International Journal of Computer Applications (0975 – 8887)
Volume 71– No.19, June 2013

FPGA Implementation of Multistage Knapsack Public Key Cryptosystem

Oday Abdul Lateef Abdul Ridha(PhD)
Communications Eng. Dept.
University of Baghdad
Baghdad - Iraq

Bassim Sayed Mohammed (PhD)
Electrical Eng.Dept.
University of Technology
Baghdad - Iraq

Thamir Rashed Saeed (PhD)
Electrical Eng.Dept.
University of Technology
Baghdad - Iraq

Jafar Wadi Abdul Sadah (PhD)
Communications Eng. Dept.
University of Baghdad
Baghdad - Iraq

ABSTRACT
This paper presents hardware implementation of multistage trapdoor knapsack public key cryptosystem which is primarily based on Merkle-Hellman scheme. So far, multistage knapsack is not broken and there has not been known a specific attack against this system. Modular multiplier is the critical and fundamental part of the hardware implementation. In this paper, Montgomery's multiplication algorithm is modified with great hardware reduction. An efficient and optimized architecturefor modular multiplication is proposed. Three stages knapsack public cryptosystem are implemented on DE2-115 FPGA development kit.The either implemented stages (encryption or decryption) take <1% of available FPGA resources. The required clock cycles for encryption process of a one stage is equal to the encryption key length plus one, whereas decryption process of a one stage requires twice decryption key length. The stages can be easy operated in pipeline to speedup cryptosystem operations.

General Terms
Security, Public key cryptosystem, FPGA

Keywords
Public key cryptosystem, hardware implementation, information security, FPGA, knapsack

1. INTRODUCTION
Cryptograph is a practical way by which secure private communication can be conducted while using untrusted media. In 1976, Diffie and Hellman [1] introduced the idea of public key cryptosystem (PKC), in which two different keys are used: one for encryption and the other for decryption. Each user keeps his decryption key secret while making the encryption key public, it can be used by everyone wishing to send message to him. Two years later, one of the well-known additive trapdoor knapsack public key cryptosystem was discovered by Merkle and Hellman [2]. Later, it was shown to be insecure [3] and attacks have alsobeen developed on stronger variants of Merkle Hellman scheme, such as Graham-Shamir [4], iterated knapsack, multiplicative knapsack [5], Chor-Rivest cryptosystem [6], Goodman-McAuley cryptosystem and Naccache-Stern[7], knapsack based probabilistic encryption scheme [8]. As it is seen, most of knapsack cryptosystems were insecure and broken. Despite this, Hussian A.H. et al. [9] proposed and published multistage trapdoor knapsack public cryptosystem. At this moment, no attack capable of breaking this system in reasonable amount of time is known. There has not been known a specific successful attack on this scheme [7]. This system is formed by cascading multistage of the well-known scheme of trapdoor knapsack cryptosystem. It offers much greater security than single stage trapdoor knapsack cryptosystem of the same length [9]. In this paper, multistage knapsack public key cryptosystem is implemented using DE2-115 FPGAdevelopment kit. The next sections of the paper are devoted to overview of knapsack and multistage knapsack algorithms, modified Montgomery modular multiplication, hardware design of multistage knapsack cryptosystem, hardware implementation of three stages knapsack cryptosystem and finally conclusions.

2. OVERVIEW OFMULTISTAGE KNAPSACK AND KNAPSACK ALGORITHMS
In this section, the multistage Knapsack and algorithms of additive trapdoor Knapsack cryptosystem are introduced.

2.1 Multistage knapsack
Figure (1) illustrates the concept of multistage trapdoor knapsack public key cryptosystem consisting of k–stage. There is a hard knapsack vector (encryption key) of length N_i for each stage and the output (cipher text) of each stage is treated as a plain text to the next stage. Again the cipher text is decrypted using k decryption stages. These stages are arranged in reversed order as shown in Fig. (1).

