On the Implementation of the Discrete Fourier Transform in the Encrypted Domain

Tiziano Bianchi, *Member, IEEE,* Alessandro Piva, *Member, IEEE,* and Mauro Barni, *Senior Member, IEEE*

***Abstract*—Signal processing modules working directly on encrypted data provide an elegant solution to application scenarios where valuable signals must be protected from a malicious processing device. In this paper, we investigate the implementation of the discrete Fourier transform (DFT) in the encrypted domain, by using the homomorphic properties of the underlying cryptosystem. Several important issues are considered for the direct DFT, the radix-2, and the radix-4 fast Fourier algorithms, including the error analysis and the maximum size of the sequence that can be transformed. We also provide computational complexity analyses and comparisons. The results show that the radix-4 FFT is best suited for an encrypted domain implementation in the proposed scenarios.**

I. Introduction

Recent advances in signal processing technology create the opportunity for a large variety of new applications ranging from multimedia content production and distribution, to advanced healthcare systems for continuous health monitoring. These developments raise several important issues concerning the security of the digital contents to be processed, including intellectual property rights management, authenticity, privacy, and conditional access.

Currently available solutions for secure manipulation of signals apply some cryptographic primitives in order to build a secure layer on top of the signal processing modules. An example of this approach is represented by the encryption of compressed multimedia signals: the multimedia content is first of all compressed through a state-of-the-art compression scheme, and next encryption of the compressed bit stream is carried out. Consequently, the bit stream must be decrypted before the content can be decompressed and processed.

These solutions typically assume that the involved parties or devices trust each other, and thus cryptography is used only to protect the data against third parties or to provide authenticity. Unfortunately, this may not be sufficient in some applications, since the owner of the data may not trust the processing devices, or those actors that are required to manipulate them. As a first example, let us consider a situation where a user (say Alice) resorts to a continuous monitoring healthcare system to analyze her medical/biological data in order to get a fast pre-alert diagnosis helping her to stay healthy. Very likely she will not trust the service provider that will be required to analyze Alice's data while they are encrypted. At the same time, the service provider may want to keep its processing algorithms secret since they represent the basis for its business. As a second example, we may consider a situation where a user wants to query a database (e.g. a database containing biometric data) without revealing to the database owner (say Bob) what he/she is looking for (again this necessity may be due to privacy reasons). It is evident that the availability of tools that allow to process an encrypted query would represent a valuable help to solve this problem.

In the following we will refer to the above approach, whereby signals are processed while they are encrypted, as s.p.e.d. (signal processing in the encrypted domain).

Though processing encrypted signals may seem a formidable, if not impossible task, a few approaches exist that make s.p.e.d. possible. The majority of these approaches concern content retrieval and content protection applications. The interested reader can find an extensive review of these secure signal processing applications in [1]. By neglecting ad hoc solutions, that are suited to solve only particular problems in very narrow scenarios, two general approaches exist to process encrypted signals: the first one is based on homomorphic encryption [2], [3], the second one relies on multiparty computation (MPC) [4]–[6].

A cryptosystem is said to be homomorphic with respect to an operation $\star$ if there exists an operator $\phi(\cdot,\cdot)$ such that for any two plain messages a and b, we have:

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

where $E[\cdot]$ ($D[\cdot]$) denotes the encryption (decryption) operator. It is evident that homomorphic encryption provides an elegant way of performing at last a reduced set of operations by working on encrypted data. Among homomorphic cryptosystems, additively homomorphic encryption plays a crucial role in practical applications, since it provides a way of applying any linear operator in the encrypted domain.

The second approach to s.p.e.d. relies on MPC [4]. Here n players want to compute the output of a function $f(\cdot)$ with n inputs, each of which is known to one of the players. A MPC protocol permits to compute the output of f without that the players have to reveal their private inputs. It has been shown [4] that, at least in principle, the output of any function f can

Xxxxx xxxxxxx xxxxx xxxx xxxxxxxxx xxxxx xxxx xxxxx xxxx xxxxxxxx xxx xxxxxxx xxxxxx xxxx xxxx xxxxxxx xx xxxx xxxxxx xxxx xxxxxx xxxxxx xx xxxxx xxxxxxx xx xxxxxxx xx xxxxxxx xxxxxxxxxx xxx xxxxxxx xxxxxxxxxxxxx xx xxxxxxxx xxxxxx xxx xxxx xxxxx xxxxx xxx xxxx

Tiziano Bianchi and Alessandro Piva are with the Dipartimento di Elettronica e Telecomunicazioni, Università di Firenze, Via S. Marta 3, I-50139, Firenze, Italy (phone: +39 055 4796380, fax: +39 055 494569, e-mail: {tiziano.bianchi, alessandro.piva}@unifi.it). Mauro Barni is with the Dipartimento di Ingegneria dell'Informazione, Università di Siena,Via Roma 56, 53100, Siena, Italy (phone: + 39 0577 233601, fax: + 39 0577 233602, e-mail: barni@dii.unisi.it)

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.

Digital Object Identifier XXXXXXXXXXXXXXXX

be computed securely through MPC; the problem however is that MPC protocols are extremely complex since they may require that several interactive rounds are carried out among the players to compute the output of f.

In this paper we focus on non-interactive s.p.e.d. Specifically we focus on the representation problems that must be solved in order to exploit homomorphic encryption for processing encrypted signals.

The classical ways to represent signals are through floating point or fixed point arithmetic. Unfortunately both floating point and fixed point arithmetic are not suited for implementation in the encrypted domain through homomorphic encryption. As a matter of fact adding two floating point numbers requires much more complex operations than simple addition: first the numbers must be expressed in a form having the same exponent, then the significands must be added and then the exponent adjusted so that the significand of the result stays in the [0.1, 1) range. Clearly these operations can not be carried out in the encrypted domain by relying on homomorphic encryption only, but some interactive protocol is required [7]. Similar considerations hold for fixed point arithmetic, where after each addition (multiplication) the resulting number must be truncated to avoid overflow and underflow errors[1].

