Information Sciences 291 (2015) 172–183

Contents lists available at ScienceDirect

Information Sciences

journal homepage: www.elsevier.com/locate/ins

Fast Fourier transform using matrix decomposition

Yicong Zhou [a,*], Weijia Cao [a], Licheng Liu [a], Sos Agaian [b], C.L. Philip Chen [a]

[a] Department of Computer and Information Science, University of Macau, Macau 999078, China
[b] Department of Electrical and Computer Engineering, University of Texas at San Antonio, San Antonio, TX 78249, USA

ARTICLE INFO

Article history:
Received 17 July 2013
Received in revised form 10 June 2014
Accepted 5 August 2014
Available online 4 September 2014

Keywords:
Fast Fourier transform
Orthogonal transform
Sparse matrix
Image encryption

ABSTRACT

To reduce both the multiplicative complexity and total number of operations, this paper introduces a modeling scheme of the fast Fourier transform (FFT) to decompose the discrete Fourier transform (DFT) matrix recursively into a set of sparse matrices. Integrating three orthogonal transforms, the Hadamard, Modified Haar and Hybrid transforms, the proposed scheme is able to obtain different FFT representations with less computation operations than state of the arts. To investigate the applications of the proposed FFT scheme, a multi-stage image encryption algorithm is also introduced. Experimental results and security analysis are provided to show its encryption performance.

1. Introduction

As one of the most frequently used operations in digital signal processing, the discrete Fourier transform (DFT) has been widely employed in various fields [11,12,30] such as optical systems [26,29], medical research [8], and image processing [20,28]. However, directly calculating an N-point DFT requires N^2 complex multiplications and $N(N-1)$ complex additions. This extremely slows down the speed of digital signal processing, especially real-time signal processing.

To reduce the computation complexity, various fast Fourier transform (FFT) algorithms have been developed [1,5,11,13,14]. A split-radix-2/8 FFT algorithm [11,22] was proposed to recursively factor a length-N DFT into one length-$\frac{N}{2}$ DFT and four length-$\frac{N}{8}$ DFTs. Including the DFT properties of periodicity and symmetry, several improved algorithms have been also developed such as the recursive FFT [24], fused FFT [21], radix-$2/2^s$ $(4 \leqslant s \leqslant m)$ FFT [6], decimation in frequency (DIF) and time (DIT) pruning scheme [17], and mixed-radix FFT [25]. They are effective to reduce the computation complexity of DFT. For example, the recursive FFT scheme was reported to have less computation complexity than the conventional algorithms for computing Fourier-like transforms [24]. In addition, the fused FFT is 15% faster than traditional implementations [21]. Different from these algorithms, this paper proposes a modeling FFT scheme to decompose DFT into a number of sparse matrices. Using different orthogonal transforms, the proposed scheme can obtain various FFT representations. We select the Hadamard, modified Haar, and Hybrid (Hadamard–Haar) transforms as examples to show its effectiveness. The proposed scheme significantly reduces the computation complexity and shows better performance than several state-of-the-art FFT methods.

In addition to low computation complexity, the proposed scheme also shows benefits in data security because it is able to protect data with multiple security levels. As an example, this paper introduces a multi-stage image encryption algorithm (MSIEA) using the proposed FFT scheme. Unlike many image encryption algorithms that protect images using parametric

* Corresponding author. Tel.: +86 853 83978458; fax: +86 853 28838314.
E-mail address: yicongzhou@umac.mo (Y. Zhou).

http://dx.doi.org/10.1016/j.ins.2014.08.022
0020-0255/

DFTs, such as the discrete fractional Fourier transform (DFrFT) [15,16,23] and phase-truncated Fourier transform [19], the proposed MSIEA integrates the image encryption processes with the FFT decomposition. Using different permutation matrices allows MSIEA to encrypt images in different security levels. Experimental results and security analysis are provided.

The rest of this paper is organized as follows. Section 2 introduces the FFT scheme and three examples using orthogonal transforms. Section 3 proposes the multi-stage image encryption algorithm. Its simulation results are provided in Section 4 and its security issues are analyzed in Section 5. Finally, Section 6 reaches a conclusion.

2. Proposed FFT scheme

For an N-point input data sequence $X = (x_0, x_1, \ldots, x_{N-1})^T$, suppose data vector $Y = (y_0, y_1, \ldots, y_{N-1})^T$ is the result of its discrete Fourier transform (DFT). The matrix format of N-point DFT and its inverse transform are defined as

$$Y = F_N X \quad \text{and} \quad X = \frac{1}{N} F_N^{-1} Y \tag{1}$$

where F_N^{-1} is an inverse matrix of the DFT matrix F_N defined by

$$F_N = \begin{pmatrix} W_N^{0\cdot 0} & W_N^{0\cdot 1} & \cdots & W_N^{0\cdot(N-1)} \\ W_N^{1\cdot 0} & W_N^{1\cdot 1} & \cdots & W_N^{1\cdot(N-1)} \\ \vdots & \vdots & \ddots & \vdots \\ W_N^{(N-1)\cdot 0} & W_N^{(N-1)\cdot 1} & \cdots & W_N^{(N-1)\cdot(N-1)} \end{pmatrix} \tag{2}$$

where $W_N^{n\cdot k} = e^{-j\frac{2\pi nk}{N}}$ with $n, k \in (0, 1, 2, \ldots, N-1)$ is so-called the twiddle factor.

Here, we introduce an FFT scheme to decompose the DFT matrix F_N into a number of sparse matrices. Its structure is illustrated in Fig. 1.

For an *N-point* ($N = 2^n$) DFT matrix, the general formula of the proposed FFT scheme is defined by

$$F_N = P_N^2 \left(I_{\frac{N}{2^{n-1}}} \oplus F^{sr}_{\frac{N}{2^{n-1}}} \oplus F^{sr}_{\frac{N}{2^{n-2}}} \oplus \cdots \oplus F^{sr}_{\frac{N}{2}} \right) P_N^1 D_N O_N \tag{3}$$

where F_N^{sr} denotes F_N or its deformation (such as being applied with permutation or scaled by factors), P_N is a permutation matrix (including the identity matrix), D_N is a diagonal matrix (including the identity matrix), O_N is the orthogonal matrix and $\oplus$ denotes the direct matrix sum defined in Eq. (4) where $A \in C^{N_1 \times N_2}$ and $B \in C^{N_3 \times N_4}$ are two matrices,

$$A \oplus B = \begin{pmatrix} A & 0 \\ 0 & B \end{pmatrix} \tag{4}$$

Utilizing orthogonal transforms with appropriate decompositions, the N-point DFT can be iteratively divided into small DFTs with or without a few number of twiddle factors. In this manner, the proposed FFT scheme significantly reduces the computation complexity. Applying different orthogonal transform matrices to O_N in Eq. (3) yields new FFT representations. Next, we will provide three examples to show the effectiveness of the proposed scheme.

