Hybrid Crypto Hardware Utilizing Symmetric-Key & Public-Key Cryptosystems

Adnan Abdul-Aziz Gutub
Center of Research Excellence in Hajj and Omrah
Umm Al-Qura University
Makkah 21955, Saudi Arabia
aagutub@uqu.edu.sa

Farhan Abdul-Aziz Khan
Department of Computer Engineering
King Fahd University of Petroleum & Minerals
Dhahran 31261, Saudi Arabia.
farhank@kfupm.edu.sa

***Abstract*— This paper proposes a hybrid crypto system that utilizes benefits of both symmetric key and public key cryptographic methods. Symmetric key algorithms (DES and AES) are used in the crypto system to perform data encryption. Public key algorithm (RSA) is used in the crypto system to provide key encryption before key exchange. Combining both the symmetric-key and public-key algorithms provides greater security and some unique features which are only possible in this hybrid system. The crypto system design is modeled using Verilog HDL. The implementation has various modules for DES, AES and RSA. The implementation also has a pseudorandom number generation unit for random generation of keys and a GCD computation unit for RSA. All the hardware modules are designed by Register Transfer Level (RTL) modeling of Verilog HDL using ModelSim SE 5.7e. showing interesting promising results.**

Keywords-Cryptography Hardware, Hybrid crypto system, Symmetric key, Public key algorithm, DES, AES, RSA

I. INTRODUCTION

Cryptography is the study of mathematical techniques related to aspects of information security such as confidentiality, data integrity, entity authentication and data origin authentication. Cryptography is not the only means of providing information security but rather one set of techniques. A crypto system is a general term referring to a set of cryptographic primitives used to provide information security services. Most often, the term is used in conjunction with primitives providing encryption and decryption [1].

The two most popular types of cryptographic algorithms are classified into either symmetric-key or public-key crypto techniques. Conventional symmetric-key cryptographic algorithms like Data Encryption Standard (DES) and Advanced Encryption Standard (AES). use a single secretly shared key to perform both encryption and decryption [1]. The advantage of symmetric-key algorithms is that they are faster in execution because of straightforward cryptographic transformations and can be pipelined to give better performance. One of the major issues with symmetric-key algorithms is to find an efficient method to agree and exchange keys securely. This problem is referred to as the key distribution problem. If the secret key is disclosed during the key exchange the whole symmetric key encryption/decryption algorithm becomes completely vulnerable [1].

On the other hand, public-key cryptography algorithms, also called asymmetric cryptographic algorithms, like Rivest-Shamir-Adleman (RSA) [6], ElGamal [7] etc. use separate public and private keys to perform encryption and decryption, respectively [1]. The public key is made widely available and is used by communicating parties to encrypt data. Only the party with the correct corresponding private key can decrypt data. Public-key algorithms are secure cryptographic algorithms because they are based on underlying mathematically hard-to-solve problems that are substantially slower than symmetric-key cryptography algorithms, which made public key algorithms commonly used in practice for the transport of keys, subsequently used for bulk data encryption and decryption by symmetric-key algorithms and other applications [1].

This paper presents a hybrid crypto system that combines the strengths of both symmetric and public-key algorithms by performing data encryption and decryption using any symmetric algorithm: DES and AES, and key encryption using public-key RSA algorithm.

The next sections of this paper is organized as follows. We first specify the related literature surveyed for the implemented cryptographic algorithms, then we briefly discuss the idea of hybrid crypto system and design of various modules in the hybrid crypto system and finally we compare some simulation and synthesis results of the different versions of the crypto system modules along with the design choices.

II. RELATED LITERATURE

The most popular symmetric-key algorithms are Data Encryption Standard (DES) and Advanced Encryption Standard (AES). The definitive source of information about the DES and AES are the NIST federal information processing standards FIPS46-3 [2] and FIPS-197 [4], respectively. The most popular public key algorithm proposed by Rivest, Shamir and Adleman is RSA [5]. The details of the RSA public key algorithm are given in [6].

