

LAPORAN TUGAS 3

IF4020 Kriptografi

**Penerapan *Elliptic Curve Cryptography* ECDSA dan SHA-3 untuk
Menandatangani Surel pada Perangkat *Mobile***



oleh :

Bintang Fajarianto / 13519138

**PROGRAM STUDI TEKNIK INFORMATIKA
SEKOLAH TEKNIK ELEKTRO DAN INFORMATIKA
INSTITUT TEKNOLOGI BANDUNG**

2022

A. Deskripsi Masalah

Pada era digital ini, informasi menjadi hal yang penting dan berharga sehingga perlu dilindungi dari pihak-pihak yang tidak berkepentingan. Surel menjadi salah satu wadah yang rawan untuk disadap dan diakses oleh pihak yang tidak sah. Sebagian besar aplikasi email yang beredar di masyarakat belum memiliki fitur enkripsi untuk menjaga kerahasiaan pesan dan juga tanda tangan digital untuk mengonfirmasi keaslian dari pesan elektronik dan juga keperluan otentikasi pengirim serta penerima pesan. Oleh karena itu, pada tugas kali ini akan dibangun sebuah aplikasi email yang mampu mengenkripsi pesan yang akan dikirim dan menambahkan serta memverifikasi tanda tangan digital pada suatu pesan.

B. Dasar Teori Singkat

a. Tanda Tangan Digital

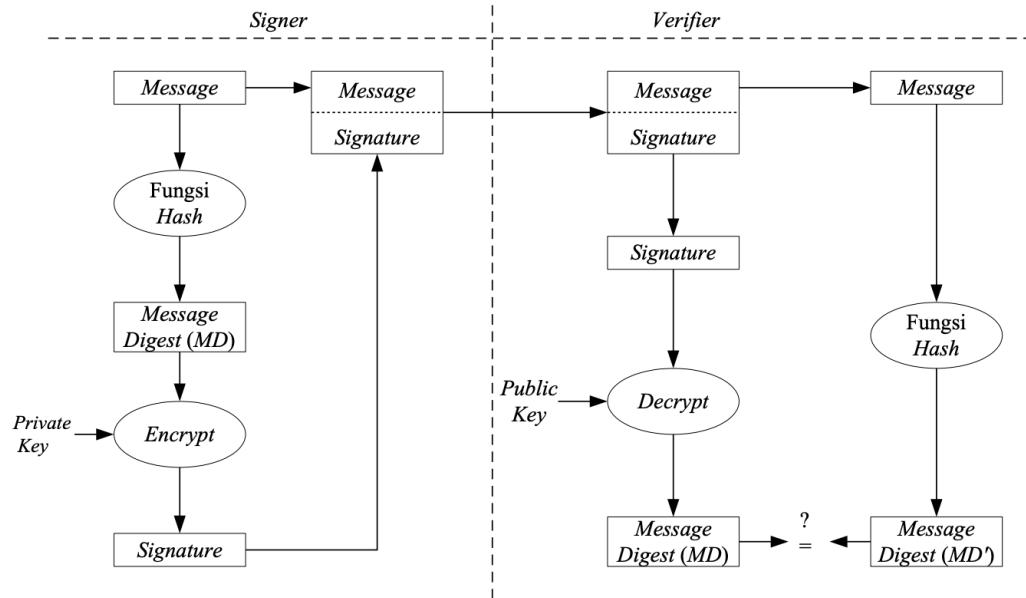
Tanda tangan digital adalah sebuah nilai kriptografis yang digunakan untuk memastikan keaslian pesan dan pengirim pesan elektronik sehingga pesan yang diterima terverifikasi dari pihak yang sah dan tidak mengalami perubahan selama proses pengiriman. Tanda tangan digital (*digital signature*) tidak sama dengan tanda tangan yang didigitasi (*digitized signature*). Tanda tangan digital bergantung pada isi pesan dan kunci yang digunakan, sehingga akan selalu berbeda. Sementara itu, tanda tangan yang didigitasi dengan cara dipindai atau difoto akan selalu sama.

Tanda tangan digital memiliki dua proses utama, yaitu penambahan tanda tangan pada pesan (*signing*) dan verifikasi tanda tangan pada pesan (*verifying*). Tanda tangan digital dibuat dengan memanfaatkan pasangan kunci kriptografi, yaitu kunci privat dan kunci publik yang dibangkitkan secara berpasangan. Kunci privat digunakan untuk menandatangani sebuah pesan elektronik, sedangkan kunci publik digunakan untuk memverifikasi tanda tangan pada sebuah pesan elektronik.

Pada proses pembuatan tanda tangan digital, pertama-tama isi pesan akan diambil untuk dilakukan proses *hashing* dengan algoritma *hash*. Pada tugas ini, algoritma *hash* yang akan digunakan adalah SHA-3 atau Keccak yang dibuat sendiri. Hasil dari proses *hash* tersebut kemudian akan dienkripsi dengan kunci privat dengan algoritma yang memanfaatkan pasangan kunci privat dan kunci publik, seperti RSA dan DSA. Pada tugas ini, algoritma yang akan digunakan adalah ECDSA yang dibuat sendiri. Hasil dari proses enkripsi inilah yang disebut dengan tanda tangan digital.

Sementara itu, dalam proses verifikasi tanda tangan, pertama-tama pesan yang telah ditandatangani akan diambil. Isi dari pesan tersebut akan dilakukan proses *hashing* dengan algoritma yang sama seperti saat membuat tanda tangan digital. Selain itu, tanda tangan digital hasil proses enkripsi menggunakan kunci privat akan didekripsi menggunakan kunci publik yang merupakan pasangan dari kunci privat saat pembangkitan kunci.

Hasil proses dekripsi dan hasil dari proses hash akan dibandingkan dalam proses verifikasi tanda tangan digital. Apabila kedua nilai sama, maka tanda tangan digital dinyatakan valid atau pesan yang dikirimkan berasal dari pihak yang sah dan tidak mengalami perubahan selama proses pengiriman.

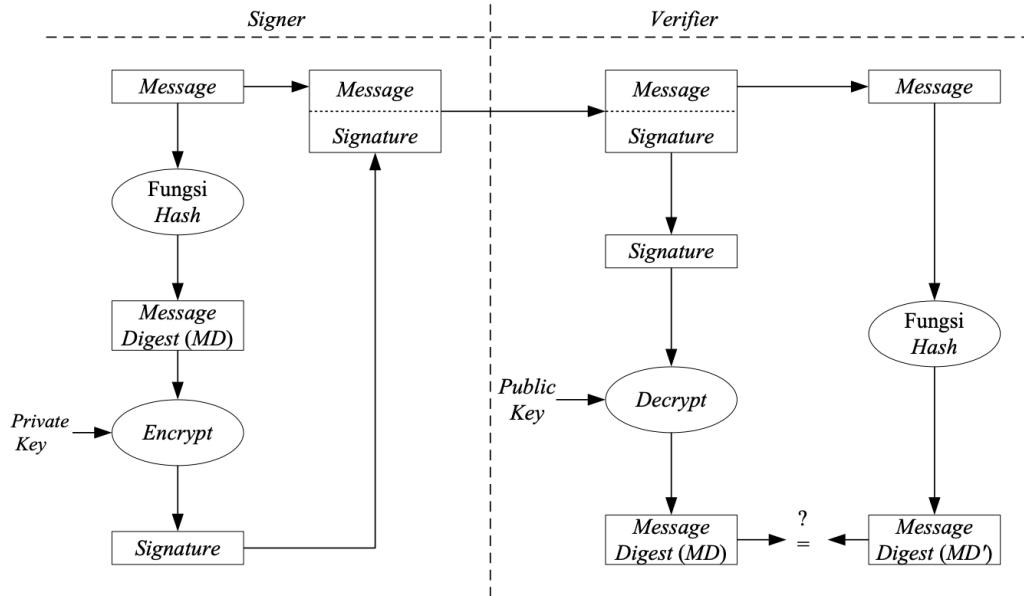


Gambar B.1. Proses Tanda Tangan Digital

b. SHA3 (Keccak)

SHA3 (*Secure Hash Algorithm 3*) adalah sebuah fungsi *hash* yang didesain oleh Guido Bertoni, Joan Daemen, Michaël Peeters, dan Gilles van Assche. Fungsi *hash* yang bernama Keccak ini merupakan pemenang *Cryptographic Hash Algorithm Competition* yang diselenggarakan oleh NIST pada tahun 2007 sampai 2012. Nama Keccak berasal dari bahasa Belanda yang memiliki arti memeras karena algoritma ini memeras data masukan menjadi sebuah nilai *hash*.

Algoritma SHA3 menerima masukan dengan ukuran yang beragam dan menghasilkan luaran yang berukuran tetap. Algoritma SHA3 mampu menghasilkan luaran dengan ukuran panjang hash yang diinginkan, seperti 256-bit, 384-bit, 512-bit, 1024-bit, atau n-bit. Dalam kompetisi yang dimenangkan oleh tim Keccak, algoritma ini menggunakan pendekatan konstruksi spons (*sponge construction*) yang mengabsorpsi masukan ke dalam sebuah *state internal* berukuran sama dengan luaran dan selanjutnya state tersebut akan dilakukan proses permutasi dengan operasi XOR dan rotasi bit. Luaran *hash* diambil dari sebagian *state internal* yang telah diproses berulang kali. Kebanyakan algoritma lain bergantung pada fungsi kompresi, namun algoritma ini menggunakan fungsi nonkompreksi untuk menyerap dan memeras digest. Berikut ini adalah ilustrasi dari algoritma SHA3.



Gambar B.2. Ilustrasi Algoritma Keccak

c. ECDSA

ECDSA (*Elliptic Curve Digital Signature Algorithm*) adalah sebuah algoritma tanda tangan digital yang menggunakan kurva eliptik sebagai perluasan dari sistem kriptografi kunci berpasangan untuk melakukan proses enkripsi dan dekripsi. Seperti algoritma tanda tangan digital lainnya, ECDSA menggunakan kunci privat untuk proses penambahan tanda tangan digital dan kunci publik untuk proses verifikasi tanda tangan digital.

Proses pembuatan tanda tangan digital menggunakan algoritma ECDSA dilakukan dengan mengambil *hash* pesan dari hasil proses *hashing* suatu isi pesan dengan algoritma *hash*. Selanjutnya akan dipilih sebuah bilangan acak yang disebut ‘nonce’ dan akan dihitung titik publik pada kurva eliptik tersebut. Titik ini akan dioperasikan menjadi sebuah bilangan representasi tanda tangan digital dengan menggunakan kunci privat dan operasi matematika atau dapat disebut sebagai *modular repetition*.

Sementara itu, dalam proses verifikasi tanda tangan digital menggunakan algoritma ECDSA, pesan yang telah ditandatangani akan diambil untuk dilakukan proses *hashing* dengan algoritma yang sama seperti saat membuat tanda tangan digital. Selain itu, tanda tangan digital yang terdapat pada pesan tersebut juga diambil. Selanjutnya akan dihitung titik publik pada kurva eliptik yang berkaitan menggunakan kunci publik yang berpasangan dengan kunci privat yang digunakan untuk menambahkan tanda tangan digital. Tanda tangan digital dinyatakan valid apabila titik publik tersebut sama dengan titik publik yang dihasilkan pada proses pembuatan tanda tangan digital.

d. Bystar Cipher Blok

Bystar cipher blok adalah algoritma cipher blok yang diimplementasikan sendiri oleh penulis. Sama seperti algoritma cipher blok lainnya, algoritma ini membagi pesan menjadi blok-blok bit yang berukuran sama dan proses enkripsi dilakukan pada setiap blok tersebut.

Dalam melakukan proses enkripsi dan dekripsi, algoritma Bystar membutuhkan kunci dengan panjang 128-bit dan membagi pesan menjadi blok-blok berukuran 128-bit. Algoritma ini cukup fleksibel karena dalam algoritma ini diterapkan validasi kunci jika ukuran tidak berjumlah 128-bit. Algoritma ini bekerja dengan baik karena telah menerapkan prinsip-prinsip cipher blok yang baik, seperti menerapkan prinsip *confusion* dengan melakukan substitusi, prinsip *diffusion* dengan melakukan permutasi, serta jaringan feistel beserta fungsi putaran yang kompleks sebanyak 16 kali.