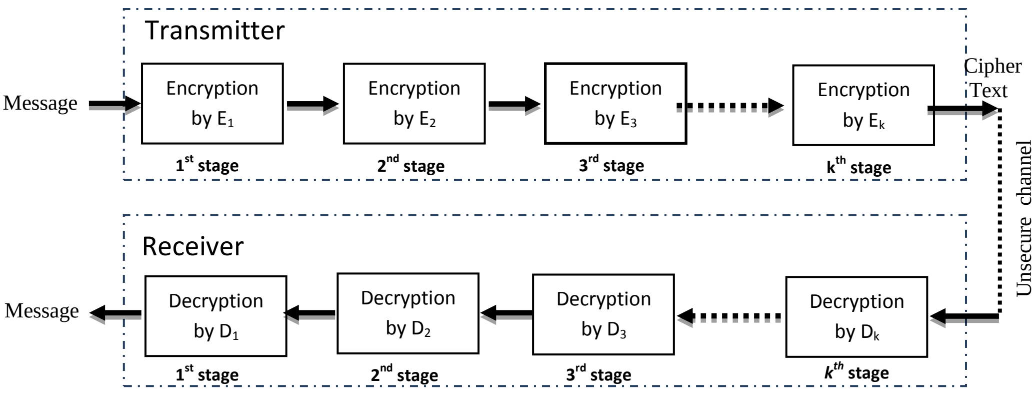


Fig1: Block diagram of k- multistage public key cryptosystem

The security of multistage knapsack as compared to single stage knapsack of length N under the condition that the length of single stage is equal to the overall length of the k-stage:

$N = N1 + N_2 + N_3 + ... + N_k$(1)

is given by [9] :

$Ø = N_1.N_2.N_3...N_k/N$ (2)

W*here Ø represents how many times the security is increased and

If $N1 = N_2 = N_3 = ... = N_o$

Then

$Ø = N_o^{k-1}/k$ (3)

Where N_o is the length of one stage ($N_o = N/k$).

As an example, using 10 multistage of length No = 30 instead of using single stage of length N = 300, the security is increased by $19683×10^8$. This means that if one hour is required to attack and break the single stage (N = 300), the 10 multistage (No = 30) requires $19683×10^8$ hours (224.6 million years).

2.2 Keys generation of the j^{th} stage

The keys generation of Merkle – Hellman trapdoor Knapsack public key cryptosystem for the j^{th} stage involves the following steps:

1- Generate anN_j-integer vector Aj' = $\{\alpha_0^j, \alpha_1^j, \alpha_2^j, ..., \alpha_{N_j-1}^j\}$ theeasily solved problem with property that each element is greater than the sum of preceding elements.

2-Choose two large numbers mj and w_j such that $m_j > \sum_{i=0}^{N_j-1} \alpha_i^j$ and GCD (w_j,m_j)=1. The secret (decryption) key of the j^{th} stage $D_j$is: Aj', m_j and wj.

3-Compute$w_j^{-1}$from the two secret integers such that $w_j^{-1}.\ w_j\, mod(m_j) = 1$

4- Compute the public key $A_j = \{a_0^j, a_1^j, a_2^j, ..., a_{N_j-1}^j\}$, the hard knapsack via

$a_j = \alpha_j . w_j\, mod(m_j)$ $i = 0,1,2,...,N_j-1$ (4)

The public (encryption) key of the j^{th} stage $E_j$is : Aj.

2.3Encryption

The plain text (message) is converted into string of binary numbers which is segmented into N_j-bits and each block $X_j = \{ x_0^j, x_1^j, x_2^j, ..., x_{N_j-1}^j \}$ is enciphered in the j^{th}stage by performing dot product with the knapsack public key A_j that is

$$y^j = \sum_{i=0}^{N_j-1} x_i^j a_i^j \qquad (5)$$

2.4Decryption

Cipher text yj is decrypted in the j^{th} stage by computing γ^j from

$$\gamma^j = y^j . w_j^{-1} mod\,(m_j) \quad (6)$$

The message is recovered by comparing γ^j with $\alpha_{N_j-1}^j$, if $\gamma^j > \alpha_{N_j-1}^j$ then $x_{N_j-1}^j$is equal to 1 , otherwise $x_{N_j-1}^j$ is equal to zero. If $x_{N_j-1}^j = 1$ then $\alpha_{N_j-1}^j$ is subtracted from γ^j and a new value is found, then comparing this value with $\alpha_{N_j-2}^j$, if this new value of γ^j is greater than $\alpha_{N_j-2}^j$, then $x_{N_j-2}^j$ is equal to 1 otherwise $x_{N_j-2}^j$ is equal to zero. This process is repeated until x_0^j is computed.

3. MODIFIED MONTGOMERY MODULARMULTIPLICATION

The most widely used algorithm for efficient modular multiplication is Montgomery's algorithm [10]. The binary Montgomery modular multiplication algorithm employs only simple addition and shift operation to avoid trial division, a critical and time consuming operation in conventional modular multiplication [11]. Montgomery multiplication, in fact, computes $x.y.w^{-1}mod(m)$ instead of $x.ymod(m)$. Some conditions are necessary for the application of Montgomery's algorithm: the modulus m needs to be relatively prime to the factor $w$and the multiplicand x and the multiplier y need to be smaller than m.In knapsack public key cryptosystem, decryption of cipher text in the j^{th} stage involves computation of $y^j w_j^{-1} mod(m_j)$. In this paper, Montgomery's algorithm is modified with great computational reduction to compute γ^j if $w_j$in the key design is chosen to be $w_j = 2^k$. Now, the only condition for application of the modified algorithm is that m must be odd number. Modified Montgomery's algorithm becomes:

