JOURNAL OF KOREA MULTIMEDIA SOCIETY, VOL. 6, NO. 4, JULY 2003

Cellular Automata and It's Applications

Jun-Seok Lee, Hyun-Ho Cho and Kyung-Hyune Rhee

ABSTRACT

This paper presents a concept of cellular automata and a modular exponentiation algorithm and implementation of a basic ElGamal encryption by using cellular automata. Nowadays most of modular exponentiation algorithms are implemented by a linear feedback shift register(LFSR), but its structure has disadvantage which is difficult to implement an operation scheme when the basis is changed frequently. The proposed algorithm based on a cellular automata in this paper can overcome this shortcomings, and can be effectively applied to the modular exponentiation algorithm by using the characteristic of the parallelism and flexibility of cellular automata. We also propose a new fast multiplier algorithm using the normal basis representation. A new multiplier algorithm based on normal basis is quite fast than the conventional algorithms using standard basis. This application is also applicable to construct operational structures such as multiplication, exponentiation and inversion algorithm for ElGamal cryptosystem.

Key words: cellular automata, modular operation, ElGamal cryptosystem, information security

1. INTRODUCTION

A cellular automata(CA) is a mathematical model for complex natural system containing large number of simple identical components with local interactions. Since the pioneering work of John von Neumann during 1950s, S. Wolfram et. al. have studied one-dimensional, null and periodic boundary additive CAs with the help of polynomial algebra[1,2]. And he first also introduced CA in application of cryptography[3]. Another applications on cryptography are reported in[4-7]. Das et. al. analyzed CA with the help of matrix algebra[8]. Applications of CAs in various fields have also been proposed such as theory and applications for error correcting codes, VLSI applications, and test pattern generators as an alternative to LFSR for built-in self-test of digital systems[9-15]. A study for the existence of isomorphism between LFSR and 1-D linear hybrid cellular automata(LHCA) is reported in[16-18].

Since a CA has a simple, regular and modular structure, it is useful for hardware design in VLSI. In this paper, it is shown that advanced programmable cellular automata(APCA) can be carried out modular exponentiation over extended field $GF(2^n)$ of $GF(2)$ and the ElGamal cryptosystem [19].

This paper is organized as follows. Section 2 introduce a basic concept of CA and ElGamal cryptosystem[20]. In Section 3, the structure of operation using the APCA is designed. And we propose a new fast scheme based on normal basis to implement the ElGamal cryptosystem in Section 4. Finally, we have conclusions in Section 5.

2. CA AND THE ELGAMAL PUBLIC KEY CRYPTOSYSTEM

2.1. Cellular automata(CA)

A cellular automata is a finite state machine which consists of a simple identical array of sites

* *Jun-Seok Lee is with the Dept. of Computer Science, Pukyong National Univ., Korea. E-mail :jslee@lisia21.net*
* *Hyun-Ho Cho is with the Dept. of Internet Security, Dong-Pusan College, Korea.*
* *Kyung-Hyune Rhee is with the Division of Electronic, Computer and Telecommunication Engineering, PKNU., Korea. E-mail : khrhee@pknu.ac.kr*

called cell. At discrete time step, each cell evolves in synchronous to the output of a function called a transition function or rule. The rule is used to determine the next state of a given cell based on state value of its neighbors. If cell i can communicate only with its k neighbors, then it is called k-neighborhood CA and the next state of cell depends on the present states of k neighbors by transition function.

Table 1 specifies some rules of 2-state 3-neighborhood NBCA. 8-bit binary number in each low, expressed in equivalent decimal form gives a convenient scheme for representing the CA rule. For a CA with cells having only 2-state 3-neighborhood, there can be a total of 2^3 distinct neighborhood configuration and a $2^{2^3}=256$ distinct rules.

The next state transition of the i-th cell can be represented as a function of the present state of the $(i-1)$-th, i-th and $(i+1)$-th cell:

$$s_i^{t+1} = f(s_{i-1}^t,\ s_i^t,\ s_{i+1}^t)$$

where, f is known as the rule of the CA denoting the combination logic and s_i^t is state of i-th cell at time step t. Rule 90 and rule 150 in Table 1 can be expressed such as following.

$$s_i^{t+1} = s_{i-1}^t \oplus s_{i+1}^t$$
$$s_i^{t+1} = s_{i-1}^t \oplus s_i^t \oplus s_{i+1}^t$$

Fig. 1 specifies a cell structure of 3-neighborhood CA.

