DISCRETE COSINE TRANSFORM OF ENCRYPTED IMAGES

Tiziano Bianchi, Alessandro Piva[*]

Università di Firenze
Dip. Elettronica e Telecomunicazioni
Via S. Marta 3, 50139, Firenze - Italy

Mauro Barni

Università di Siena
Dip. Ingegneria dell'Informazione
Via Roma 56, 53100, Siena - Italy

ABSTRACT

Processing a signal directly in the encrypted domain provides an elegant solution in application scenarios where valuable signals must be protected from a malicious processing device. In a previous paper we considered the implementation of the 1D Discrete Fourier Transform (DFT) in the encrypted domain, by using the homomorphic properties of the underlying cryptosystem. In this paper we extend our previous results by considering the application of the 2-dimensional DCT to encrypted images. The effect of the consecutive application of the DCT algorithm first by rows then by columns is considered, as well as the differences between the implementation of the direct DCT algorithm and its fast version. Particular attention is given to block-based DCT, with emphasis on the possibility of lowering the computational burden by parallel application of the encrypted domain DCT algorithm to different image blocks.

Index Terms— Discrete Cosine transforms, error analysis, homomorphic encryption, image encryption, signal processing in the encrypted domain

1. INTRODUCTION

The availability of signal processing modules that work directly on encrypted data would be of great help for applications where sensitive signals must be processed. In the image processing field, a recent example regards buyer-seller watermarking protocols [1] which prevent the seller from obtaining a plaintext of the watermarked copy, so that the image containing the buyer's watermark can not be illegally distributed to third parties by the seller. Signal processing in the encrypted domain (s.p.e.d.) is a new field of research aiming at developing a set of specific tools for processing encrypted data to be used as building blocks in a large class of applications. In image processing, one of such tools is the discrete cosine transform (DCT). The availability of an efficient s.p.e.d. DCT would allow a large number of processing tasks to be carried out on encrypted images, like the extraction of encrypted features from an encrypted image.

In [2], we considered the similar problem of implementing a discrete Fourier transform on encrypted data. Here, we will extend the previous results by considering a s.p.e.d. implementation of the DCT. We will assume that the chosen cryptosystem is *homomorphic* with respect to the addition, i.e., there exists an operator $\phi(\cdot,\cdot)$ such that

$$D[\phi(E[a],E[b])] = a + b \tag{1}$$

[*]The work described in this paper has been supported in part by the European Commission through the IST Programme under Contract no 034238 - SPEED. The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

where $E[\cdot]$ and $D[\cdot]$ denote the encryption and decryption operators. With such a cryptosystem it is indeed possible to add two encrypted values without first decrypting them and it is possible to multiply an encrypted value by a public integer value by repeatedly applying the operator $\phi(\cdot,\cdot)$. Moreover, we will assume that the cryptosystem is *probabilistic*, that is, given two encrypted values it is not possible to decide if they conceal the same value. This is fundamental, since the alphabet to which the input pixels belong is usually limited. A widely known example of a cryptosystem fulfilling both the above requirements is the Paillier cryptosystem [3], for which the operator $\phi(\cdot,\cdot)$ is a modular multiplication.

The DCT can be computed on the encrypted pixel values by relying on the homomorphic properties and the fact that the DCT coefficients are public. However, this requires several issues to be solved. The first one is that we must represent the pixel values, the DCT coefficients, and the transformed values in the domain of the cryptosystem, i.e., as integers on a finite field/ring. Another problem is that encrypted values can not be scaled or truncated by relying on homomorphic computations only. In general, for scaling the intermediate values of the computation we should allow two or more parties to interact [4]. However, since we would keep the s.p.e.d. DCT as simple as possible, it is preferable to avoid the use of interactive protocols. A final problem is that encrypting each pixel separately increases the size of the encrypted image and affects the complexity.

In this paper, we will provide solutions to the above issues. A convenient s.p.e.d. signal model will be proposed, allowing us to define both a s.p.e.d. DCT and a s.p.e.d. fast DCT and to extend them to the 2D case, by considering the consecutive application of the DCT first by rows then by columns. Moreover, we will propose a block-based s.p.e.d. DCT which permits the parallel application of the s.p.e.d. DCT algorithm to different image blocks, thus lowering both the bandwidth usage and the computational burden.

