Hindawi Publishing Corporation
EURASIP Journal on Information Security
Volume 2009, Article ID 716357, 12 pages
doi:10.1155/2009/716357

Research Article

Encrypted Domain DCT Based on Homomorphic Cryptosystems

Tiziano Bianchi,[1] Alessandro Piva,[1] and Mauro Barni (EURASIP Member)[2]

[1] *Department of Electronics and Telecommunications, University of Florence, Via Santa Marta 3, I-50139 Florence, Italy*
[2] *Department of Information Engineering, University of Siena, Via Roma 56, I-53100 Siena, Italy*

Correspondence should be addressed to Tiziano Bianchi, tiziano.bianchi@unifi.it

Received 30 March 2009; Accepted 29 September 2009

Recommended by Sen-Ching Samson Cheung

Signal processing in the encrypted domain (s.p.e.d.) appears an elegant solution in application scenarios, where valuable signals must be protected from a possibly malicious processing device. In this paper, we consider the application of the Discrete Cosine Transform (DCT) to images encrypted by using an appropriate homomorphic cryptosystem. An s.p.e.d. 1-dimensional DCT is obtained by defining a convenient signal model and is extended to the 2-dimensional case by using separable processing of rows and columns. The bounds imposed by the cryptosystem on the size of the DCT and the arithmetic precision are derived, considering both the direct DCT algorithm and its fast version. Particular attention is given to block-based DCT (BDCT), with emphasis on the possibility of lowering the computational burden by parallel application of the s.p.e.d. DCT to different image blocks. The application of the s.p.e.d. 2D-DCT and 2D-BDCT to 8-bit greyscale images is analyzed; whereas a case study demonstrates the feasibility of the s.p.e.d. DCT in a practical scenario.

1. Introduction

The availability of signal processing modules that work directly on encrypted data would be of great help to satisfy the security requirements stemming from applications wherein valuable or sensible signals have to be processed by a nontrusted party [1, 2]. In the image processing field, there are two recent examples regarding buyer-seller watermarking protocols [3] which prevent the seller from obtaining a plaintext of the watermarked copy, so that the image containing the buyer's watermark cannot be illegally distributed to third parties by the seller, and the access to image databases by means of encrypted queries [4], in order to avoid the disclosure of the content of the query image.

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, or watermark embedding in encrypted images. As a simple example, let us consider a scenario where a party P_1 needs to process an image by means of a signal processing system known by another party P_2. Let us assume that P_1 is concerned about the privacy of his image, so that not to reveal the image content to the service provider P_2, he will send the image in encrypted form. In the processing chain, it is possible that there is the need to apply a DCT to the image, for example, to apply a watermark in such a domain, or to reduce to zero some coefficients in order to reduce the image bit rate. After this step, an Inverse DCT (IDCT) will be needed; in such a scenario, both DCT and IDCT will need to be performed in the encrypted domain.

In [5, 6], we considered the similar problem of implementing a discrete Fourier transform on encrypted data. Here, we will extend the previous results by considering an s.p.e.d. implementation of the DCT. In the following we will concentrate on images, however we point out that a similar approach can be applied to 1-dimensional signals as well, like digitized audio. We will assume that an image is encrypted pixelwise by means of a cryptosystem *homomorphic* with respect to the addition that is, there exists an operator $\phi(\cdot,\cdot)$ such that

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

where $\mathcal{E}[\cdot]$ and $\mathcal{D}[\cdot]$ 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 whether they conceal the same value or not. This is fundamental, since the alphabet to which the input pixels belong usually has a limited size. As it will be detailed in the following section, a widely known example of a cryptosystem fulfilling both the above requirements is the Paillier cryptosystem [7], for which the operator $\phi(\cdot,\cdot)$ is a modular multiplication. Apart from [5, 6], previous examples of the use of homomorphic cryptosystems for performing encrypted computations can be found in buyer-seller protocols [3, 8], zero-knowledge watermark detection [9], and private scalar product computation [10].

Adopting such a cryptosystem, 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, that is, as integers on a finite field/ring. Another problem is that encrypted values cannot 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 [11, 12]. 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.

1.1. Our Contributions. Solutions to the above issues will be provided in this paper, whose rest is organized as follows. In Section 2 a brief review of homomorphic cryptosystems, with particular attention to the Paillier scheme, is given. In order to properly represent the pixel values, the DCT coefficients and the transformed values in the encrypted domain, a convenient s.p.e.d. signal model is proposed in Section 3. Such a model allows us to define in Section 4 both an s.p.e.d. DCT and an s.p.e.d. fast DCT and to extend them to the 2D case. The proposed representation permits also to avoid the use of interactive protocols, by letting the magnitude of the intermediate results propagates to the end of the processing chain. A solution to the problem of encrypting each pixel separately is proposed in Section 5. A block-based s.p.e.d. DCT, relying on a suitable composite representation of the encrypted pixels, 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. In Section 6 we consider the application of the s.p.e.d. 2D-DCT and 2D-BDCT to 8-bit greyscale images, computing the upper bound on the number of bits required in order to correctly represent the DCT outputs, and, for the s.p.e.d. 2D-BDCT, the number of pixels that can be safely packed into a single word. Section 7 describes a case study where the feasibility of the s.p.e.d. DCT in a practical scenario is analyzed. Finally, conclusions are drawn in Section 8.

2. Probabilistic Homomorphic Encryption

As already defined in the previous section, a homomorphic cryptosystem allows to carry out some basic algebraic operations on encrypted data by translating them into corresponding operations in the plaintext domain. The concept of privacy homomorphism was first introduced by Rivest et al. [13] that defined privacy homomorphisms as encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands.

According to the correspondence between the operation in the ciphertext domain and the operation in the plaintext domain, a cryptosystem can be additively homomorphic or multiplicatively homomorphic. In this paper we are interested in the former. Additively homomorphic cryptosystems allow, in fact, to perform additions, subtractions and multiplications with a known (nonencrypted) factor in the encrypted domain. More extensive processing would be allowed by the availability of an algebraically homomorphic encryption scheme, that is, a scheme that is additive and multiplicative homomorphic. Very recently, a fully homomorphic scheme has been proposed in [14], but its complexity seems too high for practical applications.

Another crucial concept for the s.p.e.d. framework is probabilistic encryption. As a matter of fact, many of the most popular cryptosystems are deterministic, that is, given an encryption key and a plaintext, the ciphertext is univocally determined. The main drawback of these schemes for s.p.e.d. applications is that it is easy for an attacker to detect if the same plaintext message is encrypted twice. Indeed, since usually signal samples assume only a limited range of values, an attacker will be easily able to decrypt the ciphertexts, or at least to derive meaningful information about them. In [15] the concept of probabilistic or semantically secure cryptosystem has been proposed. In such schemes, the encryption function $\mathcal{E}[\cdot]$ is a function of both the secret message m and a random parameter r that is changed at any new encryption. Specifically, two subsequent encryptions of the same message m result in two different encrypted messages $c_1 = \mathcal{E}[m, r_1]$ and $c_2 = \mathcal{E}[m, r_2]$. Of course, the scheme has to be designed in such a way that $\mathcal{D}[c_1] = \mathcal{D}[c_2] = m$, that is, the decryption phase is deterministic and does not depend on the random parameter r. Luckily, encryption schemes that satisfy both the homomorphic and probabilistic properties detailed above do exist. One of the most known examples is the scheme presented by Paillier in [7], and later modified by Damgård and Jurik in [16]. It should be pointed out that homomorphic cryptosystems are usually more computationally demanding than symmetric ciphers, like AES, and require longer keys to achieve a comparable level of security. Furthermore, probabilistic cryptosystems cause an intrinsic data expansion due to the adoption of randomizing parameters in the encryption function.

