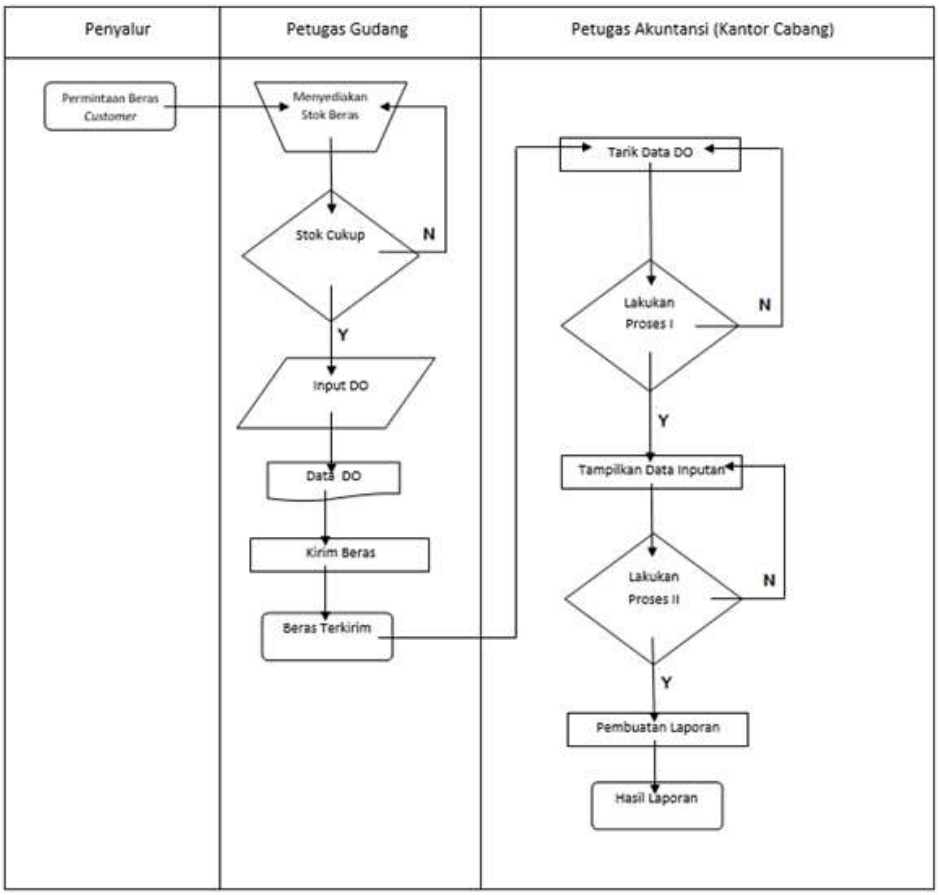
3.1 Analisis Sistem Berjalan

Penyaluran raskin berawal dari surat perintah alokasi dari Pemerintah kepada Ketua Tim Raskin Nasional, yang akan dilanjutkan kepada Gubernur dan Bupati/Walikota, yang kemudian akan diterima oleh Perum Bulog. Setelah diterima oleh Perum Bulog, kantor pusat akan menerbitkan surat perintah pengeluaran barang untuk gudang, lalu pihak gudang akan mengirim raskin ke setiap kecamatan. Setelah raskin diterima oleh pihak kecamatan, maka raskin siap dijual kepada rakyat. Alur bisnis proses dari Perum Bulog dapat dilihat pada gambar 2.



Gambar 2. Alur Bisnis Proses Perum Bulog

Adapun *flowchart* dari aplikasi yang digunakan oleh perum bulog terdapat pada gambar 3.