2. SIGNAL MODEL

We will describe the method assuming the signals are 1-D sequences. The extension to the 2-D case is straightforward by using separable processing along rows and columns. Let us consider a signal $x(n) \in \mathbb{R}$, $n = 0,\dots,M-1$. In the following, we will assume $|x(n)| \leq 1$. The scaled DCT of type II (DCT-II) of $x(n)$ is defined as

$$X(k) = \sum_{n=0}^{M-1} x(n) \cos\frac{\pi(2n+1)k}{2M}, \quad k = 0,1,\dots,M-1. \tag{2}$$

As in [2], the integer DCT is defined as

$$S(k) = \sum_{n=0}^{M-1} C_M(n,k)s(n), \quad k = 0,\dots,M-1 \tag{3}$$

where $s(n) = \lceil Q_1 x(n) \rfloor$, $C_M(n,k) = \left\lceil Q_2 \cos \frac{\pi(2n+1)k}{2M} \right\rfloor$, $\lceil \cdot \rfloor$ is the rounding function and Q_1 and Q_2 are suitable scaling factors.

Since all computations are between integers and there is no scaling, the expression above can be evaluated in the encrypted domain by relying on the homomorphic properties. If the inputs are encrypted with the Paillier cryptosystem, the s.p.e.d. DCT is

$$E[S(k)] = \prod_{n=0}^{M-1} E[s(n)]^{C_M(n,k)}, \quad k = 0, \ldots, M-1 \tag{4}$$

where all computations are done modulo N^2 [3].

3. S.P.E.D. DCT

The computation of the DCT using (3) requires two problems to be tackled with. The first one is that there will be a scaling factor between $S(k)$ and $X(k)$. The second one is that, if the cryptosystem encrypts integers modulo N, one must ensure that there is a one-to-one mapping between $S(k)$ and $S(k) \mod N$. A solution is to find an *upper bound* on $S(k)$ such that $|S(k)| \leq Q_S$, and verify that $N > 2Q_S$. We will show that $S(k)$ can be expressed in general as

$$S(k) = KX(k) + \epsilon_S(k) \tag{5}$$

where K is a suitable scaling factor and $\epsilon_S(k)$ models the quantization error. Based on the above equation, the desired DCT output can be estimated as $\tilde{X}(k) = S(k)/K$, and the upper bound is

$$Q_S = MK + \epsilon_{S,U} \tag{6}$$

where $\epsilon_{S,U}$ is an upper bound on $\epsilon_S(k)$. The value of both K and $\epsilon_{S,U}$ will depend on the particular implementation of the DCT.

3.1. Direct Computation

Let us express $s(n) = Q_1 x(n) + \epsilon_s(n)$ and $C_M(n,k) = Q_2 \cos \frac{\pi(2n+1)k}{2M} + \epsilon_C(n,k)$. If the DCT is directly computed by applying (3), then we have

$$S(k) = Q_1 Q_2 X(k) + \epsilon_S(k) \tag{7}$$

where $\epsilon_S(k) = \sum_{n=0}^{M-1} \left[Q_1 x(n) \epsilon_C(n,k) + Q_2 \epsilon_s(n) \cos \frac{\pi(2n+1)k}{2M} + \epsilon_s(n)\epsilon_C(n,k) \right]$. The scaling factor is $K_D = Q_1 Q_2$. As to the quantization error, we obtain the following upper bound

$$|\epsilon_S(k)| \leq M \left(\frac{Q_1}{2} + \frac{Q_2}{2} + \frac{1}{4} \right) = \epsilon_{S,U,D} \tag{8}$$

from which $Q_{S,D} = MQ_1Q_2 + \epsilon_{S,U,D}$.

