Primitives of Homomorphic Encryption Scheme: A Design Perspective

Project (EC57003) report submitted to

Indian Institute of Technology Kharagpur

in partial fulfilment for the award of the degree of

Master of Technology

in

Electronics and Electrical Communication Engineering

by

Sudarshan Sharma

(15EC32005)

Under the supervision of

Dr. Debdeep Mukhopadhyay and Dr. Indrajit Chakrabarti

Department of Electronics and Electrical Communication Engineering1

Indian Institute of Technology Kharagpur

Autumn Semester, 2019-20

November 28, 2019

DECLARATION

I certify that

(a) The work contained in this report has been done by me under the guidance of my supervisor.

(b) The work has not been submitted to any other Institute for any degree or diploma.

(c) I have conformed to the norms and guidelines given in the Ethical Code of Conduct of the Institute.

(d) Whenever I have used materials (data, theoretical analysis, figures, and text) from other sources, I have given due credit to them by citing them in the text of the thesis and giving their details in the references. Further, I have taken permission from the copyright owners of the sources, whenever necessary.

Date: November 28, 2019 (Sudarshan Sharma)

Place: Kharagpur (15EC32005)

CERTIFICATE

This is to certify that the project report entitled "**Primitives of Homomorphic Encryption Scheme: A Design Perspective**" submitted by **Sudarshan Sharma** (Roll No. 15EC32005) to Indian Institute of Technology Kharagpur towards partial fulfilment of requirements for the award of degree of Master of Technology in Electronics and Electrical Communication Engineering is a record of bona fide work carried out by him under my supervision and guidance during Autumn Semester, 2019-20.

Date: November 28, 2019

Place: Kharagpur

Dr. Debdeep Mukhopadhyay
Department of Computer Science and Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Date: November 28, 2019

Place: Kharagpur

Dr. Indrajit Chakrabarti
Department of Electronics and Electrical Communication Engineering
Indian Institute of Technology Kharagpur
Kharagpur - 721302, India

Abstract

Name of the student: **Sudarshan Sharma** Roll No: **15EC32005**

Degree for which submitted: **Master of Technology**

Department: **Department of Electronics and Electrical Communication Engineering**

Thesis title: **Primitives of Homomorphic Encryption Scheme: A Design Perspective**

Thesis supervisor: **Dr. Debdeep Mukhopadhyay and Dr. Indrajit Chakrabarti**

Month and year of thesis submission: **November 28, 2019**

After the emergence of cloud-based computation, Homomorphic Encryption Schemes has gained significant interest. This encryption scheme is based on Lattice-Based Hard Problems breaking those are considered difficult even the quantum computers till date, thus making them quantum secure. They also serve a suitable replacement of the existing Public Key Cryptography protocols like RSA and ECC, which are not considered quantum secure. The thesis presents the design perspective of the key elements required in the Homomorphic Encryption. The implementations are first designed using Verilog, extensively simulated for different corner cases, then tested on actual FPGA Hardware and at the end the outputs are also verified.

Acknowledgements

I would like to take the opportunity to thank my thesis advisors Prof. Debdeep Mukhopadhyay and Prof. Indrajit Chakrabarti for giving me an opportunity to work under their guidance. I am greatly indebted to them for their invaluable guidance and support.

Furthermore, I would like to thank Mr. Arnab Bag (PhD, Department of Computer Science and Engineering) and my parents who have consistently helped me to pursue research. Additionally, Friends and fellow researcher in the domain have all helped me a lot of times and made this journey memorable with out of the box ideas, insightful advices and funny jokes.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Motivation

These days cloud service plays a crucial role in the day to day life. There is a whopping increase in usage of Social Media, Internet Banking, Business solutions and numerous cloud services [Jul14]. The security and privacy over the cloud are one of the prime concern, as to get the desired computation, one needs to send data to the cloud server, and since the cloud cannot be trusted, the owner of the cloud can see, use and abuse the unencrypted data. Homomorphic Encryption (HE) is a cryptography protocol which can help to mitigate the privacy issue over the cloud without hampering the convenience and the benefits associated with the cloud. The computations processed on the cloud server are performed on encrypted data, and the result obtained in the form of encrypted data is forwarded to the user. The user can then decrypt the results through his.her private key. Some of the interesting applications of the HE are privacy-preserving services for information storage and processing in business and health-care applications [JWBN14], encrypted web-search engine [NPC13], electronic voting, and privacy-preserving prediction from consumption data in smart electricity meters [BCIV17], machine learning on encrypted data [GLN13] etc. Moreover, the HE schemes are based on learning with error hard problem which is till data proved to be quantum secure. Thus HE is a very suitable alternative for present public key cryptographic algorithms like RSA, Elliptic Curve Cryptography (ECC) etc.

1.2 Summary of the thesis

The thesis presently is organised into three chapters, Chapter 2 gives the introduction about the Homomorphic Encryption followed by description on elements of Learning with Error Problem. Chapter 3 presents the Random Bit Generator architecture design and tests. Chapter 4 explains about the Gaussian Sampler Algorithms and Architectures. Finally, conclusion and future goals are present in Chapter 5.

Chapter 2

Homomorphic Encryption

2.1 Homomorphic Encryption

The secure outsourcing of computation or in simple terms delegation of tasks to a cloud server through a secure protocol defines Homomorphic Encryption (HE) Schemes. The first step, to perform any operations, was to allow one type of operation. This is what we call partially (HE) whose most famous examples is the cryptosystem RSA [RSA78] two RSA ciphertexts can be multiplied together and yields an encryption of the product of their respective plain texts. Thus it is said that RSA is homomorphic for multiplication. The (HE) Schemes can be divided into two types, Somewhat Homomorphic Encryption (SHE) schemes which permits a number of operations known in advance and Fully Homomorphic Encryption (FHE) first step by [Gen09] , where no more constraints pertains to the computation. Thanks to the novel technique of bootstrapping which consists in homomorphically decrypting the ciphertext to refresh it, it becomes possible to evaluate any circuits on encrypted data which was difficult to implement when Gentry present the proof. However, after a series of optimisations and new Homomorphic Schemes implementations are gaining a lot of attention[Bon18]. In the subsequent years modulus switching was proposed by [Bra11] in 2011 which lower down the noise inherent in the ciphertext which increases in each operations and thus postpone the usage of the bootstrapping. The second generation of(HE) Scheme were BGV [ZBV12] and BV [Bra11], Gentry's scheme is considered as the first one. SHIELD [KGV16] is the only interesting third generation scheme. It involves matrices where second generation schemes

needed only vectors, therefore the minimal cost is higher, but due to the absence of relinearisation the cost increases far slower than for previous generations.

The actual hardware implementation requires additional building blocks to perform memory management, synchronization of parallel cores, and reliable interfacing with a host processor, etc [SSRV19]. This makes implementation of complex HE schemes in hardware very challenging. The key elements/block essential for the design are as follows:

2.1.1 Learning with Errors (LWE)

Preferably Ring Based Learning with Errors (RLWE). As the former is computationally efficient due to polynomial based compared to matrix based LWE. RLWE uses ideal lattices, Let Rq be the ring of polynomials. All coefficient of the polynomial reduced to mod q. According to RLWE: samples of the form (a, a · s + e), it is difficult to determine the secret polynomial $s \in \mathbb{R}_{ll}$, where the polynomial $a \in \mathbb{R}_{ll}$ is sampled uniformly at random and the coefficients of the error polynomial e are small samples from the error distribution ξ. [BUC19] The detailed mathematical concepts and hardness of the scheme is proved in the thesis [Ber16]. The hardware and software architecture of the lattice based cryptography has been thoroughly survey in [NDR+19]. The RLWE consists of the following elements considering the design perspective.