2.1. Paillier Cryptosystem. The Paillier cryptosystem [7] is based on the problem to decide whether a number is an Nth residue modulo N^2. This problem is believed to be computationally hard in the cryptographic community, and is linked to the hardness to factorize N, if N is the product of two large primes.

Let us now explain what an N-th residue is and how it can be used to encrypt data. Given the product of two large primes $N = pq$, the set $\mathbb{Z}_N$ of the integer numbers modulo N, and the set $\mathbb{Z}_N^*$ representing the integer numbers belonging to $\mathbb{Z}_N$ that are relatively prime with N, $z \in \mathbb{Z}_{N^2}^*$ is said to be a N-th residue modulo N^2 if there exists a number $y \in \mathbb{Z}_{N^2}^*$ such that

$$z = y^N \mod N^2. \tag{2}$$

For a complete analysis of the Paillier cryptosystem we refer to the original paper [7]. Here, we simply describe the set-up, encryption, and decryption procedures.

2.1.1. Set-Up. Select p, q big primes. The private key is the least common multiple of $(p-1, q-1)$, denoted as $\lambda = lcm(p-1, q-1)$. Let $N = pq$ and g in $\mathbb{Z}_{N^2}^*$ an element of order αN for some $\alpha \neq 0$. The order of an integer a modulo N is the smallest positive integer k such that $a^k = 1 \mod N$. In such a case, $g = N+1$ is usually a convenient choice. (N, g) is the public key.

2.1.2. Encryption. Let $m < N$ be the plaintext, and $r < N$ a random value. The encryption c of m is

$$c = \mathcal{E}[m, r] = g^m r^N \mod N^2. \tag{3}$$

2.1.3. Decryption. Let $c < N^2$ be the ciphertext. The plaintext m hidden in c is

$$m = \mathcal{D}[c] = \frac{L\left(c^\lambda \mod N^2\right)}{L(g^\lambda \mod N^2)} \mod N. \tag{4}$$

where $L(x) = (x-1)/N$. From the above equations, we can easily verify that the Paillier cryptosystem is additively homomorphic, since

$$\begin{gathered}\mathcal{E}[m_1, r_1] \cdot \mathcal{E}[m_2, r_2] = g^{m_1+m_2}(r_1 r_2)^N = \mathcal{E}[m_1 + m_2, r_1 r_2], \\ \mathcal{E}[m, r]^a = \left(g^m (r)^N\right)^a = \left(g^{am}(r)^{aN}\right) = \mathcal{E}[am, r^a].\end{gathered} \tag{5}$$

3. Signal Model for the Encrypted Domain

We will describe the proposed representation assuming the signals are 1D sequences. The extension to the 2D case is straightforward by using separable processing along rows and columns. Let us consider a signal $x(n) \in \mathbb{R}$, $n = 0, \ldots, M-1$. In the following, we will assume that the signal has been properly scaled so that $|x(n)| \leq 1$. In order to process $x(n)$ in the encrypted domain, its values have to be represented as integer numbers belonging to $\mathbb{Z}_N$. This is accomplished by first defining an integer version of $x(n)$ as

$$s(n) = \lceil Q_1 x(n) \rfloor, \tag{6}$$

where $\lceil \cdot \rfloor$ is the rounding function, and Q_1 is a suitable scaling factor and then encrypting the modulo N representation of $s(n)$, that is, $\mathcal{E}[s(n)] \triangleq \mathcal{E}[s(n) \bmod N]$ (for the sake of brevity, we omit the random parameter r).

As long as $s(n)$ does not exceed the size of N—that is, the difference between the maximum and minimum values of $s(n)$ is less than N—its value can be represented in Z_N without loss of information. If we assume $|s(n)| < N/2$, then the original value $x(n)$ can be approximated from $\mathcal{E}[s(n)]$ as

$$\hat{x}(n) = \begin{cases} \dfrac{\mathcal{D}[\mathcal{E}[s(n)]]}{Q_1}, & \text{if } \mathcal{D}[\mathcal{E}[s(n)]] < \dfrac{N}{2}, \\ \dfrac{\mathcal{D}[\mathcal{E}[s(n)]] - N}{Q_1}, & \text{if } \mathcal{D}[\mathcal{E}[s(n)]] > \dfrac{N}{2}. \end{cases} \tag{7}$$

The above representation can be used to define an integer approximation of the DCT. Let us consider the scaled DCT of type II (DCT-II) of $x(n)$, defined as

$$X(k) = \sum_{n=0}^{M-1} x(n) \cos \frac{\pi(2n+1)k}{2M}, \quad k = 0, 1, \ldots, M-1. \tag{8}$$

The corresponding integer DCT of type II is defined as [6]

$$S(k) = \sum_{n=0}^{M-1} C_M^{II}(n, k) s(n), \quad k = 0, \ldots, M-1, \tag{9}$$

where $C_M^{II}(n, k) = \lceil Q_2 \cos(\pi(2n+1)k/2M) \rfloor$ and Q_2 is a suitable scaling factor for the cosine values.

A similar approach leads to the definition of the integer inverse DCT (IDCT). The scaled IDCT, also referred to as scaled DCT of type III, is defined as

$$x(n) = \sum_{k=0}^{M-1} c(k) X(k) \cos \frac{\pi(2n+1)k}{2M}, \quad n = 0, 1, \ldots, M-1. \tag{10}$$

where

$$c(k) = \begin{cases} \dfrac{1}{2}, & \text{if } k = 0, \\ 1, & \text{if } k \neq 0 \end{cases} \tag{11}$$

The integer IDCT or integer DCT of type III can be defined as in (9) by using in place of $C_M^{II}(n, k)$ the following integer coefficients:

$$C_M^{III}(n, k) = \begin{cases} \left\lceil \dfrac{Q_2}{2} \right\rfloor, & \text{if } n = 0 \\ \left\lceil Q_2 \cos \dfrac{\pi(2k+1)n}{2M} \right\rfloor, & \text{if } n \neq 0. \end{cases} \tag{12}$$

4. s.p.e.d. DCT

Since all computations are between integers and there is no scaling, the expression in (9) can be evaluated in the encrypted domain by relying on the homomorphic properties. For instance, if the inputs are encrypted with the Paillier cryptosystem, the s.p.e.d DCT is

$$\mathcal{E}[S(k)] = \prod_{n=0}^{M-1} \mathcal{E}[s(n)]^{C_M^{II}(n,k)}, \quad k = 0, \ldots, M-1, \tag{13}$$

where all computations are done modulo N^2 [7].