DES, AES and RSA algorithms are used in the modeling of the hybrid crypto system presented in this paper. These algorithms are briefly described as follows.

978-0-7695-4959-0/13 $25.00 © 2013 IEEE
DOI 10.1109/ACSAT.2012.44

A. DES (Data Encryption Standard)

The DES algorithm is designed to encipher and decipher blocks of data consisting of 64 bits under control of a 64-bit key (only 56 bits out of 64 are used). Deciphering must be accomplished by using the same key as for enciphering, but with the schedule of addressing the key bits altered so that the deciphering process is the reverse of the enciphering process.

A block to be enciphered is subjected to an initial permutation *IP*, then to a complex key-dependent computation for 16 rounds and finally to a permutation which is the inverse of the initial permutation IP^{-1}. The key-dependent computation can be simply defined in terms of a function *f*, called the cipher function, and a function *KS*, called the key schedule.

B. AES (Advanced Encryption Standard)

The Advanced Encryption Standard (AES) is a Federal Information Processing Standard (FIPS-197 of NIST) that specifies a cryptographic algorithm for use by U.S. Government organizations to protect sensitive, unclassified information [4]. NIST selected Rijndael (pronounced as Rhine-Dahl) as the AES algorithm. The two researchers who developed and submitted Rijndael for the AES were both cryptographers from Belgium: Dr. Joan Daemen and Dr. Vincent Rijmen.

The AES algorithm is a symmetric block cipher that can process data blocks of 128 bits using cipher keys with lengths of 128, 192 and 256 bits. These three different versions of AES may be referred to as AES-128, AES-192 and AES-256. The AES algorithm consists of AES Cipher, AES Inverse Cipher and Key Expansion algorithms. The AES Cipher has the function of performing a series of mathematical operations on data blocks of 128 bits to convert a given plain text block into a cipher text block of 128 bits. The AES Inverse Cipher has the function of performing a series of mathematical operations on data blocks of 128 bits to convert a given cipher text block into a plain text block of 128 bits. For the AES algorithm, the length of the input block, the output block and the State is 128 bits. This is represented by *Nb = 4*, which reflects the number of 32-bit words (number of columns) in the State.

For the AES algorithm, the length of the Cipher Key, *K*, is 128, 192, or 256 bits. The key length is represented by *Nk = 4, 6, or 8*, which reflects the number of 32-bit words (number of columns) in the Cipher Key. For the AES algorithm, the number of rounds to be performed during the execution of the algorithm is dependent on the key size. The number of rounds is represented by *Nr*, where *Nr = 10* when *Nk = 4, Nr = 12* when *Nk = 6*, and *Nr = 14* when *Nk = 8*. The only Key-Block-Round combinations that conform to this standard are given in Fig. 1.

For both its Cipher and Inverse Cipher, the AES algorithm uses a round function that is composed of four different byte-oriented transformations:

1) byte substitution using a substitution table (S-box),
2) shifting rows of the State array by different offsets,
3) mixing the data within each column of the State array,
4) adding a Round Key to the State.

AES cipher, AES inverse cipher and AES key expansion algorithms are detailed in [4].

	Key Length *(Nk words)*	Block Size *(Nb words)*	Number of Rounds *(Nr)*
AES-128	4	4	10
AES-192	6	4	12
AES-256	8	4	14

Figure 1. AES Key–Block–Round Combinations

C. RSA Cipher

RSA cryptographic algorithm was invented in 1977 by Rivest, Shamir and Adleman [5]. It is based on the integer factorization problem. The RSA encryption and decryption primitives as given in PKCS-1 [6] are shown in Fig. 2 & Fig. 3, respectively.

RSAEP $((n, e), m)$

Input. (n, e) RSA public key

m message representative, an integer between 0 and $n - 1$

Output: c ciphertext representative, an integer between 0 and $n - 1$

Error: "message representative out of range"

Assumption: RSA public key (n, e) is valid

Steps:

1. If the message representative m is not between 0 and $n - 1$, output "message representative out of range" and stop.
2. Let $c = m^e \bmod n$.
3. Output c.