INPUT: y^j, m_j,k
OUTPUT: γ^j=$y^j w_j^{-1} mod\,(m_j)$

```
$\gamma^j \leftarrow y^j$
for $i \in 0..k-1$
    $\gamma^j \leftarrow \frac{\gamma^j}{2}$ if $(\gamma_0)^j = 0$
    $\gamma^j \leftarrow \frac{(\gamma^j + m_j)}{2}$ otherwise
return$(\gamma^j)$
```

4. HARDWARE DESIGN OF MULTISTAGE KNAPSACK CRYPTOSYSTEM

The hardware design of multistage knapsack cryptosystem consists of two parts, as early mentioned, encryption and decryption parts. Each part consists of ***k*** semi similar stages. In the following sections, the hardware design and implementation of a one stage of each parts of the cryptosystem are presented.

4.1 Hardware design of the j^th^ stage of Encryption part

Each encryption stage in the multistage knapsack cryptosystem has five types of signals, as shown in the Fig.(2).

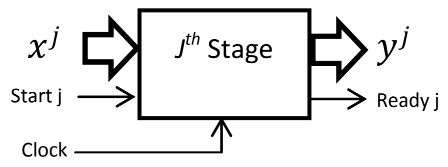


Fig2: Encryption stage

These signals are data input (x^j), data output (y^j), clock, command (*start j*), and status (*ready j*). x^j and y^j correspond to plain data and encrypted data, respectively. *Start j* signal initiates the logic circuit of stage j to start encryption process. At the end of encryption process, *ready j* signal is activated. In multistage cryptosystem, *start* and *ready* signal are used for synchronization between stages. *Ready* signal of the *j*th stage is connected to the *start* signal of the next stage. Therefore, construction of multistage cryptosystem is very easy. The function of any stage in the encryption part is to compute y^j according to Eq.(7)

$$y^j = \sum_{i=0}^{N_j-1} x_i^j a_i^j \quad (7)$$

Where a_i^j is the i_{th} element of a pre computed encryption key vector $\mathbf{A}_j$ of the *j*th stage ,

x_i^j is the i_{th} bit of input data (plain text) of the *j*th stage,

y^j is the cipher text of the *j*th stage.

The hardware design for performing the above function consists of two main parts: the first part is data path and the second part is the control logic, as shown in the Fig.(3). Data path is made up counter, summer, register, ROM, and a shift register.

