

Received November 18, 2021, accepted December 17, 2021, date of publication December 31, 2021, date of current version January 10, 2022.

Digital Object Identifier 10.1109/ACCESS.2021.3139925

Using Genetic Algorithm in Inner Product to Resist Modular Exponentiation From Higher Order DPA Attacks

HRIDOY JYOTI MAHANTA¹, KESHAB NATH², (Member, IEEE), AMIT KUMAR ROY³, KETAN KOTECHA⁴, AND VIJAYAKUMAR VARADARANJAN⁵

¹Department of Computer Science and Engineering, Assam Don Bosco University, Guwahati 781017, India

²Department of Computer Science and Engineering, Indian Institute of Information Technology, Kottayam 686635, India

³Department of Computer Science and Engineering, National Institute of Technology Mizoram, Aizawl 796012, India

⁴Symbiosis Centre for Applied Artificial Intelligence, Mulshi 412115, India

⁵School of Computer Science and Engineering, University of New South Wales, Sydney, NSW 2052, Australia

Corresponding authors: Ketan Kotecha (head@scaai.siu.edu.in) and Keshab Nath (keshabnath@iitkottayam.ac.in)

This work was supported by Symbiosis International University.

ABSTRACT Evolutionary computation techniques have always provided fascinating results in all the fields of science and engineering. However in the area of computer security, their contribution has been comparatively very less. More specifically if we consider the side-channel attacks, use of these nature based techniques have been very nominal. Therefore, we proposed a secure protocol in this paper to combat against the Higher Order Differential Power Analysis attacks on modular exponentiation based cryptosystems using one of the popular evolutionary computation techniques. The proposed work first uses Genetic Algorithm for splitting the huge exponent within the modular exponentiation into multiple non-uniform shares. Then, this shares are randomly chosen for computing individual modular exponentiation with the help of nearest neighbor algorithm. Using Genetic Algorithm, our proposed protocol can generate reasonable number of shares which exposes secret exponent at the least. As a result, it provides significant resistance to Higher Order Differential Power Analysis attacks. Moreover, randomization in computing individual modular exponentiation secures the cryptosystem from generic power analysis attacks like SPA and DPA.

INDEX TERMS Power analysis attacks, modular exponentiation, genetic algorithm, nearest neighbor algorithm, mutual information analysis.

I. INTRODUCTION

Among all the side channel attacks, power analysis attacks [1] have been a major threat to modular exponentiation within RSA cryptosystems [2]. Messerges *et al.* [3] were the first to show the possibilities of mounting Simple Power Analysis (SPA) and Differential Power Analysis (DPA) attacks on “squaring-multiplication” based implementation of modular exponentiation. Where SPA could easily identify the location of each operation by visualizing the spikes in power traces, DPA used statistical models to extract the secret key or exponent.

One approach to combat against the power analysis attacks is to remove the power consumption dependency on data and operations of the cryptosystems. This can be achieved by randomizing the secret data or fundamental operations

The associate editor coordinating the review of this manuscript and approving it for publication was Junaid Arshad ^{id}.

of the cryptographic algorithms. This technique is popularly known as hiding. It aims to mitigate SPA and DPA by minimizing the correlation between power consumption and secret data. Binary Random Initial Point (BRIP) introduced by Mamiya *et al.* [4] was designed using randomization within RSA to combat against power analysis attacks. BRIP was later extended and improved by Kim *et al.* [5], [6] and Wang *et al.* [7]. Recently, Mahanta *et al.* have presented two works to resist power analysis attacks based on data randomization [8] and operation randomization [9].

Another resisting approach is by injecting dummy instructions between the fundamental operations of the cryptosystems for uniform power consumption. But it costs computational penalty due to which it is used along with masking where, the secret intermediate values are associated or masked with some unknown random values. Fournaris *et al.* [10] and Ambrose *et al.* [22] used this approach in their works to combat against power analysis

attacks from RSA. Masking in asymmetric cryptosystems is called blinding. Binding in RSA based modular exponentiation cryptosystems, can be implemented along with the message and exponent but it needed additional multiplicative inverse operation to remove the effect of message blinding. However, the message blinding technique presented by Kim *et al.* [11] was one of those kinds which did not require any multiplicative inverse.

There appears significant works in literature which have used the above approaches to combat against the power analysis attacks. However, all these works aimed to protect cryptosystems from generic SPA and DPA attacks only. Higher order DPA (HODPA) and Correlation power analysis (CPA) attacks which have been major threats in present scenario are yet to be addressed. In order to protect a cryptosystems from HODPA and CPA, it is necessary that the secret sensitive data and their related intermediate data are split into n shares which would result into protection from n^{th} order DPA attacks [35]. Recently, Balash *et al.* [12], [13], and [36] presented a state-of-art of using inner products for masking in symmetric cryptosystems to defend HODPA. Based on Dziembowski *et al.* [14] model, they proposed a leakage resilient design, in which the shares leaked minimum information of the secret data in presence of noise. However, in their approach these shares were only random numbers which were generated in multiple iterations without any defined way.

A. OUR CONTRIBUTION

In this paper, for splitting the secret exponent in modular exponentiation into multiple shares, we have adopt the concept of inner product. Genetic Algorithm has been used in inner product (IP-GA) to form limited the number of shares of the exponent as well as make these shares diverse in each execution. To facilitate randomization, an entropy based algorithm known as nearest neighbor (E-NN) has also been proposed which randomly chooses a share among all the shares to compute individual modular exponentiation. The secure modular exponentiation along with IP-GA and E-NN is then implemented in decryption RSA and CRT-RSA of different key sizes. To measure the possible leakage from each share, mutual information analysis with the original exponent has been computed. Also, a security analysis of the proposed protocol to combat popular DPA and HODPA attacks has been presented. Conventional countermeasures like message and exponent blinding alone have severely failed to resist advanced DPA attacks like horizontal correlation analysis [45], [46] and template attacks [47], [48]. While we probe our proposed secured model to horizontal attacks and template attacks, we could find that these attacks can be resisted to significant extent. Hence, the role of our proposed protocol to combat these modern and future attacks is very crucial.

The remaining section of the paper is discussed as follows: a brief survey on related works has been provided in Section 2. Section 3 presents some preliminary concepts

related to the proposed work. In Section 4, detail discussion of the proposed protocol with implementation in RSA and CRT-RSA is presented. The security analysis of the proposed protocol with respect to DPA, HODPA and some advanced attacks have been discussed in section 5. The analysis through the results have been presented in section 6 and finally a conclusion has been formulated.

II. RELATED WORKS

Hiding aimed to randomize the order of execution of fundamental operations of the cryptosystems. As a result, through SPA it would not be possible to identify the actual sample points of the operations. Hence, if the modular exponentiation was implemented in binary methods with hiding, extraction of the correct key by locating the squaring and multiplication operations would be extremely difficult. The BRIP method [4], which was mainly designed for Elliptic Curve Cryptography (ECC), introduced a random initial point for every execution. As a result, the initial values for execution would be different at every point of execution providing resistance to DPA attacks. Kim *et al.* [5] proposed secured binary exponentiation for RSA and CRT-RSA by randomizing the message and key with help of a random number r . The effect of r was removed by two different ways of computing r^{-1} depending on whether the bit is 0 or 1. They also improved their work using random blinding [6] which was more secured and had lesser computational cost than Mamiya's. The improved method was a generalized concept that could be used for RSA as well as ECC. However, Wang *et al.* [7] challenged the works of Kim showing that their improved countermeasure could still reveal sensitive information. They modified Kim's approach and proposed to compute r^{-1} at four different points based on the all the possible combinations of 0 and 1 i.e 00, 01, 10 and 11. Recently, Mahanta *et al.* have proposed two different countermeasures for modular exponentiation based cryptosystems. The first countermeasure [8] computed comparative modular exponentiation with completely randomized exponent or key for RSA and CRT-RSA decryption to resist DPA to a large extent. In their second work [9] they executed operation randomization using Fisher Yates Shuffling technique to attain significant security against DPA attacks.

Unlike hiding, blinding associated the secret data (say x) with a blinding factor (λ) in such a way that it turns impossible to retrieve x if λ is not known. For RSA cryptosystems in which the primary operation is modular exponentiation ($m^e \bmod N$), both the message (m) and exponent (e) can be blinded. For message blinding, the blinding factor is associated in such a way that,

$$m' = m \times \lambda \bmod (N) \quad (1)$$

Similarly, the exponent can be blinded with the blinding factor and $\phi(N)$ by,

$$e' = e + \lambda \times \phi(N) \quad (2)$$

But merely using any of these techniques or both were still not sufficient enough to protect RSA and CRT-RSA from DPA attacks as seen in [15]–[19]. Besides it would also have additional computational cost of multiplicative inverse to remove effects of blinding factor. However, a secured and efficient way to compute message blinding without multiplicative inverse was presented by Kim *et al.* [11]. To make blinding more efficient, random instructions were also injected in between the fundamental operations [20]–[22]. In some other approaches, blinding with improved modular exponentiation were also proposed [23], [24]. However, none of these works addressed HODPA and CPA leaving a threat of being vulnerable to such attacks.

III. PRELIMINARIES OF PROPOSED WORK

Here, we discuss few preliminary concepts which are later employed in our proposed protocol. This section helps in better anticipating our proposed protocol which is presented in detail in section 4.