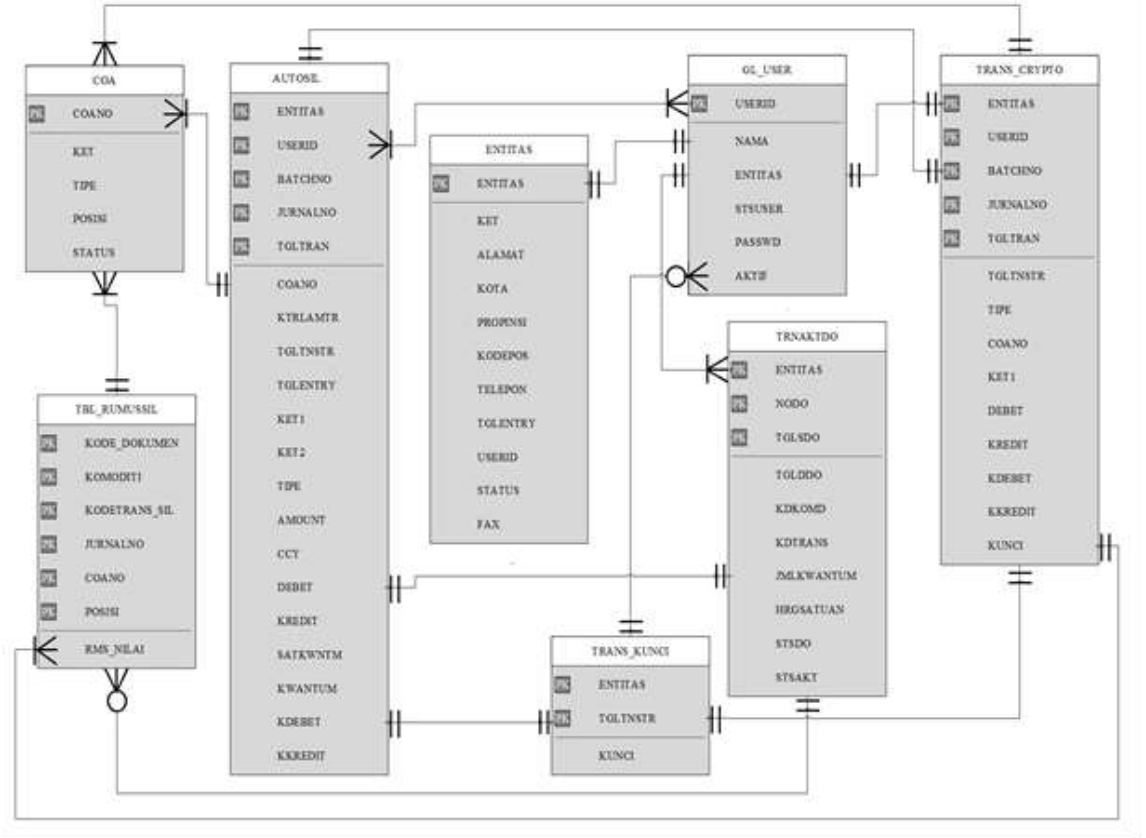
Gambar 3. Flowchart Aplikasi

Gambar 3 menjelaskan bahwa alur sistem dimulai dari bagian penyalur, bagian penyalur mengirimkan permintaan beras kepada gudang, lalu pihak gudang akan menyediakan stok beras, jika stok beras cukup untuk memenuhi permintaan beras tersebut, maka petugas gudang harus melakukan proses input dokumen *delivery*

order, namun jika stok beras belum mencukupi maka petugas gudang harus menyiapkan stok beras sampai stok beras di gudang cukup untuk memenuhi permintaan beras tersebut. Lalu pihak gudang mengirim beras tersebut. Data DO yang telah dimasukkan kedalam sistem akan ditarik secara otomatis oleh bagian akuntansi, setelah itu akan dilakukan proses pembukuan secara otomatis agar menjadi sebuah laporan keuangan.