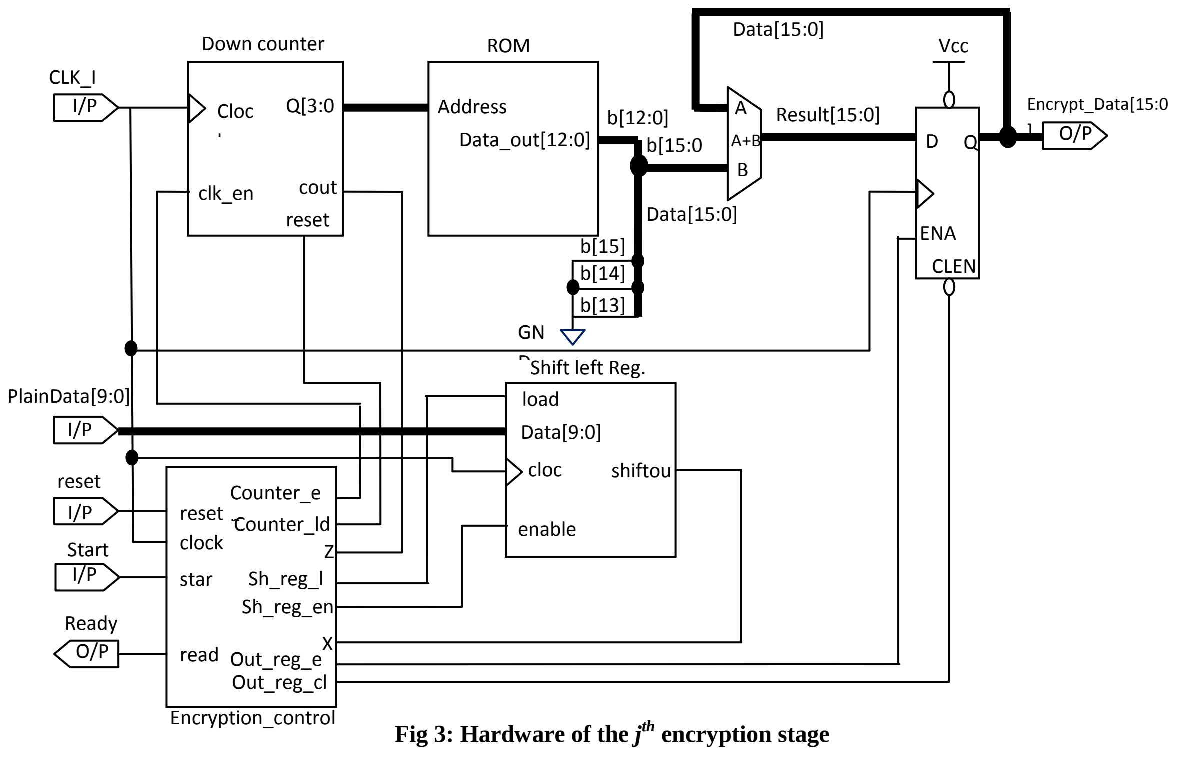


Fig 3: Hardware of the *j*th encryption stage

The summer with the register is used to implement the function of the summer in Eq.(7). Up counter is used for summer index, i.e., keeping track the index of j^{th} element and bit of encryption key and input data, respectively. Shift register is used for addressing individual bits of the input plain text. ROM is used as lookup table for storing encryption key elements. The control part of the designed circuit is used for coordination the activities of the data path in such a way that realizes Eq.(7). State machine approach is used for designing the control circuit.

4.2 Hardware design of j^{th} stage of decryption part of a multistage knapsack cryptosystem

As in encryption part, decryption part consists of *k* semi similar stages. The stages in the decryption part are arranged in the reverse order of the stages in the encryption part. Each stage of decryption part has also five types of signals, as shown in the Fig.(4).

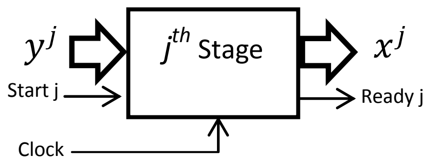


Fig4: Decryption stage

These signals are data input (y^j), data output (x^j), clock, command (*start j*), and status (*ready j*). y^j and x^j correspond to encrypted data and plain data, respectively. *Start j* signal initiates the logic circuit of stage i to start decryption process. At the end of decryption process, *ready j* signal is activated. Start and ready signals ease the process of synchronization between stages, as in encryption part. Therefore, construction of multistage is very simple and easy.The function of the j^{th}decryption stageis to computex^j from y^j according to algorithm (1).

Algorithm 1

INPUT: y, m, w^{-1}
OUTPUT: X

```
$\gamma \leftarrow y \cdot w^{-1} \bmod(m)$
for $i \in n-1..0$
  if $\gamma \geq \alpha_i$
    $\gamma \leftarrow \gamma - \alpha_i$
    $x_i \leftarrow 1$
return(X)
```

Where α_i is the j^{th} element of the decryption key,
w^{-1} and m are predefined parameters (constants),
y encrypted data,
*X*plain data.

The hardware design for implementing algorithm (1) for a one stage requires two parts circuit. In the first part, $y.w^{-1}$ mod m is computed. For this purpose, modified Montgomerymodular multiplication algorithm presented in section 3 is directly implemented. The hardware implementation of this algorithm requires data path consisting of shift register, multiplexers, counter, and an adder. The coordination of the different parts of the data path is done using a control unit, that can be implemented using state machine approach. The complete logic circuit for calculating $y.w^{-1}$ mod m is shown in Fig.(5).