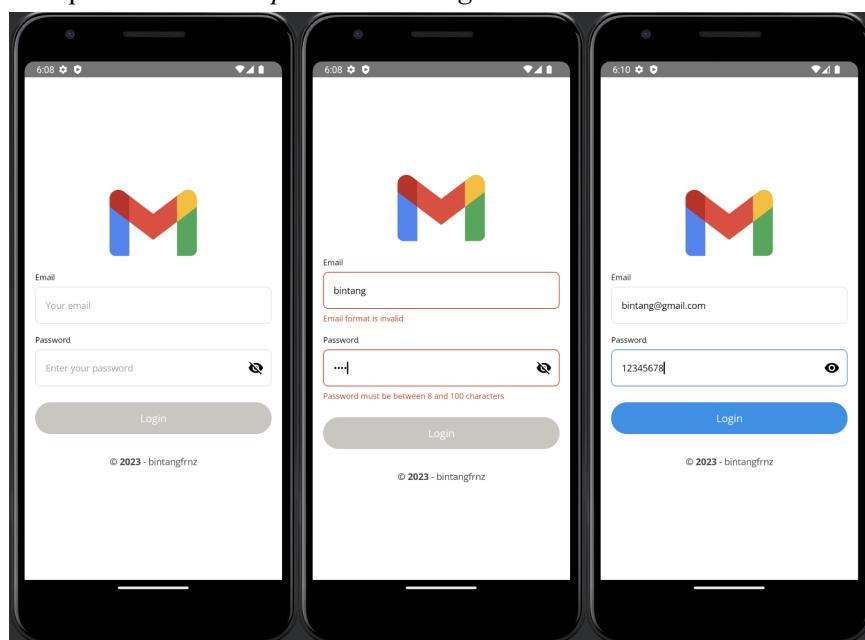
C. Rancangan Program dan Implementasi

Dalam proses perancangan pembangunan aplikasi ini, awalnya terdapat aplikasi *client* dan server yang akan dibuat. Server dan backend direncanakan akan diimplementasi menggunakan bahasa Python 3 karena cipher blok yang dibangun pada tugas sebelumnya menggunakan Python 3. Aplikasi *client* yang akan dibangun direncanakan akan menyesuaikan temuan hasil riset terkait *open source email client*. Sayangnya, penulis belum berhasil menemukan *open source client email* yang dapat digunakan untuk mengerjakan tugas ini.

Pada rencana berikutnya, penulis masih berencana untuk membangun aplikasi *client* dan server dengan aplikasi client dibuat dari awal disertai Gmail API untuk kebutuhan data pesan dan fungsi kirim pesan. Namun, karena keterbatasan penulis dalam mengimplementasikan kode backend, aplikasi ini belum berhasil untuk menggunakan Gmail API untuk proses pengiriman pesan. Oleh karena itu, penulis berencana untuk membangun aplikasi yang dapat menyimulasikan proses pengiriman pesan yang disertai fitur enkripsi dan tanda tangan digital menggunakan Jetpack Compose dengan *third party* untuk menjalankan kode Python 3 di dalam Kotlin Android sehingga penulis tidak perlu menyiapkan server dan membuat kode backend. Seluruh proses dilakukan pada sisi *client*.

a. Autentikasi

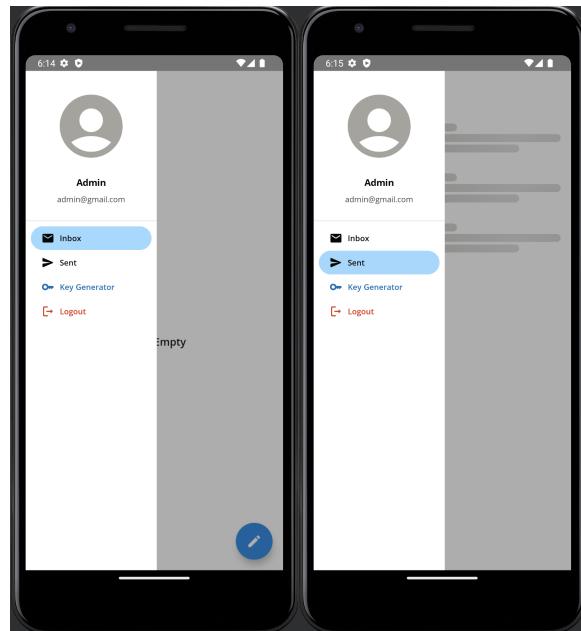
Saat pertama kali membuka aplikasi, terdapat *field* email dan password yang dapat diubah oleh pengguna. *Field* tersebut telah memiliki validator untuk format email dan juga format password (minimal 8 karakter). Namun, karena program ini hanya menyimulasikan proses pengiriman pesan, setiap email dan password yang dimasukkan akan ditolak dan hanya email admin@gmail.com dan password admin123 yang akan diterima. Selanjutnya, data pengguna tersebut akan disimpan ke dalam *simple secure storage*.



Gambar C.1. Login

b. Home

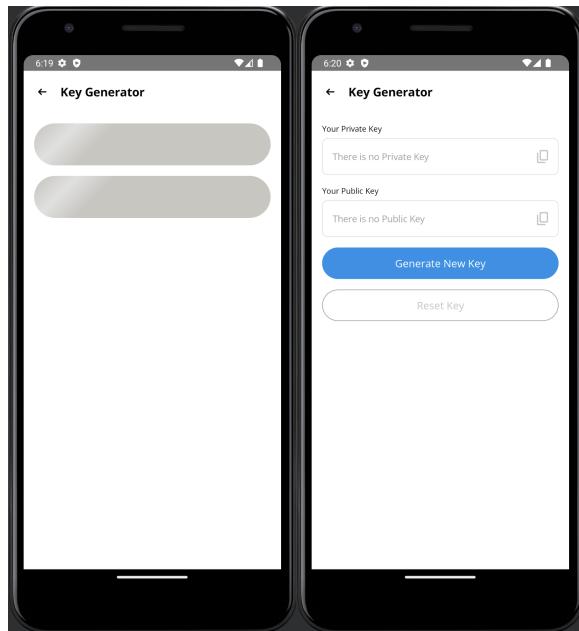
Setelah berhasil melakukan login, pengguna akan diarahkan menuju halaman Home. Pada halaman ini, untuk mendukung proses simulasi pengiriman pesan, akan diimplementasikan folder Inbox dan folder Sent yang dapat diakses melalui *navigation drawer*. Selain itu, dalam drawer tersebut pengguna juga dapat berpindah ke halaman pembangkitan kunci dan melakukan proses logout. Untuk berpindah ke halaman kirim pesan, akan disediakan *floating action button* di halaman Home.



Gambar C.3. Home

c. Key Generator

Pasangan kunci privat dan publik dibutuhkan untuk proses penambahan dan verifikasi tanda tangan digital. Aplikasi ini akan menyediakan pembangkit kunci untuk mempermudah proses penambahan dan verifikasi tanda tangan digital. Kunci yang telah dibangkitkan akan disimpan ke dalam *simple secure storage* yang tidak akan hilang meskipun pengguna melakukan logout.



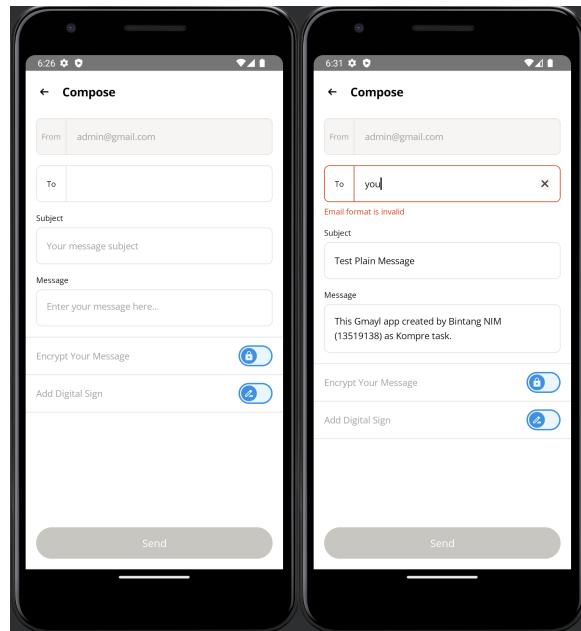
Gambar C.4. Key Generator

d. Send Mail

Dalam membangun fitur kirim pesan, terdapat tiga *field* utama yang akan disediakan untuk diisi, yaitu penerima, subjek pesan, dan isi pesan. Pesan yang terkirim akan disimpan ke dalam tabel SentMail. Apabila penerima pesan adalah pengguna saat ini, yaitu admin@gmail.com, maka pesan yang terkirim juga akan disimpan ke dalam tabel InboxMail.

Berdasarkan spesifikasi dari tugas ini, fitur ini juga harus menyertakan opsi proses enkripsi dan penambahan tanda tangan digital. Kedua fitur ini akan dibangun dengan menggunakan *toggle switch*. Apabila *toggle* berada dalam kondisi *on*, pengguna dapat memasukkan kunci untuk melakukan proses enkripsi ataupun penambahan tanda tangan digital. Apabila yang hendak dilakukan adalah proses enkripsi, maka yang dibutuhkan adalah kunci simetris. Sedangkan jika yang hendak dilakukan adalah proses penambahan tanda tangan digital, maka yang dibutuhkan adalah kunci privat.

Proses enkripsi dilakukan dengan Bystar Block Cipher yang telah dibangun pada tugas sebelumnya. Algoritma enkripsi tersebut diimplementasikan dalam bahasa Python 3 yang akan dipanggil dari aplikasi jika dibutuhkan. Algoritma Bystar Block Cipher mengenkripsi pesan menggunakan kunci 16 bytes pada level blok bit dengan ukuran 128 bit. Sementara itu, proses penambahan tanda tangan digital akan dilakukan dengan mengenkripsi hash dari pesan menggunakan algoritma Keccak yang dibangun sendiri, dengan algoritma ECDSA yang juga dibangun sendiri.



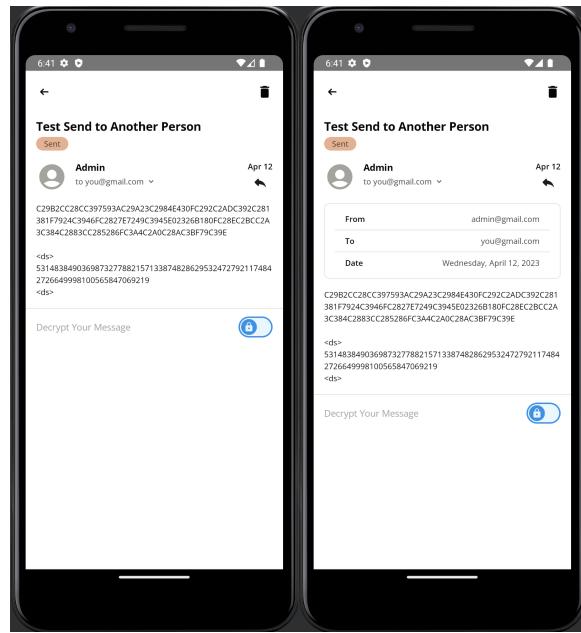
Gambar C.5. Send Mail

e. Mail Detail

Dalam membangun fitur detail pesan ini, tiga *field* utama yang terdapat dalam fitur kirim pesan, yaitu penerima, subjek pesan, dan isi pesan, akan ditampilkan dengan *field* tambahan, seperti waktu pesan terkirim. Detail pesan yang berasal dari tabel SentMail akan memiliki tag “Sent”, sedangkan detail pesan yang berasal dari tabel InboxMail akan memiliki tag “Inbox”. Pesan yang terkirim ke email pengguna saat ini, yaitu admin@gmail.com, akan memiliki format khusus yaitu menggunakan kata “me” dibandingkan menggunakan email..

Berdasarkan spesifikasi dari tugas ini, fitur ini juga harus menyertakan opsi proses dekripsi dan verifikasi tanda tangan digital. Kedua fitur ini akan dibangun dengan menggunakan *toggle switch* sama seperti pada fitur kirim pesan. Apabila *toggle* berada dalam kondisi *on*, pengguna dapat memasukkan kunci untuk melakukan proses dekripsi ataupun verifikasi tanda tangan digital. Apabila yang hendak dilakukan adalah proses dekripsi, maka yang dibutuhkan adalah kunci simetris. Sedangkan jika yang hendak dilakukan adalah proses verifikasi tanda tangan digital, maka yang dibutuhkan adalah kunci publik. Apabila yang akan dilakukan adalah keduanya, maka proses dekripsi harus dilakukan terlebih dahulu sebelum melakukan verifikasi tanda tangan digital karena pesan yang dijadikan sebagai referensi untuk proses verifikasi tanda tangan digital adalah pesan hasil dekripsi.

