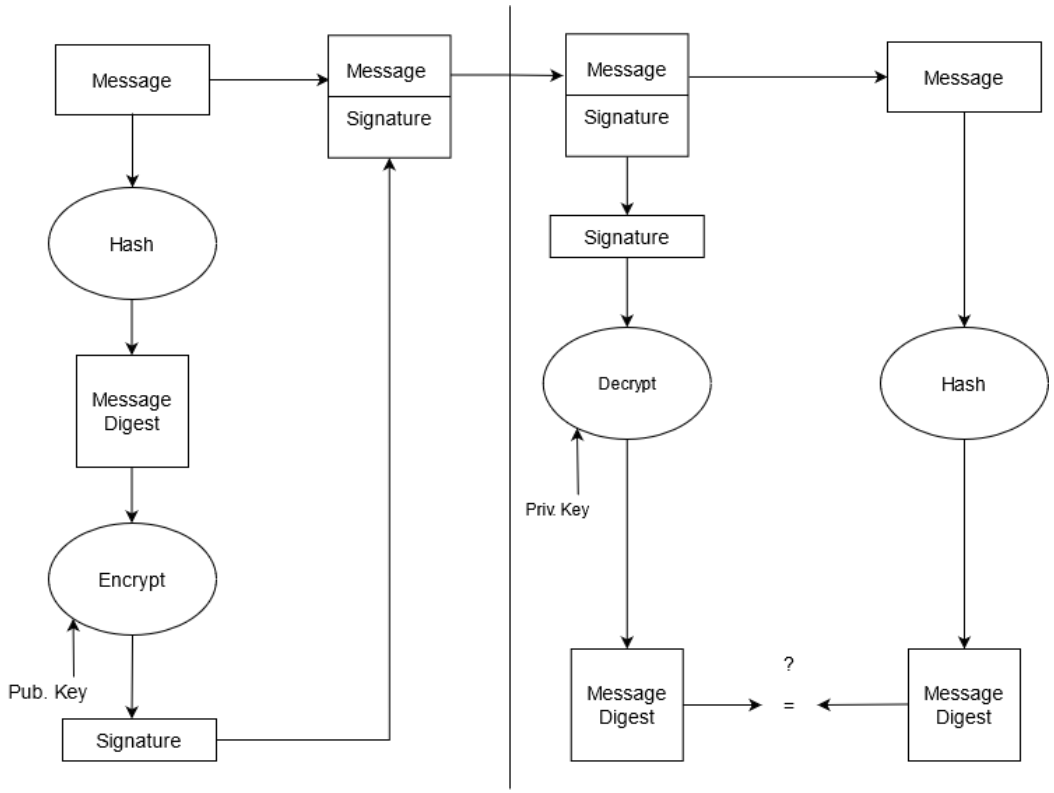
BAB III
METODE PENELITIAN

3.1 Identifikasi Masalah

Faktor keamanan pada proses *digital signature* dalam mengotentikasi sertifikat digital merupakan hal yang penting, diperlukan proses *hash* dan skema penandatanganan yang baik agar tidak terjadi pemalsuan sertifikat digital. Dalam praktiknya algoritma hash yang digunakan adalah algoritma SHA dan skema penandatanganan yang biasanya digunakan adalah RSA, Namun dengan skema penandatanganan RSA masih memungkinkan terjadinya kriptanalisis pada kunci privat RSA sehingga *digital signature* dengan bentuk nilai hash masih dapat diketahui. Kriptanalisis pada RSA ini dilakukan dengan cara memfaktorkan kunci publik sehingga dapat diketahui kunci privat dari hasil pemfaktoran tersebut. Oleh karena itu, perlu dilakukan peningkatan keamanan dengan menggunakan algoritma Diffie-Hellman sehingga kunci publik dari RSA tidak dapat dikriptanalisis.

3.2 Model Dasar

Model dasar *digital signature* menggunakan fungsi hash SHA-1 dengan skema penandatanganan RSA untuk mengetahui skema *digital signature* dapat dilihat pada Gambar 3.1



Gambar 3. 1 Skema *Digital Signature*

Dalam hal penandatanganan sertifikat digital, input *message* berupa file sertifikat digital, pada model dasar biasanya digunakan SHA-1 untuk melakukan hash sehingga bisa diperoleh *message digest* sementara enkripsi dan dekripsinya menggunakan RSA. *Message digest* yang didapatkan dari SHA-1 memiliki panjang 160-bit atau sebanyak 40 karakter *message digest*.

