Encryption System Based On Neural Network

Choi-Kuen Chan, Chi-Kwong Chan, Lap-Piu Lee, L.M. Cheng
Department of Electronic Engineering, City University of Hong Kong, 83 Tat Chee Avenue, Kowloon Tong, Hong Kong

Abstract A new cryptoengine based on a clipped Hopfield neural network is proposed. Results show that the proposed cryptoengine, which is executed in a parallel architecture, is suitable for the practical implementation of encryption technique using hardware with a compact size.

Keywords: Hopfield Neural Network, CryptoEngine, Encryption

1. Introduction

Neural networks have recently been attracting much interest because of their nonlinear mapping property, such as the Hopfield Neural Network (HNN) [1]. Due to the highly nonlinear properties of HNN, it is extremely favorable for cryptography. HNN can be viewed as associative memory. Memorized patterns are stored in the stable points of the network in which they are auto-recalled in minimum hamming distance (MHD). The stable points are referred to as the attractors of the network. Each attractor has a basin of attraction in which state vectors surrounding the attractor are attracted to it in MHD. The basin of attraction is called the convergent domain of the corresponding attractor. In the case a of clipped HNN (CHNN) [2,3], the number of stored patterns will be larger than that of a HNN and the system initial state will converge to one of the attractors randomly rather than in a predictable form as in a HNN [4]. This property can be used to randomize any outcomes of an event. A CHNN also has a property that for a few small changes in the neural synaptic weight matrix parameters, the distribution of system energy can be modified [5]. Thus a set of attractors with a large variation of domains of attraction can be obtained. This property of CHNN is extremely suitable for designing a new cryptographic scheme using manageable hardware.

With the increasing growth of data communication, the need for security and privacy has become a basic necessity. Cryptography is an essential requirement for communication privacy or concealment of data. The security of most cryptographic systems relies in the generation of unpredictable and irreproducible digital key streams for encryption and decryption. Here, a new cryptoengine based on the CHNN with 8 neurons is introduced. For easy implementation, the system is used to encrypt data sequence that is separated

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35413-2_36

R. Steinmetz et al. (eds.), *Communications and Multimedia Security Issues of the New Century*

into 8-bit data blocks. By forwarding secret keys and an 8-bit plaintext into the system, 8-bit ciphertext will be obtained.

1. The Clipped Hopfield Neural Network

The clipped Hopfield neural network (CHNN) used in this paper is a special version of the general HNN, in which the synaptic weight matrix of the network is clipped to three values {-1, 0, 1}. The network is used to store a set of $2N + 1$ N-tuple attractors which are denoted as $\boldsymbol{A}$, $\boldsymbol{B}$ and $\boldsymbol{C}$, where $\boldsymbol{A} = \{A_i \mid i = 0, 1,\ldots, N-1\}$, $\boldsymbol{B} = \{B_i \mid i = 0, 1,\ldots, N-1\}$ and $\boldsymbol{C} = \{1^N\}$, with $A_0 = \{1^{N/2}\ 0^{N/2}\}$, $B_0 = \{1^{N/2+1}\ 0^{N/2-1}\}$ and $A_{i+1} = \Theta(A_i)$ for $i = 0, 1,\ldots, N-1$, $B_{i+1} = \Theta(B_i)$ for $i = 0, 1,\ldots, N-1$. Here,

$$\{a^N\} = \{\underbrace{aa.....a}_{N}\} \tag{1}$$

and $\Theta(.)$ is a cyclic shift function. That is, if $x = \{x_0, x_1,\ldots, x_{N-1}\}$, $\Theta(x) = \{x_1, x_2,\ldots, x_{N-1}, x_0\}$.

Let $\xi = \{\xi_\mu \mid \mu = 1, 2,\ldots, 2N+1\}$ and $\xi = \{\boldsymbol{A}, \boldsymbol{B}, \boldsymbol{C}\}$. Let $\xi_{\mu,i}$ denotes the ith element of ξ_μ. The synaptic weight matrix T from neuron i to neuron j of the network can be formed as

$$Tji = \sigma\left(\left(\sum_{\mu=1}^{2N+1}(2\xi_{\mu,j}-1)(2\xi_{\mu,i}-1)\right)-1\right) \tag{2}$$

where

$$\sigma(x) = \begin{cases} +1 & x > 0 \\ 0 & x = 0 \\ -1 & x < 0 \end{cases} \tag{3}$$

Let S denotes the state vector of the network, the next state of each neuron S_i $(t+1)$ depends on the current states of other neurons in the following way:

$$Si(t+1) = f\left(\sum_{j=0}^{N-1} TijSj(t)\right), \quad i = 0, 1,...., N-1 \tag{4}$$

where $f(.)$ is a non-linear function and

$$f(x)=\begin{cases}1 & x\geq 0\\ 0 & x<0\end{cases} \tag{5}$$

For a network with N neurons, after searching through the N-dimension vector space, there are $2N + 1$ attractors in the network and every state vectors in the N-dimension vector space converges to one of the attractors.