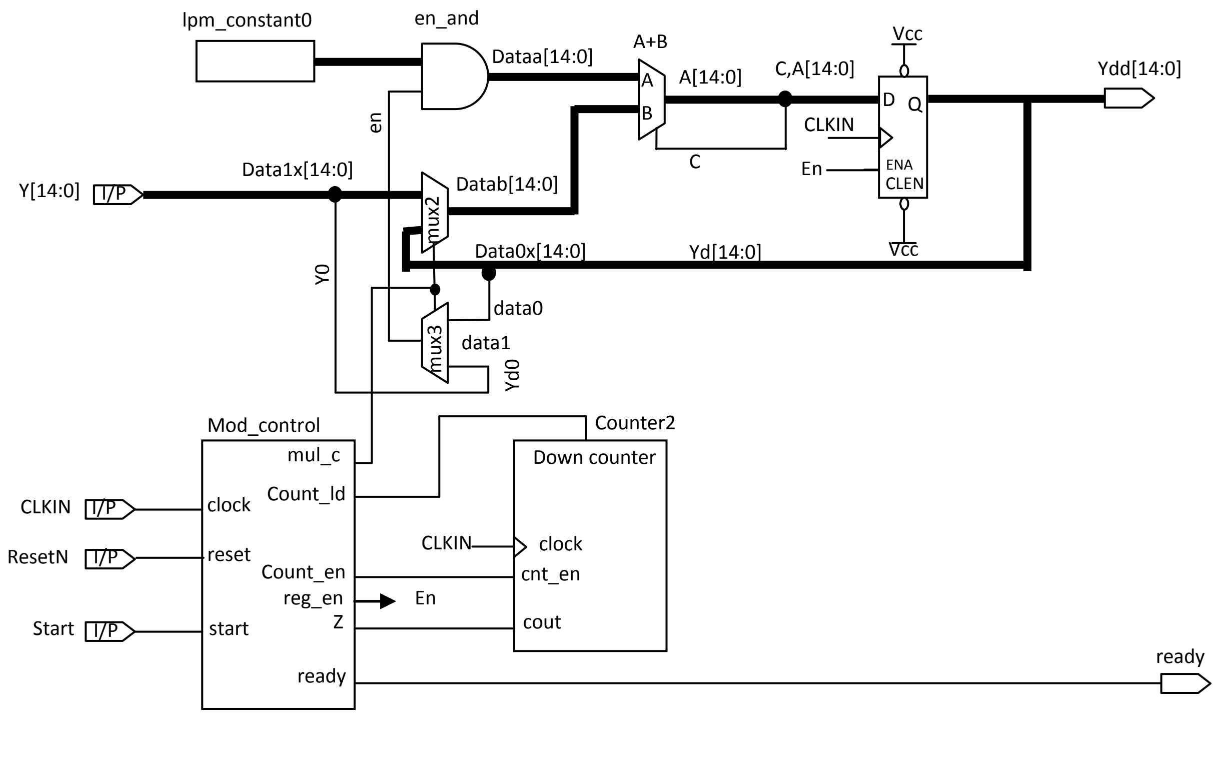


Fig5: Hardware implementation of modified Montgomery modular multiplication

In the second part of the algorithm (1), the plain data (X) is calculated from output of first part. As shown in the algorithm (1), hardware design consists of data path and a control unit also. Data path consists of down counter, ROM, registers, subtractor, and multiplexers. Down counter is used for indexing the loop in the algorithm. ROM is for storing and retrieving decryption keyof that stage. Subtractor is for subtraction and comparing process. Multiplexers are for direct date through different parts of the data path of the circuit. As in the previous circuits, coordination of the different parts of data path is done using the control unit, which is designed using state machine approach. The complete circuit of the second part is shown in the Fig.(6).

5. HARDWARE IMPLEMENTATION OF THREE STAGES KNAPSACK CRYPTOSYSTEM

As an example, using the hardware design of the previous section three stages knapsack public key cryptosystem is implemented using Cyclone IV FPGA used DE2-115 development kit. The public and the secret keys of these three stages are computed usingthe procedure of section 2 and as follows:

First stage:N_1 = 10
m_1 = 4093 w_1 = 512 &w_1^{-1}= 1367
E_1 (public or encryption key) $\mathbf{A_1}$= {1536, 2560,1539, 512, 2066, 551, 2126, 3231, 5908, 3726}
D_1 (secret or decryption key) $\mathbf{A_1'}$ = {3, 5,11, 25, 52, 105, 212, 430, 871, 1750}
Second stage:N_2 = 15
m_2= 44357 w_2= 8192&w_2^{-1} = 3395
E_2 (public key or encryption key) $\mathbf{A_2}$ = {8192, 24576, 40960, 1398, 38961, 5592,42149, 39941, 35525, 26693, 9029, 6469, 1349, 7493 }
D_2 (secret key) $\mathbf{A_2'}$ = {1, 3, 5, 11, 21, 44, 87, 173, 346, 692, 1384, 2768, 5540, 11084, 22174}
Third stage:N_3 = 19
m_3= 680337 w_3 = 16384 w_3^{-1} = 5938
E_3 (public or encryption key) $\mathbf{A'_3}$ = {1, 2, 5, 9, 20, 39, 79, 163, 329, 661, 1325, 2653, 5311, 10627, 21257, 42519, 85041, 170085, 340173}
D_3 (secret or decryption key) $\mathbf{A_3}$ = {16384, 32768, 147456, 327680, 638976, 613999, 629581, 627977, 624769, 618353, 605521, 612625, 626833, 622481, 646545, 661905, 12288, 73728}

5.1 Hardware implementation of encryption part

Three similar encryption stages of Fig.(3) with different encryption key of length 10, 15 & 19 are designed and implemented on DE2-115 development kit using Quartus II software. In each stage, all parts of the data path, except ROM, are implemented using built-in megafunctions of Quartus II software. The ROM is implemented using VHDL. State machine tool of Quartus II is used to implement the function of control circuit. The three encryption stages are compiled using Quartus II software. They used <1% of available resources of DE2-115 kit. The circuit is functionally simulated using ModelSim-Altera software using keys length of 10, 15 and 19 elements. Simulation result is shown in the Fig.(7). The first, second and third stage require 11, 16 and 20 clock cycles to complete their operation respectively.