Realization of different CA configuration on the same structure can be achieved using a control logic to control appropriate switches. Such a

Table 1. Example of rule representation

Rules	111	110	101	100	011	010	001	000
rule 60	0	0	1	1	1	1	0	0
rule 90	0	1	0	1	1	0	1	0
rule 150	1	0	0	1	0	1	1	0
rule 102	0	1	1	0	0	1	1	0

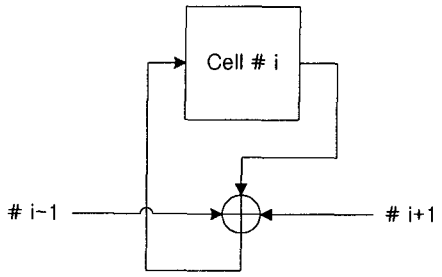


Fig. 1. A cell structure of 3-neighborhood CA

structure is referred to as a programmable CA (PCA). This structure is shown in Fig. 2 for 3-neighborhood CA, where *Cl*, *Cs*, and *Cr* are control signals applied in each cell at time step respectively.

In the CA structures, there are various boundary conditions. That are null, periodic and intermediate boundary conditions. If extreme cells, left most cell and right most cell, are connected to null, logical 0, then it is called to NBCA(null boundary CA) and if extreme cells treat to successive cell then it is called to PBCA(periodic boundary CA) and if the next state of the left and right most cell are connected to the one next to it, it is called to IBCA(intermediate boundary CA).

CA can be classified in various way. The CA employing only XOR logic are referred to as linear CA, otherwise they are called nonlinear CA. If in a CA the same rule applies to all the cells, then the CA is called a uniform CA, whereas if different

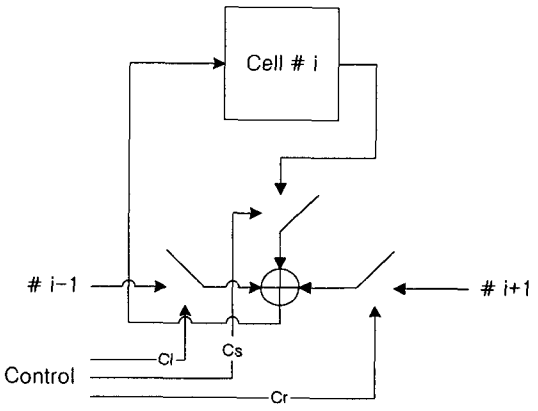
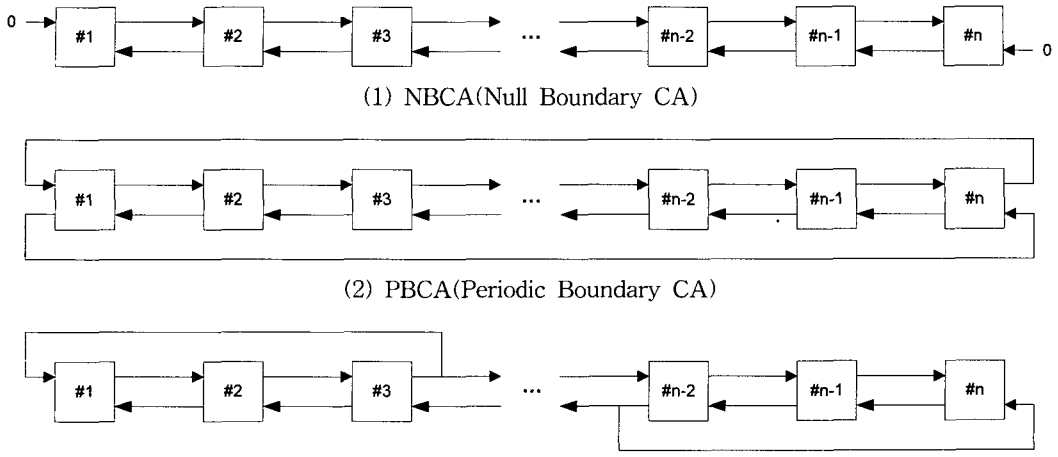


Fig. 2. A cell structure of programmable CA(PCA)



(1) NBCA(Null Boundary CA)

(2) PBCA(Periodic Boundary CA)

(3) IBCA(Intermediate Boundary CA)

Fig. 3. Boundary conditions of 1-D CA(NBCA, PBCA and IBCA)

rules are applied over different cells, then it is called a hybrid CA. Fig. 4 shows a 4-cell null boundary linear hybrid CA(4-cell NBLHCA) with rules (90, 150, 90, 150).

