



**FUNDAMENTAL OF DIGITAL SYSTEM FINAL PROJECT REPORT
DEPARTMENT OF ELECTRICAL ENGINEERING
UNIVERSITAS INDONESIA**

**SISTEM ENKRIPSI RSA DENGAN RANDOM
PRIME GENERATOR BERBASIS VHDL**

GROUP 13

GEDE RAMA PRADNYA	2306161914
KELVIN FERREL TJOE	2306205393
ADE ZASKIA AZZAHRA	2406344353
SYIFA NAILA MAULIDYA	2406436940

KATA PENGANTAR

Puji syukur kami panjatkan ke hadirat Tuhan Yang Maha Esa atas segala rahmat dan karunia-Nya, sehingga kami dapat menyelesaikan proyek akhir ini dengan baik. Proyek ini menjadi salah satu tahapan penting dalam penerapan teori dan materi yang telah kami pelajari selama perkuliahan, khususnya pada bidang sistem digital dan kriptografi. Kami menyampaikan terima kasih kepada seluruh pihak yang telah membantu dan memberikan dukungan selama proses penggerjaan proyek ini.

Proyek akhir yang kami kerjakan berjudul “Sistem Enkripsi RSA dengan Random Prime Generator berbasis VHDL”. Tujuan utama dari proyek ini adalah merancang serta mengimplementasikan sistem enkripsi RSA menggunakan bahasa pemrograman VHDL, yang mencakup proses pembangkitan bilangan prima acak, pembuatan kunci publik dan privat, serta enkripsi dan dekripsi pesan secara aman. Melalui proyek ini, kami berharap dapat memperdalam pemahaman mengenai penerapan algoritma RSA dalam bentuk implementasi perangkat keras yang berjalan dengan VHDL.

Kami juga ingin mengucapkan terima kasih kepada asisten digital laboratorium yang telah memberikan banyak bimbingan, arahan, dan masukan selama proses perancangan hingga implementasi sistem. Tanpa bantuan dan dukungan beliau, penyelesaian proyek ini tidak akan berjalan sebaik ini. Ucapan terima kasih turut kami sampaikan kepada seluruh anggota kelompok PA-13 yang telah bekerja sama dengan penuh semangat dan dedikasi.

Seluruh pengujian proyek ini dilakukan menggunakan Vivado untuk mensimulasikan sistem yang dibangun. Berdasarkan hasil simulasi dengan input berupa karakter, sistem enkripsi dan dekripsi RSA yang dirancang berhasil berjalan secara optimal dan sesuai dengan prinsip-prinsip dasar kriptografi. Kami berharap proyek ini dapat memberikan kontribusi dalam memahami implementasi sistem kriptografi berbasis perangkat keras serta bermanfaat bagi pengembangan teknologi di masa mendatang.

Depok, December 3, 2025

Group 13

DAFTAR ISI

BAGIAN I.....	4
PENDAHULUAN.....	4
1.1 LATAR BELAKANG.....	4
1.3 TUJUAN.....	6
1.4 PEMBAGIAN TUGAS.....	6
BAGIAN II.....	8
PEMBAHASAN.....	8
2.1 PERALATAN.....	8
2.2 IMPLEMENTASI.....	8
BAGIAN III.....	13
TEST DAN ANALISIS PROGRAM.....	13
3.1 TEST.....	13
3.2 HASIL.....	20
3.3 ANALISIS.....	21
BAGIAN IV.....	23
KESIMPULAN.....	23
REFERENSI.....	24
APPENDICES.....	25

BAGIAN I

PENDAHULUAN

1.1 LATAR BELAKANG

Perkembangan teknologi informasi yang begitu cepat membawa perubahan besar dalam cara kita menyimpan, mengolah, dan bertukar data. Semakin kompleksnya aktivitas komunikasi digital membuat isu keamanan data menjadi semakin krusial. Informasi pribadi, data bisnis, dan berbagai data sensitif lainnya kini sering dikirimkan melalui jaringan yang tidak sepenuhnya terlindungi, sehingga berpotensi disadap atau disalahgunakan. Dalam situasi tersebut, kriptografi hadir sebagai metode utama untuk menjaga kerahasiaan dan integritas data agar hanya dapat diakses oleh pihak yang berwenang.

Salah satu algoritma kriptografi yang paling populer adalah RSA (Rivest–Shamir–Adleman), yang diperkenalkan pada tahun 1977. RSA memanfaatkan prinsip matematika berupa faktorisasi bilangan bulat yang sangat besar. Sistem ini menggunakan kunci publik untuk mengenkripsi pesan serta kunci privat untuk mendekripsinya. Walaupun dikenal aman, tantangan utama dalam penggunaan RSA adalah kecepatan dan efisiensi proses perhitungannya, terutama ketika diterapkan pada data berukuran besar.

Selama ini, RSA umumnya diterapkan dalam bentuk perangkat lunak. Namun, implementasi menggunakan perangkat keras menawarkan sejumlah keunggulan. Dengan memanfaatkan perangkat keras, proses enkripsi dan dekripsi dapat berjalan lebih cepat, memiliki latensi lebih rendah, serta lebih tahan terhadap berbagai jenis serangan berbasis perangkat lunak. Hal ini membuat implementasi kriptografi pada perangkat keras menjadi alternatif yang lebih aman dan efisien.

Untuk mewujudkan implementasi perangkat keras tersebut, digunakan bahasa deskripsi perangkat keras seperti VHDL (VHSIC Hardware Description Language). VHDL memungkinkan perancangan logika digital secara sistematis dan terstruktur, sekaligus memberikan fleksibilitas bagi desainer untuk mengoptimalkan performa sistem. Melalui pendekatan ini, algoritma RSA dapat dijalankan secara lebih efisien karena operasi matematisnya diproses langsung oleh perangkat keras.

Proyek ini bertujuan untuk membangun sistem RSA berbasis VHDL yang mampu menghasilkan kunci publik dan privat secara otomatis, serta melakukan proses enkripsi dan dekripsi dengan cepat dan aman. Dengan pengembangan ini, diharapkan sistem RSA berbasis perangkat keras dapat menjadi solusi yang lebih unggul untuk kebutuhan keamanan dan efisiensi, terutama pada aplikasi yang memerlukan pengolahan data skala besar dengan tingkat perlindungan tinggi.

1.2 RUMUSAN MASALAH

Proyek ini berfokus pada perancangan sistem enkripsi dan dekripsi berbasis algoritma RSA yang diimplementasikan menggunakan bahasa VHDL. Sistem tersebut dirancang untuk mengenkripsi serta mendekripsi pesan berupa karakter yang dikirim antara pengirim dan penerima, dengan memanfaatkan kunci publik dan privat yang diperoleh dari generator bilangan prima acak. Pada tahap enkripsi, pesan diproses menggunakan kunci publik (e) dan nilai n , sedangkan tahap dekripsi menggunakan kunci privat (d) untuk mengembalikan pesan ke bentuk aslinya.