2.1.1.1 Discrete Gaussian Sampler

The Gaussian sampler is used to generate the error polynomial which is used in the RLWE scheme. Presently, there exists many versions of distribution samplers for the RLWE scheme but only the distribution changes which means the probabilities although the algorithm remains the same as in the sampler used in TFHE[CGGI19].

2.1.1.2 Modular Arithmetic core

The polynomial arithmetic unit is used for polynomial addition, subtraction and multiplication. Polynomial addition and subtraction can be performed in O(n) time

simply by performing coefficient-wise additions or subtractions modulo q. Computation of polynomial multiplication is the costliest operation in the ring-LWE-based cryptographic schemes.

2.1.1.3 Number Theoretic Transform

The Number Theoretic Transform (NTT) corresponds to an Finite Fourier Transform (FFT) and is considered better than FFT because FFT and inverse-FFT perform arithmetic using real numbers and thus suffer from approximation errors which is not desired in cryptographic applications. In NTT the roots of unity are taken from a finite ring Z_q. Hence all computations in NTT are performed on integers. The NTT exists if and only if n divides $d1$ for every prime divisor d of q. The multiplication of the polynomial is node in R_q. There exists different iterative and optimised designs for the NTT [LN16].

Chapter 3

Random Bit Generators

3.1 Random Bit Generators

Random Numbers Generator are the core of the Cryptography Applications, it refers to many applications such as simulation, numerical analysis, machine learning etc.The paper [MBO18] presents a survey on the hardware implementation of Random Number Generator on FPGA. Periodicity and deterministic outputs that use an operator or arithmetic functions are the main difference with the past generators. They are known in literature as "pseudorandom" or "quasirandom" number generators (PRNGs), while circuits that use a physical source to produce randomness are called "true" random number generators (TRNGs). The entropy sources present in these kinds of TRNGs are either the electronic noise of embedded components or some environmental sensors (temperature, noise, etc). The different kinds of TRNGs present in the literature are as follows:

- Phase Locked Loop (PLL) TRNGs. [FD03]
- Ring Oscillator TRNGs [MBO18] [BYV18]
- Self-Timed Ring TRNG [CFAF13]
- Metastability TRNG [VHKK08]

Considering the ease in integration, reliability, area utilisation and the throughput of the TRNGs Ring Oscillator based TRNG' have a very high speed and a higher and

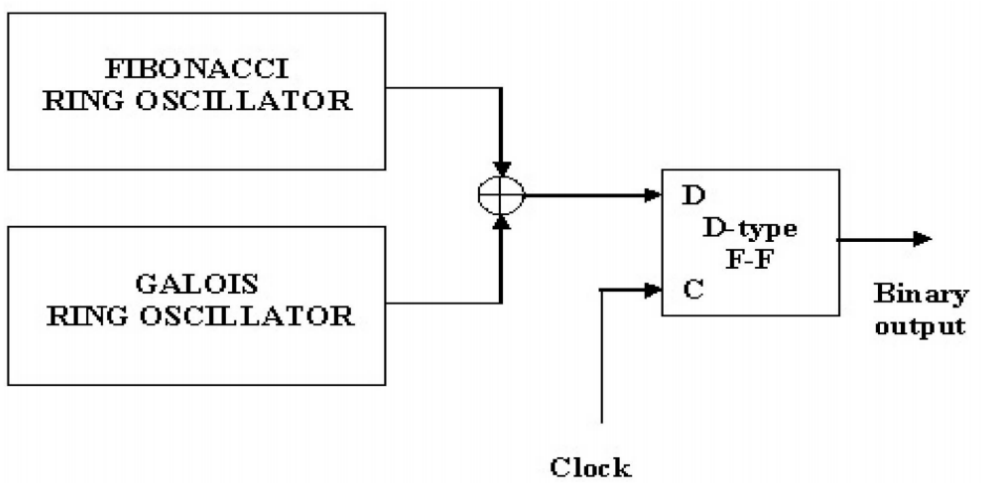

FIGURE 3.1: Random Binary Sequence Generator. *Source: [Gol06]*

more robust entropy rate in comparison with previous proposals for digital random number generators [Gol06].

3.1.1 Golic's Ring Oscillator RNG

First to use new methods and the corresponding robust techniques for generating high speed and high entropy raw binary sequence by using only logic gates in digital semiconductor technology. Then for the purely RNG output, additional post-processing of raw binary sequence that provide randomness extraction and computational security as well as increase in speed if required. The proposed method is independent of the fabrication technology.

3.1.1.1 Random Binary Sequence Generation

The combination of the Fibonacci and Galois Ring Oscillators XORed together followed by the D Flip Flop clocked by a ring oscillator as shown in the Figure 3.1 . In the Fibonacci Ring Oscillator, the inverters are used instead of the delay elements. The feedback connections are chosen so as to satisfy the basic condition that there are no fixed points in the corresponding state-transition function. As seen from the Figure 3.2 it consists of r inverters connected in cascade, The feedback connections are specified by the binary co-efficient of $f_i's$ if $f_i = 0$ the switch is open

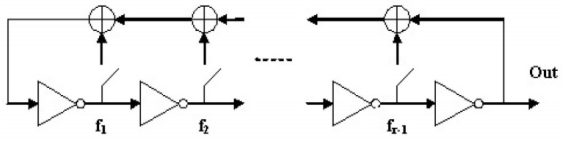

FIGURE 3.2: Fibonacci Oscillator. *Source: [Gol06]*

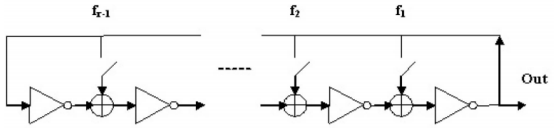

FIGURE 3.3: Galois Oscillator. *Source: [Gol06]*

is 1 it is in the closed configuration. The binary coefficients $f_i's$ are obtained from the co-efficient of feedback polynomial f(x). A proof is provided in the paper to get the essence of how the feedback polynomial is chosen. For the implementation purpose, the polynomial provided in the experiment section is used directly. The Galois ring oscillator also consists of r inverters in cascade with the $f_i's$ as the switches similar to the Fibonacci ring oscillator configuration. The randomness, as well as the robustness, is increased by XOR-ing the outputs of the two oscillators. The lengths of the two oscillators minus one should preferably be mutually prime, first, for the period of the corresponding pseudo random sequence to be maximized and, second, for the interlocking or coupling effect to be minimized. Since the oscillating signal is then a sort of mixed digital and analog noise, further randomness due to meta stability may be induced within a sampling unit, e.g., implemented as a D-type flip-flop. Its clock signal is generated by an independent ring oscillator with a lower or similar frequency as the oscillating signal.

