International Journal of Current Engineering and Technology
E-ISSN 2277 – 4106, P-ISSN 2347 - 5161

Available at http://inpressco.com/category/ijcet

Research Article

Design of Commutative Cryptography Core with Key Generation for Distributed FPGA Architecture

R. Ambika[Á*], S. Ramachandran[Ḃ], K.R. Kashwan[Ċ]

[Á]Vinayaka Missions University, Salem, India
[Ḃ]Dept. of ECE, SJB Institute of Technology, Bangalore, India
[Ċ]Dept. of ECE, Sona College of Technology, Salem, India

Accepted 05 Oct 2014, Available online 11 Oct 2014, **Vol.4, No.5 (Oct 2014)**

Abstract

Data security during communication is one of the predominant issues in modern multiple transceiver based communication. In this paper, we have presented a highly robust commutative cryptography core for distributed FPGA architecture called commutative RSA with Key generation. The commutative RSA algorithm has been developed using parallelization of Montgomery multiplication with high radix exponential modular multiplication scheme to suit FPGA implementation. The architectural design not only ensures authentication among multiple transceivers or MIMO but also reduces overheads caused due to key exchange process. The CRSA algorithm with key generation has been realized with multiple FPGA cores using VHDL. The design has been simulated using Modelsim 5.5e and synthesized using Xilinx Design Suite 14.3 targeted on Virtex-5, xc5vfx70t-2ff1136 FPGA and Vivado 12.3, Virtex-7, xc7vx330tffg1157-2L. The results obtained illustrates that the proposed architecture offers high computational efficiency with minimum overheads and memory occupancy even at higher frequency rate. The designed system works at 292 MHz in Vivado and at 199 MHz in ISE 14.3 platforms and would be compatible with a standard real time data communication hardware interface.

Keywords: *FPGA, Linear Feedback Shift Register, MIMO, Montgomery multiplication, RSA, Security*

1. Introduction

High speed data communication in multiple transceivers and Multiple Input Multiple Output (MIMO) applications demand a highly robust and secure system model that could facilitate security. The approach of cryptography plays a potential role in ascertaining security for Multiple Input Multiple Output based on RSA algorithm and plays a significant role with public key cryptography. Confidentiality, authenticity, data integrity and its non-repudiation are the major requirements in security. A number of approaches and systems have been advocated and developed for ensuring data security in competitive multiuser scenario and among them, public key cryptosystem has been recognized as one of the optimum solution (William Stallings, 2003).The public key cryptography is found to be better as compared to symmetric-key cryptosystem for achieving aforementioned objectives.

A number of public key cryptosystems have been developed, of which RSA cryptosystem (R. L. Rivest, *et al*, 1978) established itself as one of the most optimum approaches which is sufficient in delivering high performance in multi-party communication scenario with distributed processors. RSA can be an effective solution for hardware assisted applications of secure communication. In RSA algorithm, the predominant process is the computation of modular exponentiation by repeated modular multiplication process. The Montgomery modular multiplication algorithm (SchneierBruce, *et al*, 1997), (Alan Daly, *et al*, 2001) uses this technique and is suited for hardware realization. Public key algorithms are based on one way cryptosystem functions and have limitations of key exchange overheads. Therefore there is an inevitable need to develop schemes that could deliver the commutative kind of behaviour. According to commutative nature, the order in which RSA encryption is done would not influence the decryption if it is done in similar way or in a sequence. If this unique approach is incorporated with normal RSA, it could be a milestone for optimizing RSA implementation with MIMO transceiver based communication. This paper explores the commutative RSA (CRSA) algorithm with key generation for realization using distributed FPGA cores.

The rest of this paper has been organized as follows. In Section 2, the prior work has been discussed. Section 3 discusses the Mathematical background which is followed by Section 4 that presents Commutative RSA Cryptosystem core details. Section 5 discusses the experimental study and the results obtained. The last section presents conclusion and the scope for future work.

2. Related Work

(G.D. Sutter, *et al*, 2011), developed a system with enhanced Montgomery's multiplication for performing the LSB first and the MSB first algorithms alternately. Their developed system model make use of the digital serial

*Corresponding author: **R. Ambika**

scheme, Carry Save Adder (CSA)and Carry Skip Adder. The digital serial scheme is used for performing Montgomery multiplication. The CSA is used for representation of intermediate multiplication and the carry skip addition is used for reducing the critical path. Another uniqueness of this work is the pre-computation of the quotient value in Montgomery's iteration for speeding up the operation frequency. Even if this system made a better effort, it could not address the multiparty MIMO kind of situation. Further, the robust commutative approaches were not explored in this work.

(Z. Chen, *et al*,2011) proposed a parallel system implementation of Montgomery multiplication for distributed multiple cores. This approach got success in achieving balanced partitioning of the task. Additionally, the authors analysed the influence overheads caused due to inter-core communication. This work does not use the optimum commutative communication approach and the parallel Montgomery implementation.

(Nadia Nedjah, *et al*,2006), proposed three varied prototypes for implementing binary modular exponentiation. The first one possessed a sequential architecture, second preferred a parallel architecture, while the last one considered a systolic architecture.

(Sahu, *et al*, 2011) presented a noble scheme for modelling RSA public key cryptosystem that supports multiple key sizes in the range of 128 bits, 256 bits or 512 bits. The author employed this system with FPGA core.

(M. Rohit, *et al*, 2013) made an effort and developed a secure algorithm to mitigate the distribution of product of two prime numbers (n). If the factors of n are hacked, it could compromise the security provided by the RSA algorithm.

(Chhabra, *et al*, 2011) advocated a scheme that establishes itself as more secure as compared to the original RSA algorithm for digital signatures and encryption in public key cryptosystems. This scheme eliminates the requirements of transferring n, the product of two random numbers, but in essence they are large prime numbers.

(Jiang Huiping, *et al,2011)* enhanced the mechanism of RSA coprocessor while taking into account the power analysis for RSA coprocessor. Initially, the shadow approach was developed with the RSA algorithm for increasing the complicatedness for discrepancy power analysis.

