ISSN (Print) : 0974-6846
Indian Journal of Science and Technology, *Vol 10(14), DOI: 10.17485/ijst/2017/v10i14/110818, April 2017* ISSN (Online) : 0974-5645

Partitioned-Exponent Blinding: A Countermeasure against Power Analysis Attacks

Hridoy Jyoti Mahanta and Ajoy Kumar Khan

Department of CSE, Assam University, Silchar - 788011, Assam, India;
hridoy69@gmail.com, ajoyiitg@gmail.com

Abstract

Objectives: To resist power analysis attacks in modular exponentiation based cryptosystems like RSA. **Methods/Analysis:** A power analysis attack uses the instantaneous power consumed by a cryptosystems during its most sophisticated operations like encryption/decryption which uses the secret keys. Masking/blinding has proven its ability to resist such attacks by hiding the actual data during computation. As a result, analysis of the power consumption would reveal only the masked data keeping the actual secret data intact. **Findings:** Our proposed technique partitions the secret key into multiple parts and then blinds them individually before the cryptographic computations. With these blinded partitions, it would be very difficult for attackers to reveal the actual data as the power consumptions will be for blinded key not the actual key. Due to partitioning of the exponent there will be no uniformity in the power traces increasing resistance against power analysis attacks. **Applications:** All the cryptosystems have been found vulnerable to power analysis attacks, it resist modular exponentiation based cryptosystems like RSA from such attacks.

Keywords: Cryptosystem, Masking, Modular Exponentiation, Power Analysis Attacks, RSA

1. Introduction

Since the inception of Differential Power Analysis (DPA) attacks[1] in 1999, a number of techniques to counteract such attacks have appeared in literature. DPA attacks could use the instantaneous power consumed by the cryptosystem and analyze them to reveal the bit transitions that occurred during the sophisticated operations like encryption/decryption. The leakage of this information from power details was due to use of CMOS circuits within the cryptosystems. The total power consumed by CMOS depended on the operations and the data on which these operations were performed. This total power can be mathematically represented by,

$$P_{total} = P_{data} + P_{operations} + P_{constant} + P_{elec.noise}$$

The components P_{data} and $P_{operations}$ resulted into the dynamic or instantaneous power consumed during the cryptographic operations such that,

$$P_{dyn} = C\ V_{dd}^2 P_{0\to 1} F$$

where, C, V and F are the capacitance, voltage load and frequency of the cryptosystem while P was the power due to bit transition 0-1 or 1-0. As this dependency could not be removed easily due to hardware restrictions, the resisting techniques were designed so that they could 1. Randomize the order of the operations or 2. Mask the actual data and hence the power consumptions or 3. Make constant power consumption throughout the computations or 4. Execute the cryptographic operations without external power[2]. Masking was such a technique which could constant the power consumption and also hides the uniformity restricting the attackers from revealing the secret data from the power traces.

The success of masking depended on the fact that all the fundamental operations could be rephrased with the masked values and still produces the desired results[3]. But, if there were different operations like arithmetic and Boolean within the same fundamental operation, then two different masking techniques were required. In such case a suitable method to switch from one masking to another was an additional requirement.

* *Author for correspondence*

In[3] proved that even these switching methods were not secured against power analysis attacks. They further suggested that such algorithms should be avoided in real applications. In[4] presented a more secured method for switching from arithmetic to Boolean and Boolean to arithmetic masking. His work could decorrelate all the intermediate values from the masked data which was an issue with basic switching methods. In[5] presented another secured switching method improving the work[4] which they claimed to have a bottleneck despite of being very efficient. They implemented their work in SHA-1 and RC6 and found it 2.7 times faster than the work[4].

In[6] performed a higher order DPA on masked s-box of AES and DES to prove that masking still remains susceptible to power analysis attacks. Their attack was able to reveal the correct key despite of masking in 131072 plaintexts. With similar prospects in[7] performed first order DPA at gate level. Their work also showed that the glitches from the masked gates are vulnerable to DPA. As CMOS circuits always had glitches, hence general masking will always be victim to DPA at gate level. In[8] presented the concept of split mask where the input of the s-box was first masked with n and then masked with an individual random value as shown below.

$$S'(x \oplus n) = S(x) \oplus r_x; \qquad M(x \oplus n) = r_x \oplus m$$

Here, S was the s-box, r_x was a random value, S' was table of masked values and M was a table of mask values. Split mask was designed so as to resist first order DPA, however there were certain weakness like small key size, the power consumption still remained dependant on the target bit. All these weakness were revealed during the attack presented[9]. There attack was successful to reveal the complete key with only 400 power traces. A triple masking concept was presented[10], where they masked both exponent and the message. Besides, they used some inefficacy operations in between the computations which were meant to cover up the original operations. Their work was implemented in RSA cryptosystem as a technique to resist DPA attacks. A formal proof of the masking approach to resist DPA attacks was presented[11]. They proved that the information extracted from the leakage of a masked model will be negligible for higher order DPA attacks.