In this paper we focus on the above representation problem in the framework of DFT/FFT computation in the encrypted domain. In fact, DFT and FFT are very popular signal processing tools, and hence it is very likely that they will have to be implemented in s.p.e.d. applications. This is the case for instance of a pattern recognition module operating on set of frequency coefficients, a frequency-domain watermarking system embedding the watermark in the encrypted domain [8]–[10], or a filtering operation carried out in the frequency domain[2].

Specifically, we introduce a theoretical framework whereby the signal representation problem can be analyzed and solved. Several important issues are considered for the direct DFT, the radix-2, and the radix-4 fast Fourier algorithms, including the error analysis and the maximum size of the sequence that can be transformed. We also provide computational complexity analyses and comparisons. The results show that the radix-4 FFT is best suited for an encrypted domain implementation in the proposed scenarios.

Despite the focus of the paper is on DFT/FFT implementation, the theoretical framework we developed is a very general one and can be used in a wide variety of situations.

The rest of this paper is organized as follows. In Section II the e-DFT (encrypted-DFT) is defined and the related representation problem introduced. Section III reviews some properties of homomorphic and probabilistic cryptosystems and gives a summary of Paillier cryptosystem. In Section IV, we introduce a suitable signal representation for the encrypted domain. Section V is devoted to the analysis of the upper bounds on the encrypted domain representation for different DFT implementations, whereas Section VI takes into account quantization errors. In Section VII, we derive the complexity of different s.p.e.d. FFT algorithms (namely radix-2 and radix-4 FFT). Finally, in Section VIII we propose some examples of s.p.e.d. FFT designs, while some concluding remarks are given in Section IX.

[1]Given the current state of art in homomorphic encryption there is no way to compare or threshold two encrypted numbers. At the same time resorting to MPC at this low level is unacceptably complex.

[2]It could be argued that when the DFT module is either at the beginning or at the end of the processing chain, the DFT could be applied to the signal by the signal owner itself. However, this is not the case if the DFT is a step of a much more complex processing chain, where the DFT is applied to intermediate data.

II. Encrytpted domain - Discrete Fourier Transform (e-DFT)

The DFT of a sequence $x(n)$ is defined as:

$$X(k) = \sum_{n=0}^{M-1} x(n) W^{nk}, \quad k = 0, 1, \ldots, M-1 \tag{2}$$

where $W = e^{-j2\pi/M}$ and $x(n)$ is a finite duration sequence with length M. Among the appealing properties of the above transform one is that it can be implemented via fast algorithms, noted as fast Fourier transforms (FFTs).

We will consider a scenario in which the transform processor is fed with a sample-wise encrypted version of the input vector, that is

$$E[\mathbf{x}] = (E[x(0)], E[x(1)], \ldots, E[x(M-1)]) . \tag{3}$$

In order to make possible linear computations on encrypted values, 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

$$D[\phi(E[a], E[b])] = a + b \tag{4}$$

With such a cryptosystem is indeed possible to add two encrypted values without first decrypting them. Moreover, it is possible to multiply an encrypted value by a public integer value by repeatedly applying the operator $\phi(\cdot, \cdot)$.

Another required property of the cryptosystem is that it should be *probabilistic*, or *semantically secure*, that is, given two encrypted values it should not be possible to decide if they conceal the same value. This is fundamental, since the alphabet to which the input samples belong is usually limited, and a non-semantically secure cryptosystem would disclose a great amount of information about the statistical distribution of the input signal. A widely known example of a cryptosystem fulfilling both the above requirements is the Paillier cryptosystem [11], for which the operator $\phi(\cdot, \cdot)$ is a modular multiplication.

Since the DFT transform coefficients are public, the expression in (2) can be computed on an encrypted input vector by relying on the homomorphic property, as will be shown in Section IV. As we already mentioned, when the above idea is applied in practice some issues need to be addressed.

First of all, both the input samples and the DFT coefficients need to be represented as integer values. Since many practical homomorphic cryptosystems (e.g., Paillier) are based on modular operations on a finite field/ring, the inputs $x(n)$ and the output values $y(n)$ need to be correctly represented as integers on an appropriate finite field. Here, by "correctly represented" we mean that the actual value of a sample should always be

recoverable from its finite field representation. For example, in the case we are working on $\mathbb{Z}_N$, the set of integers modulo N, for each sample we should have $|x| \leq (N-1)/2$; otherwise, its magnitude will be lost due to the modulo N operations.

Secondly, FFT like algorithms should be applicable also in the encrypted domain, thus permitting to achieve the same computation savings achievable in the plain domain.

In the sequel, we will present a theoretical framework wherein the above issues can be cast and solved. A convenient signal representation for s.p.e.d. will be proposed, allowing us to define both a s.p.e.d. DFT and a s.p.e.d. FFT. We will analyze the quantization error introduced by the s.p.e.d. implementation and the maximum size of the sequence that can be transformed. Moreover, we will provide a computational complexity analysis, taking into account the requirements of different s.p.e.d. scenarios.

III. Homomorphic and Probabilistic Cryptosystems

One tool for signal processing in the encrypted domain is represented by cryptosystems that allow 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. [2]: in this paper, the authors define privacy homomorphisms as encryption functions which permit encrypted data to be operated on without preliminary decryption of the operands.

For an exact definition of a homomorphic cryptosystem, let us first introduce some notations: given a set of possible plain texts $\mathcal{M}$, a set of cipher texts $\mathcal{C}$ and a key pair $\{pk, sk\}$ (public key and secret key), a public key encryption scheme is a couple of functions $E_{pk} : \mathcal{M} \to \mathcal{C}, D_{sk} : \mathcal{C} \to \mathcal{M}$ such that, given a plaintext $m \in \mathcal{M}$, $D_{sk}(E_{pk}(m)) = m$ and such that, given a cipher text $c \in \mathcal{C}$, it is computationally unfeasible to determine m such that $E_{pk}(m) = c$, without knowing the secret key sk.