2.1. Hadamard transform based FFT representation (HDT-FFT)

Using the Hadamard transform [1,2] O_N in Eq. (3) can be defined as:

$$O_N = \prod_{i=1}^{\log N} \left(I_{\frac{N}{2^i}} \otimes H_2 \otimes I_{2^{i-1}} \right) \tag{5}$$

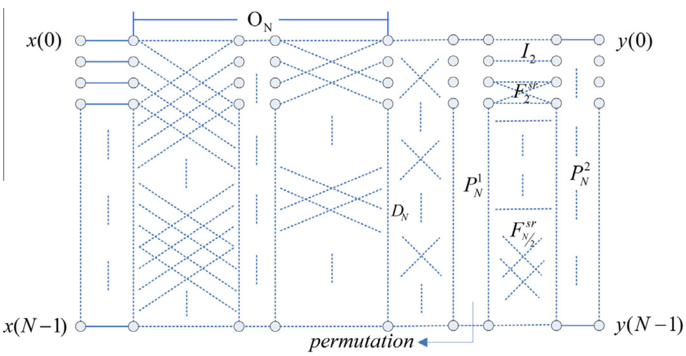

Fig. 1. The structure of the proposed N-point FFT scheme.

where H_2 is the 2×2 Hadamard transform matrix defined in Eq. (9), and I_k denotes a $k \times k$ identity matrix. $\otimes$ denotes the Kronecker product of two matrices $A \in C^{N_1 \times N_2}$ and $B \in C^{N_3 \times N_4}$,

$$A \otimes B = \left(a_{i_1,i_2}B\right)_{1 \leqslant i_1 \leqslant N_1; 1 \leqslant i_2 \leqslant N_2} \tag{6}$$

When $N = 8$, using the Hadamard matrix, Eq. (3) can be written as

$$F_8 = P_8^2(I_2 \oplus F_2^{sr} \oplus F_4^{sr})H_8 = P_8^2\left(S_8^1 D_8^1 S_8^2 P_8^1 S_8^1\right)H_8 \tag{7}$$

where sparse matrices $S_8^1 = I_2 \oplus H_2 \oplus H_4, S_8^2 = I_6 \oplus H_2$, diagonal matrix D_8^1 is defined by

$$D_8^1 = diag\left\{1, 1, 2^{-1}, -2^{-1}j, 2^{-2}, -2^{-2}j, -\frac{\sqrt{2}}{8}j, \frac{\sqrt{2}}{8}\right\} \tag{8}$$

and H_N is the Hadamard transform matrix [4,12] with the following recursive structure

$$H_{2^{k+1}} = \begin{pmatrix} H_{2^k} & H_{2^k} \\ H_{2^k} & -H_{2^k} \end{pmatrix}, k = 0, 1, \ldots \text{ and } H_1 = 1 \tag{9}$$

This iterative structure can be represented by the matrix form of fast Hadamard transform,

$$H_{2^{k+1}} = \begin{pmatrix} H_{2^k} & 0 \\ 0 & H_{2^k} \end{pmatrix}\begin{pmatrix} I_{2^k} & I_{2^k} \\ I_{2^k} & -I_{2^k} \end{pmatrix} \tag{10}$$

Using this structure recursively, one can obtain the following equation,

$$\begin{aligned} H_8 &= \left(\begin{array}{c:c} H_4 & 0 \\ \hdashline 0 & H_4 \end{array}\right)\left(\begin{array}{c:c} I_4 & I_4 \\ \hdashline I_4 & -I_4 \end{array}\right) = \left(\begin{array}{cc:cc} H_2 & 0 & & \\ 0 & H_2 & \multicolumn{2}{c}{0} \\ \hdashline & & H_2 & 0 \\ \multicolumn{2}{c:}{0} & 0 & H_2 \end{array}\right)\left(\begin{array}{cc:cc} I_2 & I_2 & & \\ I_2 & -I_2 & \multicolumn{2}{c}{0} \\ \hdashline & & I_2 & I_2 \\ \multicolumn{2}{c:}{0} & I_2 & -I_2 \end{array}\right)\left(\begin{array}{c:c} I_4 & I_4 \\ \hdashline I_4 & -I_4 \end{array}\right) \\ &= (I_4 \otimes H_2)(I_2 \otimes H_2 \otimes I_2)(H_2 \otimes I_4) = \prod_{i=1}^{3}\left(I_{\frac{8}{2^i}} \otimes H_2 \otimes I_{2^{i-1}}\right) \end{aligned} \tag{11}$$

P_8^1 and P_8^2 are two permutation matrices defined by,

$$P_8^1 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad P_8^2 = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \tag{12}$$

The permutation matrix can be represented by its permutation function. For example, the above two permutation matrices can be rewritten as,

$$P_8^1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 2 & 3 & 4 & 5 & 7 & 6 & 8 \end{pmatrix} \quad P_8^2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 5 & 3 & 7 & 2 & 6 & 4 & 8 \end{pmatrix} \tag{13}$$

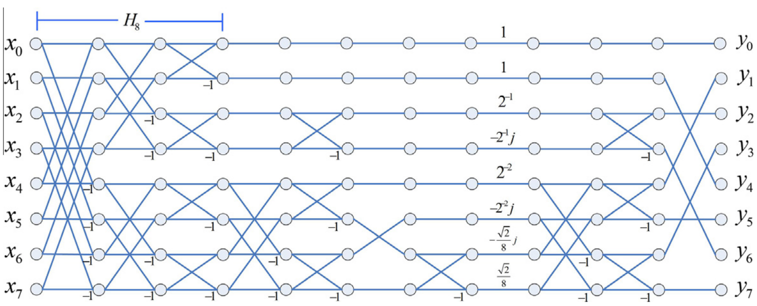

Fig. 2. The flow diagram of the 8-point HDT-FFT.

where the first row shows row indexes in the permutation matrix and the second row denotes the positions of 1s in the corresponding rows. Fig. 2 shows the flow diagram of an 8-point HDT-FFT.