Figure 2. RSA Encryption Primitive

III. HYBRID CRYPTO SYSTEM DESIGN

The traditional crypto system design uses the same key called symmetric key for both encryption and decryption and performs key exchange using methods like Diffie-Hellman key exchange etc. [1].

One major disadvantage of the traditional crypto system is that if the symmetric encryption/decryption key is revealed during key exchange or by any other means, the whole encryption/decryption process becomes unsafe [1]. Another disadvantage is that if at any stage there is a need for changing the symmetric key, the key transfer process needs to be repeated.

In order to address the above-mentioned security problems with the use of symmetric-key algorithms, public-key algorithms are combined with symmetric-key algorithms to perform key-exchange [1]. Our proposed

method is one such method to combine the strengths of public-key algorithms with symmetric key cryptographic algorithms.

RSADP (K, c)

Input: K RSA private key, where K has one of the following forms:
— a pair (n, d)
— a quintuple $(p, q, dP, dQ, qInv)$ and a possibly empty sequence of triplets (r_i, d_i, t_i), $i = 3, \ldots, u$

c ciphertext representative, an integer between 0 and $n - 1$

Output: m message representative, an integer between 0 and $n - 1$

Error: "ciphertext representative out of range"

Assumption: RSA private key K is valid

Steps:

1. If the ciphertext representative c is not between 0 and $n - 1$, output "ciphertext representative out of range" and stop.
2. The message representative m is computed as follows.
   a. If the first form (n, d) of K is used, let $m = c^d \bmod n$.
   b. If the second form $(p, q, dP, dQ, qInv)$ and (r_i, d_i, t_i) of K is used, proceed as follows:
      i. Let $m_1 = c^{dP} \bmod p$ and $m_2 = c^{dQ} \bmod q$.
      ii. If $u > 2$, let $m_i = c^{d_i} \bmod r_i$, $i = 3, \ldots, u$.
      iii. Let $h = (m_1 - m_2) \cdot qInv \bmod p$.
      iv. Let $m = m_2 + q \cdot h$.
      v. If $u > 2$, let $R = r_1$ and for $i = 3$ to u do
         1. Let $R = R \cdot r_{i-1}$.
         2. Let $h = (m_i - m) \cdot t_i \pmod{r_i}$.
         3. Let $m = m + R \cdot h$.
3. Output m.

Figure 3. RSA Decryption Primitive

The design of the proposed hybrid crypto system is based on performing encryption and decryption using symmetric key algorithm but uses public key algorithm to encrypt the symmetric key before performing key transfer. Our hybrid crypto system assumes that the public key of the destination is available beforehand. In fact, there is no explicit key transfer for exchanging the symmetric key because the corresponding symmetric key for the data block is sent to the destination as part of the encrypted data block. In our proposed hybrid crypto system, a data block is encrypted using any of the standard symmetric-key cryptographic algorithms (like DES or AES). The symmetric encryption key used to encrypt the block is then encrypted using a public key crypto algorithm (like RSA) using that destination's public key. The encrypted symmetric key is then concatenated with the encrypted data block and the whole data block is sent to the destination system. The encrypted data block sent to the destination contains the encrypted data and the corresponding publicly encrypted symmetric key to decrypt that data. The block sent to the destination has the format shown in Fig. 4.

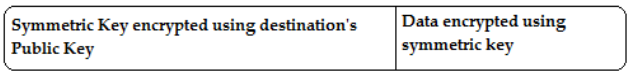

Figure 4. Format of Encrypted Block sent to destination

The destination uses its private key to decrypt the symmetric key first and then uses that symmetric key as decryption key to decrypt the received data block.
There are two advantages to using this hybrid approach:

1) The encrypted data block and the corresponding symmetric key are not transferred separately but are part of the same data block sent to the destination. This enables the sender to use a number of encryption keys without worrying about their transfer. Maximum security will be achieved by encrypting every data block with a separate encryption/decryption key.