To verify the hardware implementation of the encryption part of the three stages cryptosystem, the following message is applied to this system:
Message is {30, 843}
Output of 1stEncryption Stage {6686, 14377}
Output of 2ndEncryption Stage {156918, 32029}
Output of 3rdEncryption Stage{4204879, 4287230}

5.2 Hardware implementation of decryption Part

Three similar decryption stages of different decryption key are implemented. They are arranged in reverse order to encryption stages. Each stage consists of circuit of Fig(5) and Fig.(6) which are implemented using Quartus II. Most parts of the two circuits (except control units and the ROM) are implemented using built in megafunctions of Quartus II software. ROM for storing decryption key is implemented using VHDL. Control units of the two parts of each stage of decryption part are implemented using state machine tools of Quartus II software. The compiled circuits of the three decryption stages require < 1% of available resources of DE2-115 kit. The circuit is functionally simulated using ModelSim-Altera tool. The simulation result is shown in the Fig.(8).Encrypted message is applied to the three stage decryption part and the result is :
Output of 3rddecryption Stage {156918, 32029}
Output of 2ndDecryption Stage {6686, 14377}
Output 1stDecryption Stage {30, 843} which is the original message

6. CONCLUSIONS

Efficient and optimized implementation of multistage trapdoor knapsack public key cryptosystem on DE2-115 FPGA development kit has been presented. Multistage Knapsack is very secure and so far, it is not broken in a contrary to single stage which is insecure. Moreover, hardware implementation of multistage requires smaller word length than single stage. Montgomery's modularmultiplication algorithm is modified with great computational reduction. Three stages knapsack public cryptosystem are implemented on DE2-115 FPGA development kit. The either implemented stages (encryption or decryption) take <1% of available FPGA resources. The required clock cycles for encryption process of a one stage are equal to the encryption key length plus one. Whereas decryption process of a one stage requires twice decryption key length. The stages can be easy operated in pipeline to speedup cryptosystem operations.