2.2. The ElGamal public key encryption

The ElGamal cryptosystem is a public key cryptosystem based on the difficulty of the discrete logarithm problem over the finite fields. This algorithm have 3-step processes as following.

2.2.1. Key generation step

Each user create a public key and corresponding private key

(a) Select a large random prime p and a generate a of the multiplicative group Z_p^* of the integer modulo p

(b) Select a random integer a and compute $a^a \bmod p$ $(1 \le a \le p-2)$

(c) User A's public key is (p, a, a^a) and private key is a

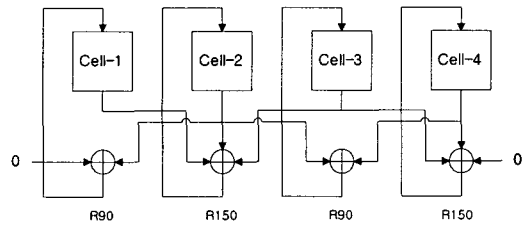


Fig. 4. A 4-cell NBLHCA with rules (90, 150, 90, 150)

2.2.2. Encryption step(sender)

Sender B encrypts a message m for receiver A

(a) Obtain user A's public key (p, a, a^a)

(b) Represent the message as an integer m in the range $Z_p = (0, 1, \cdots, p-1)$

(c) Select a random integer k, $(1 \le a \le p-2)$

(d) Compute $\beta = a^k \bmod p$ and $\gamma = m \cdot (a^a)^k \bmod p$

(e) Send the ciphertext c to user A, $c = (\beta, \gamma)$

2.2.3. Decryption step(receiver)

Receiver A decrypts message m from ciphertext c

(a) Compute $\beta^{p-1-a} = \beta^{-a} \bmod p$ using the private key a

(b) Decrypt message m by computing $\gamma \cdot (a^{-a})^k \bmod p$

3. A PROPOSED STRUCTURE TO IMPLEMENT THE ELGAMAL CRYPTOSYSTEM

In this section we describe algorithms to

implement the ElGamal cryptosystem using the elements of extended field constructed by generator $g(x)$.

To implement this system, first we propose a cell structure of advanced programmable cellular automata(APCA) and algorithms for multiplication, exponentiation and inversion by using the APCA structure. Moreover these proposed algorithms are simulated by C++ compiler for a small size n.

3.1. Advanced programmable cellular automata

A cell structure of APCA improves a structure of PCA introduced by[4]. We only consider a cell structure of APCA with 3-neighborhood dependencies. In APCA, each cell has three control signals *Cs*, *Cr* and *sel*. The control signal *sel* is used to choose cell's neighbors, that is, if *sel* signal is logical 0, then cell operation chooses itself as self input and right neighbor as right input. Otherwise cell operation choose *Data* value as self input and *Rn* value as right input.

The rules of the APCA are also based on combination of control signals of *Cs* and *Cr*, and the rules applied to each of APCA are same as those defined in Table 2.

The proposed APCA is a flexible structures, so it can be used as a conventional 3-neighborhood