Gambar 3. 2 Contoh Sertifikat Digital

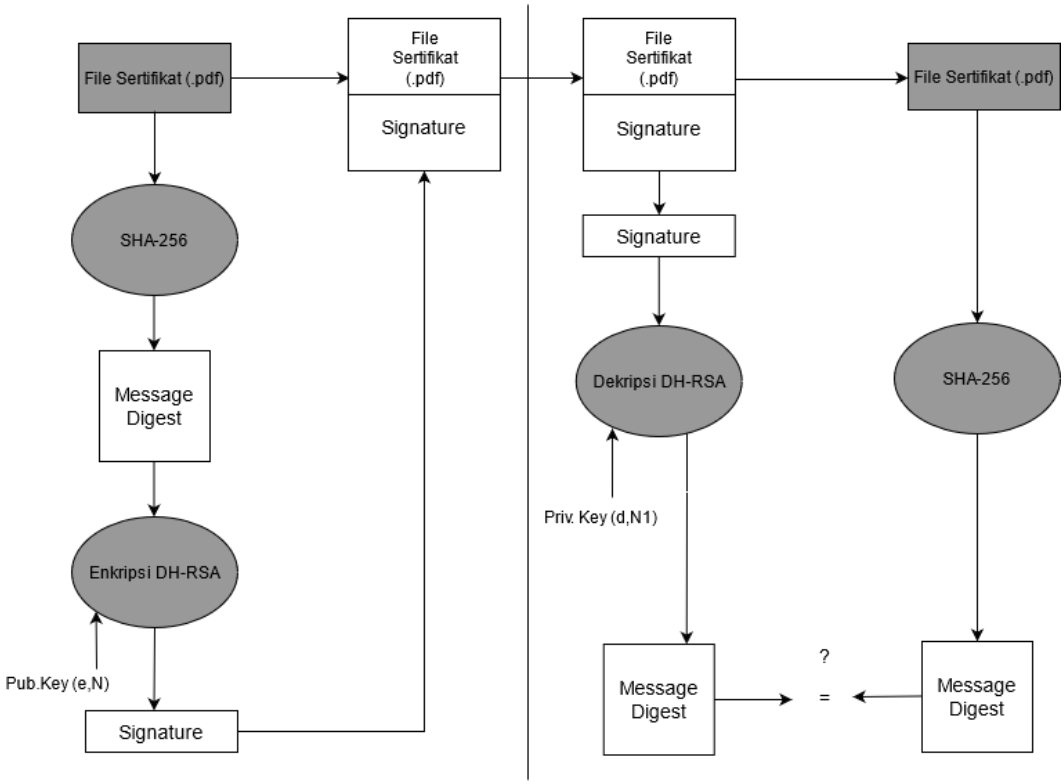
Untuk memahami bagaimana cara kerja model dasar akan diberi contoh dengan cara mencari nilai hash menggunakan SHA-1 pada sertifikat digital Gambar 3.2 di atas. Dari proses hash diperoleh *message digest* "9b0b6d48bde6646fc1c58be79e79bf0e517c2911", langkah selanjutnya adalah mengenkripsi *message digest* dengan RSA. Untuk contoh akan digunakan kunci publik $(e, n) = (17,77)$ dan kunci privat $(d, n) = (53,77)$ diperoleh cipherteks "292127211054392016054711212913182214223756215471553615529 215440161371455228291414", cipherteks dari RSA ini disebut *signature*. Selanjutnya *signature* dikirim beserta dengan file sertifikat digital. Untuk mengecek keaslian pesan, *signature* harus didekripsi dan file sertifikat harus di hash, jika hasil dekripsi sama dengan *message digest* file sertifikat digital maka file sertifikat dikatakan asli, jika berbeda maka sertifikat palsu.