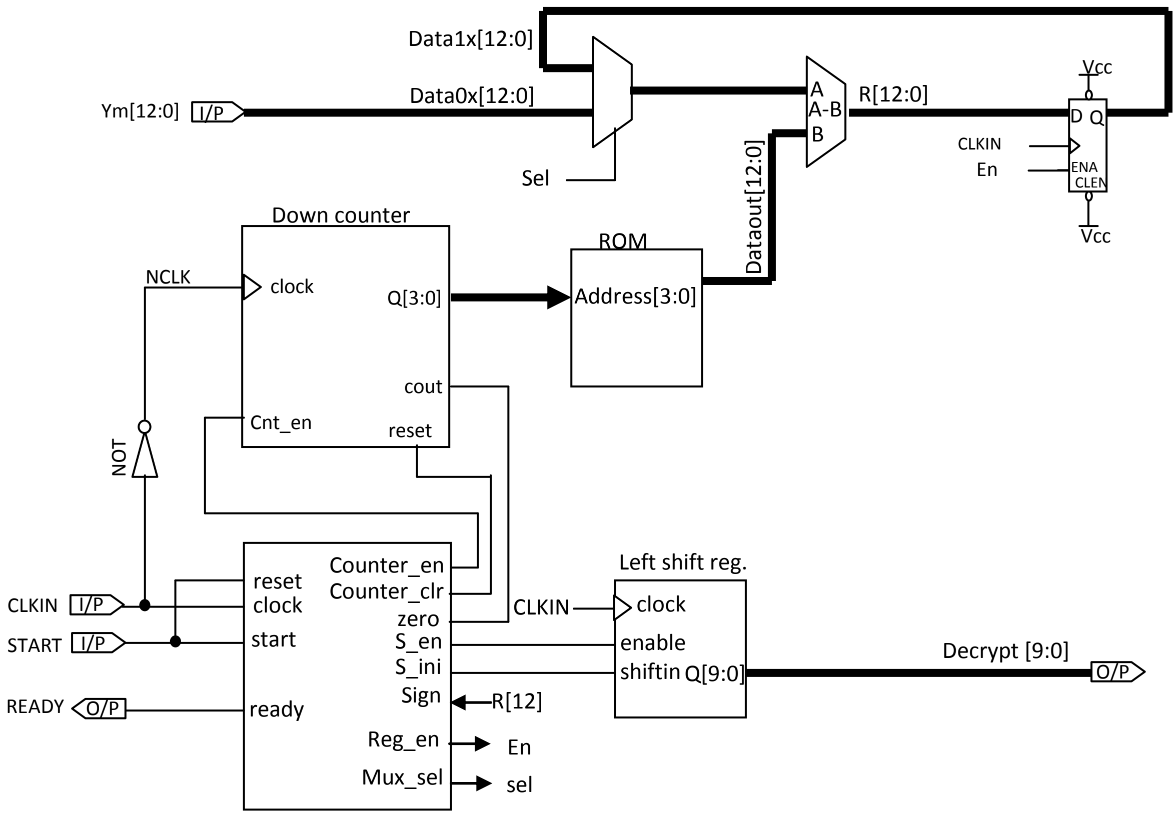
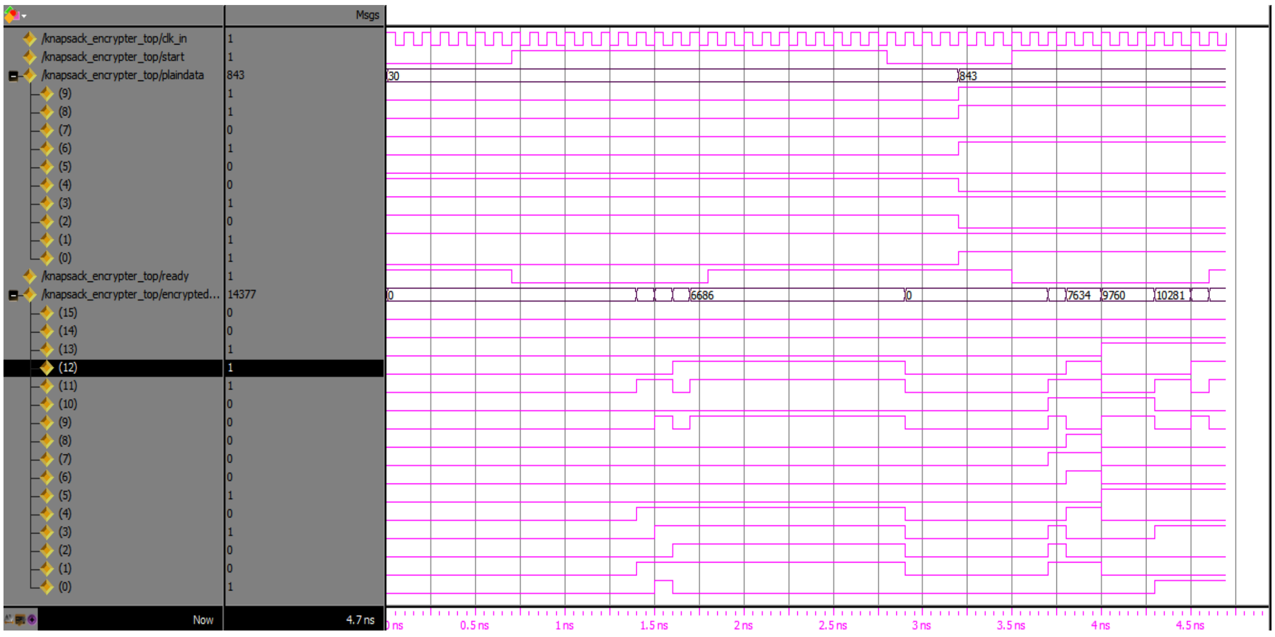


Fig 6: Hardware implementation of second part of algorithm 1



```
                                         Msgs
/knapsack_encrypter_top/clk_in           1
/knapsack_encrypter_top/start            1
/knapsack_encrypter_top/plaindata        843
  (9)                                    1
  (8)                                    1
  (7)                                    0
  (6)                                    1
  (5)                                    0
  (4)                                    0
  (3)                                    1
  (2)                                    0
  (1)                                    1
  (0)                                    1
/knapsack_encrypter_top/ready            1
/knapsack_encrypter_top/encrypted...     14377
  (15)                                   0
  (14)                                   0
  (13)                                   1
  (12)                                   1
  (11)                                   1
  (10)                                   0
  (9)                                    0
  (8)                                    0
  (7)                                    0
  (6)                                    0
  (5)                                    1
  (4)                                    0
  (3)                                    1
  (2)                                    0
  (1)                                    0
  (0)                                    1
Now                                      4.7 ns
```

