Accelerating FHE BFV Decryptor and Encryptor

Hamdani Fadhli - 13217058
Program Studi Teknik Elektro
Sekolah Teknik Elektro dan Informatika
Institut Teknologi Bandung, Jalan Ganesha 10 Bandung
E-mail (gmail): hamdanifadhli@gmail.com

***Abstract*— Traffic internet semakin hari semakin padat. Semua data yang penting dan juga privacy dikirimkan melalui raffic internet hingga data pribadi seseorang seperti nomor rekening, password, data bank dan juga rumah sakit. Sistem kriptografi sekarang mengandalkan CPU untuk dapat mengerjakan beberapa permasalahan kriptografi. Tetapi semakin secure sebuah data, waktu pemrosesankan akan semakin lambat dan juga kompleks. Dibutuhkan sebuah alat untuk mempercepat kinerja dari proses komputasi kriptografi tersebut sebagai akselerator dari fungsi tersebut.**

Keywords*— *Data, Sistem Keamanan, Kriptografi, Akselerator, Privacy Data

I. INTRODUCTION

Sebuah data yang masuk dan keluar dari internet sangatlah besar. Traffic internet sendiri memiliki *grouth rate* yang cukup tinggi. Dari [1] dapat dilihat bahwa dalam kurun waktu yang cukup singkat kenaikan pengguna internet sangatlah berkembang pesat. Cukup dalam waktu liba belas tahun, pada tahun 2000 jumlah pengguna internet mencapai 400 miliar orang. Tetapi pada tahun 2015 pengguna internet naik signifikan hingga 3.2 Triliun orang. Semua pengguna internet secara langsung mengirimkan data mereka yang dimana *cloud* tidak dapat menjamin keamanan data pengirim, oleh karena itu dibutuhkan sebuah sistem yang dapat melakukan enkripsi terhadap data.

Teknologi untuk mengamankan data salah satunya menggunakan Teknik enkripsi. Enkripsi adalah sebuah proses yang akan membuat sebuah data tidak dapat terbaca oleh pihak yang tidak memiliki kuncinya untuk membaca data tersebut. Seperti yang dijelaskan di [2] bahwa untuk melakukan sebuah enkripsi data dibutuhkan ilmu yang bernama *Cryptography*, ilmu tersebut membahas tentang *the science of secret writing.* Menurut [2] ada tiga buah tipe algoritma *Cryptography*, pertama adalah *simetrical cryptography* yang mengandung *key* yang akan digunakan untuk melakukan enkripsi dan juga dekripsi sebuah data. Yang kedua adalah *asymmetrical cryptography* yang mengandung *public key* untuk melakukan enkripsi dan juga *secret key* untuk melakukan dekripsi dari sebuah data, dan yang terakhir adalah *hash function* atau yang biasa disebut dengan *one-way cryptography* yang tidak membutuhkan key sama sekali dan juga data tidak dapat di dekripsi Kembali.

Ilmu kriptografi juga memjelaskan semakin sukar sebuah data untuk dikembalikan menjadi pesannya Kembali maka keamanan data tersebut semakin tinggi. Pada [3] bahwa proses enkripsi dengan tingkat kemaanan yang tinggi membutuhkan waktu yang lama untuk menyelesaikan komputasinya yang dijelaskan menggunakan algoritma RSA. Dijelaskan pada [4] banyak sekali orang yang akan meninggalkan sebuah sistem website jika terlalu banyak loading dan juga page yang tidak interaktif. Jika ingin ditanamkannya sebuah sistem kemanan yang tinggi untuk sebuah website maka dibutuhkan sebuah pengamanan yang tinggi dengan waktu proses yang cukup cepat. Yang dijelaskan pada [4] jika dalam lima belas detik seseorang tidak mendapatkan apa yang dia inginkan maka dia akan meninggalkan website tersebut dan akan merugikan perusahaan yang bersangkutan.

Jika kita melihat sistem keamanan dikerjakan oleh CPU maka jika ada sebuah sistem komputasi lain yang dapat menyelesaikan persamaan dengan cepat selain CPU akan jauh lebih baik jika ingin dikaitkan dengan permasalahan yang dijelaskan sebelumnya. Pada [5] dijelaskan bahwa ada beberapa hal yang dapat menjalankan sebuah permasalahan komputasi yaitu FPGA, CPU, GPU dan juga ASIC. FPGA adalah sebuah hardware yang dapat menjalankan sebuah permasalahan komputasi dengan kecepatan hardware dan juga memiliki sebuah fleksibilitas software. Selain itu juga FPGA dapat melakukan sebuah permasalahan komputasi dengan daya yang rendah.

Dijelaskan pada [6] permasalahan kriptografi juga banyak diselesaikan menggunakan FPGA untuk mempercepat sebuah proses dalam algoritma enkripsi yang akan dikomputasionalkan. Selain mempercepat juga kebutuhan daya yang dibutuhkan oleh FPGA cukup rendah untuk memaksimalkan *resource* dalam implementasi sistem keamanan data ini. Permasalahan ini juga membutuhkan waktu yang cepat untuk memenuhi kebutuhan user yang akan menggunakan internet. Paper ini akan menjelaskan bagaimana performansi beberapa jenis CPU dan juga FPGA dapat menyelesaikan permasalahan kriptografi *Fully Homomorphic Encryption* dengan skema *BFV* dan juga beberapa desain hardware yang digunakan oleh FPGA untuk menyelesaikan permasalahan komputasi tersebut.

Gateway dari alat IOT semakin dapat menghandle banyak devais IOTnya akan semakin bagus. Seperti yang dijelaskan pada [7] bahwa tiap gateway akan menerima data dari tiap alat iot tersebut. Untuk melakukan enkripsi dari data yang diterima dengan jumlah yang sangat besar akan diimplementasikan sebuah algoritma enkripsi *fully homomorphic encryption* dimana algoritma tersebut dikhususkan untuk melakukan enkripsi terhadap data berbentuk angka dan membuat data

tersebut walaupun dalam keadaan terenkripsi tetap dapat dilakukan proses komputasi. Enkripsi tetap dilakukan satu persatu tetapi dekripsi dapat dilakukan sekaligus. Oleh karena itu dibutuhkan alat yang dapat melakukan proses enkripsi terhadap algoritma *fully homomorphic encryption* dengan cepat.

II. PRELIMINARIES

Pada bagian ini akan dibahas beberapa konsep yang akan berhubungan langsung dengan algoritma enkripsi dan juga dekripsi yang akan menjadi bahan eksperiment kali ini. Pembahasan konsep ini akan dibagi menjadi tiga buah bagian mulai dari *fully homomorphic encryption* dan *fully homomorphic encryption BFV scheme.*

A. *Fully Homomorphic Encryption*

Fully Homomorphic Encryption adalah sebuah algoritma enkripsi yang memiliki dan juga mempertahankan sifat *homomorphic* tersebut. Pada prinsipnya seperti yang jelaskan pada [8] FHE memperbolehkan sebuah proses komputasional *arbitrary* terhadap data yang sudah terenkripsi. Artinya data tersebut dapat dijumlahkan, dikalikan dan juga dilakukan proses komputasi walaupun kondisinya masih dalam terenkripsi.

Proses komputasi data yang terenkripsi berarti memproses data terenkripsi. Sebagai contoh, jika user memiliki sebuah fungsi f dan menginginkan untuk mendapatkan $f(m_1, \dots, m_2)$ dengan input m_1 dan m_2 sangat logis melakukannya saat kondisi terenkripsi. Misalnya kita ambil $c_1, \dots, c_n = Encypted(m_1), \dots, Encrypted(m_n)$ dengan n adalah jumlah message yang akan di enkripsi dan c adalah sebuah ciphertext hasil enkripsi dari sebuah message. Sangat logis untuk melakukan proses komputasinya dengan cara $f(m_1, \dots, m_2) = Decrypted(f(c_1, \dots, c_2))$ karena sifat *homomorphic* nya.

Tetapi ada beberapa buah parameter yang harus dipenuhi untuk dapat melaksanakan enkripsi, dekripsi dan juga melakukan sebuah komputasi terhadap data terenkripsi. Sebagaimana yang dijelaskan pada [9] bahwa algoritma dan juga skema-skema dari FHE tersebut membutuhkan sebuah Batasan agar dapat mempertahankan sifat homomorphicnya dab hyga agar dapat dilakukan proses dekripsi sesuai dengan hasiil yang sesuai yang di harapkan dimana penjelasannya ditujukan kepada skema enkripsi Gentry.