The computation of the DCT using (9) 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 $S(k) \bmod N$. A solution is to find an *upper bound* on $S(k)$ such that $|S(k)| \le Q_S$ and verify that $N > 2Q_S$. We will show that $S(k)$ can be expressed in general as

$$S(k) = KX(k) + \epsilon_S(k), \tag{14}$$

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 $\widetilde{X}(k) = S(k)/K$, and the upper bound is

$$Q_S = MK + \epsilon_{S,U}, \tag{15}$$

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. In the following, we will add to Q_S, K, and $\epsilon_{S,U}$ the additional subscripts D and F to denote direct and fast DCT, respectively whereas the superscript $2D$ will denote the 2-dimensional versions.

4.1. Direct Computation. Let us express $s(n) = Q_1 x(n) + \epsilon_s(n)$ and $C_M^{II}(n,k) = Q_2 \cos(\pi(2n+1)k/2M) + \epsilon_C(n,k)$. If the DCT is directly computed by applying (9), then we have

$$S(k) = Q_1 Q_2 X(k) + \epsilon_S(k), \tag{16}$$

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

$$|\epsilon_S(k)| \le M\left(\frac{Q_1}{2} + \frac{Q_2}{2} + \frac{1}{4}\right) = \epsilon_{S,U,D} \tag{17}$$

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

4.2. Fast DCT. In order to obtain an s.p.e.d. version of the fast DCT, we will refer to the recursive matrix representation in [17]. Given $[\mathbf{T}_M^{II}]_{nk} = \cos(\pi(2n+1)k/2M)$, we have

$$\begin{aligned} \mathbf{T}_M^{II} &= \mathbf{P}_M \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{L}_{M/2} \end{bmatrix} \begin{bmatrix} \mathbf{T}_{M/2}^{II} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{M/2}^{II} \end{bmatrix} \\ &\quad \times \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{M/2} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{J}_{M/2} \\ \mathbf{I}_{M/2} & -\mathbf{J}_{M/2} \end{bmatrix} \\ &= \mathbf{A}_M \begin{bmatrix} \mathbf{T}_{M/2}^{II} & \mathbf{0} \\ \mathbf{0} & \mathbf{T}_{M/2}^{II} \end{bmatrix} \begin{bmatrix} \mathbf{I}_{M/2} & \mathbf{0} \\ \mathbf{0} & \mathbf{D}_{M/2} \end{bmatrix} \mathbf{B}_M, \end{aligned} \tag{18}$$

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

$$\mathbf{L}_{M/2} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 & 0 \\ -1 & 2 & 0 & & 0 & 0 \\ 1 & -2 & 2 & & 0 & 0 \\ \vdots & & & \ddots & & \vdots \\ 1 & -2 & 2 & & 2 & 0 \\ -1 & 2 & -2 & \cdots & -2 & 2 \end{bmatrix}, \tag{19}$$

$\mathbf{J}_M$ is obtained by the $M \times M$ identity matrix by reversing the column order, and $\mathbf{P}_M$ is a permutation matrix given as

$$\mathbf{P}_M = \begin{bmatrix} 1 & 0 & \cdots & 0 & 0 & 0 & \cdots & 0 \\ 0 & 0 & & 0 & 1 & 0 & & 0 \\ 0 & 1 & & 0 & 0 & 0 & & 0 \\ 0 & 0 & & 0 & 0 & 1 & & 0 \\ \vdots & & \ddots & \vdots & \vdots & & \ddots & \vdots \\ 0 & 0 & & 1 & 0 & 0 & & 0 \\ 0 & 0 & \cdots & 0 & 0 & 0 & \cdots & 1 \end{bmatrix}. \tag{20}$$

Since the only noninteger matrix in (18) is $\mathbf{D}_{M/2}$, the corresponding s.p.e.d. structure can be recursively defined as

$$\mathbf{C}_M^{II} = \mathbf{A}_M \begin{bmatrix} \mathbf{C}_{M/2}^{II} & \mathbf{0} \\ \mathbf{0} & \mathbf{C}_{M/2}^{II} \end{bmatrix} \begin{bmatrix} Q_2 \mathbf{I}_{M/2} & \mathbf{0} \\ \mathbf{0} & \widetilde{\mathbf{D}}_{M/2} \end{bmatrix} \mathbf{B}_M, \tag{21}$$

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

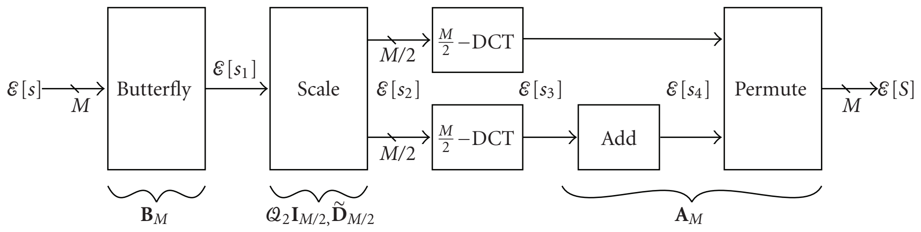


Figure 1: Block diagram of s.p.e.d. fast DCT.

The s.p.e.d. fast DCT can be implemented according to the block diagram in Figure 1. If we define $[\mathbf{s}]_k = s(k)$, $[\mathbf{S}]_k = S(k)$, and we denote as $[\mathbf{s}_i]_k = s_i(k)$, $i = 1, \dots, 4$, the results of the intermediate computations in one recursion of the s.p.e.d. fast DCT structure, the different blocks can be defined as follows. The butterfly block performs the following s.p.e.d. computations

$$\mathcal{E}[s_1(k)] = \begin{cases} \mathcal{E}[s(k)] \cdot \mathcal{E}[s(M-1-k)], & 0 \le k < \dfrac{M}{2}, \\ \mathcal{E}\left[s\left(k - \dfrac{M}{2}\right)\right] \cdot \mathcal{E}\left[s\left(\dfrac{3M}{2} - 1 - k\right)\right]^{-1}, & \dfrac{M}{2} \le k < M, \end{cases} \tag{22}$$

whereas the scale block can be defined as

$$\mathcal{E}[s_2(k)] = \begin{cases} \mathcal{E}[s_1(k)]^{Q_2}, & 0 \le k < \dfrac{M}{2}, \\ \mathcal{E}[s_1(k)]^{\widetilde{D}_{M/2}(k-M/2)}, & \dfrac{M}{2} \le k < M, \end{cases} \tag{23}$$

where $\widetilde{D}_{M/2}(k)$ is the kth element on the diagonal of $\widetilde{\mathbf{D}}_{M/2}$. The output of the scale block is split in two halves, which are recursively processed by two half size fast DCT. The lower half is further processed by the add block, which can be defined as

$$\mathcal{E}\left[s_4\left(\frac{M}{2}\right)\right] = \mathcal{E}\left[s_3\left(\frac{M}{2}\right)\right], \tag{24}$$

$$\mathcal{E}[s_4(k)] = \mathcal{E}[s_3(k)]^2 \cdot \mathcal{E}[s_3(k-1)]^{-1}, \quad \frac{M}{2} + 1 \le k < M. \tag{25}$$

Lastly, the two halves are combined and permuted according to $\mathbf{P}_M$ in order to yield the DCT outputs in the right order.