3.1.1.2 The Post Processing Circuit

Once the design for generating the raw binary sequence is done there is some sort of bias still present which is removed using the post-processing circuit (Figure 3.4) present in cascade with the binary sequence generator. It is a self clock-controlled Linear Shift Feedback Register (LSFR) in Galois configuration similar to the oscillator above with the only difference being the delay elements as flip flops instead of the inverters. The output is formed by XOR-ing the Output of the XOR gate connected to the output of the γth flipflop, where $f_{-1} = 1$. The output of the 2 input

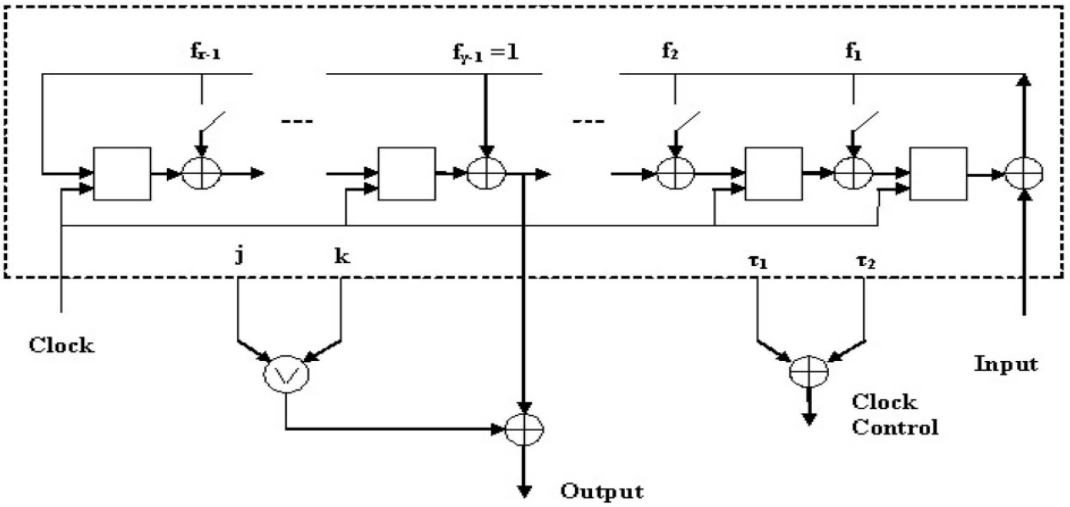

FIGURE 3.4: The Post Processing Circuit. *Source: [Gol06]*

OR gate applied to the outputs of the j^{th} and k^{th} flipflops. The clock control signal is binary, so each time the output bit is produced, the value of the clock control bit decides the number of times the LSFR should be clocked 1 or 2. The clock control is implemented in hardware by a counting logic and another D-type flip-flop by using its clock enable input. The clock control bit is generated by XORing the outputs of the 1th and 2th flip flops as shown in the Figure 3.4.

3.1.1.3 Hardware Implementation of Golic's TRNG Scheme

It is evident from the description of the circuit that the algorithm contains a lot of constants which needs to be calculated before the actual implementation. The number of the inverters in the fibonacci ring oscillator=20 in the Galois ring oscillator=21. The length of the LSFR taken as 64, the non zero co-efficient of the primitive polynomials are 64, 4, 3, 1, 0. The constants $\gamma 1 = 4$, $\gamma = 5$, j=20, k=25, $\tau 1 = 3$, $\tau 2 = 2$, $\delta 1 = 3$ and $\delta 1 = 5$. The hardware design was done on Vivado 2019.1 using the primitive FPGA Fabrics as the synthesis tool always optimised the ring oscillator redundant circuits even after using constraints. The design details is as follows:

The FPGA used was Artix 7 Digillent FPGA (xC7A35TICSG324-1L).

TABLE 3.1: Golic's RNG Implementation Details

Site Type	Used	Fixed	Available	Util%
Slice LUTs	21	0	20800	0.10
LUT as Logic	19	0	20800	0.09
LUT as Memory	2	0	9600	0.02
LUT as Shift Reg	2	0		
Slice Register	32	0	41600	0.08
Register as Flip Flop	32	0	41600	0.08

3.1.1.4 Issues with Golic's RNG Scheme

A greater number of constants involved. The selection of the primitive polynomial, the security is affected by the number of non-zero feedback coefficient. It is difficult to get the primitive polynomial of higher order. Absence of master clock in the design Only the long chain of oscillator act as clock for the output. After testing the design, It can be concluded that the paper is quite old so maybe the LUT-LUT delay would have been greater compared to the present counterparts.

3.1.2 ES-TRNG

[BYV18] presents a lightweight, AIS-31 compliant TRNG with conservative entropy estimation. Compared to the existing TRNG architectures this work an additional emphasis is done on the stochastic model based formal security analysis.Edge Sampling, variable-precision phase encoding and repetitive sampling are used. Edge Sampling is an algorithm in which we select N edges at random from a graph. Nodes connected to those edges are present in the output network. Variable-precision phase encoding is the encoding method in which Only the oscillators phase around the single edges are sampled with higher precision. Repetitive Sampling means the repetition of the sampling at higher frequency until the high precision phase region of the oscillator is captured. Proposed scheme has a throughput of 1.15 Mbps with 0.997 bits of Shannon entropy, only 10 LUT and 5 flip-flops are used on Spartan-6 FPGA (fairly lightweight). The design criterion considered are resource consumption, throughput, latency, feasibility, design efforts, resistance against attacks and a stochastic model to prove unpredictability and inner test ability of the entropy source. Most of the existing TRNS's are tested with [RV01] NIST Test, it is said that for the NIST tests the statistical evaluation of the output is bad because a

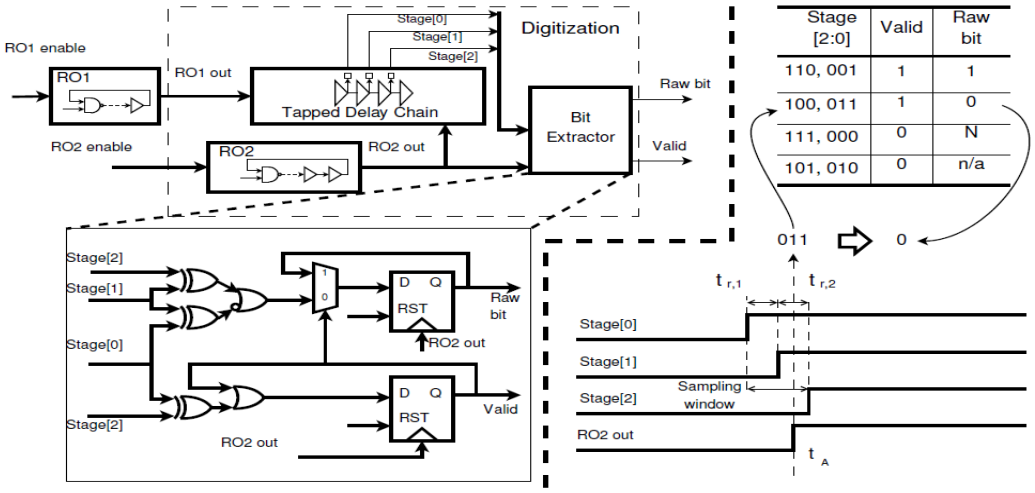

FIGURE 3.5: The Digital Noise Source Circuit. *Source: [BYV18]*

completely deterministic pseudo-random sequence could pass the test despite having no randomness. Thus, modern approaches like the BSI AIS-31 uses a stochastical model to evaluate the unpredictability.The unpredictability of the TRNG depends on some physical processes here timing phase jitter in a free running ring oscillator is leveraged to generate a digital noise source. Raw Random Numbers obtained directly from the digital noise source suffer from statistical defects bias, ideal probability of ones auto-correlation between output bits. Post Processing helps to mitigate the above-mentioned error, algorithmic post-processing extracts entropy to from raw random numbers to increase entropy per bit. Furthermore, cryptographic post-processing helps in backtracking resistance [Yan18]. The Post Process sing is performed by XORing the three consecutive raw bit generated from the digital noise source.