A. INNER PRODUCT

As mentioned in previous section, in order to protect a cryptosystem from HODPA the sensitive data (key and intermediate values) needs to be split into multiple shares. Thus, a cryptosystems will be protected from n^{th} order DPA if sensitive data is split into n shares. The significance of using inner product (IP) is its ability to represent a number (say X) as a summation of the products of random pairs. Each element of these pairs will belong to two vectors, say L and R such that, $L = l_1, l_2, \dots, l_n$ and $R = r_1, r_2, \dots, r_n$ containing n random values. The inner product represented by $\langle L, R \rangle$ can be computed by $\sum_{i=1}^n l_i \times r_i$. This approach was demonstrated in [14] which was later used in [12], [13], [36] in identifying a masking scheme for the Advanced Encryption Standard (AES) cryptosystem.

In asymmetric cryptosystems like RSA, the decryption key (d) is extremely sensitive and mostly very large. In order to protect the cryptosystems from HODPA attacks, splitting of d would be one of the most effective measures. We have used IP as an approach to split of d into multiple shares. However, instead of using it conventionally we have associated Genetic Algorithm (GA) for choosing random values to form the IP pairs (l, r). Using GA, the proposed approach could generate reasonable number shares which would leaked minimum mutual information of d . Hence, even if we consider that these shares have leakage of power consumption, but the information provided by them will be very less for successful power analysis attacks. Therefore, in our proposed protocol, we considered the following assumptions for computing inner product(a) vector R would store $n-2$ independent 64 bit random variables (b) the last, $(n-1)^{\text{th}}$ value in R would always be 1 and (c) vector L would store the product, $L_{i-1} \times R_i$ with $L_0 \in r$, where r is a 64 bit random variable generated using *randomgen()*. Hence, the number of shares generated from the secret exponent is limited by these notions.

B. GENETIC ALGORITHM

One of the most widely used evolutionary computation techniques for optimization is GA. In almost all the fields where optimization can be done, GA has always produced outstanding results [25]. GA maintains a population of possible individuals based on some selection function and other operators like mutation and recombination. Each of these individuals are provided a fitness value and using a proper fitness function GA pulls out those individuals having highest fitness. The initially selected individuals acts as parents and undergoes mutation and crossover to generate/reproduce new values/offspring which depicts the next population. To summarize, GA is mainly composed of three main operations,

- Selection: From a population of possible individuals select the ones having highest fitness values.
- Crossover: It exchanges some bits between the parents so that the new individuals inherits the parents. There can be single point crossover where each parent occurs once in the offspring or double crossover where one parent can occur at multiple times in the offspring.
- Mutation: Some bits within the selected individuals are replaced with one another based on a mutation rate which determines to what extent mutation needs to be performed.

GA has been previously used in building optimal addition chains for modular exponentiation [37], [39], and [43]. Even successful timing attacks over RSA has appeared in literature [38]. Besides, it has been also used to protect AES S-box against DPA attacks [40], [41]. In 2012, Batina *et al.* [44] presented the concept of generating evolutionary ciphers (EVOC) with evolutionary computing to resist DPA attacks. Their model could generate dynamic ciphers using TRNG which forbids not only attackers but also the designers from knowing the ciphers. These ciphers were randomly selected using an intelligent search algorithm. They have challenged the Kocher's model [1] of DPA attacks by making the selection function unknown to the attackers with dynamic ciphers. As the attackers are prevented from creating a selection function, they cannot mount DPA attacks.

However, unlike EVOC, our objective is to obtain optimal shares of the large secret exponent in modular exponentiation to resist HODPA attacks. For this purpose GA has been used in inner product to generate the optimum IP pairs (L, P) which resembles as shares of the secret exponent d and leaks least mutual information about it.

C. NEAREST NEIGHBOR ALGORITHM

Greedy approach is considered by this algorithm to search the minimum distance between two end points. Initially it was used in solving the problems like traveling salesman with efficiency but doesn't guarantees optimal solution. Presently, the algorithm is employed in various fields of machine learning, pattern recognition and image processing. The Euclidean distance shown in Equation 3 computes the nearest

neighbors for a population

$$dist(x, y) = \sqrt{\sum x_i - y_i} \quad (3)$$

where, x_i and y_i are the value of i^{th} attribute.

We extend the nearest neighbor algorithm in our proposed work to search the minimum distance between the entropy (H) based IP shares. Let, S_1, S_2, \dots, S_n be the IP shares of the secret key (d) upto n shares, then the distance from S_1 is computed by,

$$dist(S_1, S_i) = H(S_1) - H(S_i) \quad (4)$$

where, S_i equals to n^{th} shares. As our proposed approach to find the nearest neighbor is entropy, we have named the algorithm as ‘‘Entropy based Nearest Neighbor (E-NN)’’ throughout the paper. Use of E-NN enables us to randomly choose n shares of an IP share to execute an individual modular exponentiation. Thus E-NN offers operation hiding by randomizing the order of individual modular exponentiation and resist from DPA and HODPA attack as a whole.

IV. PROPOSED WORK

As mentioned in previous sections, none of the existing resting techniques proposed how to address HODPA attacks. Here, we present first Genetic Algorithm with Inner product for splitting the exponent of modular exponentiation into multiple shares. We then propose an algorithm known as nearest neighbor based on entropy (E-NN) to randomize these shares prior to execution to exhibit operation hiding for resisting SPA & DPA attacks.

A. SPLITTING EXPONENT THROUGH INNER PRODUCT WITH GENETIC ALGORITHM (IP-GA)

We have mentioned earlier that in order to resist a cryptosystem from HODPA the sensitive data with intermediate values needs to be split into multiple shares. Asymmetric cryptosystems have been built in the pillars of modular exponentiation. Specifically in RSA, a private key (d) is being used for decryption and the challenge remains in preserving the secrecy of this key making it a highly sensitive data. HODPA attacks can be resisted, if this private key can be split into multiple shares which in return will also split sensitive intermediate results. Modular exponentiation with multiple shares of d can be computed by the following property,

Property 1: To compute $m = c^d \bmod N$, where d, c, m are the private key, the cipher text and plain text respectively. If we have d_1, d_2, \dots, d_n , from d such that,

$$\begin{aligned} d &= \sum_{i=1}^n d_i, \text{ then} \\ m &= c^d \bmod N \\ &= ((c^{d_1} \bmod N)(c^{d_2} \bmod N) \dots (c^{d_n} \bmod N)) \bmod N \\ &= \prod_{i=1}^n (c^{d_i} \bmod N) \bmod N \end{aligned}$$

We start with generic GA by considering a population of size 50 ($popsize = 50$) random individuals using algorithm 1. The individuals in the population are random numbers of 64 bit that are generated using a random function. Each possible pair of individuals (l, r) from this population is assigned a fitness value f as shown below,

$$f = l \times r \quad (5)$$

Algorithm 1 Population Generation

Input: x, y , popsize
Output: population as pop[]
1 $x, y = 64$ bit random numbers;
2 $pop[] =$ array to store the populations;
3 **for** i from 1 to $popsize$ **do**
4 $r = random(x, y)$;
5 $pop[i] = r$;
6 **end**
7 Return pop[];

Using the selection function shown in algorithm 2, the pairs having highest fitness value are selected. This pair further undergoes a single point crossover and swap mutation to change their characteristics. A crossover point C_p is chosen for both l and r . The bits of these shares from C_p to the last bit are exchanged in each of these shares respectively to form l_c & r_c which further undergoes mutation. Two mutation points M_{p1} & M_{p2} have been chosen for each share l_c & r_c respectively such that M_{p1} & $M_{p2} \in (2, len)$, where len is the length of l and r . Since both the mutation points lie between 2 and len , the probability of choosing a point is $\frac{1}{len-2}$ which will be the mutation rate. As the size of each individual in the population is 64 bit, hence the mutation rate in our proposed approach will be $\frac{1}{62} \approx 0.016$. The bits at M_{p1} and M_{p2} are then exchanged within each l_c and r_c separately for mutation.

Crossover and mutation are general operators of GA which was mentioned in earlier section. But in our proposed work, these operators make significance recoding within the shares. This is analogous with blinding, which also recodes the exponent by associating it with a blinding factor. As these shares are also used for generating the next population, there will be a diverse population in every iteration.

Finally, using algorithm 1 and algorithm 2 along with crossover and mutation, we present the our proposed IP with GA (IP-GA) in algorithm 3 to split the large exponent d into multiple shares in an efficient way. More precisely, algorithm 3 generates two vectors L[] and R[] which contains values to form the inner product pairs. These vectors are later used for exhibiting hiding through randomization with proposed E-NN algorithm as discussed next. All the inner product pairs are generated using the prior assumptions, using property 2, and recombined to generate the original exponent d .

Algorithm 2 Assigning Fitness Value and Selection

```

Input: Population pop[ ]
Output: Inner product pairs (L, R)
1  prod = 1;
2  L[ ] & R[ ] to store the inner product pairs;
3  f[ ][ ] to store fitness value if each share;
4  for i from 1 to popsize do
5      for j from i+1 to popsize do
6          x = pop[i];
7          y = pop[j];
8          prod = x × y;
9          f[i, j] = prod;
10     end
11 end
12 for i from 1 to popsize do
13     for j from i+1 to popsize do
14         if f(i, j) > f(i+1, j+1) then
15             x = i;
16             y = j;
17         else
18             x = i + 1;
19             y = j + 1;
20         end
21     end
22     L[i] = pop[x];
23     R[i] = pop[y];
24 end
25 Return L[ ] & R[ ];
    
```

Property 2: $\forall i \in n, d_i = L_i \times R_i$ such that n is the size of $L[]$ & $R[]$ and $\sum_{i=1}^n d_i = d$.