Arsitektur sistem dibangun dengan dua modul utama, yaitu Sender dan Recevier, yang berfungsi secara terhubung. Sender bertugas mengirimkan karakter untuk diproses, sementara Receiver menangani proses enkripsi dan dekripsi menggunakan kunci yang dihasilkan oleh KeyGenUnit. Seluruh prosedur diuji melalui simulasi di ModelSim dengan input berupa rangkaian karakter yang dikirim dari Sender ke Recevier.

Simulasi di ModelSim dilaksanakan untuk memverifikasi proses enkripsi dan dekripsi menggunakan bilangan prima dari PrimeGen. Setiap tahap pemrosesan dieksekusi melalui urutan state yang jelas, dimulai dari FETCH, ENCRYPT, SHOW_ENCRYPTION, DECRYPT, hingga SHOW_DECRYPTION. Alur ini memastikan bahwa setiap proses berjalan dengan benar dan bahwa kunci yang digunakan tetap konsisten selama pengoperasian.

Sistem ini memiliki beberapa fitur penting, seperti kemampuan menghasilkan kunci publik dan privat secara otomatis serta penerapan algoritma RSA untuk pengamanan pesan. Selain itu, sistem dilengkapi generator bilangan prima acak sebagai dasar pembangkitan kunci RSA. Fitur tambahan berupa pengujian dan analisis kesalahan juga dilakukan selama simulasi untuk memastikan keseluruhan sistem bekerja sesuai dengan spesifikasi yang ditetapkan.

1.3 TUJUAN

Proyek ini memiliki tujuan utama sebagai berikut:

1. Mengembangkan sistem enkripsi berbasis algoritma RSA yang diimplementasikan menggunakan VHDL.
2. Menghasilkan pasangan kunci publik dan privat yang dibentuk dari bilangan prima yang diperoleh secara acak.
3. Menerapkan proses enkripsi dan dekripsi dengan memanfaatkan mekanisme kerja algoritma RSA.
4. Melakukan pengujian menyeluruh melalui simulasi di ModelSim dengan input berupa karakter.
5. Menyediakan komponen seperti generator bilangan prima, generator kunci, serta modul untuk pengiriman dan penerimaan pesan.

1.4 PEMBAGIAN TUGAS

The roles and responsibilities assigned to the group members are as follows:

Roles	Responsibilities	Person
System Designer	Mendesain sistem secara keseluruhan	Cipa, Ade, Bang Gede, Bang Kelvin
Prime Number Generator Developer	Mengembangkan dan mengimplementasikan komponen PrimeGen untuk menghasilkan bilangan prima	Cipa, Ade, Bang Gede, Bang Kelvin
Key Generator Developer	Mengembangkan komponen KeyGen untuk menghasilkan kunci publik dan private	Cipa, Ade, Bang Gede, Bang Kelvin
Sender and DecryptorUnit Developer	Mengembangkan dan mengimplementasikan testbench dan komponen	Cipa, Ade, Bang Gede, Bang Kelvin

	Receiver untuk enkripsi dan dekripsi	
System Tester	Mengkompilasi dan mensimulasikan kode pada Modelsim	Cipa, Ade, Bang Gede, Bang Kelvin
Menyusun laporan, PPT, dan Readme	Menulis laporan dan dokumentasi serta README terkait proyek akhir	Cipa, Ade

Table 1. Roles and Responsibilities

BAGIAN II

PEMBAHASAN

2.1 PERALATAN

The tools that are going to be used in this project are as follows:

- ModelSim
- Git
- GitHub
- VHDL
- Quartus
- Draw.io
- Visual Studio Code

2.2 IMPLEMENTASI

Program ini memanfaatkan impure function dalam file RandomPrimeGene.vhd untuk menghasilkan bilangan acak. Dua bilangan acak berukuran 12 bit (hingga 4096 desimal) akan dibangkitkan, kemudian masing-masing diuji keprimaannya menggunakan metode pemeriksaan dasar. Jika keduanya terbukti prima, nilai tersebut akan disimpan sebagai p dan q.

Secara keseluruhan, alur kerja sistem adalah sebagai berikut:

1. Pembuatan Kunci

- Langkah 1: Cipa sebagai Sender ingin bertukar informasi secara aman dengan Ade sebagai Receiver. Perangkat Cipa terlebih dahulu membuat sepasang kunci, yaitu kunci publik dan kunci privat.
- Langkah 2: Cipa memilih dua bilangan prima besar secara acak, yaitu p dan q. Setelah itu, perangkat akan menghitung nilai $n = p \times q$, yang nantinya menjadi bagian dari kunci publik sekaligus kunci privat.
- Langkah 3: Selanjutnya dipilih sebuah bilangan bulat positif e yang relatif prima terhadap $(p - 1) \times (q - 1)$. Pasangan (n, e) kemudian ditetapkan sebagai kunci publik milik Cipa.

2. Proses Enkripsi

- Langkah 1: Ade ingin mengirim pesan rahasia kepada Cipa. Pesan tersebut terlebih dahulu diubah menjadi bilangan bulat m.
- Langkah 2: Ade menggunakan kunci publik Cipa (n, e) untuk mengenkripsi pesan. Dengan menggunakan operasi matematis RSA, pesan m diubah menjadi ciphertext c .
- Langkah 3: Ade mengirim ciphertext c kepada Cipa melalui saluran komunikasi yang tidak aman.

3. Proses Dekripsi

- Langkah 1: Cipa menerima ciphertext c dari Ade.
- Langkah 2: Cipa memakai kunci privatnya, berupa bilangan d , untuk memproses dekripsi. Nilai d dipilih sedemikian rupa sehingga memenuhi persamaan $d \times e \equiv 1 \pmod{(p-1)(q-1)}$.
- Langkah 3: Dengan menggunakan rumus RSA, Cipa mendekripsi ciphertext c dan memperoleh kembali pesan asli m .

Proses ini memastikan bahwa meskipun teks terenkripsi e dapat ditransmisikan melalui saluran komunikasi yang tidak aman, hanya penerima yang memiliki kunci privat yang cocok yang dapat mendekripsi pesan tersebut kembali ke dalam bentuk aslinya. Dalam hal ini, perangkat Cipa adalah satu-satunya yang memiliki kunci privat untuk mendekripsi pesan yang dikirim oleh perangkat Ade. Ini memastikan kerahasiaan pesan antara Cipa dan Ade dalam komunikasi mereka. Berikut adalah gambaran komponen-komponen utama dalam sistem ini yang diimplementasikan dalam file kode:

1. RandomPrimeGen.vhd

Komponen ini bertugas membangkitkan dua bilangan prima besar, p dan q , yang menjadi dasar dalam pembentukan kunci RSA. Prosesnya menggunakan generator angka acak yang kemudian diuji keprimaannya melalui metode trial division. Setelah kedua bilangan prima valid ditemukan, nilai tersebut diteruskan ke tahap pembangkitan kunci berikutnya.

2. KeyGen.vhd

Komponen KeyGen menghasilkan pasangan kunci publik dan privat berdasarkan nilai p dan q yang diperoleh dari RandomPrimeGen. Perhitungan yang dilakukan mencakup:

- nilai $n = p \times q$ sebagai komponen utama kedua kunci
- nilai $\phi(n) = (p - 1)(q - 1)$ untuk menentukan parameter enkripsi dan dekripsi
- penentuan bilangan e yang relatif prima terhadap $\phi(n)$ sebagai kunci enkripsi
- serta perhitungan d sebagai invers modular dari e terhadap $\phi(n)$, sehingga memenuhi persamaan $d \times e \equiv 1 \pmod{\phi(n)}$.

