RABIT : Jurnal Teknologi dan Sistem Informasi Univrab
Volume xx No. xx | Januari xxx : Hal :xx-xx
ISSN CETAK : 2477-2062
ISSN ONLINE : 2502-891X

PEMBANGKITAN KUNCI YANG DIGUNAKAN UNTUK PENENTUAN KONSTANTA P DAN Q YANG PRIMA BERDASARKAN INFORMASI PERANTI

[1)] Yogi Arif Widodo, [2)] Mulyanto, S.Kom., M.Cs., dan [3)]Bedi Suprapty, S.Kom., M.Kom.
[1,2,3)]Program Studi, Teknik Informatika, Politeknik Negeri Samarinda
[1,2,3)]Jl. Cipto Mangun Kusumo Sungai Keledang – Samarinda - Indonesia
E-mail : *yogirenbox33@gmail.com, yanto1294@gmal.com, dan bedirheody@gmail.com*

ABSTRAK
Bilangan Prima adalah bilangan yang bila difaktorkan hanya habis dibagi oleh angka 1 dan dengan dirinya sendiri dan lawan dari Bilangan Prima adalah Bilangan Komposit, yaitu bilangan habis dibagi oleh bilangan yang lain lebih dari 2. keunikanya selalu berbentuk antara 6k-1 atau 6k +1. Setiap bilangan hanya memiliki 6 bentuk: 6k - 2, 6k - 1, 6k, 6k+1, 6k + 2, 6k +3 membuatnya jadi kunci penting dalam keamanan data karena hasil GCD kurang dari 10 menandakan waktu pemfaktoran cukup lama, terutama pada Kriptografi. Bilangan prima dibangkitkan dan ditetapkan dalam 2 variabel yaitu *p* dan *q*, semakin jauh rentang antara *p* dan *q*, maka $GCD\ (p-1, q-1)$ tidak terlalu besar atau lebih dari 10. Bilangan konstanta atau orde *p* dan *q* menjadi eksperimen aritmatika menggunakan kombinasi informasi peranti waktu dalam bentuk jam, menit dan detik. Pembangkitan awal ditentukan dengan membatasi batas atas prima dengan kalimat atau kata yang diubah ke *ASCII* dan nilainya dijumlahkan, menghasilkan *n* = 3400. Dengan teknik sederhana *naive solution* dimana $2\ ke\ n-1$ menghasilkan angka prima sebanyak 478 atau $arrayListPime = 2,3,5..n$. Kombinasi peranti waktu jam berperan dalam pembentukan *p* dan *q* dipengaruhi oleh jam , menit dan detik dengan ketentuan yang sedemikian rupa menjadi posisi atau *index*. Waktu yang digunakan adalah ketika terjadi aritmatika yaitu 15:05:48 GMT + 8. Zona waktu kemudian ditentukan berdasarkan probabilistik oleh *pseudorandom* untuk mengubah zona awal ke zona lain menjadi 09:05:49 GMT - 10 berdasarkan 24 jenis zona waktu. *Exception Handling* diterapkan sebagai *monitoring* konsep kombinasi informasi peranti waktu, sehingga didapat hasilnya tidak ada pengecualian tangkapan menandakan aritmatika berhasil dalam menentukan *p = 157* dan *q = 263* dan hasil GCD = 1.