B. *Fully Homomorphic Encryption Brakerski/Fan-Vercauteren Scheme*

Fullly Homomorphic Encryption memiliki anyak sekali skema, salah satunya adalah skema Brakerski/Fan-Fercaunteren. Salahh satu sifat dari FHE adalah membuat satu buah angka meledak menjadi banyak sekali angka dengan banyak sekali error. Skema ini memanfaatkan ring polynomial dan juga polynomial degree untuk menyimpan pesan yang akan di enkripsi dan juga dekripsi nantinya. Seperti yang dijelaskan pada [10] FHE ini digunakan beberapa algoritma pembagian dan juga perkalian untuk membuat sebuah algoritma enkripsi dan juga dekripsi dengan polynomial degree. Basis dari enkripsi ini sangat dekat dengan Ring Learning with Errors (RLWE). Konstruuksi dari skema ini masih berdasar dari algoritma Craig Gentry, Shai Halevi dan juga Nigal Smart. Kebelihan dari algoritma FHE generasi ke du aini adalah efisiensinya yang sudah dapat di implementasikan di beberapa buah bidang.

RWLE adalah sebuah versi cincin dari LWE seperti yang tertera pada [11]. RWLE dapat disederhanakan menggunakan quantum algorithm untuk mendapatkan vector paling sederhana dan pendek untuk lattice yang iideal. Seperti yang tertera pada [12]. Dari studi tersebut diturunkanlah beberapa buah skema untuk melakukan sebuah algoritma enkripsi dan juga dekripsi untuk menghasilkan beberapa buah permasalahan dalam pembelajaran menggunakan error ini dan juga untuk dapat mendapatkan nilai keamanan dan juga integritas data terhadap data yang akan di enkripsi.

Ada beberapa buah skema untuk mendapatkan nilai enkripsi, dekripsi dan juga process dalam komputasi ciphertext tersebut.

1) *Secret Key Generation*

Secret key generation adalah sebuah proses matematis yang dilakukan untuk mendapatkan sebuah kunci privat yang nantinya digunakan untuk melakukan dekripsi terhadap data yang sudah terenkripsi. Proses dalam mendapatkan secret key ini menggunakan sebuah algoritma random integer yang akan membuat sebuah array sepanjang degree polynomial dengan nilai random diantara 0 dan juga 1.

$$sk = [1, 0, 0, 1, 0, 1, 1, 0, 0, 1] \; dengan \; n = 10$$

Dapat dilihat dari hasil sk yang didapatkan dengan Panjang n sebagai polynomial degree dirandom antara nilai 0 dan juga 1 sepanjang n yang nantinya akan diigunakan untuk menjadi nilai secret key.

2) *Public Key Generation*

Public key generation ini adalah sebuah algoritma pembentukan nilai kunci public yang nantinya akan digunakan untuk melakukan sebuah enkripsi dari sebuah data. Kunci public ini akan saling berhubungan dengan kunci privat yang sudah didapatkan.

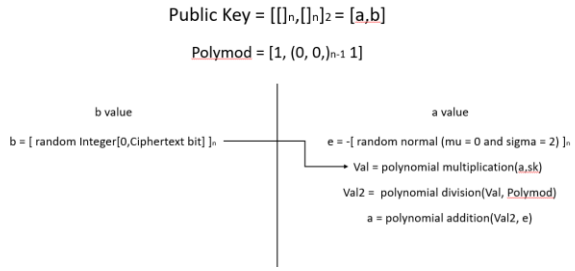


IMAGE I PUBLIC KEY GENERATION ALGORITHM

Dari image I dapat dilihat bahwa pembangkitan kunci public hanya dilakukan dengan nilai random dan juga nilai secret key yang sudah dibentuk. Oleh karena itu kunci public dan juga kunci privat adalah sepasang kunci yang saling berhubungan antara satu dan lainnya.

3) *Encrypt*

Untuk melakukan sebuah enkripsi terhadap message jika kita meminiki n polynomial degree maka kita dapat menyimpan n buah pesan dalam satu buah ciphertext. n buah pesan ini akan di gabungkan menjadi satu buah ciphertext. untuk melakukan enkripsi dibutuhkan sebuah public key dalam algoritmanya.

Message = $[m_1,\ m_2,\ m_3\ \dots\ ,\ m_n]$

Mp = $\left((Message\ \%\ t) * \left(\frac{q}{t}\right)\right)\ \%\ q$

e1 = [random normal]$_n$ e2 = [random normal]$_n$
u = [random integer[0,1]]$_n$

X_1 = polydiv(polymult(Pk[0], u),Polymod)
X_2 = polydiv(polymult(Pk[1], u),Polymod)

Ct_0 = x1+e1+Mp $\qquad$ Ct_1 = x1+e2

IMAGE II ENCRYPT ALGORITHM

Dari Image II dapat dilihat bahwa pesan yang dapat di enkripsi bergantung dengan polynomial degree yang akan digunakan. Tetapi ada sebuah constrain dimana semakin banyak polynomial degree yang akan dibentuk akan semakin lama juga proses komputasi yang nantinya akan dilakukan.

4) Decrypt

Untuk melakukan proses dekripsi dari sebuah nilai ciphertext dibutuhkan niilai kunci private dan juga modulus plain dan ciphertext.. dekripsi memiliki fungsi untuk mengembalikan nilai plaintext dari sebuah ciphertext

Ct = [ct_0, ct_1]

Scalled PT = Polyadd(ct_0, Polymult(ct_1,sk))

Decrypted PT = $\frac{Scalled\ PT * t}{q}\ \%\ t$

IMAGE III DECRYPT ALGORITHM.

Dari Image III dapat dilihat bahwa semua message dengan Panjang n dapat dikembalikan Kembali. Orang yang melakukan enkripsi seharusnya tau dia menaruh nilai enkripsi di index n ke berapa. Tetapi jika semua polynomial degree ditaruh nilai n juga tidak masalah.

5) Addition

Skema ini juga karena memiliki sifat homomorphic dapat melakukan sebuah proses penjumlahan walaupun dalam kondisi terenkripsi mau dengan nilai plaintext ataupun menggunakan nilai yang sudah terenkripsi juga. Jika ingin melakukan sebuah penjumlahan nilai tidak terenkripsi terhadap nilai yang terenkripsi dapat dilakukan proses seperti dibawah ini.

$$Ct_{new} = [[Ct_0 + NewPt], Ct_1]$$

Sedangkan jika ingin melakukan penjumlahan untuk dua buah nilai yang keduanya dalam kondisi terenkripsi dapat dilakukan dengan cara

$$Ct_{new} = [[Ct_{a0} + Ct_{b0}], [Ct_{a1} + Ct_{b1}]]$$

Hasil dari Ct_{new} tersebut akan menyimpan sebuah nilai hasil penjumlahan dari Ct a dan juga Ct b ataupun nilai Ct ditambahkan dengan sebuah plaintext.

6) Multiplication

Untuk dapat melakukan sebuah proses perkalian ciphertext jika akan mengkalikan ciphertext dan juga plaintext dan keduanya ciphertext akan melalui dua buah algoritma yang sangat berbeda satu dengan lainnya. Jika akan dilakukan sebuah perkalian plaintext dan juga ciphertext cukup mudah. Algoritma perkalian ciphertext dan juga plaintext dapat dilakukan dengan cata,

$$Ct_{new} = [[Ct_0 * NewPt], Ct_1]$$

Sedangkan untuk melakukan perkalian ciphertext dan juga ciphertext lainnya dibutuhkan sebuah evaluasi dari key yang akan digunakan. Evaluasi tersebut akan menghasilkan rlk dengan nilai,

$$rlk = [[-((p * q) * sk + e) + p * sk^2]_{p*q}, a]$$

Kemudian saat sudah didapatkan sebuah nilai rlk menggunakan ralin version evaluation seperti yang tertera pada [13] bahwa nilai rlk tersebut dapat digunakan untuk mengkalikan dua buah ciphertext. algoritma perkalian tersebut sebagai berikut,

$$(C_{2,0}, C_{2,1}) = \left[\frac{ct2[0] * rlk[0]}{p}, \frac{ct2[1] * rlk[1]}{p}\right]$$

Kemudian saat sudah dihasilkan nilai $C_{2,0}$ dan juga $C_{2,1}$ dilakukan proses untuk mendapatkan nilai ciphertext baru

$$Ct_{new} = \left[[Ct_0 + Ct_{2,0}], [Ct_1 + Ct_{2,1}]\right]$$

Ciphertext baru tersebut sudah menyimpan nilai perkalian dua buah Ciphertext yaitu Ct_1 dan juga Ct_1.