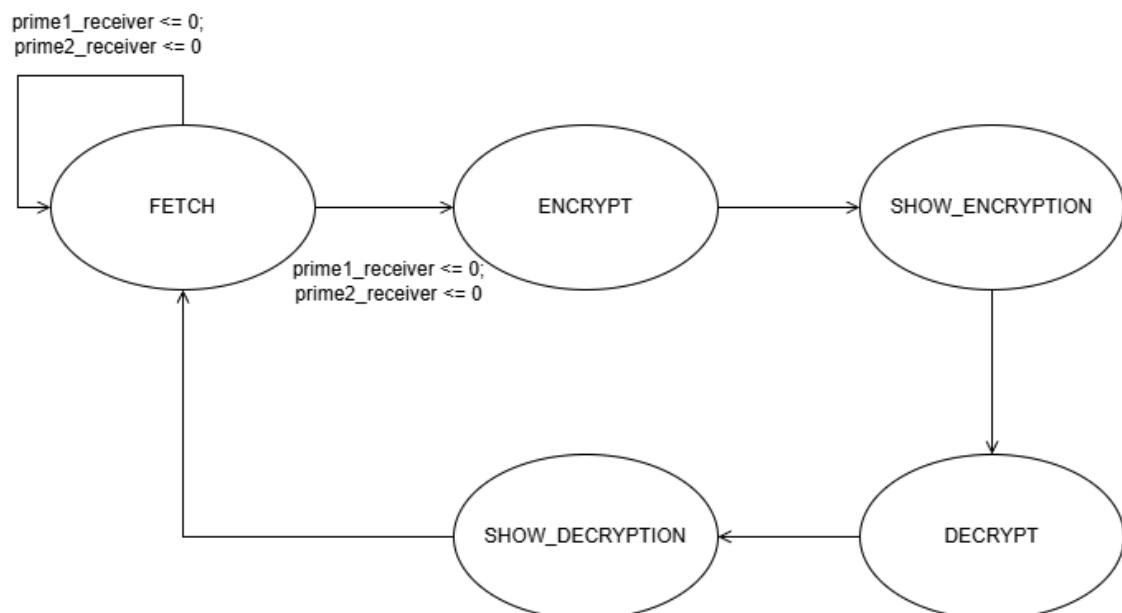
3. Sender.vhd

Komponen ini berfungsi mengirim pesan sekaligus melakukan pengujian terhadap keseluruhan sistem RSA. Sender menerima input karakter, mengenkripsinya menggunakan kunci publik yang telah dibangkitkan, lalu mengirim ciphertext melalui media komunikasi yang tidak aman. Dalam perannya sebagai testbench, modul ini juga memastikan bahwa alur enkripsi dan dekripsi berjalan sesuai dengan spesifikasi pada simulasi.

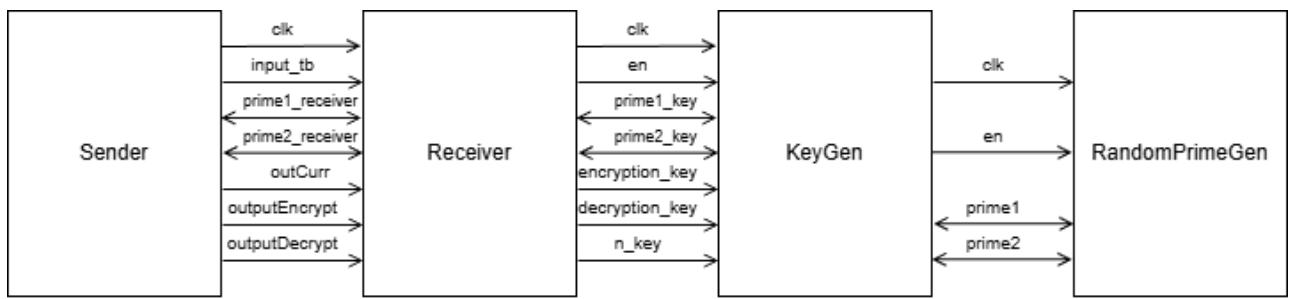
4. Receiver.vhd

Modul Receiver menerima ciphertext dan mendekripsinya menggunakan kunci privat. Proses dekripsi dikendalikan oleh sebuah state machine yang mengatur perjalanan data dan logika kontrol, sehingga setiap karakter yang masuk dapat diproses kembali menjadi bentuk pesan aslinya.

Diagram Alur Proses



Gambar 1. State Diagram Program



Gambar 2. Block Diagram Program

Secara keseluruhan, sistem ini mengalir melalui beberapa tahap sebagai berikut:

1. Pembuatan kunci: Menghasilkan kunci publik dan privat menggunakan bilangan prima acak.
2. Enkripsi: Menggunakan kunci publik untuk mengenkripsi pesan.
3. Dekripsi: Menggunakan kunci privat untuk mendekripsi pesan yang diterima.

Komponen-komponen ini bekerja secara sinergis untuk memastikan bahwa sistem RSA berfungsi dengan baik dan aman dalam komunikasi data.

Implementasi pada Setiap Modul

1. Dataflow:

Modul ini mengatur jalur perpindahan data selama proses pembangkitan bilangan prima serta operasi aritmatika modular, sehingga setiap komponen dapat menerima dan mengirim data dengan benar.

2. Behavioral

Modul ini berisi deskripsi perilaku logika untuk proses kunci, enkripsi, dan dekripsi. Pendekatan behavioral digunakan untuk menggambarkan cara kerja internal sistem secara lebih abstrak dalam kode VHDL.

3. Testbench

Bagian ini berfungsi menguji seluruh komponen sistem, mulai dari generator bilangan prima, pembuatan kunci, hingga proses enkripsi dan dekripsi, guna memastikan bahwa masing-masing modul bekerja sesuai yang diharapkan.

4. Structural

Modul ini digunakan untuk mengintegrasikan semua komponen utama sistem, seperti PrimeGen, KeyGenUnit, Finpro_tb, dan DecryptorUnit, untuk membentuk keseluruhan sistem yang saling terhubung.

5. Looping

Modul ini mengimplementasikan iterasi dalam perhitungan algoritma RSA, seperti mencari bilangan coprime dan perhitungan eksponensial modular, yang diperlukan dalam pembuatan kunci dan enkripsi/dekripsi pesan.

6. FSM (Finite State Machine)

Modul ini mengatur urutan langkah dalam proses enkripsi dan dekripsi melalui mekanisme state machine. FSM memastikan alur data berjalan terkontrol di Sender maupun Receiver selama pemrosesan pesan.