B. RANDOMIZATION WITH ENTROPY BASED NEAREST NEIGHBOR (E-NN) ALGORITHM

By splitting d into multiple shares (d_1, d_2, \dots, d_n), HODPA attacks can be resisted under necessary condition. To boost up the security prospects our proposed protocol randomize every individual modular exponentiation using E-NN algorithm. For each d (say S) shares, we compute the entropy $H(S)$ as shown below,

$$H(S) = - \sum_{i=1}^n p(S_i) \log(p(S_i)) \tag{6}$$

where, S_i with $i = 1..m$ denotes a share of d . To compute the probability value ($p(S_i)$) of every share, it is necessary to find all the possible combination (c_i) of S_i via considering its hamming weight (HW) such that if $x_i = \text{length}(S_i)$ and $h_i = \text{HW}(S_i)$ then,

$$c_i = {}^{x_i}C_{h_i} = \frac{x_i!}{h_i!(x_i - h_i)!} \tag{7}$$

$$p(S_i) = \frac{c_i}{x_i} \tag{8}$$

Algorithm 3 Proposed Inner Product With Genetic Algorithm (IP-GA)

```

Input: d
Output: L[ ], R[ ] such that  $\sum_{i=1}^n L_i \times R_i = d$  and  $L_i \in L[ ], R_i \in R[ ]$ 
1  iprod = 1;
2  newprod = 1;
3  dnew = d;
4  n = Maximum number of shares desired ;
5  for i from 0 to n do
6      if (iprod < d) then
7          population ← From Algorithm 1 ;
8          Initial shares  $L_{ini}, R_{ini}$  ← From Algorithm 2 ;
9           $L_m, R_m$  ← Shares after crossover and mutation ;
10         if ( $L_m > R_m$ ) then
11             Swap ( $L_m, R_m$ );
12         end
13         newprod =  $L_m \times R_m$  ;
14         if (i = 0) then
15              $L_i = R_m$  ;
16              $R_i = newprod$ ;
17         else
18             iprod =  $R_m \times newprod$  ;
19              $L_i = R_m$  ;
20              $R_i = iprod$ ;
21         end
22         iprod =  $L_i \times R_i$  ;
23         x =  $L_i, y = R_i$  (To generate next population using Algorithm 1);
24         if (iprod >= dnew) then
25              $L_i = dnew$  ;
26              $R_i = 1$ ;
27             break;
28         else
29             dnew = dnew - iprod ;
30         end
31     end
32 end
    
```

All of the entropy values are further stored in an array H according to the shares.

Our proposed protocol E-NN is based on greedy optimization method. Therefore, we required to select an initial local best solution. For selection, we consider first value in H as best local (S_{lb}) and compare with other entropy values in H based on its distance. The share with minimum distance based on entropy (Equation 4) from S_{lb} compared to other shares is allot as the next S_{lb} and its index in vector H is further stored into a new vector I . The operation continues till every values of the entropy are in an incremented order and their corresponding indices are stored in I . Further, the array I with random indices of the shares will ease to randomization to resist DPA attacks. The complete operation is described in the algorithm 4.

Algorithm 4 Proposed Hiding With E-NN Algorithm

Input: d_1, d_2, \dots, d_n such that $\sum_{i=1}^n d_i = d$, $H[\]$ having entropy of each share of d

Output: $I[\]$ containing random indices of the shares d

```

1  minindex = 0;
2  Slb = 0;
3  lbnew = 0;
4  differ = 0;
5  distance[ ];
6  n = number of shares ;
7  for i from 0 to n do
8      if (i > 0) then
9          Slb = H[minindex] ;
10         for k from 0 to n do
11             lbnew = H[k] ;
12             if (Slb = lbnew) then
13                 differ = 0 ;
14             else
15                 differ = Slb - lbnew ;
16             end
17             distance[k] = differ ;
18         end
19         Slb = distance[0] ;
20         for l from 1 to n do
21             lbnew = distance[l] ;
22             if (lbnew < Slb) then
23                 Slb = lbnew ;
24             end
25             minindex = l ;
26         end
27         I[i] = minindex ;
28     else
29         Slb = H[i] ;
30         for j from i + 1 to n do
31             lbnew = H[j] ;
32             if (lbnew < Slb) then
33                 Slb = H[j] ;
34                 I[i] = j ;
35             end
36         end
37         minindex = I[i] ;
38     end
39 end
    
```

Once algorithm 1 to algorithm 4 have been pre-computed, modular exponentiation can be computed using Property 1 explained earlier. However, it is worth mentioning that, the order of individual modular exponentiation is randomized due to E-NN. Algorithm 5 shows the steps for computing modular exponentiation. Here d_{inter} refers to a share of d , $result_{inter}$ is the intermediate result and $result$ is the final desired output.

Algorithm 5 Secured Modular Exponentiation With IP-GA and E-NN

Input: $M, N, L[\]$ & $R[\]$ from algorithm 3, $I[\]$ from algorithm 4

Output: $M^d \text{ mod } N$

```

1  dinter = 0;
2  result = 1;
3  n = size of I[ ];
4  for i from 0 to n do
5      index = I[i];
6      dinter = L[index] × R[index] ;
7      resultinter = Mdinter mod N ;
8      result = result × resultinter ;
9  end
10 result = (result) mod N ;
11 Return result ;
    
```

C. SECURED IMPLEMENTATION OF RSA AND CRT-RSA WITH PROPOSED IP-GA & E-NN

Our proposed approach for computing secured modular exponentiation in RSA and CRT-RSA widely considered public-key cryptosystems. Algorithm 5 can be directly used for RSA decryption with d as secret decryption key and M be the cipher text. Algorithm 6 demonstrates how our proposed approach has been implemented in CRT-RSA. Its worth mentioning that the steps have been shown during decryption only consists of cipher text c , private key d and modulo N . For CRT-RSA, the recombination has been done using Gauss combination given by,

$$s \leftarrow ((s_p \times q \times q_{inv}) + (s_q \times p \times p_{inv})) \text{ mod } N \quad (9)$$

where s is the final result and s_p, s_q are intermediate results for d_p and d_q respectively.

Algorithm 6 Proposed Secured Modular Exponentiation With IP-GA and E-NN in CRT-RSA Decryption

Input: $c, N, p, q, d_p = (d \text{ mod } (p - 1))$ & $d_q = (d \text{ mod } (q - 1))$

Output: $c^d \text{ mod } N$

```

1  Precomputation : ;
2  Ldp[ ] & Rdp[ ] ← Using Algorithm 3 for dp;
3  Ldq[ ] & Rdq[ ] ← Using Algorithm 3 for dq;
4  Idp[ ] ← Using Algorithm 4 for dp;
5  Idq[ ] ← Using Algorithm 4 for dq;
6  n = size of L[ ] & R[ ] for dp & dq;
7  Resultdp ← Using Algorithm 5 with dp as input ;
8  Resultdq ← Using Algorithm 5 with dq as input ;
9  Result ← Combining intermediate results for dp & dq using Equation 9;
10 Return result ;
    
```

V. SECURITY ANALYSIS

The security of the proposed protocol is analysis in the following steps:

- *Modular exponentiation with shares*

At any instance say $i = 1$, algorithm 5 first computes $d_{inter} = L[index] \times R[index]$. Then, $m^{d_{inter}} \bmod N$ is computed. Here, d_{inter} is the inner product of the pairs from L and R computed by algorithm 3 which is one of the n shares of d , fulfilling the necessary condition to resist HODPA.

- *Recoding of shares*

The initial IP pair L_{ini}, R_{ini} selected from the initial population using algorithm 2 are recoded to L_c, R_c during crossover and subsequently to L_m, R_m in mutation. The final IP pair consists of R_m and product of L_m, R_m . Due to recoding, it will be very difficult to guess a share from a given population providing additional security in the entire computation.

- *Hiding through randomization*

At the same instance $i = 1$, as in the first observation, algorithm 5 computes $index = I[i]$ in step 5 where, $I[]$ stores the random indices of vectors $L[]$ & $R[]$ using algorithm 4. The same can be observed in steps 4 & 5 of algorithm 6. Hence, a random share of d is being chosen every time which proves operation hiding to resist generic SPA and DPA attack.

These observations show that the proposed work computes secured modular exponentiation to resist both HODPA as well as SPA/DPA attacks. In the next section, we probe various DPA and HODPA attacks and verify the resisting strength of proposed IP-GA with E-NN.

A. RESISTANCE TO DPA ATTACKS

The primary strength of our proposed work is that the entire modular exponentiation is done asynchronously with n shares of the exponent. Algorithm 1 to algorithm 4 are computed before the actual modular exponentiation takes place. Hence, the over head is only during precomputation. Besides due to GA, with every iteration a new population is being generated and hence all the shares are different from the previous ones which strengthens the security to a very large extent. We next discuss in brief some of the popular DPA attacks to show how our proposed approach is capable to resist them.

The generic DPA attacks by Messerges *et al.* [3] examined the correlations between known bits and secret exponent to distinguish multiplication operations from squaring. Even if the modular exponentiation in algorithm 5 is performed using binary methods, each of these execution would involve only one of the shares of the exponent randomly chosen at a time. Hence, if their attack is mounted on the proposed model, correlations will be generated for any arbitrary share of the secret exponent making those attacks unsuccessful.