Dari enam buah skema tersebut didapatkan sebuah algoritma untuk melakukan enkripsi dan juga dekripsi data menggunakan FHE-BFV.

III. ACCELERATING CRYPTOGRAPHIC DECRYPTOR AND ENCRYPTOR

Pada bagian ini kita akan memberikan sebuah desain akselerator untuk fungsi dekriptor dari algoritma kriptografi *fully homomorphic encryption* untuk melakukan sebuah dekripsi dari sebuah variable terenkripsi. Dekripsi adalah sebuah kegiatan untuk mengembalikan message atau pesan dari sebuah variable yang sudah terenkripsi. Algoritma tersebut dapat dikomputasikan di beberapa jenis devais.

Untuk melakukan sebuah komputasi biasanya diselesaikan menggunakan CPU atau *Central Processing Unit.* Tetapi sebenarnya ada beberapa jenis hardware yang dapat menyelesaikan permasalahan komputasi tersebut. Seperti yang sudah dijelaskan di [14] bahwa ada empat buah devais yang dapat melakukan sebuah komputasi diantaranya adalah CPU, GPU, FPGA dan juga ASIC. Dijelaskan juga bahwa pada saat mengimplementasikan sebuah komputasi dalam suatu devais dimana CPU,GPU FPGA dan juga ASIC dibandingkan ada beberapa hal yang terlihat.

Saat keempat devais tersebut di bandingkan. CPU adalah sebuah devais yang sangat mudah untuk deprogram dan juga memiliki daya yang cukup hemat. Tetapi karena CPU digunakan Bersama dengan OS maka CPU membutuhkan waktu yang cukup lama untuk memproses sebuah komputasi, sedangkan jika kita melihat GPU untuk melakukan sebuah komputasi, dibutuhkan daya yang besar sekali tetapi kecepatan yang didapatkan lebih cepat dari kinerja CPU dan juga mudah untuk di program. Kemudian FPGA adalah sebuah devais yang cukup kompleks untuk di desain dan juga di implementasikan tetapi memiliki data yang hemat dan juga kecepatan komputasi yang jauh lebih cepat dibandingkan GPU sekalipun karena desainnya akan menjadi sebuah desain hardware. Terakhir adalah ASIC dijelaskan juga bahwa asic memiliki kemampuan komputasi yang cukup cepat sekali dan menggunakan daya yang cukup kecil untuk melakukan komputasi tetapi untuk membuat sebuah ASIC dibutuhkan waktu yang lama dan juga tidak fleksibel dalam kegunaannya. Untuk efesiensi daya dari FPGA, GPU dan juga CPU yang memakan daya paling besar untuk melakukan komputasinya adalah GPU kemudian disusul oleh CPU dan juga yang terakhir adalah FPGA seperti yang tertera pada [15].

Kemudian seperti yang tertera pada [16] bahwa penggunaan akselerator untuk mengakselerasi sebuah algoritma kriptografi adalah menggunakan FPGA. Seperti yang tertera pada thesis tersebut dapat dilihat akselerasi yang terjadi saat melakukan sebuah akselerasi pada algoritma kriptografi RSA tersebut dapat diakselerasi secara signifikan.

Kemudian kami mengimplementasikan sebuah akselerator FPGA untuk melakukan sebuah enkripsi dan dekripsi dari sebuah algoritma kriptografi *Fully Homomorphic Encryption* dengan skema BFV.

A. Design Accelerator

Untuk membuat akselerator FPGA kita harus membuat sebuah design dari akselerator tersebut. Permasalahan paling besar dari skema BFV adalah praktisnya mengimplementasi algoritma polynomial yang akan memakan waktu proses lama sekali. Ada beberapa blok design yang harus dibuat seperti blok multiplikasi, division dan juga polynomial addition.

1) Block Polynomial Multiplikasi

Untuk membuat blok multipliikasi kita dapat mengimplementasikan design systolic array. Systolic array seperti yang dijelaskan pada [17] adalah sebuah arsitektur parallel computer yang memiliki homogeneous network yang disebut dengan Data Processing Units yang dipanggil sebagai nodes.

Arsitektur perkalian yang kita buat terinspirasi dari penggunaan systolic array yang akan dimasukkan ke tiap Processing Element tiap value dari nilai perkalian A dan juga B kemudian kita stream hingga mendapatkan output.

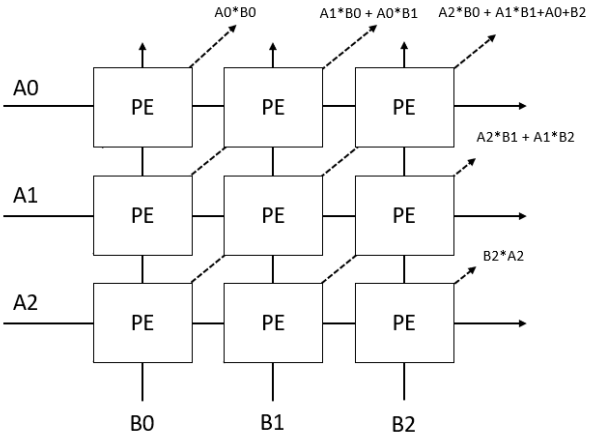


IMAGE IV POLYNOMIAL MULTIPLICATION

IMAGE IV adalah design blok polynomial multiplication dan juga tiap processing elementnya terdiri dari D Flip-Flop dan juga blok untuk perkalian dan juga penambahan sebuah integer.

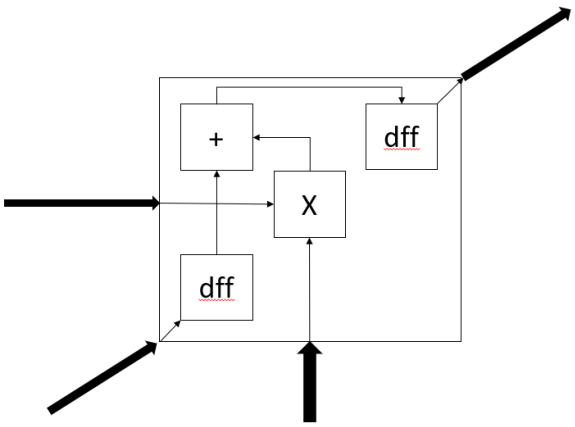


IMAGE V PROCESSING ELEMENT

Dapat dilihat dari IMAGE V. Data dari processing element yang di stream secara diagonal di save menggunakan D Flip-Flop kemudian nilai hasil perkalian A dan B juga diperlakukan dengan cara yang sama kemudian kedua buah data tersebut diteruskan ke processing element selanjutnya

2) Block Polynomial Division

Algoritma yang digunakan untuk block ini adalah pembagian sebuah nilai polynomial dengan polyMod. Polymod adalah sebuah array dengan index 0 bernilai 1 dan index n bernilai 1. Sisanya bernilai 0.

Kami mendesign sebuah block pembagian polynomial dengan polymod dengan menggunakan pengurangan saja dengan mengharapkan optimisasi pembagian polynomial dapat diselesaikan dalam satu clock cycle.

