Penerapan Algoritma RSA dalam Enkripsi Pengiriman Pesan dan Proses Identifikasi Entitas Khusus Pengirim Pesan pada Messaging App

Michael Hans – 13518056 (*Author*)
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): michaelhans777@gmail.com

***Abstract*—Keamanan menjadi salah satu faktor penting di dalam aplikasi pengirim pesan agar pesan dapat terkirim secara aman kepada penerima. Dalam konteks *messaging app*, terdapat dua hal penting yang perlu diperhatikan, yaitu keamanan saat melakukan pengiriman pesan dan kepastian bahwa pesan memang dikirim oleh pengirim pesan yang asli. Keduanya dapat difasilitasi dengan menggunakan algoritma kriptografi kunci publik dengan penerapan yang berbeda. Salah satu algoritma kriptografi kunci publik yang dapat digunakan untuk proses pengiriman pesan dan identifikasi keaslian pengirim adalah algoritma RSA. Keamanan pengiriman pesan diimplementasikan dalam bentuk *end-to-end encryption* dengan algoritma RSA, sedangkan proses identifikasi entitas khusus yang mengirimkan pesan dapat ditangani dengan menggunakan *digital signature*.**

***Keywords*—pesan, *messaging app*, kunci, enkripsi, dekripsi, blok, RSA, signature, verifikasi**

I. PENDAHULUAN

Di era teknologi informasi saat ini, keamanan menjadi salah satu unsur penting dalam menjaga integritas dari teknologi informasi. Informasi-informasi terus mengalir setiap harinya dari waktu ke waktu mulai dari informasi tentang transaksi, pemesanan, hingga informasi pribadi setiap pengguna yang terus berpindah dari tangan seorang pengguna ke pengguna lainnya. Terlebih lagi pada masa pandemi ini, aktivitas manusia semakin terbatas sehingga berbagai pihak dan perusahaan mulai beralih dari penyampaian informasi yang bersifat tradisional berubah menjadi digital.

Penyampaian informasi secara daring mulai dilakukan oleh pihak-pihak berkepentingan seperti pemerintah, bank, rumah sakit, dan pihak berwenang lainnya untuk senantiasa menyampaikan informasi-informasi penting kepada para penerima. Dampak lain terhadap orang awam adalah orang awam perlu terbiasa dalam menggunakan platform-platform penerima pesan seperti email dan *messaging app* agar dapat menerima informasi-informasi yang dikirimkan secara daring.

Oleh karena itu, keamanan menjadi faktor yang penting agar informasi yang mengalir tersebut tetap aman hingga sampai tujuan, khususnya pada aplikasi pengirim pesan *(messaging app*). Aplikasi pengirim pesan menjadi salah satu aplikasi paling penting dalam era teknologi informasi saat ini yang dapat memudahkan orang-orang untuk tetap terhubung satu sama lain tanpa harus bertemu tatap muka secara langsung. Namun, sebagaimana halnya ketika orang hanya ingin obrolannya diketahui oleh lawan bicaranya, hal tersebut juga berlaku dalam aktivitas pada aplikasi pengiriman pesan. Pemakai aplikasi *messaging app* perlu mendapatkan jaminan agar pengiriman pesan kepada pihak penerima bisa tetap terjaga dengan baik tanpa diretas pihak ketiga. Oleh karena itu, diperlukan sebuah mekanisme enkripsi pesan yang dapat menjaga integritas pesan tersebut agar tetap utuh dan tidak diretas oleh pihak ketiga hingga pesan tersebut sampai pada penerima. Salah satu algoritma yang dapat memfasilitasi kebutuhan tersebut adalah algoritma RSA.

Selain dari sisi keamanan pesan, ada kalanya pihak penerima juga perlu mendapatkan kepastian terkait pengirim yang mengirimkan pesan tersebut. Belakangan ini sudah marak terjadinya pengiriman informasi palsu atau penipuan yang mengatasnamakan entitas-entitas tertentu seperti pemerintah, bank, rumah sakit, dan pihak-pihak berwenang lainnya. Informasi tersebut disusun sedemikian rupa sehingga terlihat seperti pesan yang resmi dengan tata cara penulisan pesan yang biasa dikirimkan oleh pihak tersebut. Akibatnya adalah banyak orang awam yang terkena penipuan akibat informasi tersebut.

Oleh karena itu, dengan menggunakan algoritma yang sama (algoritma RSA), penulis mencoba untuk mengembangkan dua buah mekanisme pengiriman pesan yang dapat dipraktekkan, yaitu melakukan enkripsi pemesanan dengan algoritma kriptografi kunci publik RSA dan sekaligus melakukan suatu proses identifikasi entitas tertentu dengan algoritma yang sama namun memiliki penerapan yang berbeda di dalamnya, yaitu dalam bentuk tanda tangan digital.

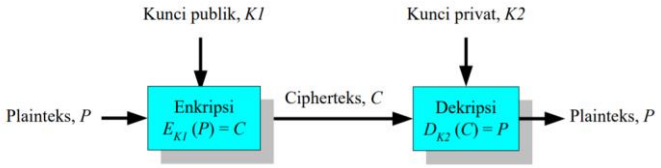
II. LANDASAN TEORI

A. Algoritma Kriptografi Kunci-Publik

Algoritma kriptografi kunci publik adalah salah satu algoritma kriptografi yang memiliki prinsip berupa kunci yang digunakan untuk melakukan enkripsi suatu pesan berbeda

dengan kunci yang digunakan untuk melakukan dekripsi suatu pesan. Ide dari kriptografi kunci publik ini berawal dari adanya masalah terkait pengiriman kunci rahasia kepada penerima pesan pada sistem kriptografi kunci-simetri, yaitu diperlukannya suatu saluran yang sangat aman dalam mengirimkan kunci tersebut. Mengingat biaya dan waktu dalam pengirimannya umumnya lambat dan mahal, muncullah ide kriptografi kunci-publik sehingga tidak perlu adanya penggunaan kunci yang sama antara pengirim dan penerima.

Makalah pertama perihal kriptografi kunci-publik ditulis oleh Whitfield Diffie (kiri) dan Martin E. Hellman (kanan) di IEEE pada tahun 1976. Keduanya adalah ilmuwan dari Stanford University dan merupakan penemu kriptografi kunci-publik. Kriptografi kunci publik disebut juga kriptografi kunci-nirsimetri karena kunci enkripsi tidak sama dengan kunci dekripsi. Istilah "publik" sendiri muncul karena kunci untuk melakukan enkripsi bersifat tidak rahasia atau publik seperti menyimpan kunci pada repositori yang dapat diakses semua orang. Sedangkan terdapat kunci privat yang rahasia yang hanya dimiliki oleh pemilik kunci privat untuk mendekripsi pesan yang diterimanya. Gambaran umum terkait kriptografi kunci publik dapat dilihat pada Gambar X berikut ini.



Gambar 1. Proses kriptografi kunci publik [1]

Terdapat dua keuntungan dari penggunaan kriptografi kunci-publik, yaitu:

1. Tidak diperlukan pengiriman kunci privat, yaitu setiap orang memiliki kunci privat masing-masing.
2. Jumlah kunci dapat ditekan, yaitu setiap orang hanya perlu memiliki sepasang kunci saja (privat dan publik), kunci publik orang lain dapat diketahui dari repositori yang bersifat publik.

Kriptografi kunci-publik didasarkan pada dua buah fakta berikut ini.