2) Key exchange does not need to be performed explicitly initially or when changing the decryption key because every block contains a publicly encrypted symmetric key which is decrypted using the private key of the destination and readily used as decryption key to decrypt the corresponding data block.

The encryption steps performed at the source in the proposed hybrid crypto system are as follows:

Encryption Steps using Hybrid Crypto System at the Source
Prerequisite:
Source has the destination's Public Key (PUK)
Inputs:
Plain Data Block (PDB)
Symmetric Key (SK)
Destination's Public Key (PUK)
Outputs:
Encrypted Data Block (EDB)
Note: EDB contains both the encrypted PDB (denoted by ED) concatenated with encrypted SK (denoted by ESK)
Encryption Steps:
1) Encrypt PDB using SK to get ED.
(Note: This can be done using any symmetric-key crypto algorithm like DES and AES)
2) Encrypt SK using destination's PUK to get ESK.
(Note: This can be done using any public key algorithm like RSA)
3) Concatenate ED with its corresponding ESK to get EDB which is sent to the destination.

$$EDB = \{ ESK , ED \}$$

The decryption steps performed at the destination in the proposed hybrid crypto system are as follows:

Decryption Steps using Hybrid Crypto System at the Destination
Prerequisite:
Destination has its Private Key (PRK)

Inputs:
Encrypted Data Block (EDB)
Note: EDB contains both the encrypted PDB (denoted by ED) concatenated with encrypted SK (denoted by ESK)
Outputs:
Plain Data Block (PDB)
Decryption Steps:
1) Decrypt ESK using PRK to retrieve SK.
(Note: This should be done using the same public key algorithm which is used at source)
2) Use the retrieved SK as decryption key to decrypt ED to get PDB.
(Note: This should be done using the same symmetric-key crypto algorithm which is used at the source)

In order to highlight the major design components, the proposed hybrid crypto system design is depicted in Fig. 5.

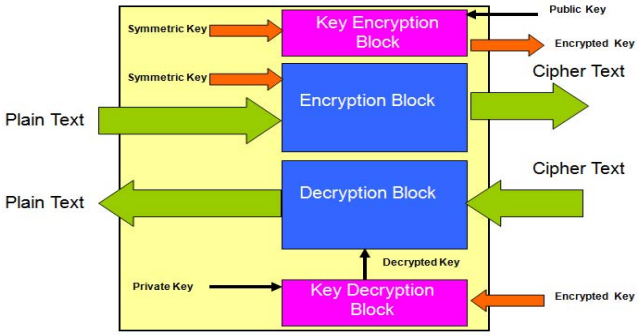

Figure 5. Block Diagram of hybrid crypto system

Although there is no explicit key transfer in the hybrid crypto system as discussed earlier, our proposed key exchange works in the manner shown in Fig. 6.

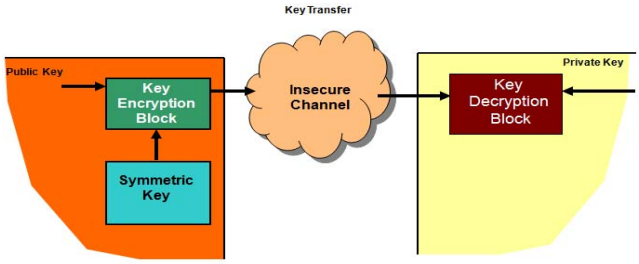

Figure 6. Key transfer using hybrid crypto system

IV. HYBRID CRYPTO SYSTEM MODELING

The crypto system is modeled using Verilog HDL. All the modules are modeled using RTL modeling. The functionality of various modules is verified by test benches. For the hybrid crypto system modeling, our chosen algorithms are DES and AES for symmetric key, and RSA for public-key encryption/decryption. It should be noted that out of the two symmetric-key algorithms, only one will be used at a time depending on the selection of the user or application. The modules in the crypto system working at the source side for encryption are shown in Fig. 7. Similarly, the modules in the crypto system working at the destination side for decryption are shown in Fig. 8.