Model *digital signature* SHA-1 dengan skema penandatanganan RSA ini memiliki keamanan yang tinggi, namun pada tahun 2005 ditemukan *collision attack* pada SHA-1 sehingga model *digital signature* dengan SHA-1 diragukan oleh banyak

orang, belum lagi skema penandatanganan menggunakan RSA masih bisa di serang dengan cara memfaktorkan kunci publik dari RSA.

3.3 Pengembangan Model

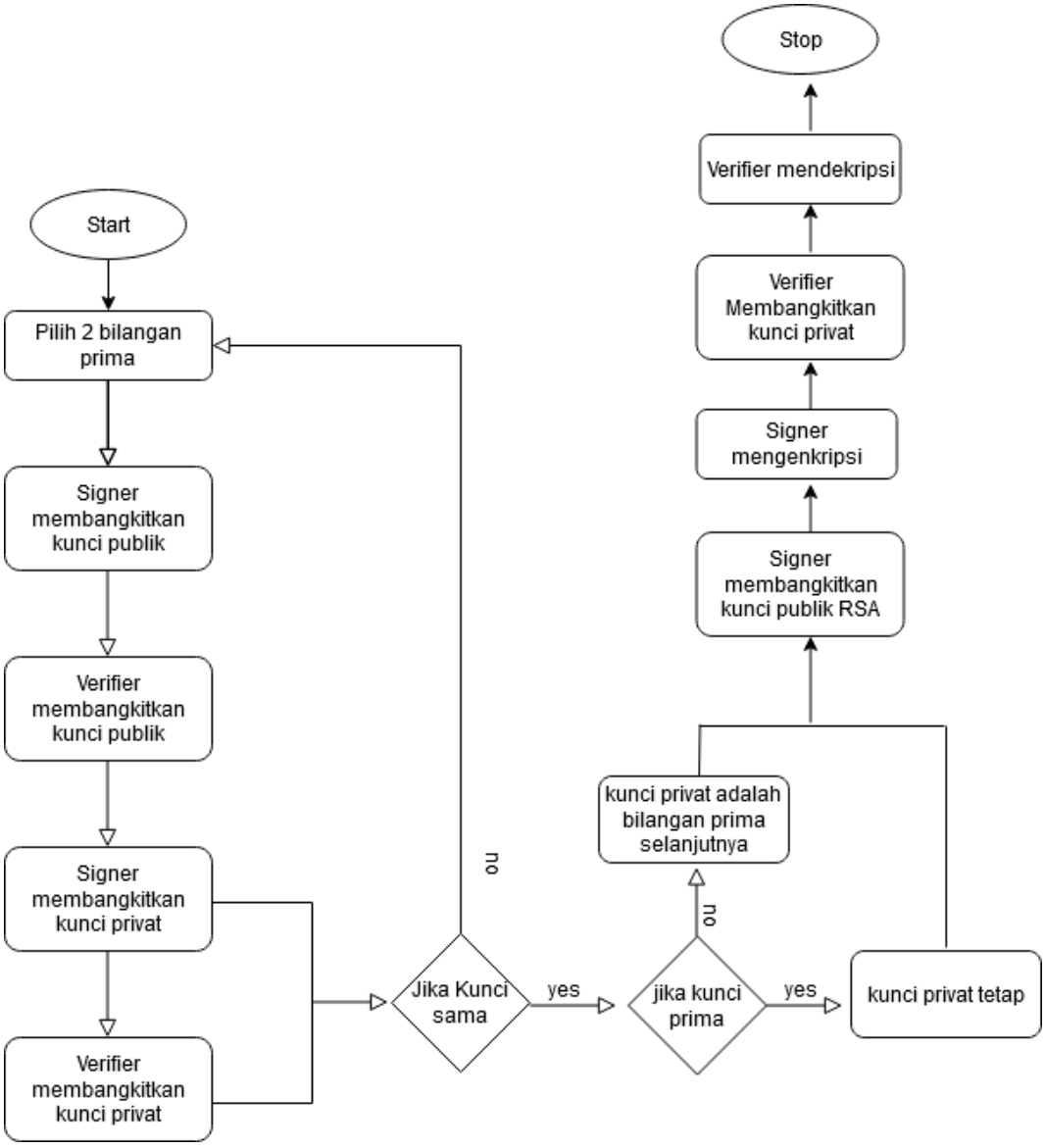
Umumnya *digital signature* diterapkan dalam pengecekan keaslian pesan teks menggunakan SHA-1 namun pada SHA-1 ditemukan penyerangan berupa *collision attack*, selain itu enkripsi yang digunakan dalam *digital signature* ini adalah RSA namun pada pengimplementasiannya RSA masih bisa di kriptanalisis dengan cara memfaktorkan kuncinya. Oleh karena itu penelitian ini menggunakan algoritma SHA-256 dan skema enkripsi Diffie-Hellman-RSA. Pengembangan model tersebut bisa digambarkan dalam Gambar 3.3



