Implementation of secure SPN chaos-based cryptosystem on FPGA

Ta Thi Kim Hue, Chu Van Lam, Thang Manh Hoang
School of Electronics and Telecommunications
Hanoi University of Science and Technology
1 Dai Co Viet, Hanoi
Email: thang@ieee.org

Safwan Al Assad
IETR Laboratory
École d’ingénieurs de l’université de Nantes
Site de la Chantrerie -Rue Christian Pauc
B.P 50609 - 44306 NANTES CEDEX 3 - France
Email:safwan.elassad@univ-nantes.fr

Abstract—In this paper, the secure substitution permutation network (SPN) chaos-based cryptosystem is realized in software (C/C++, Altera Quartus II and ModelSim) and implemented in hardware (FPGA platform). The result shows the feasibility and effectiveness of the cryptosystem. As a typical application, the image encryption/decryption is to demonstrate and verify the operation of the cryptosystem hardware.

I. INTRODUCTION

The secure SPN chaos-based cryptosystem has been proved to be the new fast and secure scheme for image encryption based on a robust uniform chaotic pseudo-random generator [1]. There, the main advantage of the key generator is in the providing of dynamic control parameters for substitution and permutation processes.

So far, most of cryptography are realized in software to verify the effectiveness of algorithms other than in hardware for communication and storage applications. In fact, the main advantage of using of hardware for cryptosystem is that multiple parts of system can operate simultaneously, i.e. the generator of secret keys, substitution and permutation processes. That makes chaotic cryptosystems suitable for applications in high speed communications or massive storage.

In this paper, the secure substitution permutation network (SPN) chaos-based cryptosystem as reported in [2] is realized in software (C/C++, Altera Quartus II and ModelSim) and implemented in hardware (FPGA platform). The result shows the feasibility and effectiveness of the cryptosystem. As a typical application, the image encryption/decryption is to demonstrate and verify the operation of the cryptosystem hardware.

II. THEORY OF CHAOS-BASED CRYPTOSYSTEM

This section presents the processes of encryption and decryption of chaos-based secure SPN. Fig. 1 shows the flow chart of the cryptosystem. There, the skew tent map and standard map are used for substitution process and permutation process, respectively. The processes of substitution and permutation are repeated for $rs$ and $rp$ times and all processes have been iterated for $r$ times. The chaotic generator creates control parameters for the chaos maps in those processes. It is noted that control parameters play as a role of secret keys in the chaos-based cryptosystem. Secret keys are changed after iterations, thus, the number of control parameters is really large.

A. The encryption algorithm

Fig. 1(a) shows the encryption process with secret key $K_j$, obtained from the key generator. $K_j$ also means as the control parameters, and which is the set used for substitution and permutation processes, $K_j = \{K_{s_j}; K_{p_j}\}$.

Firstly, the RGB image with the size of $L \times C \times W$ is transformed into the 1-D sequential bytes; where $L$ is the number of lines, $C$ is the number of columns, and $W$ is the number of colour planes. In the case of 8-bit RGB image, the 1-D sequential bytes is

$$P = [p_1 p_2 \ldots p_n] ; \quad P \in S^n ; \quad S = \{0,1,\ldots,255\}$$

Then, 1-D sequential bytes $P$ in (1) are divided into blocks of the fixed length of $T_{\text{block}} = 2^k$ bits. In the proposed cryptosystem, $k=4$ is chosen, or $T_{\text{block}} = 2$ bytes. The $b^{th}$ block contains the $i^{th}$ byte and $(i+1)^{th}$ of $P$.

$$B(b) = \left[p_{i} p_{i+1}\right]$$

$$T_{\text{block}} = 2^k \text{bytes}$$
Each byte of $B(b)$ is substituted by applying the discretized skew tent map for $rs$ times. The 8-bit discretized skew tent map is defined as below [3]

$$S^1_{ksj_i} (X) = \begin{cases} 
  \text{ceil} \left( \frac{255}{ksj_i} \times X \right) & \text{for } 0 \leq X < ksj_i \\
  \text{floor} \left( \frac{255 \times (255-X)}{255-ksj_i} \right) + 1 & \text{for } ksj_i \leq X \leq 255
\end{cases}$$

