Implementasi Keamanan Sistem IoT dengan Arduino Uno menggunakan Algoritma RSA dan Diffie-Hellman

Hugo Sabam Augusto- 13522129[1]
Program Studi Teknik Informatika
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jl. Ganesha 10 Bandung 40132, Indonesia
[1]13522129@itb.ac.id

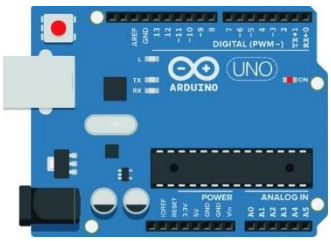
***Abstract*—Keamanan sudah menjadi prioritas utama terutama di zaman Industri 4.0 terutama dalam perkembangan sistem Internet of Thigns (IoT). Penelitian ini bertujuan untuk meningkatkan keamanan pada Sistem IoT menggunakan Arduino Uno dengan menerapkan algoritma keamanan RSA dan Diffie-Hellman. Implementasi kedua algoritma ini diharapkan dapat mengatasi tantangan keamanan yang sering dihadapi oleh perangkat IoT. RSA digunakan untuk enkripsi dan dekripsi data dan autentikasi, sementara Diffie-Hellman digunakan untuk pertukaran kunci aman antara perangkat IoT.**

***Kata Kunci*— arduino,iot,rsa,diffie-hellman**

I. PENDAHULUAN

Keamanan data memiliki peranan utama dalam melindungi kerahasiaan informasi, mengingat bahwa paparan atau kebocoran informasi tanpa kendali yang tepat dapat dimanfaatkan oleh pihak yang berpotensi merugikan. Pada era digital, perlindungan ini semakin diperkuat melalui pengembangan sistem keamanan jaringan komputer atau yang dikenal sebagai keamanan cyber (cybersecurity). Salah satu alat yang esensial dalam konteks ini adalah kriptografi, yang secara umum merupakan proses menyampaikan pesan secara rahasia dan tersembunyi. Dalam konteks teknologi digital, kriptografi menjadi disiplin ilmu yang mempelajari teknik enkripsi naskah asli dengan menggunakan kunci enkripsi, mengubahnya menjadi teks yang sulit terbaca tanpa kunci dekripsi. Sejarah kriptografi mencakup periode kejayaan Yunani kuno dan masa pemerintahan Julius Caesar di Romawi, menunjukkan evolusi konsep untuk menjaga keamanan sistem. Jenis kriptografi melibatkan hash function, public key cryptography, dan symmetric key cryptography, masing-masing dengan kegunaan dan keamanan tersendiri. Dalam konteks modern, aspek-aspek seperti autentikasi, kerahasiaan, integritas, dan ketidakbisaan untuk menyangkal transaksi menjadi fokus penting dalam memahami dan menerapkan kriptografi untuk melindungi data secara efektif.

Kriptografi, sebagai teknik penyampaian pesan yang tersembunyi, memegang peran krusial dalam menjaga keamanan data pada era digital. Dalam menggali lebih dalam mengenai penggunaan kriptografi dan implementasinya dalam keamanan cyber, pemahaman tentang definisi umum kriptografi menjadi esensial. Kata "kriptografi" berasal dari bahasa Yunani, dengan "kryptos" yang berarti rahasia dan "graphein" yang berarti menulis. Di era digital, kriptografi berkembang sebagai disiplin ilmu yang memanfaatkan teknik enkripsi untuk melindungi naskah asli, mengubahnya menjadi teks sulit terbaca tanpa kunci dekripsi. Sejarah kriptografi mencatat peranannya pada masa Yunani kuno dan pemerintahan Julius Caesar di Romawi. Jenis kriptografi mencakup hash function, public key cryptography, dan symmetric key cryptography, yang masing-masing memiliki kegunaan unik. Dalam modern kriptografi, aspek-aspek seperti autentikasi, kerahasiaan, integritas, dan ketidakbisaan untuk menyangkal transaksi menjadi fokus utama, memperkuat keamanan data di era informasi yang terus berkembang.



