Neural Cryptography for Secure Voice Communication using Custom Instructions


Department of Information Engineering and Computer Science
Feng Chia University,
No. 100 Wenhwa Rd., Seatwen, Taichung, Taiwan

Abstract - Cryptography of resource constrained devices represents a very active area of cryptographic research. Custom instructions have been widely used to achieve the conflicting demands between performance and flexibility. This paper proposes a neural cryptography implementation for secure voice communication using custom instructions to achieve real-time performance on very low resource devices. The experiments show that using only very limited hardware to implement the CIs, the 40 speed-up can be saved to speed-up the performance.

Keywords: Neural Cryptography; Custom Instruction (CI)

1 Introduction

The importance of cryptography on resource constrained devices is related to the current trend of pervasive/ubiquitous computing, which means an ever increasing demand for computing capabilities in diverse, wireless and low-resource scenarios, in both civilian and military applications, including mobile phones, smart cards, toll collection, animal and cargo tracking and electronic passports, and etc. Due to the low-resource environments, cryptographic algorithms are typically hardware-oriented, and designed to be particularly compact and efficient. The balance between security, high performance (in hardware), and low overall cost (throughput, power consumption, area, price) in low-resource environments represents a major challenge in cryptographic acceleration.

The learning and classifying abilities of neural networks can be used for different aspects of cryptography such as to learn the inverse-function of any cryptographic algorithm in cryptanalysis or to solve the key distribution problem in public-key cryptography using neural network mutual synchronization. The classical key exchange problem in cryptography are mainly based on algebraic number theory, but the synchronization phenomenon of interacting two neural nets provides a new idea to solve this problem. Good reviews of neural cryptography can be found in [1, 2], for analysis of neural learning rule and protocol dynamics. Mislovaty et al. reported that ANN was secure against brute-force attacks [3].

The custom instruction set extensions have been highly successful such as Intel MMX and SSE, AMD 3DNow! and DSP instructions for digital signal processors. A partially customizable instruction-set which can be tuned towards the specific requirements of applications by extending the basic instruction set with dedicated custom instructions within custom functional units (CFUs). By using a base processor, the design process can focus on the CIs only, that significantly reduces verification efforts and hence shortens the design cycle by sharing development tools such as compilers, debuggers, simulators. Commercial examples are Tensilica Xtensa [4], Xilinx MicroBlaze [5], and Altera Nios II [6]. The security support has been included in commercial embedded processors such as ARM SecurCore [7], STMicroelectronics SmartJ [8], and Atmel XMEGA [9]. However, most commercial cryptographic instruction extensions did not release the detail implementation to the public.

Various approaches to optimize or accelerate feedforward neural nets for embedded systems [10, 11, 12]. Santos [13] proposed a custom instruction to approximate the value of tanh() through the use of a range addressable lookup table for the acceleration of a pre-trained feedforward artificial neural network executing on a NIOS II processor.

The proposed neural cryptography for secure voice communication is to derive a set of synaptic weights as a pair of keys through the neural learning mechanism to achieve the data encryption and decryption between two private neural nets. The suitable "neural crypto instructions" are designed and implemented as CIs of Altera Nios II/e, to achieve a secure voice communication in real-time. The neural crypto CIs may retain the flexibility of original embedded processors to shorten design cost and time but increase performance as well as lower energy consumption and hardware cost.

This research was supported by MOST grant 103-2221-E-035 -053.
2 Methodology

In this paper, we combine approaches in [14], [15], [16] and [17] to accelerate the neural cryptography computation for secure voice communication on very low resource devices.

2.1 Neural Cryptography for Secure Voice Communication

A typical neural cryptography application for voice communication is depicted in Fig. 1. The voice is first preprocessed, followed by a 512-point short-time Fourier transform (STFT) of $\text{voice}(t)$ with a 20-ms Hamming window to obtain $P(n, d)$, where $n$ is the frequency bin sample index, $n = 1, \ldots, 256$, and $d$ is the frame index, $d = 1, \ldots, D$. The $P(n, d)$ will be the plain text as the input of the neural cryptography encryption. The $C(n, d)$ will be the cipher text of the output and can be sent to the unsecure public communication channels. Once the $C(n, d)$ is received, it can be decrypted to the plain text, $P'(n, d)$, by the neural cryptography decryption and converted to $\text{voice}'(t)$ by inverse short-time Fourier transform (ISTFT).