Gambar 3. 3 Skema Diffie-Hellman-RSA

Pada Gambar 3.3, secara garis besar terdapat 3 proses, yaitu proses hash dengan SHA-256, proses enkripsi dan dekripsi dengan Diffie-Hellman-RSA, dan proses autentikasi *message digest*.

Proses enkripsi dan dekripsi dijelaskan dengan diagram alir berikut



Gambar 3. 4 Diagram Alir Diffie-Hellman-RSA

Proses autentikasi *message digest* dilakukan dengan cara menyamakan *message digest* hasil dekripsi dengan hasil hash oleh verifier.

3.4 Konstruksi Program Aplikasi

Program aplikasi dibuat dengan menggunakan bahasa pemrograman Python. Dalam pembuatan aplikasi dibantu juga dengan menggunakan *library* dari Python, *library* ini akan digunakan untuk melakukan *hashing* dan membuat tampilan dari program aplikasi.

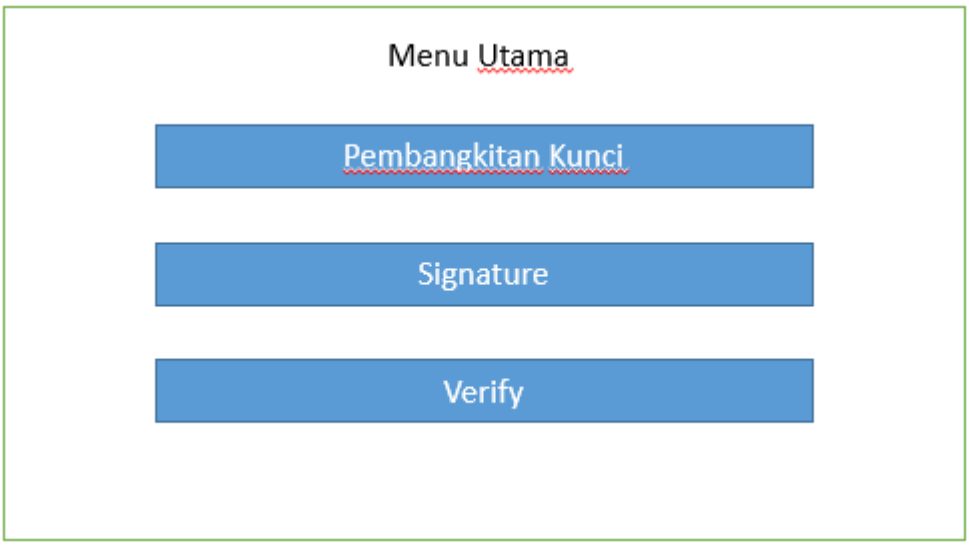
3.4.1 Input dan Output

Program aplikasi yang dibuat memiliki fungsi untuk melakukan hash pada file sertifikat digital yang memiliki ekstensi .pdf, membangkitkan kunci, mengenkripsi nilai hash, mendekripsi *signature*, serta mengecek autentikasi file.