Gambar 1.1 Arduino Uno
(Sumber : https://stock.adobe.com)

Arduino adalah platform pengembangan perangkat keras open-source dengan papan mikrokontroler seperti Arduino UNO. Mudah diprogram, memiliki banyak pin input/output, dan digunakan untuk berbagai proyek elektronika. Kelebihannya meliputi sifat open-source, kemudahan pemrograman, dan fleksibilitas, menjadikannya populer di kalangan pengembang dan pembuat.

Arduino Uno merupakan salah satu mikrokontroler yang dapat digunakan untuk mengimplementasikan konsep Internet of Things (IoT). Dengan Arduino Uno, pengguna dapat menghubungkan perangkat fisik ke internet dan mengumpulkan serta mentransmisikan data melalui jaringan. Beberapa cara untuk menghubungkan IoT dengan Arduino Uno melibatkan penggunaan sensor untuk mengukur lingkungan sekitar, kemudian mentransmisikan data tersebut ke cloud atau perangkat lain melalui koneksi internet.

Hybrid cryptography adalah pendekatan yang menggabungkan kelebihan dari kriptografi asimetris (seperti RSA) dan

kriptografi simetris (seperti Diffie-Hellman) untuk mencapai keamanan yang optimal dalam proses komunikasi dan pertukaran informasi. Dalam hybrid cryptography, biasanya, kunci enkripsi simetris digunakan untuk mengamankan data secara efisien dan cepat, sedangkan kunci enkripsi asimetris digunakan untuk mengamankan pertukaran kunci simetris itu sendiri.

Arduino Uno dapat diintegrasikan dengan kriptografi algoritma seperti RSA dan Diffie-Hellman untuk meningkatkan keamanan komunikasi antarperangkat di tingkat lokal. Penerapan ini memungkinkan Arduino Uno sebagai perangkat mikrokontroler untuk mengendalikan dan mengumpulkan data dari sensor-sensor fisik. Meskipun implementasi keamanan dan penggunaan sensor terjadi secara lokal, fondasi ini dapat menjadi langkah awal dalam menciptakan solusi Internet of Things (IoT) yang lebih luas, di mana data yang diamankan dan dihasilkan oleh Arduino Uno dapat dikirimkan dan dikelola melalui jaringan global.

II. LANDASAN TEORI

A. Teori Bilangan

Teori bilangan adalah cabang matematika murni yang mendalami sifat-sifat bilangan bulat dan struktur matematika yang muncul dari propertis tersebut. Melibatkan konsep faktorisasi bilangan, teori bilangan mempelajari bagaimana setiap bilangan bulat positif dapat diuraikan secara unik sebagai hasil kali bilangan prima.

Salah satu dari teori yang menerapkan konsep teori bilangan adalah Teorema Euclidean. Teorema Euclidean, dalam konteks teori bilangan, menyatakan bahwa ada tak terbatas banyaknya bilangan prima. Ini berarti tidak mungkin memiliki daftar lengkap bilangan prima, karena kita selalu dapat menemukan bilangan prima baru. Teorema ini merupakan kontribusi dari Euclid dan memiliki dampak signifikan dalam pengembangan teori bilangan serta aplikasinya dalam berbagai bidang matematika.

Misal ada *m* dan *n* bilangan bulat denga *n* > 0, Apabila *m* dibagi dengan *n* maka menghasilkan pembagian *q* dan menyisakan *r* sehingga menjadi sedemikian rupa

$$m = \text{nq} + \text{r}$$

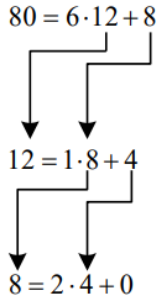
Dari penjelasan Teorema Euclidean, kita dapat menghubungkannya dengan konsep PBB (Pembagian Bersama Terbesar) dalam teori bilangan. Teorema Euclidean membuka jalan untuk memahami sifat-sifat dasar bilangan bulat, termasuk pembuktian eksistensi PBB. PBB dari dua bilangan adalah bilangan bulat positif terbesar yang dapat membagi keduanya tanpa sisa.Misal, faktor pembagi bersama 45 dan 36 adalah 1,3 dan 9. Diambil 9 karena angka faktor pembagi terbesar.

