Implementation of RSA Algorithm Secure against Timing Attacks using FPGA

Muhammad Aqeel Aslam
Department of Electrical (Electronics) Engineering
Swedish College of Engineering & Technology
Shahbaz Pur Road Rahim Yar Khan, Pakistan

Ahmad Bilal
Department of Electrical (Electronics) Engineering
Swedish College of Engineering & Technology
Shahbaz Pur Road, Rahim Yar Khan, Pakistan

Abstract **–** Data security is an important aspect of information transmission and storage in an electronic form. Cryptographic systems are used to encrypt such information to guarantee its security. To retrieve such information, the encrypted form must be first decrypted. One of the most popular cryptographic systems is the RSA public key crypto system. The larger the RSA public modulus size, the stronger will be the RSA cryptosystem. Unfortunately, the RSA is extremely vulnerable to timing attacks which can deduce the private RSA exponent due to regularity of operations in the straight forward implementation of exponentiation using the square and multiply method or its variants. Timing attacks constitute a major threat to the all systems using RSA and hence, implementations must be protected. The work reported here proposes Secure Implementation of RSA algorithm against timing attacks. This implementation is done using Verilog HDL and targeting Xilinx FPGA devices.

The "Secure Implementation of RSA Algorithm against Timing Attacks" is done by using the output of a Random Number Generator, and blinding the exponentiation operation. The Exponentiation is done by using Montgomery Reduction scheme which is the fastest known technique of exponentiation. It does not require division, as the division in hardware is quite complex. The random number generator is implemented using a Barrel shifter and to nullify the effect of random number we implemented the Extended Euclidean Algorithm which provides modulo inverse of the generated random number. To hide the operation time we multiply the random number with the private key. After exponentiation the result is then further multiplied with the inverse of the generated random number. By doing so the timing of the algorithm is changed and hence, the desired result is achieved.

Index Terms – Modular Multiplication, Exponentiation, Random Number Generator, Timing Attacks, Blinding.

I. INTRODUCTION

Encryption is a well recognized technique for the protection of data and information. It is used effectively to protect sensitive data. The transfer of data from one form to another form, which is unreadable without the secret key, is called encryption. Several techniques have been used for many years. Cryptographic systems are classified into two main categories secret key cryptosystem and public key cryptosystem. Only one key is involved in Secret key cryptosystem. This key is used for both encryption and decryption. However, public key cryptosystem uses two different keys one for encryption and other one for the decryption.

Day by day the significance of FPGA (Field Programmable Gate Array) is increasing. FPGAs are playing very important role in commercial area and research area as well. The technology of FPGA is more affordable as compared to ASICs. FPGAs are reconfigurable platforms. The choice of reconfigurable platforms for cryptographic algorithm appears to be practical and it provides high speed in applications.

In 1978 RSA algorithm was first developed by Rivest, Shamir and Aldeman. RSA is an example of public key algorithm. If the modulus size is large, the algorithm is secure. This algorithm is computationally intensive and it operates on very large integers. RSA algorithm can be used for encryption / decryption and digital signatures. RSA is the most popular method for the public key cryptosystems. The key size determines the security of RSA algorithm. The larger is the key size, more is the security.

Side channel attacks can destroy any cryptosystem. Cryptosystem takes slightly different amount of time to process different inputs. In side channel attach the information can be retrieved from the encryption device that is neither the plaintext to be encrypted nor the cipher-text resulting from the encryption process. In a side channel attack an attacker attempts to compromise a crypto system by analysing the time required to execute each operation. Each logical operation requires some time for execution. An attacker can work backward to find the input by the precise measurement of time. It is generally agreed that RSA is secure from direct attack. Performance characteristic of the cryptosystem depends upon the encryption key and data input.

Timing attacks are a form of side channel attacks. Timing attacks are based on measuring time it takes for

a unit to perform operation. Generally, encryption system requires different processing time for different inputs. Some attacks can exploit the timing measurements to find the entire secret key. Timing attacks can be used potentially against any cryptosystem including symmetric functions.
Calculating variances is fairly simple. It provides a improved way to make out correct bits. The number of samples will allow recovering the information by the properties of signal and noise. If noise is too much, than, more samples will be needed to find the actual information and the secret key.
The ability to resist side channel attacks the designer of cryptographic modules should use any of the following techniques in order to make the cryptosystem more reliable. Every operation performed by the module should be data independent in their time consumption. Whenever, different sub operations are performed according to the inputs, they should require same number of clock cycles. Time required to perform operation should be fixed for every piece of data, this will exclude the all possibilities of the timing attacks.