3.1.2.1 Digital Noise Source

The entropy source denoted by RO1is implemented as a free-running ring oscillator with an enable signal. The average period of RO1is denoted by T01. The output signal of RO1 propagates through the Digitization module.The digitization module consists of three main components, namely, a tapped delay chain, a sampling free-running ring oscillator RO2 and a bit extractor. The average period of RO2 is denoted by T02. The used tapped delay chain consists of four cascaded delay elements, the first and last one acting providing isolation for the middle two delay

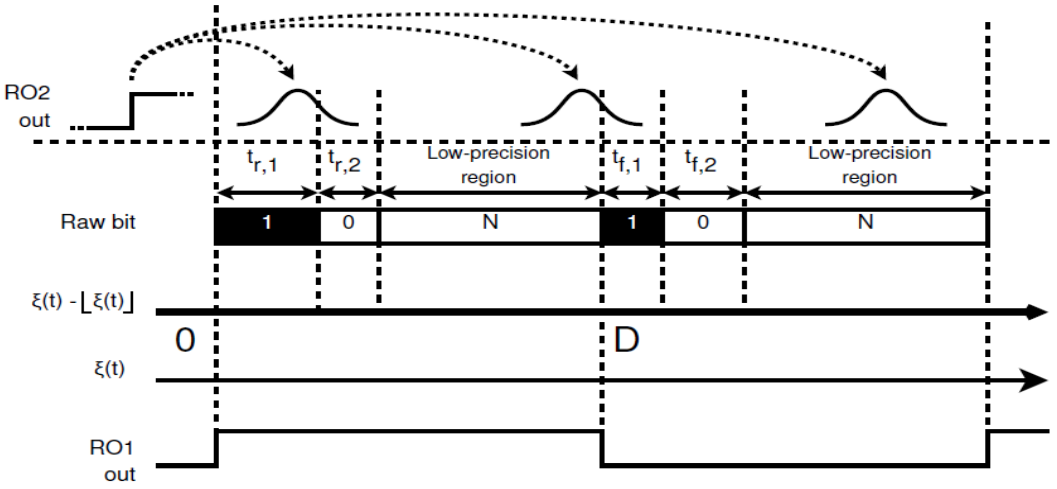

FIGURE 3.6: Variable Precision Encoding Scheme. *Source: [BYV18]*

chains. The output from the tapped delay Stage [2:0] chain is processed by the combinatorial circuit of the bit extractor which encodes these values to a single raw bit which is the output of the digital noise source and a strobe signal Valid which is responsible to disable the ROs for the new set of random bit generation.

3.1.2.2 Variable Precision Encoding

This technique Figure 3.6 is enabled by using both the tapped delay chain and the bit extractor. The oscillators phase around the single edge is sampled with higher precision and the remaining region's phase with lower precision. The position of the captured edge is encoded into a raw bit. This result in compact implementation, shorter jitter accumulation time and entropy is reduced since the low precision is used.

3.1.2.3 Repetitive Sampling

Repetitive sampling Figure 3.7 is synchronized to the high frequency signal RO2 out, aiming to reduce the time needed to hit the high-precision region, thereby improving the throughput. Once the high-precision region is hit, a Valid signal is generated. This helps in improving the throughput. The jitter accumulation time tA is chosen to allow accumulating enough jitter at the entropy source. This sampling multiple

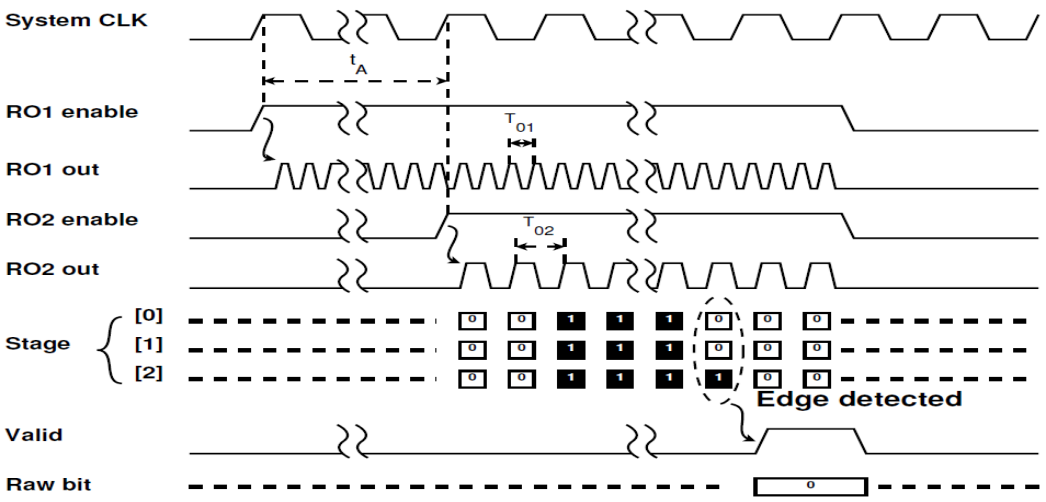

FIGURE 3.7: Waveform for Repetitive Sampling. *Source: [BYV18]*

times within a single system clock cycle, because the frequency of a free-running RO2 is higher than the system clock's.

3.1.2.4 Hardware Implementation of ES-TRNG

Since, the TRNG is very compact in terms of the implementation it can be easily implemented using the FPGA Fabrics. The author presents the design on Spartan 6 with the Hardware Architecture as seen in Figure 3.8. The tapped delay chain implemented using the carry chain and flip flops, the input of the flip flop comes from the XOR gate of the carry chain. The tapped delay chain as in Figure 3.9, the τ represent the delay due to the multiplexer and the XOR gate delay in the carry chain.

TABLE 3.2: ES-TRNG Implementation Details

Site Type	Used	Fixed	Available	Util%
Slice LUTs	9	0	20800	0.04
LUT as Logic	9	0	20800	0.04
LUT as Memory	0	0	9600	0.00
Slice Register	5	0	41600	0.01
Register as Flip Flop	5	0	41600	0.01

The FPGA used was Artix 7 Digillent FPGA (xC7A35TICSG324-1L) and the tool as Vivado 2019.1.