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 will focus on the former.

An additively homomorphic cryptosystem allows to map an addition in the plaintext domain to an operation in the ciphertext domain, that usually is a multiplication. Given two plaintexts m_1 and m_2, the following equalities are then satisfied:

$$D_{sk}(E_{pk}(m_1) \cdot E_{pk}(m_2)) = m_1 + m_2 \tag{5}$$

and, as a consequence,

$$D_{sk}(E_{pk}(m)^a) = am \tag{6}$$

where a is a public integer and

$$D_{sk}(E_{pk}(m_1) \cdot E_{pk}(m_2)^{-1}) = m_1 - m_2. \tag{7}$$

Additively homomorphic cryptosystems allow to perform in the encrypted domain additions, subtractions and multiplications with a known (non-encrypted) value. However, it is not possible to perform divisions, since this operation could lead to non integer values.

Another feature that we need is that the encryption scheme does not encrypt two equal plain texts into the same cipher text. More generally, given two encrypted values it should not be computationally feasible to decide if they conceal the same value or not. For this purpose, it is possible to define a scheme where the encryption function E_{pk} is a function of both the secret message m and a random parameter r, such that if $r_1 \neq r_2$ no adversary can distinguish $E_{pk}(m_1, r_1)$ from $E_{pk}(m_2, r_2)$, for any two secret messages m_1, m_2. Let $c_1 = E_{pk}(m, r_1)$ and $c_2 = E_{pk}(m, r_2)$, for a correct behavior the scheme has to be designed in such a way that $D_{sk}(c_1) = D_{sk}(c_2) = m$, that is the decryption phase is deterministic, not depending on the random parameter r. A scheme that satisfies the above property is commonly referred to as probabilistic [3] or semantically secure.

Luckily, encryption schemes that satisfy both the homomorphic and probabilistic properties detailed above do exist. One of the most known homomorphic and probabilistic schemes is the one presented by Paillier in [11], and later modified by Damgård and Jurik in [12].

A. Paillier Cryptosystem

The cryptosystem described in [11], usually referred to as Paillier cryptosystem, is based on the problem to decide whether a number is an N-th residue modulo N^2. This problem is believed to be computationally hard in the cryptography community, and is linked to the hardness to factorize N, if N is the product of two large primes. For a complete description of the Paillier cryptosystem we refer to the original paper [11]. Here, we simply give the encryption and the decryption procedures. The notation we use is the classic one, with $\mathbb{Z}_N$ the set of the integer numbers modulo N, and $\mathbb{Z}_N^*$ the set of invertible elements modulo N, i.e. all the integer numbers modulo N that are relatively prime with N.

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 $Z_{N^2}^*$ an element of order[3] αN for some $\alpha \neq 0$ ($g = N+1$ is usually a convenient choice). (N, g) is the public key.

2) Encryption: let $m < N$ be the plaintext, and $r < N$ a random value. The encryption c of m is:

$$c = E_{pk}(m, r) = g^m r^N \mod N^2$$

3) Decryption: let $c < N^2$ be the ciphertext. The plaintext m hidden in c is:

$$m = D_{sk}(c) = \frac{L(c^\lambda \mod N^2)}{L(g^\lambda \mod N^2)} \mod N$$

where $L(x) = \frac{x-1}{N}$. From the above equations, we can easily verify that the Paillier cryptosystem is additively homomorphic, since $E_{pk}(m_1, r_1)E_{pk}(m_2, r_2) = g^{m_1+m_2}(r_1 r_2)^N = E_{pk}(m_1 + m_2, r_1 r_2)$.

[3]The order of an integer a modulo N is the smallest positive integer k such that $a^k = 1 \mod N$.

IV. Signal Model for the Encrypted Domain

Let us consider a signal $x(n) \in \mathbb{C}$, with $x(n) = x_R(n) + jx_I(n)$, $x_{R,I} \in \mathbb{R}$. In the following, we will assume that the signal is bounded in amplitude, i.e., $|x(n)| \leq 1$, from which $|x_{R,I}(n)| \leq 1$.

In order to process $x(n)$ in the encrypted domain, the signal values must be approximated by suitable integers. This is accomplished by the following quantization process

$$s(n) = \lceil Q_1 x(n) \rfloor = \lceil Q_1 x_R(n) \rfloor + j \lceil Q_1 x_I(n) \rfloor = s_R(n) + j s_I(n) \tag{8}$$

where $\lceil \cdot \rfloor$ is the rounding function and Q_1 is a suitable scaling factor. Fore the sake of simplicity, we will assume that Q_1 is an integer. Based on the properties of $x(n)$, the quantized signal will satisfy $-Q_1 \leq s_{R,I}(n) \leq Q_1$.

In the following, we will consider the encryption of $s(n)$ as the separate encryption of both $s_R(n)$ and $s_I(n)$, i.e., $E[s(n)] = \{E[s_R(n)], E[s_I(n)]\}$. Hence, if the cryptosystem encrypts integers modulo N we need a one-to-one mapping between $z_{R,I}(n) = s_{R,I}(n) \mod N$ and $s_{R,I}(n)$, so that we can always recover the correct value of $s_{R,I}(n)$ from $z_{R,I}(n)$. This can be achieved by imposing $N \geq 2Q_1 + 1$, so that

$$s_{R,I}(n) = \begin{cases} z_{R,I}(n) & \text{if } z_{R,I}(n) < N/2 \\ z_{R,I}(n) - N & \text{if } z_{R,I}(n) > N/2 \end{cases} \tag{9}$$

The coefficients W^{nk} in (2) can be quantized using the same strategy as above. In particular, we define

$$C(u) = \lceil Q_2 W^u \rfloor = \lceil Q_2 \cos(2\pi u/M) \rfloor - j \lceil Q_2 \sin(2\pi u/M) \rfloor = C_R(u) + jC_I(u) \tag{10}$$

where Q_2 is the DFT coefficient scaling factor. Thanks to the properties of W, we have $-Q_2 \leq C_{R,I}(u) \leq Q_2$.