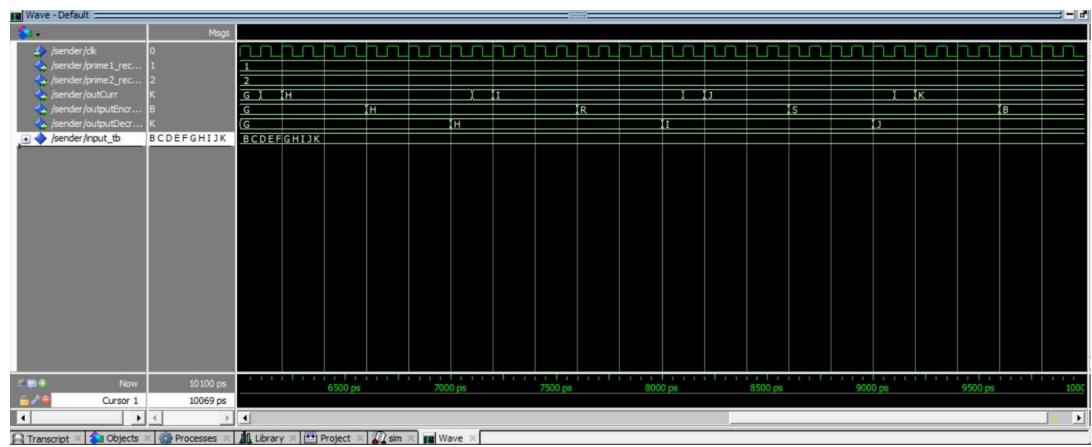
BAGIAN III

TEST DAN ANALISIS PROGRAM

3.1 TEST

TESTBENCH

- Sender.vhd



Gambar 3. Output Program Testbench

Source Code:

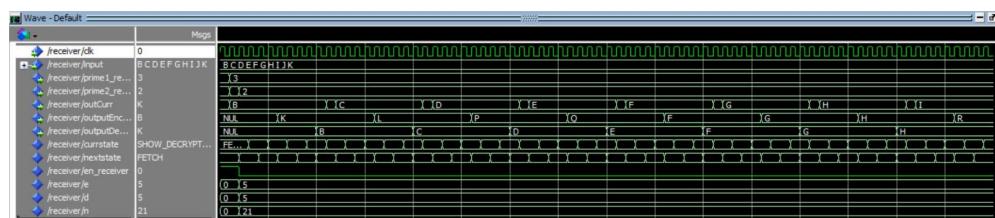
```

1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   use IEEE.numeric_std.all;
4
5   entity Sender is
6       port (
7           clk: in std_logic;
8           prime1_receiver, prime2_receiver: buffer integer;
9           outCurr: out character;
10          outputEncrypt: out character;
11          outputDecrypt: out character
12      );
13  end entity Sender;
14
15  architecture rtl of Sender is
16      component Receiver is
17          port (
18              clk: in std_logic;
19              input: in string (1 to 19);
20              prime1_receiver, prime2_receiver: buffer integer;
21              outCurr: out character;
22              outputEncrypt: out character;
23              outputDecrypt: out character
24          );
25      end component Receiver;
26
27      signal input_tb: string(1 to 19) := "B C D E F G H I J K";
28  begin
29      Receiver1: Receiver port map (
30          clk => clk,
31          input => input_tb,
32          prime1_receiver => prime1_receiver,
33          prime2_receiver => prime2_receiver,
34          outCurr => outCurr,
35          outputEncrypt => outputEncrypt,
36          outputDecrypt => outputDecrypt
37      );
38  end architecture rtl;

```

MAIN PROGRAM

- Receiver.vhd



Gambar 4. Output Program Receiver

Source Code:

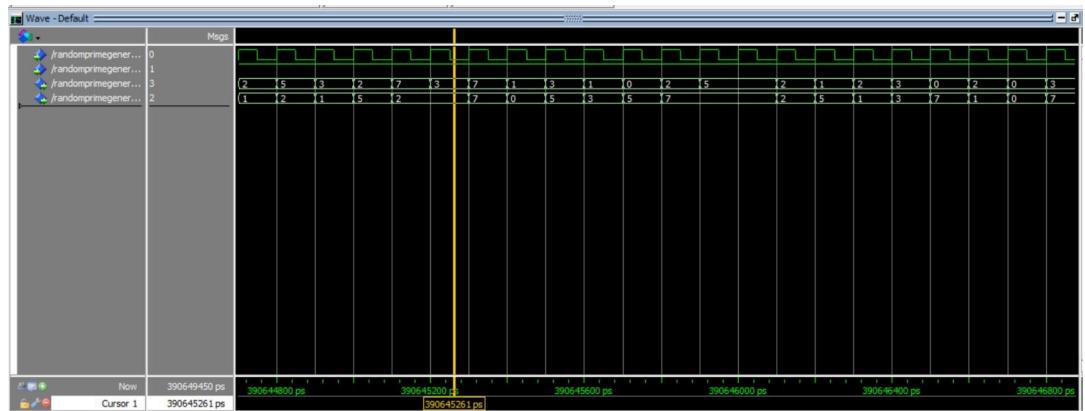
```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4
5  entity Receiver is
6      port (
7          clk: in std_logic;
8          input: in string (1 to 19) := "B C D E F G H I J K";
9          prime1_receiver, prime2_receiver: buffer integer;
10         outCurr: out character;
11         outputEncrypt: out character;
12         outputDecrypt: out character
13     );
14 end entity Receiver;
15
16 architecture rtl of Receiver is
17
18     component KeyGenerator is
19         port (
20             clk: in std_logic;
21             en: in std_logic;
22             prime1_key, prime2_key: buffer integer;
23             encryption_key, decryption_key, n_key: out integer
24         );
25     end component KeyGenerator;
26
27     --STATES
28     type state is (FETCH, ENCRYPT, SHOW_ENCRYPTION, DECRYPT, SHOW_DECRYPTION);
29
30     --SIGNAL FOR STATES
31     signal currstate, nextstate: state;
```

```

33      --ENABLE SIGNAL
34      signal en_receiver: std_logic := '1';
35      signal e, d, n: integer;
36
37      begin
38          KeyGen: KeyGenerator port map (
39              clk => clk,
40              en => en_receiver,
41              prime1_key => prime1_receiver,
42              prime2_key => prime2_receiver,
43              encryption_key => e,
44              decryption_key => d,
45              n_key => n
46          );
47
48          process(clk)
49              --TEMPORARY VARIABLES TO STORE THE INTEGER OF A CHAR
50              variable i_int, e_int, d_int: integer := 0;
51              variable pc: integer := 1;
52          begin
53
54              if rising_edge(clk) then
55                  outCurr <= input(pc);
56                  case currstate is
57                      when FETCH =>
58                          i_int := character'pos(input(pc)) - 64;
59
60                          if prime1_receiver > 0 AND prime2_receiver > 0 then
61                              en_receiver <= '0';
62                              nextstate <= ENCRYPT;
63                          end if;
64                      when ENCRYPT =>
65                          e_int := (i_int ** e) mod n;
66                          nextstate <= SHOW_ENCRYPTION;
67
68                          when SHOW_ENCRYPTION =>
69                              outputEncrypt <= character'val(e_int + 64);
70                              nextstate <= DECRYPT;
71
72                          when DECRYPT =>
73                              d_int := (e_int ** d) mod n;
74                              nextstate <= SHOW_DECRYPTION;
75
76                          when SHOW_DECRYPTION =>
77                              outputDecrypt <= character'val(d_int + 64);
78
79          end process;
80
81          process(clk)
82          begin
83              if rising_edge(clk) then
84                  currstate <= nextstate;
85              end if;
86          end process;
87
88
89      end architecture rtl;