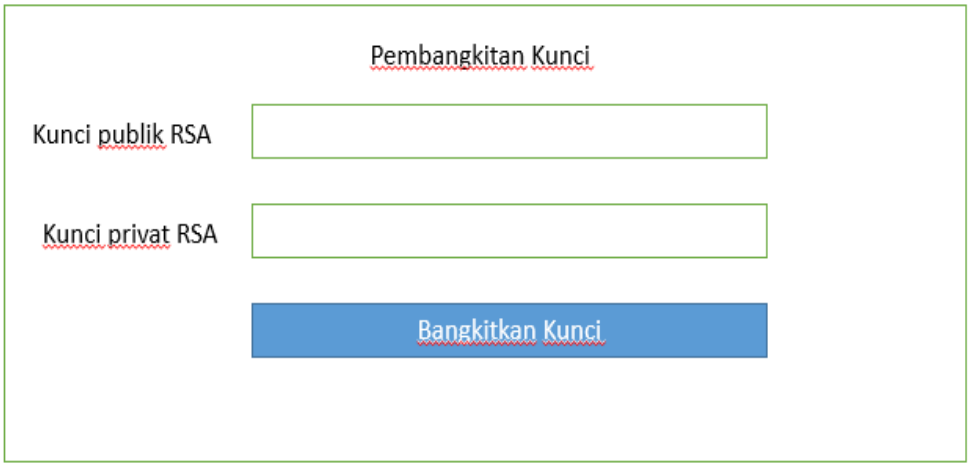
Input dari *signer* adalah kunci publik, file sertifikat digital, serta nilai hash untuk dienkripsi, sedangkan outputnya adalah nilai hash dan *signature*. Untuk verifier inputnya adalah kunci privat file sertifikat digital, serta *signature* untuk di dekripsi, sedangkan outputnya berupa nilai hash dari file dan dari *signature*, dan keaslian file sertifikat digital.

3.4.2 Rancangan Tampilan

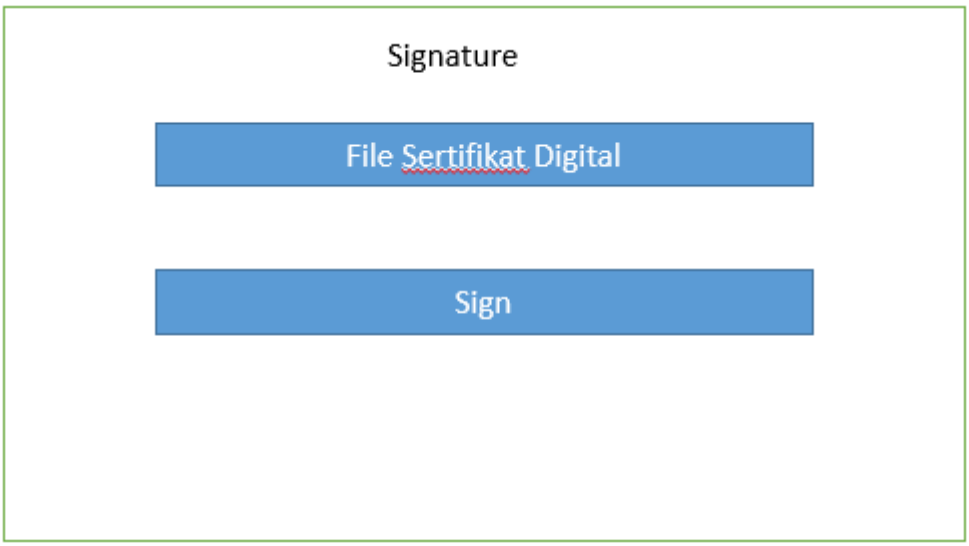
Program yang akan dibuat memiliki tampilan seperti berikut:



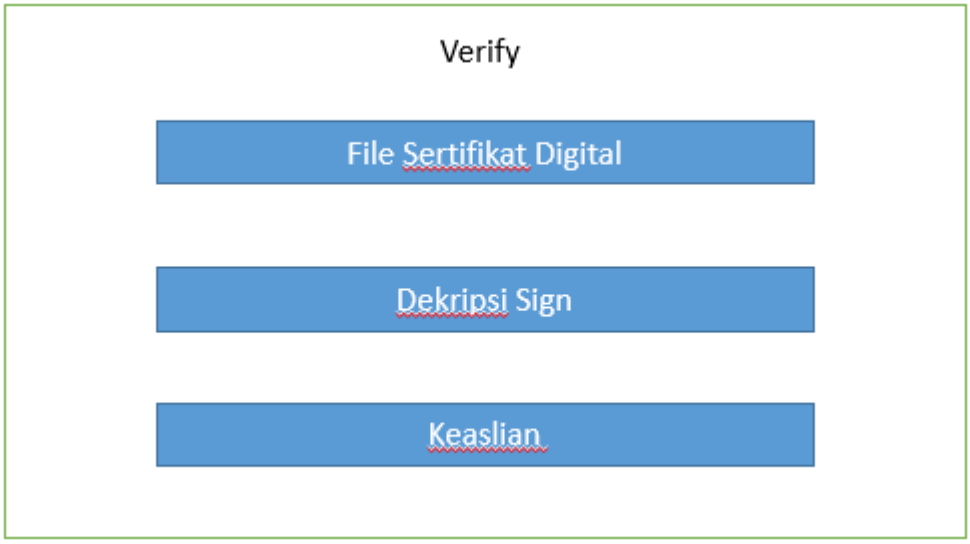
Gambar 3. 5 Menu Utama



Gambar 3. 6 Menu Pembangkitan Kunci



Gambar 3. 7 Menu *Signature*



Gambar 3. 8 Menu *Verify*

3.4.3 Algoritma *Digital signature*

Pada subbab 3.3 sudah dibahas mengenai pengembangan model dari *digital signature* ini akan menggunakan SHA-256 dengan skema penandatanganan Diffie-Hellman-RSA. Proses awal dari *digital signature* ini adalah melakukan *hashing* pada sertifikat digital dengan menggunakan SHA-256, lalu dilanjutkan dengan penandatanganan dengan menggunakan algoritma Diffie-Hellman-RSA.

3.4.3.1 *Hashing*

Proses hash dilakukan dengan cara memasukan sertifikat digital ke dalam program aplikasi. Proses *hash* dilakukan dengan menggunakan Python, dengan

menggunakan modul *hashlib*, hasil dari proses hash ini berupa nilai hash yang nantinya akan dienkripsi sehingga menjadi *digital signature*. Langkah-langkah untuk melakukan hash akan dijelaskan sebagai berikut:

1) Siapkan file sertifikat digital yang memiliki ekstensi .pdf
2) *Import* file sertifikat digital ke dalam aplikasi Python
3) File sertifikat akan diproses oleh aplikasi menggunakan *library* hashlib
4) Setelah pemrosesan selesai maka akan di dapatkan keluaran berupa nilai hash.

3.4.3.2 Pembangkitan Kunci

Pada dasarnya membangkitkan kunci algoritma Diffie-Hellman-RSA ini dilakukan dengan cara membangkitkan kunci dengan menggunakan algoritma Diffie-Hellman dan dilanjutkan membangkitkan kunci RSA yang dimodifikasi menggunakan kunci yang sudah dibangkitkan menggunakan algoritma Diffie-Hellman sehingga keamanan dari algoritma ini dapat ditingkatkan. Untuk lebih spesifik berikut adalah cara untuk membangkitkan kunci Diffie-Hellman-RSA:

1) Sepakati 2 buah bilangan prima (g, n) yang bersifat rahasia
2) Pilih suatu bilangan x untuk *signer* dan y untuk *verifier*, di mana $x < n$ dan $y < n$, x dan y bersifat rahasia.
3) Hitung nilai X oleh *signer* dan Y oleh *verifier*, dengan $X = g^x \ mod \ n$, dan $Y = g^y \ mod \ n$
4) Hitung nilai $K_1 = Y^x \ mod \ n$ oleh *signer* dan $K_2 = X^y \ mod \ n$ oleh *verifier* sehingga diperoleh $K_1 = K_2 = K$
5) Periksa nilai K apakah prima, jika tidak, maka dipilih K yang baru dimana K yang baru merupakan bilangan prima selanjutnya dari K yang tidak prima sebelumya.
6) Hitung $N = n \cdot g \cdot K$, dan $\Phi(\mathrm{N}) = (n-1)\ (g-1)$
7) Pilih suatu kunci publik e, di mana $FPB(e, \Phi(N)) = 1$
8) Pilih suatu kunci privat d di mana $e \cdot d \ \equiv 1 \ mod \ \Phi(N)$
9) Hitung $N1 = \frac{N}{K}$
10) Didapatkan lah kunci publik berupa pasangan bilangan (e, N) dan kunci privat berupa pasangan bilangan $(d, N1)$