(Xuewen Tan, *et al*, 2012)have developed Batch RSA-S1 Multi-Power RSA algorithm and optimized the overall performance of RSA decryption by adding up the load transferring approaches and multi-prime schemes in the Batch RSA algorithm.

(IputHeri K, *et al*. 2009) proposed RSA-encryption model using robust Pipelined radix-2 Montgomery's multiplication architecture and in later stage they exploited the algorithm for accomplishing higher speed and optimum computation efficiency. This approach divides the computation of Montgomery modular multiplication into numerous clock cycles for accomplishing higher speeds.

(C. Wen, *et al*, 2003) developed a model with radix-4 modular multiplication algorithm that was functional on the basis of Montgomery's algorithm and a fast radix-4 modular exponentiation algorithm for RSA public-key cryptosystem. The authors proved that this multiplier performed four-times faster as compared to a direct radix-2 implementation of Montgomery algorithm.

(P. Fournaris, *et al,2005)* developed a systolic, scalable, superfluous carry-save modular multiplier scheme and further developed RSA encryption architecture while taking into account the Montgomery modular multiplication algorithm where the integrated system was used with FPGA core.

(Perovic, *et al*, 2012) proposed a system model for RSA implementation with FPGA cores while taking key size as 1024 bits. The authors explored the system performance with factors like resource occupancy and highest operational frequency.

Numerous efforts have been made to optimize the authentication and its optimization with RSA cryptosystems and majority were implemented with hardware platforms. But considering a competitive multi-transceiver or MIMO kind of applications, these approaches are found to be limited in terms of critical latency, power factor and hence overall performance. Most of these schemes are computationally intense since they use serial Montgomery multiplication. Further, RSA approaches suffer from reorder cryptosystem limitations. The proposed implementation of commutative cryptography using Parallel Montgomery multiplication offers high processing speed and lower power consumption. It also avoids the key exchange complications.

3. Mathematical Background

In this section, the mathematical modeling and its sequential implementation for Commutative RSA system realization is presented. The CRSA algorithmic model for key generation, commutative encryption and decryption process at each of the three transceiver terminals are detailed.

3.1 Key Generation

In order to generate the pseudo random number, the Linear Feedback Shift Register (LFSR) has been used and more precisely the Fibonacci LFSR has been taken into consideration. It facilitates a better scenario for hardware or multiple core implementations for distributed core CRSA. Each output of the pseudo random number generator is checked whether it is a prime number or not. Using the prime numbers, the variables for encryption exponent (e) and decryption exponent (d) are computed. These are the commutative encryption and decryption keys which have been further employed for creating cipher texts C and later converted into plain text M. In order to enhance the overall efficiency and reduce critical delay in key generation, we have employed two parallel pseudo random generators which are succeeded by 32 bit LFSRs. The pseudo random bits generated are stored in shift registers and once it gets filled, the LFSR stops further bit generation till the register memory is available for the next random bits. It makes the system power efficient and

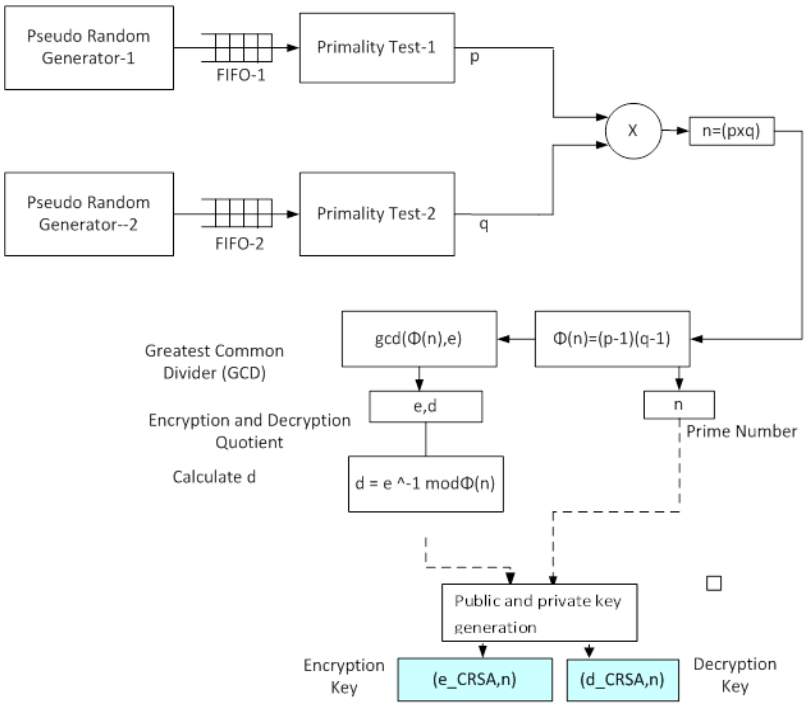

Fig. 1 Key Generation for Commutative RSA Algorithm

increases the speed. The mathematical approach and ASM chart to perform key generation has been presented in Figure 1 and Figure 2 respectively.

Figure 2 presents the functional architecture of key generation in Commutative RSA. In the designed scheme of key generation, the seed data bits of size 512 bits are fed to the key generator which ultimately generates the 1024 bits of encryption key e, decryption key d and the parameter Φ. Here the prime number of 1024 bits has been generated and is fed to the Encryption and Decryption processes. The process of Algorithm may be easily designed using Algorithmic State Machine (ASM) charts (S. Ramachandran, 2007) rather than by the traditional state diagram.

3.2 Commutative Encryption

The RSA cryptosystem is one of the optimum public key cryptography approaches. However, its overall robustness gets limited due to one way encryption and majority of existing RSA schemes suffer from reorder issues. Therefore, in order to make this system least complicated and more efficient, an approach called Commutative RSA has been proposed. In this scheme, the order in which encryption has been done would not affect the decryption if it is done in the same order. The mathematical scheme for performing this encryption is described by a pseudo algorithm presented in Figure 3. For accomplishing a robust and least critical delay encryption, we have employed a modified architecture called the Parallel Montgomery Multiplication for distributed cores. Here we have developed a high radix parallel architecture for Montgomery multiplication module using the Encryption and Decryption from our earlier work (R. Ambika *et al*, 2013). The algorithmic development and details of this modified Montgomery has been presented in Section 4.