Based on the definitions above, the integer approximation of the DFT is defined as

$$S(k) = \sum_{n=0}^{M-1} C(nk) s(n), \quad k = 0, 1, \ldots, M-1. \tag{11}$$

Since the above equation requires only integer multiplications and integer additions, it can be evaluated in the encrypted domain by relying on homomorphic properties. However, the s.p.e.d. implementation of both complex additions and complex multiplications should be considered. The implementation of a complex addition is trivial. As to a complex multiplication, two implementations can be considered [13], either requiring four real multiplications and two real additions, $sC = \{s_R C_R - s_I C_I, s_R C_I + s_I C_R\}$, or three real multiplications and three real additions, $sC = \{(s_R + s_I) C_R - s_I (C_R + C_I), (s_R + s_I) C_R - s_R (C_R - C_I)\}$. If the inputs are encrypted with the Paillier cryptosystem, when implemented in the encrypted domain such implementations become

$$E[s]^C_{(1)} \triangleq \left\{ E[s_R]^{C_R} E[s_I]^{-C_I}, E[s_R]^{C_I} E[s_I]^{C_R} \right\} \tag{12}$$

$$E[s]^C_{(2)} \triangleq \left\{ E[Z] E[s_R]^{-C_+}, E[Z] E[s_I]^{-C_-} \right\} \tag{13}$$

where $E[Z] = (E[s_R] E[s_I])^{C_R}$, $C_+ = C_R + C_I$, $C_- = C_R - C_I$ and all computations are carried out modulo N^2. For example, if we use (12) the DFT in the encrypted domain can be evaluated as

$$E[S(k)] \triangleq \prod_{n=0}^{M-1} E[s(n)]^{C(nk)}_{(1)} = \left\{ \prod_{n=0}^{M-1} E[s_R(n)]^{C_R(nk)} E[s_I(n)]^{-C_I(nk)}, \prod_{n=0}^{M-1} E[s_R(n)]^{C_I(\right. \tag{14}$$

for $k = 0, 1, \ldots, M-1$.

V. Upper Bounds and Magnitude Requirements

The computation of the DFT using (11) requires two problems to be tackled. The first one is that there will be a scaling factor between $S(k)$ and the desired value $X(k)$. The second one is that, in order to implement (11) using a cryptosystem which encrypts integers modulo N, one must ensure that the one-to-one mapping in (9) still holds for $S(k) \mod N$. Hence, according to the proposed model, one has to find an *upper bound* Q_S on $S(k)$ such that $|S_{R,I}(k)| \leq Q_S$, and verify that $N \geq 2Q_S + 1$.

In the following we will show that, irrespective of the DFT implementation, $S(k)$ can always be expressed as

$$S(k) = KX(k) + \epsilon_S(k) \tag{15}$$

where K is a scale factor depending on the particular implementation, whereas $\epsilon_S(k)$ takes into account the propagation of the quantization error.

Based on the above equation, the desired DFT output can be obtained as $\tilde{X}(k) = S(k)/K$. As to the upper bound, we have $|S(k)| \leq K|X(k)| + |\epsilon_S(k)|$. Hence, for all k we obtain

$$|S(k)| \leq \lfloor MK + \epsilon_{S,max} \rfloor \triangleq Q_S \tag{16}$$

where $\epsilon_{S,max}$ is a suitable upper bound on $\epsilon_S(k)$ and we used $|X(k)| \leq M$.

The value of K and $\epsilon_{S,max}$ depends on the scaling factors Q_1 and Q_2 and on the particular implementation of the DFT. These issues will be discussed in the following sections.

A. Direct Computation

Let us express the quantized samples in (8) and the quantized coefficients in (10) as $s(n) = Q_1 x(n) + \epsilon_s(n)$ and $C(u) = Q_2 W^u + \epsilon_W(u)$, respectively, where $\epsilon_s(n)$ and $\epsilon_W(u)$ are the quantization errors. If the DFT is computed directly by applying (11), then we have

$$S(k) = Q_1 Q_2 X(k) + \sum_{n=0}^{M-1} \left[Q_1 x(n) \epsilon_W(nk) + Q_2 \epsilon_s(n) W^{nk} + \epsilon_s(n) \epsilon \right. \tag{17}$$

The scaling factor in (15) is then $K = Q_1 Q_2$. As to the upper bound Q_S given by equation (16), due to the properties of the rounding function, $|\epsilon_{s,W}(n)| \leq 1/\sqrt{2}$. Hence $|s(n)| \leq Q_1 + 1/\sqrt{2}$, $|C(u)| \leq Q_2 + 1/\sqrt{2}$ and, after simple manipulations,

$$|S(k)| \leq M \left(Q_1 Q_2 + \frac{Q_1}{\sqrt{2}} + \frac{Q_2}{\sqrt{2}} + \frac{1}{2} \right) \tag{18}$$

from which we derive $Q_S = MQ_1 Q_2 + \lfloor Q_1/\sqrt{2} + Q_2/\sqrt{2} + 1/2 \rfloor$.

1) Real-Valued Signals: In the case of the DFT of a real-valued signal, the direct DFT computation can be expressed as

$$S(k) = \sum_{n=0}^{M-1} C_R(nk)s(n) + j \sum_{n=0}^{M-1} C_I(nk)s(n), \quad k = 0, 1, \ldots, M-1 \tag{19}$$

From the properties of both $s(n)$ and $C_{R,I}(u)$, it is evident that $|s(n)C_{R,I}(u)| \leq Q_1 Q_2$. Therefore, both the real and the imaginary part of the integer DFT values will satisfy $|S_{R,I}(k)| \leq Q_S = MQ_1Q_2$. This upper bound is lower than in the complex-valued case. However, in the case of a FFT algorithm this applies only for the first stages, since a generic stage of the FFT will consider a vector of complex valued samples. It is also worth noting that real-valued signals are usually processed by means of a half-length complex FFT [14], since this leads to a sensible reduction of the complexity. Hence, in the following only the complex-valued case will be considered.