3.2 Perancangan *Database*

Perancangan ERD bertujuan untuk memberikan gambaran mengenai *database* yang akan digunakan untuk menyimpan data-data yang diperlukan dalam aplikasi laporan keuangan ini seperti ditunjukkan pada gambar 4

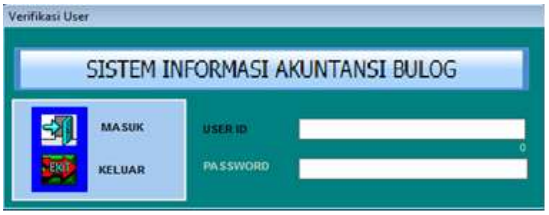


Gambar 4. ERD Aplikasi Laporan Keuangan

3.3 Kontruksi Antarmuka

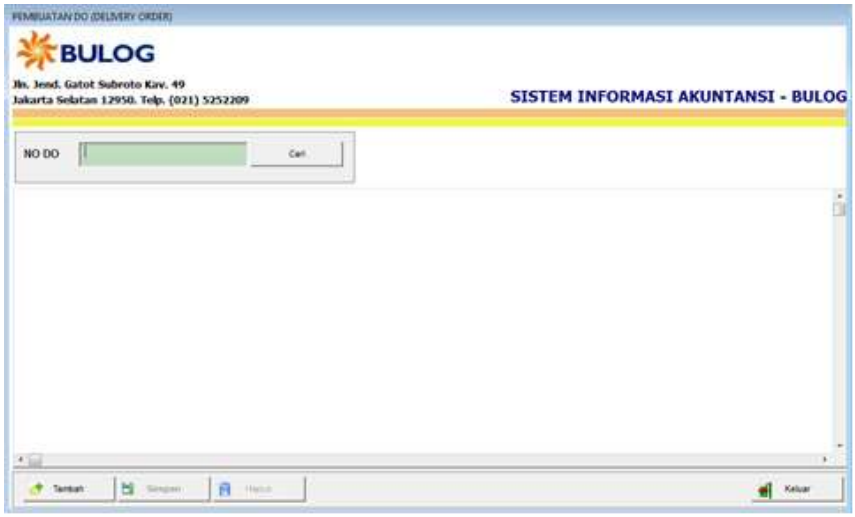
Bagian ini menjelaskan implementasi atau konstruksi tampilan dari aplikasi laporan keuangan ini.

1. Tampilan Halaman *Login*
   Pada saat pengguna aplikasi akan masuk kedalam sistem maka pengguna harus memasukkan *user* ID dan *password* yang juga memiliki hak akses pada aplikasi seperti yang ditunjukkan pada gambar 5.



Gambar 5. Tampilan Halaman *Login*

2. Tampilan Halaman Utama Aplikasi Laporan Keuangan
   Pada tampilan dari halaman utama aplikasi, jika ingin melihat dokumen DO yang di *entry* sebelumnya *user* hanya perlu memasukkan nomor dokumen di kolom atas dan memilih *button* cari, namun jika ingin membuat dokumen baru maka *user* harus memilih button tambah, setelah itu akan muncul *form input* DO seperti yang ditunjukkan pada gambar 6.



Gambar 6. Tampilan Halaman Utama

3. Tampilan Implementasi Proses *Define Order* (DO) 1
   Pada tampilan dari menu proses 1, dimana data DO yang sudah ditarik akan dilakukan proses pembukuan yang pertama (jurnal akuntansi) oleh *staff accounting* seperti yang ditunjukkan pada gambar 7.

Gambar 7. Tampilan Proses *Define Order* (DO) 1

4. Tampilan Implementasi Proses *Define Order* (DO) 2
   Pada tampilan dari menu proses 2, setelah proses 1 dilakukan maka hanya *head accounting* saja yang dapat melakukan proses kedua. Setelah proses kedua dilakukan sistem secara otomatis akan meminta sebuah kata kunci atau *key* untuk mengunci dan membuka laporan tersebut seperti yang ditunjukkan pada gambar 8.

Gambar 8. Tampilan Proses *Define Order* (DO) 2

5. Tampilan Implementasi Halaman Laporan
   Pada tampilan dari menu laporan, menu ini hanya dapat diakses oleh *head accounting* dan kantor pusat saja. Tetapi saat *user* ingin melihat laporan keuangan, *user* wajib memasukkan sebuah kata kunci atau *key* yang sebelumnya dimasukkan saat melakukan proses kedua. Jika *user* salah memasukkan *key* sebanyak tiga kali, maka sistem akan otomatis *log-off* seperti yang ditunjukkan pada gambar 9..