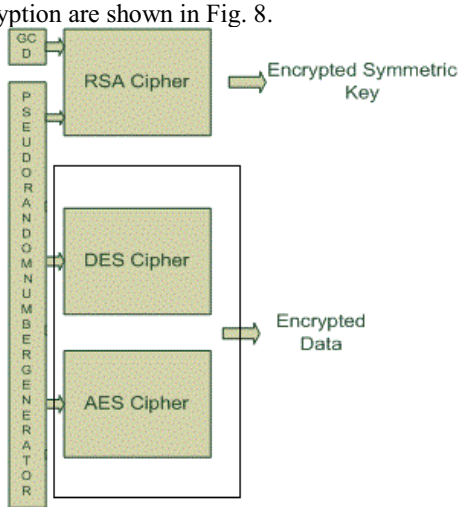

Figure 7. Modules of the hybrid crypto system at Source

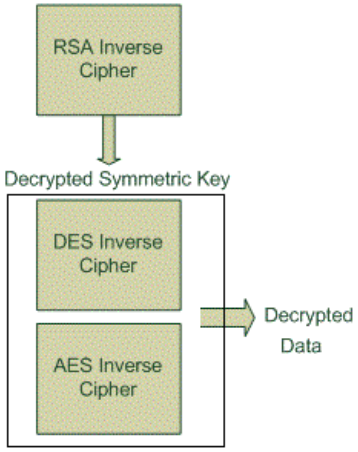

Figure 8. Modules of the hybrid crypto system at Destination

V. HYBRID CRYPTO SYSTEM IMPLEMENTATION

The proposed hybrid crypto system is implanted considering both area-optimized and performance-optimized versions. The implementation used Model Sim 5.7e which are synthesized to its gate-level using Synopsys Design Compiler. The simulation results are compared in the following section.

The area-optimized version of DES takes 16 clock cycles to perform a single encryption or decryption of 64-bit data block. The performance-optimized version of DES is heavily pipelined to provide a 64-bit data block encryption or decryption per clock cycle. The AES cipher module takes 12 clock cycles to perform a single encryption of 128-bit data block. The AES inverse cipher module takes 24 clock cycles to perform a single encryption of 128-bit data block.

The implementation of RSA module uses the right-to-left binary algorithm for modular exponentiation. The crypto system also includes a pseudo-random number generator and GCD computation unit. The pseudo-random number generator is used to generate 32-bit random numbers for possible use as keys. The GCD unit is used in RSA for GCD computation. All the modules use RTL modeling of Verilog HDL and the synthesis results are presented as below:

Synthesis Results Generated By Synopsys Design Compiler

PERFORMANCE-OPTIMIZED DES

```
****************************************
Report : area
Design : desperf
Version: 2000.05-1
****************************************
Number of ports:            186
Number of nets:            4202
Number of cells:           1673
Number of references:        19
Combinational area:      24145.000000
Noncombinational area:   13888.000000
Total cell area:         38033.000000
```

AREA-OPTIMIZED DES

```
************************************
Report : area
Design : des
Version: 2000.05-1
************************************
Number of ports:            190
Number of nets:             464
Number of cells:            196
Number of references:         7
Combinational area:       2684.000000
Noncombinational area:     448.000000
Total cell area:          3132.000000
```

AES CIPHER

```
****************************************
Report : area
Design : aescipher
Version: 2000.05-1
************************************
Number of ports:            388
Number of nets:            2355
Number of cells:           1465
Number of references:        32
Combinational area:      23746.000000
Noncombinational area:    4540.000000
Total cell area:         28286.000000
```

AES INVERSE CIPHER

```
****************************************
Report : area
Design : aesinvcipher
Version: 2000.05-1
****************************************
Number of ports:            389
Number of nets:            9447
Number of cells:           5572
Number of references:        47
Combinational area:      32550.000000
Noncombinational area:   19632.000000
Total cell area:         52182.000000
```

RSA CIPHER

```
****************************************
Report : area
Design : modmult
Version: 2000.05-1
****************************************
Number of ports:            132
Number of nets:            1168
Number of cells:            660
Number of references:        18
Combinational area:       1997.000000
Noncombinational area:    1694.000000
Total cell area:          3691.000000
```

VI. HARDWARE IMPLEMENTATION COMPARISONS

Different implementations of our hybrid cryptosystem have been compared. The study considered performance and area of all options of the implementation, i.e. area-optimized and performance-optimized versions of DES, AES cipher, Inverse Cipher, and RSA Modular Multiplier.