Arithmetic masking in asymmetric cryptosystems is popularly known as blinding[12]. As most of the asymmetric cryptosystems were based on arithmetic operations like modular exponentiation, they generally used arithmetic masking or blinding to thwart DPA attacks. In such cryptosystems, the modular exponentiation was the most sophisticated operation as it involves both message and exponent or the secret key. The power consumption details during these operations were used by the attackers for mounting DPA attacks. For asymmetric cryptosystems, blinding could be executed in message as well as the exponent or both. A message blinding method was presented[13], which required no multiplicative inversion. In general, all the message blinding methods required an additional multiplicative inversion in their computation. This additional operation had a complexity almost similar to that of modular exponentiation. In their work they have shown how message can be blinded in RSA without the inversion operation. A new direction of masking known as inner product masking[14] which could split the exponent into several first and then mask them individually has recently appeared in literature. However, it has not been implemented in asymmetric cryptosystem yet. It would be worth mentioning that message or exponent blinding alone is not enough to resist power analysis attacks or any side channel attacks[15]. Hence, it is of urgent need to come up with more sophisticated blinding techniques for asymmetric cryptosystems to thwart such attacks.

We present a blinding technique for RSA to resist power analysis attacks. This technique first partitions the secret exponent into multiple parts and then blinds them individually with an independent random data. To enhance the security, the message is also blinded prior to the fundamental computation. With both message and exponent blinding, the proposed technique would be able to resist power analysis attacks to a very large extent.

2. Partitoned-Exponent Blinding

It implements a two dimensional blinding where both the message and the exponent has been blinded with an independent random variable. However, prior to exponent blinding, it had been partitioned into multiple segments and then individually blinded. For clarity in understanding the proposed work we first explain each of these dimensions and then show how they were implemented in modular exponentiation. The proposed work has been implemented in 1024 bit, 1536 bit and 2048 bit RSA.

2.1 Blinding the Message

For blinding the message *(m)*, we have generated a random independent variable "r_m" which is first multiplied with the message. The outcome of this multiplication undergoes a modulo with "*n*" where *(e, n)* makes the key pair for modular exponentiation. This masked message *(m')* is then used for further computation.

$$r_m = random(\square)$$
$$m' = (m * r_m) mod\ n$$

2.2 Partitioning the Exponent

For partitioning the exponent into multiple parts we have first generated a random number within a range $r_p \in (x, y | \square | x < y < e)$. The range within which the random number r_p is generated depicts the number of partitions of the exponent. Larger is the value of r_p smaller is the number of partitions and vice versa. Algorithm 1 below states the partitioning in detail.

2.3 Modular Exponentiation with Partitioned Exponent Blinding

Each of the partitions generated from the exponent (e) is then individually blinded with the same independent random variable (r_m) which was used for blinding the message. However, unlike message blinding we have used the Euler's Totient*(z)* for exponent blinding. The product of the random variable (r_m) and the Euler's Totient*(z)* is added with each partition of the exponent to generated a new blinded exponent partition $\square(e\square_i')$. The number of partitions generated defines the number of iterations for which the exponent blinding needs to be performed. Mathematically, for every partition "e_i" blinding can be implemented by,

$$e_i' = e_i + \square(r\square_m \times z)$$

After the message and exponent partitions have been blinded, they then undergo modular exponentiation individually and later combined to get the masked cipher text. Algorithm 2 below shows the complete modular exponentiation method with partitioned exponent blinding. The possibility of computing modular exponentiation in as implemented in Algorithm 2 was due the property as shown,

$$c^d mod\ n = (c^{d1} mod\ n * c^{d2} mod\ n * .. * c^{dn} mod\ n) mod\ n$$

Algorithm 1: Partitioning the Exponent

Input: e, x, y where $x < y < e$

$$\sum_{i=0}^{t-1} e_i = e$$

Output: $e_1, e_2, ... e_t$ such that

Initialization:

Sum = 0, Diff = 0, e' = 0

Exp [ ] , an array to store the partitions

Procedure:

```
While (Sum<= e ) do
        r_p =  random (x . . y)  // generates a random number between x and y
        Sum = Sum + r_p
        if  ( Sum > key ) then
                Diff = e - e'
                Exp[ i ] = Diff;
        else
                e' = e' + r_p;
                Exp[ i ] = ran';
Return Exp[ i ]
```

End Procedure

Algorithm 2: Modular Exponentiation with Partitioned Exponent Blinding
Input: p, q, m, e
Output: m^e mod n

Initialization:

```
e' = 0
Result = 1        // For computing intermediate results
MResult = 1       // For computing the final result
```

Pre-computation:

```
n= p x q;
z = (p-1) x (q-1)
Exp [t]← Algorithm 1with "e" as the input which generates "t" partitions
r:= a random number
```

Procedure:

```
m' = (m * r) mod n                  // Blinding the message
fori from 0 to t do
        e' = Exp[i]
        e''= e' + (r * z)           // Blinding every partition of e
        Result = m'^e'' mod n       // Computing individual modular exponentiation
        Result  =Result * Result
end for

MResult = (Result) mod n            // Final Modulo Operation
Return MResult                      // Blinded cipher text
```

End Procedure

Before decryption the effects of the blinding needs to be removed so that the original cipher text can be retrieved. It requires few additional steps shown in Algorithm 3 which will generate the original cipher text. Figure 1 shows a visual interpretation of the complete proposed work.

3. Results and Discussions

The result was implemented in Mupad, a tool in Matlab for all the analysis. As, we had considered RSA cryptosystems for our experimentation purpose, we used the standard test data provided by PKCS v2.1.10. PKCS v2.1.10 is provided by RSA Laboratories with data like key, plain texts, cipher texts, primes, modulo etc. used in computing 1024, 1536 and 2048 bit RSA. All the results mentioned in this paper have been analysis with these standard data.

3.1 Security Analysis

Masking has been a popular approach to resist power analysis attacks in symmetric cryptosystems like DES and AES. Arithmetic masking in asymmetric cryptosystems is also known as blinding. For asymmetric cryptosystems involving modular exponentiation, there are prospects of blinding both message and the exponent or secret key. With blinding, the actual data remained hidden throughout the computation. As a result, the power consumption details would be correlated to the blind data rather than the actual data. However, if the exponent is very large (as in case of decryption) then, simple arithmetic blinding would also increase the size of the exponent increasing the computation complexity as well as time.

A general blinding approach may resist first order DPA, but might be vulnerable to higher order DPA attacks. The strength of blinding hence lies in the number of shares to

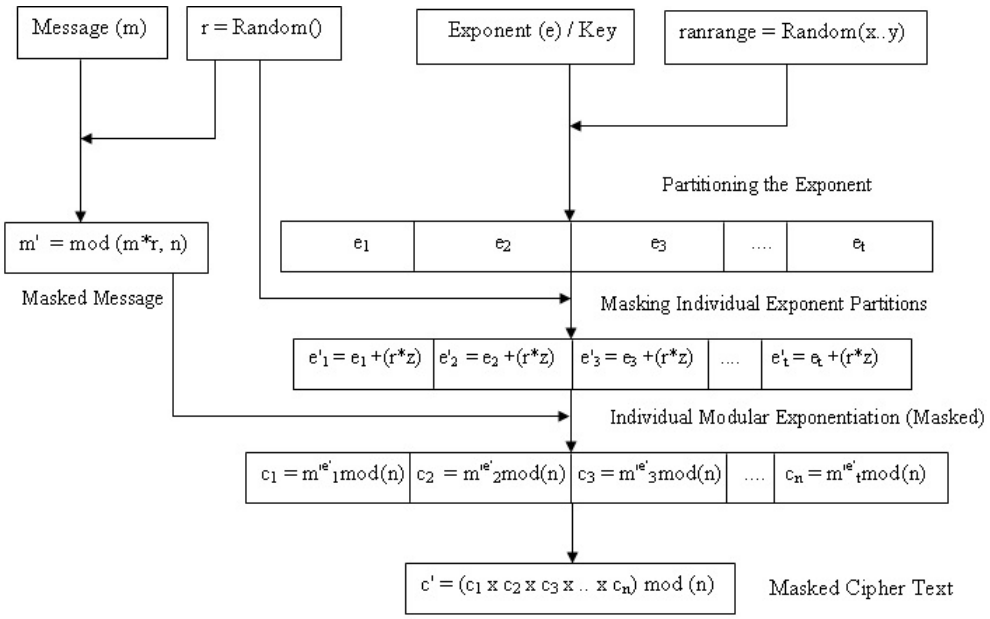

Figure 1. Modular exponentiation with partitioned exponent blinding.