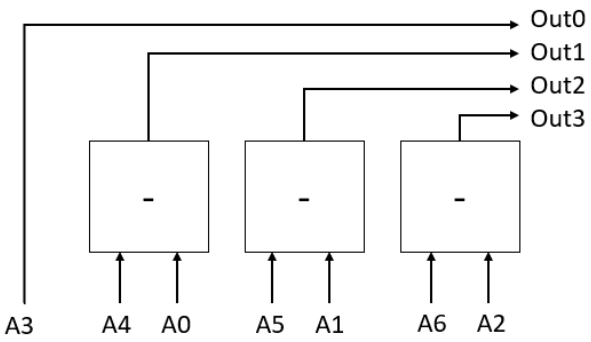


IMAGE VII DIVISION BLOCK

Dari gambar diatas kita index array paling kecil adalah nilai polynomial degree paling tinggi sedangkan untuk index array paling besar memiliki nilai polynomial degree 0. Konfigurasi pengurangannya adalah nilai polynomial degree lebih lebih kecil dikurangi dengan polynomial degree yang lebih besarnya. Output dari block ini berupa sisa bagi dari pembagian polynomial tersebut.

3) Block Polynomial Addition

Design polynomial addition dilakukan dengan cara menjumlahkan array of signal dengan array of signal yang akan kita jumlahkan.

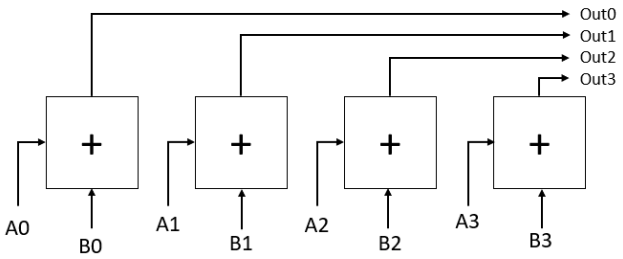


IMAGE VII ADDITION BLOCK

Dari IMAGE VII dapat kita dapatkan hasil penjumlahan dari tiap sinyal kemudian hasilnya akan menjadi output dari block polynomial addition ini.

4) Top Level Block

Kemudian kita membutuhkan sebuah blok yang akan menggabungkan semua block diatas agar menjadi sebuah block fungsi decryptor. Top level ini akan menerima input dua nilai ciphertext dan juga secret key yang kemudian akan mengeluarkan output variable yang sudah terdekripsi.

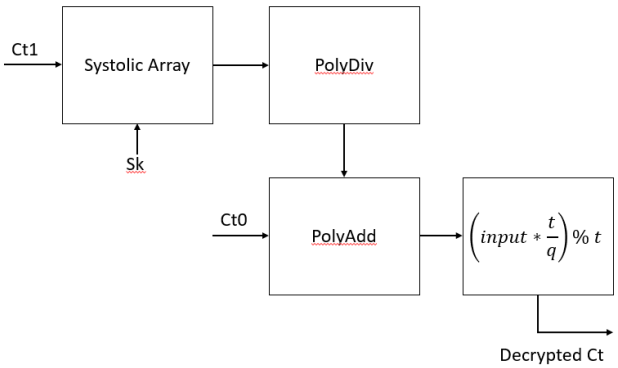


IMAGE VIII TOP LEVEL DECRYPTOR

Seperti yang terlihat pada IMAGE VIII, desain ini dapat digabungkan menjadi sebuah block untuk melakukan dekripsi dari sebuah variable. Jika kita analisis dari semua block design diatas, design ini hanya membutuhkan $n * 2 - 1$ clock cycle untuk dapat menyelesaikan seluruh proses komputasi dekripsi dari algoritma BFV ini.

5) Top Level Design Encryptor

Kemudian kita membutuhkan sebuah block yang akan menyatukan seluruh block yang sudah dijelaskan diatas agar menjadi sebuah system yang dapat mengimplementasikan enkripsi dari enkripsi *fully homomorphic encryption* dengan skema BFV. Top level ini akan menerima lima buah input yaitu Mp, e0, e1, u dan juga pk yang akan menghasilkan dua buah ciphertext. Persamaan yang akan diselesaikan adalah persamaan yang ada di bawah ini,

$$X0 = polydivremainder(polymult(Pk[0], u), [1,0,0,\dots,0,1]_n)$$
$$X1 = polydivremainder(polymult(Pk[1], u), [1,0,0,\dots,0,1]_n)$$
$$return\ ([X0 + e0 + Mp], [X1 + e1])$$

Block yang dirancang akan mendapatkan hasil $return\ ([X0 + e0 + Mp], [X1 + e1])$ yang akan menjadi sebuah ciphertext dari kumpulan message yang di enkripsi.

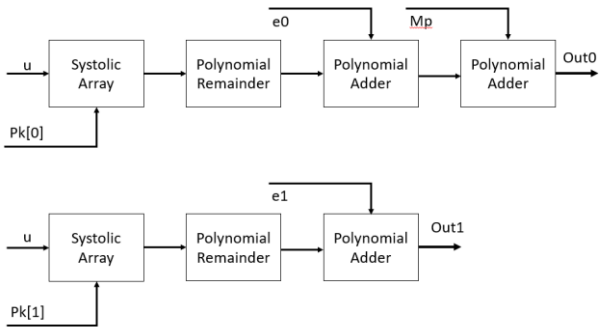


IMAGE VII Tep Level Block Encryptor

Seperti yang terlihat pada IMAGE VII, desain ini dapat digabungkan dan dikombinasikan untuk menyelesaikan sebuah komputasi dari enkripsi *fully homomorphic encryption* Brakerski/Fan-Vercaunteren ini. Design ini hanya membutuhkan n*2 – 1 clock cycle untuk dapat menyelesaikan seluruh proses komputasi enkripsi dari algoritma ini.

B. *Implemented Device*

Setelah mendesain sebuah blok yang akan melakukan sebuah enkripsi terhadap sebuah variable maka akan kita implementasikan algoritma yang kita desain pada sbeuah FPGA yaitu XILINK PYNQ Z1. Kami juga mengetest algoritma ini untuk melihat performance dari algoritma yang kita pilih menggunakan CPU. Hasil analisis performance itu akan dibandingkan dengan akselerator fpga yang kami buat.

Kami melakukan testing performance pada empat buah buah konfigurasi devais yang pertama adalah Rasberry pi 3B+, MSI Modern 15, PYNQ Z1 (CPU Only) dan PYNQ Z1 dengan akselerator FPGA. Spesifikasi dari alat yang kita gunakan adalah sebagai berikut.

1) Rasberry Pi 3B+

Kami melakukan performance test pada Rasberry Pi 3B+ dengan spesifikasi Bluethooth Broadcom BCM2837B0, dengan CPU adalah Cortex-A53 dengan clock CPU 1.4Ghz dan RAM 1Gb

2) MSI Modern 15

Kami melakukan testing performance pada MSI Modern 15 2020 dengan CPU Intel Core I5 10210U dengan Base Clock 1.6Ghz dan Boost Clock 4.2 Ghz dengan RAM 8Gb.

3) PYNQ Z1 (CPU Only)

Kami melakukan testing performance dari algoritma ini pada CPU PYNQ Z1 dengan spesifikasi CPU 650Mhz Dual Core A9 dengan RAM 512Mb.

4) PYNQ Z1 (CPU + FPGA Accelerator)

FPGA PYNQ Z1 juga kita testing performance dengan akselerator yang kita design yang sudah dijelaskan diatas yang memiliki 13300 Logic Slices, 6 LUT, 8 Flip Flop, 630 Kb Fast Block RAM dengan 4 Clock Management Tiles dan juga 220 DSP Slices.

Dari keempat devais tersebut dicari time execution untuk melakukan penyelesaian permasalahan komputasi mengenkripsi tersebut. Dari keempat devais ini juga dianalisis dna juga dilihat improvement dari accelerator yang dibuat terhadap PYNQ Z1.

IV. Experimental Results Encryptor

Untuk memperlihatkan konsep sistem yang sudah dijelaskan sebelumnya akan kita lihat performance test dari devais yang sudah kita sebutkan sebelumnya menggunakan skema Brakerski/Fan-Vercaunteren dengan algoritma Asymetrical Cryptografinya *fully homomorphic ncryption*. Hasil dari performance test ini akan dibandingkan dan juga dilihat kebisaan devais untuk melakukan enkripsi menggunakan algoritma ini.