Kata Kunci: Bilangan Prima, Informasi Peranti Waktu, P dan Q , *Android Mobile*

ABSTRACT
Prime Numbers are numbers which, when factored, are only divisible by 1 and by themselves and the opposite of Prime Numbers are Composite Numbers, i.e., those numbers are divided by other numbers more than 2. The uniqueness is always in the form of 6k-1 or 6k +1. Each number has only 6 forms: 6k - 2, 6k - 1, 6k, 6k + 1, 6k + 2, 6k +3 making it an important key in data security because GCD results of less than 10 indicate a long factoring time, especially in Cryptography . Prime numbers are generated and defined in 2 variables, p and q, the farther the range between p and q, the GCD (p - 1, q - 1) is not too large or more than 10. Constant numbers or order p and q become arithmetic experiments uses a combination of time device information in the form of hours, minutes and seconds. The initial generation is determined by limiting the upper limit of prima by a sentence or word converted to ASCII and the value is summed, resulting in n = 3400. With a simple naive solution technique where 2 to n - 1 produces a prime number of 478 or arrayListPime = 2,3,5. .n The combination of the time clock device plays a role in the formation of p and q influenced by hours, minutes and seconds with such provisions being the position or index. The time used is when arithmetic takes place which is 15:05:48 GMT + 8. The time zone is then determined based on probabilistic by pseudorandom to change the initial zone to another zone to 09:05:49 GMT - 10 based on 24 types of time zones. Exception Handling is applied as a monitoring concept of the combination of time device information, so that the results obtained are no exception catches indicating that arithmetic is successful in determining p = 157 and q = 263 and the results of GCD = 1.

Keyword: Prime Number, Information Time Device, P and Q, Android Mobile

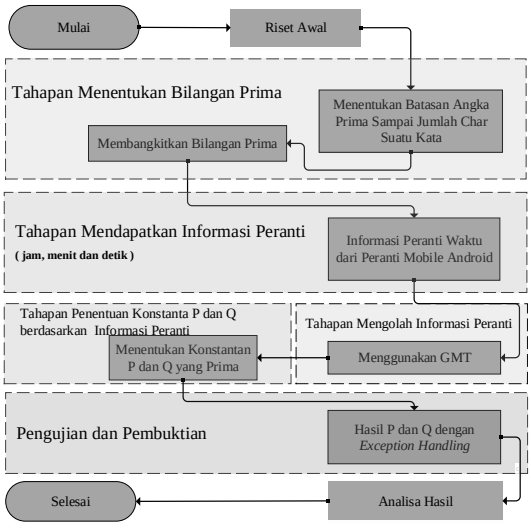
PENDAHULUAN

Bilangan prima adalah bilangan yang hanya memiliki dua faktor: 1 dan bilangan itu sendiri. Satu-satunya bilangan prima bernilai genap hanyalah 2 [1]. Kemudian apakah 1 bilangan prima? tentu saja tidak. 1 hanya memiliki 1 faktor pembagi. Kita tidak menghitung 1 sebanyak dua kali. $\forall n \in N, n > 1$, maka n selalu memiliki setidaknya 1 faktor prima.

Setiap bilangan asli lebih dari 1 yang tidak prima disebut bilangan komposit [2]. Jika n adalah suatu bilangan komposit, maka n memiliki setidaknya 1 faktor prima yang nilainya tidak lebih dari $\sqrt{n}$. Bilangan prima > 3 memiliki keunikan yang selalu berbentuk antara [3] 6k-1 atau 6k +1 [4]. Setiap bilangan hanya memiliki 6 bentuk: 6k - 2,6k - 1,6k, 6k+1 ,6k + 2,6k +3. Tapi perhatikan bahwa 6k -2,6k, 6k +2 selalu genap. Sedangkan 6k +3 adalah kelipatan 3. Maka dari itu bilangan prima yang lebih dari 3 akan selalu memiliki antara dua bentuk tadi. Hasil selanjutnya didapat mengenai bilangan prima adalah bahwa bilangan prima ada tak hingga banyaknya [5] [6]. Berdasarkan sifat Bilangan Prima maka penilitian ini mengkombinasi informasi peranti waktu jam, menit dan detik pada *android mobile* menjadi teknik penentuan konstanta *p* dan *q* juga memastikan ketentuan prosesnya terpenuhi dan menghasilkan pola tersendiri tanpa ada *NumberFormatException*.

METODE

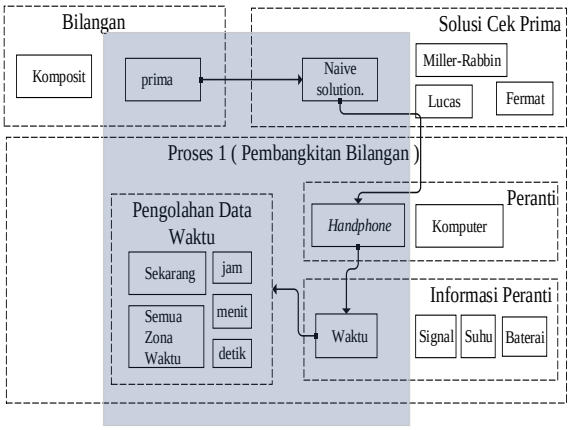
Berdasarkan pendahuluan, pembangkitan dan menentukan konstanta *p* dan *q* yang prima maka penelitian menggunakan informasi peranti dapat digambarkan dalam bentuk diagram alir metode penelitian.