2.2. Modified Haar transform based FFT representation (MHT-FFT)

When O_N in Eq. (3) is selected to be the modified Haar transform, we obtain another FFT representation, called the modified Haar transform based FFT representation (MHT-FFT). The modified Haar transform is a modification of the un-normalized Haar transform defined by a recursive structure [1,3],

$$HR_{2^{k+1}} = \begin{pmatrix} HR_{2^k} \otimes [1,1] \\ I_{2^k} \otimes [1,-1] \end{pmatrix}, \quad HR_1 = 1 \tag{14}$$

Slightly changing the structure of the un-normalized Haar transform in Eq. (14), we present the modified Haar transform as follows,

$$M_{2^{k+1}} = \begin{pmatrix} [1,1] \otimes M_{2^k} \\ [-1,1] \otimes I_{2^k} \end{pmatrix}, \quad M_1 = 1 \tag{15}$$

This iterative structure can be represented by a form of matrix product,

$$M_{2^{k+1}} = \begin{pmatrix} M_{2^k} & 0 \\ 0 & I_{2^k} \end{pmatrix} \begin{pmatrix} I_{2^k} & I_{2^k} \\ -I_{2^k} & I_{2^k} \end{pmatrix} \tag{16}$$

In MHT-FFT, O_N in Eq. (3) is replaced by the modified Haar transform matrix M_N in Eq. (16). For instance, an 8-point MHT-FFT can be generated from Eq. (3) as:

$$F_8 = P_8^2 (I_2 \oplus F_2^{sr} \oplus F_4^{sr}) D_8 M_8 P_8^1 \tag{17}$$

where M_8 is the 8×8 modified Haar transform matrix. Iteratively using Eq. (16), M_8 can be rewritten as a product of three sparse matrices, which can be considered as the fast algorithm of the modified Haar transform,

$$M_8 = \left(\begin{array}{c:c} M_4 & 0 \\ \hdashline 0 & I_4 \end{array}\right) \left(\begin{array}{c:c} I_4 & I_4 \\ \hdashline -I_4 & I_4 \end{array}\right) = \left(\begin{array}{cc:c} M_2 & 0_2 & \\ 0_2 & I_2 & 0 \\ \hdashline & 0 & I_4 \end{array}\right) \left(\begin{array}{cc:c} I_2 & I_2 & \\ -I_2 & I_2 & 0 \\ \hdashline & 0 & I_4 \end{array}\right) \left(\begin{array}{c:c} I_4 & I_4 \\ \hdashline -I_4 & I_4 \end{array}\right) \tag{18}$$

and the diagonal matrix D_8 is the twiddle factor,

$$D_8 = diag\left\{1, 1, -j, 1, -\frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j, -j, \frac{\sqrt{2}}{2} - \frac{\sqrt{2}}{2}j, 1\right\} \tag{19}$$

and the matrix $(I_2 \oplus F_2^{sr} \oplus F_4^{sr})$ can be further decomposed,

$$I_2 \oplus F_2^{sr} \oplus F_4^{sr} = S_8^2 D_8^2 S_8^1 \tag{20}$$

where the matrices $S_8^2 = I_6 \oplus M_2, S_8^1 = I_2 \oplus M_2 \oplus M_4$ and diagonal matrix $D_8^2 = diag\{1, 1, 1, 1, 1, 1, -j, 1\}, P_8^2$ and P_8^1 are two permutation matrices,

$$P_8^2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 1 & 5 & 3 & 7 & 2 & 6 & 4 & 8 \end{pmatrix} \quad P_8^1 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 \\ 8 & 7 & 6 & 5 & 4 & 3 & 2 & 1 \end{pmatrix} \tag{21}$$

The flow diagram of 8-point MHT-FFT is shown in Fig. 3.

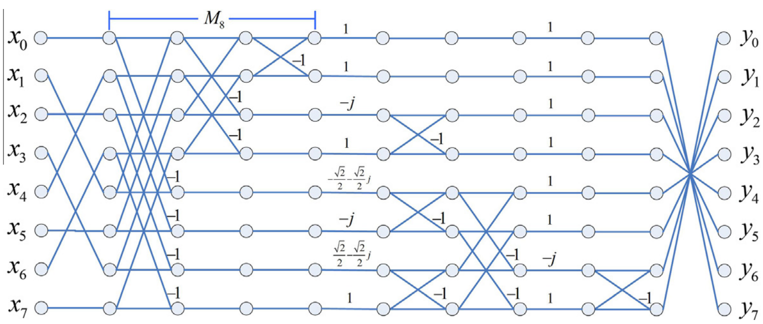

Fig. 3. The flow diagram of the 8-point MHT-FFT.

As can be seen, using MHT-FFT, the computation complexity of the 8-point FFT is reduced to 2 multiplications (does not count 2^n as the multiplication operator because it can be realized by shifting operations) and 26 additions.

2.3. Hybrid model based FFT representation (HMB-FFT)

Here, we discuss an FFT representation based on a Hybrid model, called HMB-FFT. It integrates the Haar and Hadamard transforms to split the input data sequence. The Hybrid matrix is defined as:

$$Hb_{2^{k+1}} = \begin{pmatrix} [1,-1]\otimes I_{2^k} \\ [1,1]\otimes Hb_{2^k} \end{pmatrix}, \quad k \geqslant 2; \tag{22}$$

where

$$Hb_2 = H_2 = \begin{pmatrix} 1 & 1 \\ 1 & -1 \end{pmatrix} \tag{23}$$

This model uses the 2×2 Hadamard matrix as the basic matrix, and employs the rule similar to the Haar matrix to generate the higher order of transform matrices. The Hybrid transform matrix can also be written into the form of matrix product as follows

$$Hb_{2^{k+1}} = \begin{pmatrix} I_{2^k} & 0 \\ 0 & Hb_{2^k} \end{pmatrix} \begin{pmatrix} I_{2^k} & -I_{2^k} \\ I_{2^k} & I_{2^k} \end{pmatrix} \tag{24}$$

Setting the Hybrid matrix to O_N in Eq. (3), an HMB-FFT is obtained. Here we give a 16-point HMB-FFT as an example to illustrate this FFT representation.

The 16-point HMB-FFT combining two orthogonal transforms, the Hadamard and Haar transforms, can be derived from Eq. (3) as

$$F_{16} = P_{16}^2(I_2 \oplus F_2 \oplus F_4 \oplus F_8)P_{16}^r D_{16} Hb_{16} \tag{25}$$

where Hb_{16} is the Hybrid matrix defined in Eq. (23). Iteratively using the matrix product form in Eq. (24), Hb_{16} can be written as follows:

$$\begin{aligned} Hb_{16} &= \left(\begin{array}{c|c} I_8 & 0 \\ \hline 0 & Hb_8 \end{array}\right)\left(\begin{array}{c|c} I_8 & -I_8 \\ \hline I_8 & I_8 \end{array}\right) = \left(\begin{array}{c|cc} I_8 & \multicolumn{2}{c}{0} \\ \hline & I_4 & 0 \\ 0 & 0 & Hb_4 \end{array}\right)\left(\begin{array}{c|cc} I_8 & \multicolumn{2}{c}{0} \\ \hline 0 & I_4 & -I_4 \\ & I_4 & I_4 \end{array}\right)\left(\begin{array}{c|c} I_8 & -I_8 \\ \hline I_8 & I_8 \end{array}\right) \\ &= \left(\begin{array}{c|c|cc} I_8 & \multicolumn{3}{c}{0} \\ \hline & I_4 & \multicolumn{2}{c}{0} \\ \cline{2-4} 0 & & I_2 & 0 \\ & 0 & 0 & Hb_2 \end{array}\right)\left(\begin{array}{c|c|cc} I_8 & \multicolumn{3}{c}{0} \\ \hline & I_4 & \multicolumn{2}{c}{0} \\ \cline{2-4} 0 & & I_2 & -I_2 \\ & 0 & I_2 & I_2 \end{array}\right)\left(\begin{array}{c|cc} I_8 & \multicolumn{2}{c}{0} \\ \hline 0 & I_4 & -I_4 \\ & I_4 & I_4 \end{array}\right)\left(\begin{array}{c|c} I_8 & -I_8 \\ \hline I_8 & I_8 \end{array}\right) \end{aligned} \tag{26}$$