Proses dekripsi juga dilakukan dengan Bystar Block Cipher yang telah dibangun pada tugas sebelumnya. Algoritma dekripsi tersebut diimplementasikan dalam bahasa Python 3 yang akan dipanggil dari aplikasi jika dibutuhkan. Sementara itu, proses verifikasi tanda tangan digital akan dilakukan dengan mendekripsi *signature* dan membandingkannya dengan hash dari pesan aslinya.



Gambar C.6. Mail Detail

D. Pengujian dan Analisis Hasil

a. Uji Pembangkitan Kunci

Pada aplikasi ini, proses pembangkitan kunci dapat dilakukan pada halaman Key Generator. Pembangkitan kunci dilakukan dengan memanggil kode Python 3 dari dalam aplikasi Android menggunakan *third party* bernama Chaquopy.

Saat pertama kali menjalankan aplikasi, pengguna tidak memiliki pasangan *public key* dan *private key*. Pengguna dapat membangkitkan kunci dengan menekan tombol *generate key* yang ada pada halaman tersebut. Pasangan kunci yang dihasilkan akan disimpan ke dalam *simple secure key-value storage* yang disediakan oleh *third party* bernama hawk. Pengguna dapat membangkitkan ulang pasangan kunci baru yang akan menimpa penyimpanan kunci yang lama serta menghapus pasangan kunci yang lama. Pasangan kunci ini tidak akan terhapus apabila pengguna memilih untuk keluar atau *logout* dari aplikasi.

Untuk melihat proses pembangkitan aplikasi, dapat dilihat pada lampiran Fitur: **Key Generator (KG)** dalam Gambar F.6. sampai Gambar F.9.

b. Uji Enkripsi Pesan

Pada aplikasi ini, proses enkripsi pesan dapat dilakukan pada halaman Send Mail. Proses enkripsi pesan dilakukan dengan memanggil fungsi *encrypt* dengan parameter *plain_text* dan *key* yang diimplementasikan pada kode Python 3 dari dalam aplikasi Android menggunakan *third party* bernama Chaquopy.

Saat berada di halaman tersebut, pengguna dapat memilih untuk mengenkripsi pesan yang akan dikirimkan atau tidak menggunakan sebuah *toggle switch*. Apabila toggle berada pada posisi on, pengguna dapat menekan “Encrypt Your Message” untuk menampilkan sebuah *bottomsheet*. Di dalam *bottomsheet* tersebut, disediakan sebuah *text input* dengan sebuah validator untuk pengguna memasukkan kunci simetris yang akan dipakai dalam proses enkripsi. Aplikasi ini juga mengimplementasikan sebuah *snackbar* untuk menampilkan informasi apabila kunci berhasil ditambahkan, diganti, atau dihapus. Setelah pesan tersebut terkirim, kita dapat melihat pada folder Inbox / Sent bahwa isi pesan tersebut telah terenkripsi.

Untuk melihat proses enkripsi pesan, dapat dilihat pada lampiran Fitur: **Send Mail (SM)** dalam Gambar F.13. sampai Gambar F.15.

c. Uji Dekripsi Pesan

Pada aplikasi ini, proses dekripsi pesan dapat dilakukan pada halaman Mail Detail. Proses dekripsi pesan dilakukan dengan memanggil fungsi *decrypt* dengan parameter *hex_text* dan *key* yang diimplementasikan pada kode Python 3 dari dalam aplikasi Android menggunakan *third party* bernama Chaquopy.

Saat berada di halaman tersebut, pengguna dapat memilih untuk mengdekripsi pesan yang telah kita kirim / terima atau tidak menggunakan sebuah *toggle switch*. Apabila toggle berada pada posisi on, pengguna dapat menekan “Decrypt Your Message” untuk menampilkan sebuah *bottomsheet*. Di dalam *bottomsheet* tersebut, disediakan sebuah *text input* dengan sebuah validator untuk pengguna memasukkan kunci simetris yang akan dipakai dalam proses dekripsi. Saat pengguna melakukan submit kunci yang dimasukkan, maka di bawah *toggle switch* akan ada pesan yang telah didekripsi dengan kunci yang dimasukkan. Pengguna dapat mencoba proses dekripsi dengan kunci yang lain apabila hasil dekripsi tidak menunjukkan hasil yang dapat terbaca.

Untuk melihat proses dekripsi pesan, dapat dilihat pada lampiran Fitur: **Mail Detail (MD)** dalam Gambar F.23. sampai Gambar F.25.

d. Uji Pemberian Tanda Tangan Digital

Pada aplikasi ini, proses pemberian tanda tangan digital dapat dilakukan pada halaman Send Mail. Proses ini dilakukan dengan memanggil fungsi *sign* dengan parameter *private_key* dan *message* yang diimplementasikan pada kode Python 3 dari dalam aplikasi Android menggunakan *third party* bernama Chaquopy.

Saat berada di halaman tersebut, pengguna dapat memilih untuk menambahkan tanda tangan digital pada pesan yang akan dikirimkan atau tidak menggunakan sebuah *toggle switch*. Apabila toggle berada pada posisi on, pengguna dapat menekan “Add Digital Sign” untuk menampilkan sebuah *bottomsheet*. Di dalam *bottomsheet* tersebut, disediakan sebuah *text input* untuk pengguna memasukkan kunci privat yang akan dipakai dalam proses penambahan tanda tangan digital. Aplikasi ini juga mengimplementasikan sebuah snack bar untuk menampilkan informasi apabila kunci berhasil ditambahkan, diganti, atau dihapus. Dalam *bottomsheet* tersebut, pengguna juga dapat berpindah ke halaman Key Generator untuk meng-copy kunci yang telah dibangkitkan. Setelah pesan tersebut terkirim, kita dapat melihat pada folder Inbox / Sent bahwa pada bagian bawah isi pesan tersebut telah ditambahkan sebuah tanda tangan digital dengan tag <ds>.

Untuk melihat proses penambahan tanda tangan digital, dapat dilihat pada lampiran Fitur: **Send Mail (SM)** dalam Gambar F.16. sampai Gambar F.17.

e. Uji Verifikasi Tanda Tangan Digital

Pada aplikasi ini, proses proses verifikasi tanda tangan digital dapat dilakukan pada halaman Mail Detail. Proses ini dilakukan dengan memanggil fungsi *verify* dengan parameter *public_key*, *message*, *r*, dan *s* yang diimplementasikan pada kode Python 3 dari dalam aplikasi Android menggunakan *third party* bernama Chaquopy.

Saat berada di halaman tersebut, pengguna dapat memilih untuk memverifikasi tanda tangan digital pada pesan yang telah kita kirim / terima atau tidak menggunakan sebuah *toggle switch*.

Apabila toggle berada pada posisi on, pengguna dapat menekan “Verify Digital Sign” untuk menampilkan sebuah *bottomsheet*. Di dalam *bottomsheet* tersebut, disediakan sebuah *text input* untuk pengguna memasukkan kunci publik yang akan dipakai dalam proses verifikasi tanda tangan digital. Dalam *bottomsheet* tersebut, pengguna juga dapat berpindah ke halaman Key Generator untuk meng-copy kunci yang telah dibangkitkan. Saat pengguna melakukan submit kunci yang dimasukkan, akan muncul sebuah snack bar yang menampilkan informasi apakah pesan tersebut terverifikasi atau tidak. Pengguna dapat mencoba proses verifikasi tanda tangan digital pada pesan dengan kunci yang lain apabila hasil proses verifikasi menunjukkan bahwa pesan tidak terverifikasi.

Untuk melihat proses verifikasi tanda tangan digital, dapat dilihat pada lampiran Fitur: **Mail Detail (MD)** dalam Gambar F.26. sampai Gambar F.28.

f. Uji Enkripsi dan Pemberian Tanda Tangan Digital

Pada aplikasi ini, proses enkripsi dan proses pemberian tanda tangan digital dapat dilakukan pada halaman Send Mail. Tahapan-tahapannya sama seperti yang telah dilakukan pada poin (b) Uji Enkripsi Pesan dan poin (d) Uji Pemberian Tanda Tangan Digital. Hasil dari proses ini adalah setelah pesan terkirim, kita dapat melihat pada folder Inbox / Sent bahwa isi pesan tersebut telah terenkripsi dan pada bagian bawah pesan terenkripsi tersebut telah ditambahkan sebuah tanda tangan digital dengan tag <ds>.

Untuk melihat proses enkripsi disertai penambahan tanda tangan digital, dapat dilihat pada lampiran Fitur: **Send Mail (SM)** dalam Gambar F.18.

g. Uji Dekripsi dan Verifikasi Tanda Tangan Digital (Otentikasi & Integritas)

Pada aplikasi ini, proses dekripsi dan proses verifikasi tanda tangan digital dapat dilakukan pada halaman Mail Detail. Tahapan-tahapannya sama seperti yang telah dilakukan pada poin (c) Uji Dekripsi Pesan dan poin (e) Uji Verifikasi Tanda Tangan Digital. Namun, proses dekripsi harus dilakukan terlebih dahulu sebelum melakukan verifikasi tanda tangan digital karena pesan yang dijadikan sebagai referensi untuk proses verifikasi tanda tangan digital adalah pesan hasil dekripsi.

Pada kasus ini, kita dapat menguji otentikasi dan integritas dengan kasus mengubah karakter di dalam isi email dan pasangan kunci privat dan kunci publik yang digunakan tidak berpadanan. Kasus pertama dapat kita lakukan dalam aplikasi dengan cara mendekripsi pesan dengan kunci yang salah sebelum melakukan verifikasi tanda tangan digital, sehingga karakter yang terdapat pada isi email berubah atau tidak sesuai dengan isi pesan asli. Kasus kedua dapat kita lakukan dengan memasukkan kunci publik yang tidak berpadanan dengan kunci privat yang dibangkitkan.

Untuk melihat proses dekripsi disertai verifikasi tanda tangan digital, dapat dilihat pada lampiran Fitur: **Mail Detail (MD)** dalam Gambar F.29. sampai Gambar F.32.

h. Uji Kerahasiaan Surel

Aplikasi ini tidak dapat diuji kerahasiaan surel-nya menggunakan wireshark untuk memantau data traffic karena aplikasi ini hanya menyimulasikan proses pengiriman email dengan cara menyimpan pesan ke dalam Room *database* pada Android. Aplikasi ini bahkan tidak membutuhkan koneksi internet.

E. Kesimpulan dan Saran

a. Kesimpulan

Pada tugas ini, algoritma fungsi hash SHA3 (Keccak) dan algoritma ECDSA berhasil diimplementasikan menggunakan bahasa Python 3. Antarmuka aplikasi juga berhasil diimplementasikan dengan baik menggunakan *toolkit* Jetpack Compose dengan bahasa Kotlin. Kedua program dengan bahasa yang berbeda tersebut berhasil penulis hubungkan menggunakan *third party* bernama Chaquopy.

Aplikasi ini tidak mendukung fitur kirim email yang sesungguhnya menggunakan koneksi internet karena keterbatasan penulis dalam mengimplementasikan Gmail API dan juga kurangnya pengetahuan terkait *open source email client* untuk mengerjakan tugas ini. Namun, aplikasi ini berhasil mengimplementasikan proses enkripsi dan tanda tangan digital pada aplikasi yang dibangun dengan menyimulasikan proses pengiriman email dengan cara menyimpan pesan ke dalam Room *database* pada Android.

b. Saran

Sebagai mahasiswa yang mengikuti ujian komprehensif, tugas ini sangat tidak direkomendasikan untuk dikerjakan seorang diri. Riset mengenai *email client*, pembuatan algoritma kriptografi dari awal, penambahan fitur pada *existing email client* atau pembuatan antarmuka aplikasi mobile dari awal (kasus penulis), implementasi backend dan server, itu semua membutuhkan setidaknya dua orang untuk mengerjakan tugas ini dalam waktu yang terbatas.