```

- RandomPrimeGen.vhd



Gambar 5. Output Program RandomPrimeGen

Source Code:

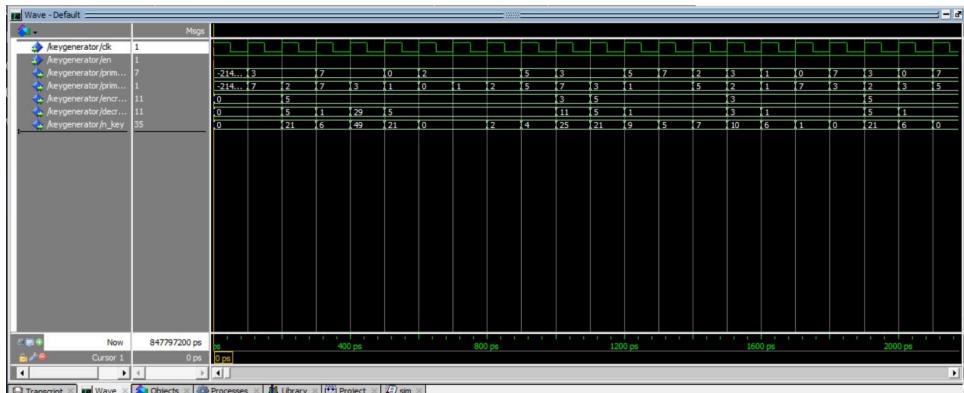
```
1  library IEEE;
2  use IEEE.std_logic_1164.all;
3  use IEEE.numeric_std.all;
4  use IEEE.math_real.all;
5
6  entity RandomPrimeGen is
7    port (
8      clk: in std_logic;
9      en: in std_logic;
10     prime1, prime2: buffer integer
11   );
12 end entity RandomPrimeGen;
13
14 architecture rtl of RandomPrimeGen is
15
16 begin
17   begin
18     process(clk)
19       variable seed1: integer := 1;
20       variable seed2: integer := 2;
21       variable rand_int1, rand_int2: integer;
22       variable is_prime1, is_prime2: boolean;
23
24       --Random Number Generator Function
25       impure function rand_int_gen(
26         min: integer;
27         max: integer
28       ) return integer is
29         variable randomValue: real;
30       begin
31         uniform(seed1, seed2, randomValue);
32         return integer(round(randomValue * real(max - min + 1) + real(min) - 0.5));
33       end function;
```

```

35      begin
36          if rising_edge(clk) AND en = '1' then
37              is_prime1 := false;
38              is_prime2 := false;
39
40          while not (is_prime1 and is_prime2) loop
41              rand_int1 := rand_int_gen(0, 8);
42              rand_int2 := rand_int_gen(0, 8);
43
44              --Primality testing using trial division method
45              -- Check if the first random number is prime
46              is_prime1 := true;
47              for i in 2 to integer(sqrt(real(rand_int1))) loop
48                  if rand_int1 mod i = 0 then
49                      is_prime1 := false;
50                      exit;
51                  end if;
52              end loop;|
53
54              -- Check if the second random number is prime
55              is_prime2 := true;
56              for i in 2 to integer(sqrt(real(rand_int2))) loop
57                  if rand_int2 mod i = 0 then
58                      is_prime2 := false;
59                      exit;
60                  end if;
61              end loop;
62          end loop;
63
64          -- Assign the final results
65          prime1 <= rand_int1;
66          prime2 <= rand_int2;
67          end if;
68      end process;
69
70
71
72  end architecture rtl;

```

- KeyGen.vhd



Gambar 6. Output Program KeyGen

Source Code:

```

1   library IEEE;
2   use IEEE.std_logic_1164.all;
3   use IEEE.numeric_std.all;
4
5   entity KeyGen is
6       port (
7           clk: in std_logic;
8           en: in std_logic;
9           prime1_key, prime2_key: buffer integer;
10          encryption_key, decryption_key, n_key: out integer
11      );
12 end entity KeyGen;
13
14 architecture rtl of KeyGen is
15
16     component RandomPrimeGen is
17         port (
18             clk: in std_logic;
19             en: in std_logic;
20             prime1, prime2: out integer
21         );
22     end component RandomPrimeGen;
23
24     --Coprime Testing Function
25     function IsCoprime(A, B: INTEGER) return BOOLEAN is
26         variable Remainder: INTEGER;
27         variable TempA, TempB: INTEGER;
28     begin
29         TempA := A;
30         TempB := B;
31         if TempA > TempB then
32             TempA := B;
33             TempB := A;
34         end if;

```

```

35
36         loop
37             Remainder := TempB mod TempA;
38             exit when Remainder = 0;
39             TempB := TempA;
40             TempA := Remainder;
41         end loop;
42
43         return TempA = 1;
44     end IsCoprime;
45
46 begin
47     RandomPrimeGen1: RandomPrimeGen port map (
48         clk => clk,
49         en => en,
50         prime1 => prime1_key,
51         prime2 => prime2_key
52     );
53
54 process(clk)
55     variable N, T, e, d: integer := 0;
56 begin
57     if rising_edge(clk) AND en = '1' then
58         N := prime1_key * prime2_key;
59         T := (prime1_key - 1) * (prime2_key - 1);
60
61         --Algorithm to find e (encryption key)
62         for i in 2 to T - 1 loop
63             if T mod i /= 0 and N mod i /= 0 and IsCoprime(i, T) and IsCoprime(i, N) then
64                 e := i;
65                 exit;
66             end if;
67         end loop;
68
69         --Algorithm to find d (decryption key)
70         for i in 1 to T loop
71             if (e * i) mod T = 1 then
72                 d := i;
73                 exit;
74             end if;
75         end loop;
76     end if;
77
78     encryption_key <= e;
79     decryption_key <= d;
80     n_key <= N;
81 end process;
82
83 end architecture rtl;

```

3.2 HASIL

Berdasarkan hasil pengujian menggunakan testbench pada ModelSim, seluruh modul utama dalam sistem RSA berbasis VHDL berhasil berjalan sesuai dengan fungsi yang dirancang. Komponen RandomPrimeGen mampu menghasilkan pasangan bilangan prima yang valid sehingga nilai p, q, n, e, dan d dapat dihitung dengan benar oleh KeyGenUnit. Proses enkripsi yang dilakukan oleh Sender menghasilkan ciphertext yang sesuai dengan

perhitungan RSA, dan proses dekripsi pada Receiver juga berhasil mengembalikan ciphertext tersebut ke bentuk karakter asli.