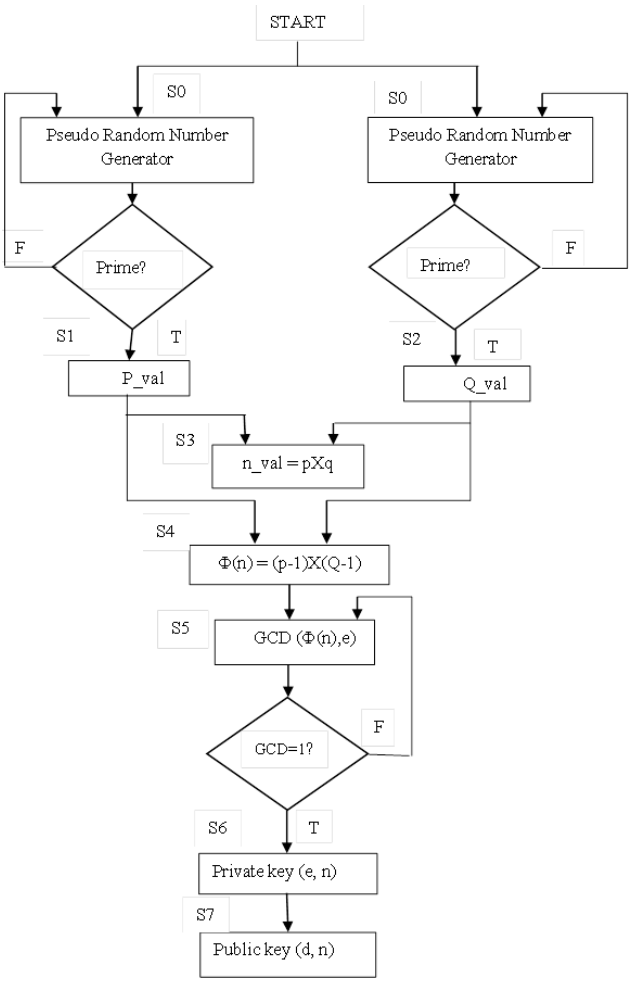

Fig. 2 ASM Chart for Commutative Key Generation

Using encryption key e, the plain text M is converted to the cipher text C. Mathematically the commutative

RSA encryption algorithm can be stated as follows:

1. Prime numbers: p ,q
2. Compute n: n = p X q
3. Plain text : $M < n$
4. Cipher text: $C = M^{e_{crsa}}(Mod\ n)$

Fig. 3 Pseudo Algorithm for Commutative Encryption

3.3 Commutative Decryption

In commutative RSA, the decryption would be done in the same fashion as was done for the Commutative encryption. In this work, we have implemented the parallel Montgomery modular exponential modules that decrypt the cipher text into a plain text. In Commutative decryption, at every individual transceiver terminal, the plain text is obtained from cipher text exponent as shown in Figure 4. The designed cryptosystem has been presented in the next section.

Algorithm for Decryption

Cipher text: C
Plain text: $M = C^{d_{crsa}}(Mod\ n)$

Fig.4 Pseudo Algorithm for Commutative

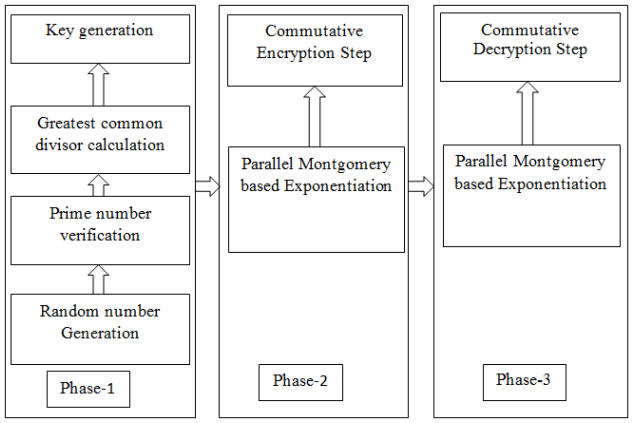

Fig. 5 Sequential Model for Commutative RSA Realization

4. Commutative RSA Cryptography Core

In this section, we have presented the overall system design of the Commutative architecture of RSA and its simulation with multiple distributed FPGA cores. The overall system realization was done in a sequential way with multiple cores developed with its individual commutative cryptosystem. The sequential phases have been presented in Figure 5.

In commutative cryptosystem architecture, the CRSA algorithm has been designed for every encompassing distributed FPGA Core. The cryptography encryption as well as decryption algorithm has been realized for each individual transceiver. The encryption and the decryption modules have been realized using the parallel implementation of Montgomery Multiplication. The design of parallel Montgomery exponential module reduces the critical delay as well as the power consumption and thus increases the overall efficiency of the Commutative RSA authentication scheme.

4.1 CRSA Oriented Montgomery Parallel Multiplier

A number of researchers have advocated the basic parallelization of the Montgomery by performing multiple self-sufficient Montgomery Multipliers for multiple cores in parallel (A. Moss, *et al*, 2007), (Marcelo E *et al*, 2005). Although they enhance the throughput of the system, unfortunately, they represent a static latency approach that is ineffective for optimizing latency of individual Montgomery multiplier modules. In real time applications, the public key cryptosystems like RSA, Elliptic Curve Cryptography (ECC) do encompass huge number of multiplier modules. It contains higher data dependencies and thus the static kind of latency approaches might not be fruitful for accelerating the single instance of public key cryptosystems like RSA, ECC or Digital Enhanced Cordless Telecommunication Standard (DECT) cipher (DSC). All these shortcomings can be eliminated while taking into account of uniform task partitioning where the single Montgomery Multiplier would be divided into varied divisions with an assumption that all sub parts are assigned uniform load. Similarly, a system needs higher tolerability for allied communication delay and thus the created inter-core communication is, in general, much slower as compared to intra-core communication situation. Therefore, for multiple core communication, the inter-core communication might be a bottleneck in parallel multipliers. The higher tolerability in delay means that the system performance of the proposed parallel architecture is not much affected by any inter-core communication. The parallel Montgomery Multipliers are eminently suited for diverse multi-core models and it makes the overall system stable even for higher throughputs.