This technique is also used for the prevention of side channel attacks. In this technique we make sure that all the exponentiations takes the same amount of time before providing the result. This is the simplest way of preventing data from side channel attacks but it degrades the overall performance of the algorithm as each operation requires different amount of time to execute but if we implement this technique each and every operation would require same amount of time.
Random delay technique provides better performance of the algorithm as compared to the constant exponentiation delay. In this technique we add random delay to the exponentiation in order to confuse the timing attacks. Kocher points out in his paper that if we do not add enough noise then the observer can still succeed by collecting additional measurements which were made for the compensation of the random delay.
Blinding is far better technique than the other stated two techniques as it is the most widely accepted method in the defence of RSA. Blinding prevents the side channel attacks on encryption system. RSA blinding introduces the randomness. Blinding makes the timing information unusable.

.

II. PROPOSED ARCHITECTURE

Our proposed structure of Secure Implementation of RSA Algorithm against Timing Attacks consists of the following steps. First we implemented random number generator using barrel shifter. The random number generated using barrel shifter comprises of 32 bits. A barrel shifter shifts the data in a digital circuit. It shifts a particular number of bits in one cycle.
Extended Euclidean Algorithm implemented to find out the modulus inverse of the generated random number. The effect of the random number has to be vanished from the original data. In order to remove the effect of the generated random number, we calculated modulo inverse of random number. The Extended Euclidean Algorithm is derived from Euclidean Algorithm. Highest Common Factor find out through Extended Euclidean Algorithm. The greatest common divisor and inverse modulo number is obtained by using Extended Euclidean Algorithm.
Modular Exponentiation was developed by using Montgomery Multiplication along with Wallace Tree Reduction scheme. Montgomery multiplication was chosen because it does not involve division. Additional module of division is omitted in Montgomery Multiplication and Wallace tree reduction is used in order to shorten the addition steps. We introduce random time to make the timing information unusable.

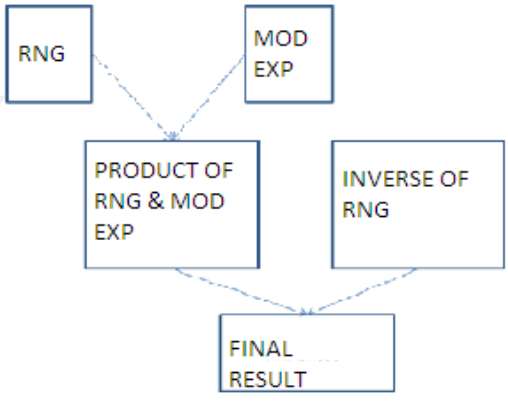

Figure1. Proposed Architecture

III. IMPLEMENTATION

Addition is the basic part of the Arithmetic Logic Unit (ALU), which is used in all other operations. The overall performance depends upon the speed of addition. We started our implementation by using Ripple Carry Adder. After some further literature review it came into notice that Ripple Carry Adder has some drawbacks in terms of delay. Therefore, we switched on to Carry Look Ahead Adder, which has less delay while propagating carry. As already mentioned, that addition operation is the most important and fundamental and it plays a major role in all operations. The delay effect is minimize by CLA. First, we made a Carry Look Ahead adder of 4 bits, we instantiate it to make a 16 bit adder. We instantiate these 16 bit adder to form the Carry Look Ahead Adder of 64 bits and then these 64 bit adders cascaded to make the required adder of 512 bits.

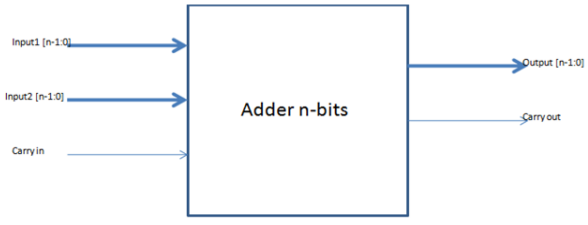

Figure 2 Addition of n-bits

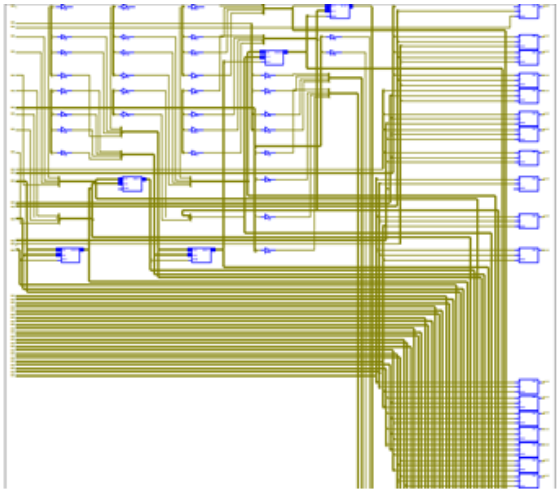

Figure 3 RTL view of Addition

The results of the Ripple Carry Adder and Carry Look ahead Adder has been shown in the following figure.