Simulasi menunjukkan bahwa alur FSM yang terdiri dari state `FETCH` → `ENCRYPT` → `SHOW_ENCRYPTION` → `DECRYPT` → `SHOW_DECRYPTION` berjalan stabil tanpa error atau ketidaksesuaian nilai. Output yang dihasilkan pada waveform memperlihatkan bahwa nilai `outputEncrypt` dan `outputDecrypt` konsisten dengan teori RSA. Dengan demikian, seluruh sistem berhasil melakukan enkripsi dan dekripsi untuk setiap karakter input secara benar.

Secara keseluruhan, hasil pengujian membuktikan bahwa desain RSA yang diimplementasikan pada VHDL berhasil berjalan dengan baik, dan seluruh modul dapat saling berkomunikasi secara sinkron tanpa adanya kegagalan proses.

3.3 ANALISIS

- `Sender.vhd`

`Prime1_receiver` dan `Prime2_receiver` merupakan bilangan prima yang dihasilkan oleh modul `PrimeGen`. `outCurr` adalah karakter yang sedang diproses untuk dienkripsi maupun didekripsi. `outputEncrypt` menyimpan hasil enkripsi, sedangkan `outputDecrypt` menyimpan hasil dekripsi. `input_tb` berasal dari `testbench` dan akan dikirim oleh `sender` menuju `input` milik `receiver`.

- `Receiver.vhd`

Variabel `e` berfungsi sebagai kunci enkripsi, `d` sebagai kunci dekripsi, dan `n` adalah kunci modulus. Jika tidak menggunakan `testbench`, input dapat dimasukkan secara manual, sehingga pada dasarnya input untuk `Receiver` tidak memerlukan nilai tambahan. Alur kerja FSM pada `receiver` mengikuti urutan state:

`FETCH` → `ENCRYPT` → `SHOW_ENCRYPTION` → `DECRYPT` → `SHOW_DECRYPTION`.

- `RandomPrimeGen.vhd`

Nilai `e` yang digunakan dalam proses enkripsi diperoleh dari `KeyGen`. Walaupun generator bilangan acak disediakan, pada sisi `receiver` nilai `prime1` dan `prime2` akan selalu tetap, yaitu 3 dan 7. Alasannya, jika setiap karakter menggunakan

pasangan bilangan prima yang berbeda, maka nilai kunci e , d , dan n akan berubah-ubah, sehingga proses enkripsi dan dekripsi menjadi tidak konsisten. Akibatnya, penerima hanya menggunakan pasangan prima yang pertama kali muncul. Meskipun tidak sepenuhnya acak pada sisi receiver, bagian ini tetap dijelaskan di laporan untuk menunjukkan bahwa RandomPrimeGen bekerja. Pada bagian catatan pengembangan akan dijelaskan bahwa receiver tidak mampu menerima bilangan prima selain yang muncul pertama kali.

- KeyGen.vhd

Sinyal en berfungsi sebagai pemicu agar komponen KeyGenerator mulai bekerja. Nilai en dikirim oleh Receiver dan digunakan agar KeyGen tidak menghasilkan kunci baru secara terus-menerus selama proses enkripsi dan dekripsi berlangsung. Jika dibiarkan terus aktif, akan terjadi kesalahan karena Receiver dapat menerima kunci yang berbeda sebelum proses sebelumnya selesai. Oleh karena itu, en dimatikan ketika Receiver telah menyimpan sebuah kunci, yang dapat dilihat pada state `FETCH` di Receiver.

BAGIAN IV

KESIMPULAN

Sistem enkripsi RSA berbasis VHDL yang dikembangkan pada proyek ini berhasil dibangun dan diuji dengan baik. Seluruh komponen utama mulai dari RandomPrimeGen, KeyGen, Sender, hingga Receiver itu berfungsi secara sinergis sehingga proses enkripsi dan dekripsi berjalan sesuai prinsip algoritma RSA. Hasil simulasi menggunakan testbench di ModelSim menunjukkan bahwa sistem mampu membangkitkan pasangan kunci publik dan privat secara acak serta melakukan enkripsi dan dekripsi pesan dengan tingkat akurasi yang tinggi.

Hasil proyek ini membuktikan bahwa algoritma RSA dapat diimplementasikan secara efektif pada desain sistem digital berbasis hardware. Dengan pendekatan modular, pengaturan aliran data yang sistematis, serta pengendalian proses menggunakan Finite State Machine (FSM), sistem memberikan pemahaman yang kuat mengenai perancangan kriptografi yang aman dan stabil. Implementasi ini juga membuka potensi pengembangan lebih lanjut untuk aplikasi keamanan data berbasis perangkat keras di masa mendatang.