2) Bounds on Complex Multiplications: The bound in (18) does not take into account the intermediate computations of a complex multiplication. This is not a problem in the case of (12), since $|s_{R,I}C_{R,I}| \leq |sC|$, that is, the intermediate values are bounded by the final value. However, in the case of (13) we have $|s_R + s_I| \leq |s|\sqrt{2}$ and $|C_R \pm C_I| \leq \sqrt{2}$, i.e., the intermediate values may exceed the final values. In order to cope with this behavior, the bound in (18) should be multiplied by a factor $\sqrt{2}$ whenever (13) is used. For the sake of simplicity, the upper bounds in the following sections will be derived under the hypothesis that (12) is used. The corresponding upper bounds in the case of (13) can be obtained by multiplying by $\sqrt{2}$.

B. Decimation in Time Radix-2 FFT

This algorithm is applied when $M = 2^\nu$ and allows the DFT to be computed in ν stages each requiring $M/2$ complex multiplications. At each stage, a pair of coefficients is obtained as a linear combination of the corresponding pair of coefficients computed at the previous stage, using the following *butterfly* structure

$$X^{(t+1)}(k_0) = X^{(t)}(k_0) + W^u X^{(t)}(k_1) \tag{20}$$
$$X^{(t+1)}(k_1) = X^{(t)}(k_0) - W^u X^{(t)}(k_1) \tag{21}$$

where the indexes k_0, k_1 and the exponent u depend on the particular stage [15]. The computation of the above butterfly can be performed in the encrypted domain by applying the proposed model, yielding

$$S^{(t+1)}(k_0) = Q_2 S^{(t)}(k_0) + C(u)S^{(t)}(k_1) \tag{22}$$
$$S^{(t+1)}(k_1) = Q_2 S^{(t)}(k_0) - C(u)S^{(t)}(k_1). \tag{23}$$

Note that the multiplication by Q_2 is required in order to add (or subtract) integers which are related to the corresponding complex coefficients by the same scale factor. Hence, the integer implementation of the FFT algorithm requires M integer multiplications at each stage.

As to the upper bound analysis, the two branches of the butterfly are equivalent. Without loss of generality, let us consider the first branch. If we express $S^{(t)}(k_0) = K^{(t)}X^{(t)}(k_0) + \epsilon_S^{(t)}(k_0)$, then we have

$$\begin{aligned} S^{(t+1)}(k_0) =& Q_2 K^{(t)} \left(X^{(t)}(k_0) + W^u X^{(t)}(k_1) \right) \\ &+ Q_2 \left(\epsilon_S^{(t)}(k_0) + W^u \epsilon_S^{(t)}(k_1) \right) \\ &+ K^{(t)} X^{(t)}(k_1) \epsilon_W(u) + \epsilon_S^{(t)}(k_1)\epsilon_W(u) \end{aligned} \tag{24}$$

from which we derive the following recursive relations

$$K^{(t+1)} = Q_2 K^{(t)} \tag{25}$$

$$\begin{aligned} \epsilon_S^{(t+1)}(k_0) =& Q_2 \left(\epsilon_S^{(t)}(k_0) + W^u \epsilon_S^{(t)}(k_1) \right) \\ &+ K^{(t)} X^{(t)}(k_1) \epsilon_W(u) + \epsilon_S^{(t)}(k_1)\epsilon_W(u). \end{aligned} \tag{26}$$

At the first stage $S^{(0)}(n) = s(\tilde{n}) = Q_1 x(\tilde{n}) + \epsilon_s(\tilde{n})$, where $\tilde{n}$ indicates n in bit reverse order, so that the recursion starts with $K^{(0)} = Q_1$ and $\epsilon_S^{(0)}(k_0) = \epsilon_s(\tilde{k}_0)$. By using the initial conditions in (26), it is easy to derive the scale factor as $K = K^{(\nu)} = Q_1 Q_2^\nu$.

As to the evaluation of the final upper bound, we consider an equivalent recursive relation on an upper bound of the quantization error represented in (26), given as

$$|\epsilon_S^{(t+1)}| \leq \left(2Q_2 + \frac{1}{\sqrt{2}} \right) |\epsilon_S^{(t)}| + \frac{2^t}{\sqrt{2}} K^{(t)}. \tag{27}$$

By using the initial condition $|\epsilon_S^{(0)}| \leq \frac{1}{\sqrt{2}}$ in (27), the upper bound on the final quantization error can be expressed as

$$|\epsilon_S^{(\nu)}| \leq \frac{1}{\sqrt{2}} \left(2Q_2 + \frac{1}{\sqrt{2}} \right)^\nu + \sum_{l=0}^{\nu-1} \frac{2^{\nu-1-l}}{\sqrt{2}} Q_1 Q_2^{\nu-1-l} \left(2Q_2 + \frac{1}{\sqrt{2}} \right)^l \tag{28}$$

However, the above upper bound does not take into account the properties of the twiggle factors W^u, which in particular cases may be quantized with smaller quantization errors than the upper bound $|\epsilon_W| \leq 1/\sqrt{2}$ or even without any quantization error. In particular, at the first stage we have $W^u = 1$, whereas at the second stage we have $W^u = \{1, j\}$. In both cases, no integer multiplication is required, and the butterflies can be modified so that no scaling factor is introduced. Therefore, $K^{(2)} = Q_1$ and by using (26), it is easy to derive the scale factor as $K = K^{(\nu)} = Q_1 Q_2^{\nu-2}$.