3.2. Fast DCT

In order to obtain a s.p.e.d. version of the fast DCT, we will refer to the recursive matrix representation in [5]. Given $[\mathbf{T}_M]_{nk} = \cos \frac{\pi(2n+1)k}{2M}$, we have

$$\begin{aligned} \mathbf{T}_M =& \mathbf{P}_M \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{M/2} \end{bmatrix} \begin{bmatrix} \mathbf{T}_{M/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{M/2} \end{bmatrix} \\ & \times \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{M/2} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{J}_{M/2} \\ \mathbf{I}_{M/2} & -\mathbf{J}_{M/2} \end{bmatrix} \\ =& \mathbf{A}_M \begin{bmatrix} \mathbf{T}_{M/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{M/2} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{M/2} \end{bmatrix} \mathbf{B}_M \end{aligned} \tag{9}$$

where $\mathbf{D}_{M/2} = \operatorname{diag}\left\{ \cos \frac{\pi}{2M}, \cos \frac{3\pi}{2M}, \ldots, \cos \frac{(M-1)\pi}{2M} \right\}$,

$$\mathbf{L}_{M/2} = \begin{bmatrix} 1 & 0 & 0 & \ldots & 0 & 0 \\ -1 & 2 & 0 & & 0 & 0 \\ 1 & -2 & 2 & & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 1 & -2 & 2 & & 2 & 0 \\ -1 & 2 & -2 & \ldots & -2 & 2 \end{bmatrix},$$

$\mathbf{J}_M$ is obtained by the $M \times M$ identity matrix by reversing the column order and $\mathbf{P}_M$ is a suitable permutation matrix (see [5]).

Since the only non integer matrix in (9) is $\mathbf{D}_{M/2}$, the corresponding s.p.e.d. structure can be recursively defined as

$$\mathbf{C}_M = \mathbf{A}_M \begin{bmatrix} \mathbf{C}_{M/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{M/2} \end{bmatrix} \begin{bmatrix} Q_2 \mathbf{I}_{M/2} & \mathbf{0} \\ \mathbf{0} & \tilde{\mathbf{D}}_{M/2} \end{bmatrix} \mathbf{B}_M \tag{10}$$

where we define $\tilde{\mathbf{D}}_{M/2} = \lceil Q_2 \mathbf{D}_{M/2} \rfloor$.

As to the upper bound analysis, let us consider the mth stage of the recursion and express the quantized matrices as $\tilde{\mathbf{D}}_{2^m} = Q_2 \mathbf{D}_{2^m} + \mathbf{E}_D^{(m)}$ and $\mathbf{C}_{2^m} = K^{(m)} \mathbf{T}_{2^m} + \mathbf{E}_T^{(m)}$. Then, we can rewrite (10) as

$$\begin{aligned} \mathbf{C}_{2^{m+1}} =& \mathbf{A}_{2^{m+1}} \begin{bmatrix} K^{(m)} \mathbf{T}_{2^m} + \mathbf{E}_T^{(m)} & \mathbf{0} \\ \mathbf{0} & K^{(m)} \mathbf{T}_{2^m} + \mathbf{E}_T^{(m)} \end{bmatrix} \\ & \times \begin{bmatrix} Q_2 \mathbf{I}_{2^m} & \mathbf{0} \\ \mathbf{0} & Q_2 \mathbf{D}_{2^m} + \mathbf{E}_D^{(m)} \end{bmatrix} \mathbf{B}_{2^{m+1}} \\ =& K^{(m)} Q_2 \mathbf{T}_{2^{m+1}} \\ & + \mathbf{A}_{2^{m+1}} \left\{ \begin{bmatrix} K^{(m)} \mathbf{T}_{2^m} & \mathbf{0} \\ \mathbf{0} & K^{(m)} \mathbf{T}_{2^m} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_D^{(m)} \end{bmatrix} \right. \\ & + \begin{bmatrix} \mathbf{E}_T^{(m)} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_T^{(m)} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_D^{(m)} \end{bmatrix} \\ & \left. + \begin{bmatrix} \mathbf{E}_T^{(m)} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_T^{(m)} \end{bmatrix} \begin{bmatrix} Q_2 \mathbf{I}_{2^m} & \mathbf{0} \\ \mathbf{0} & Q_2 \mathbf{D}_{2^m} \end{bmatrix} \right\} \mathbf{B}_{2^{m+1}} \\ =& K^{(m+1)} \mathbf{T}_{2^{m+1}} + \mathbf{E}_T^{(m+1)} \end{aligned} \tag{11}$$