The performance of all these algorithms is measured as the number of clock cycles required to perform a single encryption or decryption. The cell area results are obtained by synthesizing modules using LSI_10k library of Synopsys Design Compiler. The detailed synthesis results are presented in Table 1.

Version	Performance (No. of Clock Cycles)	Area (No. of Cells)
Area-Optimized DES	16	3132
Performance-Optimized DES	1	38033
AES Cipher	12	28286
AES Inverse Cipher	24	52182
RSA Modular Multiplier	Variable	3691

Table 1. Performance & Area Results of Implemented Versions of the hybrid crypto algorithms

It is observed from the Table 1 comparison that although performance-optimized DES version operates 16 times faster than area-optimized version; it also takes approximately 12 times more area on chip than area-optimized version. For this work, only a simple implementation of AES cipher and inverse cipher was implemented. Performance-optimized implementation of AES Cipher exist [22] and can be

integrated into the design of the hybrid crypto system. Similarly, RSA Cipher is implemented in its simplest form using right-to-left binary algorithm for modular exponentiation. Several, performance-optimized implementations [8-21] exist and any of them can be seamlessly incorporated into this proposed hybrid crypto system.

It is recommended that crypto system designers need to take special consideration to the different area and delay trade-offs before deciding about the final design and the choice of algorithms. The study concentrated on the form of hybrid crypto system built using combination of AES as symmetric-key and RSA as public-key algorithm that simplest and giving fair security. A combination of AES with Elliptic Curve cryptography will be investigated in some future work.

VII. Conclusion

In this research paper, a design of hybrid crypto system is presented that utilizes both symmetric-key and public-key cryptographic algorithms. The symmetric key algorithms DES and AES are used for data encryption/decryption. The public key algorithm RSA is used for encrypting the secret key before performing key distribution. DES cipher, DES inverse cipher, AES cipher and AES inverse cipher modules are part of the system. Moreover the crypto system also contains pseudo-random number generator for random key generation and a GCD computation unit for use in RSA cipher.

The hybrid crypto system is implemented using Verilog HDL optimized with both options Area-optimized and performance-optimized. All optimization versions for DES and AES algorithms are designed. The crypto system is modeled using RTL modeling of Verilog HDL and synthesized to gate level using Synopsys Design Compiler. Comparison of DES and AES versions are also presented giving the research clear need for design choice. The proposed hybrid crypto systems provides a novel framework for implementing highly secure crypto systems by combining strengths of presently available symmetric-key and public-key crypto algorithms.

Acknowledgements

The authors wish to acknowledge the assistance and support provided by King Fahd University of Petroleum & Minerals (KFUPM). Thanks to Center of Research Excellence in Hajj and Omrah, Umm Al-Qura University (UQU), Makkah, Saudi Arabia, for collaborative moral support toward the achievements in this work.

References

[1] Alfred J. Menezes, Paul C. van Oorschot, Scott A. Vanstone, “Handbook of Applied Cryptography”, CRC Press 1997.

[2] Federal Information Processing Standards Publication 46-3, “DATA ENCRYPTION STANDARD (DES)", NIST, October 25, 1999.

[3] Special Publication 800-67, “RECOMMENDATION FOR THE TRIPLE DATA ENCRYPTION ALGORITHM (TDEA) BLOCK CIPHER”, NIST, May 2004.