Mengenai spesifikasi tugas ini, langkah baiknya tugas ini dapat dikerjakan pada *platform* apapun, baik itu aplikasi *mobile* maupun aplikasi web. Hal itu dapat mempermudah mahasiswa dalam mengembangkan aplikasi yang diinginkan berdasarkan keahlian masing-masing. Selain itu, proses integrasi antara algoritma kriptografi yang akan dibangun (Keccak & ECDSA) serta yang telah ada (*block cipher*) dengan aplikasi dapat dilakukan dengan lebih mudah menyesuaikan keahlian masing-masing mahasiswa dalam menggunakan *framework* aplikasi pada *platform* tertentu.

F. Lampiran

a. Repository

Android-app: <https://github.com/bintangfrnz/Gmayl>

b. Aplikasi (.apk)

Android-app: <https://github.com/bintangfrnz/Gmayl>

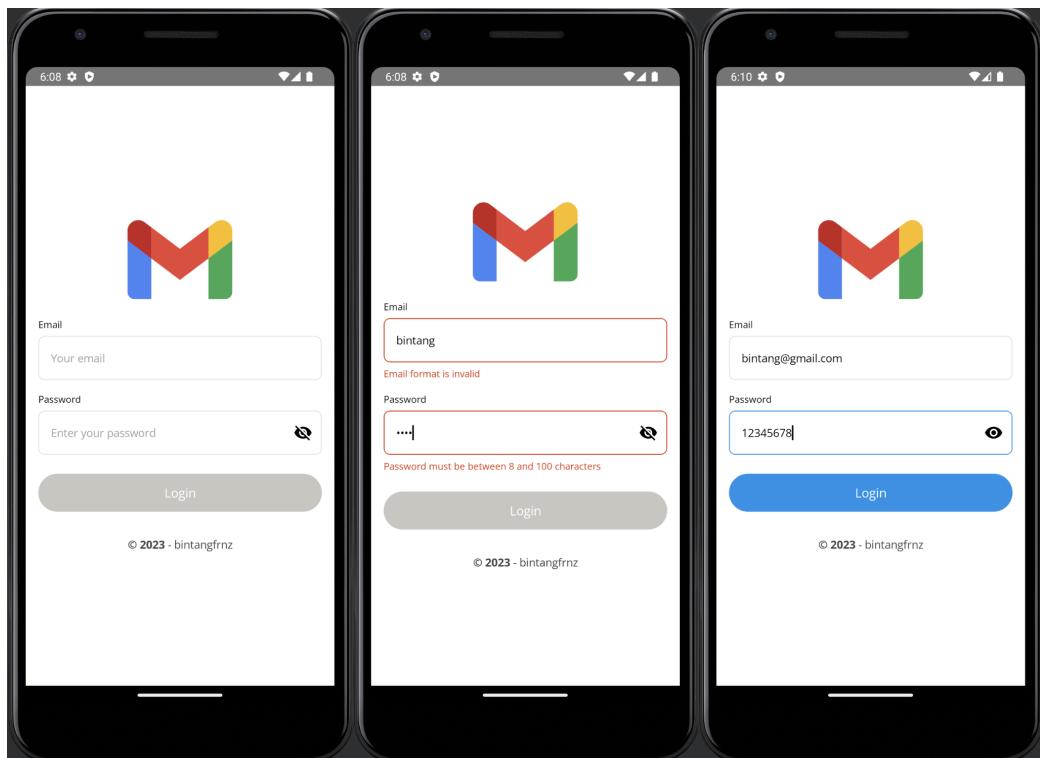
c. Tampilan Layar Aplikasi

1) Splash Screen

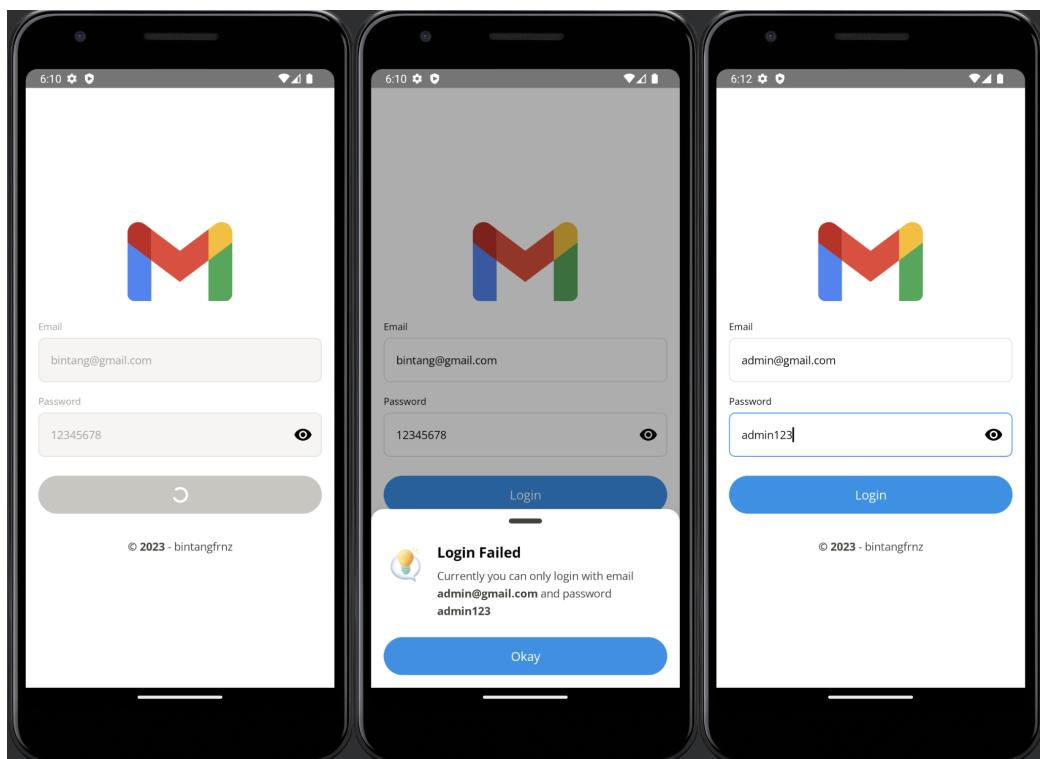