A. *Encryption Parameters*

Untuk melakukan validasi yang fiar kita harus menyamakan parameter yang digunakan dalam algoritma ini. Parameter yang dipilih adalah sebagai berikut

- Nilai ciphertext modulus sebesar 32 bit.
- Nilai plaintext modulus sebesar 16 bit.
- Nilai Polynomial Modulus Degree 4, 8 dan juga 16

Akan dicoba proses enkripsi menggunakan variasi polynomial degree diatas untuk melakukan enkripsi 50 hingga 1000 buah data dalam tiap prosesnya dimana antara proses akan mengalami kenaikan sejumlah 50 data tiap prosesnya dengan nilai plaintext yang berbeda-beda. Akan dilihat performance dari tiap devais yang akan dilakukan pada testing ini.

B. *Experiment Results*

Performance test dilakukan dengan tiga buah variasi polynomial degree yang berbeda beda. Polynomial degree yang digunakan adalah 4, 8 dan juga 16. Akan dilihat juga perbedaan waktu eksekusi dengan devais yang berbeda-beda dari keempat buah devais yang dipilih.

1) Polynonial Degree 4

Kami melakukan percobaan untuk melakukan enkripsi n buah variable dengan polynomial degree 4. Percobaan ini dilakukan menggunakan nilai plaintext dan ciphertext modulus yang sama untuk melihat performance dari tiap devais untuk melakukan komputasi ini. Variable n buah kita modifikasi untuk melihat kinerja devais hingga throttling dan melihat apakah devais dapat melakukan enkripsi dengan stabil atau tidak. Selain itu juga nilai plaintext dari tiap run kami buat sama untuk melihat performance yang didapatkan.

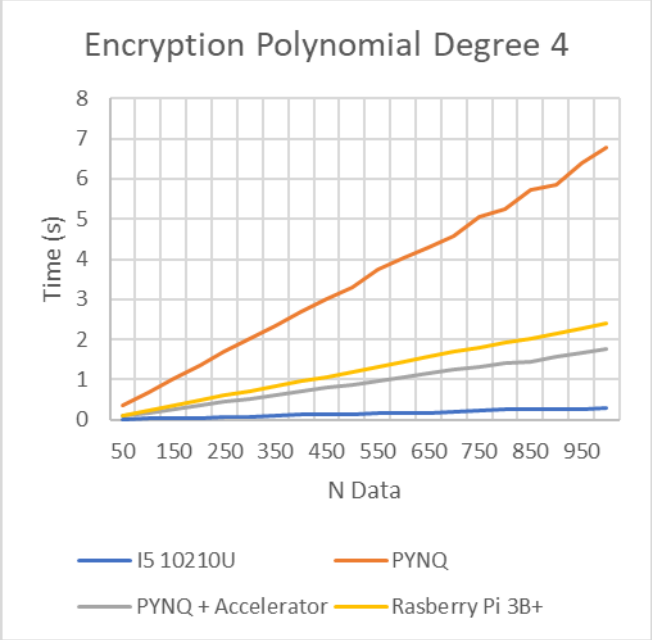


Fig. 1. Encryption Performance in Polynomial Degree 4

Dari Fig 1 dapat dilihat performance yang paling cepat dan handal untuk melakukan enkripsi ini adalah CPU Intel Core I5 10210U, kemudian disusul oleh CPU PYNQ dengan akselerator FPGA yang sudah kami desain. Kemudian disusul dengan Rasberry Pi 3B+ dan juga CPU PYNQ tanpa dibantu oleh akselerator FPGA. Hasil enkripsi rata-ratanya didapatkan waktu secepat,

- PYNQ Z1 CPU : 4.562 ms/data
- PYNQ Z1 CPU + FPGA : 1.2025 ms/data
- Intel Core I5 10210U CPU : 0.205 ms/data
- Rasberry Pi 3B+ : 1.639 ms/data

Dapat dilihat kenaikan performance dari PYNQ Z1 yang awalnya tanpa akselerator membutuhkan waktu 4.562 ms tiap datanya naik jauh hingga hanya membutuhkan waktu 1.2 ms tiap datanya. Performance ini juga jika kita tidak menggunakan python untuk melakukan pembacaan dan penulisan wire ke fpga hanya membutuhkan 0,028 us tiap datanya tetapi pembacaan dan penulisan dari CPU ke GPU PYNQ membutuhkan waktu yang sangat Panjang. Selain itu juga performance yang terbaik didapatkan oleh Inter Core I5 10210U sebesar 0.205 ms tiap datanya. Performance Rasberry Pi 3B+ untuk melakukan komputasi ini membutuhkan 1.639 ms tiap datanya.

2) Polynonial Degree 8

Kemudian kami juga melakukan percobaan yang sama dengan polynomial degree yang lebih tinggi yaitu 8. Dalam teorinya jika polynomial degreenya naik maka waktu yang dibutuhkan untuk memprosesnya juga akan naik secara eksponensial. Kami melakukan test performance seperti yang dilakukan sebelumnya dengan menggunakan keempat buah devais yang sudah kami tentukan. Kemudian performance yang didapatkan akan dilihat dan juga dianalisis hasil yang didapatkannya.

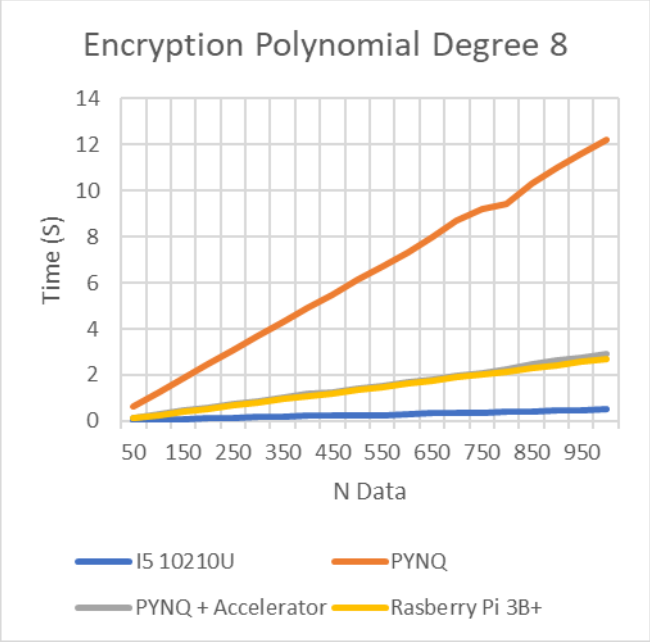


Fig. 2. Encryption Performance in Polynomial Degree 8

Dari Fig 2 dapat dilihat performance yang paling handal untuk melakukan komputasinya tetap intel core I5 10210U tetapi CPU PYNQ dengan akselerator juga mengalami akselerasi yang tinggi jika dibandingkan dengan tidak menggunakan akselerator. Raspberry pi juga mendapatkan performance yang tidak jauh dibandingkan dengan PYNQ CPU dengan akselerator FPGAnya. Kemudian hasl nkripsi rata-rata juga kami dapatkan secepat,

- PYNQ Z1 CPU : 8.328 ms/data
- PYNQ Z1 CPU + FPGA : 1.945 ms/data
- Intel Core I5 10210U CPU : 0.346 ms/data
- Rasberry Pi 3B+ : 1.84 ms/data

Dari data rata-rata yang dapat terlihat diatas performance upgrade hanya menggunakan CPU dan juga ditambahkan akselerator dari PYNQ Z1 semakin meningkat upgradenya tetapi performance yang didapatkan masih tidak jauh berbeda dari raspberry pi 3b+. PYNQ Z1 membutuhkan waktu 8.328 ms tiap datanya tetapi jika kita tambahkan akselerator hanya membutuhkan waktu 1.945 ms tiap data terenkripsinya. Kemudian raspberry pi 3b+ mendapatkan performance yang hamper sama dengan akselerator PYNQ yaitu secepat 1.84 ms tiap datanya kemudian performance yang paling baik tetap didapatkan oleh Intel Core I5 10210U CPU dengan kecepatan secepat 0.346 ms tiap datanya

3) Polynonial Degree 16

Kami juga melakukan percobaan dengan polynomial degree yang jauh lebih tinggi lagi yaitu dengan polynomial degree 16. Kami masih menggunakan spesifikasi yang ditentukan yang sama dengan devais yang sama tetapi dengan polynomial degree 16 kita dapat mengecek apakah data terenrkipsi tersebut jika menggunakan FPGA dengan akselerator apakah kecepatan write dan read cpu ke fpga dapat mengalahkan performance dari processor pynq atau tidak.