4.2 Parallel Montgomery Multiplier module

The parallel implementation of Montgomery multiplication for multi-core applications is effective for RSA cryptosystems and its applications. In a number of cryptosystems, a series of multiplication functions operating concurrently are required. For example, the modular exponentiation is estimated while employing the chain of processes like modular multiplication and squaring. Multiplication and squaring are in general implemented by means of an integer multiplication followed by a modular reduction with certain predefined modulus. In case of RSA algorithm, the modular reduction is a pseudo random variable. Reduction in overall computational complexity is one of the significant contributions of Montgomery residue depiction and its resultant multiplication algorithm (Bunimov, *et al*, 2002), (Zhimin Chen, *et al*,*2011*).The design of Montgomery

Multiplication Algorithm was presented in detail in our earlier work (R. Ambika *et al*, 2013) and the same is used in the present work. In operational Montgomery multiplication process, initially the functional operands are transformed into their allied Montgomery residue representation. That is followed by performing integer

multiplication and squaring. Finally, the result is converted back into its generic integer representation.

4.3 Montgomery Multiplication with Multiple Transceiver

Majority of the Montgomery algorithms need 32 bits precision for the computation. For implementing the parallel Montgomery multiplication, few schemes like bipartite (A. Moss, *et al*, 2007), (Marcelo E, *et al*, 2005) and tripartite (Marcelo E, *et al*, 2008) were developed. In this work, multiple core hardware architecture is proposed for Commutative RSA encryption and decryption. The developed scheme involves highly efficient communication between incorporated multiple cores. This is a robust and efficient CRSA approach and offers real time functional hardware architecture.

4.4 Commutative nature of CRSA

Let the variables G_A and G_B represent the group members required to perform communication over the secure plane. Two prime variables $Param\_P_p^{CRSA}$ and $Param\_Q_q^{CRSA}$ are selected from a pseudo random number generator in order to compute the encryption key and decryption key. The products of two prime numbers $Prop\_N^{CRSA}$ and $Prop\_\phi^{CRSA}$ are calculated as follows:

$$Prop\_N^{CRSA} = \left[\left(Prop\_P_p^{CRSA}\right) \times \left(Prop\_Q_q^{CRSA}\right)\right] \quad (1)$$

$$Prop\_\phi^{CRSA} = \left[\left(Prop\_P_p^{CRSA} - 1\right) \times \left(Prop\_Q_q^{CRSA} - 1\right)\right] \quad (2)$$

Using these expressions, it can be found that

$$Param\_N_A{}^{CRSA} = Param\_N_B{}^{CRSA} \quad (3)$$

$$Porp\_\phi_A{}^{CRSA} = Prop\_\phi_B{}^{CRSA} \text{ for } A \text{ and } B \quad (4)$$

The key pairs called encryption key pair of A and B has been achieved by following expression.

$$\left(Prop\_N_A{}^{CRSA}, Porp\_E_A{}^{CRSA}\right) \text{ and } \left(Prop_{N_B}{}^{CRSA}, Prop_{E_B}{}^{CRSA}\right) \quad (5)$$

The $Param\_E^{CRSA}$ is retrieved using randomly selected variables in such a way that it is a co-prime of $Prop\_\phi^{CRSA}$ or in other terms, it can be expressed as

$$\mathcal{F}n_{GCD}(Prop\_E^{CRSA}, Prop\_\phi^{CRSA}) = 1 \quad (6)$$

where $\mathcal{F}n_{GCD}(x, y)$ is the greatest common divisor (GCD) function that exists between variables x and y.

Similarly, the decryption key pair of A and B is presented in terms of:

$$\left(Prop_{N_A}{}^{CRSA}, Prop_{D_A}{}^{CRSA}\right) \text{ and } \left(Prop_{N_B}{}^{CRSA}, Prop_{D_B}{}^{CRSA}\right) \quad (7)$$

and the property $Prop\_D^{CRSA}$ is calculated based on the expression.

$$Prop\_D^{CRSA} = \left(Prop_E{}^{CRSA}\right)^{-1} Mod\left(Prop_N{}^{CRSA}\right) \quad (8)$$

Consider Enc_X as the encrypted data X, the encryption operation may be expressed as:

$$Enc_X = Y^{Prop\_E^{CRSA}} Mod(Prop\_N^{CRSA}) \quad (9)$$

Likewise the resulting commutative RSA decryption functions on the retrieved encrypted data may be expressed as:

$$Dec_Y = Enc_X{}^{Prop\_D^{CRSA}} Mod(Prop\_N^{CRSA}) \quad (10)$$

5. Results and Discussions

A highly robust cryptosystem based on commutative RSA algorithm has been designed and simulated on multiple FPGA cores. The overall system model has been coded using VHDL with multiple transceivers. The design is simulated using Modelsim and synthesized using Xilinx Design Suite 14.3.

Frequency of operation in simulation has been set to 100 MHz. The various encryption and decryption data values at each user's location are computed using equations (1) to (10) and presented in Tables 1, 2 and 3. The data values shown in these three tables are remapped for the VHDL RTL codes as presented in Table 4. The original data which is required to be encrypted is shown as data_pram in Figures 6 to 8. The encryption key is shown as e_pram and the encrypted output data is shown as cypher.

The encryption starts at time 150 ns and completes processing at 10035 ns for user 1 terminal as shown in simulation waveforms presented in Figure 6. Similarly, for user 2 terminal, encryption commences at time 10035 ns and ends at 22195 ns, whereas for User 3 terminal, start and end times are 22195 ns and 34115 ns respectively as presented in Figures 7 and 8. The decryption timings for the three user terminals are presented in Figures 9, 10 and 11.