Gambar 9. Tampilan Halaman Laporan

3.4 *Pseudocode* Algoritma MD5

MD5 memproses teks masukan ke dalam blok-blok bit sebanyak 512 bit, kemudian dibagi ke dalam 32 bit sub blok sebanyak 16 buah. Keluaran dari MD5 berupa 4 buah blok yang masing – masing 32 bit yang mana akan menjadi 128 bit yang biasa disebut nilai *hash*. Setiap pesan yang akan dienkripsi, terlebih dahulu dicari berapa banyak bit yang terdapat pada pesan. Kita anggap sebanyak b bit. Di sini b adalah bit non negatif *integer*, b bisa saja nol dan tidak harus selalu kelipatan delapan. MD-5 terdapat empat buah word 32 bit *register* yang berguna untuk menginisialisasi *message digest* pertama kali. Register – register ini di inisialisasikan dengan bilangan hexadesimal. Tahapan Algoritma MD5 dapat dilihat pada gambar 10.

```
Constant ulong HP_ALGID    = 1
Constant ulong HP_HASHVAL  = 2
Constant ulong HP_HASHSIZE = 4
ULong lul_hProv, lul_hHash, lul_error, lul_DataLen
Blob lblob_hash, lblob_result
String ls_msgtext, ls_null

// Get handle to a key container
SetNull(ls_null)
If Not CryptAcquireContext(lul_hProv, ls_null, iCryptoProvider, &
         iProviderType, CRYPT_VERIFYCONTEXT + CRYPT_MACHINE_KEYSET) Then
   of_GetLastError(lul_error, ls_msgtext)
   SignalError(lul_error, "CryptAcquireContext:~r~n~r~n" + ls_msgtext)
End If

// Create the hash object
If Not CryptCreateHash(lul_hProv, iHashAlgorithm, 0, 0, lul_hHash) Then
   of_GetLastError(lul_error, ls_msgtext)
   CryptReleaseContext(lul_hProv, 0)
   SignalError(lul_error, "CryptCreateHash:~r~n~r~n" + ls_msgtext)
End If

// Add data to the hash object
If Not CryptHashData(lul_hHash, ablob_data, Len(ablob_data), 0) Then
   of_GetLastError(lul_error, ls_msgtext)
   CryptDestroyHash(lul_hHash)
   CryptReleaseContext(lul_hProv, 0)
   SignalError(lul_error, "CryptHashData:~r~n~r~n" + ls_msgtext)
End If
// Determine size of hash value
CryptGetHashParam(lul_hHash, HP_HASHVAL, lblob_hash, lul_DataLen, 0)

// Get the hash value
lblob_hash = Blob(Space(lul_DataLen))
If Not CryptGetHashParam(lul_hHash, HP_HASHVAL, lblob_hash, lul_DataLen, 0) Then
   of_GetLastError(lul_error, ls_msgtext)
   CryptDestroyHash(lul_hHash)
   CryptReleaseContext(lul_hProv, 0)
   SignalError(lul_error, "CryptGetHashParam:~r~n~r~n" + ls_msgtext)
End If
lblob_result = BlobMid(lblob_hash, 1, lul_DataLen)

// Release crypto objects
CryptDestroyHash(lul_hHash)
CryptReleaseContext(lul_hProv, 0)

// return the result in hex
Return of_Blob2Hex(lblob_result)
```