1. Komputasi untuk enkripsi / dekripsi pesan mudah dilakukan.
2. Secara komputasi hamper tidak mungkin (*infeasible*) menurunkan kunci privat bila diketahui kunci publik.

B. Algoritma RSA

RSA merupakan algoritma kunci-publik yang paling terkenal dan paling banyak aplikasinya dalam enkripsi data sehari-hari. Kata RSA ini diambil dari nama dari tiga peneliti dari MIT yang mengembangkannya, yaitu Ronald Rivest, Adi Shamir, dan Len Adleman pada tahun 1976. Keamanan algoritma RSA terletak pada sulitnya memfaktorkan bilangan bulat yang besar menjadi faktor-faktor prima.

Berikut ini adalah beberapa properti penting di dalam algoritma RSA.

1. p dan q bilangan prima (rahasia)
2. n = p . q (tidak rahasia)
3. $\Phi(n) = (p - 1)(q - 1)$ (rahasia)
4. e (kunci enkripsi) (tidak rahasia)
   Syarat: PBB(e, $\Phi(n)$) = 1, PBB = pembagi bersama terbesar = gcd (*greatest common divisor*)
5. d (kunci dekripsi) (rahasia)
   d dihitung dari $d \equiv e^{-1} \pmod{\Phi(n)}$
6. m (plainteks) (rahasia)
7. c (cipherteks) (tidak rahasia)

Dari kelima properti, dapat dirumuskan sebuah kunci publik dan kunci privat sebagai berikut.

- Kunci publik = (e, n)
- Kunci privat = (d, n)

Berikut ini adalah langkah-langkah untuk melakukan enkripsi pesan dengan menggunakan algoritma RSA.

1. Nyatakan pesan menjadi blok-blok plainteks: m_1, m_2, m_3, m_4, … (syarat: $0 \le m_i < n - 1$).
2. Untuk setiap blok plainteks, hitunglah blok cipherteks c_i untuk blok plainteks m_i menggunakan kunci publik e dengan persamaan berikut ini.

$$c_i = m_i^e \bmod n \quad (1)$$

3. Gabungkan blok-blok cipherteks tersebut secara terurut sehingga menjadi satu cipherteks utuh.

Berikut ini adalah langkah-langkah untuk melakukan dekripsi cipherteks dengan menggunakan algoritma RSA.

1. Nyatakan cipherteks ke dalam blok-blok cipherteks: c_1, c_2, c_3, c_4, … (syarat: $0 \le c_i < n - 1$).
2. Untuk setiap blok cipherteks, hitunglah blok plainteks m_i untuk blok cipherteks c_i menggunakan kunci privat d dengan persamaan berikut ini.

$$m_i = c_i^d \bmod n \quad (2)$$

3. Gabungkan setiap blok plainteks tersebut secara terurut sehingga menjadi satu plainteks utuh.

C. Messaging App

Messaging App adalah sebuah aplikasi yang dipasang pada sebuah perangkat mobile yang berfungsi sebagai media komunikasi dua arah dalam mengirimkan pesan dari pengirim pesan kepada penerima pesan. Perbedaan yang mendasar dengan media komunikasi lainnya adalah pesan dikirimkan secara instan (*instant messaging*) dan memfasilitasi komunikasi melalui suara maupun video. Beberapa aplikasi *messaging app* yang terkenal dan banyak dipakai oleh masyarakat umum adalah WhatsApp, Messenger, Telegram, dan LINE.

Pada mulanya, *messaging app* didesain sedemikian rupa untuk komunikasi secara privat antara satu pihak kepada pihak yang lain. Namun seiring berkembangnya zaman, maka *messaging app* juga turut berkembang dengan pemanfaatan beberapa teknologi baru seperti pengiriman *broadcast* kepada orang-orang dalam jumlah besar, mekanisme *end-to-end encryption*, dan adanya teknologi chatbot yang memanfaatkan pemrosesan bahasa alami di dalamnya.

End-to-end encryption saat ini sudah menjadi protokol keamanan yang berlaku ketika perusahaan mengembangkan sebuah *messaging app*. Dengan adanya *end-to-end encryption* tersebut, konten dari pesan yang dikirimkan dijamin hanya dapat dilihat dan dibuka oleh penerima pesan, termasuk pihak penyedia layanan aplikasi tidak dapat melakukan dekripsi atau membaca pesan yang dikirimkan kepada penerima tersebut.

Saat ini, *messaging app* tidak hanya dimanfaatkan oleh para individu atau komunitas tertentu, melainkan juga dimanfaatkan oleh pihak pemerintah atau pihak perusahaan untuk melakukan pengiriman informasi kepada orang-orang yang berhak mendapatkan informasi tersebut. Pihak-pihak khusus seperti bank, rumah sakit, dan pemerintah secara perlahan-lahan beralih dari teknologi SMS menuju *messaging app* dalam mengirimkan pesan-pesan penting. Oleh sebab itu, pihak penyedia layanan *messaging app* juga turut memfasilitasi kebutuhan tersebut dengan memberikan perlakuan khusus kepada pihak-pihak tersebut seperti tanda *verified* pada nomor-nomor atau user ID tertentu sehingga para pengguna lain bisa lebih cepat mengetahui pengirim yang mengirimkan pesan penting tersebut.

D. End-to-End Encryption

End-to-end encryption adalah suatu istilah yang diberikan ke dalam mekanisme pengiriman pesan dengan melakukan enkripsi pada pesan yang akan dikirimkan sehingga pesan tersebut dapat terkirim dari ujung ke ujung secara aman. Definisi “aman” yang dimaksud adalah pesan dapat terkirim kepada penerima pesan tanpa adanya pihak lain yang meretas pesan tersebut dan hanya penerima pesan itu saja yang dapat membukanya. Pesan yang terenkripsi tersebut akan didekripsi oleh pihak penerima dengan menggunakan sebuah kunci privat yang hanya dimiliki oleh penerima tersebut di dalam perangkat yang dimilikinya.

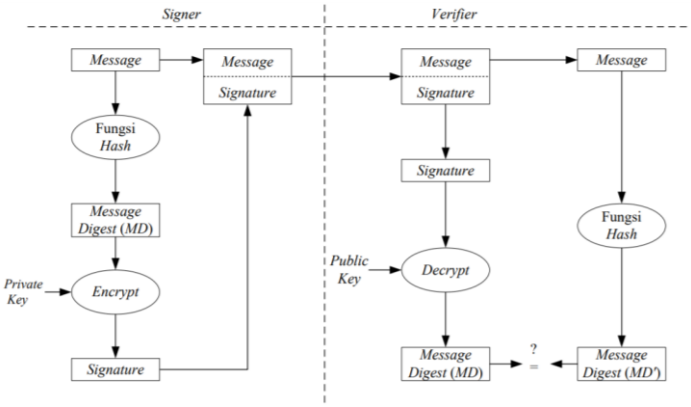
Mekanisme *end-to-end encryption* ini dapat memastikan pesan yang dikirimkan kepada penerima pesan tidak dapat dibuka oleh siapapun selain penerima itu sendiri. Pada aplikasi pengirim pesan, end-to-end encryption ini membuat pesan tersebut juga tidak dapat dibaca oleh pihak server sendiri karena pesan yang disimpan pada server adalah pesan yang terenkripsi dan hanya dapat dibuka oleh penerima pesan yang memiliki kunci dekripsinya.