(3)

The value after $rs$ iterations is

$$Y = S^{rs}_{ksj_i} (X)$$

(4)

Where, the round $j$, $ksj_i$ is a control parameter for the round $j^{th}$ of substitution process. $X$ takes the value of $p_i$ or $p_{i+1}$. $Ks_j$ is the set of control parameters for the round $j^{th}$ and $Ks_j = [ksj_1; \ldots; ksj_i; \ldots; ksj_{rs}]$ for the substitution process. $ksj_i$ is created by Key_generator. This means that a control parameter is commonly used for every byte in a certain block.

Next, sequential bytes obtained in the substitution process are represented in the 2-D array of bits for permutation process, with the size $M \times M$, $(M = 2^{k/2})$ as

$$B(b) = \begin{bmatrix} b(0,0) & b(0,1) & \ldots & b(0,M-1) \\
   b(1,0) & b(1,1) & \ldots & b(1,M-1) \\
   \vdots & \vdots & \ddots & \vdots \\
   b(M-1,0) & b(M-1,1) & \ldots & b(M-1,M-1) \end{bmatrix}$$

(5)

where $b(m,n)$ takes binary value, i.e. ‘0’ or ‘1’; $0 \leq m, n \leq M-1$. In case of $k = 4$, the size of array is $4 \times 4$. This ensures that the standard map is bijective and one-to-one mapping. Let us define pairs $(x_i, y_i)$ and $(x_{i+1}, y_{i+1})$ be the positions of bit $b(m,n)$ at the rounds $i^{th}$ and $(i+1)^{th}$ in the permutation process, respectively.

In the permutation process, the position of bits in 2-D bit array is permuted by using the standard map [6] as given in (6), for $rp$ times.

$$F^1 \left( \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right) = \text{mod} \left( \left[ \begin{bmatrix} x_i + y_i \\ r_x + r_y \end{bmatrix} \right], \left[ \begin{bmatrix} M \\ M \end{bmatrix} \right] \right)$$

(6)

with $y = \frac{M \times x + r_x}{255}$. New position of $b(m,n)$ is obtained by

$$\left[ \begin{bmatrix} x_{i+1} \\ y_{i+1} \end{bmatrix} \right] = F^{rp} \left( \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right)$$

(7)

where $Kp_j$ is control parameters for the permutation process, $Kp_j = K; r_x; r_y$ with $K = krp_j \in \{1; 2; \ldots; 2^{N-1}\}$, $rp_j$ is secret key for the permutation round $i^{th}$ and encryption round $j^{th}$, $(1 \leq j \leq r)$. $N$ is the number of bits representing for the state variables of the map in Key_generator [5], and $N - l$ least significant bits are extracted from the output of the block of creation of control parameters as in Fig. 7 [2]. $r_x$ and $r_y$ are row and column numbers of referential position which is to avoid the steady state of chaotic map due to the corner pixels, with $r_x, r_y \in 0, 1, \ldots, M-1$. These processes are repeated for $r$ encryption rounds to achieve the avalanche effect.

### B. The decryption algorithm

The decryption is carried out with similar processes as in encryption but in the reversed order as shown Fig. 1(b). The cipher blocks are arranged in the form of bit sequence, then reshaped into square matrices with size $M \times M$ (with $M = 2^{k/2}$) for the inverse permutation process. It is noted that the standard map using modulo operation as in (6) is not an invertible function. Thus, the original block is recovered from the permuted block by using the map in the same way as in the encryption process.