Gambar 10. *Pseudocode* MD5

3.5 *Pseudocode* Algoritma Enkripsi RSA

Tahapan yang dilakukan pada aplikasi untuk algoritma dekripsi RSA dapat dilihat pada gambar 11.

```
ULong lul_hProv, lul_hRSA, lul_hKey, lul_error, lul_DataLen, lul_BufLen
Blob lblob_data, lblob_buffer, lblob_result
String ls_null, ls_msgtext

// Get handle to a key container
SetNull(ls_null)
If Not CryptAcquireContext(lul_hProv, ls_null, iCryptoProvider, &
         iProviderType, CRYPT_VERIFYCONTEXT + CRYPT_MACHINE_KEYSET) Then
   of_GetLastError(lul_error, ls_msgtext)
   SignalError(lul_error, "CryptAcquireContext:~r~n~r~n" + ls_msgtext)
End If

// Create the RSA object
//If Not CryptCreateRSA(lul_hProv, iRSAAlgorithm, 0, 0, lul_hRSA) Then
   of_GetLastError(lul_error, ls_msgtext)
   CryptReleaseContext(lul_hProv, 0)
   SignalError(lul_error, "CryptCreateRSA:~r~n~r~n" + ls_msgtext)
//End If

// RSA the password
lblob_data = Blob(as_password)
//If Not CryptRSAData(lul_hRSA, lblob_data, Len(lblob_data), 0) Then
   of_GetLastError(lul_error, ls_msgtext)
   //CryptDestroyRSA(lul_hRSA)
   CryptReleaseContext(lul_hProv, 0)
   SignalError(lul_error, "CryptRSAData:~r~n~r~n" + ls_msgtext)
//End If

// Derive a session key from the RSA object
If Not CryptDeriveKey(lul_hProv, iEncryptAlgorithm, lul_hRSA, 0, lul_hKey) Then
   of_GetLastError(lul_error, ls_msgtext)
   //CryptDestroyRSA(lul_hRSA)
   //CryptReleaseContext(lul_hProv, 0)
   SignalError(lul_error, "CryptDeriveKey:~r~n~r~n" + ls_msgtext)
End If
// allocate buffer space
lul_DataLen = Len(ablob_data)
lblob_buffer = ablob_data + Blob(Space(lul_DataLen))
lul_BufLen = Len(lblob_buffer)

If ab_encrypt Then
   // Encrypt data
   If CryptEncrypt(lul_hKey, 0, True, 0, &
            lblob_buffer, lul_DataLen, lul_BufLen) Then
      lblob_result = BlobMid(lblob_buffer, 1, lul_DataLen)
   Else
      of_GetLastError(lul_error, ls_msgtext)
      SignalError(lul_error, &
         "CryptEncrypt:~r~n~r~n" + ls_msgtext)
   End If

End If

// Release crypto objects
CryptDestroyKey(lul_hKey)
//CryptDestroyRSA(lul_hRSA)
CryptReleaseContext(lul_hProv, 0)

Return lblob_result
```

Gambar 11. *Pseudocode* Algoritma Enkripsi RSA

3.6 *Pseudocode* Algoritma Dekripsi RSA

Tahapan yang dilakukan pada aplikasi untuk algoritma dekripsi RSA dapat dilihat pada gambar 12.

```
ULong lul_hProv, lul_hRSA, lul_hKey, lul_error, lul_DataLen, lul_BufLen
Blob lblob_data, lblob_buffer, lblob_result
String ls_null, ls_msgtext

// Get handle to a key container
SetNull(ls_null)
If Not CryptAcquireContext(lul_hProv, ls_null, iCryptoProvider, &
         iProviderType, CRYPT_VERIFYCONTEXT + CRYPT_MACHINE_KEYSET) Then
   of_GetLastError(lul_error, ls_msgtext)
   SignalError(lul_error, "CryptAcquireContext:~r~n~r~n" + ls_msgtext)
End If

// Create the RSA object
//If Not CryptCreateRSA(lul_hProv, iRSAAlgorithm, 0, 0, lul_hRSA) Then
   of_GetLastError(lul_error, ls_msgtext)
   CryptReleaseContext(lul_hProv, 0)
   SignalError(lul_error, "CryptCreateRSA:~r~n~r~n" + ls_msgtext)
//End If

// RSA the password
lblob_data = Blob(as_password)
//If Not CryptRSAData(lul_hRSA, lblob_data, Len(lblob_data), 0) Then
   of_GetLastError(lul_error, ls_msgtext)
   //CryptDestroyRSA(lul_hRSA)
   CryptReleaseContext(lul_hProv, 0)
   SignalError(lul_error, "CryptRSAData:~r~n~r~n" + ls_msgtext)
//End If

// Derive a session key from the RSA object
If Not CryptDeriveKey(lul_hProv, iEncryptAlgorithm, lul_hRSA, 0, lul_hKey) Then
   of_GetLastError(lul_error, ls_msgtext)
   //CryptDestroyRSA(lul_hRSA)
   //CryptReleaseContext(lul_hProv, 0)
   SignalError(lul_error, "CryptDeriveKey:~r~n~r~n" + ls_msgtext)
End If
 // allocate buffer space
 lul_DataLen = Len(ablob_data)
 lblob_buffer = ablob_data + Blob(Space(lul_DataLen))
 lul_BufLen = Len(lblob_buffer)

    // Decrypt data
    If CryptDecrypt(lul_hKey, 0, True, 0, &
             lblob_buffer, lul_DataLen) Then
       lblob_result = BlobMid(lblob_buffer, 1, lul_DataLen)
    Else
       of_GetLastError(lul_error, ls_msgtext)
       SignalError(lul_error, &
          "CryptDecrypt:~r~n~r~n" + ls_msgtext)
    End If

 // Release crypto objects
 CryptDestroyKey(lul_hKey)
 //CryptDestroyRSA(lul_hRSA)
 CryptReleaseContext(lul_hProv, 0)

 Return lblob_result
```