Gambar 1. Diagram Alir Metode Penelitian

Kerangka Konsep Penelitian

Kerangka konsep penelitian (teori atau konsep ilmiah yang digunakan sebagai dasar penelitian) menjelaskan hubungan atau gabungan alur sebagai ruang lingkup penelitian.



Gambar 2. Kerangka Konsep Penelitian

Bilangan Prima merupakan bilangan bulat positif, sifat pembagiannya [7] melahirkan konsep-konsep aritmetika modulo, dan salah satu konsep bilangan bulat yang digunakan dalam penghitungan komputer. Dengan ditemukannya bilangan prima, teori bilangan berkembang semakin jauh dan lebih mendalam. Banyak dalil dan sifat dikembangkan berdasarkan bilangan prima.

Pada penelitian [8] bilangan prima merupakan bilangan istimewa dalam Al-Qur'an karena definisi bilangan prima yaitu bilangan yang tidak bisa dibagi dengan bilangan lain kecuali satu dan bilangan itu sendiri yang menampilkan sifat Allah yang tidak dapat dibagi dengan siapapun kecuali diri-Nya.

Berdasarkan penilitian [9] Pengecualian atau *Exception Handling* merupakan cara bersih memeriksa kesalahan tanpa mengacaukan kode dan mampu menangkap pengecualian sebuah aritmatika salah satunya *NumberFormatException*. Klausa tangkapan diikuti blok coba (*try and catch*), setiap blok tangkapan merupakan pengecualian yang

menangani jenis pengecualian, Pengecualian ini cocok digunakan pada Pengujian dan Pembuktian terhadap proses Tahapan Penentuan Konstanta P dan Q berdasarkan Informasi Peranti.

HASIL

Hasil proses tahapan menentukan bilangan prima, mendapatkan informasi peranti, mengolah informasi peranti dan penentuan konstanta p dan q berdasarkan informasi peranti, pengujian dan pembuktian dan analisa hasil menggunakan perangkat *visual studio code*, *android studio*, dan *android mobile*.

Tahapan Menentukan Bilangan Prima

Tahapan ini memiliki 2 langkah yakni Menentukan Batasan Angka Atas Prima Sampai Jumlah Suatu *Char* dan Membangkitkan Bilangan Prima.

a. Menentukan Batasan Angka Prima Sampai Jumlah Suatu *Char*, Misalnya dari kalimat "Politeknik Negeri Samarinda Tahun 2020" Diuraikan menjadi kode *ASCII* yang diperlihatkan pada Tabel 1.

Tabel 1 Hasil Karakter ke *ASCII*

char	P	o	...	n
ASCII	80	111	...	n

Kemudian dengan persamaan 1.1 didapat totalnya = 3400.

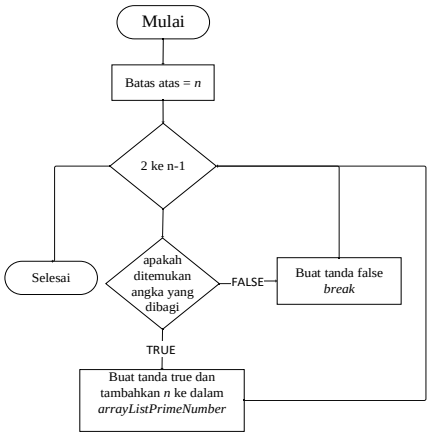
$$total = \sum_{i=1}^{n} Ui \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots (1.1)$$

dimana :
Total = Batas Atas Prima
Ui = Nilai Karakter Pada *ASCII*

Matematikawan membuktikan bahwa bilangan prima terbesar itu tidak ada, Pada Juli 2018 bilangan prima 'terbesar' ditemukan, yaitu $2^{277.232.917} - 1$ yang diketahui [10]. Proses pembatasan prima mengkonsumsi sebuah waktu yang berhubungan dengan tahapan pengolahan informasi peranti jam, menit dan detik.