Algorithm 3: Removing the effects of Blinding prior to Decryption
Input: Blinded cipher text
*Output:*Original cipher text (Removing the blinding effects)

Initialization:

c' ←blinded cipher text from Algorithm 2
Exp [t]← "t" partitions of the exponent from Algorithm 1
r ← random number from Algorithm 2
IntRes = 1, c=1 // Variables to hold intermediate results

Procedure:

fori from t-1 down to 0 do
e' = Exp [i]
IntRes = $\left(\frac{1}{r}\right)^{e' mod\ n}$

c = *(c * IntRes) mod n*
end for
Return c

End Procedure

which the target data may be divided. If the target data was divided to *"d"* shares, it would be vulnerable to *"d+1"* th order DPA providing resistance to a large extent. In this work we have partitioned the target data which is mainly the secret exponent *(e)* into *"n"* shares where $n \in 6, 8, 11$ &*14*. The value of *n* can also exceed the ones that we have considered however it would need additional memory to store these partitions.

Algorithm 1 used a random independent variable within a range for partitioning the exponent. As this value changed, so would the partitions breaking the uniformity in consecutive implementations. Also, each of these partitions has been blinded with another random number which reduced the dependency of the operation

on the actual data. That is, as all the operations were done on the blinded data, the actual data remained intact and independent. These features meet most of the theoretical aspects for resisting power analysis attacks. Also, while the entire computation was done on the blinded data, the resultant power traces when collected would also be based on these blinded data. Hence, correlation between the data and power traces would not leak information about the actual bits of the target data challenging successful DPA attacks.

Further, in this work we had not only blinded the exponent but also the message. With these two dimensional blinding (exponent and message) we could achieve a higher order resistance from DPA attacks. Besides, removing the effect of the random variable before decryption was also very simple, keeping the overall complexity same as general modular exponentiation. The additional complexity for blinding would be very less and hence could be negligible as it involved only addition and multiplication.

3.2 Computation Time

Although from security perspectives the proposed work meets all the theoretical aspects of resisting DPA attacks, it was necessary to analyze its impact on computation time. For, this purpose we have computed the average computation time take by the proposed approach in multiple partitions for three key variances of RSA i.e. 1024, 1536 and 2048 bit RSA. As mentioned before we considered four different types of partitions 6, 8, 11 and 14 partitions of the exponent. In all these cases we found for all the variances of RSA, as the number of partitions increased, so did the computation time. Table 1 with Figures 2, 3, 4 and 5 shows the impact of partitions in all variances of RSA in the average computation time.

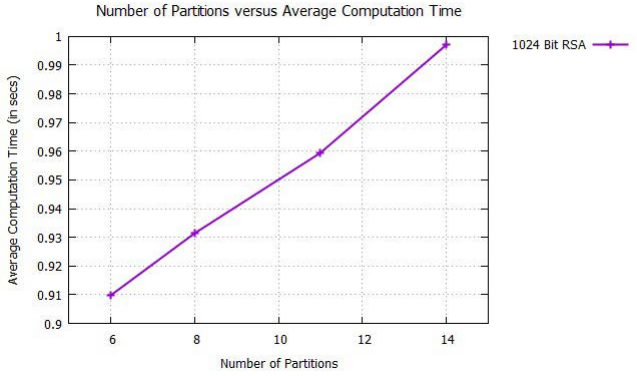

Figure 2. Partitions versus average computation time for 1024 Bit RSA.