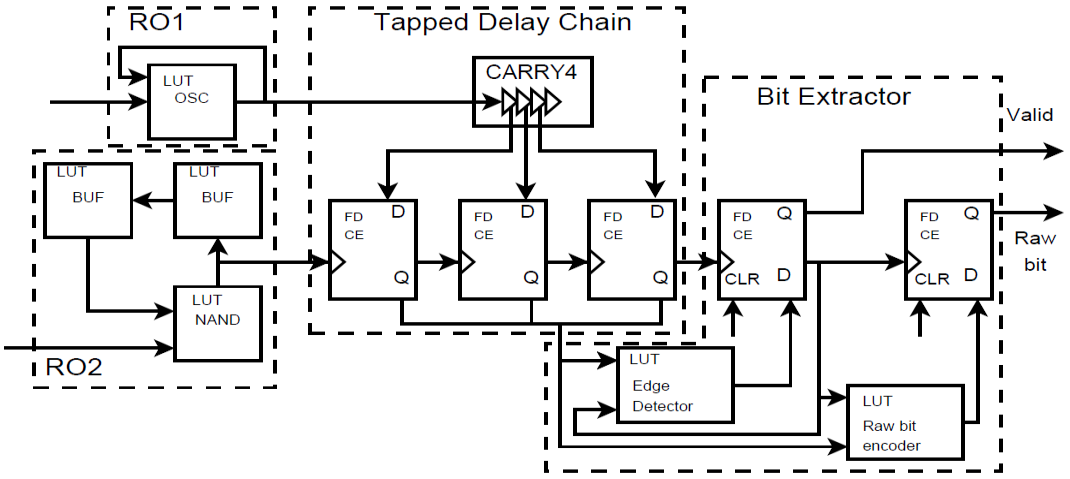

FIGURE 3.8: FPGA Architecture *Source: [BYV18]*

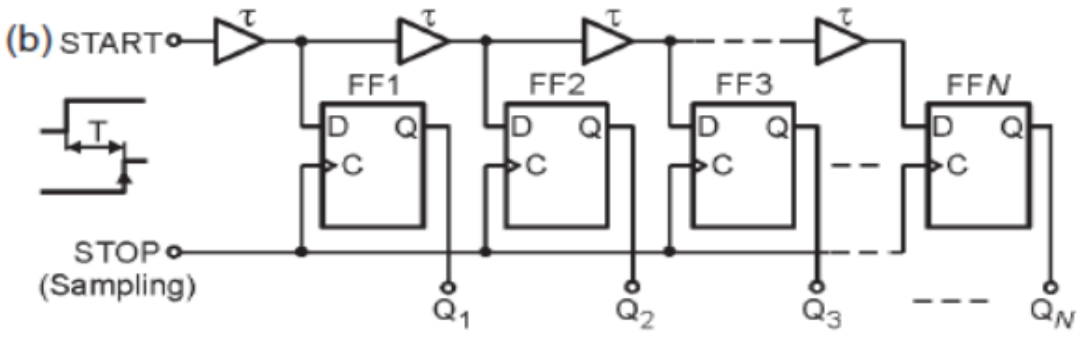

FIGURE 3.9: Tapped Delay Chain

Chapter 4

Discrete Gaussian Sampler

4.1 Discrete Gaussian Sampler

The continuous Gaussian distribution with the mean $c \in \mathbb{R}$ and standard deviation $\sigma > 0$ is defined as follows. Given that E is a random variable on $\in \mathbb{R}$, then for $x \in \mathbb{R}$.

$$Pr(E = x) = \frac{e^{-(x-c)^2/2\sigma^2}}{\sigma\sqrt{2\pi}} \tag{4.1}$$

The discrete Gaussian distribution on $\in \mathbb{Z}$ with standard deviation $\sigma > 0$ and mean 0 is defined as follows. Given E as a random variable in $\in \mathbb{Z}$

$$Pr(E = z) = \frac{e^{-(z)^2/2\sigma^2}}{S} \tag{4.2}$$

where S is termed as the normalisation factor approximated as $\sigma\sqrt{2\pi}$. The value of the parameter S can be calculated by summing the probability from $-\infty$ to ∞. The constant S is used to define the distribution as if S is know the standard deviation can be calculated through the approximate relation mentioned.

$$S = 1 + 2\sum_{z=1}^{\infty} e^{-(z)^2/2\sigma^2} \tag{4.3}$$

The sampled discrete Gaussian distribution accuracy is measured using the statistical distance. The statistical distance between the theoretical discrete distribution and the sampled distribution is kept as small as possible to satisfy the security proof of the cryptography protocols [Fol15].

4.1.1 Existing Methods

Since, It is known that the tail bound of the Gaussian distribution is infinitely long, none of the sampling algorithms can over the entire tail bound. So, sampling is performed up to a tail bound. This tail bound is the responsible for the statistical distribution. The tail bound is calculated according to the maximum statistical distribution allowed by the security parameter. Thus, the sampling methods needs either a large number of floating point operations or stored large pre-computed tables. Due to this reason the implementation of these samplers become difficult on constrained memory devices without high precision floating points arithmetic. [Roy17]

The detailed survey on the various method of sampling discrete distribution is summarised in [DG14]. The two major sampling methods are rejection sampling and the inversion sampling method. In Rejection Sampling a points is generated randomly and then those points that are inside the distribution are accepted. This method is slow for Gaussian (High rejection rate) also many random bits required due to high rejection rate for sampled values near the tail. The rejection sampling method doesn't use pre-computed tables. However, because of the high rejection rate the latency is hindered. A modified version of this algorithm along with significant speedup and lower precision is presented by [DN12]. Moreover, In inversion method a uniform random variable U in the interval [0, 1] is obtained and the output Z is computed according to the inequality. where p_i is Pr(Z=i).

$$F(Z-1) = \sum_{i<Z} p_i < U \leq \sum_{i<Z+1} p_i = F(Z) \tag{4.4}$$

[Pei10] proposed to use this method to sample discrete Gaussians. He used precomputed tables to store the F(Z) values and the inequality was solved performing a binary search in the table. This method is fast and does not require high precision floating point arithmetic, but it also requires the presence of precomputed tables.

These tables can be relatively large, depending on the Gaussian parameter and the tailcut, and we also need multiple instances of them: one for each different parameter combination. When this algorithm is implemented in a hardware due to the summation involved the comparator size increases. Also, the random bits required from the uniform distribution needs to have a high precision.

4.1.2 Knuth Yao Algorithm

The Knuth Yao Algorithm [KY76] uses a random walk method of the Discrete Distribution Generating (DDG) to perform sampling using the probabilities of the elements in the sample space.

4.1.2.1 Features

The number of random bits required is very close to entropy of distribution almost near optimal. Knuth Yao proposed a random walk model for sampling from any non-uniform distribution. The algorithm can be implemented through a series of optimisations and is one of the state of art Gaussian sampler implementation on constrained devices.

4.1.2.2 Method

The first implementation of the algorithm was given by [SRVV14]. The method is fairly simple. The sample set say contains m elements and the i element is sampled with the probability of p_i. So, convert the probability in binary and represent in the form of a DDG tree. The tree is traversed according to a random bit starting from the root, when the next node is leaf node the algorithm terminates and the sampled value is given as output.

4.1.2.3 DDG Tree

Discrete Distribution Generating Trees is a rooted binary tree. There are two types of nodes in the DDG tree.

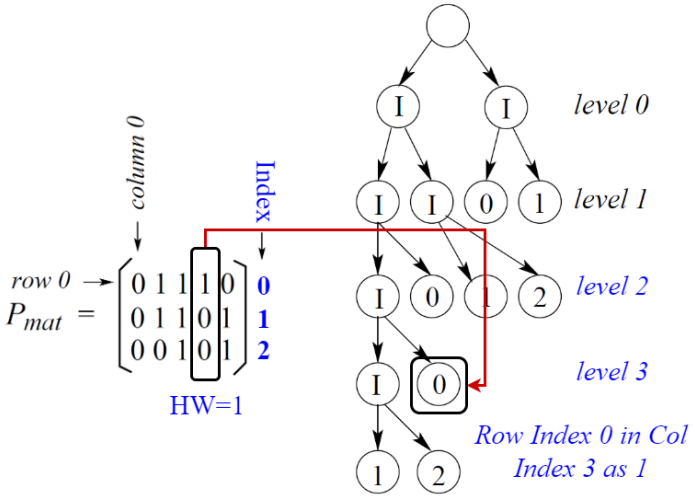

FIGURE 4.1: DDG Tree creation using the P_{mat}. *Modified from Source: [SRVV14]*

- Intermediate Node
- Terminal Nodes

The intermediate nodes are responsible for the creation of next set of levels in the DDG tree. It is obtained from the probability matrix of the sample set, since the length of the sample set is m, the length of the number of rows in the probability matrix would be m. The number of column say p of the probability matrix say P_mat depends on the floating point precision. The i^{th} sample element will have the probability p_i.