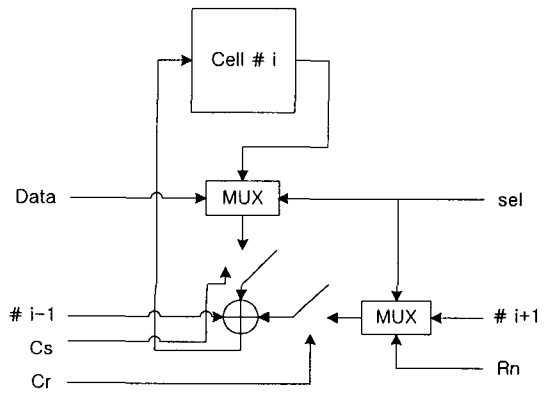


Fig. 5. A cell structure of advanced PCA

Table 2. Rules of APCA

Cs	*Cr*	Rules	Transition function f
1	0	rule 60	$s_i^{t+1} = s_{i-1}^t \oplus s_i^t$
0	1	rule 90	$s_i^{t+1} = s_{i-1}^t \oplus s_{i+1}^t$
1	1	rule 150	$s_i^{t+1} = s_{i-1}^t \oplus s_i^t \oplus s_{i+1}^t$
0	0	rule 240	$s_i^{t+1} = s_{i-1}^t$

CA as well as LFSR and the division circuit by choosing control signal values properly. If *sel* signal is 0 then APCA is a conventional 3-neighborhood CA. If *sel* signal is 1 then this structure is the polynomial modular multiplier which has *Data* value and current CA state as input values, where signals *Cs* and *Cr* are coefficients of multiplicand and primitive polynomial respectively. This sturcture is a new modular multiplier structure using a cellular automata.

3.2. The polynomial modular multiplier using the APCA

The modular multiplication operation can be constructed by performing multiplication and division operation at the same time. In general, these circuits are constructed by LFSR[19]. However, we consider a APCA structure for the modular multiplication circuit such like Fig. 6. A set-up values of *Cs*, *Cr*, *sel* and *Data* can be established as following:

- *Data* : coefficients of a multiplier polynomial
- *Cs* : coefficients of a multiplicand polynomial
- *Cr* : coefficients of a primitive polynomial
- *sel* : logical 1

where, the value of *Data*, that is, coefficients of multiplier polynomial, is applied in a descending order at all cell.

3.3. The exponentiation and inversion algorithm

The polynomial modular exponentiation,

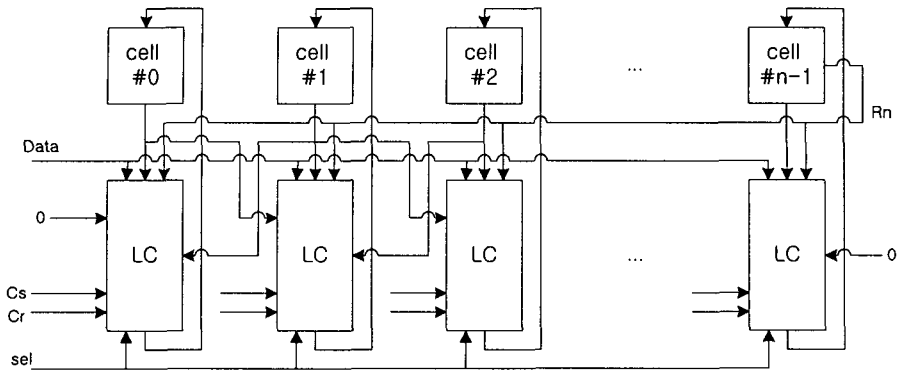


Fig. 6. The multiplication circuit using a APCA

$s(x)^e \bmod p(x)$, can be implemented by using two multiplication circuits such as a circuit shown in Fig. 6, where e is represented by an n-tuple binary vector, $e = (e_0, e_1, e_2 \cdots, e_{n-1})$ and $p(x)$ is an n-degree primitive polynomial generated over extended field $GF(2^n)$. One circuit is used for a multiplication operation and other circuit is used for a square-multiply. The procedure of this exponentiation algorithm is given as following:

- Step 1 : $C = 1$, $E = s(x)$
- Step 2 : If $e_i = 1$ then compute $C = C \cdot E \bmod p(x)$ otherwise go to Step 3
- Step 3 : Compute $E = E^2 \bmod p(x)$
- Step 4 : Repeat Step 2 and Step 3 till n-1 is reached
- The result is in C.

Where, Step 2 is achieved by a multiplier proposed in section 3.2 and square-multiply in Step 3 is also achieved using another multiplier. The simulation is demonstrated by a software tool.