b. Membangkitkan Bilangan Prima dengan mengeliminasi angka bukan prima [11]. Penerapanya sederhana dilakukan dengan *naive solution* sebagai berikut:

1. Ketika Melalui semua angka dari 2 ke *n*-1, maka setiap nomor periksa apakah ia membagi *n*.
2. Jika ditemukan angka yang dibagi, akan mengembalikan tanda *false*
3. Sebaliknya *true* dan simpan nilai *n* ke dalam *arrayListPrimeNumber*.



Gambar 3. *FlowChart Naive Solution*

Nilai *n* telah ditentukan pada tahap sebelumnya yaitu *n* = batas atas prima. Hasil rentang 1 sampai *n* membangkitkan 478 angka prima (jumlah angka prima)

Tahapan Mendapatkan Informasi Peranti

Informasi Peranti yang didapatkan memiliki 3 variabe yaitu jam, menit, dan detik. Proses mendapatkannya dibaca oleh peranti *Mobile Android* dengan fungsi yang sudah tersedia di kotlin yang diperlihatkan pada Gambar 4.

Gambar 4. Potongan Kode *Kotlin* Mendapatkan Informasi Peranti Waktu Sekarang

Data waktu yang didapat masih berupa nilai keseluruhan waktu yaitu 1594886148236 yang kemudian diformat menjadi (HH:mm:ss) untuk menjadikanya jam, menit dan detik.

Dengan fungsi yang sudah tersedia di kotlin, dapat digunakan *syntax* sebagai berikut :

$$val\ dfTime = SimpleDateFormat(\text{HH:mm:ss})$$

Maka didapatkan waktu sekarang 15:55:48 sebagai Informasi Peranti dengan zona awal yang didapat GMT +8.

Tahapan Mengolah Informasi Peranti

Informasi Peranti diolah kembali untuk menghasilkan informasi yang probabilstik berdasarkan waktu jam, menit dan detik menggunakan *Greenwich Mean Time Zone* (GMT) sebagai pengubah.

Seluruh zona waktu telah didefinisikan sebelumnya ke dalam *arrayTime* sebagai zona lain.

Tabel 2 Daftar Waktu Indonesia Tengah

Waktu Tengah Dunia			
GMT (-)			GMT (+)
GMT-1	GMT-6	GMT+1	GMT+6
GMT-2	GMT-7	GMT+2	GMT+7
GMT-3	GMT-8	GMT+3	**GMT+8**
GMT-4	GMT-9	GMT+4	GMT+9
GMT-5	GMT-10	GMT+5	GMT+10
	GMT-11		GMT+11
			GMT+12
			GMT+13

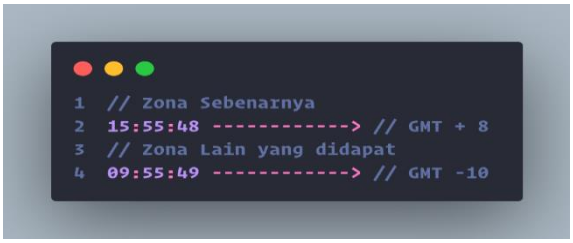
Pemilihan posisi atau *index* untuk *arrayTime* berdasarkan keluaran dari nilai *integer* oleh *sudorandom*, sebagai zona lain. Dengan fungsi yang sudah tersedia di kotlin, dapat digunakan syntax sebagai berikut :

$$val\ sudoRandom = (listZoneTime.indices).random()$$

Maka hasil nilai *sudoRandom* = 9.
Sehingga didapat *arrayTime* [*sudoRandom*] = GMT -10.
Kemudian dilakukan konversi waktu sekarang 15:55:48 GMT +8 ke GMT -10 yang diperlihatkan pada Gambar 5 dan Gambar 6.
.

Gambar 5. Potongan Kode *Kotlin* Konversi Zona Waktu

Informasi yang digunakan adalah zona lain, perubahan zona sendiri merupakan proses, tujuanya mengkonsumsi sebuah waktu ketika mendapatkan informasi waktu itu sendiri.