Property The number of terminal nodes at a level say i in a DDG tree is given by the hamming weight of the i^{th} column of the probability matrix.

4.1.2.4 DDG Traversal

4.1.2.5 DDG Creation

As described in the Figure 4.1, the root node of the DDG splits into two intermediate nodes as the first column of the P_{mat} will always be 0 since the binary representation of probabilities will always be of the perform $0.b_{-1}b_{-2}.....b{-}p$ where p is the number of columns in P_{mat}. The first level of the DDG is constructed using the 1^{st} column of the P_{mat}. In the Figure 4.1 for the level 3, consider the 3^{rd} column, the hamming

weight of the 3^{rd} is 1. So, the only terminal node in this level would be the row index 1 which is written on the right node of level 3. The terminal node are marked with the row index where 1 is present in the selected column vector. It is important to node that the the nodes at a level are filled in the right to left fashion, the row index are search for 1's in the selected column vector in bottom up fashion which means first the last row index is checked for 1 if found the index is noted in the rightmost node of the selected level.

The traversal starts from the root node, a random bit is required to predict the next node. If the random bit is say 0 the left node is selected and vice-versa. The sampling operation is terminated when it hits a terminal node, the terminal node gives the row index of the sample in the P_{mat}.

4.1.2.6 Optimisation in DDG Traversal

In the hardware, it is very difficult to create the entire DDG tree due to large memory overhead. The $(i+1)^{th}$ level of the DDG tree can be created from the $(i)^{th}$ level. According to the algorithm the i^{th} level of the DDG needs only the hamming weight of the i^{th} column and the algorithm terminates once it hits the terminal nodes. The Figure 4.2 explains the traversal of the DDG tree, it can be observed that the visited node is a terminal node in the i^{th} level. Now, since hamming weight of i^{th} level is known, the number of intermediate nodes is known as (Terminal + Intermediate = Total Nodes in the Level). The visited node k is d nodes to the right of the terminal node n, d=n-k. Now, we just need to find whether we hit the terminal node or intermediate node in the $(i+1)^{th}$ level the number of terminal nodes would be on the right side which will be decided by the $(i+1)^{th}$ column's hamming weight. Depending on the random bit and the value of 2d the decision can be inferred.

4.1.2.7 Optimisation in P_{mat} Storage

The probability matrix P_{mat} can be stored efficiently, the storage depends on both on tail bound and on precision. Data is stored in the RAM of the hardware, Data fetching from the RAM increases the computation time which effects the performance of the algorithm. As P_{mat} is accessed column wise, store P_{mat} columns in ROM Words. It can be observed that near the bottom of the column large number

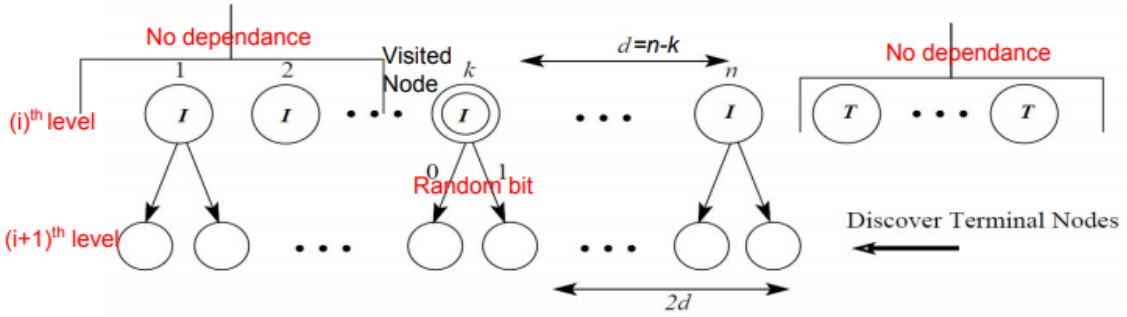

FIGURE 4.2: Optimisation of DDG Traversal. *Modified from Source: [SRVV14]*

of zeros are present in the P_{mat}. The column length cab be used to compress the zeros, it is the length of the top portion after which the chunk of bottom zeros start. The differential column length i.e for the next column store the difference in the column length of the present and next column. The Figure 4.4 depicts the P_{mat} conversion to compressed ROM words, here underscore is used to separate the two columns, also the values 0 and 1 not included in the rectangle are the differential column length 0 means same column length and 1 means increment column length by 1. If the sum of all the probabilities for a certain tail bound is not equal to 1. It is unsure whether the Knuth Yao Algorithm will hit a terminal node. To mitigate this issue, use one extra sample point valued as (1-P_{sum}) which if hit is discarded and the probability of such event is very very low. Hence, this sample point is regarded as out of range event

4.1.3 Hardware Implementation of Knuth Yao Algorithm

Based on the algorithm discussed in the previous sections the main components involved in the hardware architecture are as follows:

- ROM stores the probability matrix (compressed) the probabilities are accessed through ROM counter Initially the counter is cleared and incremented one by one to fetch data from higher ROM locations.
- Data from the ROM is stored in a SCAN register, which is a left shift register A counter is used to count the number of bits scanned from the ROM when this value reaches the WORD SIZE of the ROM, ROM counter is incremented and data is fetched from the ROM.