Now, let $g(x)$ be a generator polynomial over extended field $GF(2^n)$. Then any element over extended field $GF(2^n)$ is expressed by a generator $g(x)$ according to the standard basis and it is clear that $g(x)^{2^n-1} = 1$. Therefore inversion of any element over $GF(2^n)$ can be computed by using $g(x)^{-a} = g(x)^{2^n-1-a}$. This is also implemented by an exponentiation algorithm as mentioned above.

3.4. Implementation of the ElGamal cryptosystem

This section shows the implementation of ElGamal cryptosystem over $GF(2^n)$ using the CA. To implement the ElGamal cryptosystem, procedures are given as follows:

3.4.1. Key generation step

(a) Select $p(x)$ of degree n and generator $g(x)$ over extended field $GF(2^n)$

(b) Select random number a and compute $g(x)^a \bmod p(x)$

(c) User A's public key is $(g(x), p(x), g(x)^a \bmod p(x))$ and private key is a

3.4.2. Encryption step

(a) Replace the message $m(x)$ into the element of extended field $GF(2^n)$

(b) Select random number k, where $1 \le k \le 2^n - 2$

(c) Compute $\beta = g(x)^k \bmod p(x)$ and $\gamma = m(x) \cdot g(x)^{ak} \bmod p(x)$

(d) Send the ciphertext $c = (\beta, \gamma)$ to user A

3.4.3. Decryption step

(a) Compute $\beta^{-a} = \beta^{2^n-1-a} = g(x)^{ak}$ using the private key a

(b) Decrypt message $m(x)$ by computing $\gamma \cdot \beta^{-a} = (m(x) \cdot g(x)^{ak} \cdot g(x)^{-ak}) \bmod p(x)$

4. A NEW MULTIPLIER BASED ON NORMAL BASIS

The extension field $GF(2^n)$ of $GF(2)$ is considered as an n-dimensional vector space defined over nk $GF(2)$. To express all elements of a field, a set of n linearly independent vectors can be chosen, which is called as a basis. The most commonly used bases are following :

- Standard basis : an easy way to choose a basis is an ordered set $(1, \alpha, \alpha^2, \cdots, \alpha^{n-1})$ where $\alpha \in GF(2^n)$. It is called the standard basis.
- Normal basis : If the set of elements $(\beta^{2^0}, \beta^{2^1}, \beta^{2^2} \cdots, \beta^{2^{n-1}})$ forms a basis, then this is called the normal basis, where $\beta \in GF(2^n)$.

In general, the operation structure based on normal basis is very effective in performing operations such as finding the inverse element and the squaring or exponentiation of a finite field element. In this section, we propose a new multiplier based on normal basis. A proposed structure is very fast than other conventional algorithm in performing the arithmetic operation such as multiplication, exponentiation and inversion, and it is possible to construct the operation structure for implementation of ElGamal encryption algorithm.

4.1. Multiplier operation based on normal basis

The multiplier implementation of a field $GF(2^n)$ is very difficult in the view point of hardware complexity. However, in the case of normal basis, by using a suitable element β, a hardware complexity can be reduced, where β is an element of a field $GF(2^n)$. This basis is called the optimal normal basis. The construct of optimal normal basis is studied by Gao, et al.[21-26]. This multiplier is constructed as following.

Let $x, y \in GF(2^n)$ be elements represented by normal basis. Then these can be written by

$$x = (a_0, a_1, \cdots, a_i, \cdots, a_{n-1})$$

$$y = (b_0, b_1, \cdots, b_i, \cdots, b_{n-1})$$

where, a_i and b_i are elements of $GF(2)$, $i \geq 1$. Now, let $z = x \cdot y$, then z is written by

$$z = \sum_{k=0}^{n-1} b_k \cdot (x \cdot \beta^{2^k})$$

That is, to implement $z = x \cdot y$, it is required the computation of $x \cdot \beta^{2^k}$ $(0 \leq k \leq n-1)$. Our algorithm only requires the computation of $x \cdot \beta$. The procedure of our algorithm is as following:

- Step 1 : Compute $x \cdot \beta$
- Step 2 : If $b_k = 1$, store the result in a buffer otherwise go to Step 3
- Step 3 : Left cyclic shift values of x and buffer respectively by 1-bit
- Step 4 : Repeat for Step 1 to Step 3 till $k = n-1$

Moreover, the implementation of $x \cdot \beta$ can be accomplished with only two-cell dependency by using an optimal normal basis and the hardware complexity requires only n-1 XOR gates, and the number of clock for this computation is only n.