Dengan memahami konsep PBB (Pembagian Bersama Terbesar) yang muncul dari Teorema Euclidean, kita dapat mencapainya secara efisien melalui Algoritma Euclidean. Algoritma ini menyederhanakan proses pencarian PBB dari dua bilangan dengan langkah-langkah iteratif menggunakan sifat-sifat dasar Teorema Euclidean.

Misalkan *m* dan *n* adalah bilangan bulat tak negatif dengan m lebih besar sama dengan n.Diperoleh langkah-langkah secara berturut-turut pembagian untuk memperoleh cara berikut:

$$r_0 = r_1 q_1 + r_2 \qquad 0 \le r_2 < r_1$$
$$r_0 = r_2 q_2 + r_3 \qquad 0 \le r_3 < r_2$$
$$\vdots$$
$$r_0 = r_{n-1} q_{n-1} + r_n \qquad 0 \le r_n < r_{n-1}$$
$$r_0 = r_n q_n + 0$$

Jadi, Apabila dihubungkan dengan Teorema Euclidean, PBB dari m dan n adalah sisa terakhir yang tidak nol dari runtunan pembagian tersebut.Berikut ilustrasi perhitungannya, diketahui *m* = 80 dan *n* = 12, disini sudah terpenuhi syarat bahwa *m* lebih besar sama dengan *n*.



Gambar 2.1 ilustrasi perhitungan PBB
(Sumber : https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/14-Teori-Bilangan-Bagian1-2023.pdf)

Jadi, bisa dilihat bahwa pembagian terakhir sebelum 0 adalah 4, maka dari Algoritma Euclidean di atas, dapat diambil bahwa Faktor PBB dari 80 dan 12 adalah 4.

Beralih ke Aritmatika Modulo. Aritmatika Modulo membahas hubungan antar bilangan dalam kelompok tertentu, di mana setiap operasi aritmatika dilakukan terhadap sisa hasil bagi bilangan dengan suatu modulus. Penerapan Aritmatika Modulo memiliki peran penting dalam kriptografi, teori kode, dan berbagai bidang matematika terapan lainnya. Misalkan *a* dan *m* bilangan bulat dengan m > 0. Maka operasi modulo **a mod m** atau bisa dibaca dengan "a modulo m" akan memberikan sisa jika *a* dibagi dengan *m*. Notasi dapat diperjelas menjadi

$$a \bmod m = r$$

Sehingga bisa membentuk persamaan dari teorema Euclidean yaitu

$$a = mq + r \quad 0 \le r < m$$

Dari persamaan di atas, m disebut modulus atau modulo dan hasil dari aritmatika modulo m terletak di dalam sebuah himpunan {0,1,2,3,….,m-1}.

Aritmatika modulo bisa digunakan di dalam Kekongruenan. Misalkan kita memiliki persamaan modulo 38 mod 5 = 3 dan 13 mod 5 = 3. Dilihat dari hasil bahwa kedua persamaan hasil modulo yang sama yaitu 3, sehingga bisa dikatakan dengan

$$38 \equiv 13 \ (mod\ 5)$$

Modulo juga memiliki balikan atu biasa disebut moduo invers.Apabila meninjau Kembali aritmatika bilangan real, sedemikian sehingga a x b = 1.Dengan kata

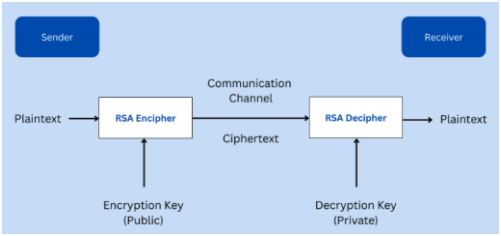
lain, jika a adalah bilangan tidak-nol, balikannya adalah $\frac{1}{a}$, yang dapat diilustrasikan dengan contoh 4 x $\frac{1}{4}$ = 1. Di dalam aritmatika modulo, balikan modulo sebuah bilangan bulat lebih sulit untuk dihitung. Untuk sebuah modulo memiliki balikan, haruslah memenuhi syarat : Jika a dan m memiliki relative prima dan $m > 1$, maka balikan dari a (mod m) itu ada. Sedemikian sehingga

$$xa \equiv 1 \ (mod \ m), \quad / \quad a^{-1}(mod \ m) = x$$

B. Algoritma RSA

RSA adalah algoritma kriptografi asimetris yang menggunakan sepasang kunci, yaitu kunci publik dan kunci privat. Kunci publik diberikan kepada semua orang, sedangkan kunci privat tetap rahasia. Contoh penerapan algoritma ini adalah ketika klien mengirimkan kunci publiknya ke server untuk meminta data. Server mengenkripsi data menggunakan kunci publik klien, dan hanya klien yang dapat mendekripsi data tersebut menggunakan kunci privatnya.