![Fig. 1. A typical neural cryptography application for voice communication.](image)

A four stage multilayer feedforward neural network will be used for neural cryptograph encryption and decryption as shown in Fig. 2.

![Fig. 2. A four stage multilayer feedforward neural network for neural cryptograph.](image)

The first two stages are consisted of one non-linear function layer, $\tanh()$, and one linear function layer to perform the neural cryptography encryption. The last two stages are still consisted of the same neural net structure to perform the neural cryptography decryption. The random and unpredictable initial values are used by the backpropagation learning rule to train the four stage multilayer feedforward neural network using the training set that the desired target patterns are equivalent to the original input patterns, $P(n, d)$. Once the required minimum mean-square error (MSE) between neural net outputs, $P'(n, d)$, and the desired targets, $P(n, d)$, is achieved, the synaptic weights of the first and last two stages will be the encryption key and the decryption key correspondingly.

2.2 Custom Instruction

Fig. 3 shows the block diagram of a typical custom instruction processor. The interface between the base processor and the custom functional unit (CFU) only includes the control signals for the CI encoding and the synchronization of multi-cycle custom instructions. The input and output bandwidths of the data transfer buses are limited by the number of read ports and write ports of the general purpose register file (GPRF) of the base processor. This simple interface keeps the base processor data-path unchanged to simplify the CI implementation and to reduce the design and verification cost.

![Fig. 3. A typical custom instruction processor.](image)

Typical multiply-accumulators (MACs) are implemented as CIs to achieve parallel computation between the synaptic weights and neuron output signals. The property of non-linear hyperbolic tangent sigmoid function, $\tanh()$, is implemented as a single CI to approximate the value of $\tanh()$ through the use of a hybrid range addressable lookup table to store the mapping data from C code precision simulation according the [15].

Several data movement CIs are implemented to explicitly move additional input and output operands between the base processor and the state registers in CFU so that the
performance of CIs will not be limited by the available data bandwidth between the base processor and CFU. The CI scheduling and state register assignment can be optimized according to sequential ordering of data for better use of the CIs to achieve a faster execution time.

3 Results

Firstly, we use the MATLAB to determine the topology parameters of feedforward neural networks and conduct the cryptography feasibility analysis. Secondly, we use C++ programs to normalize the MATLAB double floating point results to the correct range of the fixed point values as shown in Fig. 4. The appropriate bit-precision of fixed point is then selected for hardware implementation. Finally, we implement the proposed neural cryptography on an embedded processor with CIs to speed up the execution of secure voice communication on very low resource embedded processors.

![Conversion from MATLAB results to C++ for the bit-precision analysis.](image)

3.1 MATLAB Results

Fig. 5 depicts the original voice(t) as well as the results of STFT, abs(P(n, d)) and angle(P(n, d)), respectively.

![The original voice(t), abs(P(n, d)), and angle(P(n, d)).](image)

The Equ. (1) is defined as the mean-square error (MSE) of the four stage multilayer feedforward neural network to measure the quality of learning results after the 1000 iterations of the back-propagation weight update equations. Usually the smaller MSE obtains the better quality of neural network representations.

$$\text{MSE} \triangleq \sum_{n=1}^{N} \sum_{d=1}^{D} \frac{(P(n,d) - C(n,d))^2}{N \times D} \quad (1)$$

The total entropy is defined in Equ. (2) to measure the discrepancy between the plain text and the cipher text. The larger total entropy is the better effectiveness of the encrypted voice.

$$\text{Total Entropy} \triangleq \sum_{n=1}^{N} \sum_{d=1}^{D} \frac{(P(n,d) - C(n,d))^2}{N \times D} \quad (2)$$

Fig. 6 depicts the encrypted signals of voice(t) and the encrypted cipher texts of Ca(n, d) and Cb(n, d), respectively. Fig. 7 shows the decrypted voice'(t) as well as the decrypted plain texts of abs(P(n, d)) and angle(P(n, d)), respectively.