A pseudo-code for this multiplier is given as following.

```
multiplication(x, y){
  buffer = 0;
  for(k=0; k<=n-1; k++){
    if(IsBitSet(y,k)) buffer^=Fbox(x);
    x=left_cyclic_shift(x);
    buffer=left_cyclic_shift(buffer);
  } return buffer;
}
```

For example, in the case of a normal basis multiplier with $p(x) = 1 + x^3 + x^{10}$ as a primitive polynomial and $\beta = \alpha^{93}$ as optimal normal basis, we get a vector representation and logic circuit to $x \cdot \beta$ given in Fig. 7. This logic circuit can be easily

$$x \cdot \beta = (a_5,\ a_0 \oplus a_5,\ a_5 \oplus a_8,\ a_5 \oplus a_7,\ a_2 \oplus a_5,\ a_5 \oplus a_6,\ a_3 \oplus a_5,\ a_5 \oplus a_9,\ a_1 \oplus a_5,\ a_4 \oplus a_5)$$

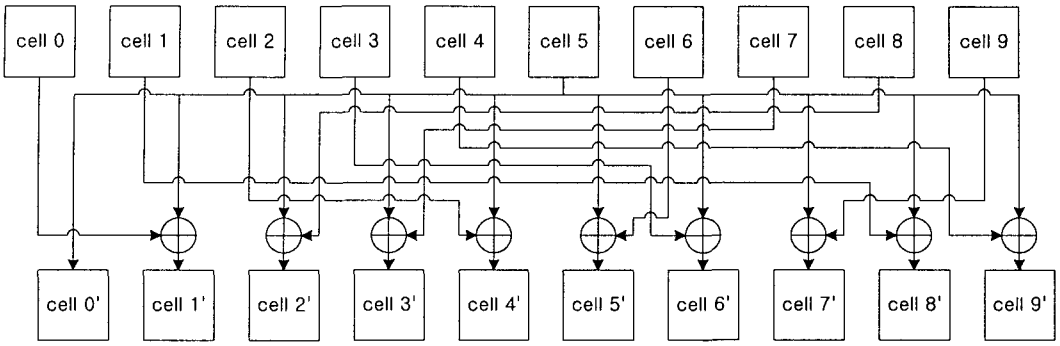


Fig. 7. The logic circuit of $x \cdot \beta$ for $p(x) = 1 + x^3 + x^{10}$, $\beta = \alpha^{93}$

implemented by LHCA.

4.2. Squaring operation

When field elements over $GF(2^m)$ are expressed in normal basis, representation of x^2 for any element x can be easily obtained from the representation of x by simply a right cyclic shift. This is a natural property of normal basis elements. Therefore, if $x=(a_0,\ a_1,\ a_2,\ \cdots,\ a_{n-1})$ is any element which is expressed in normal basis then x^2 is written by

$$x^2=(a_{n-1},\ a_0,\ a_1,\ \cdots,\ a_{n-2}).$$

As a shown, it has a very brief representation computing to a standard basis.

4.3. Exponentiation operation

Exponentiation operation can be performed by using the multiplication and squaring operation described in Section 4.1. A pseudo-code is given as following:

```
Exponentiation(x, power){
  initialize mul_buffer;
  initialize squ_buffer;
  if(power==0) return mul_buffer;
  if(power==1) rerurn x;
  temp=x;
  for(k=0; k<=I; k++){
    if(k==0){
      squ_buffer=mul_buffer;
      if(IsBitSet(power, k))
      mul_buffer=multiplication(x, squ_buffer);
    }
    else{
      temp=right_cyclic_shift(temp);
      squ_buffer=temp;
    }
    if(IsBitSet(power, k))
      mul_buffer=multiplication(mul_buffer,
      squ_buffer);
  } return mul_buffer;
}
```

4.4. Inversion operation

The inversion of any element expressed in normal basis is also computed by using the fact that $x^{-1} = x^{2^n-2}$. This algorithm is implemented by our proposed algorithms such as multiplication, squaring and exponentiation algorithm as mentioned in Section 4.3.