As to the upper bound analysis, let us consider the mth stage of the recursion and express the quantized matrices as $\widetilde{\mathbf{D}}_{2^m} = Q_2\mathbf{D}_{2^m} + \mathbf{E}_D^{(m)}$ and $\mathbf{C}_{2^m}^{II} = K^{(m)}\mathbf{T}_{2^m}^{II} + \mathbf{E}_T^{(m)}$, where $\mathbf{E}_D^{(m)}$ and $\mathbf{E}_T^{(m)}$ denote the quantization errors on the corresponding matrix entries. We can rewrite (21) as

$$\begin{aligned} \mathbf{C}_{2^{m+1}}^{II} &= \mathbf{A}_{2^{m+1}} \begin{bmatrix} K^{(m)}\mathbf{T}_{2^m}^{II} + \mathbf{E}_T^{(m)} & \mathbf{0} \\ \mathbf{0} & K^{(m)}\mathbf{T}_{2^m}^{II} + \mathbf{E}_T^{(m)} \end{bmatrix} \\ &\quad \times \begin{bmatrix} Q_2\mathbf{I}_{2^m} & \mathbf{0} \\ \mathbf{0} & Q_2\mathbf{D}_{2^m} + \mathbf{E}_D^{(m)} \end{bmatrix} \mathbf{B}_{2^{m+1}} \\ &= K^{(m)}Q_2\mathbf{T}_{2^{m+1}}^{II} \\ &\quad + \mathbf{A}_{2^{m+1}} \left\{ \begin{bmatrix} K^{(m)}\mathbf{T}_{2^m}^{II} & \mathbf{0} \\ \mathbf{0} & K^{(m)}\mathbf{T}_{2^m}^{II} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_D^{(m)} \end{bmatrix} \right. \\ &\quad + \begin{bmatrix} \mathbf{E}_T^{(m)} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_T^{(m)} \end{bmatrix} \begin{bmatrix} \mathbf{0} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_D^{(m)} \end{bmatrix} \\ &\quad \left. + \begin{bmatrix} \mathbf{E}_T^{(m)} & \mathbf{0} \\ \mathbf{0} & \mathbf{E}_T^{(m)} \end{bmatrix} \begin{bmatrix} Q_2\mathbf{I}_{2^m} & \mathbf{0} \\ \mathbf{0} & Q_2\mathbf{D}_{2^m} \end{bmatrix} \right\} \mathbf{B}_{2^{m+1}} \\ &= K^{(m+1)}\mathbf{T}_{2^{m+1}}^{II} + \mathbf{E}_T^{(m+1)}. \end{aligned} \tag{26}$$

From the previous 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 $\mathbf{s} = [s(0), s(1), \dots, s(M-1)]^T$. 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

$$\mathbf{C}_{2^\nu}^{II}\mathbf{s} = K^{(\nu)}Q_1\mathbf{T}_{2^\nu}^{II}\mathbf{x} + K^{(\nu)}\mathbf{T}_{2^\nu}^{II}\mathbf{e}_s + \mathbf{E}_T^{(\nu)}Q_1\mathbf{x} + \mathbf{E}_T^{(\nu)}\mathbf{e}_s. \tag{27}$$

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 $|\epsilon_S(k)| \le MK^{(\nu)}/2 + (Q_1 + 1/2)\|\mathbf{E}_T^{(\nu)}\|_\infty$, where $\|\cdot\|_\infty$ denotes the maximum absolute row sum norm of a matrix. Based on (26), we can give an equivalent recursive relation on $\|\mathbf{E}_T^{(m)}\|_\infty$ as

$$\left\|\mathbf{E}_T^{(m+1)}\right\|_\infty \le (2^{m+1} - 1)\left[2^m K^{(m)} + \left\|\mathbf{E}_T^{(m)}\right\|_\infty (2Q_2 + 1)\right], \tag{28}$$

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

$$\left\|\mathbf{E}_T^{(\nu)}\right\|_\infty \le \sum_{k=0}^{\nu-1}(2Q_2+1)^k 2^{\nu-k} Q_2^{\nu-k} \prod_{r=\nu-k}^{\nu}\left(2^{r+1}-1\right) = \epsilon_{E,U} \tag{29}$$

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

$$|\epsilon_S(k)| \le \frac{MQ_2^\nu}{2} + \left(Q_1 + \frac{1}{2}\right)\epsilon_{E,U} = \epsilon_{S,U,F}. \tag{30}$$

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

The above analysis can be extended also to the fast IDCT. It suffices to consider $[\mathbf{T}_M^{III}]_{nk} = \cos(\pi(2k+1)n/2M)$ and, thanks to $\mathbf{T}_M^{III} = (\mathbf{T}_M^{II})^T$,

$$\begin{aligned}\mathbf{T}_M^{III} &= \begin{bmatrix}\mathbf{I}_{M/2} & \mathbf{I}_{M/2}\\ \mathbf{J}_{M/2} & -\mathbf{J}_{M/2}\end{bmatrix}\begin{bmatrix}\mathbf{I}_{M/2} & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_{M/2}\end{bmatrix}\begin{bmatrix}\mathbf{T}_{M/2}^{III} & \mathbf{0}\\ \mathbf{0} & \mathbf{T}_{M/2}^{III}\end{bmatrix}\\ &\quad\times\begin{bmatrix}\mathbf{I}_{M/2} & \mathbf{0}\\ \mathbf{0} & \mathbf{L}_{M/2}^T\end{bmatrix}\mathbf{P}_M^T\\ &= \mathbf{B}_M^T\begin{bmatrix}\mathbf{I}_{M/2} & \mathbf{0}\\ \mathbf{0} & \mathbf{D}_{M/2}\end{bmatrix}\begin{bmatrix}\mathbf{T}_{M/2}^{III} & \mathbf{0}\\ \mathbf{0} & \mathbf{T}_{M/2}^{III}\end{bmatrix}\mathbf{A}_M^T.\end{aligned} \tag{31}$$

It is easy to show that the model in (27) can be applied also to the integer IDCT, so that the upper bound in (30) holds for the IDCT as well.

4.3. 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 (14). 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 $|S(k)| \le MK + \epsilon_{S,U}$ and $|\epsilon_S(k)| \le \epsilon_{S,U}$.