Gambar F.1. Splash Screen

2) Fitur: Login

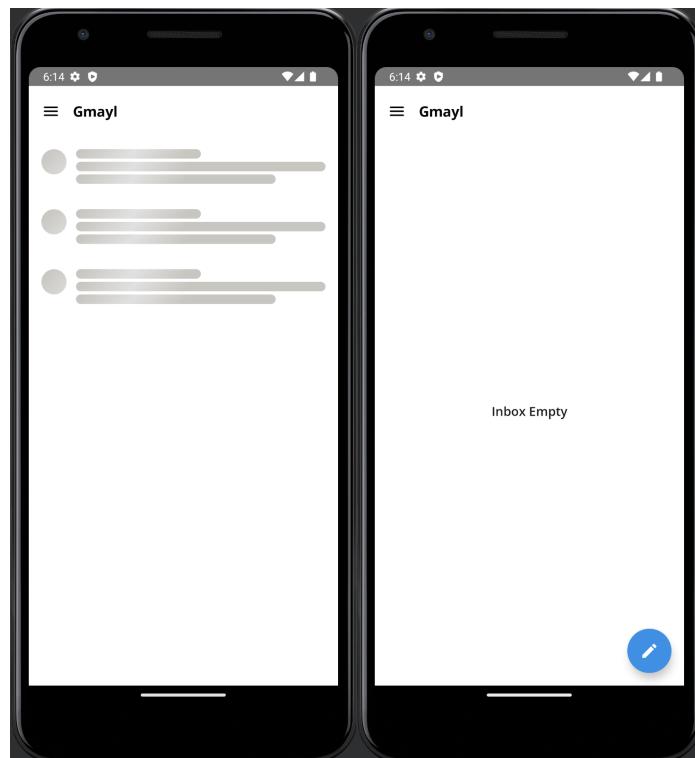


Gambar F.2. Login - Validator

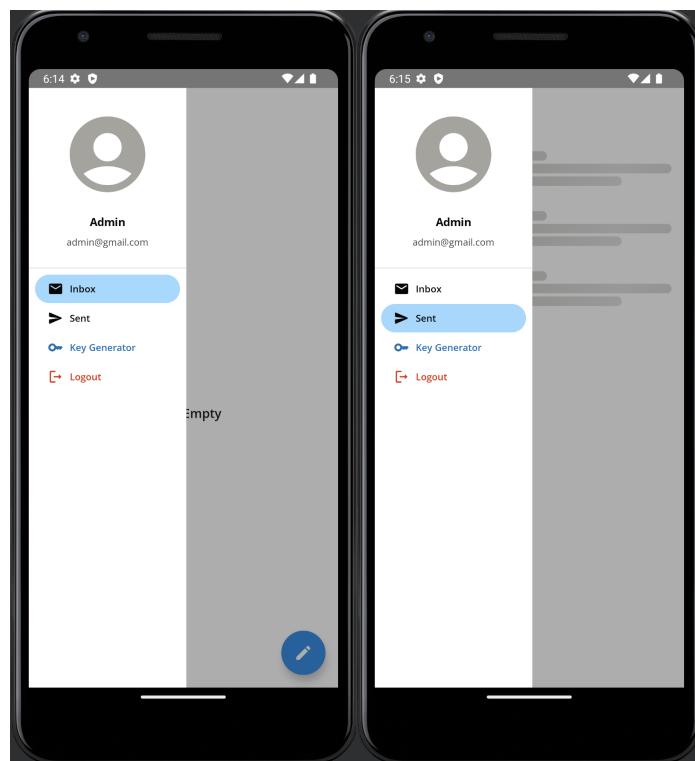


Gambar F.3. Login - Result

3) Fitur: Home

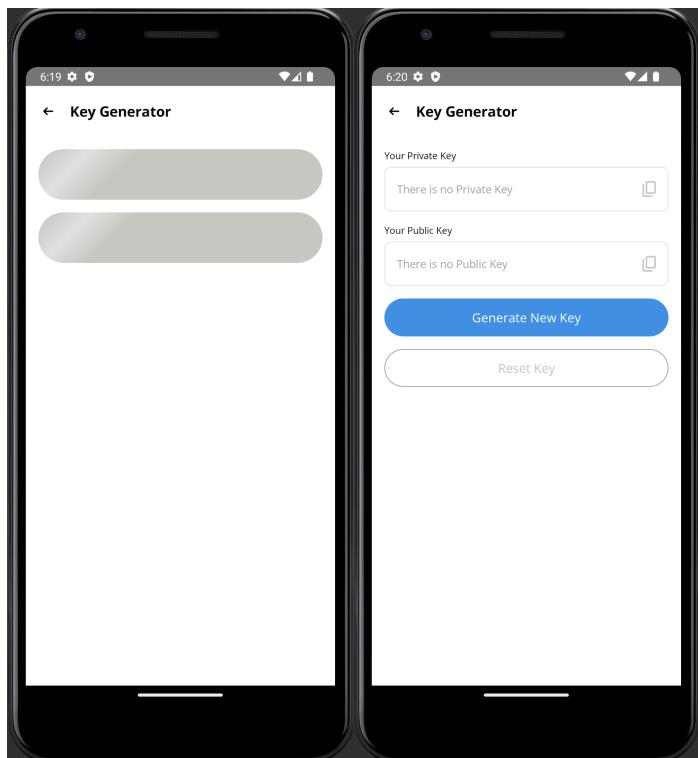


Gambar F.4. Home: Default

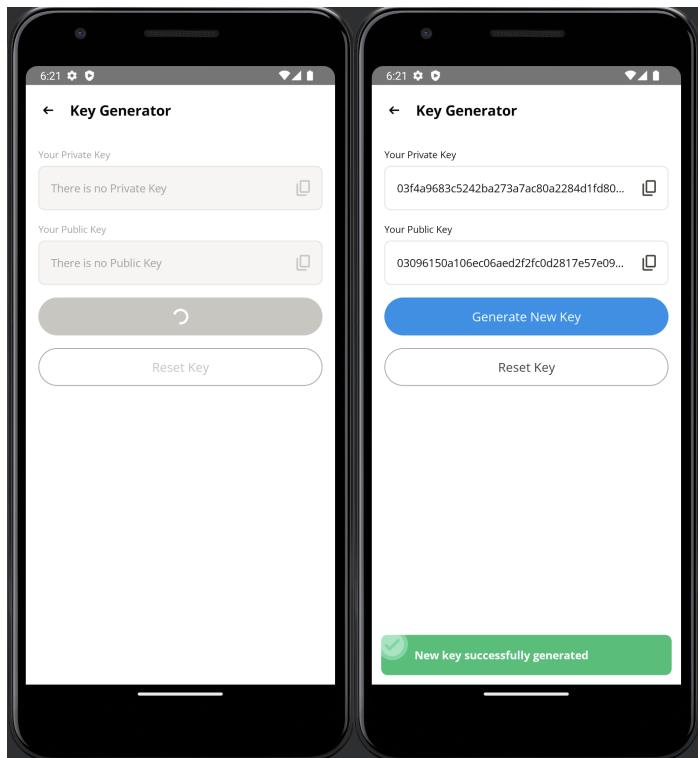


Gambar F.5. Home - Drawer

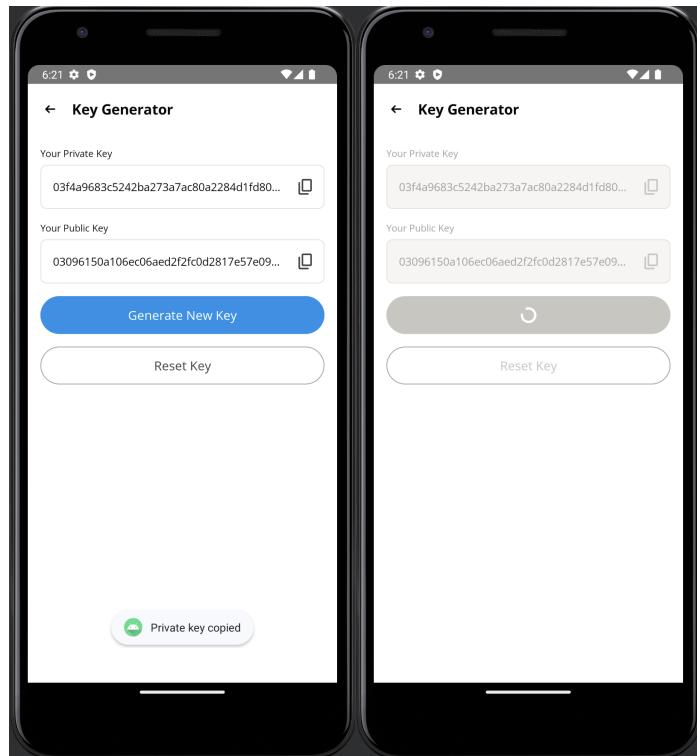
4) Fitur: Key Generator (KG)



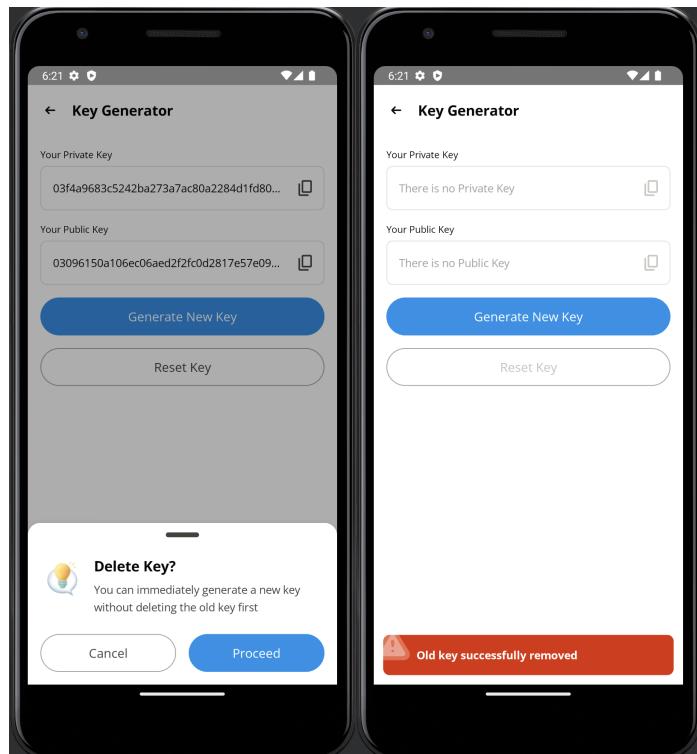
Gambar F.6. KG: Default



Gambar F.7. KG: Generate Key

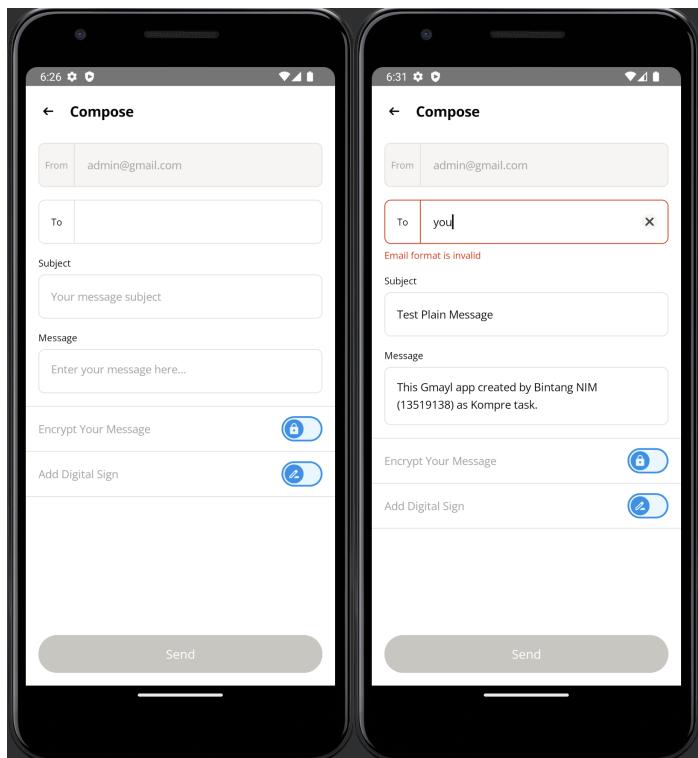


Gambar F.8. KG: Copy Key & Regenerate Key

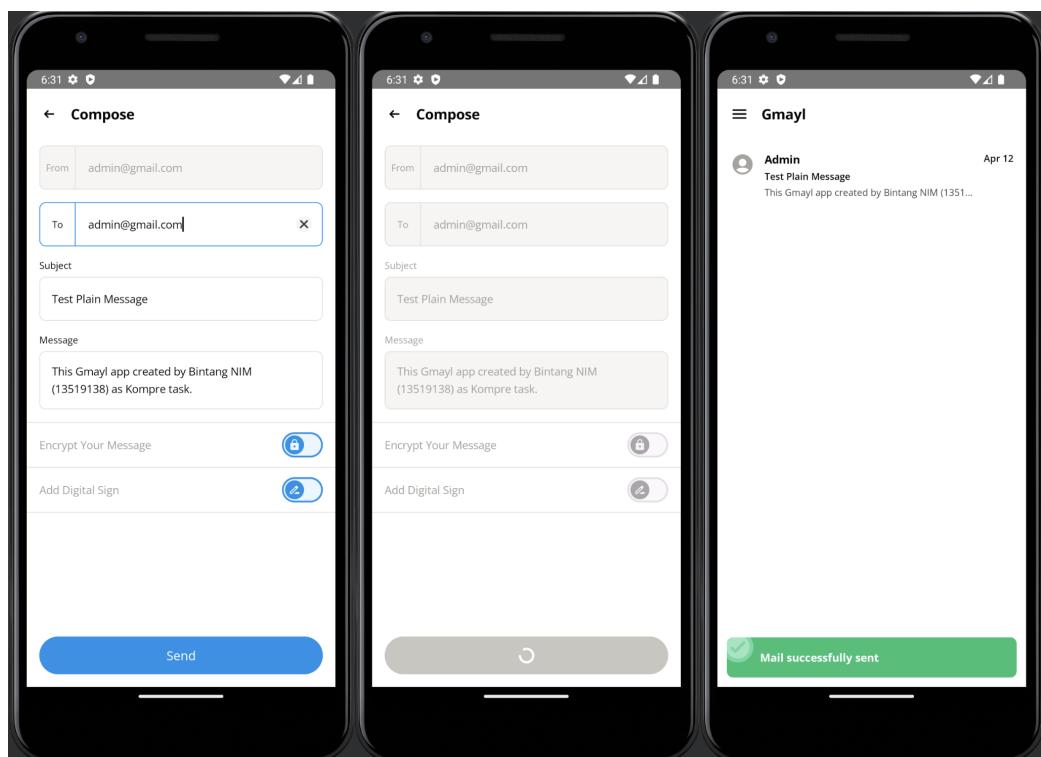


Gambar F.9. KG: Delete Key

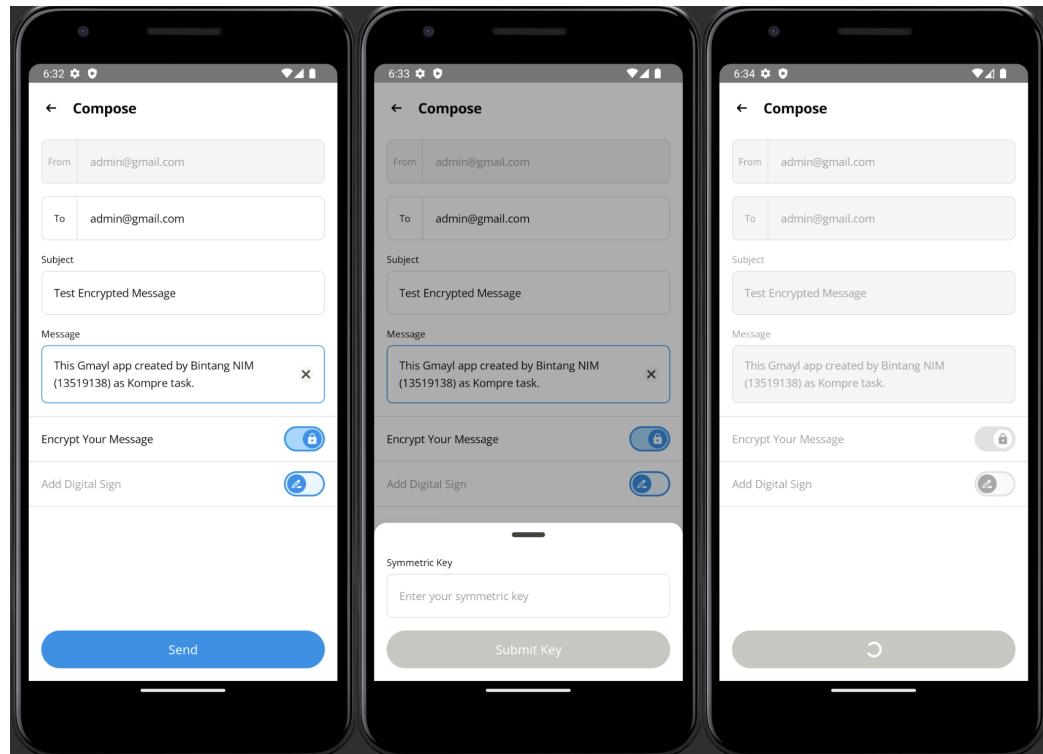
5) Fitur: Send Mail (SM)



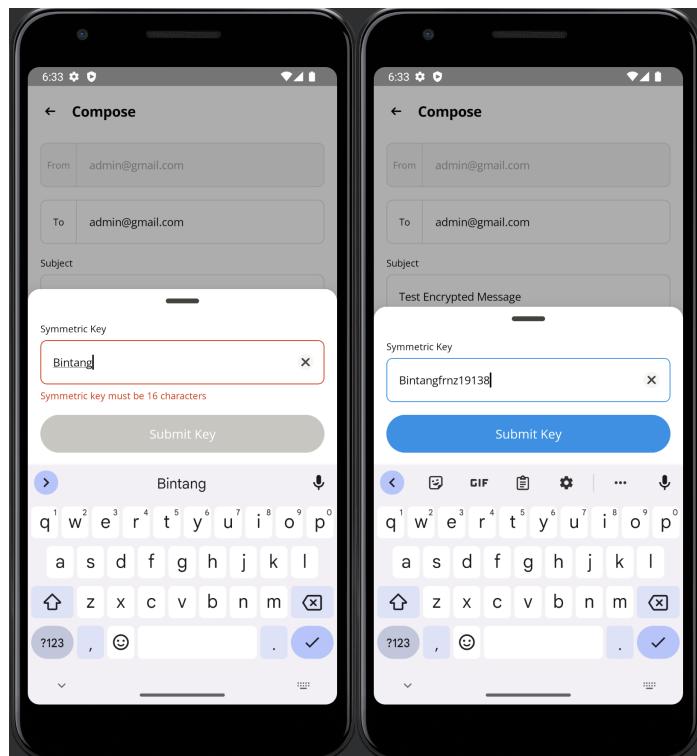
Gambar F.10. SM: Plain - Default (1)