Other attacks mostly targeted exponent and message blinding type countermeasures. These attacks would first guess the blinding factor and then target to recover the key. However, most of them were implemented for small exponents only.

Our proposed approach has been designed for large exponents of 1024-2048 bits and hence feasibility of these attacks are very less. But, it would be worth mentioning that we have not taken any measures on message blinding and hence may be vulnerable to attacks based on it like Witteman *et al.* [33] and Wunan *et al.* [32].

The Double Count Attacks (DCA) proposed by Kaminaga *et al.* [28] proposed a position checker tool for $2^t - ary$ implementation of RSA. Their tool could reconstruct the entire exponent when 1536 bit RSA was implemented using $2^6 - ary$ method. But, algorithm 4 in our proposed work randomizes the exponentiation using E-NN algorithm due to which reconstructing the exponent using DCA would be extremely difficult. Similar to DCA, due to randomization, Big Mac attacks proposed by Walter [29] and Horizontal correlation attacks [30] which was an extension to Big Mac attacks could also be resisted by our work. For both these attacks, correct time synchronization of the modular exponentiation was a critical factor which is broken by choosing random shares of d as can be seen in algorithm 5.

B. RESISTANCE TO HODPA ATTACKS

The proposed attack model by Kim *et al.* [31] was based on second-order correlation power analysis attacks on RSA. Their attack model was enhanced work of Okeya-Sokurai model [42] who targeted the BRIP countermeasure of Mamiya *et al.* [4] on ECC. For mounting their attack, first power traces were collected without any consideration of plain texts or cipher texts. As the target operation was multiplication only, these traces were reconstructed by extracting and concatenating the signals of multiplication from power traces produced during the main exponentiation. The newly constructed traces were defined as $C_i (1 \leq i \leq n)$ for n traces. The points corresponding to multiplication signals $C_i = M_{i,0} || M_{i,1} || \dots || M_{i,n-1}$ were separately defined as $C_i = (C_{i,0}, \dots, C_{i,1*L-1}) || C_{i,1*L}, \dots, C_{i,2*L-1} || \dots || C_{i,(n-1)*L}, \dots, C_{i,n*L-1} ||$, where $M_{i,k} (0 \leq k \leq n-1)$ was the multiplication signal corresponding to $(k+1)^{th}$ action in the i^{th} power trace with L be the length of the multiplication power trace and n be the size of the secret exponent.

Then second-order correlation was computed on the reconstructed power traces $C_i (1 \leq i \leq n)$. These correlated traces $CT_i (2 \leq i \leq n)$ implied the relation between multiplication at corresponding MSB of the secret exponent and other multiplications. The correlation traces showing high correlation values revealed the secret key bits. These points will be the point of interest which could not be visually distinguished. This new attack could easily find points of interest and does not require any profiling stage making it practical. Besides, this attack required fewer power traces than Okeya-Sakurai model [42].

However, it could be clearly seen that the triumph of Kim's model [31] totally depends on synchronized exponent bits else recovery won't be possible. Countermeasures with message blinding would fall to their attacks but if we probe our proposed secured model into their attack it can be

seen that during one modular exponentiation only a single share of the secret exponent is computed. It can be seen in step 6 & 7 of algorithm 5, $d_{inter} = L[index] \times R[index]$ and $result_{inter} = M^{d_{inter}} \bmod N$. The power traces that would be reconstructed with our model say P_j ($1 \leq j \leq m$) and corresponding correlated traces PT_j ($2 \leq j \leq m$) will be for m bits of j^{th} share of secret key. Hence, the number of power traces that would be required to recover bits from all the shares will increase significantly to a very large extent. Further, even if bits of each exponent share was correctly revealed, for accurate recovery of the secret key each of the shares has to be recombined in proper order. However, we have proposed E-NN in algorithm 3 for hiding the operations which would make it computationally infeasible to recover the actual secret exponent. The requirement of large number of power traces and additional computation in rearranging the shares in correct order would make their attack or any similar HODA attacks impractical.

Our proposed work would however be vulnerable to those DPA attacks which targets the key generation for RSA like Vuillaume et al. [34]. We summarize all these findings in Tab. 1 showing the various attacks which our proposed approach can resist and which are beyond. There are many resisting techniques to protect SPA/DPA in RSA and CRT-RSA in literature. However, it was seen that none of these works have addressed HODPA so far except ours. Comparison of our proposed work with some relevant SPA/DPA resisting protocols for modular exponentiation based cryptosystems such as RSA is shown in Tab. 2.

C. RESISTANCE TO HORIZONTAL CORRELATION & TEMPLATES ATTACKS

One of the special cases of power analysis attacks is Horizontal correlation analysis. Such an attack on exponentiation was presented by Clavier et al. in [45] where they exploited the vulnerability of RSA encryption with a single power curve. In their attack model each single bit of the secret exponent (d) was determined by identifying the multiplication operations with the message (m) performed during the encryption. Considering that s bits of d say $d_{v-1}, d_{v-2}, d_{v-s}$ are known, the $(s+1)^{th}$ bit was identified to be 1 if and only if the next operation is a multiplication with m . Their analysis showed that with l bit multiplier, on a RSA encryption of key size n , $(\frac{n}{l})^2$ segments will be generated in a single power trace. Thus longer keys will be at a higher risks with their attack. They further showed that general countermeasures like exponent blinding or secured methods like “multiply-always” or “montgomery ladder” will susceptible to their attack. However, randomizing the execution order during exponentiation can be an effective solution to protect such attacks. The horizontal attacks in ECC by Dugardin et al. [46], where the scalar multiplications in ECC were targeted for revealing the secret key also showed that general blinding or randomization of scalar multiplications will not efficient against their attack. In context to our proposed approach, we are able to split large exponent to

smaller shares and then have changed the execution order of every shares using the E-NN algorithm. Hence, if Clavier’s attack model is probed, then by identifying the multiplication operations only the bits of the exponent may be revealed. But to obtain the secret key in proper order, each of the shares need to be identified and then rearranged which will be difficult because each shares will be of different size and with t shares, the number of combinations possible will be 2^t . Similar resistance can be provided to the attack model of Dugardin et al. [46].

Template attacks have been challenging the security of public key cryptosystems like RSA and ECC to great extent. These attacks combine statistical modeling and power analysis with two different phases, first for template acquisition and second for template matching. The online template attacks by Batina et al. [47], [48] shows the potential threat and severity of such attacks in ECC. Their attack model only uses one template trace per bit scalar which can be applied to all kind of scalar multiplications. The pattern matching was done using Pearson’s correlation coefficient. Their attack on “Double-and-Add-Always” algorithm was based on the two possible outcomes $2P$ and $3P$ which were dependent on the MSB to be 0 or 1. Considering these two possible states the key bits can be recovered iteratively. Their attack also targets the Montgomery Ladder, Side-channel atomicity approaches. The strength of their attack can be revealed by the fact that countermeasures which uses blinding or bit randomization does not effect their attack. In context with our proposed approach in algorithm 5, the exponentiation in line number 7 is computed randomly for one of the shares generated using algorithm 3. As Genetic algorithm has been used to create the shares, for every implementation of the algorithm will generate new shares and eventually new patterns for the same secret exponent. With the attack proposed by Batina et al. [47], [48], while correlating patterns the representation points will be always different. Even if this factor is overcome and key bits of every share is depicted but still due to random execution of the shares, the bits have to be rearranged in correct order for complete reveal of the correct key. This will be difficult because of the same fact as horizontal correlation analysis that with t shares the number of combination possible will be 2^t . But, still our proposed approach will not guarantee complete security from Batina’s online template attack models.

VI. RESULTS AND DISCUSSION

We analyze the results for our proposed protocol in this section. Two parameters have been analyzed, complexity and mutual information analysis. We first present a brief detail on the standard benchmark used in our implementation.

A. BENCHMARK

To evaluate the proposed work, we consider the benchmarks offered by Public key Cryptography Standards (PKCS) v2.1.10. PKCS which is devised and published by RSA Security has offered benchmarks for numerous

TABLE 1. Resistance of proposed work.

Attacks	Can be Resisted
General DPA [Messerges(1999)]	Yes
Big Mac Attacks [Walter(2001)]	Yes
Horizontal Correlation Analysis [Clavier(2010)]	Yes
Fouque's [Fouque(2006)]	Yes
DCA [Kaminaga(2015)]	Yes
Horizontal Correlation Analysis [Clavier(2010)]	Yes
Kim's [Kimm(2010)]	Yes
Schindler & Itoh [Schindler(2011)]	Yes
Wittman's [Wittman(2011)]	No
Vuilaume's [Vuilaume(2012)]	No
Wunan's [Wang(2015)]	No
Batina's Attacks [Batina(2014)], [Batina(2017)]	May be

TABLE 2. Comparison with previous resisting methods (AB = Not mentioned in the work).

Previous Works	RSA	CRT-RSA	SPA & DPA	HODPA
Mamiya et al. [Mamiyas(2004)]	Secured	AB	Yes	No
Kim et al. [Kim(2004)]	Secured	AB	Yes	No
Kim et al. [Kim(2005)]	Secured	AB	Yes	No
Fournaris et al. [Fournaris(2012)]	Secured	Secured	Yes	No
Kim et al. [Kim(2014)]	Secured	Secured	Yes	No
Choi et al. [Choi(2016)]	Secured	Secured	Yes	No
Kim et al. [Kim(2016)]	Secured	Secured	Yes	No
Mahanta et al. [Mahanta(2017)]	Secured	Secured	Yes	No
Mahanta et al. [Mahanta(2017)]	Secured	Secured	Yes	No
Our proposed work	Secured	Secured	Yes	Yes

asymmetric techniques from early 90s. The mentioned version offered 10 distinct sets of N, p, q, d for RSA along with d_p, d_q for CRT-RSA. Size of “d” and “N” are 1024 bit, 1025 bit, . . . , 1031 bit, 1536 bit and 2048 bit. There are six different plain texts with corresponding cipher texts for each such set. The results shown here are for 1024, 1536 and 2048 bit of “d” and “N” only.