3.4.3.3 Enkripsi dan Dekripsi

Proses enkripsi dan dekripsi Diffie-Hellman-RSA ini sama seperti proses enkripsi dan dekripsi RSA di mana proses enkripsi dimulai dengan mengubah plainteks ke dalam bentuk blok-blok terlebih dahulu dan dikomputasi tiap-tiap bloknya sehingga menghasilkan cipherteks, begitupun dekripsinya cipherteks dibagi menjadi tiap-tiap blok terlebih dahulu dan dikomputasi setiap bloknya hingga menghasilkan plainteks. Namun ada perbedaan pada komputasi Diffie-Hellman-RSA berikut rumus perhitungan enkripsi dan dekripsi:

- Enkripsi

  $c_i = (m_i)^e mod\ N$

- Dekripsi

  $m_i = (c_i)^d mod\ N1$

3.5 Keamanan *Digital Signature* dengan Algoritma SHA-256 dan Skema Penandatanganan Diffie-Hellman-RSA

Kekuatan *digital signature* dengan algoritma SHA-256 dengan skema Diffie Hellman-RSA terletak pada algoritma hashing yang digunakan yakni SHA-256 dan skema kriptografi Diffie Hellman-RSA. Untuk keamanan dari SHA-256 ada 3 sifat yang memastikan bahwa SHA-256 merupakan fungsi hash yang sangat aman, yaitu:

1. Tidak mungkin merekonstruksi data awal dari nilai hash yang diperoleh. Untuk merekonstruksi data awal yang berupa pesan dari nilai hash menggunakan *brute force* perlu melakukan 2^{256} percobaan.
2. Tidak mungkin dua pesan berbeda memiliki nilai hash yang sama (*collision*).
3. Perubahan kecil pada data asli mengubah nilai hash yang dihasilkan secara signifikan sehingga tidak terlihat nilai hash baru berasal dari data serupa (*avalanche effect*).

Adapun keamanan dari skema Diffie-Hellman-RSA terletak pada kunci publik dari Diffie-Hellman-RSA. Kunci publik Diffie-Hellman-RSA memiliki tingkat kesulitan yang tinggi untuk difaktorkan menjadi faktor-faktor primanya bahkan lebih sulit dari pemfaktoran kunci publik RSA. Untuk memfaktorkan kunci publik RSA, dapat dicari dengan memfaktorkan n menjadi dua buah bilangan prima p dan q. Jika nilai p dan q berhasil didapatkan maka nilai $\Phi(n) = (p-1)(q-1)$ dapat ditemukan dan kunci privat d di mana $e.d = 1\ mod\ \Phi(n)$ dapat diketahui sehingga

pesan asli pun dapat diketahui. Berbeda dengan halnya RSA kunci publik Diffie-Hellman-RSA ($N = n.g.K$) pemfaktorannya terdiri dari tiga buah bilangan prima sehingga menambah kerumitan dalam memecahkan kunci Diffie-Hellman-RSA. Sama seperti RSA untuk memecahkan kunci Diffie-Hellman-RSA perlu mengetahui nilai n dan g untuk mencari nilai $\Phi(n) = (n-1)(g-1)$, agar dapat memperoleh nilai d sehingga bisa melakukan dekripsi dan mengetahui pesan asli, namun untuk memperoleh nilai p dan q diperlukan nilai $K1$ yang didapatkan dari pertukaran kunci Diffie-Hellman.

3.6 Validasi

Pada tahap ini akan dilakukan percobaan untuk membuktikan kebenaran dari *digital signature* menggunakan algoritma SHA-256 dengan skema penandatanganan Diffie-Hellman-RSA, dengan cara memberi masukan suatu file sertifikat digital yang asli pada proses *signature*, dan pada proses verifikasi akan digunakan file sertifikat digital yang palsu sehingga dapat terbukti bahwa nilai hash dari kedua sertifikat berbeda dan file sertifikat yang diuji merupakan sertifikat palsu.