Keamanan RSA didasarkan pada kesulitan faktorisasi bilangan besar, di mana kunci publik dan privat terdiri dari hasil perkalian dua bilangan prima besar. Peningkatan panjang kunci secara eksponensial meningkatkan kekuatan enkripsi, meskipun kunci 1024-bit dianggap rentan.

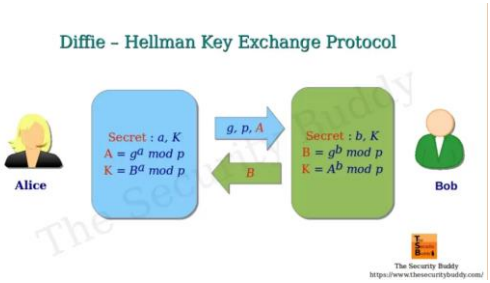


Gambar 2.2 Flowchart algoritma RSA
(Sumber : https://www.boardinfinity.com/blog/rsa-algorithm-in-c/)

Prosedur pembuatan pasangan kunci di dalam RSA dilakukan secara bertahap.Pertama ambil 2 buah bilangan prima, sebut saja p dan q dengan kedua variable tersebut berstatus dirahasiakan.Lalu kalikan p dan q dan simpan dalam variable n lalu cari *totient euler* (phi Euler) dengan cara $m = (p - 1)(q - 1)$. Setelah itu, ambil sebuah variable public sebagai kunci public (e) yang relatif prima terhadap m(alias PBB dari e dan m bernilai 1). Lalu cari kunci dekripsi (d) dari persamaan kekongruenan $ed \equiv 1 \ (mod \ m)$. Terakhir, setelah didapatkan semua variable yang dibutuhkan, tinggal kita masukkan ke rumus enkripsi-dekripsi

$$C = P^e \ mod \ n \quad (enkripsi)$$
$$P = C^d \ mod \ n \quad (dekripsi)$$

C. Algoritma Diffie-Hellman



Gambar 2.2 Flowchart algoritma RSA
(Sumber : https://www.thesecuritybuddy.com/encryption/how-does-diffie-hellman-key-exchange-protocol-work/)

Protokol pertukaran kunci Diffie-Hellman memungkinkan dua pihak untuk merahasiakan pertukaran kunci di lingkungan yang tidak aman. Diffie-Hellman menggunakan konsep matematis untuk menghasilkan kunci bersama tanpa mentransmisikan kunci tersebut secara langsung. Dalam pertukaran kunci ini, kedua pihak sepakat pada parameter tertentu untuk menghasilkan kunci privat masing-masing. Pendekatan ini membuat pertukaran kunci aman dari pihak ketiga yang mungkin memantau komunikasi, dan kunci yang dihasilkan dapat digunakan untuk menyusun kunci enkripsi yang digunakan selama sesi komunikasi.

Langkah-langkah cara kerja algoritma Diffie-Hellman dalam pertukaran kunci adalah sebagai berikut:

1. **Parameter Awal**
   Setiap pihak memilih parameter awal bersama yaitu sebuah bilangan prima p dan generator g.
2. **Pilihan kunci rahasia**
   Setiap pihak memilih kunci privatnya sendiri, a untuk pihak A dan b untuk pihak B.
3. **Penghitungan kunci public**
   Kedua Pihak akan menghitung kunci publiknya:
   $$A = g^a \ mod \ p$$
   $$B = g^b \ mod \ p$$
4. **Pertukaran kunci public**
   Pihak A mengirim kunci A ke pihak B
   Pihak B mengirim kunci B ke pihak A
5. **Perhitungan kunci bersama**
   $$K = B^a \ mod \ p$$
   $$K = A^b \ mod \ p$$