$(x_i, y_i)$ is the initial position of bit $b(m,n)$ which is needed to permuted. $x_i$ and $y_i$ are input values for the standard map in the inverse permutation process. In this process, if output values at the $q^{th}$ round are equal to input ones, output values of the $(q - 1)^{th}$ round, $x'_i$ and $y'_i$, are the original position of bit $b(m,n)$.

$$\left[ \begin{bmatrix} x'_i \\ y'_i \end{bmatrix} \right] = F^{q-1} \left( \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right) \quad \text{if} \quad \left[ \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right] = F^q \left( \begin{bmatrix} x_i \\ y_i \end{bmatrix} \right)$$

(8)

Next, the bit matrix is represented into blocks of two bytes for the inverse substitution process. The inverse of $S^1_{ksj_i} (X)$ is as

$$S^{-1}_{ksj_i} (Y) = \begin{cases} 
  X_1 & \text{if } m(Y) = Y, \quad X_1 \geq \frac{255 - X_1}{255 - Ks_j}
  \\
  X_2 & \text{if } m(Y) = Y, \quad X_1 \leq \frac{255 - X_1}{255 - Ks_j}
  \\
  X_3 & \text{if } m(Y) = Y + 1
\end{cases}$$

(9)

where $X_1 = \text{floor} \left( \frac{ks_j \times Y}{255} \right)$; $X_2 = \text{ceil} \left( \frac{ks_j \times Y}{255} \right)$; $X_1 \geq \frac{255 - X_1}{255 - Ks_j}$; $X_1 \leq \frac{255 - X_1}{255 - Ks_j}$; $m(Y) = Y + 1$. After $rs$ rounds, the recovered data of substitution process is as

$$X = S^{-rs}_{ksj_i} (Y)$$

(10)

The control parameter of inverse substitution process, $ksj_i$, is synchronized with that the substitution process in every round. These inverse processes are repeated for $r$ rounds to accomplish the decryption process.

It is noted that the secret key $K_r_{r-j-1}$ for the $(r - j - 1)^{th}$ decryption round is similar to that $K_j$ for the $j^{th}$ encryption round. Here, $K_j$ and $K_{r-j-1}$ are sets of the control parameters, $K_j = \{ksj; Kp_j\}$ and $K_{r-j-1} = \{ksj_{r-j-1}; Kp_{r-j-1}\}$.

### III. Design of the Chaos-based Cryptosystem on FPGA

In this section, the above-described cryptosystem is realized on the digital logic platform of field-programmable gate arrays (FPGA). FPGAs are reprogrammable hardware chips dedicated for digital circuits, and circuits are designed by programming with the use of hardware description languages such as Verilog or VHDL [7].

The architecture of the SPN chaos-based cryptosystem on FPGA is illustrated in Fig. 2. There, the configuration of cryptosystem integrates both encryption and decryption in the design. In the design, the encryptor performs on the input data,
and the ciphertext is stored in the buffer. Then, the ciphertext in the buffer is decrypted and the recovered data is sent to the output.

This configuration of the SPN chaos-based cryptosystem is described using by VHDL and implemented on Altera DE2 Board. Key_generator produces control parameters for encryption and decryption processes, $K_j$ and $K_{r-j-1}$, respectively, as illustrated in Fig.1. Control_block sends all required signals to control the operation of Encryptor, Decryptor and Key_Generator.

A. The design of the encryption and decryption on FPGA

The design of the encryptor and decryptor are shown in Fig.3 and Fig.4, with loops for encryption/decryption round times. The encryptor has three blocks named Substitution block, Permutation block and Control block, i.e. for substitution and permutation processes and control. The skew tent map and standard map are implemented in Substitution block, Permutation block, respectively. In the decryptor, counterpart blocks are Inverse_Sub block, Inverse_Per block and Inverse_Control block.

In this design, the number of rounds for substitution, permutation and encryption processes are chosen as $rs = 4$, $rp = 1$ and $r = 7$, respectively. The simulation tools for implementation on FPGA are used to illustrate the encryption and decryption processes. There, the block of two bytes with values of $p_1 = 100$ and $p_2 = 200$ is put into the encryptor, and two bytes with values of $c_1 = 215$ and $c_2 = 226$ are obtained at the output. According, the waveforms in the encryptor and decryptor are shown in Fig.5 and Fig.6, respectively. It is clear that the time needed for encryption process is 9215 picoseconds in simulation. This amount of time in simulation is equivalent to 1843 picoseconds when it is implemented on Altera DE2 Board working at 50MHz. In the decryption process, the amount of time in simulation and implementation is 70.5 nanoseconds and 14.2 nanoseconds, respectively.