E. Tanda Tangan Digital

Tanda Tangan Digital (*Digital Signature*) adalah salah satu bentuk tanda tangan yang melibatkan perhitungan algoritma dan kriptografi tertentu yang berfungsi untuk menandatangani dan memvalidasi keaslian berkas-berkas digital seperti dokumen, file, atau perangkat lunak. Tanda tangan digital digunakan untuk memastikan bahwa berkas atau pesan yang dikirim secara digital milik sumber pengirim dan dapat mencapai penerima yang dituju dalam format aslinya tanpa gangguan atau perubahan apapun.

Tanda tangan digital memiliki nilai kriptografis yang bergantung dari isi pesan dan kunci. Perbedaan yang mendasar dengan tanda tangan pada dokumen cetak adalah tanda tangan digital selalu berbeda-beda antara satu isi dokumen dengan dokumen lain. Perbedaan mendasar dengan mekanisme enkripsi pesan dengan kriptografi kunci publik adalah posisi kunci publik dan kunci privat yang berbeda serta perbedaan dari plainteks yang dienkripsi. Pada tanda tangan digital, kunci privat digunakan untuk melakukan enkripsi yang dibutuhkan pada proses penandatanganan. Sedangkan kunci publik digunakan untuk melakukan dekripsi yang dibutuhkan pada proses verifikasi tanda tangan digital yang ada. Selain itu, plainteks yang dienkripsi adalah sebuah *message digest* yang diperoleh dari hash terhadap pesan orisinil. Proses verifikasi dilakukan dengan melihat kesamaan dua buah message digest, yaitu message digest hasil hash terhadap pesan yang diterima dan message digest hasil dekripsi terhadap signature.

Berikut ini adalah mekanisme umum dalam melakukan tanda tangan digital terhadap sebuah pesan tertentu.



Gambar 2. Mekanisme tanda tangan digital [1]

Dalam prakteknya, terdapat dua buah algoritma tanda tangan digital yang banyak digunakan, yaitu algoritma RSA dan algoritma ElGamal. Pada RSA, algoritma enkripsi dan dekripsi identik, sehingga proses signature dan verifikasi juga identik. Selain RSA, terdapat algoritma yang dikhususkan untuk tanda-tangan digital, yaitu Digital Signature Algorithm (DSA), yang merupakan bakuan (standard) untuk Digital Signature Standard (DSS). Pada DSA, algoritma signature dan verifikasi berbeda.

III. Rencana Penyelesaian Masalah

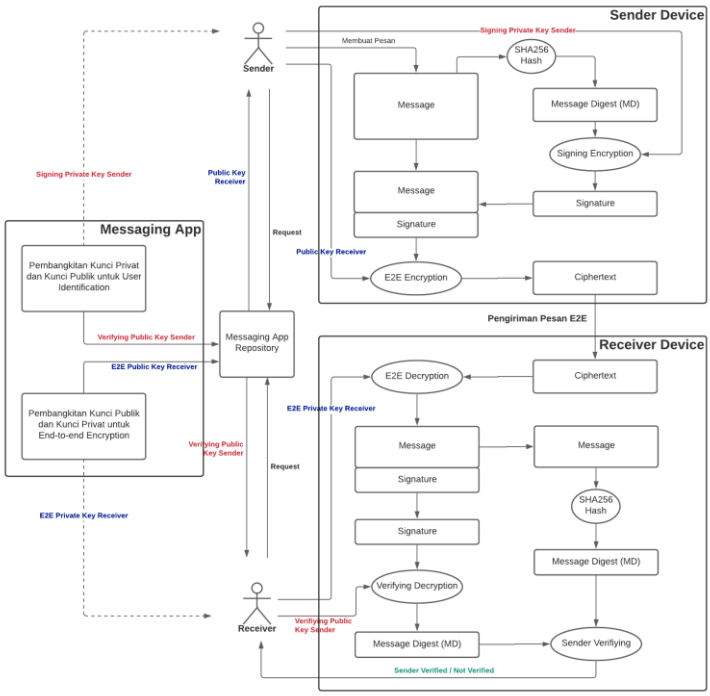
Seperti yang telah dijelaskan sebelumnya, algoritma RSA merupakan salah satu algoritma kriptografi kunci publik yang dapat dimanfaatkan dalam pengamanan pesan agar pesan dapat sampai pada pihak penerima dengan aman tanpa diretas oleh

pihak ketiga. Dalam kasus aplikasi pengirim pesan (*messaging* app), algoritma RSA dapat diterapkan pada proses *end-to-end encryption* ketika melakukan pengiriman pesan.

Pada makalah ini, penulis mencoba untuk mengembangkan penerapan algoritma RSA yang ada bukan hanya untuk proses *end-to-end encryption* saja, melainkan juga untuk proses pengidentifikasian entitas khusus agar penerima pesan bisa mengetahui pengirim pesan yang sesungguhnya dan mengetahui kredibilitas dari pengirim pesan tersebut. Berikut ini adalah rancangan solusi dan implementasi dari penerapan algoritma RSA pada *messaging app* tersebut.

A. Rancangan Solusi dan Arsitektur

Berikut ini adalah rancangan solusi yang diterapkan untuk menguji hipotesis sebelumnya.



Gambar 3. Rancangan arsitektur pengamanan pesan dan proses identifikasi pengirim (sumber: Dokumen penulis)

Rancangan arsitektur pengamanan pesan dan proses identifikasi digambarkan pada Gambar 3. Terdapat dua buah entitas yang berperan dalam arsitektur ini, yaitu sender dan receiver. Sebelum kedua pihak melakukan pengiriman dan penerimaan pesan, akan dibangkitkan dua buah pasang kunci RSA, yaitu sepasang kunci RSA yang dimiliki oleh receiver untuk proses enkripsi pesan dan sepasang kunci RSA yang dimiliki oleh sender untuk proses pengidentifikasian keaslian pengirim dalam bentuk digital signature. Setiap kunci publik yang dibangkitkan akan disimpan ke dalam *messaging app repository* yang dapat diakses secara publik dalam proses enkripsi pesan dan verifikasi pesan. Oleh karena itu, secara harafiah setiap *user* akan memiliki dua pasang kunci, yaitu sepasang kunci untuk enkripsi dan sepasang kunci untuk *digital signature*.

Sender akan melakukan *request* terhadap server untuk mendapatkan kunci publik receiver sebagai kunci untuk melakukan *end-to-end encryption*. Sender membuat pesan yang akan dikirimkan kepada receiver. Sebagai penanda atau bukti bahwa pesan tersebut memang dikirimkan oleh sender yang asli, terdapat penambahan pada pesan yang dikirimkan berupa *digital signature* yang disisipkan pada bagian belakang pesan awal. *Digital signature* tersebut diperoleh dari message digest hasil dari hash terhadap pesan awal. Plainteks yang terdiri dari message dan signature tersebut akan dienkripsi menggunakan kunci publik receiver yang sudah diterima sebelumnya sehingga dihasilkan sebuah cipherteks yang siap dikirimkan kepada receiver.