Algoritma Diffie-Hellman memanfaatkan sifat eksponensial dalam aritmetika modulo, sehingga walaupun $A^b \ mod \ p$ dan $B^a \ mod \ p$ secara matematis berbeda, keduanya akan menghasilkan nilai yang sama, yaitu kunci K. Pendekatan ini menjaga kerahasiaan pertukaran kunci di lingkungan yang tidak aman.

$$(g^b)^a \ mod \ p = (g^a)^b \ mod \ p = g^{ab} \ mod \ p = K$$

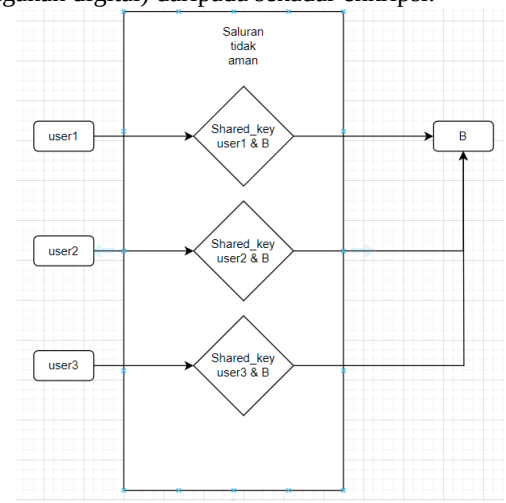
Dari persamaan di atas, bisa disimpulkan bahwa apabila kita memiliki nilai g,a,p maka akan mudah untuk mendapatkan kunci bersama, namun apabila kita hanya memiliki kunci bersama, sangat susah untuk mencari nilai a, sehingga memerlukan informasi tambahan.

III. IMPLEMENTASI

Untuk Implementasi Kombinasi dari RSA dan Diffie-Hellman itu sendiri. Pertama, Diffie-Hellman Key Exchange: A dan B melakukan pertukaran kunci Diffie-Hellman untuk menghasilkan kunci bersama (shared key). Dalam proses ini, kunci bersama ini bisa digunakan untuk mengenkripsi dan mendekripsi, dan merupakan kunci simetris sebagai bukti autentikasi antar 2 pihak.

Kedua, Enkripsi Shared Key menggunakan RSA: Salah satu pihak, Kasus ini hanya User yaitu anggap A mengenkripsi kunci bersama (shared key) dengan kunci public RSA (e). Tujuan dari langkah ini adalah menyertakan tanda tangan digital, bukan sekadar enkripsi. . Pihak A mengirimkan pesan terenkripsi (kunci bersama yang ditandatangani) kepada B melalui saluran tidak aman. Pihak lain, B(Mikrokontroler Arduino), yang memiliki kunci private RSA(d), dapat menggunakan kunci mendekripsi pesan dan mendapatkan kunci bersama

Ketiga,Verifikasi dan Dekripsi: Pihak B menerima pesan dan menggunakan kunci privat RSA nya (d) untuk mendekripsi pesan. Dalam proses ini, B juga dapat memverifikasi tanda tangan digital yang diterapkan oleh A menggunakan kunci publik A. Tanda tangan digital digunakan untuk otentikasi dan integritas pesan, sedangkan enkripsi dilakukan untuk menjaga kerahasiaan. Kombinasi Diffie-Hellman dengan RSA dapat memberikan otentikasi dan pertukaran kunci yang aman di saluran tidak aman. Ini adalah contoh dari apa yang sering disebut sebagai "Hybrid Cryptosystem" yang menggabungkan keuntungan dari kriptografi simetris dan asimetris.Jadi, dalam skenario ini, RSA digunakan lebih untuk otentikasi dan integritas pesan (penandatanganan digital) daripada sekadar enkripsi.



Gambar 3.1 Sketsa alur program
(Sumber : dokumentasi pribadi)

Shared key untuk user1 dan B dengan user2 dan B itu berbeda, sehingga B bisa mengetahui autentikasi user1, user2, user3, dan untuk mencegah adanya pihak misalnya pura-pura menjadi user1 padahal aslinya user3, dibuat sistem enkripsi RSA di saluran tidak aman. Dengan ini tingkat keamanan menjadi lebih tinggi.