Simulation of the proposed protocol is done in *Mupad*, a part of Matlab 2014b. Due to very large (1024-2048) bit value, Mupad is very compatible. We also consider, 8 Gb RAM 2.27 GhZ configured workstation for implementing our proposed protocol to achieve the results.

B. COMPLEXITY

In order to address HODPA attacks, the proposed work first splits the secret exponent into multiple shares. Then, each individual modular exponentiation is randomly implemented with these shares one at a time. However, these steps are being precomputed prior to actual modular exponentiation. Hence, there is an overhead in the time complexity due to the precomputation.

As the sensitive data has been split using inner product with GA, the over complexity will be similar to complexity of GA. Hence,

$$\begin{aligned}
 O(Split) &= O(O(Fitness)) + O(Crossover) + O(Mutation) \\
 &= O(g(nm + mn + n)) \\
 &= O(gnm)
 \end{aligned}$$

where, g = number of generations, n = number of population and m = number of individuals in each population. Since in our case m is fixed hence the overall complexity for splitting of sensitive variable is $O(gn)$.

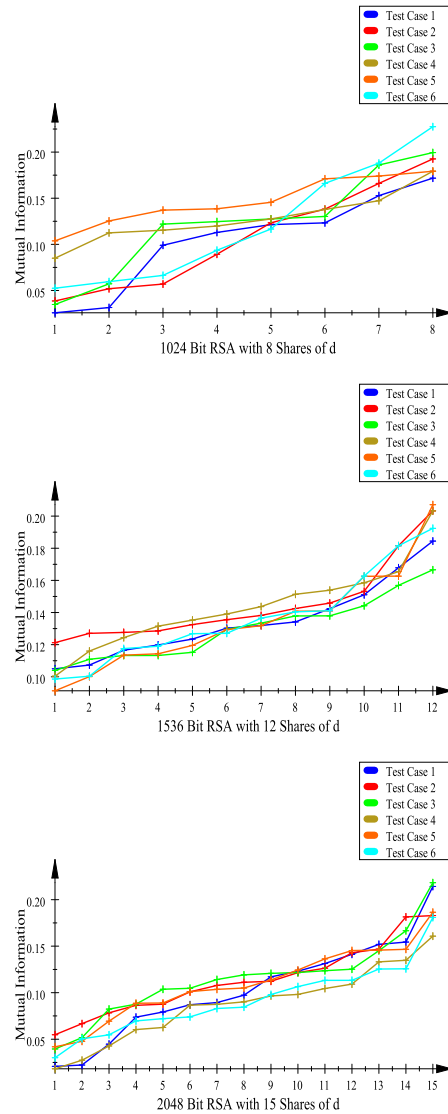


FIGURE 1. Mutual information of shares for 1024 Bit, 1536 Bit, and 2048 Bit RSA.

For hiding, the shares have been randomized using E-NN in algorithm 4. Since for finding every share of d , a new population is generated hence the number of shares will be equal to number of populations i.e. n . Further, in algorithm 4, each of the n shares computes distance from $n-1$ shares without repetition. Hence, the complexity for hiding will be,

$$O(Hiding) = O(n(n - 1))$$

However, algorithm 3 & 4 are precomputed as can be seen in algorithms 5 & 6. Hence, the proposed work have a large potential to combat against the attacks such as DPA and HODPA but at an computational overhead of $O(gn) + O(n(n - 1))$.

The precomputation for the shares performed in algorithm 3 and randomizing the order of execution through algorithm 4 will also consume some additional space. With

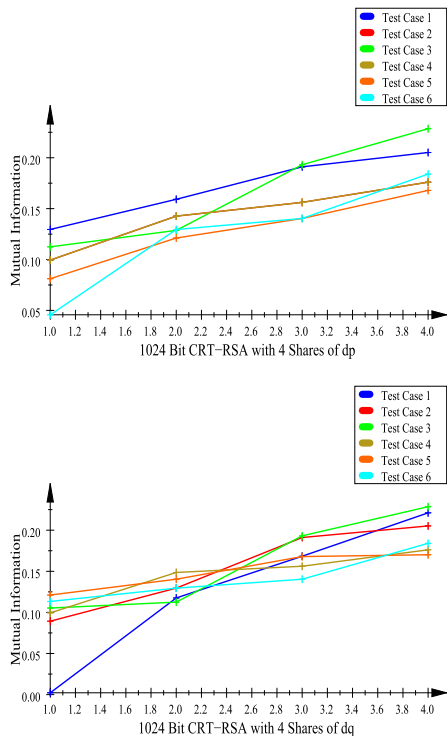


FIGURE 2. Mutual information of shares for 1024 Bit CRT-RSA.

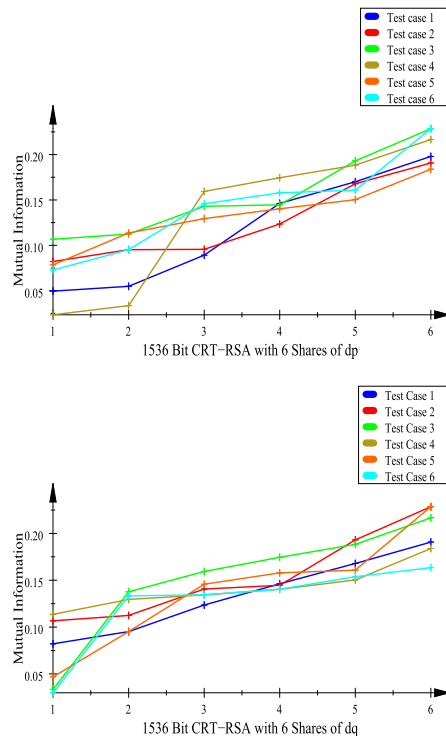


FIGURE 3. Mutual information of shares for 1536 Bit CRT-RSA.

a fixed population size, the selection done in algorithm 2 composes the vectors $L[]$ and $R[]$ to store the m shares. These vectors are finally reconstructed with proposed IP-GA in algorithm 3 with an additional space of $O(m)$. For entropy based nearest neighbor approach in algorithm 4, two vectors $H[]$ and $distance[]$ will be required with an additional space of $O(m + m) \equiv O(m)$. Thus, the overall additional cost of space due to precomputation in the proposed work will be $O(m)$ where $m =$ number of shares generated from d . In all our computations, we could find that the maximum number of shares generated was 15 ($m = 15$) in case of 2048 bit RSA.

C. MUTUAL INFORMATION ANALYSIS

It is an entropy based measurement which depicts the amount of mutual information two independent values shares. It is mainly a distinguisher introduced in 2008 [26] to mount HODPA attacks. It leads to successful recovery of key with minimum information of the leakage device [27]. MIA has proven to be an effective measurement for successfully mounting HODPA attacks.

However, in our analysis we have used Mutual Information Analysis (MIA) to measure the amount of information each individual share of the sensitive variable would leak if any HODPA attack is mounted. The lesser is the MIA value, higher will be the security. For each share we computed the entropy $H(S)$ using Equation 6 as in proposed E-NN algorithm. Similar to Equation 6. we can compute the entropy

of the sensitive variable (d) for all the possible values by,

$$H(d) = - \sum_{i=1}^l p(d_i) \log(p(d_i)) \tag{10}$$

Using both Equation 6 & 10 we can compute the joint entropy of S and d by,

$$H(d, S) = - \sum_{i=1, j=1}^{n, l} p(d_i, S_j) \log(p(d_i, S_j)) \tag{11}$$

However, since S and d are partial independent variables and few outcome of S is known. Therefore, the uncertainty of knowing d can be represented by conditional entropies of the two variables as shown below,

$$H(d|S) = \sum_{j=1}^l p(S_j) H(d|S_j) \tag{12}$$

where,

$$H(d|S_j) = - \sum_{i=1}^n p(d_i|S_j) \log(p(d_i|S_j)) \tag{13}$$

Taking these entropy varieties, the mutual dependency between two variables X and Y is referred as mutual information can be depicted by,

$$\begin{aligned} I(d; S) &= H(d) - H(d|S) \\ &= H(S) - H(S|d) \\ &= H(d) + H(S) - H(d, S) \end{aligned}$$

TABLE 3. Mutual information (I) for shares of 1024-bit RSA key.

Shares	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5	Test Case 6
1	0.025784332	0.038617414	0.034834907	0.084992602	0.103778347	0.052428622
2	0.031444544	0.051876459	0.057201051	0.112412635	0.125342896	0.059551365
3	0.099015577	0.056922745	0.121944435	0.115351437	0.137067404	0.066308482
4	0.112858827	0.089283034	0.124662719	0.119892129	0.138513439	0.093575568
5	0.121390293	0.123355467	0.127549489	0.127497762	0.145623695	0.116603467
6	0.123283432	0.138191262	0.130270298	0.137785056	0.171019457	0.166047151
7	0.152850235	0.166047151	0.186079673	0.147305824	0.174018033	0.187997549
8	0.171750886	0.192615991	0.199485945	0.179270238	0.179270238	0.227551405

TABLE 4. Mutual information (I) for shares of 1536-bit RSA key.