From the above equation, we have both a recursive relation on the scaling factor and a recursive relation on the quantization error. Let us consider the vector of quantized inputs $\mathbf{s} = [s(0), s(1), \ldots, s(M-1)]^T$. With a notation similar to the scalar case, we can express $\mathbf{s} = Q_1 \mathbf{x} + \mathbf{e}_S$, where $\mathbf{x}$ is vector containing the input values and $\mathbf{e}_S$ is a vector of quantization errors. Hence, the s.p.e.d. fast DCT is given by

$$\mathbf{C}_{2^\nu} \mathbf{s} = K^{(\nu)} Q_1 \mathbf{T}_{2^\nu} \mathbf{x} + K^{(\nu)} \mathbf{T}_{2^\nu} \mathbf{e}_s + \mathbf{E}_T^{(\nu)} Q_1 \mathbf{x} + \mathbf{E}_T^{(\nu)} \mathbf{e}_s. \tag{12}$$

As to the scaling factor, we have $K_F = K^{(\nu)} Q_1$. Since $K^{(0)} = 1$, it is easy to derive the final scaling factor as $K_F = Q_2^\nu Q_1$. As to the quantization error, we have $|\epsilon_S(k)| \leq MK^{(\nu)}/2 + (Q_1 + 1/2)||\mathbf{E}_T^{(\nu)}||_\infty$, where $||\cdot||_\infty$ denotes the maximum absolute row sum norm of a matrix. Based on (11), we can give an equivalent recursive relation on $||\mathbf{E}_T^{(m)}||_\infty$ as

$$||\mathbf{E}_T^{(m+1)}||_\infty \leq (2^{m+1} - 1) \left[2^m K^{(m)} + ||\mathbf{E}_T^{(m)}||_\infty (2Q_2 + 1) \right] \tag{13}$$

where we used $||\mathbf{A}_{2^{m+1}}||_\infty = 2^{m+1} - 1$, $||\mathbf{B}_{2^{m+1}}||_\infty = 2$, $||K^{(m)}\mathbf{T}_{2^m}||_\infty = 2^m K^{(m)}$, and $||\mathbf{E}_D^{(m)}||_\infty = 1/2$. At the start of the recursion we have $||\mathbf{E}_T^{(0)}||_\infty = 0$, since $\mathbf{T}_1 = 1$ and there is no quantization error. Hence, an upper bound on $||\mathbf{E}_T^{(\nu)}||_\infty$ can be derived as

$$||\mathbf{E}_T^{(\nu)}||_\infty \leq \sum_{k=0}^{\nu-1} (2Q_2+1)^k 2^{\nu-k} Q_2^{\nu-k} \prod_{r=\nu-k}^{\nu} (2^{r+1}-1) = \epsilon_{E,U} \tag{14}$$

from which we derive the upper bound on the quantization error as

$$|\epsilon_S(k)| \leq \frac{MQ_2^\nu}{2} + \left(Q_1 + \frac{1}{2}\right)\epsilon_{E,U} = \epsilon_{S,U,F}. \tag{15}$$

Finally, the upper bound on $S(k)$ is $Q_{S,F} = MQ_1Q_2^\nu + \epsilon_{S,U,F}$.

4. EXTENSION TO 2D-DCT

In the case of separable processing of the rows and the columns of an image, the expressions derived in the preceding section can be extended to the 2D case in an easy way. Let us assume that the 2D-DCT processes first the rows and then the columns. After the processing of the rows, the input to the next DCT will be expressed as in (5). Hence, the scaling factor can be obtained by substituting Q_1 with K whereas the upper bound on the quantization error can be derived by noting that $|S(k)| \leq MK + \epsilon_{S,U}$ and $|\epsilon_S(k)| \leq \epsilon_{S,U}$.