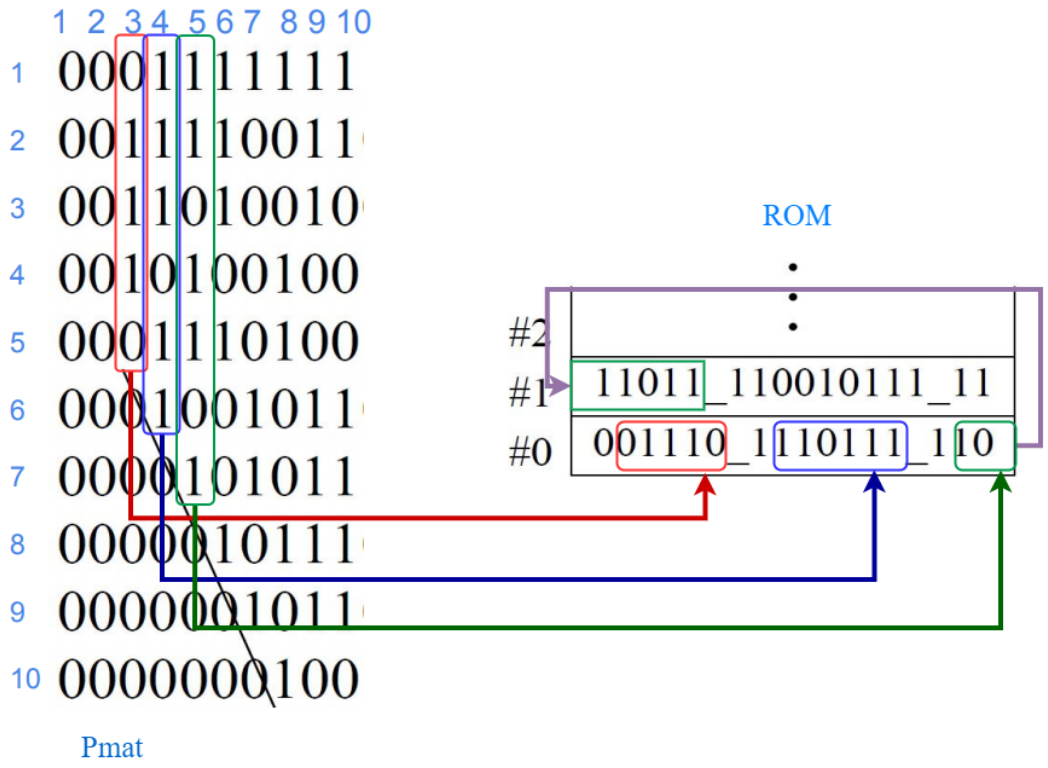

FIGURE 4.3: Optimisation of Matrix Storage. *Modified from Source: [SRVV14]*

- A Random Bit generator is required to generate the random bit for DDG traversal. Random bit generator based on the approach of [Gol06] is use in the Paper True or pseudo random generators can be used On an average only 5 bits are required, thus the random bit generator can be slow.

- Upcounter column length stores the column lengths of different columns of P_{mat}. Initialised to first non-zero length of P_{mat}. During the Random walk the counter remains as it is or increase by 1

- A Row Down counter is used to count the number of rows in the P_{mat}, during column scanning i.e HW. At the start this value is set to the Column length and then decremented one by one to reach zero.

- During the random walk the "d" is stored in the distance register. It is updated to 2d or 2d+1 depending on the random bit. Later each detection of terminal node i.e the 1 in the column decrements this distance register by 1. Any moment when d¡0. The terminal node is returned as the sample output and the sampling process terminates.

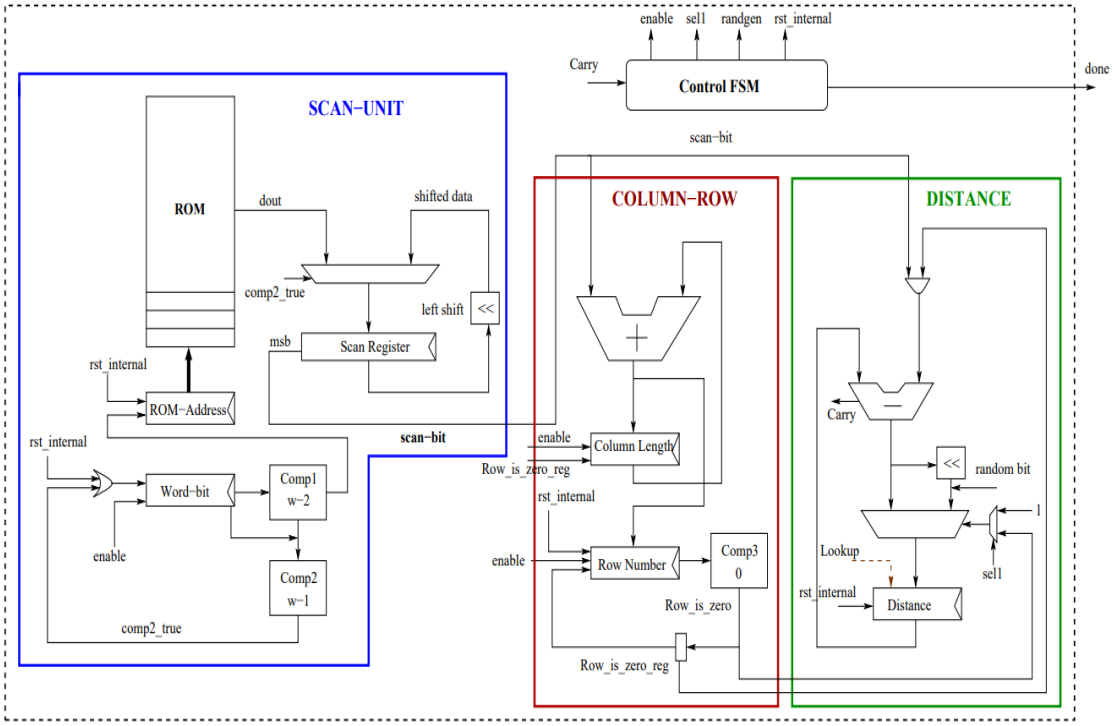

FIGURE 4.4: Hardware Implementation of Knuth Yao Algorithm. *Source: [SSRV14]*

- A Finite State Machines (FSM) Circuit is used to control the various multiplexers, counters present the architecture.

Chapter 5

Conclusion

5.1 Conclusion

The number of optimised Homomorphic Encryption scheme is increasing at a faster rate, thus sticking to primitives and key components of a better alternative. As the primitives can be tweaked and different scheme can therefore be generated. The hardware implementation of the primitives comprises of FPGA Fabrics, and this gives the most optimised implementation of any design on FPGAs. The design tool used for the simulations, programming hardware and verification are Vivado 2019.1 and Xilinx ISE. Diligent Basys 3 FPGA boards with Xilinx Artix 7 are used for testing.

The Random Bit Generators, especially the TRNG's are based on the physical noise sources for the FPGA implementation the noise source is mainly the ring oscillator jitter. The implementation of the TRNGs can be further optimised by choosing the locations (co-ordinates) of the LUTs present in the FPGA, and different typologies can be studied affecting the throughput and the randomness of the raw bit. The post-processing circuit is used to remove the bais form the random raw bit generator from the noise source.

The Gaussian Sampler Architecture is implemented using the novel Knuth Yao Algorithm which is heavily optimised by the authors. This opens the room for the Side-Channel vulnerability. Presently, constant time implementation and certain masking schemes are used along with the standard implementation of Gaussian

Sampler. It could be interesting to look for any optimisation which can be exploited either through side channel or fault analysis.

5.2 Future Goals

The upcoming tasks and goals are ennumerated as follows:

- Verification through test suit of the logged RNG data implemented on FPGA.
- Integration of different modules present in the Gaussian Sampler.
- Gaussian Sampler testing on FPGA followed by test suit based verification.
- Analyse the optimisations in the Gaussian Sampler to check for Side Channel vulnerability.
- Implementation of Modular Arithmetic and Number Theoretic Transforms.
- Integration of modules to create a fully homomorphic encryption scheme.

Bibliography

[BCIV17] Joppe W. Bos, Wouter Castryck, Ilia Iliashenko, and Frederik Vercauteren. Privacy-friendly forecasting for the smart grid using homomorphic encryption and the group method of data handling. In Marc Joye and Abderrahmane Nitaj, editors, *Progress in Cryptology - AFRICACRYPT 2017*, pages 184–201, Cham, 2017. Springer International Publishing.

[Ber16] Federico Bergami. *Lattice-Based Cryptography*. PhD thesis, Universita di Padova and Universite de Bordeaux, 7 2016.

[Bon18] Guillaume Bonnoron. *A journey towards practical fully homomorphic encryption*. PhD thesis, Universite Bretange Loire Mathstic, 5 2018.