Gambar 6. Hasil Zona Awal dan Zona Lain

Tahapan Penentuan Konstanta P dan Q Berdasarkan Informasi Peranti

Penentuan telah dilakukan dengan melihat syarat sebagai berikut:

1. Bilangan yang prima telah didapatkan dalam bentuk *arrayListPrimeNumber* hasilnya berdasarkan pada Tahapan Membangkitkan Bilangan .

2. Informasi Peranti telah didapatkan dalam bentuk bagian dari waktu jam, menit dan detik. Hasilnya diperlihatkan pada Gambar 6.

Kemudian menentukan p dan q dimana $arrayListPrimeNumber = p = q$ untuk menghasilkan prima yang deterministik dari informasi peranti yang probabilistik.

a. Menentukan Konstanta P yang Prima , Penentuan ini sederhana, dengan menghitung persamaan 1.1 didapat *indexP* = 36.

($P_{penentuan}$ ……..……(1.1)

$$hh * n = indexP$$

Dimana :
hh = informasi peranti waktu jam
n = 4

Maka didapat nilai *p*[*indexP*] = 157.

Jika *n* memiliki nilai yang lebih besar dari 2 , misal 3 maka memiliki tujuan terbentuknya *p* yang prima cukup besar.

Dengan *p* yang besar, memiliki kesempatan *Greatest Common Divisor* GCD(*p, q*) yang hasilnya kecil atau proses pemfaktoran yang memakan waktu.

b. Menentukan Konstanta Q yang Prima nilai *q* memiliki aturan mirip dengan nilai *p*, tetapi memiliki 5 keputusan perhitungan ($q_{keputusan}$) dari 6 ketentuanya ($q_{ketentuan}$).

($q_{ketentuan}$)……..…(2.1)
K1 = *p*
K2 = informasi peranti waktu menit
K3 = informasi peranti waktu detik
K4 = K2 + K3
K5 = K1 * K2
K6 = K2 * K3

($q_{keputusan}$)……..……(2.2)

$$K[n] = \begin{cases} n = 0, & K[(jml\ prima - k1)] \\ K1 < Kn\ dan\ K1\ != Kn, & K[n] \end{cases}$$

Dimana :
K[n] = *arrayListPrimeNumber* [n]
jml prima = *arrayListPrimeNumber.size*

Dengan persamaan 2.1 dan 2.2 didapat K1 > K2 dan mengembalikan nilai K2 seperti yang diperlihatkan pada Tabel 3.

Tabel 3 Hasil ($q_{keputusan}$) dan ($q_{ketentuan}$)

informasi peranti waktu			09:05:49
arrayListPrimeNumber [n].size			478
K1	36	K1 < Kn && K1 != Kn	*RETURN value Kn as [n] as K[n]*
K2	55	K1 < K2 && K1 != K2	*TRUE*
K3	49	K1 < K3 && K1 != K3	*TRUE*
K4	104	K1 < K4 && K1 != K4	*TRUE*
K5	495	K1 < K5 && K1 != K5	*TRUE*
K6	2695	K1 < K6 && K1 != K6	*TRUE*
K[(jml prima-k1)]	442	n = 0 (*else*)	
$q_{keputusan}$		K[n]	K[55]
k[n] =	arrayListPrimeNumber [n]		
k[31] =	263		

K2 = 55 sebagai *index* atau [n] dengan begitu K[n] = K[55] = *indexQ*, Maka didapat nilai *q*[*indexQ*] = 263.

Tahapan ini berhasil menentukan dan menghasilkan p = 157 dan q = 263 sesuai ketentuan yang ditetapkan dan telah diuji pada pengujian primalitas dengan *naive solution* dan pembuktian terhadap Tahapan Penentuan Konstanta P dan Q Berdasarkan Informasi Peranti dengan *Exception Handling*.

Pengujian dan Pembuktian

Pengujian dan Pembuktian dilakukan terhadap hasil *p* dan *q* dengan *naive solution* dan *Exception Handling*, selain untuk cek bilangan prima sederhana terhadap *p* dan q, juga bisa digunakan sebagai pembangkit bilangan prima.