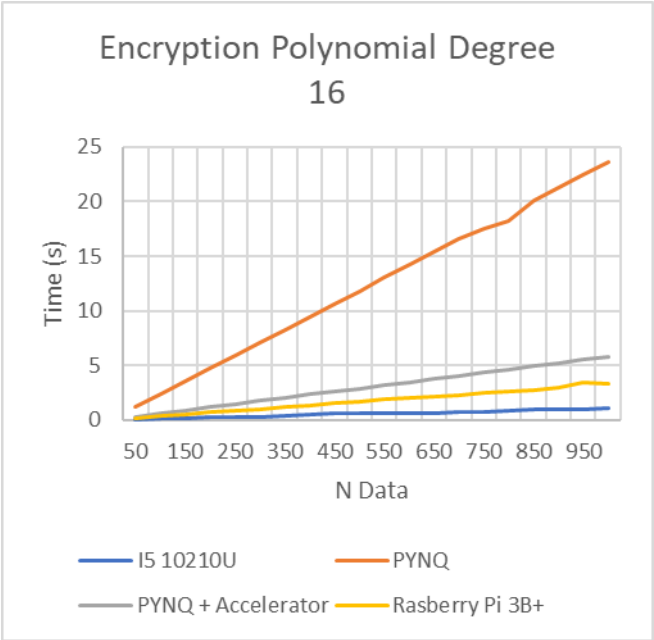


Fig. 3. Encryption Performance in Polynomial Degree 16

Dapat dilihat dari Fig 3 bahwa peningkatan yang didapatkan dari pengguaan PYNQ CPU saja dan ditambahkan dengan akselerator terjadi peningkatan yang signifikan. Tetapi performance upgrade ini tidak jauh berbeda dengan kecepatan komputasi yang didapatkan oleh Rasberry Pi 3B+ ini.

Performance paling tinggi tetap didapatkan oleh Intel Core I5 10210U. kemudian kita akan lihat rata-rata waktu proses dari tiap datanya sebagai berikut,

- PYNQ Z1 CPU : 16.1 ms/data
- PYNQ Z1 CPU + FPGA : 3.94 ms/data
- Intel Core I5 10210U CPU : 0.724 ms/data
- Rasberry Pi 3B+ : 2.28 ms/data

Dari hasil yang didapatkan diatas dapat dilihat bahwa PYNQ Z1 mengalami peningkatan upgrade performance dibandingkan dengan polynomial degree yang lebih kecil tetapi peningkatan performance ini masih belum cukup jika dibandingkan dengan Rasberry pi 3B+ untuk melakukan enkripsi. Tetapi performance dari raspberry pi 3b+ ini jika dibandingkan dengan PYNQ Z1 dengan akselerator juga tidak jauh berbeda. Performance terbaik masih didapatkan oleh intel core I5 dengan kecepatan 0.724 ms tiap datanya.

4) Summarization

Data kumulatif yang didapatkan jika performance dari ketiga variasi polynomial degree ini kita rata-ratakan akan mendapatkan hasil sebagai berikut,

- PYNQ Z1 CPU : 9.657 ms/data
- PYNQ Z1 CPU + FPGA : 2.364 ms/data
- Intel Core I5 10210U CPU : 0.425 ms/data
- Rasberry Pi 3B+ : 1.919 ms/data

Performance upgrade yang didapatkan oleh PYNQ jika ditambahkan dengan akselerator meningkat sebesar 4.1 kali dibandingkan dengan bertaruh pada CPUnya saja. Kecepatan upgrade mendapatkan performance 2.364 ms tiap datanya. Performance terbaik didapatkan oleh Intel Core I5 dengan kecepatan rata-ratanya secepat 0.425 ms tiap datanya dan raspberry Pi 3B+ membutuhkan 1.919 ms tiap datanya untuk melakukan komputasi enkripsi mengunakan *asymmetrical cryptography* dengan algoritman *Fully Homomorphic Encryption Brakerski/Fan-Vercaunteren scheme.*

V. Experimental Results Decryptor

Untuk memperlihatkan konsep sistem yang sudah dijelaskan sebelumnya akan dicoba diimplementasikan pada kelima buah devais yang sudah dipilih tersebut. Skema yang akan digunakan menggunakan skema BFV dengan algoritma kriptografi *fully homomorphic encryption*.

A. *Parameter yang akan di proses*

Untuk melakukan validasi yang fair kita harus menyamakan parameter yang digunakan dalam algoritma ini. Parameter yang akan dipilih adalah.

- Nilai ciphertext modulus sebesar 32 bit
- Nilai plaintext modulus sebesar 16 bit
- Nilai Polynomial Modulus Degree 4, 8 dan juga 16.

Akan dicoba proses dekripsi menggunakan variasi polynomial degree diatas dengan mencobanya melakukan dekripsi 50-1000 buah data dalam tiap prosesnya dimana tiap proses akan mengalami kenaikan buah data sebesar 50 data tiap prosesnya. Akan dilihat juga performance dari tiap devais yang akan dilakukan testing ini.

B. *Hasil dan Analisis*

Testing dilakukan menggunakan tiga buah variasi polynomial degree yaitu bernilai 4,8 dan juga 16. Akan dilihat perbedaan waktu eksekusi dengan devais yang berbeda beda dari ketiga buah variasi tersebut.

1) Polynomial Degree 4

Dilakukan percobaan mendekripsi n buah varible dengan polynomial degree 4. Percobaan ini dilakukan menggunakan nilai plaintext dan ciphertext modulus yang sama dengan tiap eksekusinya dilakukan dengan n buah variable yang berbeda beda.

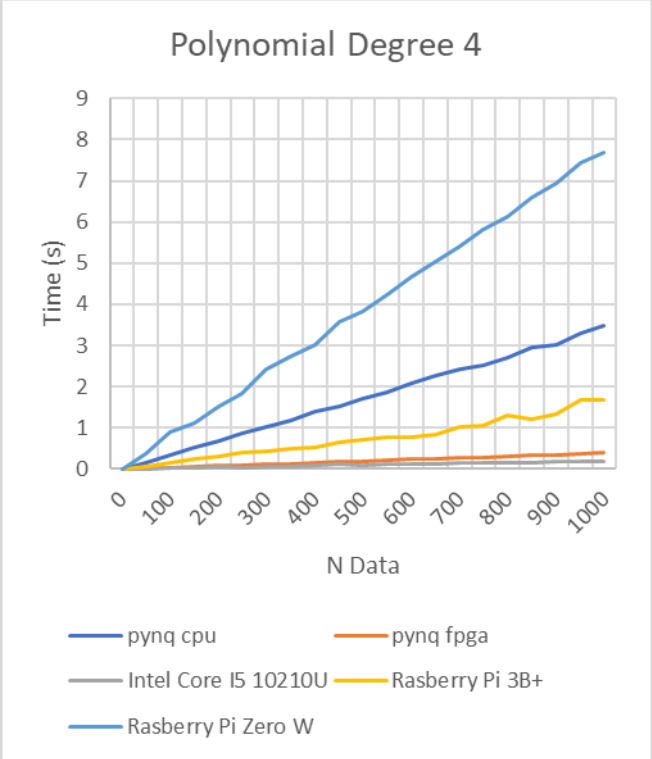


Fig. 1. Decryption Performance in Polynomial Degree 4

Dari Fig 1 dapat dilhat bahwa performance paling tinggi didapatan oleh CPU Intel Core I5 10210U, kemudian disusul oleh PYNQ Z1 dengan akselerator FPGA, Rasberry Pi 3B+, PYNQ CPU tanpa akselerator dan juga Rasbery Pi Zero W. hasil dekripsi rata-ratanya didapatkan waktu secepat,

- PYNQ Z1 CPU : 3.439 ms/data
- PYNQ Z1 FPGA : 0.387 ms/data
- Intel Core I5 10210U : 0.202 ms/data
- Rasberry Pi 3B+ : 1.482 ms/data
- Rasberry Pi Zero W : 7.774 ms/data

Dapat dilihat kenaikan performance dari FPGA PYNQ Z1 yang menggunakan cpu saja sangat kalah jauh dibandingkan dengan raspberry pi 3B+ tetapi saat menggunakan akselerator FPGA yang sudah di desain terjadi improvement performance sebesar 8.9 kali lipat. Performance yang didapatkan oleh PYNQ Z1 dengan akselerator dibandingkan dengan Rasberry Pi 3B+ memiliki improvement 3.83 kali lebih cepat.