B. The design of the key generator on FPGA

Architecture of the key generator utilized the perturbation model [4]. The key generator consists in three perturbed discretized logistic maps connected in parallel as seen Fig.7. $X_1(0)$, $X_2(0)$ and $X_3(0)$ are the initial conditions of the used discretized logistic map in (11), and their values are ranged from 1 to $2^N - 1$. The discrete logistic map is defined as

$$f(X(n-1)) = \begin{cases} \text{floor} \left( \frac{X(n-1)(2^N - X(n-1))}{2^N - 1} \right) & \text{if } X(n-1) \notin \Omega \\ 2^N - 1 & \text{if } X(n-1) \in \Omega \end{cases}$$

(11)

Where $\Omega = [3 \times 2^{N-2};...;2^N]$. The perturbed sequence is given by

$$X_i(n) = \begin{cases} f(X_i(n-1)) \oplus Q_k(n) & \text{if } 0 \le j \le k_i - 1 \\ f(X_i(n-1)) & \text{if } k_i \le j \le N - 1 \end{cases}$$

(12)

where $X_i(n)$ is represented in the format of $N$ bits as $X_i(n) = x_{N-1}x_{N-2}x_{N-3}...x_0; i = 1, 2, 3; x_j \in A_b = [0, 1], N = 32$.

The perturbation is applied on the least weight $k_i$ bits of $X_i(n)$. The sample $Y_i(n)$ is used instead of $X_i(n)$ defined as

$$Y_i(n) = \text{mod} \left( X_i(n), 2^{N-1} \right)$$

(13)

where $N$ and $l$ are defined in the subsection II-A. The different control parameter $K_{s_j}$ is for different rounds in the substitution process, chosen $r = 7$, $rs = 4$, $rp = 1$ [2]; or

$$K_{s_j} = [k_{j1}; k_{j2}; k_{j3}; k_{j4}]; 1 \neq j \neq 7$$

(14)

where,

$$\begin{align*}
K_{j1} &= \text{mod} \left( A \circ B \circ C, 2^8 \right) \\
K_{j2} &= \text{mod} \left( (A \wedge B) \circ (\neg A \wedge C), 2^8 \right) \\
K_{j3} &= \text{mod} \left( (\neg A \wedge B) \circ (B \wedge C) \circ (\neg C \wedge A), 2^8 \right) \\
K_{j4} &= \text{mod} \left( A + B + C, 2^8 \right)
\end{align*}$$

$$A = Y_1(n) \wedge Y_2(n)$$

$$B = Y_1(n) \wedge Y_3(n)$$

$$C = Y_2(n) \wedge Y_3(n)$$

(15)

The control parameter for the permutation process, $K_{p_j} = \{kp_{ji}, r_x, r_y\}$, is changed for different rounds with the rule as

$$\begin{align*}
kp_{ji} &= k(n) \\
r_x &= r_1(n) \\
r_y &= r_2(n)
\end{align*}$$

(16)

where,

$$\begin{align*}
k_n &= (Y_1(n) \wedge Y_2(n)) \circ (\neg Y_1(n) \wedge Y_3(n)) \\
r_1(n) &= \text{mod} \left( Y_1(n) + Y_2(n) + Y_3(n), M \right) \\
r_2(n) &= \text{mod} \left( Y_1(n) \circ Y_2(n) \circ Y_3(n), M \right)
\end{align*}$$

(17)

“$\wedge$” is the bitwise AND operation, “$\neg$” is the bitwise complement operation, “$\circ$” is the bitwise exclusive OR (XOR). The number of the control parameters ($K_{s_j}$ and $K_{p_j}$) required for the substitution and permutation processes is dependent on the number of the iteration rounds. Linear feedback shift register (LFSR) [9] with finite cycle length of 31 bits is employed to create the sequence of disturbance $Q_{k_i}(n)$ for control parameter generation as given in Fig.7, where the primitive polynomial generator is chosen in this design with the degree of 5. The initials for Delay Flip Flops (DFFs) in LFSR is random. Then, the precision of the first, second, third
LFSR is chosen as $k_1 = 21; k_2 = 23; k_3 = 24$ as mentioned in [2]. In addition, the precision of system parameters of Logistic maps in the key generator is 32 bits. Thus, the length of secret key is 164 bits, i.e. $3 \times 32 + 21 + 23 + 24$. The generated sequence of bits from passes all the NIST’s tests [8] for the statistic. Moreover, secret keys are changed after each rounds, thus the key space is large enough to resist from the brute force attack. The secret keys consist of the permutation key $K_p$ and substitution key $K_s$, which are dependent on the precision ($N = 32$ in this design). Fig.10 shows output data of the control parameter generator with different input 28 initial values with $n = 1000$ sequences.