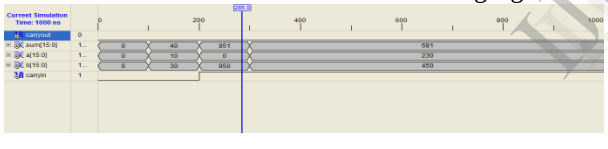

Figure 4 16 bit Ripple Carry Adder

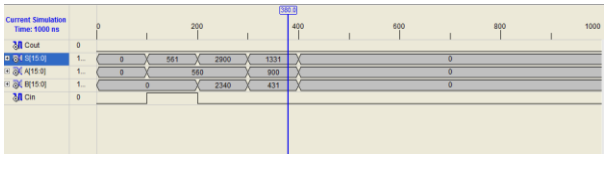

Figure 5 16 bit Carry Look Ahead Adder

Multiplication was done by suing Shift and Add method. In Modular Multiplication this requires an additional module of division. Division is the most complex part of the hardware. We switch on to Booth multiplier and 16 bit multiplication results were shown in the following figure. Booth multiplication also has some limitations. Since we are working on the security of the system, so we have already loss some of the speed. There are two methods which are quite useful in Modular Arithmetic. They are Wallace Tree Reduction and Dadda Tree Reduction.

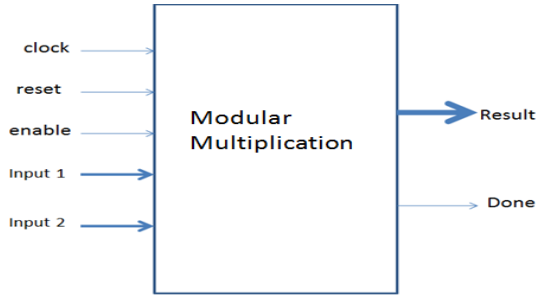

Figure 6 Modular Multiplication

Therefore, we proffered Wallace Tree Reduction Method in order to gain some speed in the RSA Algorithm.

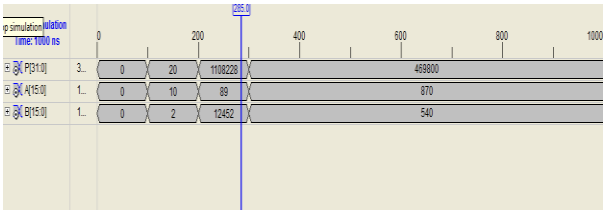

Figure 7 Shift and Add Multiplier

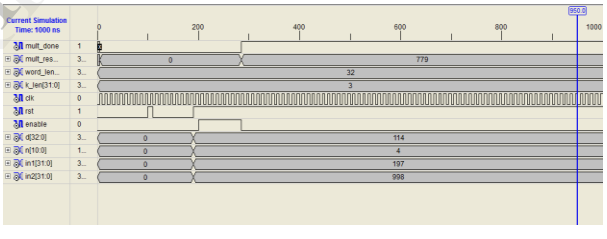

Figure 8 Modular Multiplication

Square and multiply algorithm is used effectively to calculate Modular Exponentiation. In most of the cryptographic protocols this algorithm is used.