The encrypted data are input as incipher for Decryption processing. The decryption key is different from the encryption key and is shown as d_pram and the decrypted output is shown as originalplaintext in the waveforms. It may be noted that the encrypted data of user 1 will be the input data for user 2 and so on. Similarly for the decryption. For example, user 1 data 7487875 is encrypted as shown in Fig. 6 and decrypted as shown in Fig. 11 recovering back the same data. This proves the commutative nature of the algorithm and the RTL design. The encrypted and decrypted output data have also been verified to be correct from Cryptography calculator http://rsatools.wforums.net/. From the waveforms, it can be seen that the complete encryption as well as the decryption process takes 100 clock cycles or 10 µs at 100 MHz.

The Synthesis, Place and Route have been run on the RTL design. The design was synthesized using Xilinx Design Suite 14.3 targeted on Virtex-5, xc5vfx70t-2ff1136 FPGA. The design for both the encryption and the decryption utilizes about 67% of the chip resources as

Table 1: Data at User 1 Terminal

USER 1		
Data	**Decimal**	**Hexadecimal**
p_val	59083	E6CB
q_val	33223	81C7
n_pram	1962914509	74FFB2CD
e_pram	699776239	29B5BCEF
data_pram	7487875	724183
cypher	848084699	328CBEDB
d_pram	1389794659	52D69563
Original plain text after decoding	7487875	724183

Table 2: Data at User 2 Terminal

USER 2		
Data	**Decimal**	**Hexadecimal**
p_val	59083	E6CB
q_val	33223	81C7
n_pram	1962914509	74FFB2CD
e_pram	1154032391	44C92307
data_pram	848084699	328CBEDB
cypher	752490942	2CDA19BE
d_pram	1608356723	5FDD9373
Original plain text after decoding	848084699	328CBEDB

Table 3: Data at User 3 Terminal

USER 3		
Data	**Decimal**	**Hexadecimal**
p_val	59083	E6CB
q_val	33223	81C7
n_pram	1962914509	74FFB2CD
e_pram	627898457	256CF859
data_pram	752490942	2CDA19BE
cypher	553018001	20F66291
d_pram	1057410797	3F06CEED
Original plain text after decoding	752490942	2CDA19BE

Table 4 Mapping of Data used in equations and waveforms

Designation of Data in Equations	**Designation of Data in Waveforms**
$Prop\_P_p^{CRSA}$	p_val
$Prop\_Q_q^{CRSA}$	q_val
$Prop\_N^{CRSA}$	n_pram
$Prop\_E^{CRSA}$	n_pram
Y	data_pram
Enc_X	cypher
$Prop\_D^{CRSA}$	d_pram
Dec_Y	originalplaintext

Table 5 Device Utilization of Encryption & Decryption RTL Design

Device utilization	**Utilized**	**Available**	**Utilization (%)**
Number of Slice Registers	31298	44800	69
Number of Slice LUTs	30129	44800	67
Number of Bonded IOBs	2051	640	320
Number of fully used LUT-FF pairs	20258	41169	49