5. CONCLUSIONS

We considered one dimensional CA as a polynomial modular exponentiation operation tool for a given extended field $GF(2^n)$ of $GF(2)$. To implement this structure we reconstructed a cell

structure of CA. The proposed structure is a very flexible one which can be effectively applied to the operation scheme which needs a frequent change of the multiplier, multiplicand and primitive polynomial. Using the proposed structure we implemented ElGamal cryptosystem and simulated for a simple example by the C++ complier.

Furthermore, we proposed a new multiplier algorithm based on optimal normal basis, and constructed operation schemes for the implementation of ElGamal cryptosystem. Our proposed scheme is very fast algorithm than the conventional schemes to compute the arithmetic operations such as multiplication, squaring, exponentiation and inversion operation.

For further research, we are trying to improve the performance of our scheme and apply it to other crytposystems. It would be interesting to try to construct various CA which are applicable to cryptographic primitives such as modular multiplication, division and exponentiation over an extended field $GF(p^n)$, $n \geq 1$. Moreover the proposed structure can be easily modified to apply basic Diffie-Hellman key exchange protocol.

6. REFERENCES

[1] Olivier Martin, Andrew M. Odlyzko, S. Wolfram, Algebraic properties of cellular automata, *Communications in Mathematical Physics*, vol.93, pp.219-258, March 1984.

[2] S. Wolfram, *Cellular Automata and Complexity*, Addison Welsely Publishing Company, 1994.

[3] S. Wolfram, Cryptography with Cellular Automata, *in Advanced in Cryptology, Crypto 85 Proceedings, Lecture Notes in Computer Science*, vol.218, pp.429-432, 1986.

[4] S. Nandi, B.K. Kar, P.P. Chaudhuri, Theory and applications of cellular automata in cryptography, *IEEE Transactions of Computers*, vol.43, pp.1346-1357, Dec. 1994.

[5] M. Mihaljevic, Y. Zhang, H. Imai, A Fast and Secure Stream Cipher based on Cellular Automata over $GF(2^n)$, *IEEE Global Telecommunications Conference*, GLOBECOM 98, vol. 6, pp.3250-3255, 1998.

[6] M. Mihaljevic, H. Imai, A Family of Fast Keystream Generations Based on Programmable Linear Cellular Automata over $GF(q)$ and Time-Variant Table, *IEICE Transactions on Fundamentals*, vol.E82-A, no.1, pp. 32-39, 1999.

[7] B. Srisuchinwong, T.A. York, Ph. Taslides, A Symmetric Cipher using autonomous and non-autonomous cellular automata, *IEEE Global Telecommunications Conference, GLOBECOM 95*, pp.1172-1177, 1995.

[8] A.K. Das, A. Ganguly, A. Dasgupta, S. Bhawmik, P.P. Chaudhuri, Efficient characterization of cellular automata, *IEE Proceeding E, Computer and Digital Techniques*, vol.137, no.1, pp.81-87, Jan. 1995.

[9] S. Lin, D.J. Costello Jr., *Error Control Coding: Fundamentals and Applications*, Prentice-Hall, Inc., Englewood Cliffs, New Jersey, 1983

[10] P.P. Chaudhuri, A.R. Chowdhury, S. Nandi, S. Chattopadhyay, *Additive Cellular Automata: Theory and Applications Volume I*, IEEE Computer Society Press, 1997.

[11] B. Srisuchinwong, Ph. Tsalide, T.A. York, P.J. Hicks, A. Thanailakis, VLSI implementation of mod-p multipliers using homomorphisms and hybrid cellular automata, *IEE Proceedings E, Computer and Digital Techniques*, vol.139, no.6, pp.486-490, Nov. 1992.

[12] A.K. Das, P.P. Chaudhuri, Vector Space Theoretic Analysis of Additive Cellular Automata and Its Application for Pseudoexhaustive Test Pattern Generation, *IEEE Transactions on Computers*, vol.42, no.3, pp.340-352, 1993.

[13] J. Rajski, G. Mrugalski, J. Tyszer, Comparative Study of CA-based PRPGs and

LFSRs with Phase Shifters, *IEEE Proceedings of 17th VLSI Test Symposium*, pp.236-245, 1999.