and D_{16} is the diagonal matrix,

$$\begin{aligned} D_{16} &= diag\left\{1, W_{16}^1, W_{16}^2, W_{16}^3, W_{16}^4, W_{16}^5, W_{16}^6, W_{16}^7, 1, W_{16}^2, W_{16}^4, W_{16}^6, 1, W_{16}^4, 1, 1\right\} \\ &= diag\{1, 0.9239 - 0.3827j, 0.7071 - 0.7071j, 0.3827 - 0.9239j, -j, -0.3827 - 0.9239j, -0.7071 \\ &\quad - 0.7071j, -0.9239 - 0.3827j, 1, 0.7071 - 0.7071j, -j, -0.7071 - 0.7071j, 1, -j, 1, 1\} \end{aligned} \tag{27}$$

and P_{16}^2 is the permutation matrix,

$$P_{16}^2 = \begin{pmatrix} 1 & 2 & 3 & 4 & 5 & 6 & 7 & 8 & 9 & 10 & 11 & 12 & 13 & 14 & 15 & 16 \\ 1 & 9 & 5 & 13 & 3 & 11 & 7 & 14 & 2 & 10 & 6 & 15 & 4 & 12 & 8 & 16 \end{pmatrix} \tag{28}$$

Moreover, according to Eq. (3), the matrix F_8 can be further decomposed. The following F_8^{sr} is the row permutation of F_8,

$$F_8^{sr} = P_8^c(I_2 \oplus F_2 \oplus F_4)P_8^r D_8 Hb_8 \tag{29}$$

where F_2 and F_4 are the 2-point and 4-point DFT matrices, and the diagonal matrix $D_8 = diag(1, W_8^1, W_8^2, W_8^3, 1, W_8^2, 1, 1)$, Hb_8 is the 8×8 Hybrid transform matrix, and I_2, I_4 are identity matrices. Therefore, we have

$$I_2 \oplus F_2 \oplus F_4 \oplus F_8 = P_{16}^r S_{16}^2 D_{16}^2 S_{16}^3 P_{16}^3 S_{16}^2 S_{16}^1 P_{16}^3 \tag{30}$$

where D_{16}^2 is also a diagonal matrix defined by

$$D_{16}^2 = diag\left\{1, 1, 2^{-1}, -2^{-1}j, 2^{-2}, -2^{-2}j, -0.1768j, 0.1768, 1, 1, 2^{-1}, 2^{-1}j, 1, 1, 1, 1\right\} \tag{31}$$

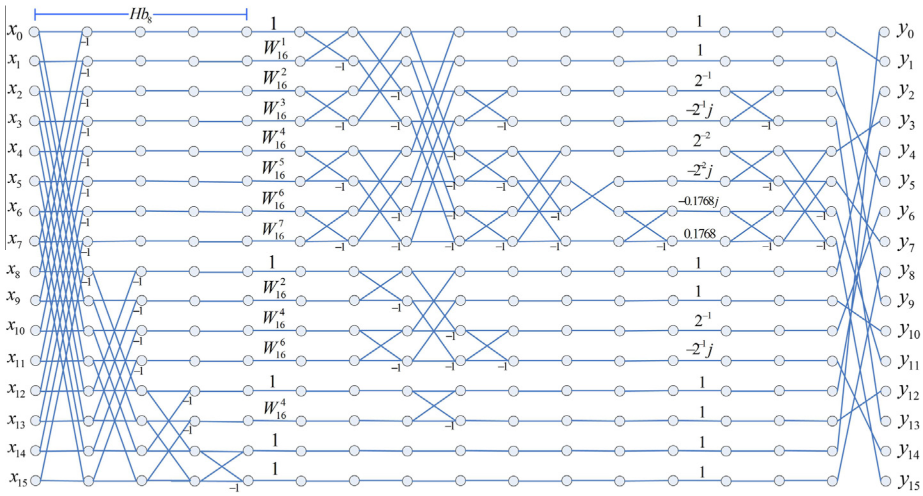

Fig. 4. The flow diagram of the 16-point HMB-FFT.

P_{16}^{3} and P_{16}^{r} are two permutation matrices,

$$P_{16}^{3}=\begin{pmatrix}1&2&3&4&5&6&7&8&9&10&11&12&13&14&15&16\\1&2&3&4&5&7&6&8&9&10&11&12&13&14&15&16\end{pmatrix}P_{2^{k+1}}^{r}=\begin{pmatrix}0&I_{2^k}\\P_{2^k}^{r}&0\end{pmatrix},\ P_{2}^{r}=I_2 \tag{32}$$

and $P_N^c=(P_N^r)^{-1}, S_{16}^3=I_6\oplus H_2\oplus I_8, S_{16}^1$ and S_{16}^2 can be further decomposed as,

$$\begin{cases}S_{16}^1=H_{16}^3H_{16}^2H_{16}^1\\S_{16}^2=H_{16}^5H_{16}^4\end{cases} \tag{33}$$

where $H_{16}^3=(Hb_2\otimes I_4)\oplus(Hb_2\otimes I_2)\oplus Hb_2\oplus I_2, H_{16}^2=(I_2\otimes(Hb_2\otimes I_2))\oplus(I_2\otimes Hb_2)\oplus I_4, H_{16}^1=(I_4\otimes Hb_2)\oplus I_8, H_{16}^5=I_4\oplus\ (Hb_2\otimes I_2)\oplus I_8$, and $H_{16}^4=I_2\oplus Hb_2\oplus(I_2\otimes Hb_2)\oplus I_2\oplus Hb_2\oplus I_4$.

Fig. 4 shows the flow diagram of the 16-point HMB-FFT structure. From this example, one can observe that, utilizing the hybrid model for the FFT representation, the computation complexity of the 16-points FFT is reduced to 10 multiplications (the scaling factor 2^n is not counted as the multiplication operator because it can be realized by shifting operations) and 88 additions.