Shares	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5	Test Case 6
1	0.104988829	0.121251207	0.103930544	0.100458377	0.091326604	0.098623099
2	0.107302598	0.127073764	0.110906716	0.116005055	0.100117491	0.100383269
3	0.11653235	0.12762937	0.113288053	0.124366222	0.11348589	0.117599252
4	0.119792844	0.128543375	0.113288053	0.131527345	0.1142907	0.11903936
5	0.123497017	0.132482083	0.115249345	0.135279543	0.119556613	0.126839194
6	0.130295238	0.135559776	0.129569437	0.13909336	0.129326751	0.127062084
7	0.131932603	0.138222372	0.133385829	0.143708063	0.13185107	0.13647575
8	0.134171651	0.142501176	0.137889503	0.151351524	0.140797153	0.140421639
9	0.14237431	0.145866362	0.137962651	0.153872187	0.140984699	0.141139317
10	0.151026398	0.153215769	0.144229698	0.158495743	0.162553018	0.162728434
11	0.167843252	0.181547803	0.156879518	0.165454108	0.162667963	0.181547803
12	0.18447667	0.20327663	0.166595092	0.20327663	0.207089019	0.192392375

TABLE 5. Mutual information (I) for shares of 2048-bit RSA key.

Shares	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5	Test Case 6
1	0.020785817	0.054779298	0.039524208	0.018245416	0.041858615	0.030135873
2	0.022251738	0.066756774	0.051651149	0.027469464	0.047779844	0.050455023
3	0.044675875	0.078338354	0.082343624	0.042724531	0.069390977	0.054767311
4	0.073675759	0.086434688	0.087618996	0.060405	0.088665344	0.069638114
5	0.079284712	0.087618996	0.103639771	0.062616403	0.08889592	0.072091051
6	0.087009333	0.100893297	0.104727874	0.086597506	0.1011597	0.07397018
7	0.089375679	0.107752167	0.114159991	0.087461689	0.103563841	0.083001823
8	0.097311644	0.1111178109	0.119130539	0.090164096	0.1049928	0.084455569
9	0.117009712	0.112282973	0.120787994	0.096511097	0.113663591	0.098017318
10	0.123129969	0.121538498	0.12168967	0.098153445	0.124323667	0.106565493
11	0.131335127	0.126356282	0.123613454	0.104470691	0.136374144	0.113231266
12	0.141271711	0.142965543	0.125242445	0.109193518	0.145280191	0.113284992
13	0.151865565	0.146816155	0.145082737	0.133104778	0.145734758	0.125453465
14	0.154430056	0.181477077	0.16660207	0.134838389	0.146614483	0.125699111
15	0.214152686	0.182859289	0.218190122	0.16077551	0.186419257	0.180772129

TABLE 6. Mutual information (I) for shares of 1024-bit CRT-RSA d_p & d_q .

Key type	Shares	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5	Test Case 6
d_p	1	0.205148886	0.176026283	0.228493614	0.176026283	0.167898886	0.183925637
d_p	2	0.191076684	0.156156678	0.193090357	0.156156678	0.1403037	0.1403037
d_p	3	0.129524186	0.099430694	0.112463103	0.099430694	0.121115057	0.129469691
d_p	4	0.159159736	0.142626271	0.12868196	0.142626271	0.081119404	0.045720572
d_q	1	0.221044261	0.205148886	0.228493614	0.176026283	0.167898886	0.183925637
d_q	2	0.11751926	0.191076684	0.193090357	0.156156678	0.1403037	0.1403037
d_q	3	0.002272095	0.129524186	0.112463103	0.099430694	0.121115057	0.129469691
d_q	4	0.168255056	0.089274094	0.105331998	0.148547797	0.170106559	0.113295679

TABLE 7. Mutual information (I) for shares of 1536-bit CRT-RSA d_p & d_q .

Key type	Shares	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5	Test Case 6
d_p	1	0.054881403	0.167898886	0.228493614	0.216573011	0.183925637	0.228493614
d_p	2	0.197938752	0.190722825	0.193090357	0.188335469	0.1403037	0.145743071
d_p	3	0.170032277	0.082159511	0.112463103	0.174461415	0.129469691	0.160765782
d_p	4	0.049523166	0.123499911	0.106640025	0.159331941	0.113568531	0.157841227
d_p	5	0.089149227	0.095194024	0.144620106	0.033519599	0.150324022	0.095194024
d_p	6	0.146228211	0.095585145	0.142978029	0.023490979	0.078480228	0.072838054
d_q	1	0.167898886	0.228493614	0.216573011	0.183925637	0.228493614	0.133292283
d_q	2	0.190722825	0.193090357	0.188335469	0.1403037	0.145743071	0.1403037
d_q	3	0.082159511	0.112463103	0.174461415	0.129469691	0.160765782	0.163457144
d_q	4	0.123499911	0.106640025	0.159331941	0.113568531	0.157841227	0.153742221
d_q	5	0.095194024	0.144620106	0.033519599	0.150324022	0.095194024	0.029870769
d_q	6	0.146228211	0.140580848	0.137697567	0.134355426	0.04680708	0.134355426

Since the information leaked from a device built within CMOS for HODPA is actually the transition taking from from 1 → 0 and 0 → 1, the hamming weight is employed to model the leakage. In context to inner product the gross

leakage is presented as,

$$Leak(L, R) = HW(L_1) + HW(R_1), \dots, HW(L_n) + HW(R_n) \tag{14}$$

TABLE 8. Mutual information (I) for shares of 2048-bit CRT-RSA d_p & d_q .

Key type	Shares	Test Case 1	Test Case 2	Test Case 3	Test Case 4	Test Case 5	Test Case 6
d_p	1	0.054881403	0.205148886	0.176026283	0.183925637	0.183925637	0.234582283
d_p	2	0.197938752	0.191076684	0.156156678	0.1403037	0.178408318	0.193090357
d_p	3	0.170032277	0.129524186	0.099430694	0.129469691	0.175257729	0.170032277
d_p	4	0.049523166	0.123618028	0.027768986	0.113568531	0.141070514	0.064835592
d_p	5	0.089149227	0.031386767	0.123033988	0.150324022	0.148554061	0.117987884
d_p	6	0.140731343	0.055176777	0.021719592	0.121454872	0.139548149	0.057670692
d_p	7	0.125105086	0.126482635	0.114039123	0.084613623	0.104458685	0.093685059
d_p	8	0.040646595	0.123786669	0.121144975	0.128117642	0.107873564	0.017811189
d_q	1	0.054881403	0.205148886	0.183925637	0.234582283	0.098790794	0.054881403
d_q	2	0.197938752	0.191076684	0.1403037	0.193090357	0.083414744	0.123256761
d_q	3	0.170032277	0.129524186	0.129469691	0.170032277	0.065852174	0.050945707
d_q	4	0.049523166	0.123618028	0.113568531	0.064835592	0.135580992	0.123539524
d_q	5	0.089149227	0.031386767	0.150324022	0.117987884	0.104011185	0.138300429
d_q	6	0.140731343	0.055176777	0.121454872	0.057670692	0.10350638	0.125266186
d_q	7	0.125105086	0.126482635	0.084613623	0.093685059	0.10234416	0.134410262
d_q	8	0.132862605	0.132106971	0.09577338	0.115055911	0.126194328	0.091462583

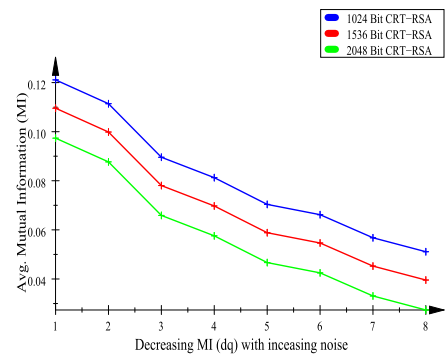
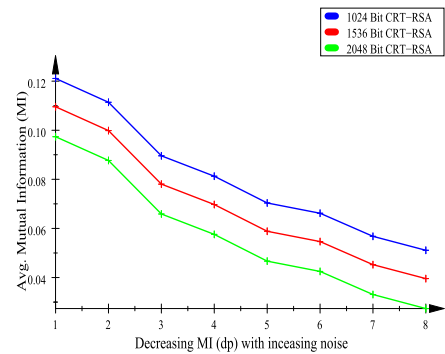
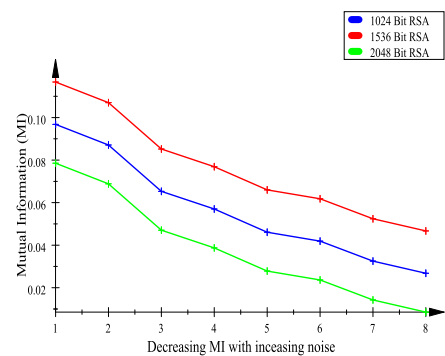
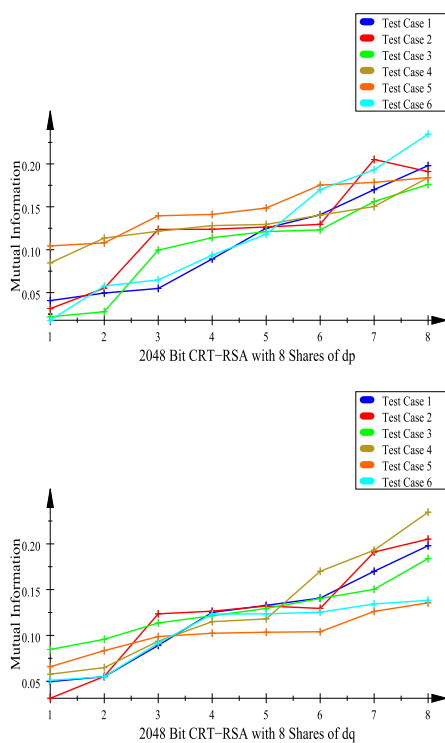


FIGURE 4. Mutual information of shares for 2048 Bit CRT-RSA.