Pada Tahapan Menentukan Bilangan Prima yaitu pada Membangkitkan Bilangan Prima tepatnya Gambar 3 diprogramkan seperti yang diperlihatkan Gambar 6

```
// loop 2 - batas atas jumlah seluruh element kata/kalimat
for (normalNumber in 2..limitPrimeNumber(wordLimitPrime)) {
// loop dari 2..limit ex : 2..3 = 2,3
    var isPrime = true // flag isPrima || !isPrima = false
    // mencari bilangan prima
    for (primeNumber in 2 until normalNumber)
// loop dari 2 until num ex : 2 until 4 = 2,3
        if (normalNumber % primeNumber == 0) {
// check normalNumber is prime ? ex : 4 % 2 == 0 !isPrime
            isPrime = false // set flag to false
            break // berhenti membaca next script dan kembali ke perulangan awal
        }
    if (isPrime) // flagPrime true ?
        arrayListPrimeNumber.add(normalNumber.toString())
// memasukan angka normalNumber ke ArrayList
}
```

Gambar 6. Potongan Kode *Kotlin* Membangkitkan Bilangan Prima dengan *Naive Solution*

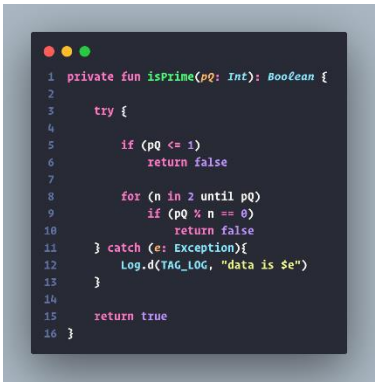
Tabel 4 Hasil Bilangan Prima dengan Naive Solution

arrayListPrime	prima	2	3	5	..	3391
	size	1	2	3	..	478

Kemudian dari hasilnya adalah benar sebuah bilangan prima yang setiap nilainya adalah *true*.

Analisa Hasil

Hasil *p* dan *q* yang dibangkitkan berdasarkan informasi peranti waku jam, menit dan detik merupakan bilangan prima yang rata-rata menghasilkan panjang *p* dan *q* sebanyak 7 *bit* sampai 14 *bit* selama uji pembangkitan sebanyak 12 kali dalam tempo waktu setiap 5 menit dalam 1 jam dan benar *p* dan *q* adalah bagian dari bilangan prima berdasarkan uji primalitas sederhana dengan *naive solution* yang diperlihatkan pada Gambar 8 yang tidak jauh berbeda dengan Gambar 6.



```
private fun isPrime(pQ: Int): Boolean {

    try {

        if (pQ <= 1)
            return false

        for (n in 2 until pQ)
            if (pQ % n == 0)
                return false
    } catch (e: Exception){
        Log.d(TAG_LOG, "data is $e")
    }

    return true
}
```

Gambar 8. Potongan Kode *Kotlin* Cek Prima *Naive Solution*

Berdasarkan hasil Tahapan Penentuan Konstanta P dan Q Berdasarkan Informasi Peranti.

Penerapan *exception handling* berhasil tidak menangkap pengecualian dalam konsep penentuannya, maupun pengecualian secara menyeluruh.

```
try {
    // blok kode proses kombinasi informasi peranti waktu
} catch (e: ArithmeticException) { // menangkap pengecualian aritmatika
    Log.d(TAG_LOG, "catch errArithm $e")
} catch (e: NumberFormatException){
// menangkap pengecualian NumberFormatException
    Log.d(TAG_LOG, "catch errNum $e")
} catch (e : Exception){ // menangkap pengecualian lainya / menyeluruh
    Log.d(TAG_LOG, "catch err $e")
}
```

Gambar 7 Potongan Kode *Kotlin* Pengecualian Proses

Hasil kombinasi informasi peranti waktu jam, menit dan detik, memberikan pola sedemikian rupa terhadap hasil *p* dan *q*, dengan bantuan informasi berupa nilai yang digunakan sebagai posisi atau *index* dan telah dibuktikan penentuan dalam ketentuan *p* dan *q* dengan *monitoring Exception Handling*, rumusnya telah berfungsi untuk setiap bilangan yang dibangkitkan atau ditentukan.