Perangkat keras yang digunakan untuk uji-coba implementasi ini menggunakan Mikrokontroller Arduino Uno.Pertama, saya inisialisasi berbagai variable dahulu dengan kode berikut (dalam Bahasa C++ dengan header Arduino.h).Sebelum itu, perlu ditekankan bahwa dalam implementas ini, hanya digunakan integer berjumlah kecil karena kekuatan komputasi masih tergolong lemah dan lambat untuk integer yang besar, perlu memerlukan library tamabahan seperti BigInt.

```
#include <Arduino.h>

const int p = 17;
const int q = 7;
const int g_DH = 5;
int a = 0; // Nilai awal a, dapat diubah selama runtime
const int b = 5;

int e, d, n, shared_key_DH, A, B, signed_key, verified_key;
boolean aChanged = false; // Tandai apakah nilai a telah diubah
```

Gambar 3.2 kode inisialisasi variabel
(Sumber : dokumentasi pribadi)

Terlihat di kode bahwa saya pertama meng-inisialisasi kedua variable *p* dan *q* (bilangan prima) secara berturut yaitu 17 dan 7.*p* ini akan digunakan sebagai variable prima yang digunakan nantinya Ketika RSA dan Diffie-Hellman. Berikutnya, untuk variable generator Diffie-Hellman(Kedepannya akan disingkat DH) saya pasang angka 5. Lalu terdapat variable *a* / private key pengguna (dapat diubah-ubah selama runtime) dan *b* saya buat sebagai kunci rahasia milik si mikrokontroller Arduino bernilai 5. Lalu masih banyak variable lain yang nantinya akan didapatkan dari berbagai algoritma yaitu *e,d,n,shared_key_DH,A,B,signed_key*, dan *verified_key*.

```
void setup() {
  Serial.begin(9600);

  n = p * q;
  int totient = (p-1) * (q-1);
  Serial.print(" Totient:  "); Serial.println(totient);
  e = select_e(totient);
  d = mod_inverse(e, totient);
  Serial.println("Enter the value for private key A (a): ");
}
```

Gambar 3.3 kode *setup* sebelum runtime
(Sumber : dokumentasi pribadi)

Terlihat jelas dari kode *setup* bahwa dilakukan perhitungan variable *n* yang nantinya digunakan sebagai nilai modulo di RSA.Lalu dicari nilai *totient* (atau *m* bila di dalam landasan teori) dengan rumusnya kemudian kunci publik *e* yang nantinya digunakan untuk enkripsi *shared_key*. Lalu kita mencari nilai *d* yang digunakan nanti untuk dekripsi *shared_key* dengan fungsi *mod_inverse* yang berfungsi mengembalikan nilai balikan modulo dari *e* dan *totient* / *m* sebagai parameter fungsi

$$ed \equiv 1 \ (mod\ m).$$

```
A = modPow(g_DH,a,p);
B = modPow(g_DH,b,p);
shared_key_DH = modPow(B, a,p);
//shared_key_dh = modPow(A,b,p);

signed_key = encrypt(shared_key_DH, e, n);
verified_key = decrypt(signed_key, d, n);
```

Gambar 3.4 Potongan kode bagian looping
(Sumber : dokumentasi pribadi)

Potongan kode selanjutnya akan memasuki ke bagian looping

/ *runtime*. Disini baru akan dilakukan salah satu proses dari algoritma DH yaitu bagian penghitungan kunci public DH. untuk A (sebagai user) dan B (milik mikrokontroller Arduino). Terlihat di kode bahwa saya menggunakan fungsi "modPow".Fungsi ini bisa dieksekusi dengan modPow(int a, int b,int p) untuk menghitung $a^b \ mod \ p$.