As to the upper bound, in the case of the first two stages the expression in (27) simplifies as $|\epsilon_S^{(t+1)}| \leq 2|\epsilon_S^{(t)}|$, since $\epsilon_W = 0$. Hence, by using as initial condition $|\epsilon_S^{(2)}| \leq 4/\sqrt{2}$ in (27) (since $|\epsilon_S^{(0)}| \leq 1/\sqrt{2}$), the upper bound on the quantization error can be expressed as

$$\begin{aligned} |\epsilon_S^{(\nu)}| \leq& \frac{4}{\sqrt{2}} \left(2Q_2 + \frac{1}{\sqrt{2}} \right)^{\nu-2} \\ &+ \sum_{l=0}^{\nu-3} \frac{2^{\nu-1-l}}{\sqrt{2}} Q_1 Q_2^{\nu-3-l} \left(2Q_2 + \frac{1}{\sqrt{2}} \right)^l = \epsilon_{S,R2,max}, \quad \nu > \end{aligned} \tag{29}$$

from which we derive the final upper bound on $S(k)$ as $Q_S = MQ_1Q_2^{\nu-2} + \lfloor \epsilon_{S,R2,max} \rfloor$.

C. Decimation in Time Radix-4 FFT

This algorithm can be employed when $M = 4^\mu$ and allows the DFT to be computed in μ stages each requiring $3M/4$ complex multiplications. The rationale of the radix-4 algorithm is that an M-point DFT can be evaluated as a linear combination of four $M/4$-point DFTs. Using this strategy, at each stage four new coefficients are obtained as a linear combination of four coefficients computed at the previous stage, using the following *radix-4 butterfly* [16]

$$X^{(t+1)}(k_l) = \sum_{i=0}^{3} X^{(t)}(k_i) W^{ui}(-j)^{il}, \quad l = 0, \ldots, 3. \quad (30)$$

Without loss of generality, the upper bound can be evaluated by considering the integer version of the first branch in the butterfly, that is

$$S^{(t+1)}(k_0) = Q_2 S^{(t)}(k_0) + C(u)S^{(t)}(k_1) + C(2u)S^{(t)}(k_2) + C(3u)S^{(t)}(k_3). \quad (31)$$

By using the same model as with the radix-2 case, the following recursive relations can be derived

$$K^{(t+1)} = Q_2 K^{(t)} \quad (32)$$

$$|\epsilon_S^{(t+1)}| \leq \left(4Q_2 + \frac{3}{\sqrt{2}}\right)|\epsilon_S^{(t)}| + 3\frac{4^t}{\sqrt{2}}K^{(t)}. \quad (33)$$

Moreover, at the first stage of the radix-4 FFT algorithm $W^u = 1$, so that the recursion begins with $K^{(1)} = Q_1$ and $|\epsilon_S^{(1)}| \leq 4/\sqrt{2}$. The scale factor is given as $K = K^{(\mu)} = Q_1 Q_2^{\mu-1}$, whereas the quantization error upper bound is

$$|\epsilon_S^{(\mu)}| \leq \frac{4}{\sqrt{2}}\left(4Q_2 + \frac{3}{\sqrt{2}}\right)^{\mu-1} + \sum_{l=0}^{\mu-2} \frac{4^{\mu-1-l}}{\sqrt{2}} Q_1 Q_2^{\mu-2-l}\left(4Q_2 + \frac{3}{\sqrt{2}}\right)^{l} = \epsilon_{S,R4,max} \quad (34)$$

from which we derive the upper bound on $S(k)$ as $Q_S = MQ_1Q_2^{\mu-1} + \lfloor \epsilon_{S,R4,max} \rfloor$.

The parameters determining the upper bounds for the different DFT implementations are summarized in Table I.

VI. Error Analysis

The effects of finite precision arithmetic on DFT/FFT computation have been extensively investigated in the literature [17]–[19]. However, the encrypted domain implementation of the DFT introduces some important differences with respect to the classical fixed point case. Since there is no rescaling after multiplication (as we said we can not rescale the encrypted values due to the limited set of operations made available by homomorphic encryption), there is no computational noise, which is one of the most relevant noise sources in fixed point implementations. Hence, the error introduced by the proposed DFT/FFT is only due to the quantization of the twiddle factors.

In order to estimate the overall quantization error on the DFT values, the noise-to-signal ratio (NSR) will be evaluated. We will assume that both $\epsilon_s(n)$ and $\epsilon_W(u)$ are i.i.d. variables with zero mean and variance σ_q^2 and $\sigma_{W,q}^2$, respectively. Although $\epsilon_W(u)$ is deterministic, this will produce a simple estimate of the quantization noise.

As to the input signal, its NSR can be estimated as

$$\eta = \frac{\sigma_q^2}{Q_1^2 \sigma_x^2} \quad (35)$$

where σ_x^2 indicates the signal power.

If we consider the DFT/FFT computation, the output NSR can be estimated as

$$\tilde{\eta} = \frac{\sigma_{\epsilon_S}^2}{K^2 M \sigma_x^2} \quad (36)$$

where $\sigma_{\epsilon_S}^2$ is the variance of $\epsilon_S(k)$. In the case of the direct DFT computation, by relying on equation (17) and neglecting the terms $\epsilon_s(n)\epsilon_W(nk)$, we can estimate $\sigma_{\epsilon_S}^2 \approx MQ_1^2\sigma_x^2\sigma_{W,q}^2 + MQ_2^2\sigma_q^2$. Hence, it is easy to derive

$$\tilde{\eta}_D = \eta + \frac{\sigma_{W,q}^2}{Q_2^2}. \quad (37)$$

In the case of a radix-2 FFT, the variance of the error at the tth stage can be recursively approximated by relying on equation (26) as

$$\sigma_{\epsilon_S^{(t+1)}}^2 \approx 2Q_2^2\sigma_{\epsilon_S^{(t)}}^2 + 2^t\sigma_x^2\sigma_{W,q}^2(K^{(t)})^2. \quad (38)$$