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

$$K_D^{2D} = Q_2K_D = Q_2^2Q_1, \tag{32}$$

$$\epsilon_{S,U,D}^{2D} = M\left(\frac{MK_D}{2} + Q_2\epsilon_{S,U,D} + \frac{\epsilon_{S,U,D}}{2}\right), \tag{33}$$

$$Q_{S,D}^{2D} = M^2K_D^{2D} + \epsilon_{S,U,D}^{2D}, \tag{34}$$

whereas in the case of the fast DCT we obtain

$$K_F^{2D} = Q_2^\nu K_F = Q_2^{2\nu}Q_1, \tag{35}$$

$$\epsilon_{S,U,F}^{2D} = MQ_2^\nu\epsilon_{S,U,F} + (MK_F + \epsilon_{S,U,F})\epsilon_{E,U}, \tag{36}$$

$$Q_{S,F}^{2D} = M^2K_F^{2D} + \epsilon_{S,U,F}^{2D}. \tag{37}$$

In the case of nonseparable processing, the upper bound on the output of the s.p.e.d. DCT can be derived in the same way as in the one-dimensional case. For instance, a direct nonseparable 2D-DCT will lead to the same upper bound as in (17). Even if this will reduce the upper bound with respect to the separable case, a nonseparable implementation will have a greater complexity. In the following, only the separable case will be considered.

4.4. Security. Concerning the security of the s.p.e.d. DCT, if we work with a semantically secure cryptosystem, the security is automatically achieved that is, the output of the s.p.e.d. DCT does not reveal anything about the DCT inputs. Under the assumption that deciding N-residuosity classes in $\mathbb{Z}_{N^2}^*$ is hard, that is, given $w \in \mathbb{Z}_{N^2}^*$ it is not possible to decide in polynomial time whether w is an N-residue or not, the Paillier cryptosystem can be proved to be semantically secure [7]. If the assumption is relaxed to the hardness of computing N-residuosity classes, the security of the plaintext bits of a Paillier encryption, and hence of the proposed scheme, depends on the knowledge of the size of the plaintext. The interested reader can find a discussion on such topics in [18].

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. Firstly, 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. However, the size of the encrypted word, that is, N, is fixed by the security requirements. Minimum security requirements for the Paillier cryptosystem impose the use of at least 1024 bits for N. This means that, irrespective of the size of the plaintext pixels, each encrypted pixel will be represented as an encrypted word of at least 1024 bits. The result is that the outputs of the block-based s.p.e.d. DCT will be far from exploiting the full bandwidth of the modulus N. Secondly, each block undergoes exactly the same processing. Hence, this could permit 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 an s.p.e.d. block DCT (BDCT) based on a *composite* representation of the input pixels [19]. For the sake of simplicity, we can assume the image as a one-dimensional signal, since the extension to the 2D case is straightforward using separable processing. Moreover, let us assume that the input pixel values have been quantized as in Section 3, that is, they satisfy the relation $|s(n)| \le Q_1$.

We define the *composite* representation of $s(n)$ of order R and base B as

$$s_C(k) = \sum_{i=0}^{R-1} s_i(k)B^i, \quad k = 0, 1, \ldots, M-1, \tag{38}$$

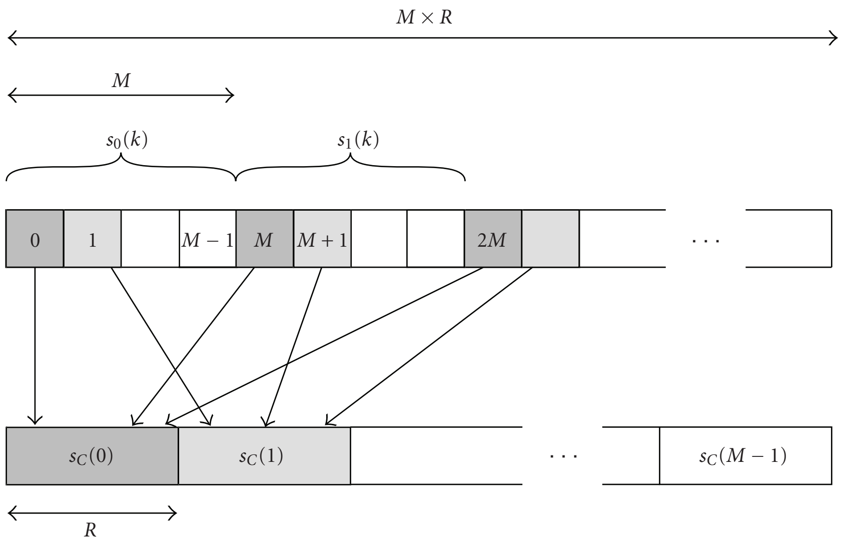


FIGURE 2: Graphical representation of an M-polyphase composite representation having order R. The values inside the small boxes indicate the indexes of the samples of $s(n)$. Identically shaded boxes indicate values belonging to the same composite word.

where $s_i(k)$, $i = 0, 1, \ldots, R-1$, indicate R disjoint subsequences of the image pixels $s(n)$.

The kth element of the composite signal $s_C(k)$ represents a word where we can pack R samples of the original signal, chosen by partitioning the original signal samples $s(n)$ into M sets of R samples each. In the following, we will consider the so-called M-polyphase composite representation (M-PCR), where the partitioning of $s(n)$ is given by $s_i(k) = s(iM + k)$. As shown in Figure 2, in this representation each composite word contains R samples which are spaced M samples apart in the original sequence, that is, belonging to one of the Mth order polyphase components of signal $s(n)$.

For the *composite* representation, the following theorem is valid.

Theorem 1. *Let us assume that*

$$|s(n)| < Q_1 \quad \forall n, \tag{39}$$

$$B > 2Q_1, \tag{40}$$

$$B^R \le N, \tag{41}$$

where N is a positive integer, and let $s_C(k)$ *be defined as in* (38)*. Then, the following holds:*

$$0 \le s_C(k) + \omega_Q < N, \tag{42}$$

where $\omega_Q = Q_1 \sum_{i=0}^{R-1} B^i = Q_1((B^R - 1)/(B-1))$*. Moreover, the original pixels can be obtained from the composite representation as*

$$s_i(k) = \left\{ \left[(s_C(k) + \omega_Q) \div B^i \right] \bmod B \right\} - Q_1. \tag{43}$$

Proof. let us express

$$s_C(k) + \omega_Q = \sum_{j=0}^{R-1} \left[s_j(k) + Q_1 \right] B^j. \tag{44}$$

Thanks to (39) and (40), we have $0 \le s_j(k) + Q_1 \le 2Q_1 \le B-1$. Hence, $s_C(k) + \omega_Q$ can be considered as a positive base-B integer whose digits are given by $s_j(k) + Q_1$. Moreover, since $s_C(k) + \omega_Q$ has R digits, it is bounded by

$$s_C(k) + \omega_Q \le \sum_{j=0}^{R-1} (B-1) B^j = B^R - 1 < N, \tag{45}$$

where the last inequality comes from (41). As to the second part of the theorem, for each i we have

$$s_C(k) + \omega_Q = B^i \sum_{j=i}^{R-1} \left[s_j(k) + Q_1 \right] B^{j-i} + \sum_{j=0}^{i-1} \left[s_j(k) + Q_1 \right] B^j. \tag{46}$$

Thanks to the properties of $s_j(k) + Q_1$, we have $\sum_{j=0}^{i-1} [s_j(k) + Q_1] B^j \le B^i - 1$. Hence

$$\begin{aligned} \left[s_C(k) + \omega_Q \right] \div B^i &= \sum_{j=i}^{R-1} \left[s_j(k) + Q_1 \right] B^{j-i} \\ &= B \sum_{j=i+1}^{R-1} \left[s_j(k) + Q_1 \right] B^{j-i-1} + s_i(k) + Q_1 \end{aligned} \tag{47}$$

from which (43) follows hence completing the proof. □

When dealing with encrypted data, the first part of the previous theorem demonstrates that the composite representation can be safely encrypted by using a homomorphic cryptosystem defined on modulo N arithmetic: as long as the hypotheses of the theorem hold, the composite data $s_C(n)$ takes no more than N distinct values, so the values of the composite signal can be represented modulo N without loss of information. (i.e., it is possible to define a one-to-one mapping between $s_C(n)$ and $[s_0(n), s_1(n), \ldots, s_{R-1}(n)]$.)