![The encrypted signals of voice(t) and the encrypted cipher texts of Ca(n, d) and Cb(n, d).](image)

Fig. 7. The original voice(t), abs(P(n, d)), and angle(P(n, d)).
The Signal-to-Noise Ratio (SNR) of the received voice is defined in Eqn. (3) to evaluate the quality between the original voice and the decrypted voice. The larger SNR is the better quality of the decrypted voice.

\[
\text{SNR}_{\text{time}} = 20 \log_{10} \left( \frac{SNR_{\text{voice}(i)} + SNR_{\text{voice}(i)} + \cdots + SNR_{\text{voice}(i)}}{N} \right) \tag{3}
\]

The MATLAB results are shown in Table 1–3 which are used to conduct the feasibility analysis of neural cryptography for secure voice communication and to determine the neural network topology parameters according to the number of hidden neurons and the number of bits for data type representations. Table 1 shows the MSE results of our proposed neural network after the 1000 iterations of the back-propagation learning rules to train 400 randomly generated data. Table 2 shows the total entropy and Table 3 shows the SNR for various numbers of hidden neurons with respect to different data types. The larger number of hidden neurons and the larger number of bits will result in the larger hardware cost and the longer execution time.

### Table 1. The MSE of our proposed neural net after learning.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Number of Neurons</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>96</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-precision Floating point</td>
<td>4.72E-04</td>
<td>8.02E-04</td>
<td>8.57E-04</td>
<td>1.49E-03</td>
<td>1.31E-03</td>
<td>5.20E-03</td>
<td></td>
</tr>
<tr>
<td>32 Bits Signed Integer</td>
<td>6.68E-04</td>
<td>1.59E-04</td>
<td>1.64E-04</td>
<td>1.91E-04</td>
<td>2.24E-04</td>
<td>2.95E-04</td>
<td></td>
</tr>
<tr>
<td>16 Bits Signed Integer</td>
<td>1.15E-03</td>
<td>3.03E-03</td>
<td>3.20E-03</td>
<td>4.89E-03</td>
<td>3.83E-03</td>
<td>6.26E-03</td>
<td></td>
</tr>
<tr>
<td>8 Bits Signed Integer</td>
<td>1.21E-01</td>
<td>9.40E-02</td>
<td>2.92E-01</td>
<td>1.64E-01</td>
<td>1.64E-01</td>
<td>6.64E-01</td>
<td></td>
</tr>
</tbody>
</table>

### Table 2. The total entropy between the plain and cipher texts.

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Number of Neurons</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>96</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-precision Floating point</td>
<td>8.43E-01</td>
<td>1.28E+00</td>
<td>1.17E+01</td>
<td>6.53E+01</td>
<td>4.14E+01</td>
<td>4.58E+02</td>
<td></td>
</tr>
<tr>
<td>32 Bits Signed Integer</td>
<td>8.40E-01</td>
<td>1.28E+00</td>
<td>1.17E+01</td>
<td>6.52E+01</td>
<td>4.13E+01</td>
<td>4.58E+02</td>
<td></td>
</tr>
<tr>
<td>16 Bits Signed Integer</td>
<td>7.27E-01</td>
<td>1.99E+00</td>
<td>8.92E+00</td>
<td>6.05E+01</td>
<td>2.07E+02</td>
<td>1.99E+02</td>
<td></td>
</tr>
<tr>
<td>8 Bits Signed Integer</td>
<td>5.04E-01</td>
<td>3.15E+00</td>
<td>9.95E+00</td>
<td>6.44E+01</td>
<td>1.46E+01</td>
<td>1.44E+01</td>
<td></td>
</tr>
</tbody>
</table>

### Table 3. The SNRs of received voice from decrypted data.

#### 3.2 Custom Instruction Results

We choose the Altera Nios II/e as the very low resource target base processor for the DE2-70 board run at 100Mhz. Thus, the number of input and output ports for general purpose register file is 2 and 1. The software cycle count of a primitive instruction is estimated by the cycle count in the execution stage of the Altera Nios II/e. The hardware cycle count of a CI is estimated by synthesizing the corresponding template using Altera Quartus II. The cycle count of data transfers between the Nios II/e and the SR of CFU is single cycle latency.

The matrix multiplications and additions are performed by MAC CIs and data movement CIs. The CI block diagrams of the 32-bit implementation is depicted in Fig. 8. The 16-bit and 8-bit data types are shown in Fig. 9. Fig. 10 shows the simulation results of the 8-bit \( \tanh() \).