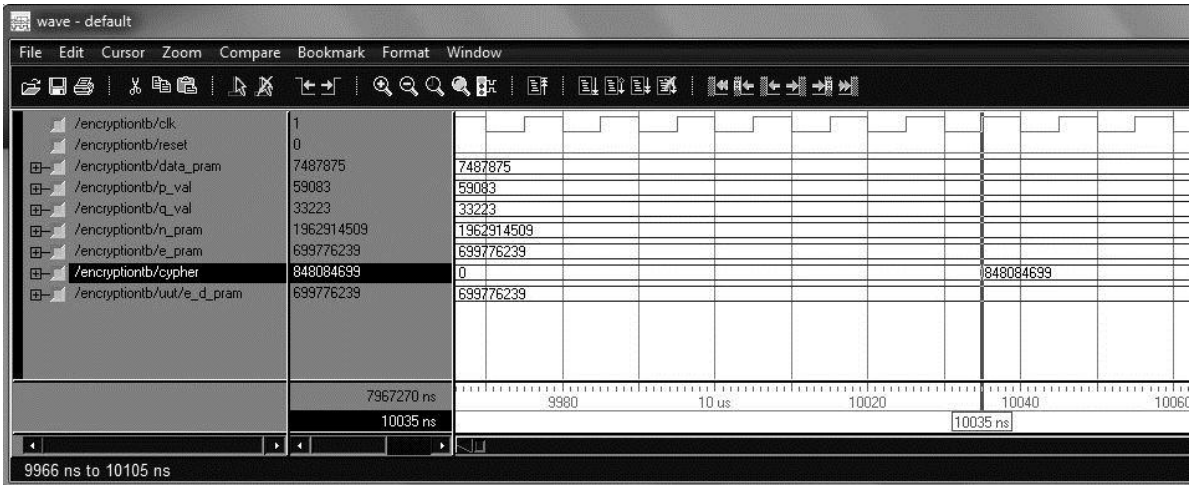

Fig. **6** Waveform of CRSA Encryption phase1 as per equations 1 to 9

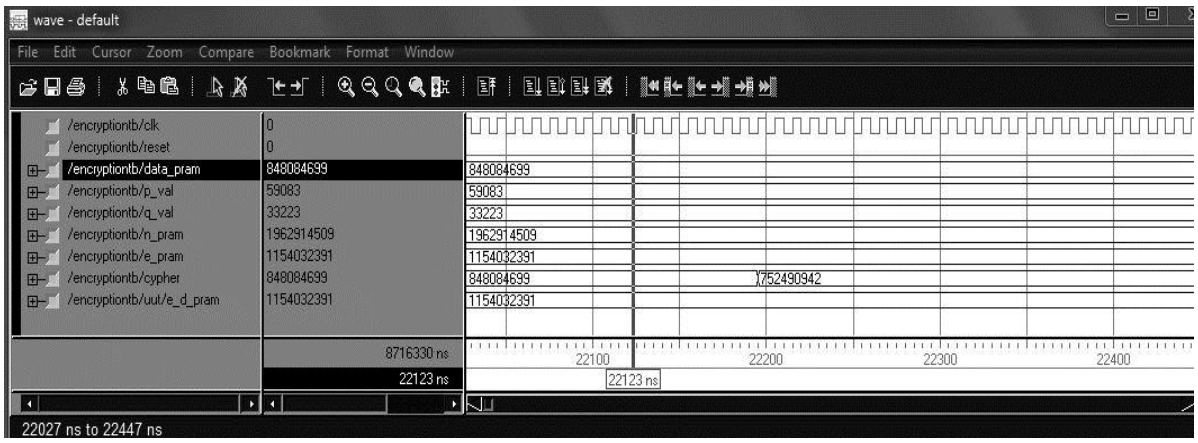

Fig. 7 Waveform of CRSA Encryption Phase 2 according to equations 1 to 9

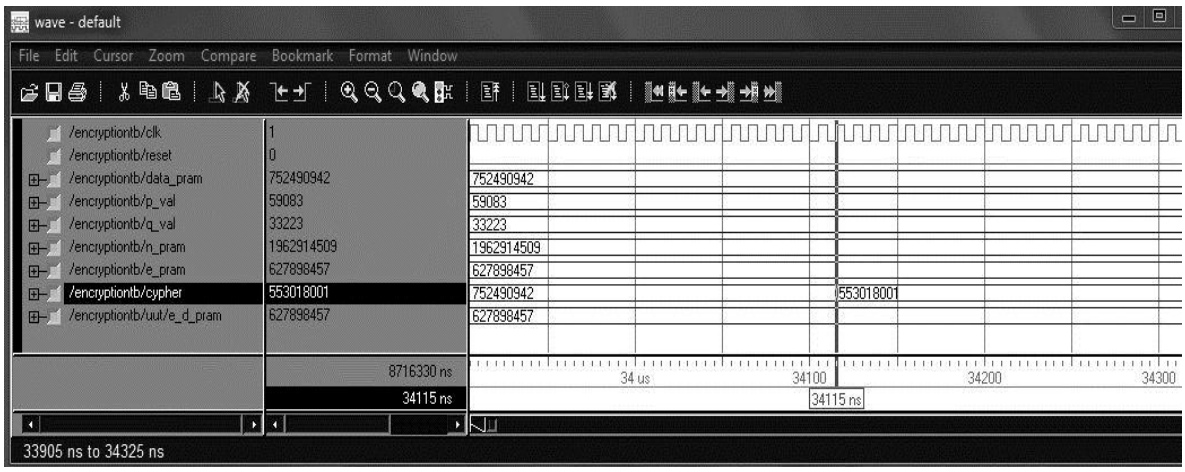

Fig. **8** Waveform of CRSA Encryption phase 3 according to equations 1 to 9

presented in Table 5. The maximum operating frequency reported by the Xilinx Design Suite 14.3 tool is 199 MHz as shown in Table 6. Therefore, the processing time for the complete encryption and the decryption together is about 5 µs. The RTL view of the Commutative RSA Key generation is shown in Figure 12. The RTL views of Encryption and Decryption are not presented since details are too numerous to be accommodated in this paper.

Table 6 Timing analysis for CRSA using ISE 14.3

RTL Design	**Maximum Frequency (MHz)**
Key Generation	316
Overall CRSA (Encryption & Decryption)	199

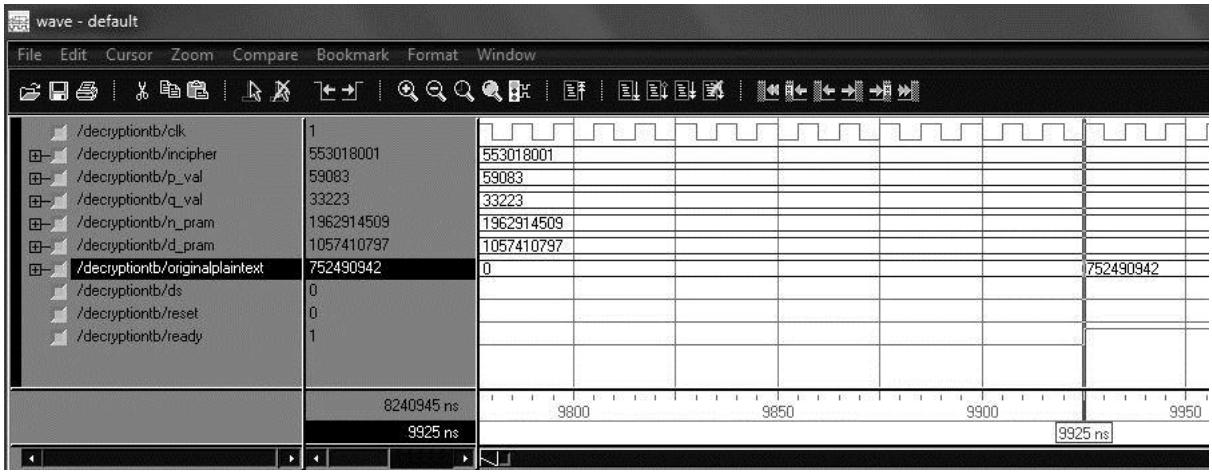

Fig. 9 Waveform of CRSA Decryption phase 1 according to equations 1 to 10 for user 3