2.4. Comparisons

Because the multiplication dominates the computation complexity of DFT, reducing the number of multiplications is an effective way to low down its computation costs. Integrating orthogonal transforms such as the Hadamard, modified Haar and Hybrid transforms, the proposed FFT scheme is able to iteratively decompose the N-point DFT into a number of sparse matrices. It reduces both the number of multiplications and total number of operations, and thus significantly reduces computation complexity.

Table 1 compares the computation complexity of the proposed scheme with those of three existing FFT algorithms including the traditional FFT [7], radix-2/8 FFT [5] and FFT with MSR-CORDIC [18]. HDT-FFT, MHT-FFT and HMB-FFT are three configurations of the proposed FFT scheme. $C^{\times}$ and C^{+} denote the numbers of real multiplications and additions, respectively. The scaling factor 2^n is not counted as the multiplication operator because it can be implemented by shifting operations. The computation complexity results of existing FFT algorithms in Table 1 directly come from the corresponding literatures.

Comparing three existing FFT algorithms in Table 1, the proposed FFT scheme with three different configurations has significantly less number of multiplications. In some cases, the number of additions is also greatly reduced. For example, to

Table 1
Comparison of computation complexity of different FFT algorithms.

N	Traditional FFT [7]		Radix-2/8 FFT [5]		MSR-CORDIC [18]		HDT-FFT		MHT-FFT		HMB-FFT	
	$C^{\times}$	C^{+}	$C^{\times}$	C^{+}	$C^{\times}$	C^{+}	$C^{\times}$	C^{+}	$C^{\times}$	C^{+}	$C^{\times}$	C^{+}
8	12	24	4	52	4	52	2	46	2	26	2	92
16	32	64	20	148	24	152	10	98	24	58	10	88
32	80	160	68	388	88	408	50	242	48	122	62	428

implement the 16-point DFT, HMB-FFT requires only 10 multiplications and 88 addictions. It save 14 multiplications and 64 additions compared with MSR-CORDIC proposed in [18]. The proposed FFT scheme outperforms these existing ones.

3. Proposed multi-stage image encryption algorithm

The orthogonal transforms, including DFT are widely used in image encryption [9,23,30]. Using the proposed FFT scheme, this section introduces a multi-stage image encryption algorithm (MSIEA). It is able to protect images with multiple security level. Fig. 5 shows its block diagram.

MSIEA has following encryption steps:

1. Divide the input image into subimages (for example, 128 × 128). Then each subimage is pre-processed by two permutation matrices:

$$E_{sub} = K^1 M_{sub} K^2 \tag{34}$$

where M_{sub} denotes the original subimage, K^1, K^2 are two permutation matrices (secret keys) with the same size of M_{sub}, and E_{sub} is the processed subimage.

2. Further decompose the pre-processed subimage into $m \times m$ ($m = 8$ or 16) non-overlapping blocks, and each block is transformed into the frequency domain,

$$T_m = F_m B_m F_m^T \tag{35}$$

where B_m is the $m \times m$ block obtained from the processed subimage in step 1, T_m is the encrypted block, F_m is the encryption system core, which is generated by insetting several secret keys (permutation matrices) into FFT matrices, and F_m^T is the transpose of F_m.

$$F_{m,l} = K_1 T_8^1 K_2 T_8^2 \cdots K_l T_N^k \tag{36}$$

where k and l are the numbers of sparse matrices T_N and secret keys K, respectively.

3. Reconstruct the transformed blocks to obtain the output encrypted image.

The multi-stage of MSIEA has two meanings: one is that the original image is decomposed by two stages, the other is that there are multi-level secret keys in each decomposition.

Here we give three cases of MSIEA that utilize three representations of the proposed FFT scheme as the system core F_m for image encryption. However, users have the flexibility to select other settings.

Case 1: Use HDT-FFT as the encryption system core, and $m = 8, l = 4$

$$F_{8,4} = K_1 S_8^1 D_8 K_2 S_8^2 K_3 (S_8^1)^T K_4 S_8 \tag{37}$$

Case 2: Use MHT-FFT as the encryption system core, and $m = 8, l = 3$

$$F_{8,3} = K_1 S_8^2 D_8^2 K_2 S_8^1 D_8 K_3 M_8 \tag{38}$$

Case 3: Use HBM-FFT as the system core, and $m = 16$, $l = 5$

$$F_{16,6} = K_1 S_{16}^2 D_{16}^2 K_2 S_{16}^3 K_3 S_{16}^2 K_4 S_{16}^1 D_{16}^1 K_5 (WH_{16}) \tag{39}$$

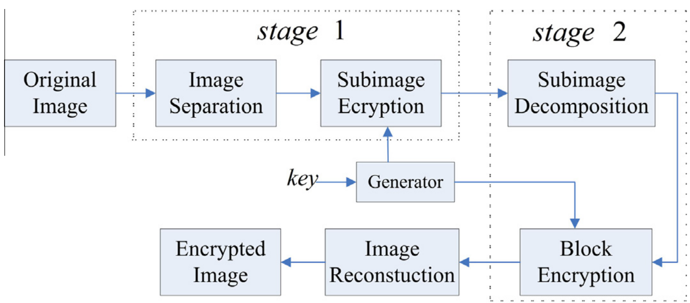

Fig. 5. The block diagram of MSIEA.

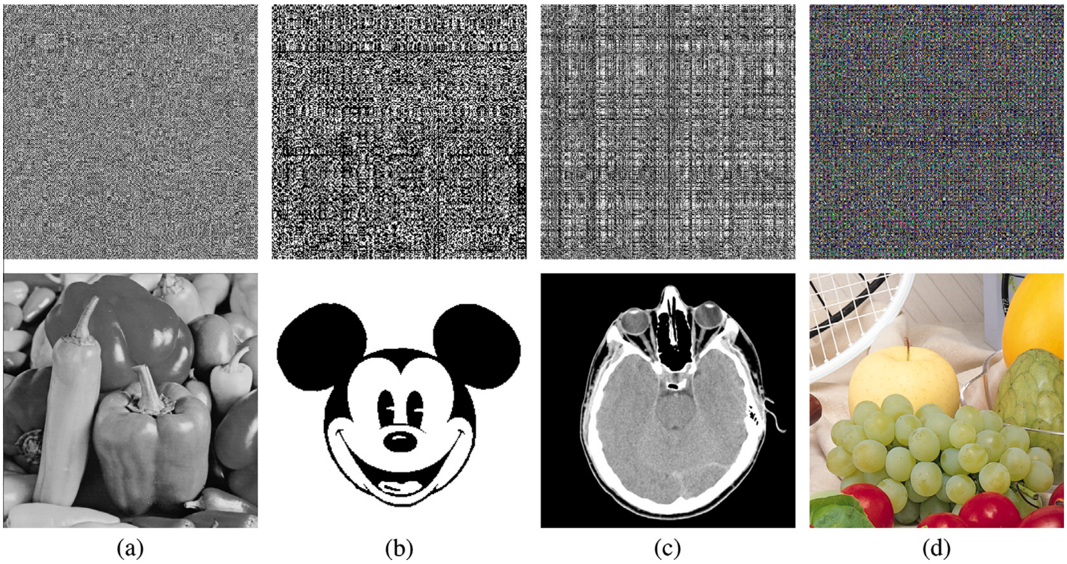

Fig. 6. Image encryption using MSIEA: the top and bottom rows show the encrypted and reconstructed images. (a) grayscale and (b) binary images using *Case 1*; (c) medical and (d) color images using *Case 2*.