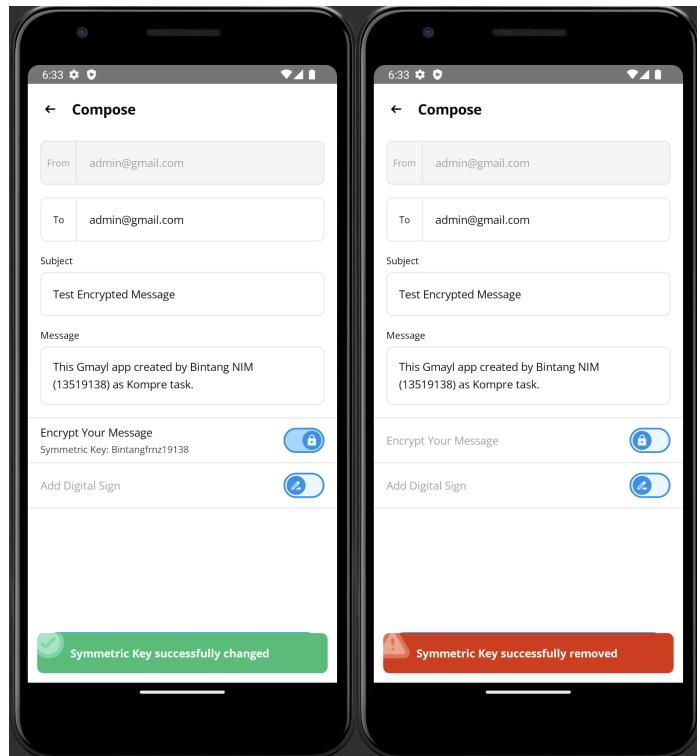
Gambar F.11. SM: Plain - Success (2)



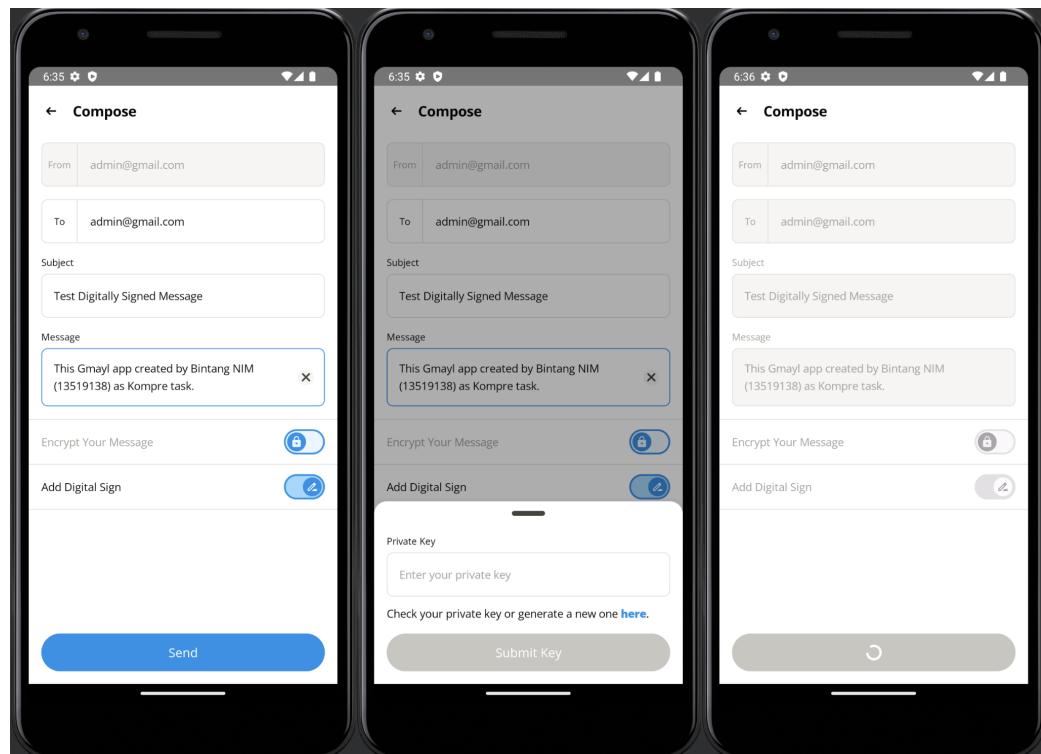
Gambar F.13. SM: Encrypt - Default (1)



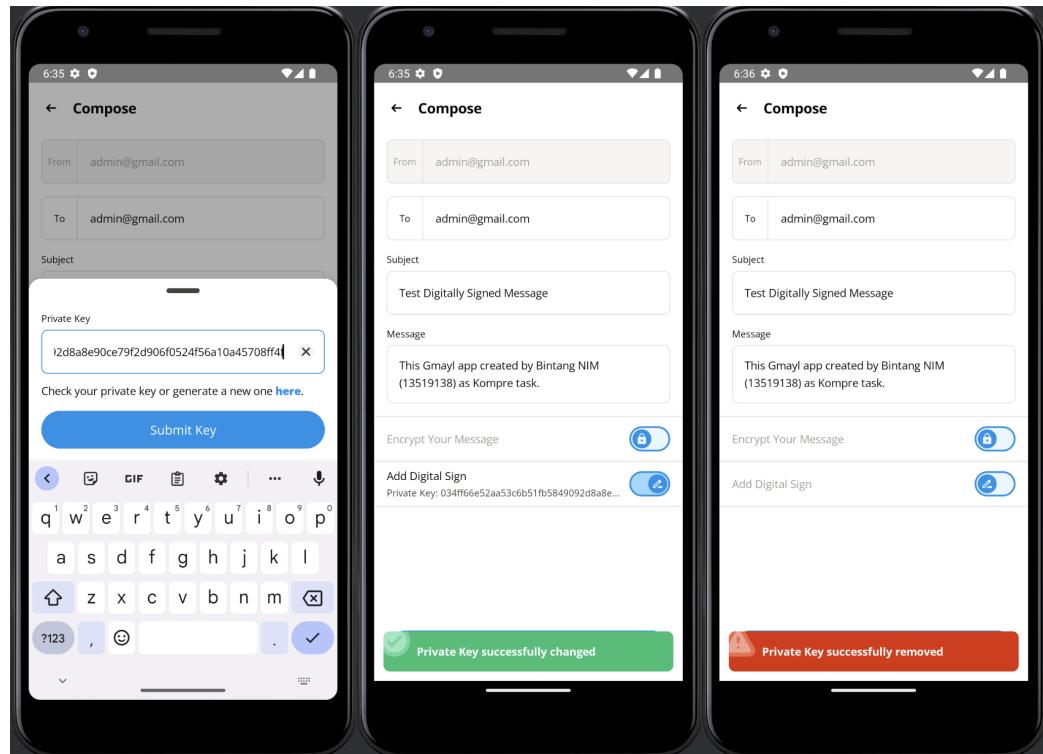
Gambar F.14. SM: Encrypt - Validator (2)



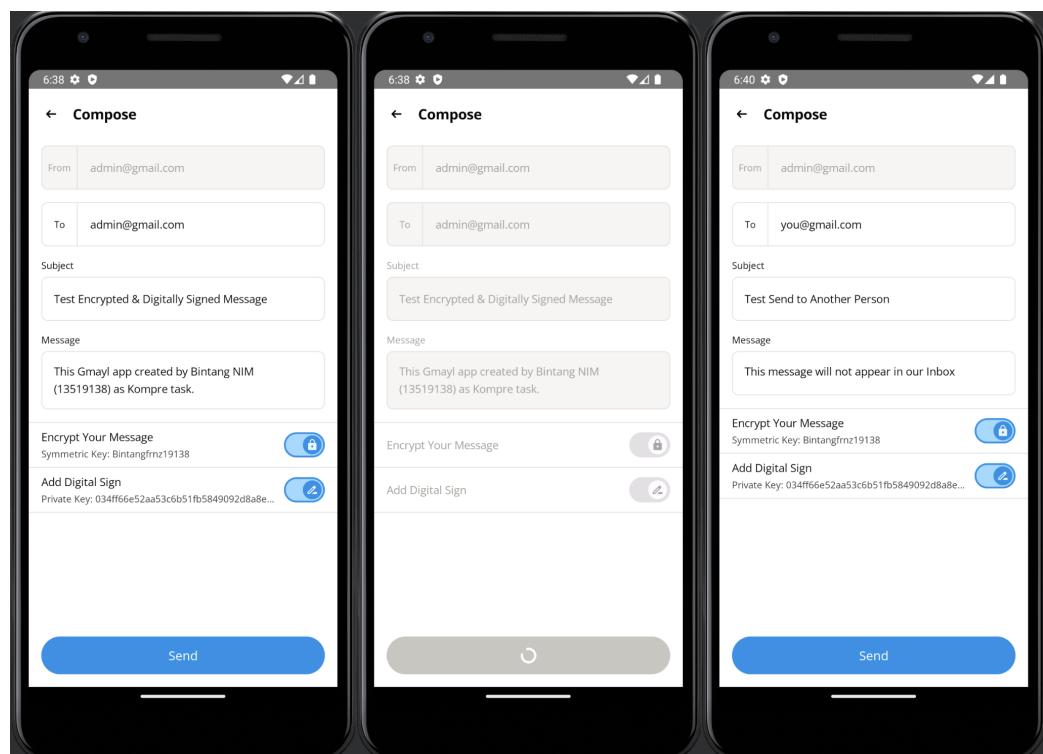
Gambar F.15. SM: Encrypt - Result (3)



Gambar F.16. SM: Digitally Sign - Default (1)

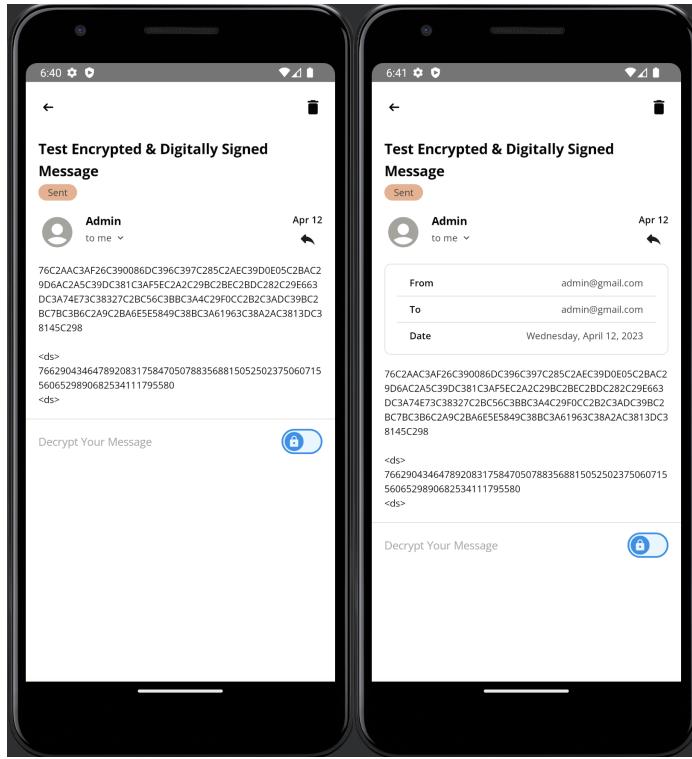


Gambar F.17. SM: Digitally Sign - Result (2)