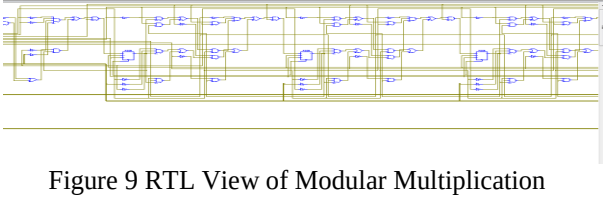

Figure 9 RTL View of Modular Multiplication

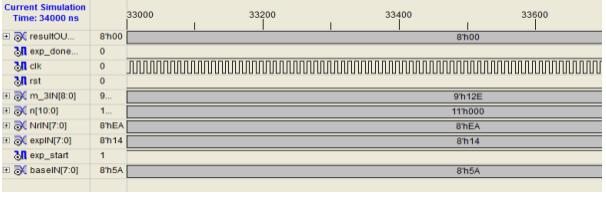

Figure 10 Simulation Result of Modular Multiplication

Random number plays a vital role in the encryption / decryption. RNG changes the exact time of the operation. It changes the time of operation such sufficiently that the attacker cannot find the exact operation time. Pseudo Random Number Generators are implementing in hardware. For practical use we established 32 bit random number. It provides 2^{32} combinations which are sufficient and have a wide range. We use Barrel shifter for generation of random number. Barrel Shifter has multiple usages in digital design systems. These registers can be used in Encryption / Decryption, Digital Signal Processing, Wireless Communications, Data Integrity Checksum, Data Compression, Random Numbers Generation, Direct Sequence Spread Spectrum (DSSS), Scrambler / Descrambler and Optimized Counters. The following figure shows the block diagram of the Random Number Generator.

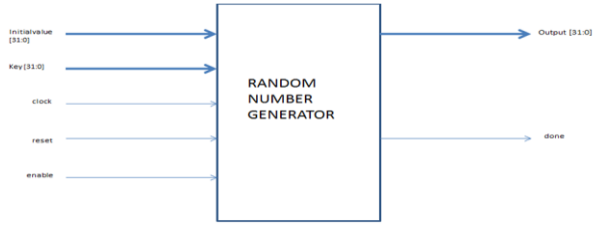

Figure 10 Random Number Generator

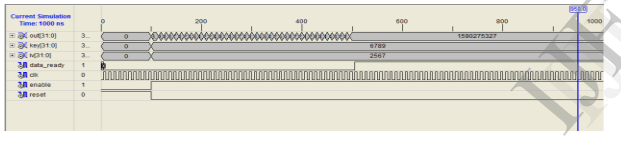

Figure 11 Simulation Result of 32 bit RNG

IV. RESULTS

The following diagram shows the simulation result of the proposed architecture of RSA Algorithm

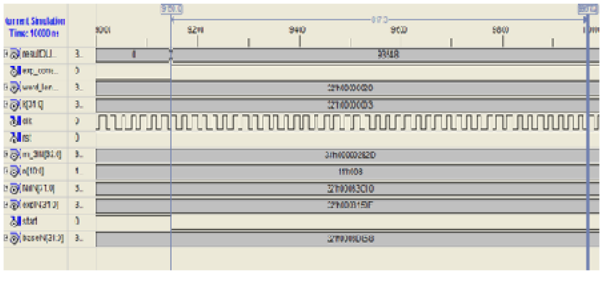

Figure 12 Simulation Result of Proposed Architecture

The synthesis result for the implementation of Random Number Generator is expressed in the following table:

Logic Components	Total	Used
Adder / Sub tractors	3	3-bit Adder (1) &32-bit a (2)
Registers / Flip Flops		101
Xors	3	32-bit Xor

Table 1 Synthesis Results of Random Number Generator

The following table shows the clock cycles of each operation. A random number of 32 bits was generated.

Initial Value	Key	Generated Random Nu	Hamming wei	Clock C
32'h EA	32'h51	32'hF8EDBAAD	21	40
32'h 2654	32'hB3	32'hC1F6A7CA	18	40
32'h1987	32'h80	32'h3E6A5A38	16	40
32'h6DD61	32'h30	32'h7F81C7E8	18	40
32'h48FFEA	32'h23BA	32'h25E38CCE	16	40
32'hBC6146	32'h96B	32'h6B5F7A09	18	40
32'h3C413	32'hCF3	32'h7315FAE	17	40
32'h9C4800	32'h640	32'h3BE75E9C	20	40
32'h8180209	32'hECA	32'h6E0F29B3	17	40
32'h48FA533	32'h5B38	32'h6BA931A4D	18	40
32'h499602D	32'h4996	32'h4668CD10	12	40

Table 2 Clock Cycles of Random Number Generator

The above table indicates that when we change the initial value or key, anew random number is generated but the number of cycles for each operation remains the same. In every execution a new random number is attained. The every generated number is different from

the previous number, hence introducing the random number.
The following table shows the number of clock cycles in each operation of multiplication.

Multiplicand	Multiplier	Multiplication Result	Clock Cycles
135790	864203	11220	100
2345678	456789	12868	98
2468761	98791	8604	101
147952	290541	737233	101
246789	123678	385537	99

Table 3 Number of clock cycles for Multiplication

The following table shows number of clock cycles involved in each operation of Exponentiation.