Fig 7 : Hardware simulation result of first encryption stage

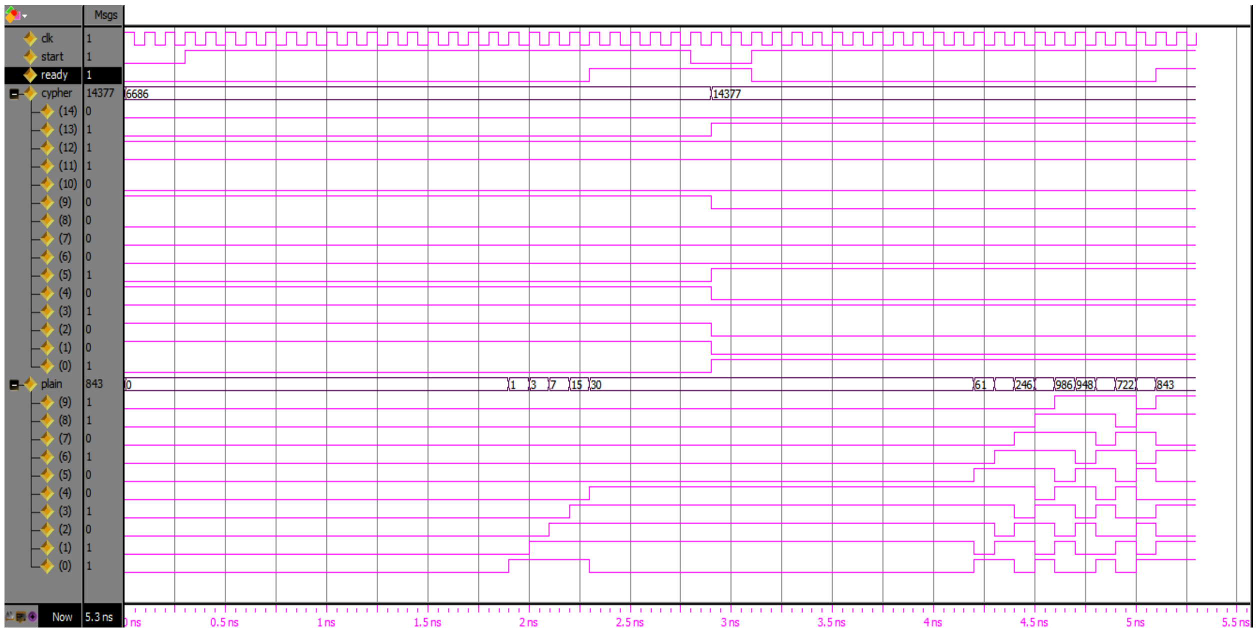


Fig8: Hardware simulation result of first decryption stage

7. REFERENCES

[1] W. Diffie and M. Hellman : New direction in cryptography,IEEE Trans. Information Theory Vol. IT-22 pp. 644-654, 1976.

[2] R.C. Merkle and M. Hellman: Hiding information andsignatures intrapdoor knapsacks, IEEE Trans. InformationTheory Vol. IT-24, pp 525-530, 1978.

[3] B. Chor and R.L. Rivest : A knapsack type public keycryptosystembased on arithmetic in finite fields, IEEE Trans.Information Theory Vol. IT-34, pp901-909, 1988.

[4] L. Adleman:On Breaking Generalized Knapsack Public KeyCryptosytems, Proc.15th Annual ACM Symposium onTheory of Computing, pp. 402-412,1983.

[5] A.M. Odlyzko:Cryptoanalytic attacks on the multiplicativeknapsack cryptosystem and on Shamir's fast signaturescheme, IEEE Trans.Information Theory Vol. IT-30, pp. 594-601, 1984.

[6] C.P. Schnorr and H.H. Horner: Attacking the Chor-Rivestcryptosystem by improved lattice reduction, J. ElectronicColloquium on Computational Complexity, ECCC Vol.2, No.26 pp. 1-12, 1995.

[7]M.K. Lai : Knapsack cryptosystems, the past and the future, available at http:// ww.ics.uci.edu/~ming/knapsack.html, 2001.

[8] M. S. Lee : Improved cryptanalysis of a knapsack-basedprobabilisticencryption scheme, J. Information Science 222,pp. 779-783, 2013.

[9] H.A. Hussain, J.W.A. Sada, S.M. Kalipha: New multistageknapsack public-key cryptosystem, Int. J. Systems ScienceVol. 22, No. 11 pp.2313-2320, 1991.

[10]P.L. Montgomery: Modular multiplication without trialdivision, Math. Comput.Vol.44, No. 170, pp. 519-521, 1985.

[11]G.D.Sutter, J.P. Deschamps& J.L. Iman'a: Modularmultiplication and exponentiation architectures for fast RSAcryptosystem based on digital serial computation, IEEETrans. On Industrial Electronics Vol. 58, No.7 July 2011.