Let Λ_i denote the convergent domain of attractor ξ_i. For all $i = 0,1,\ldots,2N$, we have

$$\Lambda_i \cap \Lambda_j = \phi \qquad \forall i \neq j \tag{6}$$

If a simple permutation is performed on the attractors ξ_i and the synaptic weight matrix T i.e.

$$\xi'_i = \xi_i P \qquad \text{for } i = 0,1,\ldots,2N \tag{7}$$

and

$$T' = PT\widetilde{P} \tag{8}$$

where P is a permutation matrix and $\widetilde{P}$ is the transpose of P. The network formed with T' and ξ'_i has the same properties as that formed with T and ξ_i, while $\Lambda'_i \neq \Lambda_i$, for $i = 0,1,\ldots,2N$. As a result, for the same input imposed on the network, it will converge to different attractor for different P. By changing P, different sequences are obtained.

2. The Output Mapping Function

The output mapping function (OMF) defined here is used to map the 8-bit output from the CHNN to 2-bit sequence. After passing through the CHNN, every inputs imposed on the network will eventually converge to one of the $2N$+1 attractors. We can divide the set of attractors $\boldsymbol{A}$ into four groups, Ψ_{A0}, Ψ_{A1}, Ψ_{A2} and Ψ_{A3}, and $\boldsymbol{B}$ into Ψ_{B0}, Ψ_{B1}, Ψ_{B2} and Ψ_{B3}, each has equal number of attractors. Let x denotes the input to the OMF (output from the CHNN) and y denotes the output from the OMF, we have

$$y=\begin{cases}00 & \textit{if } x\in\Psi_{A0} \textit{ or } \Psi_{B0}\\ 01 & \textit{if } x\in\Psi_{A1} \textit{ or } \Psi_{B1}\\ 10 & \textit{if } x\in\Psi_{A2} \textit{ or } \Psi_{B2}\\ 11 & \textit{if } x\in\Psi_{A3} \textit{ or } \Psi_{B3}\end{cases} \tag{9}$$

If $x = C$, 00, 01, 10, or 11 will be output alternately. That is, the first time $x = C$, 00 will be output; the second time $x = C$, 01 will be output; the third time $x = C$, 10 will be output; the forth time $x = C$, 11 will be output; this then repeats. As a result, a balanced output sequence is obtained.

3. Encryption with the Proposed Cryptoengine

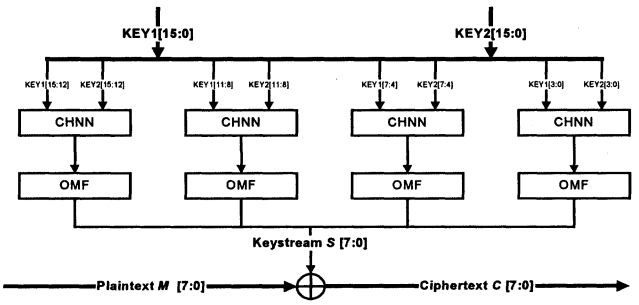
Figure 1 shows the basic structure of the proposed cryptoengine. In this figure, KEY1 and KEY2 are two 16-bit secret keys (or two subkeys of a 32-bit key) which are generated from any secure random number generator. The two 16-bit secret keys are used to generate an 8-bit keystream, denoted by S. The encryption is achieved by forwarding an 8-bit plaintext M, which is then xored with S. Finally an 8-bit ciphertext C is obtained, i.e.

$$C = S \oplus M \tag{10}$$

To achieve a decryption, the 8-bit ciphertext and the same keys are presented in the proposed cryptoengine in the same way, i.e.

$$M = S \oplus C \tag{11}$$

From equations (7) and (8), with the change of the permutation matrix P, the synaptic weight matrix and the set of attractors in the CHNN are different. This gives rise to a different output from the CHNN. Thus after passing through the OMF, a different keystream S is obtained, even though the same keys are used.