Due to the leakage, the mutual information will hence be, $I(d; Leak(L, R))$, where d is the actual secret data. Tables 3, 4 and 5 elaborates MIA of the proposed work in RSA of size 1024, 1536 and 2048 bits for 6 test cases. Figure 1 represents these results graphically. Similarly, for all variances of CRT-RSA, MIA has been furnished in tables 6 to 8 along with figures 2, 3 and 4 for the same test cases.

D. MUTUAL INFORMATION ANALYSIS WITH GAUSSIAN NOISE

In general, every leaked share is associated with an independent Gaussian noise which affects the entire leakage. Hence,

FIGURE 5. Decrease in mutual information of shares with increase in noise.

the leakage model in Equation 14 with noise will change into,

$$Leak(L, R) = HW(L_1 + \epsilon) + HW(R_1 + \epsilon), \dots, + HW(R_n + \epsilon) \tag{15}$$

where, ϵ denotes Gaussian noise computed by,

$$\epsilon = \frac{1}{2\sqrt{2\pi}} e^{-\frac{(\epsilon-\mu)^2}{2\sigma^2}} \quad (16)$$

where, μ = mean and σ = standard deviation. In our analysis, we have also computed the mutual information for the shares associated with Gaussian noise. We have considered $\mu = 0$ and the range of σ from 0.4, 0.6, 0.8, 1.0, 1.2, 1.4, 1.6 and 1.8 to see the effects. We found that for each increase in σ , the mutual information was immensely effected. Larger the noise, lesser was the amount of mutual information leaked from the shares. Fig. 5 shows the decrease in mutual information from the shares after addition of noise for both RSA and CRT-RSA.

VII. CONCLUSION

A secured approach of computing modular exponentiation to combat DPA and HODPA attacks is presented in this paper. The proposed approach first splits the sensitive data (secret exponent) into multiple shares via inner product using Genetic Algorithm. Thereafter, with E-NN technique, these shares are randomly chosen to computes individual modular exponentiation. Hence, through splitting the proposed work addresses HODPA and with hiding through randomization it addresses SPA & DPA attacks. With an pre-computation overhead of $O(gn) + O(n(n-1))$ in time and $O(m)$ in space, our proposed work can challenge HODPA as well as some of the popular DPA attacks on RSA and CRT-RSA to an appreciable extent.

ACKNOWLEDGMENT

HJM thanks Assam Don Bosco University for infrastructure support.

REFERENCES

- [1] P. Kocher, J. Joshua, and B. Jun, "Differential power analysis," in *Proc. CRYPTO*. Berlin, Germany: Springer, 1999, p. 789.
- [2] R. L. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Commun. ACM*, vol. 21, no. 2, pp. 120–126, Feb. 1978.
- [3] T. S. Messerges, E. A. Dabbish, and R. H. Sloan, "Power analysis attacks of modular exponentiation in smart cards," in *Proc. CHES* in Lecture Notes in Computer Science, vol. 1717, 1999, pp. 144–157.
- [4] H. Mamiya, A. Miyaji, and H. Morimoto, "Efficient countermeasures against RPA, DPA, and SPA," in *Proc. CHES*. Berlin, Germany: Springer, 2004, pp. 343–356.
- [5] C. Kim, J. Ha, S. H. Kim, S. Kim, S. M. Yen, and S. Moon, "A secure and practical CRT-based RSA to resist side channel attacks," in *Proc. Int. Conf. Comput. Sci. Appl.*, Berlin, Germany: Springer, 2004, pp. 150–158.
- [6] C. K. Kim, J. C. Ha, S. J. Moon, S. M. Yen, W. C. Lien, and S. H. Kim, "An improved and efficient countermeasure against power analysis attacks," *IACR Cryptogr. ePrint Arch.*, Tech. Rep. 2005/022, 2005. [Online]. Available: <http://eprint.iacr.org/2005/022.pdf>
- [7] Y. Wang, J. Leiwo, T. Srikanthan, and L. Jianwen, "An efficient algorithm for DPA-resistant RSA," in *Proc. IEEE Asia Pacific Conf. Circuits Syst. (APCCAS)*, Dec. 2006, pp. 1659–1662.
- [8] H. J. Mahanta and A. K. Khan, "Comparative modular exponentiation with randomized exponent to resist power analysis attacks," *Arabian J. Sci. Eng.*, vol. 42, no. 8, pp. 3423–3434, 2017.
- [9] H. J. Mahanta and A. K. Khan, "Securing RSA against power analysis attacks through non-uniform exponent partitioning with randomisation," *IET Inf. Secur.*, vol. 12, no. 1, pp. 25–33, Jan. 2018.
- [10] A. P. Fourmaris and O. Koufopavlou, "Protecting CRT RSA against fault and power side channel attacks," in *Proc. IEEE Comput. Soc. Annu. Symp. (VLSI)*, Aug. 2012, pp. 159–164.
- [11] H. Kim, D. G. Han, S. Hong, and J. Ha, "Message blinding method requiring, no. multiplicative inversion for RSA," *ACM Trans. Embedded Comput. Syst.*, vol. 13, no. 4, pp. 1–10, 2014.
- [12] J. Balasch, S. Faust, B. Gierlichs, and I. Verbauwhede, "Theory and practice of a leakage resilient masking scheme," in *Proc. ASIACRYPT*, Berlin, Germany: Springer, 2012, pp. 758–775.
- [13] J. Balasch, S. Faust, and B. Gierlichs, "Inner product masking revisited," in *Proc. EUROCRYPT*. Berlin, Germany: Springer, 2015, pp. 486–510.
- [14] S. Dziembowski and S. Faust, "Leakage-resilient circuits without computational assumptions," in *Proc. Theory Cryptogr. Conf.*, Berlin, Germany: Springer, 2012, pp. 230–247.
- [15] P. A. Fouque, S. K. Jacques, G. Martinet, F. Müller, and F. Valette, "Power attack on small RSA public exponent," in *Proc. CHES*. Berlin, Germany: Springer, 2006, pp. 339–353.
- [16] W. Schindler and K. Itoh, "Exponent blinding does not always lift (partial) spa resistance to higher-level security," in *Proc. Int. Conf. Appl. Cryptogr. Netw. Secur.*, Berlin, Germany: Springer, 2011, pp. 73–90.
- [17] W. Schindler and A. Wiemers, "Power attacks in the presence of exponent blinding," *J. Cryptograph. Eng.*, vol. 4, no. 4, pp. 213–236, Nov. 2014.
- [18] S. Bauer, "Attacking exponent blinding in RSA without CRT," in *Proc. COSADE*, Berlin, Germany: Springer, 2012, pp. 82–88.
- [19] W. Schindler and A. Wiemers, "Generic power attacks on RSA with CRT and exponent blinding: New results," *J. Cryptograph. Eng.*, vol. 7, no. 4, pp. 1–18, 2017.
- [20] J. A. Ambrose, R. G. Ragel, and S. Parameswaran, "RIJID: Random code injection to mask power analysis based side channel attacks," in *Proc. 44th ACM/IEEE Design Autom. Conf.*, Jun. 2007, pp. 489–492.
- [21] J. A. Ambrose, R. G. Ragel, and S. Parameswaran, "A smart random code injection to mask power analysis based side channel attacks," in *Proc. 5th IEEE/ACM Int. Conf. Hardw./Softw. Codesign Syst. Synth. (CODES+ISSS)*, 2007, pp. 51–56.
- [22] J. A. Ambrose, R. G. Ragel, and S. Parameswaran, "Randomized instruction injection to counter power analysis attacks," *ACM Trans. Embedded Comput. Syst.*, vol. 11, no. 3, pp. 1–28, 2012.
- [23] Y. Choi, D. Choi, H. Lee, and J. Ha, "An improved square-always exponentiation resistant to side-channel attacks on RSA implementation," *Intell. Autom. Soft Comput.*, vol. 22, no. 3, pp. 353–363, Jul. 2016.
- [24] H. Kim, Y. Choi, D. Choi, and J. Ha, "A secure exponentiation algorithm resistant to a combined attack on RSA implementation," *Int. J. Comput. Math.*, vol. 93, no. 2, pp. 258–272, Feb. 2016.
- [25] H. de Garis, "Introduction to evolutionary computing," *Evol. Comput.*, vol. 12, no. 2, pp. 269–271, Jun. 2004.
- [26] B. Gierlichs, L. Batina, P. Tuyls, and B. Preneel, "Mutual information analysis," in *Proc. CHES* in Lecture Notes in Computer Science, vol. 5154. Berlin, Germany: Springer, 2008, pp. 426–442.
- [27] N. V. Charvillat and F. Standaert, "Mutual information analysis: How, when and why?" in *Proc. CHES*, Berlin, Germany: Springer, 2009, pp. 429–443.
- [28] M. Kaminaga, H. Yoshikawa, and T. Suzuki, "Double counting in 2^l-ary RSA precomputation reveals the secret exponent," *IEEE Trans. Inf. Forensics Security*, vol. 10, no. 7, pp. 1394–1401, Jul. 2015.
- [29] C. D. Walter, "Sliding Windows succumbs to big MAC attack," in *Proc. CHES*, Berlin, Germany: Springer, 2001, pp. 286–299.
- [30] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Horizontal correlation analysis on exponentiation," in *Proc. 12th Int. Conf. Inf. Commun. Secur.*, in Lecture Notes in Computer Science, vol. 6476, 2010, pp. 46–61.
- [31] H. Kim, T. H. Kim, J. C. Yoon, and S. Hong, "Practical second-order correlation power analysis on the message blinding method and its novel countermeasure for RSA," *ETRI J.*, vol. 32, no. 1, pp. 102–111, Feb. 2010.
- [32] W. Wan, W. Yang, and J. Chen, "An optimized cross correlation power attack of message blinding exponentiation algorithms," *China Commun.*, vol. 12, no. 6, pp. 22–32, Jun. 2015.
- [33] M. F. Witteman, J. G. Van Woudenberg, and F. Menarini, "Defeating RSA multiply-always and message blinding countermeasures," in *Proc. CT-RSA*, Berlin, Germany: Springer, 2011, pp. 77–88.
- [34] C. Vuillaume, T. Endo, and P. Wooderson, "RSA key generation: New attacks," in *Proc. COSADE*, Berlin, Germany: Springer, 2012, pp. 105–119.
- [35] S. Chari, C. Jutla, J. R. Rao, and P. Rohatgi, "Towards sound approaches to counteract power-analysis attacks," in *Proc. CRYPTO*, Berlin, Germany: Springer, 1999, pp. 398–412.