2) Polynomial Degree 8

Kemudian dilakukan percobaan yang sama dengan polynomial degree lebih tinggi lagi yaitu menggunakan polynomial degree 8. Kemudian akan dilakukan test performance yang sama dengan percobaan sebelumnya.

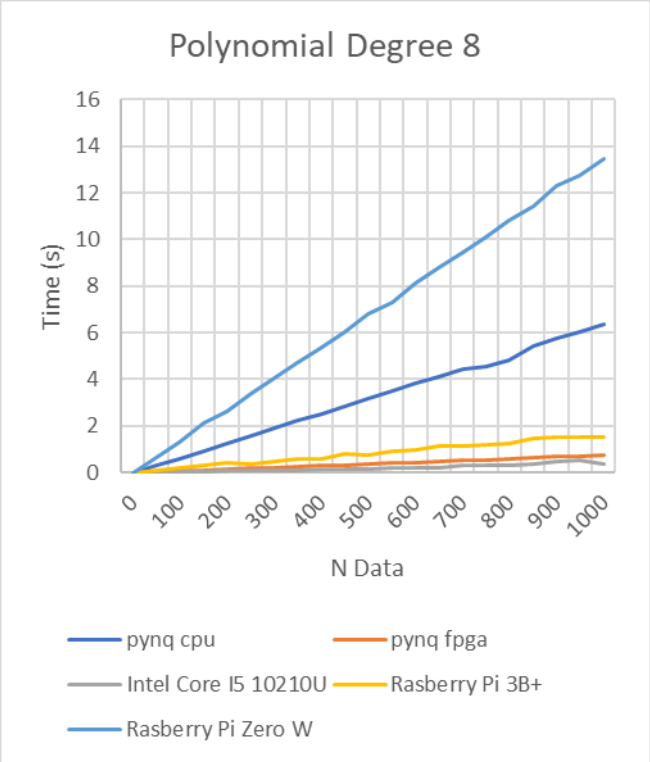


Fig. 2. Decryption Performance in Polynomial Degree 8

Dapat dilihat dari Fig 2, saat menggunakan polynomial degree 8 terjadi peningkatan performance dekripsi dari tiap devais, kenaikan waktu yang dibutuhkan dari tiap devais untuk melakukan proses dekripsi ini naik dua kali lipat dibandingkan dengan polynomial degree 4. Terlihat juga performance dari tiap devais memiliki selisih waktu yang lebih jauh lagi dibandingkan dengan polynomial degree yang lebih kecil.

Masih sama dengan polynomial degree 4 performance dimenangkan oleh processor Intel Core I5 10210U kemudian disusul dengan FPGA dengan akselerator, kemudian disusul oleh performance dari Rasberry Pi 3B+, PYNQ Z1 CPU tanpa akselerator dan yang terakhir adalah Rasberry Pi Zero W. hasil dekripsi rata-ratanya didapatkan waktu secepat,

- PYNQ Z1 CPU : 6.293 ms/data
- PYNQ Z1 FPGA : 0.744 ms/data
- Intel Core I5 10210U : 0.369 ms/data
- Rasberry Pi 3B+ : 1.705 ms/data
- Rasberry Pi Zero W : 13.435 ms/data

Dapat diluhat Kembali bahwa PYNQ Z1 terjadi akselerasi yang signifikan juga sebesar 8.45 kali lebih cepat dibandingkan hanya menggunakan CPU dan menjadi jauh lebih cepat dibandingkan dengan performance Rasberry Pi 3B+. Perbandingan kecepatan PYNQ Z1 dibandingkan Rasberry Pi 3B+ didapatkan akselerasi sebesar 2.29 kali lebih cepat.

3) Polynomial Degree 16

Kemudian dilakukan percobaan yang sama dengan polynomial degree lebih tinggi lagi yaitu menggunakan polynomial degree 16. Kemudian akan dilakukan test performance yang sama dengan percobaan sebelumnya.

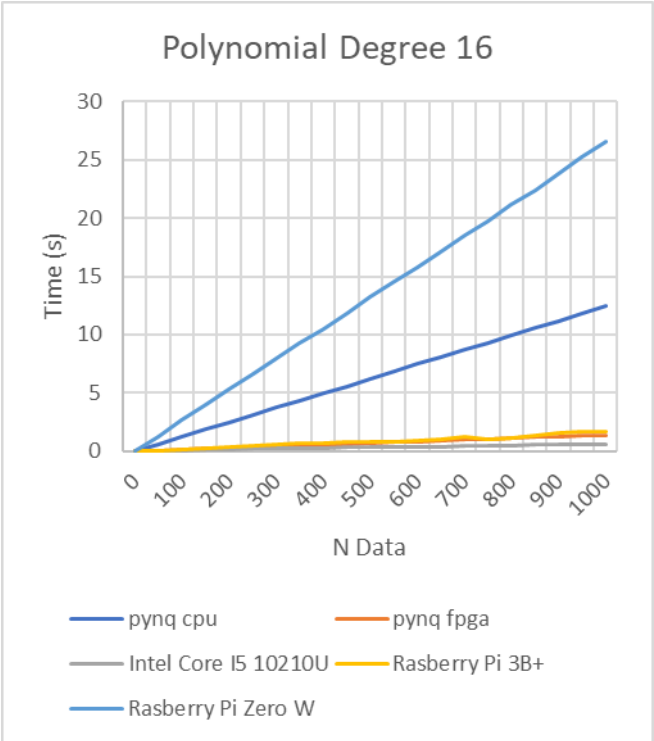


Fig. 3. Decryption Performance in Polynomial Degree 16

Dapat dilihat pada Fig 3, performance yang didapatkan antar devais perbandingannya tidak jauh berbeda jika dibandingkan dengan percobaan sebelumnya. Terjadi peningkatan kecepatan hamper 3 kali lipat dibandingkan dengan polynomial degree 4. Rata-rata performance dekripsi yang didapatkan sebagai berikut,

- PYNQ Z1 CPU : 12.427 ms/data
- PYNQ Z1 FPGA : 1.407 ms/data
- Intel Core I5 10210U : 0.701 ms/data
- Rasberry Pi 3B+ : 1.654 ms/data
- Rasberry Pi Zero W : 26.35 ms/data

Performance yang didapatkan hasil dari akselerator decryptor yang dijalankan di FPGA didapatkan hasil akselerasi sebesar 8.831 kali lebih cepat dibandingkan dengan

menggunakan CPU saja. Kemudian didapatkan juga performance 1.175 kali lebih cepat jika menggunakan PYNQ Z1 FPGA ini dibandingkan dengan menggunakan Rasberry Pi 3B+ untuk mengimplementasikan Dekripsi skema BFV ini.

4) Summarization

Data kumulatif yang didapatkan jika dirata-ratakan dari tiap polynomial degree yang berbeda ini didapatkan hasil,

- PYNQ Z1 CPU : 7.386 ms/data
- PYNQ Z1 FPGA : 0.846 ms/data
- Intel Core I5 10210U : 0.424 ms/data
- Rasberry Pi 3B+ : 1.614 ms/data
- Rasberry Pi Zero W : 15.853 ms/data

Terjadi akselerasi sebesar 8.731 kali pada PYNQ jika menggunakan akselerator yang sudah di desain dan juga jika menggunakan PYNQ Z1 FPGA terjadi akselerasi 1.9 kali lebih cepat dibandingkan dengan menggunakan Rasberry Pi 3B+ untuk melakukan dekripsi. Performance yang buruk didapatkan oleh Rasberry Pi Zero W yang sangat tidak cocok jika digunakan untuk menjadi sebuah alat untuk melakukan komputasi pada algoritma kriptografi ini

VI. CONCLUSSIONS

Untuk mengamankan data digunakan sebuah cara untuk mengamankan sebuah data yang dinamakan dengan kriptografi. Dalam kriptografi ini ada beberapa buah algoritma yang dapat digunakan untuk mengamankan data. Salah satu algoritmanya adalah algoritma kunci asimetrik. Pada algoritma tersebut ada beberapa skema yang dapat mempertahankan sifat homomorphic. Paper ini menjelaskan tentang performance yang didapatkan saat melakukan dekripsi menggunakan algoritma kriptografi kunci asimetrik dengan nama *fully homomorphic encryption* dengan skema BFV.