X[z, y] denotes the bit position from y to z of label X.

Figure 1. Structure of the proposed cryptoengine.

4. Hardware Realization

The proposed cryptoengine is synthesised using the Synopsys VHDL integrated simulator and implemented in a Xilinx FPGA chip. The synthesis results of the cryptoengine modules are tabulated in Table 1. The performance of the proposed cryptoengine is compared with the Data Encryption Standard (DES) [6]. From the synthesis results of the DES modules in [7], the total number of configurable logic blocks (CLB) used in DES is 2704, while in the proposed cryptoengine is (16 x 4 + 1 x 4 + 4) = 72. Since the system can be executed in parallel, the total time for the proposed cryptoengine to encrypt one 8-bit data is (11.22 + 3.3 + 3.3) = 17.82ns, and the total time required to encrypt a 64-bit data is (17.82 x 8) = 142.56ns. From [7], the total time required to encrypt a 64-bit data in DES is 318.98ns. The comparisons show that size of the proposed cryptoengine is 97% smaller than DES, while the speed of the proposed cryptoengine for encrypting same bit of data as in DES is 55% less. Therefore, the proposed cryptoengine is more suitable for the implementation of encryption when the hardware size and encryption time is crucial.

Table 1. Synthesis results of the proposed cryptoengine modules.

Module	CLB required	*Timing (ns)
CHNN	16	11.22
XOR	4	3.3
OMF	1	3.3

** Timing is measured under the Xilinx Xfpga_4000e-3 library parameters: path_full,delay_max, max_paths, and WCCOM operation conditions.*

Because of its compact size and fast operation, the proposed cryptoengine is suitable for efficient implementation in a smart card, with two secret keys each of size 16-bit. The keys can be master key of the smart card and ID of the smart card.

5. Security

In the proposed cryptoengine, with the nonlinear dynamic property of the Hopfield neural network, the following properties will occur under different keystreams: different ciphertext may be generated from the same plaintext; same ciphertext may be generated from different plaintext; different plaintext may give rise to different ciphertext. Thus, the scheme is guarded against cryptanalyst's ciphertext only attack. Moreover, neither a chosen plaintext attack nor a known plaintext attack [8] can be used to find the secret keys.

In the proposed design, four CHNN each with 8 neurons ($N = 8$) are cascaded together in parallel which is same as a single neural network with size $N = 32$. This arrangement will not scarify the security as $N = 32$ but on the other hand the efficiency is enhanced by a basic network of $N = 8$, since CHNN can be implemented and executed in a parallel architecture.

6. Conclusions

In this paper, we have presented a new construction of a cryptoengine that is based on the nonlinearly dynamic property of a clipped Hopfield neural network. The proposed cryptoengine can be used to encrypt 8-bit data each time. By cascading more CHNN in parallel, data other than 8-bit can also be encrypted easily. Since the proposed cryptoengine is simple in architecture in which only simple exclusive or functions and neural synaptic weight matrixes are used, it is suitable to be implemented with VLSI technology.

References

[1] J.J.Hopfield, Neural Networks and Physical Systems with Emergent Collective Computational Abilities, Proc. Natl. Acad. Sci. USA1982; 79: 2554-2558.

[2] Donghui Guo, Zhengxiang Chen, Ruitang Liu, Boxi Wu, A modified Hopfield Model of Neural Network, Journal of Xiamen University (Natural) 1993; 32(1): 33-40.

[3] Chi-Kwong Chan, L.M.Cheng, A Configurable Nonlinear Filter Generator, Electronic Letters 1998; 34(4).

[4] R.J.McEliece, E.C.Posner, E.R.Rodemich, S.S.Vankatesh, The Capacity of the Hopfield Associative Memory, IEEE Trans. Inform. Theory 1987; IT-33(4): 461-482.

[5] D. Guo, L. M. Cheng & L. L. Cheng, A New Symmetric Probabilistic Encryption Scheme Based on Chaotic Attractors of Neural networks , Applied Intelligence 1999; 10(1).

[6] Federal Information Processing Standard (FIPS) 46, Data Encryption Standard, National Bureau of Standards 1977.

[7] W. P. Choi & L. M. Cheng, Modelling the Crypto-processor from design to synthesis, Lecture notes in Computer Science, Springer 1999; 1717: 25-36.

[8] A.J.Menezes, P.C.Oorschot & S.A.Vanstone, *Handbook of Applied Cryptography*, CRC Press, New York, 1997.