Gambar 12. *Pseudocode* Algoritma Dekripsi RSA

3.7 Pengujian Aplikasi

Pengujian Aplikasi yang dilakukan dengan menggunakan *white box testing* yang dilakukan untuk menguji apakah algoritma yang digunakan dapat mengamankan data laporan keuangan. Hasil pengujian dapat dilihat pada gambar 13.

		No.	Tool Type	CIPHERTEXT RSA /MD5	Plaintext	Kesimpulan
Cryptanalysis Tools	EverCrack	1	Shift Char(Unicode) Cipher	e115551ff9d846dc66c51fd8a011bd93	▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒	VALID
		2	Shift Char(Unicode) Cipher	d1b2bd8171937165bc0cf06425041d10	▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒	VALID
		3	Shift Byte(ASCII) Cipher	e115551ff9d846dc66c51fd8a011bd93	¥qquuuq¦¦y¤xtv¤£vv£uq¦¤xi¡pqq¢¤ys	VALID
		4	Shift Byte(ASCII) Cipher	d1b2bd8171937165bc0cf06425041d10	¤q¢r¢¤xqwqyswqvuc£p£¦pvtruptq¤qp	VALID
		5	Alpha Shift Cipher	e115551ff9d846dc66c51fd8a011bd93	NULL	VALID
		6	Alpha Shift Cipher	d1b2bd8171937165bc0cf06425041d10	NULL	VALID
		7	Key Shift Char(Unicode) Cipher	e115551ff9d846dc66c51fd8a011bd93	Ø§©¤ÉÉ«É~§ÇÕ#ÖØÉ~ÕÉÇØ~	VALID
		8	Key Shift Char(Unicode) Cipher	d1b2bd8171937165bc0cf06425041d10	×Å¤ÇØ«É§#§Ç×ÉÉÉ¯¥¦Ø¤	VALID
		9	Key Shift Byte(ASCII) Cipher	e115551ff9d846dc66c51fd8a011bd93	Ø§©¤ÉÉ«É~§ÇÕ#ÖØÉ~ÕÉÇØ~	VALID
		10	Key Shift Byte(ASCII) Cipher	d1b2bd8171937165bc0cf06425041d10	×Å¤ÇØ«É§#§Ç×ÉÉÉ¯¥¦Ø¤	VALID
		11	Alpha Key Shift Cipher	e115551ff9d846dc66c51fd8a011bd93	w115551jh9h846ft66u51wh8s011fw93	VALID
		12	Alpha Key Shift Cipher	d1b2bd8171937165bc0cf06425041d10	v1d2fw8171937165fv0gh06425041w10	VALID
		13	Substitution Cipher	e115551ff9d846dc66c51fd8a011bd93	e115551ff9d846d366351fd810112d93	VALID
		14	Substitution Cipher	d1b2bd8171937165bc0cf06425041d10	d1222d81719371652303f06425041d10	VALID
		15	Alpha Substitution Cipher	e115551ff9d846dc66c51fd8a011bd93	e115551ff9d846d366351fd810112d93	VALID
		16	Alpha Substitution Cipher	d1b2bd8171937165bc0cf06425041d10	d1222d81719371652303f06425041d10	VALID
	Decipher Mode	17	DSS (Decipher Mode)	e115551ff9d846dc66c51fd8a011bd93	bÇ1╡æŸl"M‡i4,"ó ³J•GO‡¡L4Œ4ó,ÿ•4	VALID
		18	DSS (Decipher Mode)	d1b2bd8171937165bc0cf06425041d10	cÇb╡±■åA*?"¤vçÆ◀¶D8@g%sOÑÿ7	VALID
		19	Caesar Cipher (Decipher Mode)	e115551ff9d846dc66c51fd8a011bd93	^**...*_2|1-/]\/\.*_]1Z)**[]2,	VALID
		20	Caesar Cipher (Decipher Mode)	d1b2bd8171937165bc0cf06425041d10	]*[+[]1*0*2,0*/.[\j_)/-+.)-*[*}	VALID