The parameters obtained from the software simulation with C/C++ and those from software-based FPGA realization on Altera Quartus II and ModelSim have been matched with those from the experimental result on Altera DE2 Board.

### IV. PERFORMANCE ANALYSIS OF THE CRYPTOSYSTEM

The secure SPN algorithm of chaos-based cryptosystem has been implemented on the Altera FPGA DE2 Board, Cyclone® II 2C35 FPGA. Using the same hardware platform and the
Fig. 6. Simulation waveforms of the decryption process for the plain with values $c_1 = 215$ and $c_2 = 226$

Fig. 8. Architecture of logistic map block

Fig. 9. Configuration of the perturbation model

Fig. 10. Output data of the chaotic generator, the 1000 time samples
same tools of optimization, the report of device’s resource utilization for the encryptor and decryptor is given in Table 1.

As an exemplar case, the image with the size of $L \times C \times W = 256 \times 256 \times 3$ is considered by dividing into $98304$ blocks of two bytes. The amount of time is required for encryption and decryption processes is $181.174$ microseconds and $1.386$ milliseconds, respectively. With the same image and in the case of Visual C/C++ Software on the PC with processor Core i3 working at $2.13$ Ghz and $2.0$ GB RAM memory, the amount of time for encryption and decryption processes is around $4$ seconds and $32$ seconds, respectively. It is clear that the speed of cryptosystem working on hardware at the clock $50$ Mhz is quite faster in compared with that in software on the PC. However, the hardware resource required for the cryptosystem is considerably large due to simultaneous operation in key generator and encryption/decryption processes. The plain image, cipher image and decipher as shown in Fig.11(a)-11(c) are obtained from the operation of cryptosystem on FPGA. The histogram of corresponding images is as in Fig.11(d)-11(f). It is observed that the cipher image has uniform distribution. This means that the correlation in neighbor pixels is removed to keep secure.

V. CONCLUSION

In this paper, chaos-based cryptosystem has been implemented using the uniform structure for the application of digital image security both in software (C/C++, Altera Quartus II and ModelSim) and on hardware (FPGA platform). The experimental result has showed that the implementation of SPN on the FPGA platform can be used for the application of image encryption in particular and of data encryption in general. Hardware implementation of the chaos-based cryp-

### TABLE I

<table>
<thead>
<tr>
<th>DEVICE’S RESOURCE UTILIZATION REPORT</th>
<th>Logic Elements</th>
<th>Combinational Functions</th>
<th>Dedicated Logic Registers</th>
<th>Pins</th>
</tr>
</thead>
<tbody>
<tr>
<td>Encoder uses</td>
<td>33216</td>
<td>33216</td>
<td>33216</td>
<td>475</td>
</tr>
<tr>
<td>%</td>
<td>12%</td>
<td>11%</td>
<td>4%</td>
<td>8%</td>
</tr>
<tr>
<td>Decryptor uses</td>
<td>4940</td>
<td>4649</td>
<td>1204</td>
<td>36</td>
</tr>
<tr>
<td>%</td>
<td>15%</td>
<td>14%</td>
<td>4%</td>
<td>8%</td>
</tr>
</tbody>
</table>

tosystem on hardware provides the advantage of high speed encryption, while acceptable security level is assured due to a large set of secret keys [10]. The future work is to optimize the performance and to improve security level for the model.

ACKNOWLEDGMENT

This work is supported by the Vietnam’s National Foundation for Science and Technology Development (NAFOSTED) under Grant No.102.99-2010.17.

REFERENCES