The security key of the proposed MSIEA is composed of three parts: the number of subimages and their permutation matrices, the number of blocks decomposed from the subimages and the orthogonal transform in the system core. For image decryption, the inverse transform is applied to blocks to reconstruct the original image.

4. Experimental results and comparisons

This section provides several simulation results and compares the encryption speed of MSIEA with these of two existing encryption methods.

We have tested MSIEA on different types of images varying from natural images to synthetic images along with different sizes and color formats. Fig. 6 shows four encryption results using three mentioned encryption cores. For color image encryption, MSIEA is used to encrypt each color plane individually and then combine the encrypted color planes to obtain the encrypted color image. From Fig. 6, it can be observed that MSIEA is able to protect different types of images by transforming them into noise-like or texture-like images, and that MSIEA with different configurations has a similar encryption performance.

To show the encryption efficiency of the proposed MSIEA, we compare the MSIEA's computation complexity with those of two state-of-art encryption methods, the Tao's algorithm [23] and Wang's algorithm [27]. We use these algorithms to encrypt an image (Fig. 7(a)) with the size varying from 16×16 to 1024×1024. Experiments are carried out on a workstation with Intel Core i7 2.8 GHz and 4 GB RAM running Window 7 operating system. The comparison results are shown in Fig. 7. As we can observe, with the increase of the image size, the encryption time of MSIEA (the blue curve) gradually changes within 0.16 s for the image size of 1024×1024. However, the Tao's and Wang's algorithms need 5.93 and 35.79 s to encrypt a 1024×1024 image, which are 37 and 223 times more than the proposed MSIEA, respectively. Therefore, MSIEA is more efficient than these two existing image encryption algorithms.

5. Security analysis

This section analyzes the security performance of the proposed MSIEA. We discuss several its security issues including the security key space, key sensitivity and noise attack.

5.1. Security key space

The security key space of an encryption algorithm denotes total number of possible combinations of its security keys. The Brute-force attack is one common attack in which an attacker attempts to guess the correct security keys of an encryption algorithm by exhaustively searching its security key space. Thus, a sufficient large key space can ensure that the encryption algorithm withstands the Brute-force attack.

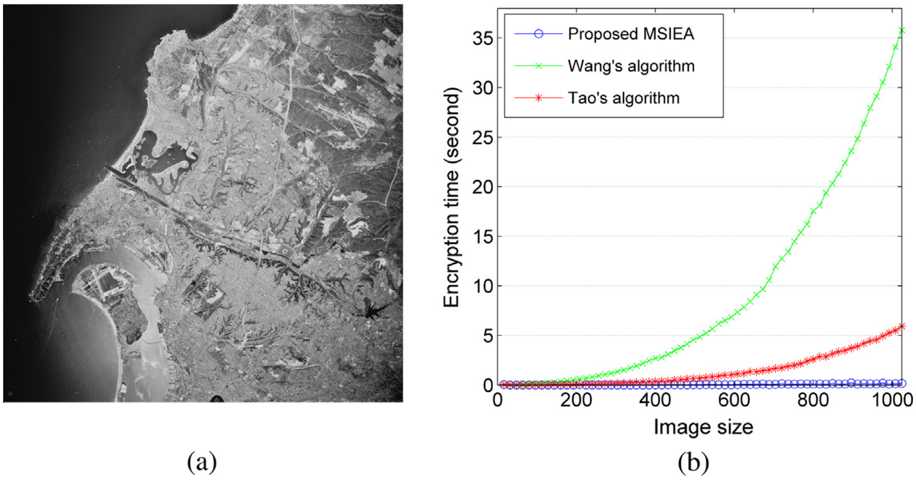

Fig. 7. Comparison of the computation complexity of different encryption algorithms. (a) The original image with a size of 1024×1024 and (b) Encryption time vs the image size.

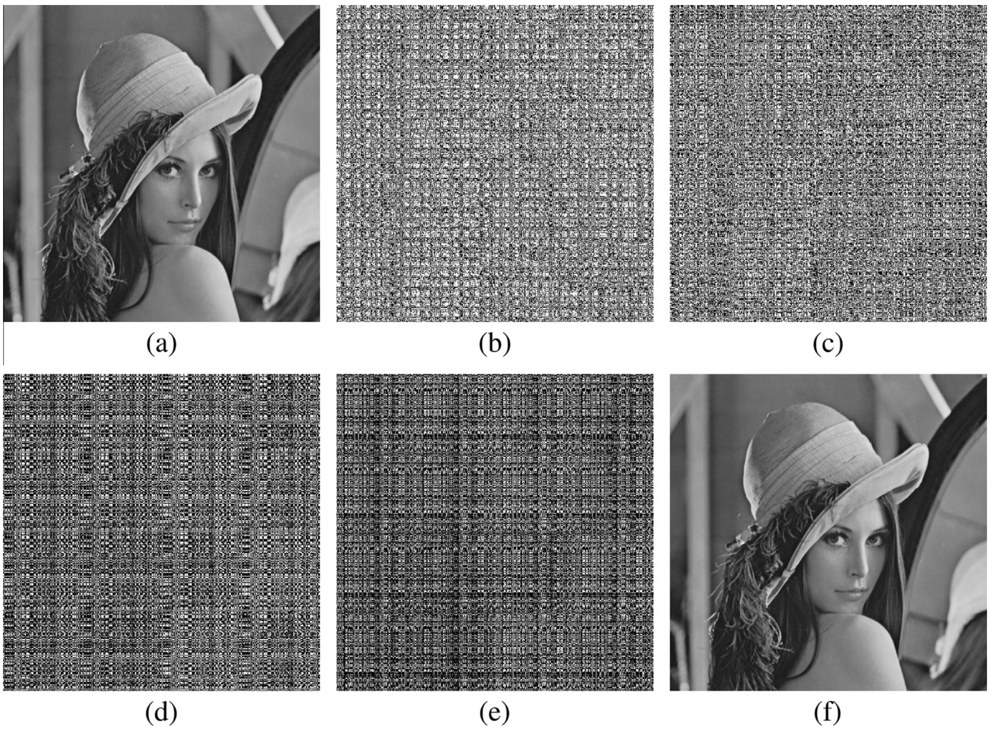

Fig. 8. Key sensitivity test of the proposed MSIEA with the system core in *Case 3*: (a) the original image with 512×512, (b) and (c) show the real and imaginary parts of the encrypted image, (d)–(f) show the decrypted images using the (d) totally wrong security keys, (e) 90% and (f) 100% correct security keys.