Setelah cipherteks sampai pada perangkat dari receiver, cipherteks tersebut akan didekripsi dengan algoritma RSA dan menggunakan kunci privat yang dimiliki receiver pada saat pembangkitan pasangan kunci enkripsi RSA awal. Hasil akhir dari dekripsi ini adalah sebuah plainteks yang terdiri dari pesan utama dan signature. Selanjutnya receiver akan melakukan request terhadap server untuk mendapatkan kunci publik verifikasi milik sender untuk melakukan verifikasi terhadap signature yang diperoleh. Proses verifikasi dilakukan dengan melakukan dekripsi terhadap signature tersebut sehingga dihasilkan sebuah message digest. Message digest hasil dekripsi signature akan dicocokkan dengan message digest hasil hash terhadap pesan utama. Apabila keduanya adalah message digest yang sama, akan dikirimkan sebuah penanda berupa *verified message* kepada receiver sebagai bukti bahwa pesan tersebut dikirim oleh *sender* yang asli. Perlu diperhatikan bahwa signature hanya akan cocok apabila plainteks didekripsi dengan benar pada saat proses dekripsi *end-to-end encryption*.

B. Batasan dan Asumsi

Berdasarkan rancangan solusi dan arsitektur yang sudah diusulkan, penulis membuat sebuah aplikasi berbasis web sederhana untuk mendemonstrasikan cara kerja dari enkripsi pengamanan pesan dan proses identifikasi entitas khusus dengan menggunakan algoritma RSA. Berikut ini adalah batasan dan asumsi yang digunakan dalam proses implementasi pembangunan aplikasi ini.

1. Aplikasi messaging app dibangun dengan menggunakan ReactJS sebagai aplikasi antar muka (*Front End*) dan Python Flask sebagai *Back End* atau API Service untuk melakukan proses enkripsi, dekripsi, pembangkitan kunci RSA, dan pengambilan data dari *message repository*.
2. Fungsi hash yang digunakan untuk menghasilkan Message Digest adalah fungsi SHA-256 bawaan dari library Python sehingga penulis tidak mengimplementasikan fungsi hash dari *scratch*.
3. Penyimpanan data *messages* dan profil user disimpan di dalam basis data NoSQL pada Firestore Cloud.
4. Penyimpanan kunci privat pada perangkat diimplementasikan dengan menyimpan kunci privat tersebut di dalam folder khusus kunci privat secara lokal.

5. Fokus utama aplikasi adalah untuk menunjukkan hasil enkripsi pesan yang dikirimkan ke pengirim, menunjukkan hasil dekripsi cipherteks yang masuk ke penerima pesan (*receiver*) dan keaslian pesan dilihat dari terverifikasinya pesan tersebut, yaitu pesan berasal dari pengirim yang asli.

C. Implementasi Algoritma RSA

Sesuai dengan algoritma RSA pada landasan teori sebelumnya, algoritma RSA menerima input berupa blok-blok m_i yang berisi sebuah nilai $0 \leq m_i < n - 1$. Pada prakteknya pada *messaging app*, pesan yang dimasukkan oleh user adalah sebuah string yang terdiri atas kumpulan karakter seperti karakter alfabet (a-z), karakter numerik (0-9), dan karakter simbol-simbol lainnya yang terdapat pada keyboard. Oleh karena itu, perlu dilakukan *preprocessing* terhadap pesan sebelum pesan tersebut bisa dienkripsi oleh algoritma RSA.

Preprocessing dilakukan dengan melakukan konversi setiap karakter pada pesan / input user menjadi nilai ASCII terkait dalam representasi heksadesimal. Representasi heksadesimal dipilih agar proses konversi dari string menuju representasi heksadesimal bersifat *reversible*. Setiap karakter pada string selalu direpresentasikan sebagai satu byte pada rentang nilai 0x00 sampai 0xff sehingga proses konversi heksadesimal ke string dapat dilakukan dengan mengkonversi setiap dua karakter heksadesimal. Setiap nilai ASCII yang diperoleh akan dikonkatenasi secara terurut sehingga diperoleh *sequence* ASCII dari setiap karakter input user. *Sequence* ASCII dalam representasi heksadesimal tersebut akan dikonversi lagi ke dalam bentuk integer agar s*equence* ASCII dapat diproses pada algoritma RSA yang menerima input berupa bilangan integer. Hasil akhir preprocessing tetap bertipe data string agar dapat memudahkan proses pembagian ke dalam blok-blok yang dibutuhkan untuk proses enkripsi dan dekripsi algoritma RSA.

Pada implementasi RSA yang dilakukan oleh penulis, ukuran blok bergantung pada besar nilai n dari kunci publik dan kunci privat yang digunakan. Ukuran blok yang dibentuk saat proses enkripsi dan dekripsi juga berbeda agar proses enkripsi dan dekripsi dapat berjalan dengan lancar. Berikut ini adalah modifikasi yang diterapkan pada algoritma RSA agar dapat memfasilitasi kebutuhan enkripsi pesan dalam jumlah besar.

1. Untuk menjamin bahwa setiap blok $0 \leq m_i < n - 1$, panjang blok ditetapkan menjadi banyaknya digit dari bilangan n dikurangi dengan satu sehingga setiap blok mi akan memenuhi $0 \leq m_i < 10^{len(n)-1}$ dengan len(n) adalah fungsi untuk menghitung banyaknya digit dari bilangan n. Karena $10^{len(n)-1} < n-1$, maka $0 \leq m_i < n - 1$ juga terpenuhi.
2. Jika panjang plainteks tidak habis dibagi len(n), maka ditambahkan beberapa karakter 0 sebagai padding agar plainteks dapat habis dibagi len(n) sehingga setiap blok plainteks berukuran sama.
3. Setiap blok yang sudah dienkripsi akan dipadatkan ke dalam blok berukuran len(n) digit sehingga setiap cipherteks akan dipadatkan pada format berukuran banyaknya digit pada bilangan n. Tujuan pemadatan tersebut agar blok-blok cipherteks yang diperoleh dari pemecahan blok-blok ketika proses dekripsi sama dengan blok-blok cipherteks dari hasil enkripsi.

Ilustrasi secara lebih lengkap dijelaskan dengan menggunakan kasus berikut ini. Misalkan terdapat *plain message* berisi 'Nasabah Yth' dengan public key berupa (2908781, 96133) dan private key berupa (53621, 96133). Dari kedua key tersebut, diperoleh bilangan n sebesar 96133 dengan banyak digit bilangan n adalah 5. Berikut ini adalah *state-state* perubahan yang terjadi dari *plain message* menuju *encrypted message*.

TABEL I. TABEL STATE ALGORITMA RSA

State	Nilai
Plain message	Nasabah Yth
Sequence ASCII	4e61736162616820597468
Plainteks (Integer)	947564118716261770551266 32
Blok-Blok Plainteks	**00**94 7564 1187 1626 1770 5512 6632
Blok-Blok Cipherteks	94499 37314 25935 00281 38887 40929 28059
Cipherteks (Integer)	9449937314259350028138887 4092928059
Encrypted message	12332dc6d1f7b62b63c1c061fe5c3b

Proses dekripsi dari pesan dilakukan dengan menggunakan kunci dekripsi yang ada dan perubahan state terjadi dari *encrypted message* menuju *plain message*.

D. Implementasi Aplikasi Messaging App

Aplikasi *Messaging App* dibangun berbasis web dengan tiga buah komponen utama, yaitu Front End, Back End, dan Repository. Front End adalah komponen yang berfungsi untuk menampilkan pesan dan media interaksi antara user dengan aplikasi *messaging app* secara *friendly.* Back End adalah komponen yang berfungsi untuk mengelola data-data dan fungsi-fungsi pengamanan pesan yang diperlukan untuk keperluan *messaging app*. Sedangkan Repository digunakan sebagai tempat penyimpanan pesan-pesan dan data terkait user yang dapat menggunakan aplikasi *messaging app.* Detail terkait API yang digunakan untuk proses pengamanan pesan dan pengambilan data dari repositori dapat dilihat pada tabel di bawah ini.