If we set the initial conditions $\sigma_{\epsilon_S^{(2)}}^2 = 4\sigma_q^2$ and $K^{(2)} = Q_1$, then we have $\sigma_{\epsilon_S}^2 = 2^\nu Q_2^{2(\nu-2)}\sigma_q^2 + (\nu-2)2^{\nu-1}\sigma_x^2\sigma_{W,q}^2 Q_1^2 Q_2^{2(\nu-3)}$. Therefore, the NSR can be expressed as

$$\tilde{\eta}_{R2} = \eta + \frac{\nu-2}{2}\frac{\sigma_{W,q}^2}{Q_2^2}. \quad (39)$$

Finally, in the case of a radix-4 FFT the variance of the error at the tth stage can be recursively approximated in a similar way as

$$\sigma_{\epsilon_S^{(t+1)}}^2 \approx 4Q_2^2\sigma_{\epsilon_S^{(t)}}^2 + 3\cdot 4^t\sigma_x^2\sigma_{W,q}^2(K^{(t)})^2 \quad (40)$$

and, since the initial conditions are $\sigma_{\epsilon_S^{(1)}}^2 = 4\sigma_q^2$ and $K^{(1)} = Q_1$, the error at the last stage is $\sigma_{\epsilon_S}^2 = 4^\mu Q_2^{2(\mu-1)}\sigma_q^2 + 3(\mu-1)4^{\mu-1}\sigma_x^2\sigma_{W,q}^2 Q_1^2 Q_2^{2(\mu-2)}$. Therefore, the NSR is given by

$$\tilde{\eta}_{R4} = \eta + \frac{3(\mu-1)}{4}\frac{\sigma_{W,q}^2}{Q_2^2}. \quad (41)$$

A. Comparison with Plaintext Implementation

Since the value of Q_1, and hence η, will be fixed by the properties of the input signal, the above formulas permit to evaluate the degradation introduced on the encrypted DFT coefficients as a function of Q_2. A fair design criterion could be that of choosing a value of Q_2 which yields a similar degradation with respect to that introduced by a plaintext FFT implementation.

In the following, comparisons will be made using a plaintext radix-2 FFT. Considering other plaintext FFTs yields similar results. According to the way a plaintext FFT is implemented, two cases need to be analyzed:

TABLE I
SCALING FACTOR K AND UPPER BOUND Q_S FOR DIFFERENT S.P.E.D. DFT IMPLEMENTATIONS.

	K	Q_S
Direct	Q_1Q_2	$MQ_1Q_2 + \lfloor Q_1/\sqrt{2} + Q_2/\sqrt{2} + 1/2 \rfloor$
DIT radix-2 ($M = 2^\nu$)	$Q_1Q_2^{\nu-2}$	$MQ_1Q_2^{\nu-2} + \lfloor \frac{4}{\sqrt{2}} \left(2Q_2 + \frac{1}{\sqrt{2}}\right)^{\nu-2} + \sum_{l=0}^{\nu-3} \frac{2^{\nu-1-l}}{\sqrt{2}} Q_1 Q_2^{\nu-3-l} \left(2Q_2 + \frac{1}{\sqrt{2}}\right)^l \rfloor$
DIT radix-4 ($M = 4^\mu$)	$Q_1Q_2^{\mu-1}$	$MQ_1Q_2^{\mu-1} + \lfloor \frac{4}{\sqrt{2}} \left(4Q_2 + \frac{3}{\sqrt{2}}\right)^{\mu-1} + \sum_{l=0}^{\mu-2} \frac{4^{\mu-1-l}}{\sqrt{2}} Q_1 Q_2^{\mu-2-l} \left(4Q_2 + \frac{3}{\sqrt{2}}\right)^l \rfloor$

1) *Fixed point implementation*: if a plaintext radix-2 FFT is implemented on a fixed point hardware with registers having b_2 bits and scaling is performed at each stage, an equivalent encrypted domain implementation should satisfy $\tilde{\eta} - \eta \leq 4M \cdot 2^{-2b_2}/3\sigma_x^2$ [15], where σ_x^2 is the power of the input signal, assumed white. If we assume a uniformly distributed quantization error on the DFT coefficients, i.e., $\sigma_{W,q}^2 = 1/6$, we have

$$Q_2 \geq \begin{cases} \sigma_x \cdot 2^{b_2 - \nu/2 - 3/2} & \text{DFT} \\ \sigma_x \sqrt{\nu - 2} \cdot 2^{b_2 - \nu/2 - 2} & \text{radix-2 FFT} \\ \sigma_x \sqrt{3\nu/2 - 3} \cdot 2^{b_2 - \nu/2 - 5/2} & \text{radix-4 FFT} \end{cases} \tag{42}$$

where we have assumed $\nu = 2\mu$. As a consequence, the s.p.e.d. implementation will require in the worst case $n_2 = \log_2 Q_2 \approx b_2 - \nu/2 + 0.5 \log_2 \nu$ bits for quantizing the twiddle factors, i.e., we save approximately $\nu/2$ bits with respect to a plaintext fixed point implementation.

2) *Floating point implementation*: in the case of a plaintext radix-2 FFT implementation on a floating pointing hardware using f_2 bits for the fractional part, the NSR bound is given by $\tilde{\eta} - \eta \leq 2\nu \cdot 2^{-2f_2}/3$ [15]. Hence

$$Q_2 \geq \begin{cases} \sqrt{1/4\nu} \cdot 2^{f_2} & \text{DFT} \\ \sqrt{(\nu - 2)/8\nu} \cdot 2^{f_2} & \text{radix-2 FFT} \\ \sqrt{(3\nu/2 - 3)/16\nu} \cdot 2^{f_2} & \text{radix-4 FFT.} \end{cases} \tag{43}$$

In this case, the quantization of the twiddle factors in the s.p.e.d. implementation requires approximately the same number of bits as the fractional part of the floating point registers.

VII. COMPLEXITY ANALYSIS

The complexity of the proposed DFT implementation in the encrypted domain depends on several parameters which are related to the used cryptosystem, its homomorphic properties, to the input signal, and the desired NSR level.