As mentioned in Section 3, the security key of the proposed MSIEA consist of three portions: (1) The number of subimages and their permutation matrices used in the pre-processing; (2) The number of blocks; and (3) The orthogonal transforms used in the proposed FFT scheme.

To calculate the security key space of the proposed MSIEA, we use a 32×32 grayscale image as an example. *KS* denotes the possible combinations of the security keys. First, MSIEA divides the original image into 4 subimages with the size of 16×16 and changes pixel positions within subimages using permutation matrices. Thus, possible key combinations in this stage is $KS_1 = 4 \times 16! \times 16! = 1.75 \times 10^{27}$. These subimages are then decomposed into sixteen 8×8 blocks which are transformed into the frequency domain using the proposed FFT scheme with 2 keys. Assume the propose FFT scheme is only selected from three foregoing cases. The number of key possibilities in this step: $KS_2 = 16 \times 2 \times 3 = 96$. Therefore, total possible key combinations of the proposed MSIEA are $KS = KS_1 \times KS_2 = 1.68 \times 10^{29}$. It is obviously huge enough to resist the Brute-force attack.

5.2. Key sensitivity test

Fig. 8 shows the results of key sensitivity test of the proposed MSIEA. One can see that MSIEA is highly sensitive to its security key changes. Even using the 90% correct security keys still results in a texture-like unrecognized decrypted image as shown in Fig. 8(e). Only employing the correct security keys can completely recover the original image as shown in Fig. 8(f).

5.3. Noise attack analysis

Noise attack is to test the capability of an encryption algorithm that recovers the original information from the encrypted image after being transmitted over noise channels. To evaluate the MSIEA's performance in against the noise attack, we select the image in Fig. 8(a) as the original image and *Case 3* as its system core. We compare MSIEA with two existing algorithms: the Tao's algorithm [23] and Wang's algorithm [27]. The original image is firstly encrypted by these algorithms. White additive noises with different strengthes (10, 30, 50 dB) are then added into these encrypted images, respectively. They are recovered by the corresponding algorithms. The recovered images are shown in Fig. 9. As can be seen, with the increase of the noise strength, recovered images contain more noise. The images recovered by the Tao's and Wang's

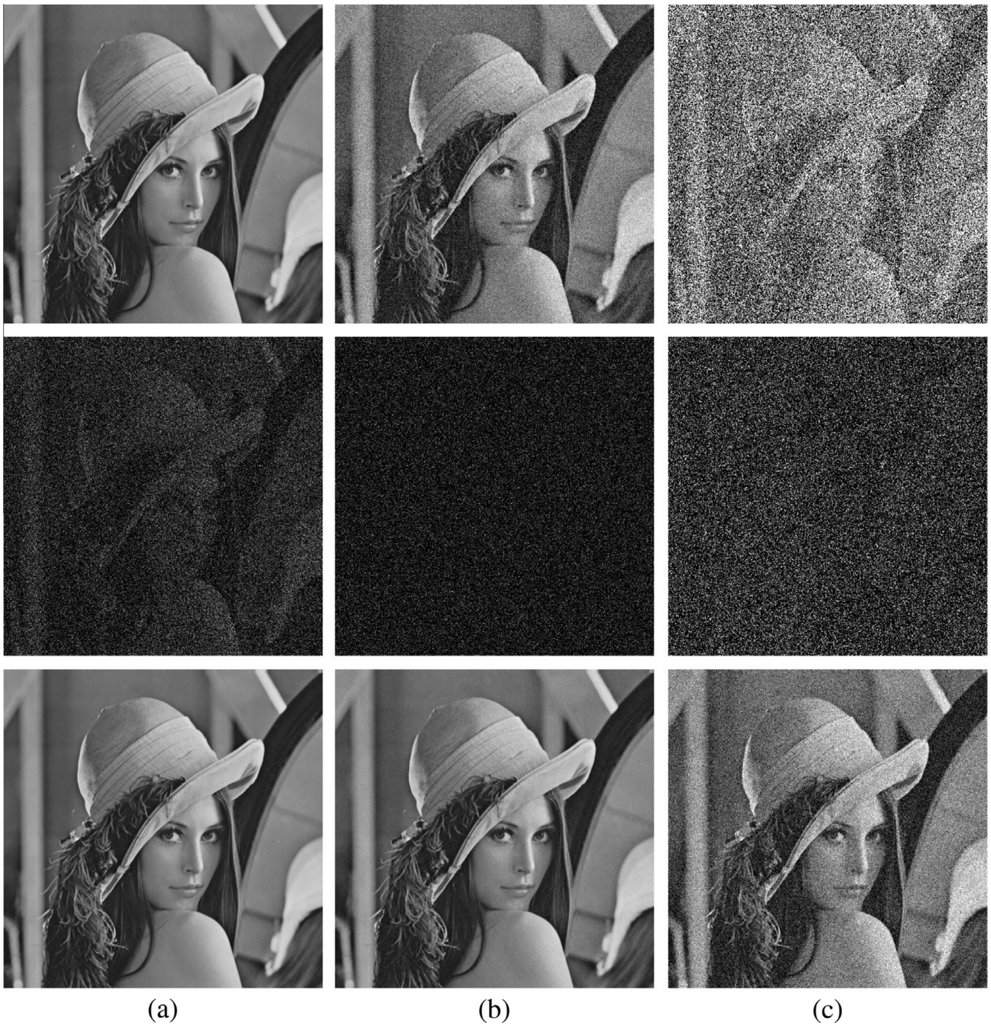

Fig. 9. Noise attack to different image encryption algorithms. The first, second and third rows show the recovered images by the Tao's algorithm [23], Wang's algorithm [27] and MSIEA, from their encrypted images with the noise strengths of (a) 10 dB, (b) 30 dB, and (c) 50 dB, respectively.

Table 2
MSE results of the images recovered by different algorithms.

Noise (dB)	Tao's	Wang's	MSIEA
10	3.32	4507	0.1693
30	333.4	6.986×10^3	0.3917
50	3.319×10^4	1.900×10^4	531

algorithms are almost unrecognized under 50 dB noise while the recovered image by MSIEA has much better visual quality for all noise levels.

To quantitatively evaluate the performance of different encryption algorithms in against the noise attack, we use the mean square error (MSE) defined in [10] to measure the difference between the original and recovered images. Table 2 lists the MSE values of the original image in Fig. 8a) and the recovered images in Fig. 9. We can observe that the MSE results of the Tao's and Wang's algorithms are much larger than those of the proposed MSIEA. This further demonstrates that MSIEA outperforms two existing encryption algorithms in against the noise attack.