In the case of the direct DCT implementation, this leads to

$$K_D^{2D} = Q_2 K_D = Q_2^2 Q_1 \tag{16}$$

$$\epsilon_{S,U,D}^{2D} = M\left(\frac{MK_D}{2} + Q_2\epsilon_{S,U,D} + \frac{\epsilon_{S,U,D}}{2}\right) \tag{17}$$

$$Q_{S,D}^{2D} = M^2 K_D^{2D} + \epsilon_{S,U,D}^{2D} \tag{18}$$

whereas in the case of the fast DCT we obtain

$$K_F^{2D} = Q_2^\nu K_F = Q_2^{2\nu} Q_1 \tag{19}$$

$$\epsilon_{S,U,F}^{2D} = MQ_2^\nu \epsilon_{S,U,F} + (MK_F + \epsilon_{S,U,F})\,\epsilon_{E,U} \tag{20}$$

$$Q_{S,F}^{2D} = M^2 K_F^{2D} + \epsilon_{S,U,F}^{2D}. \tag{21}$$

5. S.P.E.D. BLOCK-BASED DCT

Several image processing algorithms, instead of applying the DCT to the whole image, subdivide it into equal sized (usually square) blocks and compute the DCT of each block. The size of such blocks is usually quite small: typically 8×8 blocks or 16×16 blocks are used in most of the applications.

From the s.p.e.d. perspective, this suggests two things: first, even if rescaling is not applied, in the case of a block based s.p.e.d. DCT the maximum value of the DCT outputs will not be very high. Since the modulus of practical cryptosystems has a size of one thousand bits or more, it is expected that the outputs of the s.p.e.d. block-based DCT will be far for exploiting the full bandwidth of the modulus. Second, each block undergoes exactly the same processing. Hence, this permits a parallel processing of several blocks by simply packing the pixels having the same position within the blocks in a single word.

In order to exploit the above ideas, we propose a s.p.e.d. block DCT (BDCT) based on a different representation of the input pixels. Let us consider R distinct blocks of an image. For the sake of simplicity, we can assume the blocks as one-dimensional, having size M, since the extension to the 2D case is straightforward using separable processing. Let us define the block bandwidth as $B = \lfloor \sqrt[R]{N} \rfloor$. Moreover, let us assume that the input pixel values have been quantized as in Section 2. The pixels having the same position within each block are packed in a single word as

$$s_P(n) = \sum_{i=0}^{R-1} [s_i(n) + Q_1]B^i \tag{22}$$

where $s_i(n)$ denotes the pixel having position n within the ith block. If $s_i(n) + Q_1 < B$, $i = 0, 1, \ldots, R-1$ then the packed word can be thought as a base B positive number whose digits are given by $s_i(n) + Q_1$. Note that the offset Q_1 is required in order to have positive digits. The above condition is surely satisfied if $B > 2Q_1$. Moreover, thanks to the definition of B we have $|s_P(n)| < N$. Hence, given the modulo N representation of $s_P(n)$ one can always extract the correct values of $s_i(n)$, $i = 0, 1, \ldots, R-1$ (That is, it is possible to define a one-to-one mapping between $s_P(n)$ and $[s_0(n), s_1(n), \ldots, s_{R-1}(n)]$.).

The s.p.e.d. BDCT is defined as

$$S_P(k) = \sum_{n=0}^{M-1} s_P(n)C(n,k) - \Omega(k) \tag{23}$$

where

$$\Omega(k) = \sum_{i=0}^{R-1} \left[Q_1 \sum_{n=0}^{M-1} C(n,k) - Q_S\right] B^i. \tag{24}$$

Theorem 1 *The BDCT satisfies*

$$S_P(k) = \sum_{i=0}^{R-1} [S_i(k) + Q_S]B^i \tag{25}$$

where $S_i(k)$ is the s.p.e.d. DCT of $s_i(n)$. Moreover, if $B > 2Q_S$ then the BDCT is correctly defined modulo N.