Hasil pada penilitian [12] membuktikan bahwa bahasa pemrograman *kotlin* dapat mengurangi waktu kompilasi, waktu eksekusi dan dapat meningka keringkasan. Maka berdasarkan hal tersebut kemampuan konsep sederhana ini menjadi efisien.

KESIMPULAN

Penlitian dan percobaan terhadap rancangan dan pengujian yang telah dilakukan menghasilkan kesimpulan sebagai berikut:

Proses mendapatkan waktu ketika terjadi aritmatika yang diterapkan bergantung peranti yang digunakan, ketika peranti memiliki

ruang *memory* pengunaan yang besar, mempengaruhi data waktu.

Perhitungan dan proses lebih cepat (berbeda). Sehingga data waktu dan perhitungan membuat hasil p dan q lebih efisien dengan melihat hasil GCD $(p-1, q-1)$ tidak terlalu besar dan rentang dua variabel itu sendiri.

Pemanfaatan zona waktu menghasilkan 2 jenis ketentuan yang terhubung yaitu bilangan prima dan ketentuannya deterministik berdasarkan batas atas dan ketentuan p dan q nya dapat dikatakan probabilistik karena bergantung pada hasil keluaran *pseudorandom* dalam memilih zona lain dimana 15:55:48 GMT +8 menjadi 09:55:49 GMT – 10.

DAFTAR PUSTAKA

[1] Cahyo Dhea Arokhman Yusufi, *Heuristic - For Mathematical Olympiad Approach*. Jakarta: Math Heuristic, 2020.

[2] M. K. Harahap, "Membangkitkan Bilangan Prima Marsenne dengan metode Bilangan Prima Probabilistik Solovay – Strassen," vol. 1, no. Oktober, 2019.

[3] K. Chiewchanchairat, P. Bumroongsri, dan S. Kheawhom, "Improving fermat factorization algorithm by dividing modulus into three forms," *KKU Eng. J.*, vol. 40, no. March, hal. 131–138, 2016, doi: 10.14456/kkuenj.2015.1.

[4] J. W. P. Ferreira, "The Pattern of Prime Numbers," *Appl. Math.*, vol. 08, no. 02, hal. 180–192, 2017, doi: 10.4236/am.2017.82015.

[5] T. Sciences, "Dirichlet ' s Theorem Related Prime Gap," vol. 10, hal. 305–310, 2016.

[6] R. Meštrović, *Euclid's theorem on the infinitude of primes: a historical survey of its proofs (300 B.C.--2017) and another new proof*. 2018.

[7] F. F. Firmansyah, "Kajian matematis dan penggunaan bilangan prima pada algoritma kriptografi RSA (Rivest, Shamir, dan Adleman) dan algoritma kriptografi Elgamal [skripsi]," Malang (ID): Universitas Islam Negeri Maulana Malik Ibrahuim Malang, 2015.

[8] R. H. Sari, "Apakah Integrasi Islam dapat Membudayakan Literasi Matematika ?," *Semin. Mat. dan Pendidik. Mat. UNY*, hal. 655–662, 2017.

[9] J. Kumari, S. Singh, dan A. Saxena, "An Exception Monitoring Using Java," vol. 3, no. 2, hal. 12–18, 2015.

[10] "Untuk Apa Mencari Bilangan Prima Terbesar? - Anak Bertanya." [Daring]. Tersedia pada: https://anakbertanya.com/untuk-apa-mencari-bilangan-prima-terbesar/. [Diakses: 18-Jun-2020].

[11] A. TH dan B. MB, "The Unique Natural Number Set and Distributed Prime Numbers," *J. Appl. Comput. Math.*, vol. 06, no. 04, 2017, doi: 10.4172/2168-9679.1000368.

[12] M. J. Arockiajeyanthi,] T Mrs, dan Kamaleswari, "KOTLIN-A New Programming Language for the Modern Needs," *Int. J. Sci. Eng. Manag.*, vol. 2, no. 12, hal. 2456–1304, 2017.