6. Conclusion

Integrating different orthogonal transforms, this paper has proposed a fast Fourier transform scheme to iteratively decompose the N-point DFT into a set of mall sparse matrices. To show its effectiveness and performance, we have provided three illustrative examples using the Hadamard, modified Haar and Hybrid transforms. We also compared their computation complexity with existing FFT algorithms. To investigate the application of the proposed FFT scheme, we have introduced a multi-stage image encryption algorithm. Experimental comparisons and security analysis have demonstrated the proposed algorithm has good encryption performance and is able to protect different types of images with multiple security levels.

Acknowledgements

This work was supported in part by the Macau Science and Technology Development Fund under Grant FDCT/017/2012/A1 and by the Research Committee at University of Macau under Grants MYRG2014-00003-FST, MRG017/ZYC/2014/FST, MYRG113(Y1-L3)-FST12-ZYC and MRG001/ZYC/2013/FST.

References

[1] S. Agaian, H.G. Sarukhanyan, K.O. Egiazarian, J. Astola, Multidimensional Discrete Unitary Transforms: Representation, Partitioning, and Algorithms, SPIE Press, 2011.
[2] S. Agaian, K. Tourshan, J.P. Noonan, Parametric Slant-Hadamard transforms with applications, IEEE Signal Process. Lett. 9 (2002) 375–377.
[3] S. Agaian, K. Tourshan, J.P. Noonan, Parameterisation of Slant-Haar transforms, IEE Proc. – Vis. Image Signal Process. 150 (2003) 306–311.
[4] S.S. Agaian, O. Caglayan, Super fast Fourier transform, in: SPIE Electronic Imaging, pp. 60640F–60640F-12.
[5] S. Bouguezel, M.O. Ahmad, M.N.S. Swamy, A new radix-2/8 FFT algorithm for length-$q \times 2^m$ DFTs, IEEE Trans. Circ. Syst. I 51 (2004) 1723–1732.
[6] S. Bouguezel, M.O. Ahmad, M.N.S. Swamy, A general class of split-radix FFT algorithms for the computation of the DFT of length-2^m, IEEE Trans. Signal Process. 55 (2007) 4127–4138.
[7] J. Cooley, J. Tukey, An algorithm for machine computation of complex Fourier series, Math. Comput. 19 (1965) 297–301.
[8] A.P. Dhawan, Medical Image Analysis, Wiley-IEEE Press, 2011.
[9] B. Gaurav, Q.M.J. Wu, B. Raman, Discrete fractional wavelet transform and its application to multiple encryption, Inf. Sci. 223 (2013) 297–316.
[10] R.C. Gonzalez, R.E. Woods, Digital Image Processing, Prentice Hall, 3 ed., 2008.
[11] A.M. Grigoryan, S.S. Agaian, Split manageable efficient algorithm for Fourier and Hadamard transforms, IEEE Trans. Signal Process. 48 (2000) 172–183.
[12] A.M. Grigoryan, S.S. Agaian, Multidimensional Discrete Unitary Transforms: Representation, Partitioning, and Algorithms, CRC Press, 2003.
[13] W.C. Huang, C.P. Li, H.J. Li, A computationally efficient DFT scheme for applications with a subset of nonzero inputs, IEEE Signal Process. Lett. 15 (2008) 206–208.
[14] S.G. Johnson, M. Frigo, A modified split-radix FFT with fewer arithmetic operations, IEEE Trans. Signal Process. 55 (2007) 111–119.
[15] J.B. Lima, L.F.G. Novaes, Image encryption based on the fractional Fourier transform over finite fields, Signal Process. 94 (2014) 521–530.
[16] Z. Liu, S. Li, W. Liu, S. Liu, Opto-digital image encryption by using Baker mapping and 1-D fractional Fourier transform, Opt. Lasers Eng. 51 (2013) 224–229.
[17] M.M. Modeso, A.E. Miguel, C. Albertina, Input and/or output pruning of composite length FFTs using a DIF-DIT transform decomposition, IEEE Trans. Signal Process. 57 (2009) 4124–4128.
[18] S.Y. Park, Y.J. Yu, Fixed-point analysis and parameter selections of MSR-CORDIC with applications to FFT designs, IEEE Trans. Signal Process. 60 (2012) 6245–6256.
[19] W. Qin, X. Peng, Asymmetric cryptosystem based on phase-truncated Fourier transforms, Opt. Lett. 35 (2010) 118–120.
[20] S. Rawat, B. Raman, A blind watermarking algorithm based on fractional Fourier transform and visual cryptography, Signal Process. 92 (2012) 1480–1491.
[21] E.E. Swartzlander, H.H. Saleh, FFT implementation with fused floating-point operations, IEEE Trans. Comput. 61 (2012) 284–288.
[22] D. Takahashi, An extended split-radix FFT algorithm, IEEE Signal Process. Lett. 8 (2001) 145–147.
[23] R. Tao, X.Y. Meng, Y. Wang, Image encryption with multiorders of fractional Fourier transforms, IEEE Trans. Inform. Forensics Secur. 5 (2010) 734–738.
[24] T.K. Truong, P.D. Chen, L.J. Wang, Y. Chang, I.S. Reed, Fast, prime factor, discrete Fourier transform algorithms over $GF(2^m)$ for $8 \leqslant m \leqslant 10$, Inf. Sci. 176 (2006) 1–26.
[25] L. Wang, X. Zhou, G. Sobelman, R. Liu, Generic mixed-radix FFT pruning, IEEE Signal Process. Lett. 19 (2012) 167–170.
[26] Q. Wang, Q. Guo, L. Lei, J. Zhou, Iterative partial phase encoding based on joint fractional Fourier transform correlator adopting phase-shifting digital holography, Opt. Commun. 313 (2014) 1–8.

[27] Y. Wang, S. Zhou, A novel image encryption algorithm based on fractional Fourier transform, in: IEEE International Conference on Computer Science and Service System (CSSS), 2011, pp. 72–75.
[28] Y.G. Yang, X. Jia, S.J. Sun, Q.X. Pan, Quantum cryptographic algorithm for color images using quantum Fourier transform and double random-phase encoding, Inf. Sci. 277 (2014) 445–457.
[29] K. Zhang, J.U. Kang, Graphics processing unit accelerated non-uniform fast Fourier transform for ultrahigh-speed, real-time Fourier-domain OCT, Opt. Express 18 (2010) 23472–23487.
[30] Y. Zhou, k. Panetta, S. Agaian, Image encryption using discrete parametric cosine transform, in: Conference Record of the Forty-Third Asilomar Conference on Signals, Systems and Computers, 2009, pp. 395–399.