Exponentiation Result with blinding	Clock Cycles
90494	99
89971	94
45314	95
46800	94
43310	94

Table 4 Number of Clock Cycles for Exponentiation

The following table indicates the number of clock cycles when RSA simulated.

Base	Exponent	Result	Clock Cycles
161984	161998	43310	1023ns
236789 975310	456721 864209	23334 64973	1021ns 1020ns
194023	196507	51104	1020ns
251992	201999	93548	1020ns

Table 5 Number of Clock Cycles for RSA

The following table shows the clock cycles for each operation when blinding is performed on the RSA.

RNG	Base	Exponent	Clock Cycles
YES	251992	201999	1020ns
YES	161984	161998	1200ns
YES	236789	456721	1197ns
YES	975310	864209	1200ns
YES	194023	196507	12020ns

Table 6 Number of Clock Cycles for Blinded RSA

The following table indicates the comparison between the above two tables.

Base	Exponent	Clock Cy	Base	Exponent	Clock Cy	Difference
251992	201999	1120ns	251992	201999	1020ns	100ns
161984	161998	1200ns	161984	161998	1023ns	177ns

Table 7 Difference between Clock Cycles

V. CONCLUSION

In this research we have developed a hardware model of RSA cryptographic system which provides more security to our cryptographic system. The thesis begins with the introductory description of the cryptographic systems. RSA technique is the most popular and widely method, which is used for the encryption. The goal set in this thesis assignment was fulfilled. The thesis presented the enhancement of the security. The implemented RSA algorithm is much more secure. The above thesis was under taken in order to develop a secure Implementation of RSA algorithm against Timing Attacks.
The methodology implemented for the security of RSA was blinding. Blinding was used in order to avoid the timing attacks and improve the security of the general RSA algorithm. Montgomery multiplier was used as it is fastest multiplication technique till now

REFERENCES

[1] Implementation of Efficient Modular Exponentiation on Reconfigurable Platforms, August 2008, Kashif Latif, Department of Electronic and Power Engineering, College of

Marine Engineering (PNEC), Karachi, National University of Sciences and Technology, Rawalpindi

[2] Timing Attacks on Implementation of RSA, Diffie-Hellman, DSS and Other systems, Paul C. Kocher, Cryptography Research, Inc. 607 Market Street, 5th Floor San Francisco, CA 94105, USA

[3] Attacks on the RSA Cryptosystem, VaibhavVaish, Department of Computer Science & Engineering, Indian Institute of Technology, New Delhi, India, 1999

[4] Implementing a 1024 bit RSA on FPGA, 2003, Jing Lu and Wan Qian, Reconfigurable Network Group, Applied Research Lab, Department of Computer Science and Engineering, Washington University in St. Louis

[5] Timing Attacks on software Implementation of RSA, Project Report, Harshman Singh 903-40-5260, June 07, 2004

[6] Efficient Hardware Design and Implementation of AES Cryptosystem, Pravin B. Ghewari et al. International Journal of Engineering Science and Technology, Vol. 2(3), 2010, 213-219

[7] FPGA Implementation of RSA, Public-Key Cryptographic Coprocessor, MiCE Department, Faculty of Electrical Engineering, Universiti Teknologi Malaysia, 8 13 10 UTM Skudai, Johor, Malaysia

[8] Implementation of RSA Cryptosystem Using Verilog, Chiranth E, Chakravarthy H.V.A, Nagamohanareddy P, Umesh T.H, Chethan Kumar M, International Journal of Scientific & Engineering Research Volume 2, Issue 5, May-2011 1, ISSN 2229-5518, IJSER © 2011

[9] The Protection of Modular Exponentiation Operands from Their Reconstruction by Simple Power Analysis, Akram A. Moustafa and Saleh Alomar, Proceedings of the World Congress on Engineering and Computer Science 2009 Vol I, WCECS 2009, October 20-22, 2009, San Francisco, USA

[10] Assorted Attacks on the RSA Cryptographic Algorithm Edmond J. Murphy, Boston College Computer Science Department Professor Howard Straubing May 9, 2005

[11] An Efficient VLSI Architecture for Rivest-Shamir-Adleman Public-key Cryptosystem, Department of Electrical Engineering Tamkang University, Tamsui, Taiwan 251, R.O.C., Tamkang Journal of Science and Engineering, Vol. 7, No 4, pp. 241_250 (2004)