We propose now an s.p.e.d. block DCT (BDCT) based on the composite representation of the input pixels. Let us consider R distinct blocks of an image, assumed as one-dimensional, having size M. Let us define the block bandwidth as $B = \lfloor \sqrt[R]{N} \rfloor$. Moreover, let us assume that the input pixel values $s(n)$ have been quantized.

The blockwise DCT can be defined as

$$u_i(r) = \sum_{n=0}^{M-1} C_M^{II}(n,r)s(iM+n) \quad r = 0,1,\ldots,M-1. \quad (48)$$

Since the transform has a repeated structure, it is suitable for a parallel implementation. If the pixels having the same position within each block are packed in a single word according to the M-PCR representation into $s_C(k)$, as in (38), we can define the equivalent parallel blockwise DCT as

$$u_C(r) = \sum_{k=0}^{M-1} C_M^{II}(k,r)s_C(k), \quad r = 0,1,\ldots,M-1. \quad (49)$$

Proposition 1. *If $B > 2Q_S$, then $u_i(r)$, $i = 0,1,\ldots,R-1$, can be exactly computed from the modulo N representation of $u_C(r)$.*

Proof. let us consider the following equalities:

$$\begin{aligned} u_C(r) &= \sum_{k=0}^{M-1} C_M^{II}(k,r)\sum_{i=0}^{R-1} s(iM+k)B^i \\ &= \sum_{i=0}^{R-1}\left[\sum_{k=0}^{M-1} C_M^{II}(k,r)s(iM+k)\right]B^i \quad (50) \\ &= \sum_{i=0}^{R-1} u_i(r)B^i. \end{aligned}$$

Then, it suffices to note that $|u_i(r)| \leq Q_S$ and replace Q_1 with Q_S in the proof of Theorem 1. □

By exploiting the composite representation, we can process R blocks by using a single s.p.e.d. DCT. This means that the complexity of the s.p.e.d. BDCT is reduced by a factor R with respect to that of a pixelwise implementation, since the size of the encrypted values will be the same irrespective of the implementation. Moreover, the bandwidth usage is also reduced by the same factor, since we pack R pixels into a single ciphertext.

Finally, we would like to point out that the fast DCT algorithm can be used for the BDCT as well. The fast BDCT algorithm is simply obtained by computing the fast DCT of the composite signal $s_C(n)$. In order to verify that the above algorithm is correct, it suffices to substitute $C(n,k)$ in (49) with the (n,k) element of the matrix $\mathbf{C}_M^{II}$ as defined in (21).

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).

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**

Since the values of Q_S in (34)–(37) 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 $Q_{S,Z}^{2D} < 2M^2K_Z^{2D}$, this can be expressed as

$$\left\lceil \log_2 Q_{S,Z}^{2D} \right\rceil + 1 < 2\nu + \left\lceil \log_2 K_Z^{2D} \right\rceil + 2 = n_{U,Z}, \quad (51)$$

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 cannot 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

$$R_{max} = \left\lfloor \frac{\log_2 N}{\log_2 \lceil 2Q_{S,Z} \rceil} \right\rfloor \approx \left\lfloor \frac{\lfloor \log_2 N \rfloor}{\log_2 \lceil 2Q_{S,Z} \rceil} \right\rfloor = R_{U,Z}. \quad (52)$$

In Table 2, we give some values of $R_{U,Z}$ considering DCT sizes ranging from 4×4 to 64×64 and different precisions. Specifically, $R_{U,D}$ indicates the value of $R_{U,Z}$ obtained with a direct implementation of the DCT, while $R_{U,F}$ indicates the corresponding value for a fast implementation of DCT. The results demonstrate that the composite representation permits to significantly reduce both the bandwidth requirements and the complexity, especially for the fixed point case.

TABLE 2: Upper bounds on the number of blocks R that can be processed in parallel by an s.p.e.d. $M \times M$ 2D-BDCT. $Z = \{D, F\}$ indicates a direct or a fast implementation of the DCT. $Q_2 = 2^{15}$ 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. We have assumed $\lfloor log_2 N \rfloor = 1023$.

	$Q_2 = 2^{15}$		$Q_2 = 2^{36}$		$Q_2 = 2^{65}$	
M	$R_{U,D}$	$R_{U,F}$	$R_{U,D}$	$R_{U,F}$	$R_{U,D}$	$R_{U,F}$
4	24	12	12	6	7	3
8	23	8	11	4	7	2
16	22	6	11	3	7	1
32	21	4	11	2	6	1
64	20	4	11	2	6	1

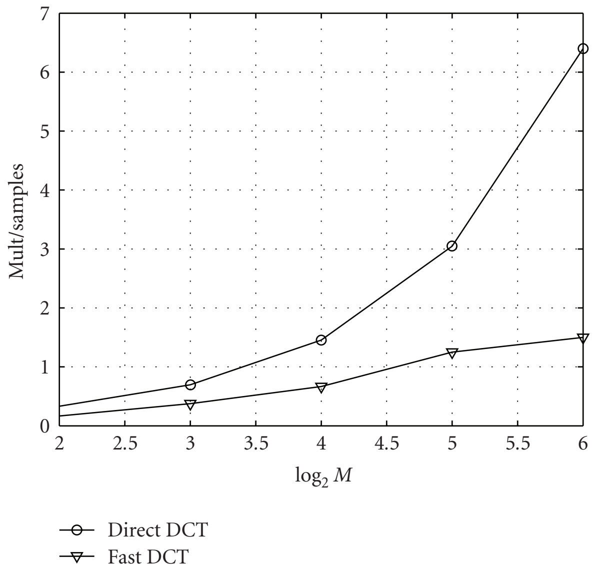
It is worth noting that a direct implementation allows to increase $R_{U,Z}$ up to seven times with respect to the fast BDCT. Since the BDCT usually works with small sized blocks, the complexity of the direct implementation will not be much higher than that of the fast implementation. To give some figures, let us consider the number of multiplications per sample required by the different implementations. The complexity of a direct M-point DCT is M^2 multiplications: if we consider a separable implementation, an $M \times M$ DCT will require $2M$ M-point DCTs to compute M^2 output samples. Since a BDCT can compute $R_{U,D}$ DCTs in parallel, this results in a complexity of