- **Decryptor**

Kami membuat sebuah desain akselerator yang dapat diimplementasikan pada FPGA PYNQ Z1 untuk melakukan sebuah proses dekripsi. Selain itu juga kami melakukan test performance untuk melihat performance komputasi dekripsi menggunakan raspberry pi 3B+, PYNQ Z1 tanpa akselerator, PYNQ Z1 dengan akselerator, Rasberry Pi Zero W dan juga Intel Core I5 10210U.

Hasil yang didapatkan dari melihat performance tersebut adalah intel Core I5 sangat handal untuk melakukan komputasi ini kemudian disusul oleh, PYNQ Z1 dengan akselerator yang sudah kami desain, Rasberry Pi 3B+, PYNQ Z1 tanpa akselerator dan yang terakhir adalah Rasberry Pi Zero W.

Kami melakukan test performance dengan niilai polynomial degree 4,8 dan juga 16, plaintext modulus 16 bit dan ciphertext modulus 32 bit. Hasil rata-rata performance yang kami dapatkan adalah PYNQ Z1 mendapatkan 7.386 ms/data, PYNQ Z1 + Akselerator mendapatkan 0.846 ms/data, Intel Core I5 10210U 0.424 ms/data, raspberry pi 3B+ secepat 1.614 ms/data dan yang terakhir adalah raspberry pi zero w dengan kecepatan 15.853 ms/data.

- **Encryptor**

Dari hasil yang kami dapatkan terjadi peningkatan performance yang signifikan antara melakukan komputasi menggunakan CPU saja dan juga melakukan komputasi algoritma dekripsi ini dengan bantuan FPGA. Peningkatan performance akselerator tersebut mendapatkan peningkatan 8.73 kali lebih cepat dibandingkan jika menggunakan CPU saja dan dengan bantuan akselerator ini FPGA Z1 cukup dapat berrsaing untuk melakukan komputasi dibandingkan dengan Rasbery Pi 3B+. Peningkatan Performance sangat besar dibandingkan hanya menggunakan CPU untuk menyelesaikan semua permasalahan komputasi ini. Terjadi improvement yang cukup besar jika menggunakan FPGA dan Rasberry Pi 3B+ untuk menyelesaikan permasalahan ini sebesar 1.9 kali lebih cepat.

Hasil yang kami dapatkan dari performance yang sudah dianalisis adalah Intel Core I5 10210U memiliki kecepatan komputasi yang paling cepat dibandingkan dengan devais yang lainnya. Kemudian PYNQ Z1 dengan akselerator dan juga Raspberry Pi 3B+ memiliki performance yang tidak jauh berbeda di beberapa polynomial degree dan performance PYNQ Z1 CPU saja yang sangat tidak handal untuk melakukan komputasi.

Kami melakukan test performance dengan polynomial degree 4, 8 dan juga 16. Plaintext modulus 16 bit dan juga ciphertext modulus sebesar 32 bit. Rata-rata performance yang didapatkan dari tiap polynomal degree dan juga tiap devais didapatkan PYNQ CPU membutuhkan 9.657 ms tiap datanya kemudian jika kita tambahkan dengan desain akselerator yang kita buat naik hingga 2.364 ms tiap datanya kemudian disusul oleh Raspberry Pi 3B+ dengan performance 1.919 ms tiap datanya dan juga Intel Core I5 10210U yang membutuhkan waktu 0.425 ms tiap data terenkripsinya.

Dari hasil tersebut kami mendapatkan terjadi peningkatan performance yang signifikan jika kita melakukan enkripsi pada CPU PYNQ Z1 saja dengan CPU PYNQ Z1 menggunakan akseleratornya. Performance yang didapatkan naik sebesar 4.1 kali lipat. Walaupun terjadi bottle neck dari pembacaan dan penulisan wire dari CPU PYNQ Z1 ke FPGAnya yang membutuhkan waktu yang sangat Panjang. Performance dari CPU + FPGA PYNQ Z1 masih dapat bersaing dengan CPU Rasberry Pi 3B+ yang dapat melakukan komputasi sedikit lebih cepat dibandingkan dengan akselerator yang kami desain tetapi performance yang terbaik masih didapatkan oleh CPU Intel Core I5 10210U dengan perbedaan performance 5.56 kali lebih cepat dibandingkan dengan akselerator PYNQ Z1. Terapi jika kita bandingkan dengan cost to performance yang didapatkan PYNQ Z1 unggul dibandingkan dengan Intel Core I5.

VII. REFERENSI

[1] Sanou dan Brahima, “ICT revolution and remaining graphic,” ICT Data and Statistics Division, Geneva, 2015.

[2] G. C. Kesslr, An Overview of Cryptography, Daytona Beach: University's Daytona Beach, 2020.

[3] B. Schmidt, M. Schimmler dan H. Schroeder, High-Speed Cryptography, Toronto: Fields Institute, 2006.

[4] D. Zheng, “The 15 Second Rule: 3 Reasons Why Users Leave a Website,” Thursday May 2020. [Online]. Available: https://www.crazyegg.com/blog/why-users-leave-a-website/. [Diakses 9 April 2021].

[5] T. Hamada, K. Benkrid, K. Nitadori dan M. Taiji, “A Comparative Study on ASIC, FPGAs, GPUs and General PurposeProcessors in the O(N^2) Gravitational N-body Simulation,” Nagasaki University's, Nagasaki, 2009.

[6] M. A. A. Afify, “Adaptive Hardware Cryptography Engine Based on FPGA,” Menoufiya University Faculty of Electronic Engineering Dept. of Computer Science and Engineering , Shibin el Kom, 2011.

[7] J. Folkens, Building a Gateway to the Internet of Things, Dallas: Texas Instruments, 2014.

[8] A. F, B. C, C. C, A. A, R. C.A. dan StrandM, A Guide to Fully Homomorphic Encryption, Trondheim: Norwegian University of Science and Technology Departement of Telematics, 2009.

[9] V. Vaikuntanathan, S. Halevy, C. Gentry dan M. Dijk, Fully Homomorphic Encryption over the Integers, New York: MIT and IBM, 2010.

[10] F. Fan dan F. Vercaunteren, Somewhat practically fully homomorphic encryption, Leuven-Heverlee: Katholieke Universiteit Leuven, COSIC & IBBT, 2012.

[11] O. Regev, On lattices, learning with errors, random linear codes and cryptography, STOC: ACM, 2005.

[12] F. Vercaunteren dan S. Nigel P., Fully Homomorphic Encryption with Relative Small Key and Ciphertext sizes, PKC: Springer, 2010.

[13] S. Halevi, Y. Polykov dan V. Shoup, An Improved RNS Variant of the BFVHomomorphic Encryption Scheme, New York: IBM Research, NJIT Cybersecurity Research Center, 2018.

[14] T. Hamada, K. Benkrid, K. Nitadori dan M. Taiji, A comparative study on ASIC, FPGAs, GPUs and general purpose processors in the O(N-2) gravitational N-body simulation, Nagasaki: University of Nagasaki, 2009.

[15] W. Luk dan D. B. Thomas, Comparing Performance and Energy Efficiency ofFPGAs and GPUs for High Productivity Computing, United Kingdom: Department of Computing, Imperial College LondonLondon, United Kingdom, 2010.

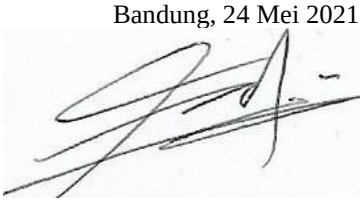
[16] O. Paldanius, Reference Cryptographic Accelerator, Helsinki: Helsinki Metropolia University of Applied Sciences, 2018.

[17] H. T. Kung, Systolic Architectures, which permit multiple computations for each memory access, can speed execution of compute-bound problems without increasing I/O requirements, Pittsburgh: IEEE, 1982.

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, 24 Mei 2021



Hamdani Fadhli