- [36] J. Balasch, S. Faust, B. Gierlichs, C. Paglialonga, and F. X. Standaert, "Consolidating inner product masking," in *Proc. ASIACRYPT* in Lecture Notes in Computer Science, vol. 10624. Berlin, Germany: Springer, 2017, pp. 724–754.
- [37] N. Nedjah and L. M. Mourelle, "Fast pre-processing for the sliding window method using genetic algorithms," *Int. J. Comput. Syst. Signal*, vol. 4, no. 2, pp. 11–21, 2003.
- [38] H. Ali and M. Al-Salami, "Timing attack prospect for RSA cryptanalysts using genetic algorithm technique," *Int. Arab J. Inf. Technol.*, vol. 1, no. 1, pp. 80–84, 2004.
- [39] N. Nedjah and L. M. Mourelle, "Efficient pre-processing for large window-based modular exponentiation using ant colony," *Informatica*, vol. 29, pp. 155–161, Sep. 2005.
- [40] L. Batina, D. Jakobovic, N. Mentens, S. Picek, L. P. A. De, and D. Sisejkovic, "S-box pipelining using genetic algorithms for high-throughput AES implementations: How fast can we go?" in *Proc. INDOCRYPT*, Berlin, Germany: Springer, 2014, pp. 322–337.
- [41] S. Picek, B. Ege, L. Batina, D. Jakobovic, L. Chmielewski, and M. Golub, "On using genetic algorithms for intrinsic side-channel resistance: The case of AES S-box," in *Proc. 1st Workshop Cryptogr. Secur. Comput. Syst. (CS2)*, 2014, pp. 13–18.
- [42] L. Batina, D. Jakobovic, N. Mentens, K. Okeya, and K. Sakurai, "Power analysis breaks elliptic curve cryptosystems even secure against the timing attack," in *Proc. INDOCRYPT*, in Lecture Notes in Computer Science, vol. 1977. Berlin, Germany: Springer, 2000, pp. 178–190.
- [43] E. Vazquez-Fernandez, C. Cadena, and D. A. Reyes-Gomez, "A genetic algorithm with a mutation mechanism based on a Gaussian and uniform distribution to minimize addition chains for small exponents," in *Proc. IEEE Congr. Evol. Comput. (CEC)*, Jul. 2016, pp. 935–940.
- [44] M. Tang, Z. Qiu, M. Yang, P. Cheng, S. Gao, S. Liu, and Q. Meng, "Evolutionary ciphers against differential power analysis and differential fault analysis," *Sci. China Inf. Sci.*, vol. 55, no. 11, pp. 2555–2569, Nov. 2012.
- [45] C. Clavier, B. Feix, G. Gagnerot, M. Roussellet, and V. Verneuil, "Horizontal correlation analysis on exponentiation," in *Proc. Int. Conf. Inf. Commun. Secur.*, Berlin, Germany: Springer, 2010, pp. 46–61.
- [46] M. Dugardin, L. Papachristodoulou, Z. Najm, L. Batina, J. Danger, and S. Guilley, "Dismantling real-world ECC with horizontal and vertical template attacks," in *Proc. Int. Workshop COSADE*, Berlin, Germany: Springer, 2010, pp. 88–108.
- [47] L. Batina, L. Chmielewski, L. Papachristodoulou, and P. Schwabe, and M. Tunstall, "Online template attacks," in *Proc. INDOCRYPT*, Berlin, Germany: Springer, 2014, pp. 21–36.
- [48] L. Batina, L. Chmielewski, L. Papachristodoulou, and P. Schwabe, and M. Tunstall, "Online template attacks," *J. Cryptograph. Eng.*, vol. 9, no. 1, pp. 21–36, 2017.



AMIT KUMAR ROY received the master's and Ph.D. degrees in computer science and engineering from Assam University, India. Currently, he is serving as a Faculty Member with the Department of Computer Science and Engineering, National Institute of Technology Mizoram, Aizawl, India. He has published many research papers at international journals and conference proceedings. His research interests include security on wireless mesh networks, networks security, and cryptography.



KETAN KOTECHA has expertise and experience in cutting-edge research and projects in AI and deep learning for the last 25 years. He has published over 100 articles widely in a number of excellent peer-reviewed journals on various topics ranging from cutting-edge AI, education policies, teaching-learning practices, and AI for all. He has published three patents and delivered key note speeches at various national and international forums, including at the Machine Intelligence Laboratory, USA, IIT Bombay under World Bank Project, and the International Indian Science Festival organized by the Department of Science Technology, Government of India.

He was a recipient of two SPARC projects worth 166 lakh rupees from MHRD, Government of India, in AI in collaboration with Arizona State University, USA, and The University of Queensland Australia, and also the recipient of numerous prestigious awards, such as Erasmus+ Faculty Mobility Grant to Poland, DUO-India Professors Fellowship for research in responsible AI in collaboration with Brunel University London, U.K., LEAP Grant at the University of Cambridge, U.K., UKIERI Grant with Aston University, U.K., and a grant from Royal Academy of Engineering, U.K., under Newton Bhabha Fund. Currently, he is an Associate Editor of IEEE Access journal.



VIJAYAKUMAR VARADARANJAN received the Diploma degree (Hons.), the B.E. degree (Hons.) in CSE, the MBA degree (Hons.) in HRD, the M.E. degree (Hons.) in CSE, and the Ph.D. degree from Anna University, in 2012. He was a Professor and an Associate Dean of the School of Computing Science and Engineering, VIT University, Chennai, India. He has more than 18 years of experience including industrial and institutional. He also served as the Team Lead in industries like

Satyam, Mahindra Satyam, and Tech Mahindra for several years. He is currently an Adjunct Professor with the School of Computer Science and Engineering, University of New South Wales, Sydney, Australia. He is also a Visiting Postdoctoral Scientist with the Centro de Tecnologia, Federal University of Piauí, Brazil. He has published many articles in national and international level journals/conferences/books. He has initiated a number of international research collaborations with universities in Europe, Australia, Africa, Malaysia, Singapore, and North and South America. He had also initiated joint research collaboration between VIT University and various industries. He also organized several international conferences and special sessions in the USA, Vietnam, Africa, Malaysia, and India, including ARCI, IEEE, ACSAT, ISRC, ISBCC, and ICBC. His research interests include computational areas covering grid computing, cloud computing, computer networks, cyber security, and big data. He received the University-Level Best Faculty Award, for the year 2015 to 2016. He also received First Rank Award for his M.E. degree. He is a member of several national and international professional bodies, including IFSA, EAI, BIS, ISTE, IAENG, CSTA, and IEA. He is a reviewer of IEEE TRANSACTIONS, Inderscience, and Springer journals. He is also the lead guest editor for few journals in Inderscience, Springer, Elsevier, IOS, UM, and IGI Global.



HRIDYOY JYOTI MAHANTA received the Ph.D. degree from Assam University (A Central University), Silchar, under the guidance of Dr. Ajoy Kumar Khan. He worked as an Assistant Professor with the Department of Computer Science and Engineering, Assam Don Bosco University, Guwahati. His research interests include mainly applied cryptography, artificial intelligence, and machine learning.



KESHAB NATH (Member, IEEE) received the M.Tech. degree in information technology from Assam University, India, in 2014, and the Ph.D. degree in information technology from North-Eastern Hill University, in 2019. He is an Assistant Professor with the Department of Computer Science and Engineering, Indian Institute of Information Technology Kottayam. He has been working on various application areas, such as data mining in complex and evolving graphs. Currently, he has

been working on computer vision and graph neural networks (GNNs). He has published extensively in these areas in top journals and conference proceedings. He is in the technical committee of various reputed international journals and conferences. He is a member of ACM.