TABEL II. TABEL API SERVICE

API	Method	Fungsionalitas
/generate-key	POST	Membangkitkan pasangan kunci publik dan kunci privat untuk keperluan end-to-end encryption. Kunci publik disimpan pada Firestore, sedangkan kunci privat disimpan pada folder private_key
/generate-signature-	POST	Membangkitkan pasangan kunci publik dan kunci privat untuk keperluan

API	Method	Fungsionalitas
key		signing dan verifying. Kunci publik disimpan pada Firestore, sedangkan kunci privat disimpan pada folder private_key
/add	POST	Melakukan enkripsi terhadap message yang akan dikirimkan dan menambahkan message baru ke dalam repository *messaging app*
/messages	GET	Mengembalikan seluruh pesan yang diterima receiver dan mendekripsi pesan tersebut.
/raw-messages	GET	Mengembalikan seluruh pesan yang diterima receiver tanpa melakukan dekripsi pada pesan tersebut.

Messaging App Repository diimplementasikan dalam basis data NoSQL Firestore Cloud dengan tiga buah tabel sebagai berikut.

TABEL III. TABEL PADA REPOSITORY

Tabel	Deskripsi	Key-Value list
messages	Daftar pesan yang sudah terenkripsi yang disimpan yang dikirim dari sender ke receiver	sender:string, receiver: string, message: string, timestamp: date
users	Daftar user yang terdaftar pada *messaging app* beserta public key yang dapat digunakan ketika mengirim pesan ke user bersangkutan	username:string public_key: string
verified_users	Daftar entitas khusus beserta public key untuk melakukan verifikasi keaslian pesan	username:string public_key: string

IV. HASIL DAN PEMBAHASAN

A. *Inisialisasi Pengujian*

Pada tahap pengujian, ditetapkan dua user yang akan berperan sebagai *sender* dan *receiver*, yaitu Shadow Bank sebagai *sender* dan Michael Hans sebagai *receiver*. Pengujian diambil dari sebuah kasus berupa pengiriman sebuah informasi penting yang disampaikan oleh pihak Shadow Bank kepada nasabah yang bernama Michael Hans. Informasi penting tersebut bisa terkait dengan informasi keuangan nasabah, transaksi nasabah, atau pengumuman yang diberikan oleh Shadow Bank.

Sebelum keduanya dapat melakukan pengiriman pesan, *messaging app* akan membangkitkan dua pasang kunci untuk keperluan *end-to-end encryption* dan proses identifikasi entitas khusus. Pembangkitan kunci dilakukan dengan memanggil kedua API endpoint dengan konfigurasi sebagai berikut.

TABEL IV. TABEL PEMBANGKITAN KUNCI

API Endpoint	Body (JSON)	Response (JSON)
/generate-key	{ user: “MichaelHans” }	{ "private_key": "13744231093667,87093738739141", "public_key": "4851211,87093738739141" }
/generate-signature-key	{ user: “ShadowBank” }	{ "private_key": "3540443,1396201978657", "public_key": "548864962643,1396201978657" }

B. *Pengujian dan Analisis*

Setelah mendapatkan kedua pasang kunci, maka proses pengiriman pesan sudah bisa dilakukan. Terdapat lima buah kasus uji yang akan dilakukan terhadap solusi arsitektur yang sudah diimplementasikan sebelumnya.

1. Kasus 1: Pesan berhasil diterima dan terverifikasi

   Kasus ini merupakan kasus normal yang terjadi apabila pihak Shadow Bank secara resmi mengirimkan pesan kepada Michael Hans sebagai nasabah dari bank tersebut. Berikut ini adalah *input* dan *response* yang diperoleh saat pengiriman pesan dari Shadow Bank kepada Michael Hans.

API Endpoint: /add
Method: POST
Body:

```
{
   "sender": "ShadowBank",
   "receiver": "MichaelHans",
   "message": "Nasabah Yth. Saat ini transaksi Anda sedang kami proses. Pastikan no HP Anda dalam keadaan aktif untuk verifisikasi. Call Shadow Bank 13518056."
}
```

Public key receiver: 4851211,87093738739141
Signing private key sender: 3540443,1396201978657

Message + signature: “Nasabah Yth. Saat ini transaksi Anda sedang kami proses. Pastikan no HP Anda dalam keadaan aktif untuk verifisikasi. Call Shadow Bank 13518056.<ds>3ffa040e69569b024de5e382047222915028f3c5f869627d27e592606401ed77acf76f918f60f8041e24473fafffbcd627d86940a100c7fa271f3cc9f84cc5afaeab2401ac75</ds>”

Response:

```
{
      "message":
"13b3e5aab8e926fdfa7655530cebc7c018a671658053776b2ad494d90a347d6752d870da59c01ce48c1ab8a6c8372fed25176307939377bfc0598e39579288c170d872bfdfbda25c28123d05c01050b2dc4aeb1b985921d83e3e899e644e3ab594b1b431de3caaafdf3e34fcb0f9f2f07e5a8f4251041f23329cea8ac74c095e0211e843edc0ea6a558e3715fd844e90ce043b608ebdd214e0e300e1d8740e983b0641194e3092399f5d44f28a7c51630d634261dd2c5e587aa77641d32605d2a3dfc26b81e76beb503ab20d53dfa9e54102844ad91dd9fabeb97faa5e5dee5b738434f058453afb720194269727233e9fa5edd8035407dc44fcfc423ea84b23cdc09951822630a891ec924bd5a962c2781544baa5815d30aae27a50c994edb4865ab1e43b6b5e9cf0d55fe9bc5f5819291c5361651f1835d947f430f83804e",
      "receiver": "MichaelHans",
      "sender": "ShadowBank",
      "timestamp": "2021-12-17 00:46:38.431836"
}
```

Pada sisi Front End, user akan login sebagai “MichaelHans” untuk mengakses *messaging app* dan melihat pesan yang masuk dari ShadowBank. Untuk mengambil seluruh *messages*, aplikasi melakukan GET Request ke endpoint messages sebagai berikut.

API Endpoint: /messages
Method: GET
Params: receiver= MichaelHans

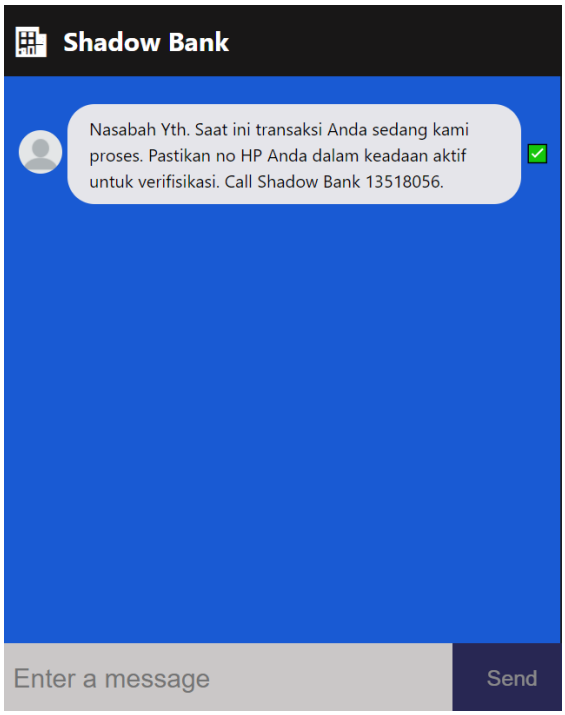
Private key receiver: 13744231093667,87093738739141
Verifying public key sender: 548864962643,1396201978657