Untuk logika jalannya program ini sudah siap dan tinggal di jalankan di dalam mikrokontroller Arduino.Pertama, akan diambil dahulu beberapa sampel percobaan untuk dilihat, yang pertama saat kita ambil a (nilai private user) = 2. Hasil bisa didapatkan dari Serial Monitor Arduino sebagai berikut:

```
Totient / m:  96
n           : 119
Enter the value for private key A (a):  2
Setting private key (a) to: 2
Diffie-Hellman Parameters:
  Prime (p): 17
  Generator (g): 5
  Private Key A (a): 2
  Private Key B (b): 5
  Public Key A (A): 8   13
  Public Key B (B): 14
  Shared Key: 9

RSA Parameters:
  Public Key (e, n): (5, 119)
  Private Key (d, n): (77, 119)

Signed and Encrypted Key: 25
Verified and Decrypted Key: 9
```

Dari data di atas, terdapat beberapa variabel kunci, seperti nilai A dan B ($A = g^a \ mod \ p$ atau sebaliknya), yang merupakan kunci publik a dan b yang nantinya akan digunakan dalam algoritma Diffie-Hellman (DH) untuk menghasilkan kunci bersama yang bersifat rahasia antara dua pihak. Setelah mendapatkan nilai kunci bersama menggunakan rumus K = B^a mod p, misalnya 9, user akan mengenkripsi kunci tersebut dengan kunci publik (e), menghasilkan nilai yang didekripsi menjadi 25. Selanjutnya, kunci bersama yang terenkripsi dikirimkan dari A ke B melalui saluran tidak aman, seperti internet.

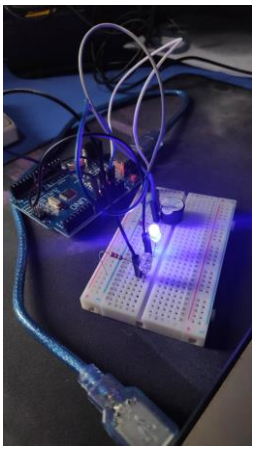
Ketika B menerima kunci bersama yang terenkripsi, B akan mendekripsi dan melakukan validasi serta verifikasi terhadap kunci tersebut, terkait dengan identitas pengirim, misalnya, B dengan user1. Setelah proses verifikasi dan validasi selesai, B dapat melanjutkan ke proses selanjutnya, contohnya akses ke suatu perangkat, yang dalam implementasi ini direpresentasikan oleh lampu LED dan buzzer.

Namun, perlu diperhatikan bahwa jumlah pengguna yang dapat mengakses Mikrokontroller Arduino terbatas, yaitu hanya 16 orang. Jika jumlah pengguna melebihi batas ini, maka ada potensi terjadinya kolisi nilai kunci bersama, di mana dua pengguna yang berbeda dapat memiliki nilai kunci bersama yang sama dengan B. Hal ini terjadi karena dalam algoritma DH, nilai yang dihasilkan di-modulus oleh p, dalam kasus ini sebesar 17, sehingga interval nilai hanya (p-1). Setiap kunci bersama juga memiliki nilai enkripsi yang unik. pastinya di dalam interval (n-1), dalam kasus ini bernilai $n = p \ x \ q = 117$. Berikut data pasangan kunci bersama dengan private key user dengan nilai enkripsinya. Namun, ada beberapa kasus unik yang sama temukan ketika mengumpulkan data, apabila suatu user menggunakan private key(a) = 4,maka dihasilkan kunci bersama bernilai 13 yang dihasilkan dari algoritma DH. Selanjutnya, ketika di enkripsi, dihasilkan pula angka yang sama yaitu 13 , sama pula ketika nilai kunci privat user bernilai 16 maka akan dihasilkan kunci bersama bernilai 1 dan kunci enkripsi bersama juga bernilai 1.

Tabel 3.1 persebaran nilai kunci

a	Shared_key (DH)	Encrypted_Shared_Key (RSA)
1	14	63
2	9	25
3	7	28
4	13	13
5	12	3
6	15	36
7	6	41
8	16	67
9	3	5
10	8	43
11	10	40
12	4	72
13	5	31
14	2	32
15	11	44
16	1	1
17	14 (kolisi dengan a = 1)	63 (kolisi dengan a = 1)

Sekarang untuk representasi, dibuatlah rangkaian Arduino seperti berikut



Gambar 3.5 Foto rangkaian arduino
(Sumber : dokumentasi pribadi)

Untuk implementasi ini, saya menggunakan cara representasi verifikasi dan validasi hubungan antara pengguna dan Arduino melalui dua lampu LED dan buzzer. Saya merancang algoritma sederhana, di mana jika *verified_key* (kunci bersama yang telah diverifikasi) yang didekripsi memiliki nilai ganjil, maka LED

kiri akan menyala. Sebaliknya, jika *verified_key* memiliki nilai genap, maka LED kanan akan menyala. Selain itu, jika nilai verified_key di luar rentang (0-16), yang menunjukkan pengguna tidak dikenal oleh Arduino, buzzer akan menyala. Hal ini dirancang untuk memberikan sinyal visual dan suara terkait status hubungan antara Arduino dan pengguna.