$$\mathcal{C}_D = \frac{2M}{R_{U,D}} \text{ mult/sample.} \tag{53}$$

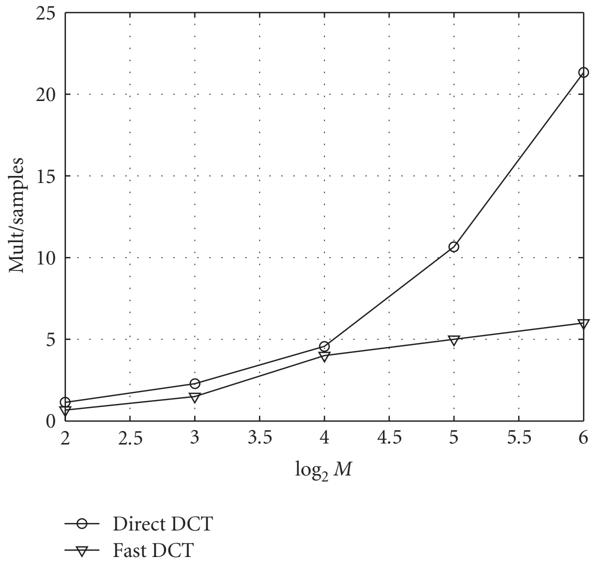
As to the fast M-point DCT, the complexity is $(M/2)\log_2 M$ multiplications [20]. By using similar arguments, the complexity of a fast BDCT implementation can be then evaluated as

$$\mathcal{C}_F = \frac{\log_2 M}{R_{U,F}} \text{ mult/sample.} \tag{54}$$

In Figure 3, we compare the complexity of direct and fast BDCT for two different precisions. The complexity of the fast BDCT is always below that of the direct implementation. However, it is worth noting that for small BDCT sizes, for example, up to 16×16, the complexity of the direct implementation is only slightly larger than that of the fast implementation. 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 the price of a very small increase of complexity.



(a)



(b)

FIGURE 3: Complexity of direct BDCT versus fast BDCT. 8-bit input values ($Q = 2^7$) have been assumed. We have assumed $\lfloor \log_2 N \rfloor = 1023$. (a) $Q_T = 2^{15}$; (b) $Q_T = 2^{65}$.

7. Implementation Case Study

The feasibility of the s.p.e.d. DCT in a practical scenario is verified by considering its use in a buyer-seller watermarking protocol. Namely, we consider the secure embedding of a watermark as described in [8, 21]. In this scenario, a seller receives the bits of the watermark encrypted with the public key of a buyer—the output of a previous protocol between him and the buyer—and embeds them into a set of features extracted from the digital content he owns. The output of this procedure is a set of watermarked and encrypted features that are sent to the buyer. In the following, such a protocol will be referred to as secure watermark embedding (SWE).

In our case study, we assume that the content is an image and that the features are obtained by applying a block 2D-DCT to the pixel values. We also assume that the seller wants to perform the inverse DCT (IDCT) of the watermarked features in the encrypted domain, before sending them to the buyer. This can be justified by his wish to keep the actual transform secret, so as to expose as little details as possible regarding the watermarking algorithm. Another reason for

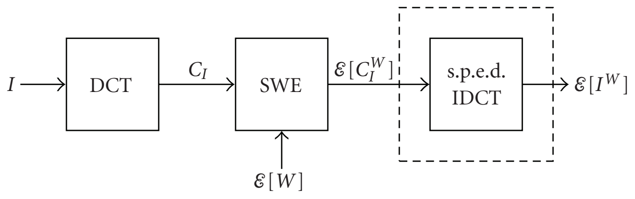


Figure 4: Secure watermark embedding scenario.

Table 3: Execution times (in seconds) of the different implementations. The row labeled as "packing" refers to the conversion from encrypted samplewise representation to encrypted composite representation. The row labeled as "DCT" refers to the actual DCT computation.

256	IDCT	F-IDCT	B-IDCT	BF-IDCT
packing	—	—	20.6	51.2
DCT	164.2	79.8	7.2	10
total	**164.2**	**79.8**	**27.8**	**61.2**
512	IDCT	F-IDCT	B-IDCT	BF-IDCT
packing	—	—	83.8	208.2
DCT	663.2	319.7	29	40
total	**663.2**	**319.7**	**112.8**	**248.2**
1024	IDCT	F-IDCT	B-IDCT	BF-IDCT
packing	—	—	334.4	850
DCT	2647.7	1277.1	115.2	159.6
total	**2647.7**	**1277.1**	**449.6**	**1009.6**

using the s.p.e.d. IDCT is the possibility of applying some postprocessing to the watermarked image before distributing it. Common postprocessing steps are the use of a perceptual mask [22] or the insertion of a synchronization pattern [23].

The scheme we consider is summarized in Figure 4. The image is divided into square blocks of 8×8 pixels and an 8×8 (I)DCT is applied to each block. We will assume that the plaintext DCT and SWE building blocks are already available and we will concentrate on the implementation of the s.p.e.d. IDCT block. Two different implementations are considered: a separable direct IDCT as described in Section 4.1; a separable fast IDCT as described in Section 4.2. As to the data representation, both a pixelwise/coefficientwise representation and a composite representation as described in Section 5 are considered. The combination of the former choices results in four alternative s.p.e.d. implementations: pixelwise direct IDCT (IDCT), pixelwise fast IDCT (F-IDCT), composite (block) direct IDCT (B-IDCT), and composite (block) fast IDCT (BF-IDCT).

The aforementioned versions have been implemented in C++ using the GNU Multi-Precision (GMP) library [24] and the NTL library [25], which provide software optimized routines for the processing of integers having arbitrary length. All versions have been run on an Intel(R) Core(TM)2 Quad CPU at 2.40 GHz, used as a single processor. In order to verify the feasibility of the s.p.e.d. approach, we measured the execution times of the four versions using three different image sizes: 256×256, 512×512, and 1024×1024. In all tests, the marked features are represented as 8-bit integers ($Q_1 = 2^7$) and the cosine values are quantized as 16-bit integers ($Q_2 = 2^{15}$). The image features are encrypted with the Paillier's cryptosystem, using a modulo N of 1024 bits.

The correctness of the s.p.e.d. DCT implementation has been verified by comparing its output with the output of an analogous plaintext DCT implementation, as well as by verifying the amount of error introduced after the application of a standard plaintext DCT followed by an encrypted domain IDCT. With the used precision, the normalized MSE after the DCT-IDCT chain was on the order of $3 \cdot 10^{-3}$. As to block DCT, its correctness has been verified by checking that the output of B-(I)DCT, after decryption and unpacking, was identical to the output of the corresponding (I)DCT.