Gambar 13. Hasil Pengujian *White Box Testing*

Gambar 13 menunjukkan hasil pengujian *White Box Testing* yang dilakukan oleh penulis dengan menggunakan *Cryptanalysis Tools*. Pada uji ini jumlah *Tools* yang digunakan sebanyak 20 kali, yang digunakan pada *ciphertext* untuk mendapatkan *plaintext*, sehingga di peroleh pengujian keberhasilan metode = 20/20 x 100% = 100%

4. Kesimpulan dan Saran

4.1 Kesimpulan

Berdasarkan hasil penelitian yang telah dilakukan dalam penelitian ini, maka dapat diambil simpulan bahwa implementasi algoritma MD5 dan RSA dapat diterapkan kedalam aplikasi laporan keuangan untuk mengamankan data laporan keuangan yang bersifat rahasia yang telah ditunjukkan pada keberhasilan yang dilakukan pada tahapan pengujian *white box* yang telah dilakukan.

4.2 Saran

Saran-saran yang dapat dilakukan untuk pengembangan aplikasi ini adalah sebagai berikut :

1. Saat ini, menu *input data* DO yang dilakukan di gudang, bisa diubah bahkan dihapus, walaupun data tersebut sudah ditarik dan dilakukan proses 1 dan proses 2 oleh bagian akuntansi. Jadi sebaiknya data DO tersebut tidak dapat diubah ataupun dihapus jika sudah ditarik dan dilakukan proses 1 dan proses 2 oleh bagian akuntansi.
2. Sebaiknya ada pemberitahuan secara sistem di aplikasi Perum Bulog apabila data DO telah di *input* atau di *entry* oleh bagian gudang.

5. Daftar Pustaka

[1] Agung, Halim. Budiman. (2015). Implementasi *Affine Chiper* dan RC4 Pada Enkripsi File Tunggal, Prosiding SNATIF. ISBN: 978-602-1180-21-1

[2] Kromodimoeljo, Sentot. (2009). Teori dan Aplikasi Kriptografi. Jakarta : SPK IT Consulting.

[3] Sofwan, Aghus. (2011), Aplikasi Kriptografi Dengan Algoritma *Message-Digest* (MD5), Diponogero.

[4] Firmansyah, Eka Risky. (2012), Algoritma Kriptografi Dan Contohnya, Jakarta.

[5] Ayuningtyas, Arinta Nugrahani. (2012), Metode Kriptografi RSA Pada *Priority Dealer* Untuk Layanan Penjualan dan Pemesanan *Handphone* Berbasis J2ME, Surabaya.

[6] Devha, Chandra Putra. (2013). Pengamanan Pesan Rahasia Menggunakan Algoritma Kriptografi Rivest Shamir Adleman (RSA), Bandung.

[7] Agung, Halim. Ferry. (2016). Kriptografi Menggunakan *Hybrid Cryptosystem* dan *Digital Signature*, Jurnal JATISI Vol 3. No.1.