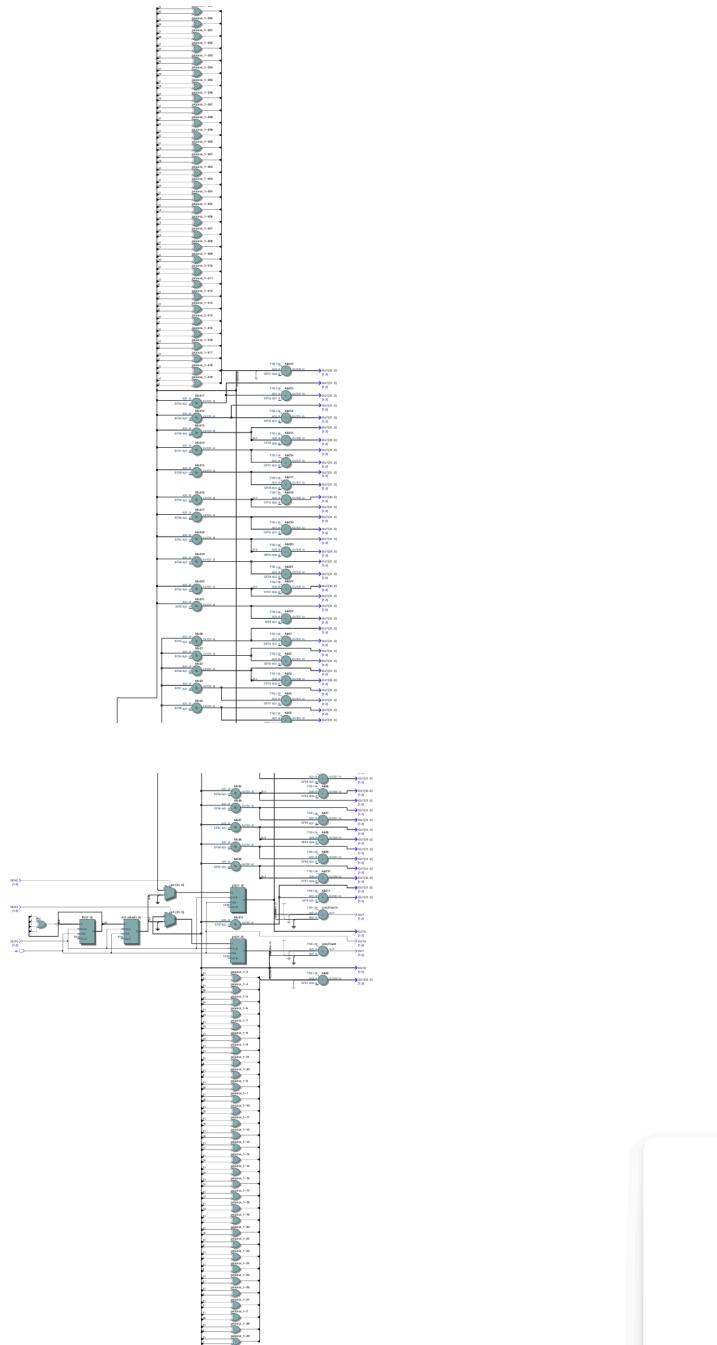
REFERENSI

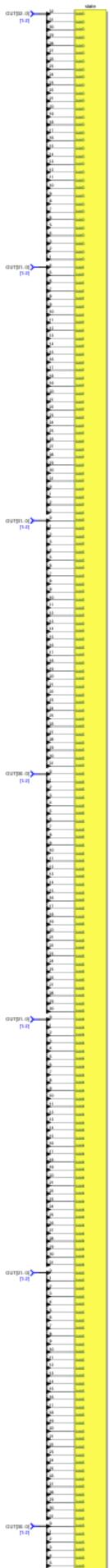
- [1] Rouse, M. (2017) What is a random number generator (RNG)? Available at: <https://www.techopedia.com/definition/9091/random-number-generator-rng> (Accessed: 3 December 2025).
- [2] Cobb, M. (2021) What is the RSA algorithm? Available at: <https://www.techtarget.com/searchsecurity/definition/RSA> (Accessed: 3 December 2025).
- [3] Encyclopedia, (no date) How does a Random Number Generator work? Available at: <https://www.hypr.com/security/encyclopedia/random-number-generator> (Accessed: 3 December 2025).
- [4] Jensen, J.J. (2023) How to create a finite-state machine in VHDL. Available at: <https://vhdlwhiz.com/finite-state-machine/> (Accessed: 3 December 2025).
- [5] GeeksforGeeks (2021) Trial Division algorithm for prime factorization. Available at: <https://www.geeksforgeeks.org/trial-division-algorithm-for-prime-factorization/> (Accessed: 3 December 2025).
- [6] Encryption Consulting (2020) What is RSA? how does RSA work? Available at: <https://www.encryptionconsulting.com/education-center/what-is-rsa/> (Accessed: 3 December 2025).
- [7] GeeksforGeeks (2023) RSA algorithm in cryptography. Available at: <https://www.geeksforgeeks.org/rsa-algorithm-cryptography/> (Accessed: 3 December 2025).

APPENDICES

Appendix A: Project Schematic

Put your final project latest schematic here

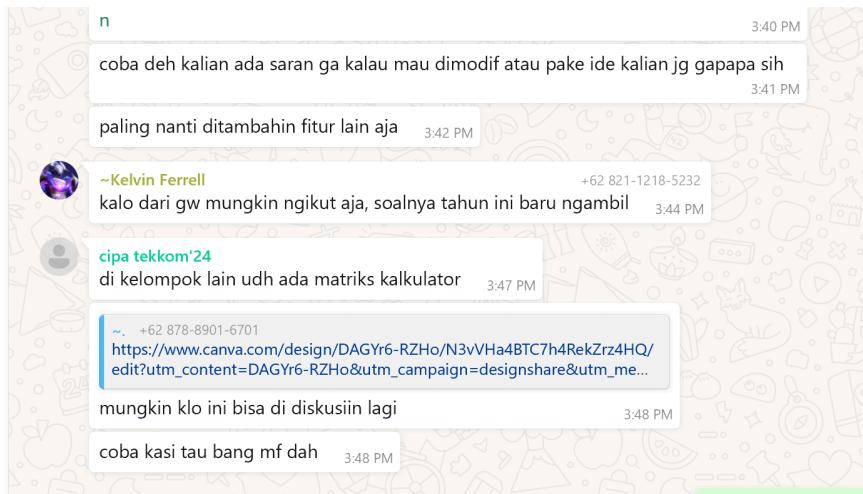
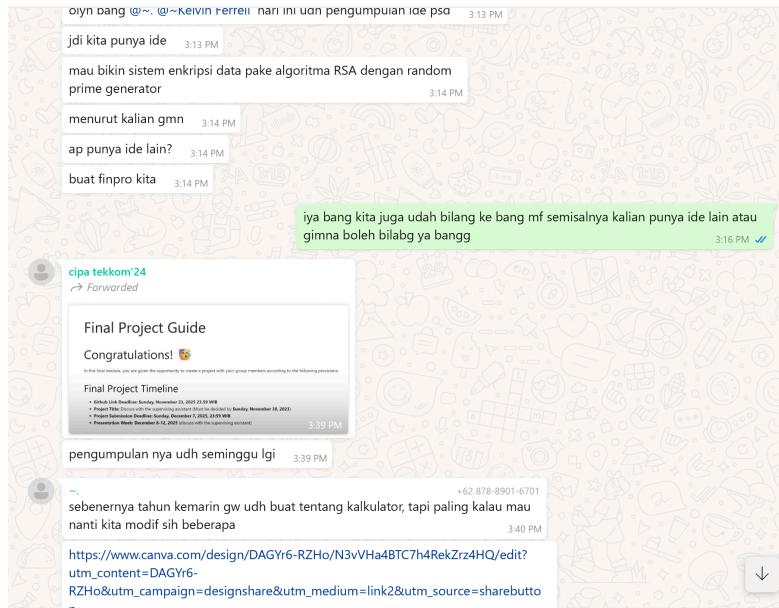