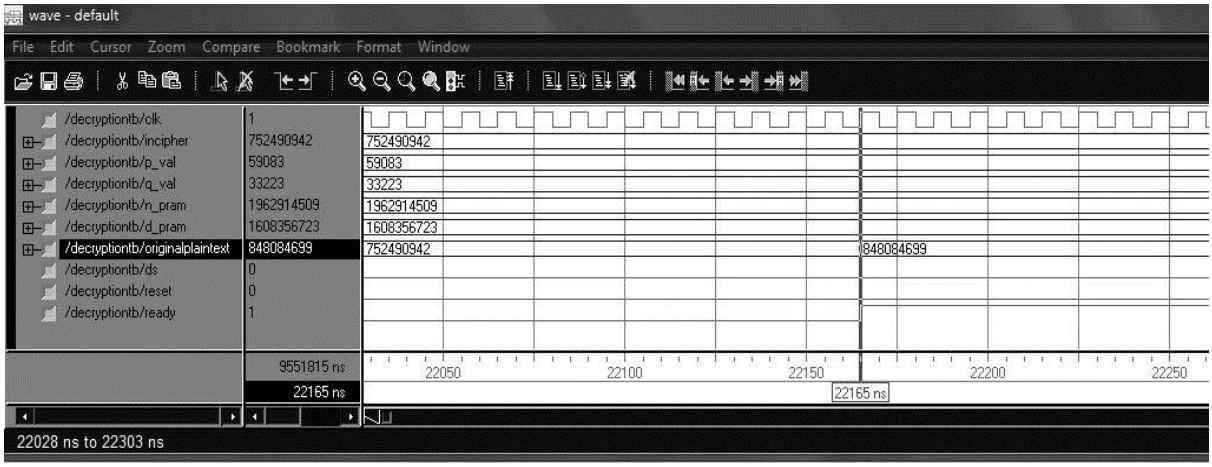

Fig. 10 Waveform of CRSA Decryption phase 2 according to equations 1 to 10 for user 2

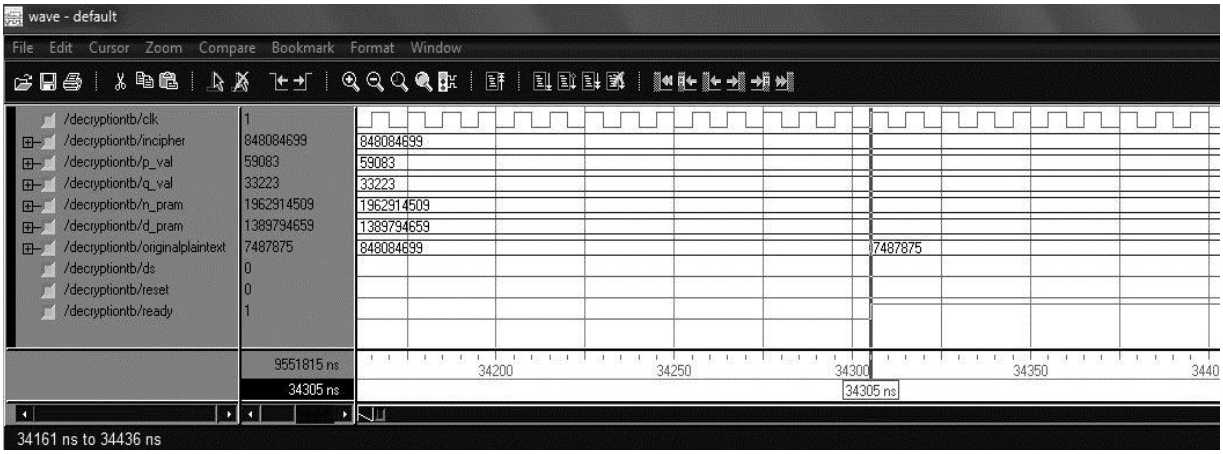

Fig. 11 Waveform of CRSA Decryption phase 3 according to equations 1 to 10 for user 1

Conclusion

This paper presented a novel encryption-decryption system design called Commutative RSA with key generation realized using VHDL. This can be used for MIMO transceiver based public infrastructure communication. A scheme of parallelized Montgomery multiplication with high radix has been used for accomplishing higher rate encryption and decryption of RSA algorithm. It exhibits high processing speed, thus suited for real time applications.

References

William Stallings,(2003) Cryptography and Network Security: Principles and Practices. 3rd Edition.

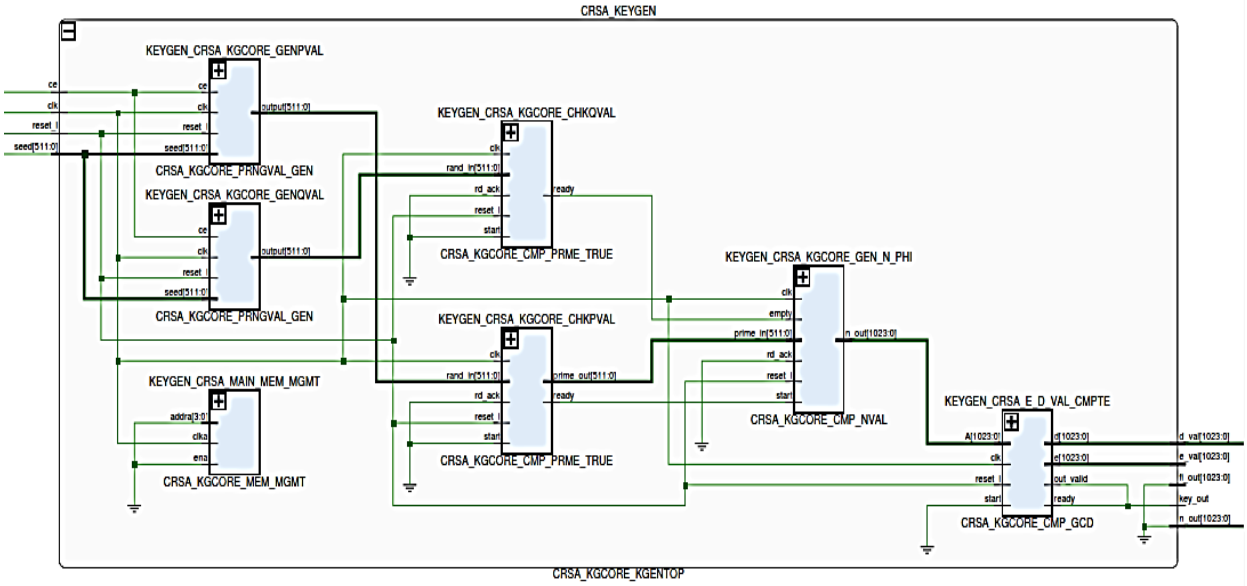

Fig. 12 RTL View of CRSA Key Generation

R. L. Rivest, A. Shamir, L. Adleman, (1978), A Method for obtaining digital signatures and public-Key cryptosystems *Comm. ACM*, Vol. 21, Issue 2; pp.120-126.