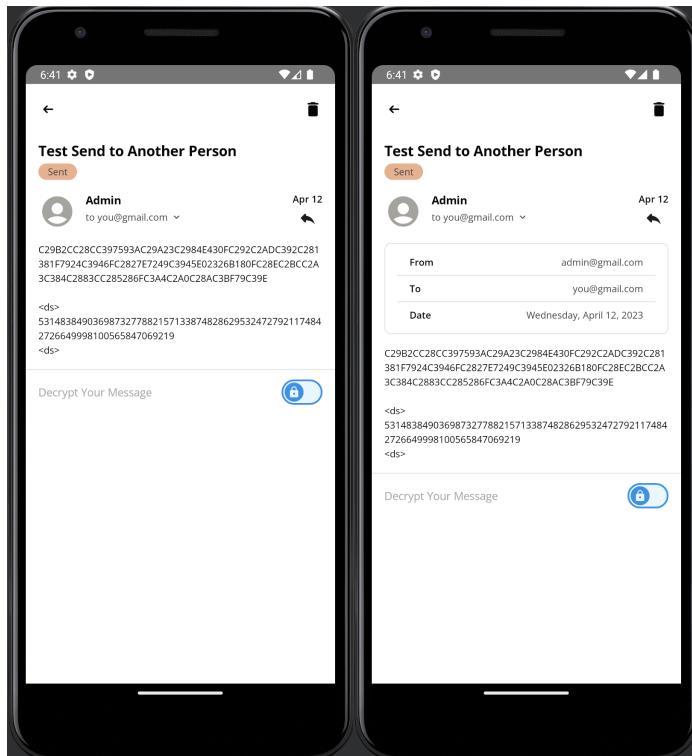


Gambar F.18. SM: Encrypt & Digitally Sign

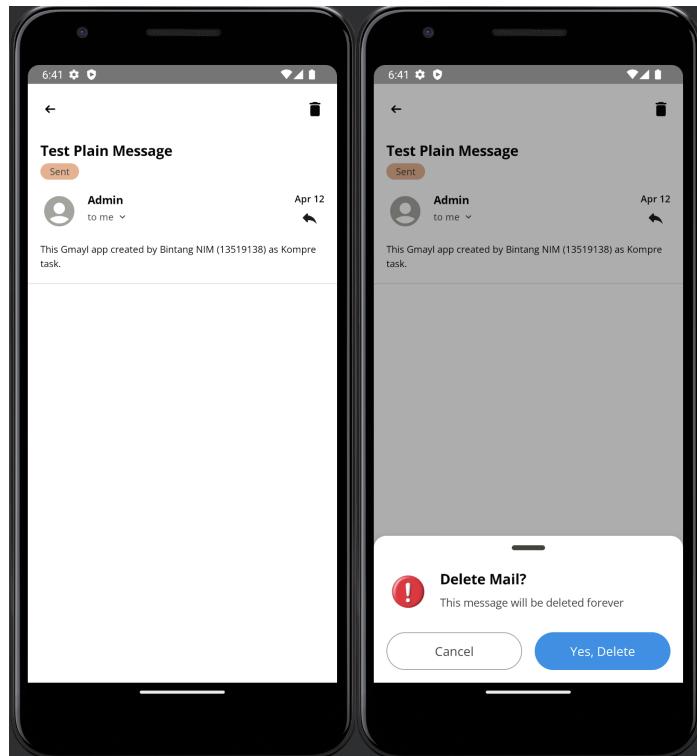
6) Fitur: Mail Detail (MD)



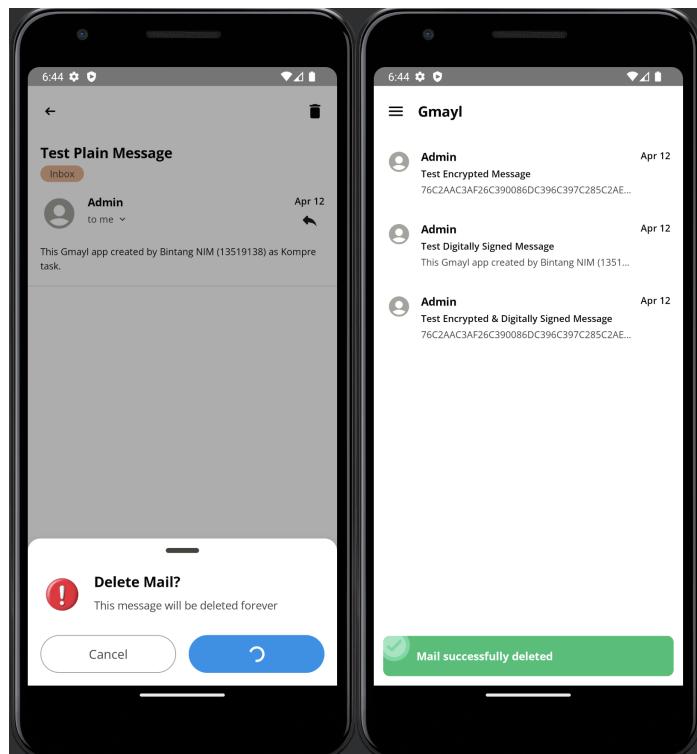
Gambar F.19. MD: Inbox Mail



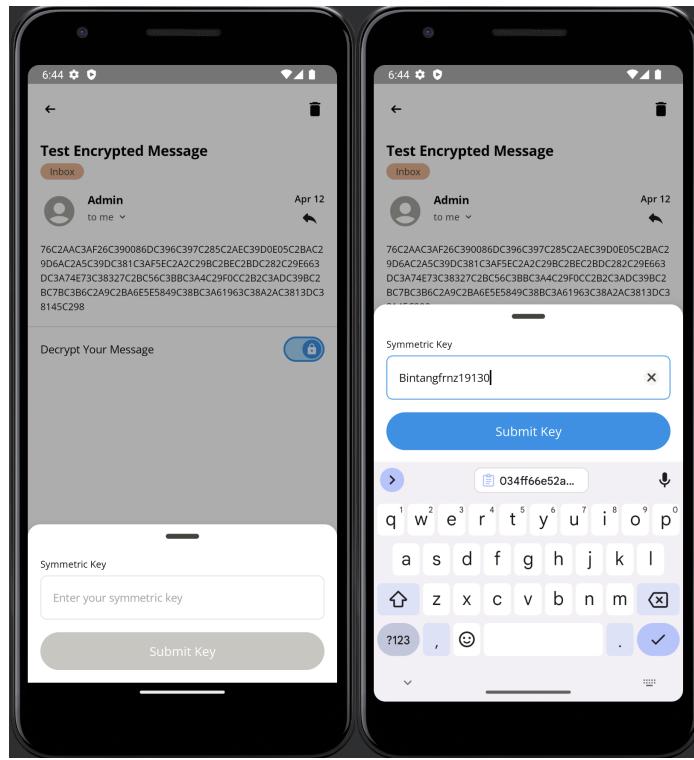
Gambar F.20. MD: Sent Mail



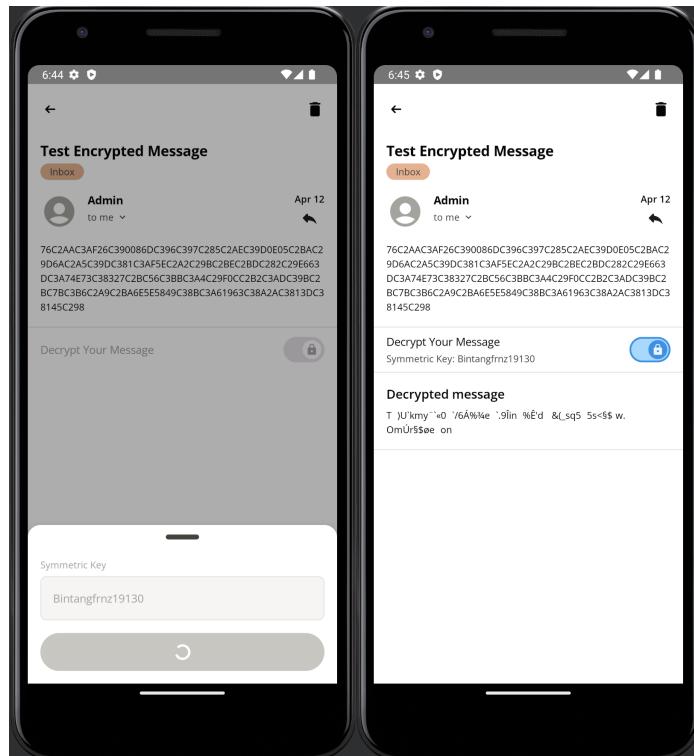
Gambar F.21. MD: Delete Mail - Default (1)



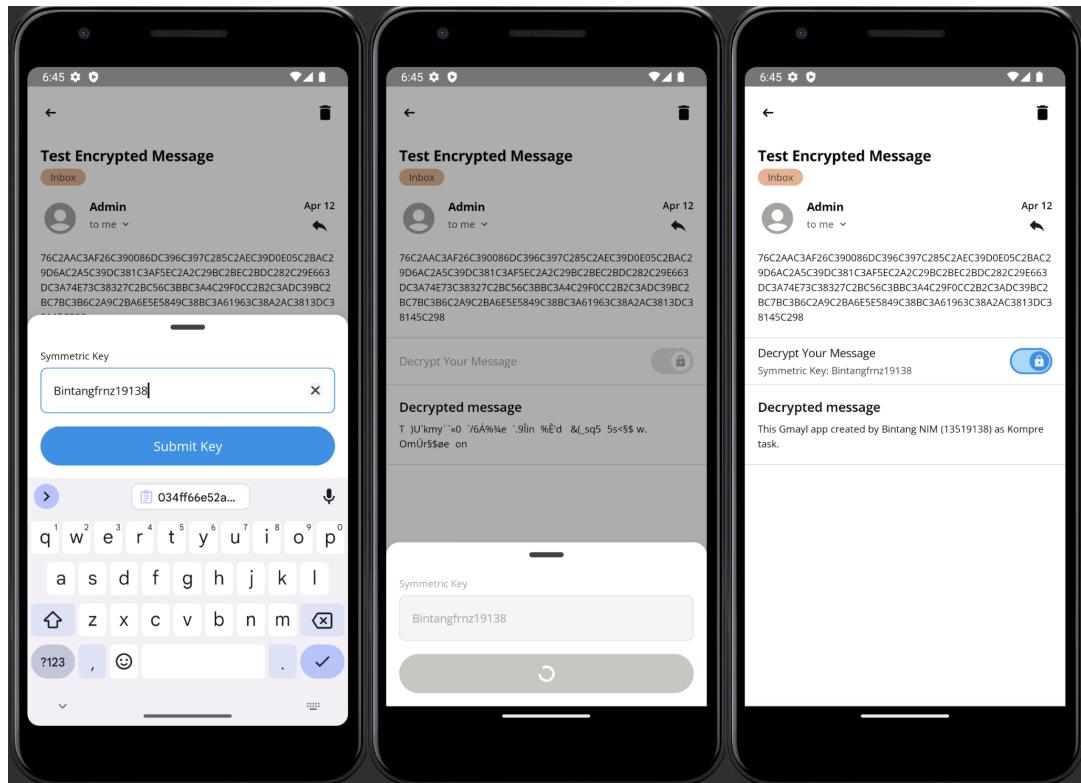
Gambar F.22. MD: Delete Mail - Success (2)



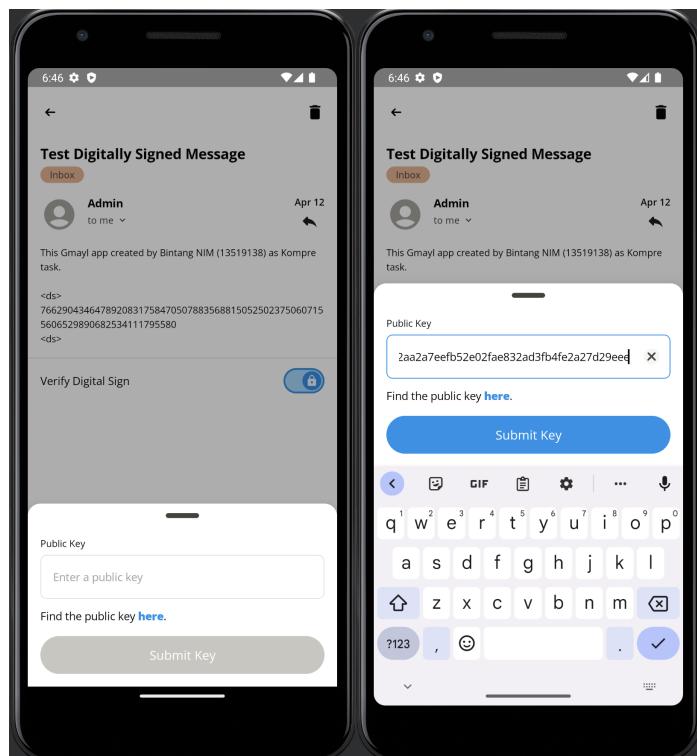
Gambar F.23. MD: Encrypted - Default (1)