Proof: let us consider the following equalities

$$\begin{aligned}\sum_{n=0}^{M-1} s_P(n)C(n,k) &= \sum_{n=0}^{M-1}\sum_{i=0}^{R-1} [s_i(n)+Q_1]B^i C(n,k) \\ &= \sum_{i=0}^{R-1}\sum_{n=0}^{M-1} [s_i(n)+Q_1]C(n,k)B^i \\ &= \sum_{i=0}^{R-1}\left[S_i(k) + Q_1\sum_{n=0}^{M-1} C(n,k)\right]B^i.\end{aligned} \tag{26}$$

Then, by subtracting (24) from the last equality (25) is readily proved. Moreover, if $B > 2Q_S$ then $|S_P(k)| < N$, so that it is possible to recover $S_i(k)$, $i = 0, 1, \ldots, R-1$ from $S_P(k)$.

By using the BDCT we are able to process R blocks using a single s.p.e.d. DCT. Therefore, the complexity of the s.p.e.d. BDCT is reduced by a factor R with respect to that of the s.p.e.d. DCT. Moreover, also the bandwidth usage is reduced by the same factor, since we pack R pixels into a single cyphertext. However, note that extracting a single encrypted coefficient from the packed word requires some interactive protocol.

Finally, we would like to point out that the fast DCT algorithm can be used for the BDCT as well. The fast BDCT algorithm can be described by the following steps: 1) compute the fast DCT of the packed signal $s_P(n)$; 2) compute the offset $\Omega(k)$ by applying the

fast DCT to a vector containing all Q_1s and by using $Q_{S,F}$ in (24); compute the fast BDCT by subtracting $\Omega(k)$ found in 2) from the result of 1). In order to verify that the above algorithm is correct, it suffices to substitute $C(n,k)$ in (23)-(24) with the (n,k) element of the matrix $\mathbf{C}_M$ as defined in (10).

6. NUMERICAL EXAMPLES

We will consider the application of the s.p.e.d. 2D-DCT and 2D-BDCT to square $M \times M$ 8-bit greyscale images. The quantization scaling factor can be assumed as $Q_1 = 128$. As to Q_2, we will assume that the cosine values are quantized so as not to exceed the quantization error of the corresponding plaintext implementation. Three plaintext implementations are considered: 1) 16-bit fixed point (XP); 2) single precision floating point (FP1); 3) double precision floating point (FP2). In the first case, we can assume $Q_2 = 2^{15}$. In the floating point case, since the smallest magnitude of a cosine value is equal to $\sin(\pi/2M)$, we need $Q_2 > 2^f / \sin(\pi/2M)$, where f is the number of bits of the fractional part of the floating point representation. For the sake of simplicity, we will assume $M \leq 4096$, so that we can choose $Q_2 = 2^{36}$ (FP1) and $Q_2 = 2^{65}$ (FP2).

Since the values of Q_S in (18)-(21) can be huge, in the case of the full frame DCT we will consider an upper bound on the number of bits required in order to correctly represent the DCT outputs. If we assume $Q_{S,Z}^{2D} < 2M^2 K_Z^{2D}$, this can be expressed as

$$\lceil \log_2 Q_{S,Z}^{2D} \rceil + 1 < 2\nu + \lceil \log_2 K_Z^{2D} \rceil + 2 = n_{U,Z} \qquad (27)$$

where $\nu = \log_2 M$ and $Z = \{D, F\}$. Note that if $\log_2 N > n_{U,Z}$, it follows that $N > 2Q_{S,Z}$. In Table 1, we give some upper bounds considering different values of M and Q_2. Highlighted in bold are the cases which can not be implemented relying on a 1024-bit modulus, which is a standard in several cryptographic applications. As can be seen, except for the case of FP2, a full frame s.p.e.d. DCT can be always implemented relying on a standard modulus.