[4] Federal Information Processing Standards Publication 197, "ADVANCED ENCRYPTION STANDARD (AES)", NIST, November 26, 2001.

[5] R. L. Rivest, A. Shamir and L. Adleman, “A method for obtaining digital signatures and public-key cryptosystems”, Communications of the ACM, Volume 21, No. 2, Pages 120-126, February 1978.

[6] PKCS #1 v2.1 “RSA CRYPTOGRAPHY STANDARD”, RSA Laboratories, June 14, 2002.

[7] T. Elgamal, “A public key cryptosystem and a signature scheme based on discrete logarithms”, IEEE Transactions on Information Theory, Volume 31 No. 4, Pages 469-472, July 1985.

[8] P. L. Montgomery, “Modular Multiplication without Trial Division”, Mathematics of Computation, Volume 44, No. 170, 1985, Pages 519-521.

[9] Chiou-Yng Lee; Jenn-Shyong Horng; I-Chang Jou; Erl-Huei Lu; “Low-complexity bit-parallel systolic Montgomery multipliers for special classes of GF(2m),” IEEE Transactions on Computers, Volume 54, Issue 9, Sept. 2005 Page(s):1061 - 1070

[10] S. E. Eldridge and C. D. Walter, “Hardware implementation of Montgomery's modular multiplication algorithm”, IEEE Trans. Computer, Volume 42, Pages 693-699, June 1993.

[11] D.Page, N.P.Smart, “Parallel cryptographic arithmetic using a redundant Montg omery representation” IEEE Transactions on Computers, Volume 53, Issue 11, Nov. 2004 Page(s):1474 – 1482.

[12] A. F. Tenca, C. K. Koc; “A scalable architecture for modular multiplication based on Montgomery's algorithm,” IEEE Transactions on Computers,Volume 52, Issue 9, Sept. 2003 Page(s):1215 – 1221.

[13] T. Blum, C. Paar, “High-radix Montgomery modular exponentiation on reconfigurable hardware,” EEE Transactions on Computers, I Volume 50, Issue 7, July 2001 Page(s):759 – 764.

[14] J.-C. Bajard, L.-S. Didier, P. Kornerup, “An RNS Montgomery modular multiplication algorithm,” IEEE Transactions on Computers, Volume 47, Issue 7, July 1998 Page(s):766 – 776.

[15] N. Takagi and S. Yajima, “Modular multiplication hardware algorithms with a redundant representation and their application to RSA cryptosystem”, IEEE Transactions on Computers, Volume 41 Issue 7, Pages 887 -891, July 1992.

[16] Nibouche, A. Bouridane and M. Nibouche, “Architectures for Montgomery's multiplication”, Computers and Digital Techniques, IEE Proceedings, Pages 361-368, November 2003.

[17] C.-C. Yang, T.-S. Chang and C.-W. Jen, “A new RSA cryptosystem hardware design based on montgomery' s algorithm”, IEEE Transactions, Circuits Systems II, Volume 45, Pages 908-913, July 1998.

[18] C. D. Walter, “Still faster modular multiplication”, Electronic Letter, Volume 31, Pa ges 263-264, February 1995.

[19] H. Orup and P. Kornerup, “A high-radix hardware algorithm for calculating the exponential Me modulo N”, Proceedings, IEEE 10th symposiums Computer Arithmetic, Pages 51-56, June 1991.

[20] C. K. Koc, T. Acar and B. S. Kaliski Jr, “Analyzing and comparing Montgomery multiplication algorithms”, IEEE Micro Chip, Systems, Software and Applications, Pages 26-33, June 1996.

[21] Koc, C.K., “Montgomery reduction with even modulus,” Computers and Digital Techniques, IEE Proceedings, Volume 141, Issue 5, Sept. 1994 Page(s):314 – 316.

[22] A. Bouhraoua, “Design Feasibility Study For a 500 Gbits/s AES Cypher/Decypher Engine”, Proceedings of the International Conference on Microelectronics (ICM’06), 16-19 December 2006, KFUPM, Dhahran, Saudi Arabia.