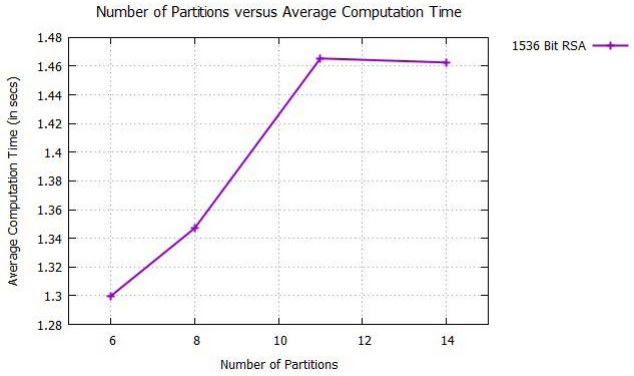

Figure 3. Partitions versus average computation time for 1536 Bit RSA.

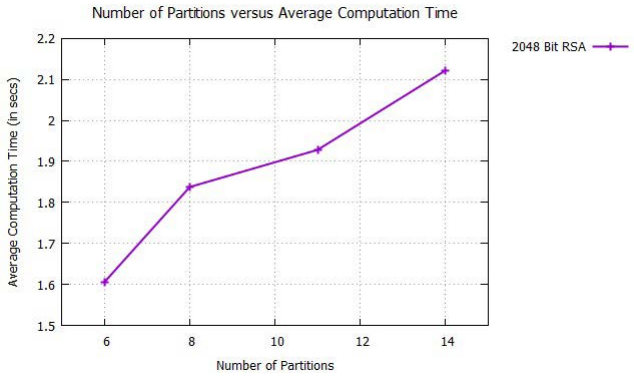

Figure 4. Partitions versus average computation time for 2048 Bit RSA.

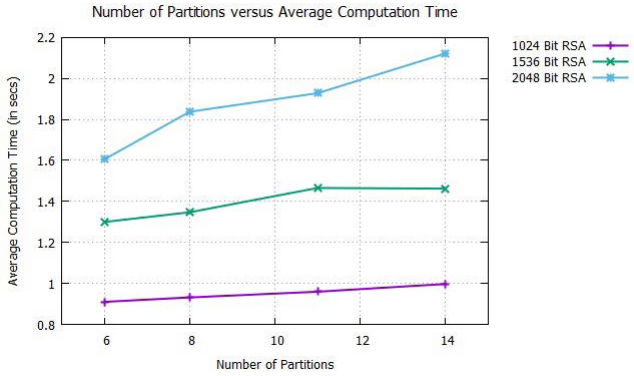

Figure 5. Partitions versus average computation time for all variance of RSA.

Table 1. Average computation time in partitioned-exponent blinding

No. of Partitions	Average Computation Time (in seconds)		
	1024 Bit RSA	1536 Bit RSA	2048 Bit RSA
6	0.9098	1.2998	1.606
8	0.9314	1.347	1.8376
11	0.9594	1.4654	1.9282
14	0.997	1.4626	2.1218

To check the performance of the proposed work we also computed the conventional blinding without partitioning the exponent. The results showed that, partitioned-exponent blinding had less average computation time for 1024 and 1536 bit RSA. However, in case of 2048 bit RSA, the average computation time increased by 0.43% which could be negligible. Hence, we could conclude that the proposed work did not affect the performance of conventional blinding technique indeed it performed better than the earlier. Table 2 and Figure 6 illustrates the above facts in details.

Table 2. Comparison of exponent blinding with partitioned-exponent blinding

RSA Size	Average Computation Time (in seconds)	
	Exponent Blinding	Partitioned-Exponent Blinding
1024	1.047	0.9494
1536	1.416	1.3937
2048	1.8654	1.8734

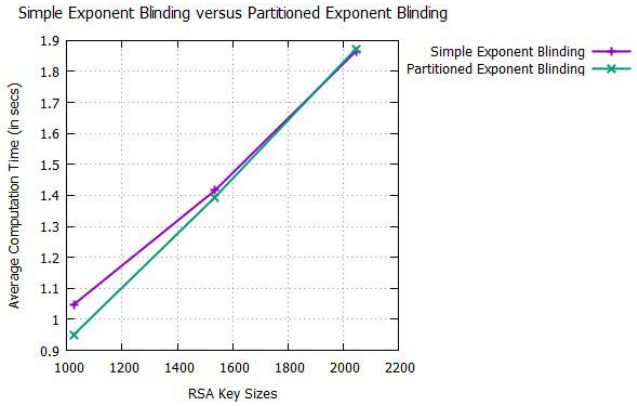

Figure 6. Exponent blinding with partitioned-exponent blinding.