As to the s.p.e.d. 2D-BDCT, we consider an estimate of the number of pixels that can be safely packed into a single word. A safe implementation requires $B = \lceil 2Q_{S,Z} \rceil$. Since we must have $B < \sqrt[R]{N}$, this leads to

$$R_{max} = \left\lfloor \frac{\log_2 N}{\log_2 \lceil 2Q_{S,Z} \rceil} \right\rfloor \approx \left\lfloor \frac{\lfloor \log_2 N \rfloor}{\log_2 \lceil 2Q_{S,Z} \rceil} \right\rfloor = R_{U,Z}. \qquad (28)$$

In Table 1, we give some values of $R_{U,Z}$ considering 8×8 and 16×16 BDCTs and different precisions. The results demonstrate that the s.p.e.d. BDCT approach can effectively reduce both the bandwidth requirements and the complexity, especially for the fixed point case. It is worth noting that a direct implementation allows us to increase $R_{U,Z}$ up to three times with respect to the fast BDCT. Since the BDCT usually works with small sized blocks, the complexity of the direct implementation will not be much higher than that of the fast implementation. For instance, an 8×8 fast DCT requires 12 multiplications [6] versus the 64 multiplications of a naive direct DCT. Hence, there can be cases in which it is preferable to employ a direct s.p.e.d. BDCT, since this will reduce the bandwidth usage at a small cost in complexity.

7. CONCLUDING REMARKS

We have considered the implementation of the DCT on an encrypted image relying on the homomorphic properties of the underlying cryptosystem. It has been shown how the maximum allowable DCT size depends on the modulus of the cryptosystem, on the chosen DCT implementation, and on the required precision. We have also proposed a s.p.e.d. block DCT which is based on the packing of several pixels into a single encrypted word, thus permitting the parallel application of the s.p.e.d. DCT algorithm to different image blocks. The results demonstrate that the proposed s.p.e.d. BDCT can effectively lower both the bandwidth usage and the computational burden. Our approach gives useful design criteria for the implementation of s.p.e.d. image processing modules and suggests other issues to be addressed in future research on s.p.e.d. topics, for example the tradeoff between bandwidth usage and complexity.

Table 1. Upper bounds (in bits) on the output values of s.p.e.d. 2D-DCTs having different size. $Q_2 = 2^{15}$ is equivalent to a 16 bit fixed point implementation. $Q_2 = 2^{36}$ and $Q_2 = 2^{65}$ are equivalent to a single precision and a double precision floating point implementations, respectively. A square $M \times M$ 2D-DCT has been considered.

	$Q_2 = 2^{15}$		$Q_2 = 2^{36}$		$Q_2 = 2^{65}$	
M	$n_{U,D}$	$n_{U,F}$	$n_{U,D}$	$n_{U,F}$	$n_{U,D}$	$n_{U,F}$
64	51	201	93	453	151	801
256	55	265	97	601	155	**1065**
1024	59	329	101	749	159	**1329**
4096	63	393	105	897	163	**1593**

Table 2. Upper bounds on the number of blocks R that can be processed in parallel by a s.p.e.d. $M \times M$ 2D-BDCT. We have assumed $\lfloor \log_2 N \rfloor = 1023$.

	$Q_2 = 2^{15}$		$Q_2 = 2^{36}$		$Q_2 = 2^{65}$	
M	$R_{U,D}$	$R_{U,F}$	$R_{U,D}$	$R_{U,F}$	$R_{U,D}$	$R_{U,F}$
8	23	8	11	4	7	2
16	22	6	11	3	7	1

8. REFERENCES

[1] N. Memon and P. Wong, "A buyer-seller watermarking protocol," *IEEE Trans. on Image Proc.*, vol. 10, no. 4, pp. 643–649, Apr. 2001.

[2] T. Bianchi, A. Piva, and M. Barni, "Implementing the discrete Fourier transform in the encrypted domain," in *Proc. of ICASSP 2008*, to appear.

[3] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Lecture Notes in Computer Science*. 1999, vol. 1592, pp. 223–238, Springer-Verlag.

[4] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," *Lecture Notes in Computer Science*, vol. 2045, pp. 280–299, 2001.

[5] Yonghong Zeng, Lizhi Cheng, Guoan Bi, and A. C. Kot, "Integer DCTs and fast algorithms," *IEEE Trans. Signal Processing*, vol. 49, no. 11, pp. 2774–2782, Nov. 2001.

[6] Hsieh Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Trans. Acoust., Speech, Signal Processing*, vol. 35, no. 10, pp. 1455–1461, Oct. 1987.