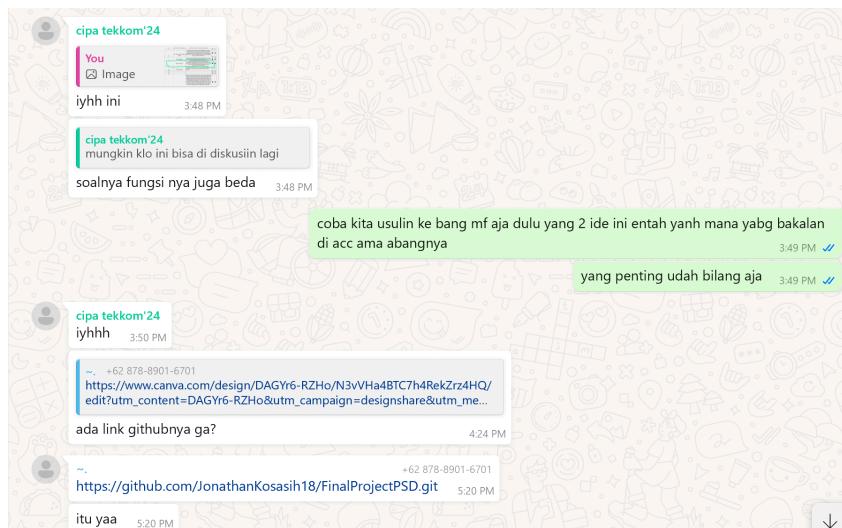


Appendix B: Documentation

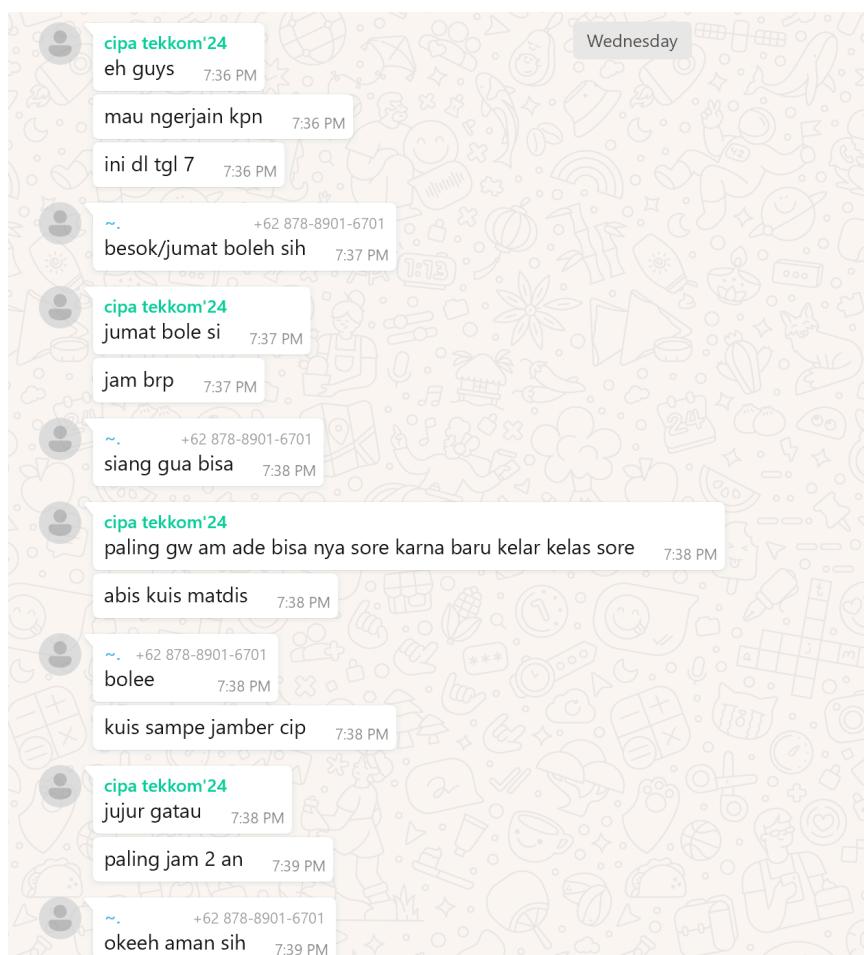
Put the documentation (photos) during the making of the project

30 November 2025





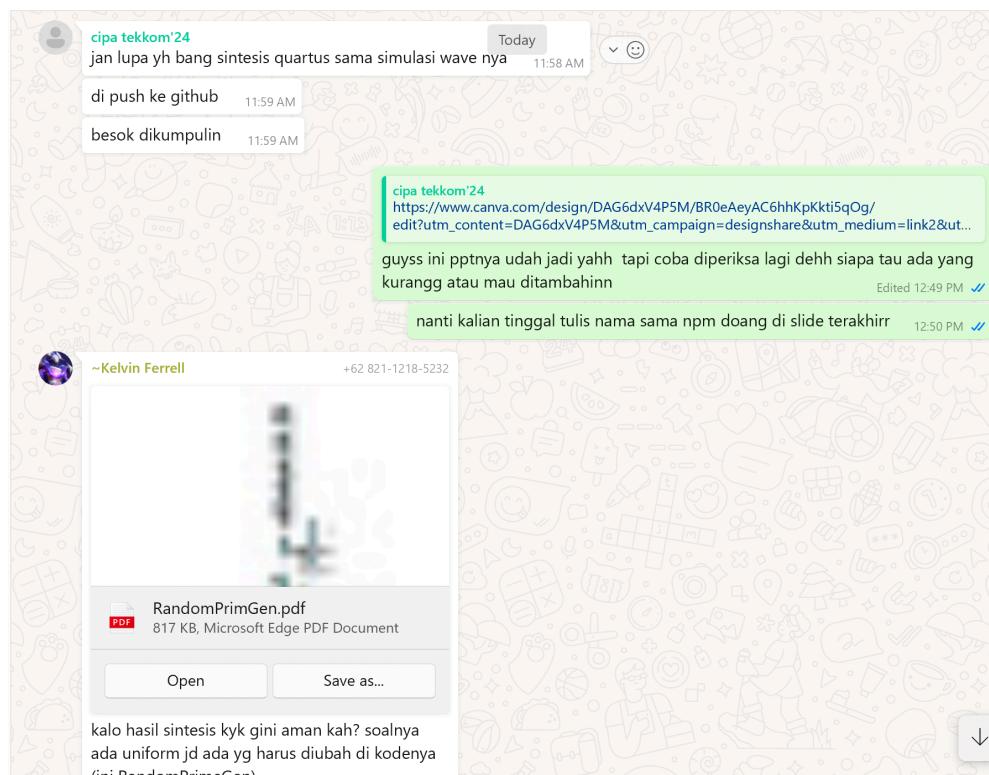
3 Desember 2025

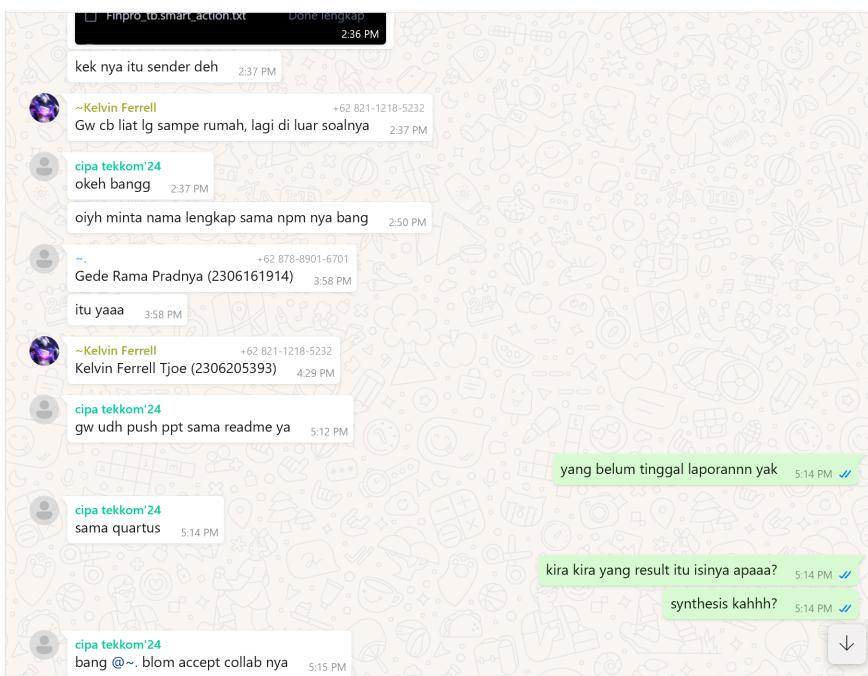


5 Desember 2025



6 Desember 2025





7 Desember 2025