4. Conclusion

The paper presented an approach to compute modular exponentiation with partitioned-exponent blinding. Here, both the message as well as the exponent was blinded. Unlike conventional blinding techniques, the proposed work first partitioned the exponent into multiple segments and then blinded them individually. Thereafter, each of these blinded exponents with the message computed modular exponentiation and finally combined to derive the desired blinded cipher text. The paper also presented the additional steps to remove the effects of blinding prior to decryption. The proposed approach was implemented in 1024, 1536 and 2048 bit RSA with multiple plaintexts. The results showed that as the number of partitions increased so did the average computation time. However, in comparison with conventional blinding, the proposed work had less average computation time for 1024 and 1536 bit RSA. But, for 2048 bit RSA, the average computation time was slightly more. The security and computation time analysis showed that the proposed approach can be contribute largely in resisting power analysis attacks on modular exponentiation based cryptosystems like RSA.

5. Acknowledgement

The authors would like to thank MeitY, Government of India for all the support.

6. References

1. Kocher P, Jaffe J, Jun B. Differential power analysis. Proceedings of Annual International Cryptology Conference; USA: Springer Berlin Heidelberg. 1999. p. 388–97. Available from: Crossref
2. Sridhar KP, Saravanan S, Sai RV. Counter measure against side channel power analysis attacks in Cryptographic Devices. Indian Journal of Science and Technology. 2014Apr; 7(4):15–20.
3. Coron JS, Goubin L. On boolean and arithmetic masking against differential power analysis. Proceedings of International Workshop on Crytographic Hardware and Embedded Systems (CHES); USA: Springer Berlin Heidelberg. 2000. p. 231–7. Available from: Crossref
4. Goubin L. A sound method for switching between boolean and arithmetic masking. Proceedings of International Workshop on Crytoghraphic Hardware and Embedded Systems (CHES); France: Springer Berlin Heidelberg. 2001. p. 3–15. Available from: Crossref
5. Coron J, Tchulkine A. A new algorithm for switching from arithmetic to Boolean masking. Proceedings of International Workshop on Crytoghraphic Hardware and Embedded Systems (CHES); Germany: Springer Berlin Heidelberg. 2003. p. 89–97.
6. Standaert FX, Peeters E, Quisquater JJ. On the masking countermeasure and higher order power analysis attacks. Proceedings of IEEE International Conference on Information Technology: Coding and Computing; USA. 2005. p. 562–5. Available from: Crossref
7. Mangard S, Popp T, Gammel BM. Side channel leakage of masked CMOS gates. Proceedings of Cryptographer's Track at the RSA Conference; USA: Springer Berlin Heidelberg. 2005. p. 351–65. Available from: Crossref
8. Gebotys CH. A split-mask countermeasure for low-energy secure embedded systems. ACM Transactions on Em-

bedded Systems. 2006 Aug; 5(3):577–612. Available from: Crossref
9. Coron JS, Kizhvatov I. Analysis of split mask countermeasure for embedded systems. Proceedings of 4th Workshop on Embedded Systems Security; France. 2009. p. 3–12. Available from: Crossref
10. Jin JF, Lu EH. Resistance of DPA on RSA smartcard. Proceedings IEEE 5th International Conference on Information Assurance and Security; China. 2009. p. 406–9. Available from: Crossref
11. Prouff E, Rivain M. Masking against side channel attacks: A formal security proof. Proceedings Annual International Conference on the Theory and Applications of Cryptographic Techniques; Greece: Springer Berlin Heidelberg. 2013. p. 142–59. Available from: Crossref
12. Mangard S, Oswald E, Popp T. Power analysis attacks: Revealing the secrets of smart cards. Springer Science and Business Media. 2008; 31:163–285.
13. Kim H, Han DG, Hong S, Ha J. Message blinding method requiring no multiplicative inversion for RSA. ACM Transaction on Embedded Computing Systems. 2014 Nov; 13(4):80–9. Available from: Crossref
14. Balasch J, Faust S, Gierlichs B. Inner product masking revisited. Proceedings Annual International Conference on the Theory and Applications of Cryptographic Techniques; Bulgaria: Springer Berlin Heidelberg. 2015. p. 486–510. Available from: Crossref
15. Schindler W. Exclusive exponent blinding is not enough to prevent any timing attack on RSA. Journal of Cryptographic Engineering. 2016 Nov; 6(2):101–19. Available from: Crossref