Message + signature: “Nasabah Yth. Saat ini transaksi Anda sedang kami proses. Pastikan no HP Anda dalam keadaan aktif untuk verifisikasi. Call Shadow Bank 13518056.<ds>3ffa040e69569b024de5e382047222915028f3c5f869627d27e592606401ed77acf76f918f60f8041e24473fafffbcd627d86940a100c7fa271f3cc9f84cc5afaeab2401ac75</ds>”
Message digest:
91f6608f6352dd3efc1b044a510f46e8e94aaed270337484e1f35f9ee5d2bd3a
Message digest’’:
91f6608f6352dd3efc1b044a510f46e8e94aaed270337484e1f35f9ee5d2bd3a

Response:

```
[
   {
      "message": "Nasabah Yth. Saat ini transaksi Anda sedang kami proses. Pastikan no HP Anda dalam keadaan aktif untuk verifisikasi. Call Shadow Bank 13518056.",
      "receiver": "MichaelHans",
      "sender": "ShadowBank",
      "timestamp": "2021-12-17 00:46:38.431836",
      "verify": true
   }
]
```

Message yang diperoleh kemudian akan ditampilkan pada sisi Front End. Tampilan pesan dari sisi Front End dapat dilihat pada gambar berikut ini.



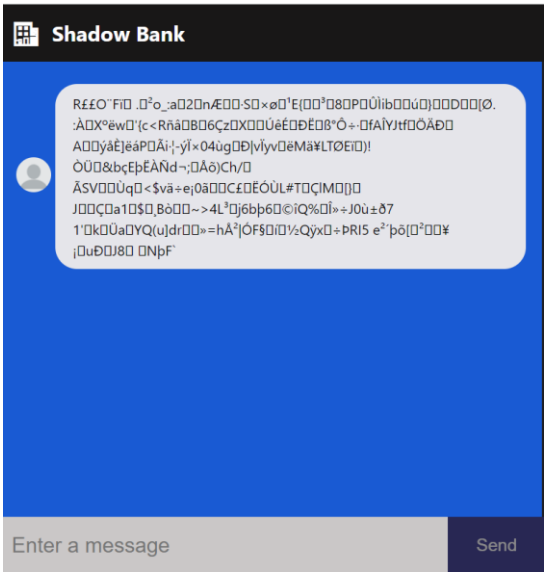
Gambar 4. Pesan berhasil didekripsi dan terverifikasi (sumber: Dokumen penulis)

Pada gambar 4 di atas, dapat ditunjukkan bahwa pesan tersebut berhasil didekripsi menggunakan kunci privat yang sesuai ditandai dengan isi pesan yang memiliki makna di dalamnya. Kemudian pesan juga menunjukkan status bahwa pesan tersebut berhasil diverifikasi sehingga keaslian pesan tersebut tetap terjaga dan pesan memang dikirimkan dari pihak sender yang asli.

2. Kasus 2: *Encrypted* message didekripsi dengan menggunakan kunci privat dekripsi yang tidak sesuai.

   Dengan mengambil pesan yang sama pada kasus 1, kunci privat receiver semula dimodifikasi dari 13744231093667, 87093738739141 menjadi 13744231093668, 87093738739141.

   Berikut ini adalah tampilan dari plainteks hasil dekripsi apabila digunakan kunci privat enkripsi yang tidak sesuai.



Gambar 5. Pesan gagal didekripsi akibat kunci privat yang tidak sesuai (sumber: Dokumen penulis)

Perhatikan bahwa pesan hasil dekripsi tidak menggambarkan pesan asli yang sesungguhnya, melainkan sebuah teks tanpa makna. Hal ini disebabkan karena kunci privat dekripsi tidak cocok digunakan untuk mendekripsi pesan tersebut. Dengan demikian, hanya *receiver* asli yang dapat membuka pesan tersebut karena *receiver* memiliki kunci privat yang cocok untuk membuka pesan tersebut.

3. Kasus 3: Pesan ditandatangani dan diverifikasi dengan pasangan kunci *digital signature* yang tidak sesuai.

   Pada kasus ini, terdapat pesan baru yang dikirimkan oleh pihak Shadow Bank yang berisi pesan sebagai berikut.

> Nasabah Yth. Saat ini Shadow Bank telah membuka produk asuransi kesehatan yang bisa Anda dapatkan secara gratis selama 6 bulan pertama. Klik link di bawah ini untuk informasi lebih lanjut. s.id/ShadowBankAsuransi.

   Asumsikan bahwa pesan tersebut bukan dikirimkan oleh pihak Shadow Bank secara resmi, melainkan pihak ketiga yang berhasil menyamar sebagai Shadow Bank dan mengirimkan pesan tersebut kepada nasabah Michael Hans. Dalam kasus ini, kunci privat untuk

tanda tangan berbeda dengan yang tersimpan pada perangkat asli dari Shadow Bank.

- Kunci privat tanda tangan asli: 3540443, 1396201978657
- Kunci privat tanda tangan palsu: 8652971, 568091892103

Berikut ini adalah *input* dan *response* yang diterima dari pengiriman pesan pada kasus ke-3 ini.

API Endpoint: /add
Method: POST
Body:


```
{
   "sender": "ShadowBank",
   "receiver": "MichaelHans",
   "message": "Nasabah Yth. Saat ini Shadow Bank telah membuka produk asuransi kesehatan yang bisa Anda dapatkan secara gratis selama 6 bulan pertama. Klik link di bawah ini untuk informasi lebih lanjut. s.id/ShadowBankAsuransi."
}
```


Public key receiver: 4851211,87093738739141
Signing private key sender: 8652971,568091892103

Message + signature: “Nasabah Yth. Saat ini Shadow Bank telah membuka produk asuransi kesehatan yang bisa Anda dapatkan secara gratis selama 6 bulan pertama. Klik link di bawah ini untuk informasi lebih lanjut. s.id/ShadowBankAsuransi.<ds>1108cf2e3b857537936c37361a5ed405089aa91f76e5cfe73519b0c01e27f22299cb67f06a80b873d769b1ceaafddbc9392735494bca558d29b316bd2be1dbae8a8e881fc8a</ds>

Response:


```
{
      "message": "13b3e5aab8e926fdfa7655530cebc7c018a671658053776b2ad494d90a347d6752d870da59c01ce48c1ab8a6c8372fed25176307939377bfc0598e39579288c170d872bfdfbda25c28123d05c01050b2dc4aeb1b985921d83e3e899e644e3ab594b1b431de3caaafdf3e34fcb0f9f2f07e5a8f4251041f23329cea8ac74c095e0211e843edc0ea6a558e3715fd844e90ce043b608ebdd214e0e300e1d8740e983b0641194e3092399f5d44f28a7c51630d634261dd2c5e587aa77641d32605d2a3dfc26b81e76beb503ab20d53dfa9e54102844ad91dd9fabeb97faa5e5dee5b738434f058453afb720194269727233e9fa5edd8035407dc44fcfc423ea84b23cdc09951822630a891ec924bd5a962c2781544baa5815d30aae27a50c994edb4865ab1e43b6b5e9cf0d55fe9bc5f5819291c5361651f1835d947f430f83804e",
      "receiver": "MichaelHans",
      "sender": "ShadowBank",
      "timestamp": "2021-12-17 00:46:38.431836"
}
```


Pada sisi Front End, user akan login sebagai “MichaelHans” untuk mengakses *messaging app* dan melihat pesan yang masuk dari ShadowBank. Untuk mengambil seluruh *messages*, aplikasi melakukan GET Request ke endpoint messages sebagai berikut.