### Table 4

<table>
<thead>
<tr>
<th>Data Type</th>
<th>Number of Neurons</th>
<th>8</th>
<th>16</th>
<th>32</th>
<th>64</th>
<th>96</th>
<th>128</th>
</tr>
</thead>
<tbody>
<tr>
<td>Double-precision Floating point</td>
<td>1.44E-01</td>
<td>1.51E+01</td>
<td>1.41E+01</td>
<td>1.01E+01</td>
<td>1.91E+01</td>
<td>9.31E+00</td>
<td></td>
</tr>
<tr>
<td>32 Bits Signed Integer</td>
<td>1.45E-01</td>
<td>1.52E+01</td>
<td>1.42E+01</td>
<td>1.01E+01</td>
<td>1.91E+01</td>
<td>9.55E+00</td>
<td></td>
</tr>
<tr>
<td>16 Bits Signed Integer</td>
<td>9.61E-01</td>
<td>6.55E+00</td>
<td>6.15E+00</td>
<td>3.57E+00</td>
<td>3.14E+00</td>
<td>6.77E+01</td>
<td></td>
</tr>
<tr>
<td>8 Bits Signed Integer</td>
<td>3.01E-01</td>
<td>8.58E-02</td>
<td>1.19E-01</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td>0.00E+00</td>
<td></td>
</tr>
</tbody>
</table>

### 3.3 Performance analysis and discussion

Basically for the better sound quality and the more security of the voice communication, we need the more number of hidden neurons for the neural net and the more number of bits for the arithmetic computations. The larger numbers of neurons or bits will result in the more hardware cost and the execution time. The desired sound quality can be chosen using the SNR of received voice from Table 2 as
well as the desired security can be chosen using the total entropy between the plain and cipher texts from Table 3. However, the numbers of neurons or bits is not a linear relationship with the quality and security.

From these experiments, we may suggest using the number of bits to determine the desired sound quality and using minimum number of neurons to decide the security. Because of a lot of simple and low cost perturbation algorithms for encryption and decryption can enhance the desired security. Once the numbers of neurons and bits are decided, the hardware cost of CI implementations will be determined by using the Table 4 according to the overall computation overheads of real-time applications, usually the number of STFT points within a fixed latency Hamming window.

<table>
<thead>
<tr>
<th>Computational Custom Instructions</th>
<th># of CIs</th>
<th>Total # of LEs</th>
<th>Total cycles</th>
<th>Total reduced cycles</th>
</tr>
</thead>
<tbody>
<tr>
<td>None(16-bit)</td>
<td>0</td>
<td>700</td>
<td>53720416</td>
<td>0</td>
</tr>
<tr>
<td>Four 16-bit MACs</td>
<td>2</td>
<td>865 +700+ 66(SR)+ 99(MAC)</td>
<td>52393536</td>
<td>1326880</td>
</tr>
<tr>
<td>Four 16-bit MACs Two 16-bit tanh()</td>
<td>3</td>
<td>19331 +700+ 66(SR)+ 99(MAC)+ 18466(tanh)</td>
<td>1327216</td>
<td>52393200</td>
</tr>
<tr>
<td>None(8-bit)</td>
<td>0</td>
<td>700</td>
<td>53720416</td>
<td>0</td>
</tr>
<tr>
<td>Eight 8-bit MACs</td>
<td>2</td>
<td>865 +700+ 66(SR)+ 99(MAC)</td>
<td>52393488</td>
<td>1326928</td>
</tr>
<tr>
<td>Eight 8-bit MACs Four 8-bit tanh()</td>
<td>3</td>
<td>897 +700+ 66(SR)+ 99(MAC)+ 32(tanh)</td>
<td>1327016</td>
<td>52393400</td>
</tr>
</tbody>
</table>

Table 4. The total numbers of execution cycles and logic elements (LEs) for various CI implementations for neural cryptography using a four stage multilayer feedforward neural network with 32 neurons.

4 Conclusions

This paper presented an approach to implement neural cryptography for secure voice communication in real-time. Our approach took benefits from both software and hardware to make flexible and scalable neural computation available on resource limited embedded processors with good performance for the neural cryptography. The experiments show that using only very limited hardware to implement the CIs, the 300 speed-up can be saved to speed-up the performance. The future work of this paper will be the continuous alternation between theoretical investigation and practical implementation of neural CIs to simultaneously achieve the neural crypto, learning, classifying abilities of various application programs running on the same embedded system.

5 References