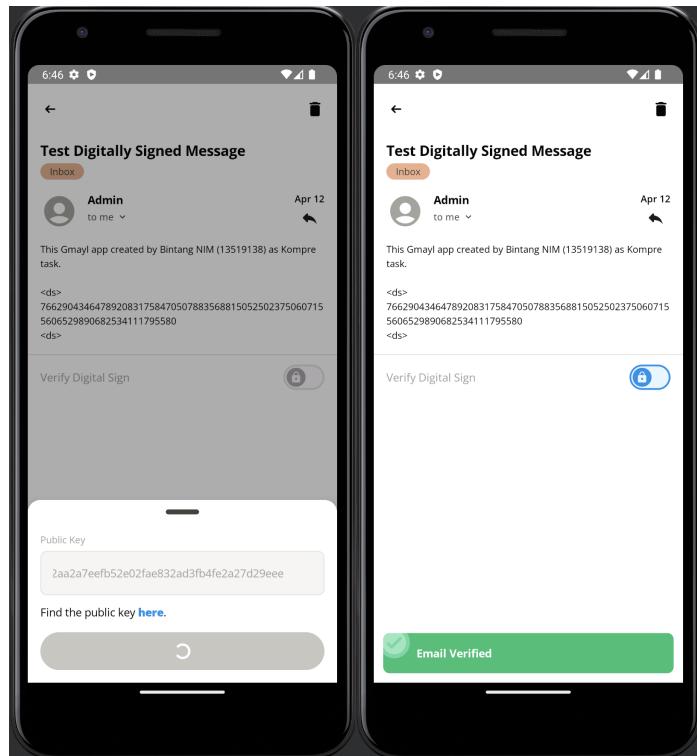
Gambar F.24. MD: Encrypted - Wrong Decryption (2)



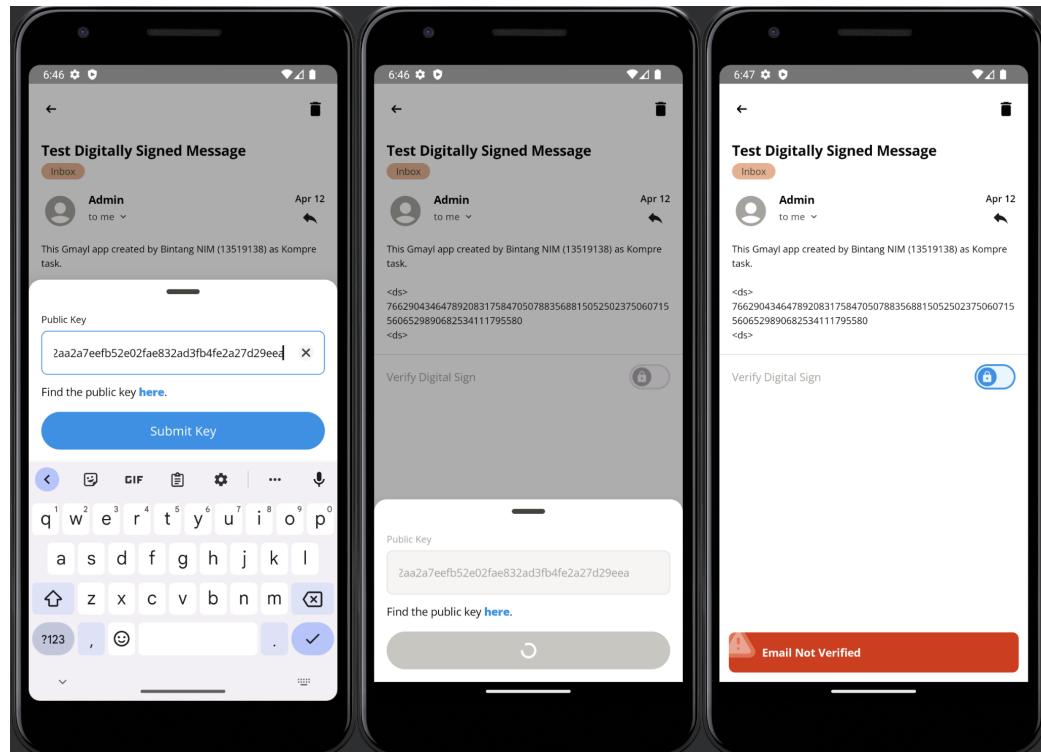
Gambar F.25. MD: Encrypted - Correct Decryption (3)



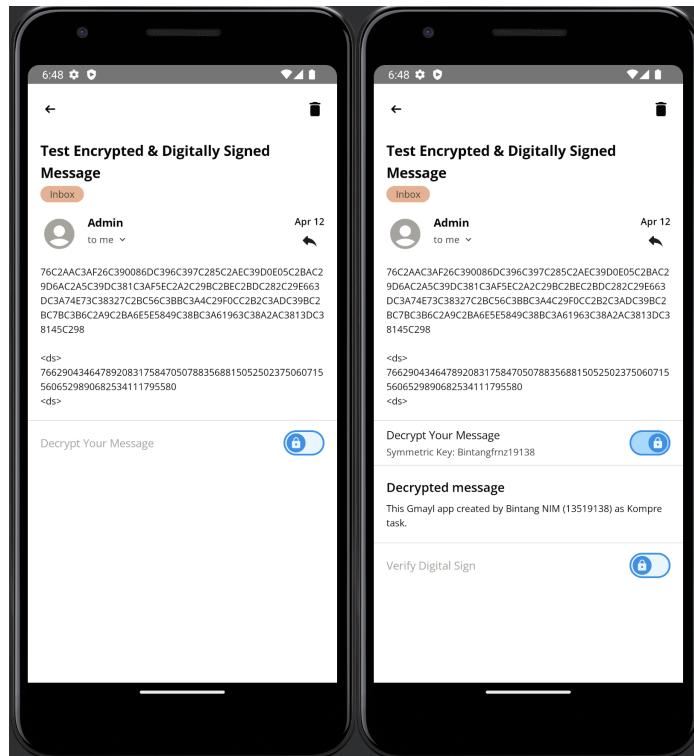
Gambar F.26. MD: Digitally Signed - Default (1)



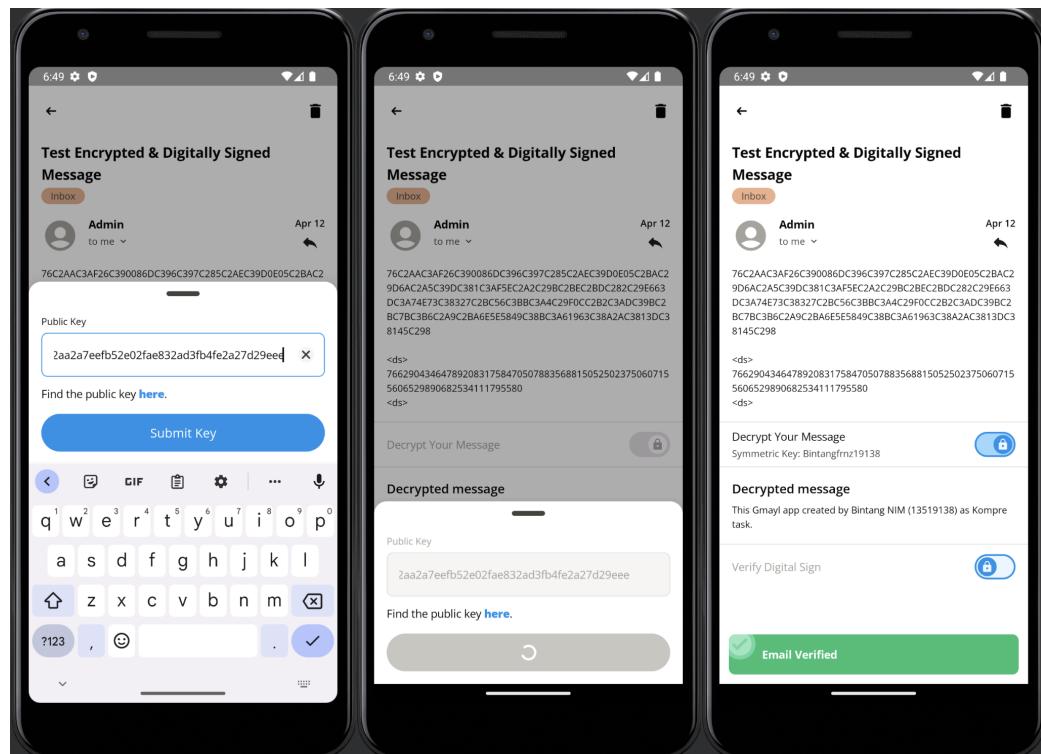
Gambar F.27. MD: Digitally Signed - Verified (2)



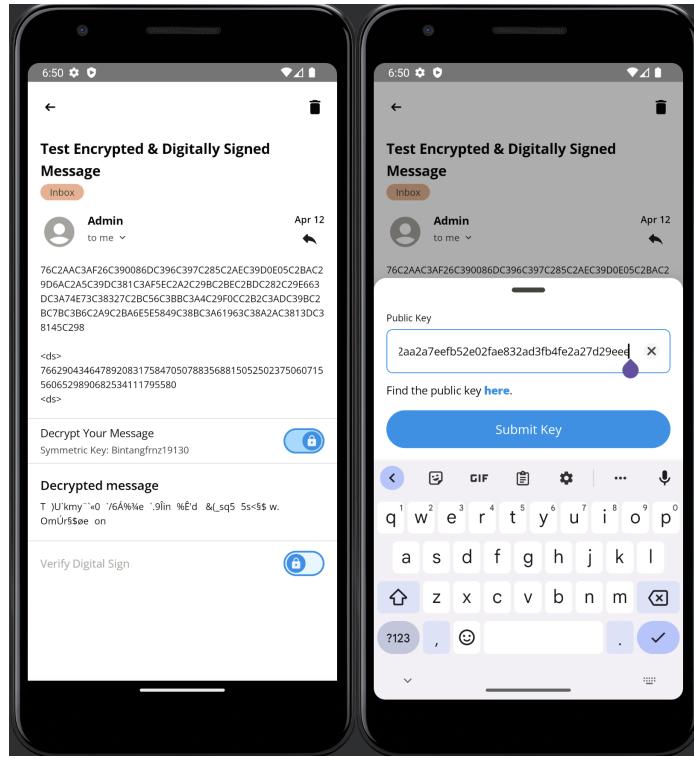
Gambar F.28. MD: Digitally Signed - Unverified (3)



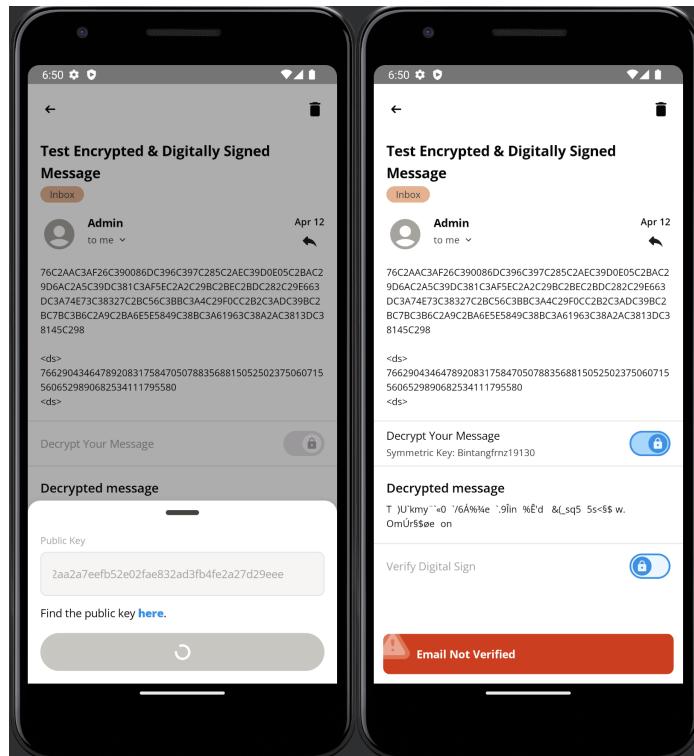
Gambar F.29. MD: Encrypted & Digitally Signed - Correct Decryption (1)



Gambar F.30. MD: Encrypted & Digitally Signed - Verified (2)



Gambar F.31. MD: Encrypted & Digitally Signed - Wrong Decryption (3)



Gambar F.32. MD: Encrypted & Digitally Signed - Unverified (4)