[Bra11] Vaikuntanathan V. Brakerski, Z. Efficient fully homomorphic encryption from (standard) lwe. *FOCS, pp. 97–106*, 2011.

[BUC19] Utsav Banerjee, Tenzin S. Ukyab, and Anantha P. Chandrakasan. Sapphire: A configurable crypto-processor for post-quantum lattice-based protocols. *IACR Transactions on Cryptographic Hardware and Embedded Systems*, 2019(4):17–61, Aug. 2019.

[BYV18] M. Grujic N. Mentens B. Yang, V. Rozic and I. Verbauwhede. Es-trng: A high-throughput, low-area true random number generator based on edge sampling. *IACR Transactions on Cryptographic Hardware and Embedded Systems, vol. 2018, no. 3, pp. 267–292*, 2018.

[CFAF13] A. Cherkaoui, V. Fischer, A. Aubert, and L. Fesquet. A self-timed ring based true random number generator. In *2013 IEEE 19th International Symposium on Asynchronous Circuits and Systems*, pages 99–106, May 2013.

[CGGI19] Ilaria Chillotti, Nicolas Gama, Mariya Georgieva, and Malika Izabachène. Tfhe: Fast fully homomorphic encryption over the torus. *Journal of Cryptology*, Apr 2019.

[DG14] Nagarjun C. Dwarakanath and Steven D. Galbraith. Sampling from discrete gaussians for lattice-based cryptography on a constrained device. *Applicable Algebra in Engineering, Communication and Computing*, 25(3):159–180, Jun 2014.

[DN12] Léo Ducas and Phong Q. Nguyen. Faster gaussian lattice sampling using lazy floating-point arithmetic. In Xiaoyun Wang and Kazue Sako, editors, *Advances in Cryptology – ASIACRYPT 2012*, pages 415–432, Berlin, Heidelberg, 2012. Springer Berlin Heidelberg.

[FD03] Viktor Fischer and Miloš Drutarovský. True random number generator embedded in reconfigurable hardware. In Burton S. Kaliski, çetin K. Koç, and Christof Paar, editors, *Cryptographic Hardware and Embedded Systems - CHES 2002*, pages 415–430, Berlin, Heidelberg, 2003. Springer Berlin Heidelberg.

[Fol15] J. Follath. Gaussian sampling in lattice based cryptography. *Tatra Mountains Mathematical Publications*, 2015.

[Gen09] Craig Gentry. *A FULLY HOMOMORPHIC ENCRYPTION SCHEME*. PhD thesis, STANFORD UNIVERSITY, 11 2009.

[GLN13] Thore Graepel, Kristin Lauter, and Michael Naehrig. Ml confidential: Machine learning on encrypted data. In Taekyoung Kwon, Mun-Kyu Lee, and Daesung Kwon, editors, *Information Security and Cryptology – ICISC 2012*, pages 1–21, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg.

[Gol06] J. D. J. Golic. New methods for digital generation and postprocessing of random data. *IEEE Transactions on Computers*, 55(10):1217–1229, Oct 2006.

[Jul14] Sundararajan E. Othman Z. Jula, A. Cloud computing service composition: A systematic literature review. expert systems with applications. *41(8), 3809–3824. https://doi.org/10.1016/j.eswa.2013.12.017*, 2014.

[JWBN14] K. Lauter J. W. Bos and M. Naehrig. Private predictive analysis on encrypted medical data. *Journal of Biomedical Informatics*, 2014.

[KGV16] A. Khedr, G. Gulak, and V. Vaikuntanathan. Shield: Scalable homomorphic implementation of encrypted data-classifiers. *IEEE Transactions on Computers*, 65(9):2848–2858, Sep. 2016.

[KY76] D. Knuth and A. Yao. *Algorithms and Complexity: New Directions and Recent Results*, chapter The complexity of nonuniform random number generation. Academic Press, 1976.

[LN16] Patrick Longa and Michael Naehrig. Speeding up the number theoretic transform for faster ideal lattice-based cryptography speeding up the number theoretic transform for faster ideal lattice-based cryptography. *IACR Cryptology ePrint Archive report 2016/504*, 2016.

[MBO18] J.-F. Couchot M. Bakiri, C. Guyeux and A. K. Oudjida. Survey on hardware implementation of random number generators on fpga: Theory and experimental analyses. *Comput. Sci. Rev., vol. 27, pp. 135–153*, 2018.

[NDR+19] Hamid Nejatollahi, Nikil Dutt, Sandip Ray, Francesco Regazzoni, Indranil Banerjee, and Rosario Cammarota. Post-quantum lattice-based cryptography implementations: A survey. *ACM Comput. Surv.*, 51(6):129:1–129:41, January 2019.

[NPC13] K. Qin N. Peng, G. Luo and A. Chen. Query-biased preview over outsourced and encrypted data. *Scientific World Journal*, 2013.

[Pei10] Chris Peikert. An efficient and parallel gaussian sampler for lattices. In Tal Rabin, editor, *Advances in Cryptology – CRYPTO 2010*, pages 80–97, Berlin, Heidelberg, 2010. Springer Berlin Heidelberg.

[Roy17] Sujoy Sinha Roy. *Public Key Cryptography on Hardware Platforms: Design and Analysis of Elliptic Curve and Lattice-based Cryptoprocessors*. PhD thesis, KU Leuven, 5 2017.

[RSA78] R. L. Rivest, A. Shamir, and L. Adleman. A method for obtaining digital signatures and public-key cryptosystems. *Commun. ACM*, 21(2):120–126, February 1978.

[RV01] SOTO J. NECHVATAL J. SMID M. BARKER E. LEIGH S. LEVENSON M. VANGEL M. BANKS D. HECKERT A. DRAY J. RUKHIN, A. and S VO. A statistical test suite for random and pseudorandom number generators for cryptographic applications, 2001.

[SRVV14] Sujoy Sinha Roy, Frederik Vercauteren, and Ingrid Verbauwhede. High precision discrete gaussian sampling on fpgas. In *Revised Selected Papers on Selected Areas in Cryptography – SAC 2013 - Volume 8282*, pages 383–401, Berlin, Heidelberg, 2014. Springer-Verlag.

[SSRV14] F. Vercauteren S. S. Roy, O. Reparaz and I. Verbauwhede. Compact and side channel resistant discrete gaussian sampling. *Cryptology ePrint Archive, Report 2014/591*, 2014.

[SSRV19] K. Jarvinen F. Vercauteren S. S. Roy, F. Turan and I. Verbauwhede. Fpga-based high-performance parallel architecture for homomorphic computing on encrypted data. *Cryptol. ePrint Arch., Tech. Rep. 2019/160*, 2019.

[VHKK08] Ihor Vasyltsov, Eduard Hambardzumyan, Young-Sik Kim, and Bohdan Karpinskyy. Fast digital trng based on metastable ring oscillator. In Elisabeth Oswald and Pankaj Rohatgi, editors, *Cryptographic Hardware and Embedded Systems – CHES 2008*, pages 164–180, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg.

[Yan18] Bohan Yang. *True Random Number Generators for FPGAs*. PhD thesis, KU Leuven, 9 2018.

[ZBV12] C. Gentry Z. Brakerski and V. Vaikuntanathan. (leveled) fully homomorphic encryption without bootstrapping. *ITCS, S. Goldwasser, ed., ACM, New York*, 2012.