SchneierBruce,CryptographieappliquéeAlgorithmesprotocoles et codes source en C (1997). 2ème edition; *International Thomson Publishing France -Applied Cryptography-Protocols, Algorithms, and Source Code in C* - 2nd Edition..

Alan Daly, William Marnane, (2001), Efficient Architectures for implementing Montgomery Modular Multiplication and RSA Modular Exponentiation on Reconfigurable Logic, University College Cork Ireland.

R. Ambika, S. Ramachandran, K. R. Kashwan,(2013), Securing Distributed FPGA System using Commutative RSA Core; *Global Journal of Researches in Engineering Electrical and electronics Engineering*, Vol. 13, Issue 15, version 1.0, pp.47-58.

Eldridge, S.E., and Walter, C.D., (1993), 'Hardware Implementation of Montgomery's Multiplication Algorithm', *IEEE Trans. Computers.* Vol. 42, No.6, pp. 693–699.

Elbirt, A.J., and Paar, C., (1999), Towards an FPGA Architecture Optimized for Public-Key Algorithms; the SPIE Symposium on Voice, *Video and Communications.*

Blum, Paar, (1999), Montgomery Exponentiation on Re-configurable Hardware'. *Proc. 14th Symposium on Computer Arithmetic*, pp. 70–77.

Kim, Y.S.; Kang, W.S, (2000), Choi J.R.; Implementation of 1024-bit processor for RSA cryptosystem'.*http:/ /www.ap-asic. org/ 2000/proceedings/10-4.pdf*

Bunimov V, Schimmler M, Tolg B,(2002), A Complexity-Effective Version of Montgomery's Algorithm; Presented at the Workshop on *Complexity Effective Designs (WECD02)*.

Gustavo D. Sutter; Jean-Pierre Deschamps; José Luis Imaña,(2011), Modular Multiplication and Exponentiation Architectures for Fast RSA Cryptosystem Based on Digit Serial Computation; *IEEE transactions on industrial electronics*, vol. 58, No. 7.

Zhimin Chen; Patrick Schaumont(2011), A Parallel Implementation of Montgomery Multiplication on Multicore Systems: Algorithm, Analysis, and Prototype; *IEEE transactions on computers*, Vol. 60, No. 12.

Nadia Nedjah; Luiza de MacedoMourelle, (2006), Three Hardware Architectures for the Binary Modular Exponentiation: Sequential, Parallel, and Systolic; *IEEE transactions on circuits and systems—i: regular papers*, Vol. 53, No. 3.

Sushanta Kumar Sahu; Manoranjan Pradhan, (2011), FPGA Implementation of RSA Encryption System; *International Journal of Computer Applications* (0975 – 8887), Vol. 19– No.9.

Minni, Rohit; Sultania, Kaushal; Mishra, Saurabh; Vincent, Durai Raj,(2013), An algorithm to enhance security in RSA, *Fourth International Conference on Computing, Communications and Networking Technologies (ICCCNT)*, DOI: 10.1109/ICCCNT. 2013.6726517, pp.1-4,

Chhabra, A.; Mathur, S.,(2011), Modified RSA Algorithm: A Secure Approach, *International Conference on Computational Intelligence and Communication Networks (CICN)*, pp. 545-548.

Jiang Huiping; Yang Guosheng,(2011)Resistant against power analysis for a fast parallel high-radix RSA algorithm; *International Conference on Electric Information and Control Engineering (ICEICE)*; pp. 1668-1671.

Xuewen Tan; Yunfei Li,(2012) Parallel Analysis of an Improved RSA Algorithm, *International Conference on Computer Science and Electronics Engineering (ICCSEE)*, Vol.1, pp. 318-320.

IputHeri, K.; AsepBagja, N.; Purba, R.S.; Adiono, T, (2009), Very fast pipelined RSA architecture based on Montgomery's algorithm, *International Conference on Electrical Engineering and Informatics*, ICEEI '09.5-7 Aug 2009; Vol. 02, pp. 491-495.

Jin-Hua Hong; Cheng-Wen Wu , (2003) Cellular-Array Modular Multiplier for Fast RSA Public-Key Cryptosystem Based on Modified Booth's Algorithm;*IEEE transactions on very large scale integration (vlsi) systems*, vol. 11, No. 3.

A. P. Fournaris; O. Koufopavlou, (2005) A new RSA encryption architecture and hardware implementation based on optimized Montgomery multiplication; *Proc. IEEE ISCAS*, pp. 4645–4648.

Perovic, N.S.;Popovic-Bozovic, M., (2012), FPGA implementation of RSA crypto algorithm using shift and carry algorithm; *20th Telecommunications Forum (TELFOR)*, pp. 1040 – 1043.

S. Ramachandran. (2007) , Digital VLSI Systems Design, *Springer*, Netherlands.

N. Costigan and P. Schwabe,(2009), Fast Elliptic-Curve Cryptography on the Cell Broadband Engine, *Proc. Int'l Conf. Cryptology in Africa: Progress in Cryptology (AFRICACRYPT '09),* pp. 368-385,

R. Szerwinski and T. Guneysu, (2008), Exploiting the Power of GPUs for Asymmetric Cryptography, Proc. Workshop Cryptographic Hardware and Embedded System (CHES '08), pp-79-99.

A. Moss, D. Page, and N.P. Smart,(2007), Toward Acceleration of RSA Using 3D Graphics Hardware, *Proc. IMA Int'l Conf. Cryptography and Coding 2007*, pp. 213-220.

Marcelo E; Kaihara; Naofumi Takagi, (2005), Bipartite modular multiplication; *In Proceedings of cryptographic Hardware and Embedded Systems - CHES 2005*, number 3659 in Lecture notes in Computer Science, pp-.201-210. Springer-Verlag.

Marcelo E; Kaihara;Naofumi Takagi, (2008), Bipartite modular multiplication method; *IEEE Transactions on Computers;*57(2):pp.157-164, *http://rsatools.wforums.net/*