API Endpoint: /messages
Method: GET
Params: receiver= MichaelHans

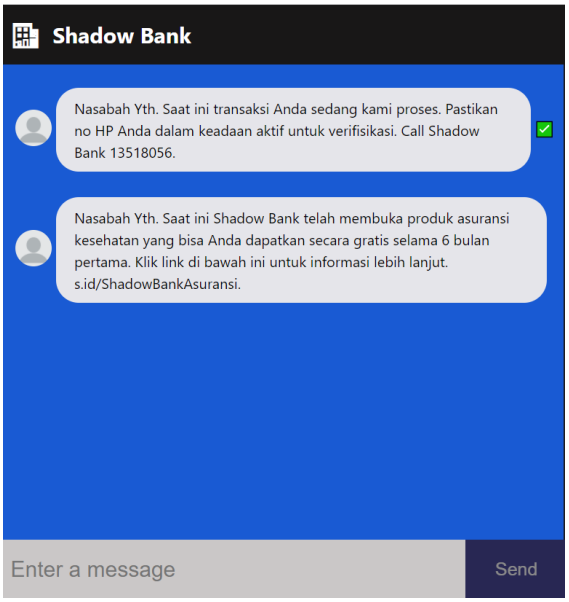
Private key receiver: 13744231093667,87093738739141
Verifying public key sender: 548864962643,1396201978657

Proses Verifikasi untuk Pesan Kedua:
Message + signature: “Nasabah Yth. Saat ini Shadow Bank telah membuka produk asuransi kesehatan yang bisa Anda dapatkan secara gratis selama 6 bulan pertama. Klik link di bawah ini untuk informasi lebih lanjut. s.id/ShadowBankAsuransi.<ds>1108cf2e3b857537936c37361a5ed405089aa91f76e5cfe73519b0c01e27f22299cb67f06a80b873d769b1ceaafddbc9392735494bca558d29b316bd2be1dbae8a8e881fc8a</ds>”
Message digest:
f82c82166f06b54c7315f9971348976f7cf698db704403fefa93de4cba45c6e0
Message digest’’:
s:♫&¹âz¥È▼ÏLZ→]7X0ÏG3?ùªaZ`j*Ëw²↨mXÄÓc¼åKõªÆ çTj¨»~¶=>á9¥

Response:


```
[
   {
      "message": "Nasabah Yth. Saat ini transaksi Anda sedang kami proses. Pastikan no HP Anda dalam keadaan aktif untuk verifisikasi. Call Shadow Bank 13518056.",
      "receiver": "MichaelHans",
      "sender": "ShadowBank",
      "timestamp": "2021-12-17 00:46:38.431836",
      "verify": true
   },
   {
      "message": "Nasabah Yth. Saat ini Shadow Bank telah membuka produk asuransi kesehatan yang bisa Anda dapatkan secara gratis selama 6 bulan pertama. Klik link di bawah ini untuk informasi lebih lanjut. s.id/ShadowBankAsuransi.",
      "receiver": "MichaelHans",
      "sender": "ShadowBank",
      "timestamp": "2021-12-17 01:20:01.292357",
      "verify": false
   }
]
```


Message yang diperoleh kemudian akan ditampilkan pada sisi Front End. Tampilan pesan dari sisi Front End dapat dilihat pada gambar berikut ini.



Gambar 6. Kedua pesan berhasil didekripsi namun pesan kedua tidak terverifikasi
(sumber: Dokumen penulis)

Tangkapan layar di atas menunjukkan bahwa kedua pesan berhasil didekripsi sehingga keduanya berhasil menggambarkan pesan yang bermakna. Namun, terdapat perbedaan antara kedua pesan tersebut, yaitu pesan pertama terverifikasi dan pesan kedua tidak

terverifikasi dengan tidak adanya simbol verifikasi di samping kanan pesan tersebut.

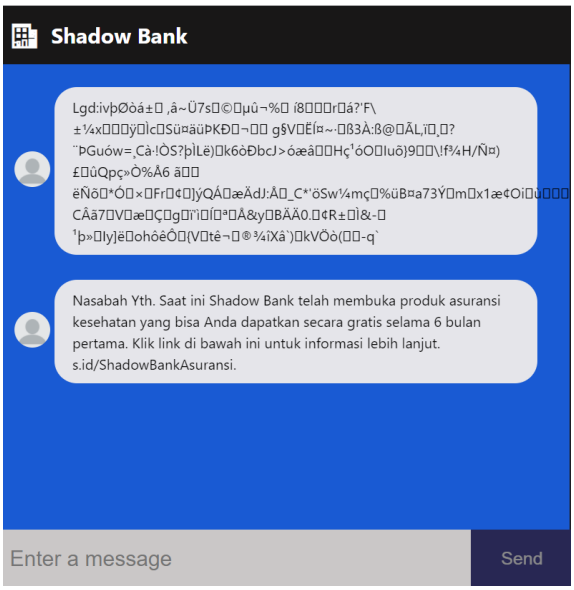
Hal tersebut disebabkan karena kunci privat yang digunakan untuk penandatanganan tidak cocok dengan kunci publik yang digunakan untuk verifikasi. Hal ini dapat terjadi ketika ada pihak ketiga yang berpura-pura berperan sebagai entitas berwenang. Namun karena kunci privat yang digunakan pihak ketiga tidak sinkron dengan kunci publik pada repositori, maka pesan tersebut tergolong pesan yang tidak terverifikasi.

4. Kasus 4: Penambahan satu karakter pada *encrypted-message* pada repositori.

Pada kasus ini, *encrypted message* dari pesan pertama pada kasus ke-3 akan disisipkan sebuah karakter sembarang di dalamnya. Penyisipan karakter tersebut menghasilkan sebuah *updated message* sebagai berikut.


```
{
    "message":
"13b3e5aab8e926fdfa7655530cebc7c018a6716580537776b2ad494d
90a347d6752d870da59c01ce48c1ab8a6c8372fed25176307939377
bfc0598e39579288c170d872bfdfbda25c28123d05c01050b2dc4aeb
1b985921d83e3e899e644e3ab594b1b431de3caaafdf3e34fcb0f9f2f
07e5a8f4251041f23329cea8ac74c095e0211e843edc0ea6a558e371
5fd844e90ce043b608ebdd214e0e300e1d8740e983b0641194e3092
399f5d44f28a7c51630d634261dd2c5e587aa77641d32605d2a3dfc
26b81e76beb503ab20d53dfa9e54102844ad91dd9fabeb97faa5e5de
e5b738434f058453afb720194269727233e9fa5edd8035407dc44fcf
c423ea84b23cdc09951822630a891ec924bd5a962c2781544baa581
5d30aae27a50c994edb4865ab1e43b6b5e9cf0d55fe9bc5f5819291c
5361651f1835d947f430f83804e1",
    "receiver": "MichaelHans",
    "sender": "ShadowBank",
    "timestamp": "2021-12-17 00:46:38.431836"
}
```


Berikut ini adalah pesan yang muncul pada layar *receiver* ketika seluruh pesan didekripsi dengan menggunakan kunci privat yang bersesuaian.