[14] P.S. Cardoso, M. Strum, J.R. de A. Amazonas, W.J. Chau, Comparison between Quasi-Uniform Linear Cellular Automata and Linear Feedback Shift Registers as Test Pattern Generators for Built-In Self-Test Applications, *IEEE Proceedings of 12th Symposium on Integrated Circuits and Systems Design*, pp.198-201, 1999.

[15] S. Bhattacharjee, S. Sinha, C. Chattopadhyay, P.P. Chaudhuri, Cellular automata based scheme for solution of Boolean equations, *IEE Proceedings E, Computer and Digital Techniques*, vol.143, no.3, 1996.

[16] K. Cattell, M. Serra, Analysis of One-Dimensional Multiple-Value Linear Cellular Automata, *IEEE Proceedings of the 20th International Symposium on Multiple Valued Logic*, pp.402-409, 1990.

[17] K. Cattell, J, Muzio, Analysis of One-Dimensional Linear Hybrid Cellular Automata over $GF(q)$, *IEEE Transactions on Computers*, vol.45, no.7, pp.782-792, 1996.

[18] K. Cattell, Synthesis of On-Dimensional Linear Hybrid Cellular Automata, *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems*, vol.15 no.3, pp.325-335, 1996.

[19] Solomon W. Golomb, *Shifter Register Sequences*, Holden-Day, Inc., 1967.

[20] Alfred J. Menezes, Paul C. van Oorschot and Scott A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[21] Shuhong Gao, *Normal Bases over Finite Fields*, Ph. D Thesis, Combinatorics and Optimization, University of Waterloo, 1993.

[22] I.S. Hsu, T.K. Truong, L.J. Deutsch, and I.S. Reed, A Comparison of VLSI Architecture of Finite Field Multipliers Using Dual, Normal, or Standard Bases, *IEEE Transactions on Computers*, vol.37, no.6, June 1988.

[23] Y.R. Shayan, T. Le-Ngoc, The least complex parallel massey-Omura multiplier and its LCA and VLSI designs, *IEE Proceedings G*, vol. 136, no.6, Dec. 1989.

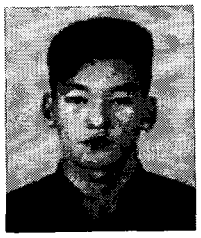
[24] Chung-Chin Lu, A Search of Minimal Key Functions for Normal Basis Multipliers, *IEEE Transactions on Computers*, vol.46, no.5, May 1997.

[25] Rana Barua, Samik Sengupta, Architecture for Arithmetic over $GF(2^m)$, *10th International Conference on VLSI Design*, pp.465-468, Jan. 1997.

[26] B. Sunar, C.K. Koc, An Efficient Optimal Normal Basis Type II Multiplier, *IEEE Transactions on Computers*, vol.50, no.1, Jan. 2001.

Jun-Seok Lee

He received the M.S. degree in Electronic Engineering from Dongeui University, Busan Korea, in 1998, and is currently a Ph.D. candidate in Computer Science at Pukyong National University. His research interests include cellular automata and it's applications, coding theory, cryptographic algorithms, cryptanalysis, secure group communication.



Hyun-Ho Cho

He received the M.S. degree ir Computer Science from Pukyong National University Busan Korea, in 2002, and works as full-time lecturer in Department of Information Security at DongPusan College. His research interests include cellular automata and it's applications, Integer Number theory, Signcryption, Secure Key Management, Elliptic Curve Cryptosystem, secure group communication, Hacking Techniques based on Linux Operating System.

Kyung-Hyune Rhee

He received his M.S. and Ph.D. degrees from KAIST (Korea Advanced Institute of Science and Technology), Daejeon Korea in 1985 and 1992, respectively. He worked as a senior researcher in ETRI (Electronics and Telecommunications Research Institute), Daejeon Korea from 1985 to 1993. He is currently a professor in the Division of Electronic, Computer and Telecommunication Engineering of Pukyong National University. His research interests center on key management and its applications, mobile communications security and security evaluation of cryptographic algorithms.

For information of this article, please send e-mail to: jslee@lisia21.net(Jun-Seok Lee)