The execution times are reported in Table 3. From the comparison between the pixelwise representation and the composite representation, it is evident that the latter permits to sensibly reduce the computational complexity of an s.p.e.d. DCT. Interestingly, the B-IDCT proves slightly more efficient than the BF-IDCT, confirming that the direct DCT implementation may be preferable when combined with the composite representation. In the considered scenario, we assume that the inputs to the s.p.e.d. DCT are encrypted samplewise. Hence, both B-IDCT and BF-IDCT require the conversion from an encrypted samplewise representation to an encrypted composite representation. Such a conversion can be done thanks to the homomorphic properties of the cryptosystem:

$$\mathcal{E}[s_C(k)] = \prod_{i=0}^{R-1} \mathcal{E}[s_i(k)]^{B^i}, \quad k = 0, 1, \ldots, M-1. \qquad (55)$$

From Table 3, we can notice that the time required by this conversion is greater than the time required to perform an s.p.e.d. DCT. Since the overall computational complexity of B-IDCT and BF-IDCT is given as the sum of both times, this reduces the performance gain achievable by the composite representation. Namely, B-IDCT is about three times faster than F-IDCT; whereas BF-IDCT is only slightly faster than F-IDCT.

8. Concluding Remarks

We have considered the implementation of the DCT on an encrypted image by 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 an 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.

To evaluate the proposed solutions, we have considered the application of the s.p.e.d. 2D-DCT and 2D-BDCT to 8-bit greyscale images, computing the upper bound on the number of bits required in order to correctly represent the DCT outputs, and, for the s.p.e.d. 2D-BDCT, the number of pixels that can be safely packed into a single word. The results demonstrate that 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 the price of a very small increase of complexity.

The feasibility of the application of the s.p.e.d. DCT in a practical buyer-seller watermarking protocol has been also verified, providing promising results. Future research will be devoted to the design and implementation of the complete buyer-seller watermarking protocol.

Acknowledgments

The work described in this paper has been partially supported by the European Commission through the IST Programme under Contract no 034238-SPEED and by the Italian Research Project (PRIN 2007): "Privacy aware processing of encrypted signals for treating sensitive information." 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.

References

[1] A. Piva and S. Katzenbeisser, "Signal processing in the encrypted domain," *EURASIP Journal on Information Security*, vol. 2007, Article ID 82790, 1 pages, 2007.

[2] Z. Erkin, A. Piva, S. Katzenbeisser, et al., "Protection and retrieval of encrypted multimedia content: when cryptography meets signal processing," *EURASIP Journal on Information Security*, vol. 2007, Article ID 78943, 20 pages, 2007.

[3] N. Memon and P. W. Wong, "A buyer-seller watermarking protocol," *IEEE Transactions on Image Processing*, vol. 10, no. 4, pp. 643–649, 2001.

[4] J. Shashank, P. Kowshik, K. Srinathan, and C. V. Jawahar, "Private content based image retrieval," in *Proceedings of the 26th IEEE Conference on Computer Vision and Pattern Recognition (CVPR '08)*, pp. 1–8, June 2008.

[5] T. Bianchi, A. Piva, and M. Barni, "Implementing the discrete Fourier transform in the encrypted domain," in *Proceedings of IEEE International Conference on Acoustics, Speech, and Signal Processing (ICASSP '08)*, pp. 1757–1760, Las Vegas, Nev, USA, March-April 2008.

[6] T. Bianchi, A. Piva, and M. Barni, "On the implementation of the discrete Fourier transform in the encrypted domain," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 1, pp. 86–97, 2009.

[7] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology*, vol. 1592 of *Lecture Notes in Computer Science*, pp. 223–238, Springer, New York, NY, USA, 1999.

[8] M. Kuribayashi and H. Tanaka, "Fingerprinting protocol for images based on additive homomorphic property," *IEEE Transactions on Image Processing*, vol. 14, no. 12, pp. 2129–2139, 2005.

[9] A. Adelsbach, S. Katzenbeisser, and A.-R. Sadeghi, "Watermark detection with zero-knowledge disclosure," *Multimedia Systems*, vol. 9, no. 3, pp. 266–278, 2003.

[10] B. Goethals, S. Laur, H. Lipmaa, and T. Mielikäinen, "On private scalar product computation for privacy-preserving data mining," in *Proceedings of the 7th International Conference on Information Security and Cryptology (ICISC '04)*, vol. 3506 of *Lecture Notes in Computer Science*, pp. 104–120, 2004.

[11] A. C. Yao, "Protocols for secure computations," in *Proceedings of the 23rd IEEE Symposium on Foundations of Computer Science*, pp. 160–164, Chicago, Ill, USA, November 1982.

[12] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," in *Proceedings of the International Conference on the Theory and Application of Cryptographic Techniques (EUROCRYPT '01)*, vol. 2045 of *Lecture Notes in Computer Science*, pp. 280–299, Springer, London, UK, 2001.

[13] R. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of Secure Computation*, R. A. DeMillo, R. J. Lipton, D. P. Dobkin, and A. K. Jones, Eds., pp. 169–179, Academic Press, New York, NY, USA, 1978.

[14] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *Proceedings of the 41st Annual ACM Symposium on Theory of Computing (STOC '09)*, pp. 169–178, Bethesda, Md, USA, 2009.

[15] S. Goldwasser and S. Micali, "Probabilistic encryption," *Journal of Computer and System Sciences*, vol. 28, no. 2, pp. 270–299, 1984.

[16] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of Paillier's probabilistic public-key system," in *Proceedings of the 4th International Workshop on Practice and Theory in Public Key Cryptography*, vol. 1992 of *Lecture Notes In Computer Science*, pp. 119–136, 2001.

[17] Y. Zeng, L. Cheng, G. Bi, and A. C. Kot, "Integer DCTs and fast algorithms," *IEEE Transactions on Signal Processing*, vol. 49, no. 11, pp. 2774–2782, 2001.

[18] D. Catalano, R. Gennaro, and N. Howgrave-Graham, "The bit security of Paillier's encryption scheme and its applications," in *Proceedings of the International Conference on the Theory and Application of Crypto Graphic Techniques (EUROCRYPT '01)*, pp. 229–243, Springer, Innsbruck, Austria, May 2001.

[19] T. Bianchi, A. Piva, and M. Barni, "Efficient pointwise and blockwise encrypted operations," in *Proceedings of the 10th ACM Workshop on Multimedia and Security*, pp. 85–90, Oxford, UK, 2008.

[20] H. Hou, "A fast recursive algorithm for computing the discrete cosine transform," *IEEE Transactions on Acoustics, Speech, and Signal Processing*, vol. 35, no. 10, pp. 1455–1461, 1987.

[21] J. P. Prins, Z. Erkin, and R. L. Lagendijk, "Anonymous fingerprinting with robust QIM watermarking techniques," *EURASIP Journal on Information Security*, vol. 2007, Article ID 31340, 13 pages, 2007.

[22] F. Bartolini, M. Barni, V. Cappellini, and A. Piva, "Mask building for perceptually hiding frequency embedded watermarks," in *Proceedings of the 5th IEEE International Conference on Image Processing (ICIP '98)*, vol. 1, pp. 450–454, Chicago, Ill, USA, October 1998.

[23] P. Moulin and A. Ivanovic, "The Fisher information game for optimal design of synchronization patterns in blind watermarking," in *Proceedings of the 8th IEEE International Conference on Image Processing (ICIP '01)*, vol. 2, pp. 550–553, Thessaloniki, Greece, October 2001.

[24] "GNU Multiple Precision Arithmetic Library," http://gmplib.org/.

[25] "NTL: A library for doing number theory," http://www.shoup.net/ntl/.