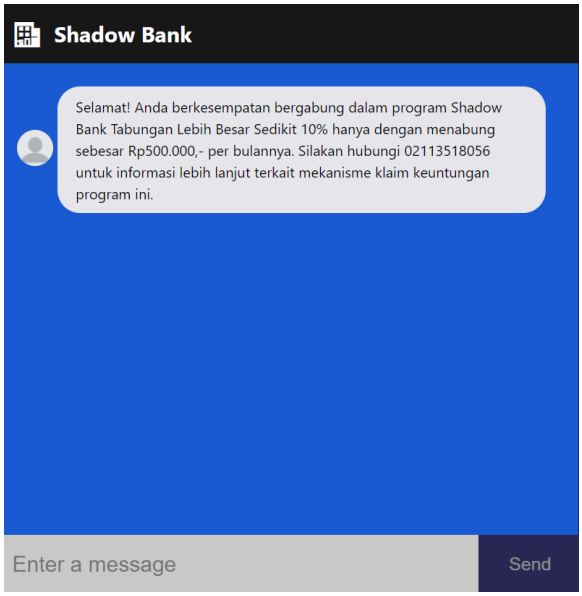


Gambar 7. Pesan pertama gagal didekripsi akibat penyisipan satu karakter pada *encrypted message* (sumber: Dokumen penulis)

Tangkapan layar di atas menunjukkan bahwa pesan pertama gagal didekripsi akibat penambahan satu buah karakter pada *encrypted message*. Akibat dari penyisipan tersebut adalah pesan gagal didekripsi secara otomatis sehingga pesan secara otomatis tidak terverifikasi. Kejadian ini dapat terjadi apabila terdapat pihak ketiga yang memiliki akses ke repositori memperbaharui data pesan pada repositoro secara sengaja. Karena pesan tersebut gagal didekripsi, pihak receiver segera mengetahui bahwa pesan tersebut sudah dimodifikasi oleh pihak ketiga.

5. Kasus 5: Pesan tanpa tanda tangan digital

Pesan juga dapat dikirimkan oleh pihak Shadow Bank tanpa adanya tanda tangan digital di dalamnya. Ketika pesan tersebut dibaca dari sisi *receiver*, maka pesan tersebut secara otomatis tergolong pesan yang tidak terverifikasi seperti pada Gambar 8 berikut ini.



Gambar 8. Pesan tanpa tanda tangan digital berstatus tidak terverifikasi (sumber: Dokumen penulis)

C. Analisis Secara Keseluruhan

Berdasarkan hasil pengujian yang ada, penulis menarik beberapa poin penting terkait pengamanan pesan dan identifikasi pengirim dengan menggunakan algoritma RSA. Algoritma RSA dinilai cocok untuk proses *end-to-end encryption* karena pesan tersebut hanya dapat dibaca oleh *receiver* yang memiliki kunci privat untuk mendekripsinya. Segala perubahan atau modifikasi yang terjadi baik pada *encrypted message* bisa segera diketahui oleh *receiver* ketika hasil dekripsi pada *encrypted message* tidak memberikan pesan yang bermakna.

Selain itu, proses identifikasi keaslian pengirim dapat dilakukan dengan menggunakan algoritma RSA yang diterapkan dalam bentuk tanda tangan digital. Suatu pesan berhasil diverifikasi apabila tanda tangan digital yang disisipkan pada pesan ditandatangani dan diverifikasi menggunakan pasangan kunci *signature* yang bersesuaian. Apabila terdapat sepasang kunci yang digunakan tidak cocok, maka pesan juga secara otomatis tidak terverifikasi. Hal tersebut juga berlaku apabila pesan tidak ditandatangani oleh pengirim. Pesan tanpa tanda tangan memiliki makna yang sama dengan dokumen yang tidak ditandatangani, yaitu pesan yang belum mendapatkan persetujuan dari pembuat pesan.

Apabila pesan tersebut dikirimkan oleh pihak khusus seperti bank, rumah sakit, atau pemerintah, maka pesan tersebut menjadi diragukan keasliannya apabila tidak ada status terverifikasi pada pesan yang diterima oleh *receiver*. Oleh karena itu, algoritma RSA yang diterapkan berhasil melakukan dua hal kriptografi sekaligus pada *messaging app*, yaitu *end-to-end encryption* dan proses identifikasi entitas khusus dengan tujuan untuk memastikan bahwa pesan dikirimkan oleh pengirim yang asli.

V. KESIMPULAN

Algoritma RSA adalah salah satu algoritma kriptografi kunci publik yang banyak digunakan dalam pengamanan pesan. Selain pengamanan pesan, algoritma RSA juga dapat dikombinasikan dengan mekanisme *digital signature* yang berfungsi untuk melakukan proses identifikasi entitas khusus pada *messaging app*. Dengan demikian, keamanan dalam layanan pengiriman pesan tidak hanya dari sisi *end-to-end encryption*, melainkan adanya proses verifikasi tertentu agar pesan yang dikirimkan memang asli dari pihak pengirim sehingga pesan fiktif dan pesan penipuan lainnya dapat segera diketahui dengan melihat status verifikasi suatu pesan.

UCAPAN TERIMA KASIH

Penulis mengucapkan puji dan syukur yang sebesar-besarnya kepada rahmat Tuhan Yang Maha Esa atas berkat dan rahmat-Nya sehingga makalah berjudul “Penerapan Algoritma RSA pada Enkripsi Pengiriman Pesan dan Proses Identifikasi Entitas Tertentu pada Messaging App” dapat diselesaikan dengan baik. Penulis mengucapkan terima kasih kepada selaku dosen pengajar IF4020 Kriptografi, Dr. Rinaldi Munir, S.T, M.T. yang telah memberikan bimbingan dan ilmu terkait materi Kriptografi ini, khusunya pada algoritma kriptografi kunci publik dan pengaplikasiannya. Penulis juga mengucapkan terima kasih kepada teman-teman yang telah memberikan dukungan kepada penulis untuk menuliskan makalah ini. Tak lupa penulis juga mengucapkan terima kasih kepada para penulis sumber referensi yang telah memberikan ilmu yang dibutuhkan penulis untuk menyelesaikan makalah ini.

REFERENCES

[1] Munir, Rinaldi. 2021. Slide Kuliah Kriptografi (Bandung: Institut Teknologi Bandung)

[2] Team, K., Aver, H., & Starikova, A. (n.d.). *What is end-to-end encryption and what are its pros and cons.* Daily English Global blogkasperskycom. [Online]. Available: https://www.kaspersky.com/blog/what-is-end-to-end-encryption/37011/. [Accessed: 13-Dec-2021].

[3] Jacobo Quintanilla and Philippe Stoll. “Messaging apps,” *Engine Room*. [Online]. Available: https://library.theengineroom.org/messaging-apps/. [Accessed: 17-Dec-2021].

[4] T. G., “10 most secure messaging apps - best encrypted Chat App Solutions,” *Stream RSS*, 05-Feb-2015. [Online]. Available: https://getstream.io/blog/most-secure-messaging-apps/. [Accessed: 17-Dec-2021].

PERNYATAAN

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

Jakarta, 18 Desember 2021

Michael Hans
13518056