Dengan pendekatan ini, perangkat memberikan tanggapan yang mudah dipahami terkait keamanan dan verifikasi pengguna. Lampu LED dan buzzer berperan sebagai indikator visual dan suara yang jelas untuk menginformasikan status verifikasi hubungan, memudahkan pemahaman pengguna terkait keberhasilan atau kegagalan verifikasi dengan Arduino.

IV. KESIMPULAN

Implementasi keamanan sistem IoT dengan Arduino Uno menggunakan algoritma RSA dan Diffie-Hellman menunjukkan bahwa pendekatan ini memberikan lapisan keamanan yang kuat terhadap pertukaran kunci dan komunikasi di saluran tidak aman. Melalui Diffie-Hellman, tercapai pertukaran kunci simetris yang aman antara perangkat IoT Arduino Uno. Selanjutnya, dengan memanfaatkan algoritma RSA, implementasi ini memberikan otentikasi, integritas, dan keamanan tambahan melalui tanda tangan digital. Pendekatan ini, yang menggabungkan keunggulan kriptografi simetris dan asimetris, memberikan solusi yang andal dalam menjaga keamanan sistem IoT di lingkungan yang rentan terhadap serangan.

V. UCAPAN TERIMA KASIH

Saya ingin menyampaikan rasa terima kasih yang tulus kepada Tuhan Yang Maha Esa atas bimbingan-Nya selama penulisan karya ilmiah ini. Juga, terima kasih yang sebesar-besarnya untuk Pak Rinaldi Munir dan Pak Monterico Adrian, Dosen Matematika Diskrit, yang telah memberikan bimbingan dan ilmu pengetahuan yang sangat berarti. Keberhasilan penyelesaian karya ilmiah ini tidak terlepas dari kontribusi mereka. Terima kasih atas dedikasi dan bantuan yang diberikan. Semoga Tuhan selalu memberkati langkah-langkah kita semua.

REFERENCES

[1] Arduino.Arduino Uno Rev3 Datasheet. https://docs.arduino.cc/resources/datasheets/A000066-datasheet.pdf (Diakses 7 Desember 2023 pukul 18.00)

[2] Amazon Web Services. What is IoT?. https://aws.amazon.com/id/what-is/iot/ (Diakses 7 Desember 2023 pukul 19.00).

[3] Munir,Rinaldi. Teori Bilangan (Bag.1), https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/14-Teori-Bilangan-Bagian1-2023.pdf , (Diakses 8 Desember 2023 pukul 15.00

[4] Munir,Rinaldi. Teori Bilangan (Bag.3), https://informatika.stei.itb.ac.id/~rinaldi.munir/Matdis/2023-2024/16-Teori-Bilangan-Bagian3-2023.pdf, (Diakses 8 Desember 2023 pukul 15.00)

[5] GeeksforGeeks. (09 Nov, 2023). RSA Algorithm in Cryptography. https://www.geeksforgeeks.org/rsa-algorithm-cryptography/ (Diakses 8 Desember 2023, pukul 16.00).

[6] Comparitech. (August 24, 2023). Diffie-Hellman Key Exchange. https://www.comparitech.com/blog/information-security/diffie-hellman-key-exchange/ (Diakses 8 Desember 2023 pukul 19.00).

[7] DoubleOctopus. Diffie Hellman Algorithm. https://doubleoctopus.com/security-wiki/encryption-and-cryptography/diffie-hellman-algorithm/#:~:text=Diffie%20Hellman%20uses%20a%20private,for%20a%20symmetric%20key%20algorithm. (Diakses 8 Desember 2023, pukul 20.00).

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.

Bandung, 3 Desember 2023

Hugo Sabam Augusto
13522129