For the sake of simplicity, in this paper we will assume that a Paillier cryptosystem or one of its extensions are used. Hence, each addition between plaintexts will be translated into a modular multiplication between cyphertexts, and each multiplication between plaintexts will be translated into a modular exponentiation of a cyphertexts to a plaintext. Moreover, an encrypted subtraction requires a modular division, which is usually more complex than a modular multiplication [20], [21]. In the following, we will consider an implementation of subtractions requiring one modular multiplication and one modular inversion. The same holds for exponentiations to negative exponents, usually implemented as $(a^{-e})^{-1} \mod n$.

TABLE II
COMPUTATIONAL COMPLEXITY OF S.P.E.D. FFT ALGORITHMS: A COMPLEX MULTIPLICATION IS IMPLEMENTED THROUGH FOUR REAL MULTIPLICATIONS.

	radix-2	radix-4	DFT
MEs	$3M \log_2 M - 6M$	$\frac{7}{4} M \log_2 M - \frac{7}{2} M$	$4M^2$
MMs	$3M \log_2 M - 2M$	$\frac{11}{4} M \log_2 M - \frac{3}{2} M$	$4M^2 - 2M$
MIs	$3M \log_2 M - 4M$	$\frac{9}{4} M \log_2 M - \frac{5}{2} M$	$2M^2$

As a result, the complexity will be evaluated as the number of modular exponentiations (ME), modular multiplications (MM), and modular inversions (MI) which are required for implementing the DFT/FFT algorithms in the encrypted domain.

The DFT/FFT algorithms are based on complex-valued arithmetic. Hence, the complexity of a complex addition, a complex subtraction and a complex multiplication have to be translated into ME, MM, and MI. As to a s.p.e.d. complex addition, it always requires two MMs, while the complexity of a complex subtraction is two MMs and two MIs. As to a complex multiplication, we consider the two implementations in (12)-(13), either requiring four MEs, two MMs and one MI, or three ME, three MMs and two MIs. Moreover, if we assume that the sign of the multipliers is uniformly distributed, either two additional MIs or one and a half additional MIs should be considered on the average. Finally, if a complex value is multiplied by a real value (rescaled) the complexity is always two MEs and one MI on the average.

The complexity of the direct DFT is simply M^2 complex multiplications and $M(M-1)$ complex additions. The complexity of radix-2 can be derived as follows. Each stage of the radix-2 FFT, except the first two stages, requires $M/2$ complex multiplications plus $M/2$ rescalings of complex values when implemented in the encrypted domain. Moreover, each stage requires also $M/2$ complex additions and $M/2$ complex subtractions. A similar procedure can be used to derive the complexity of the encrypted radix-4 FFT. In this case each stage, except the first stage, requires $3M/4$ complex multiplications plus $M/4$ rescalings of complex values. Moreover, each radix-4 stage requires also M complex additions and M complex subtractions. The complexity results for the different algorithms in terms of MEs, MMs and MIs are summarized in Tables II and III.

A remark about the encrypted DFT/FFT complexity regards the different weight of the different modular operations. If $n_2 = \lceil \log_2 Q_2 \rceil$, a modular exponentiation will require on the average $3n_2/2$ modular multiplications. Hence, in several

TABLE III
COMPUTATIONAL COMPLEXITY OF S.P.E.D. FFT ALGORITHMS: A COMPLEX MULTIPLICATION IS IMPLEMENTED THROUGH THREE REAL MULTIPLICATIONS.

	radix-2	radix-4	DFT
MEs	$\frac{5}{2}M\log_2 M - 5M$	$\frac{11}{8}M\log_2 M - \frac{11}{4}M$	$3M^2$
MMs	$\frac{7}{2}M\log_2 M - 3M$	$\frac{25}{8}M\log_2 M - \frac{9}{4}M$	$5M^2 - 2M$
MIs	$\frac{13}{4}M\log_2 M - \frac{9}{2}M$	$\frac{39}{16}M\log_2 M - \frac{23}{8}M$	$\frac{9}{2}M^2$

practical cases the cost of the modular multiplications is negligible. This may not hold true in the case of the modular inversions, whose complexity is in general higher than that of a modular multiplication. Hence, the choice of the most convenient implementation between the four multiplication scheme and the three multiplication scheme will depend on the actual implementation of a modular inversion/division.

When the modulus used by the cryptosystem remains the same independently of the implementation, the above estimates can be directly compared and the results are similar to the classical case. However, in order to maintain the same value of N, the maximum allowable value of Q_2 is reduced in the case of the FFT algorithms. Since there can be some applications in which this may not be acceptable because of the requirements on the quantization noise, the analysis of the overall complexity is strongly dependent on the actual values of Q_1 and Q_2.

VIII. PRACTICAL EXAMPLES

As shown in the previous sections, the choice of the parameters of the DFT to be implemented in the encrypted domain depends on several aspects, including the constraints imposed by the cryptosystem, the required SNR, and the complexity of the implementation. The aim of this section is to provide design criteria which take into account the different issues raised by the s.p.e.d. DFT. First, we will try to identify some typical scenarios that can be encountered in practical applications. Therefore, we will show how the proposed analysis allows us to assess which s.p.e.d. DFT algorithm is more suitable for a particular scenario. In the derivation of such criteria, the analysis made in the previous sections has a crucial role, since it allows us to give general rules for the choice of the s.p.e.d. DFT parameters, which will apply also in more general scenarios.

We will assume that each encrypted domain implementation fulfills the same requirements in terms of input and output NSR as the plaintext version. Moreover, for the sake of simplicity we will assume that both Q_1 and Q_2 are powers of two, i.e., $Q_1 = 2^{n_1}$ and $Q_2 = 2^{n_2}$. Finally, we will indicate the bit length of the modulus used by Paillier as $n_P = \lceil \log_2 N \rceil$. For security reasons, recent applications usually requires $n_P \geq 1024$.

In a non-encrypted domain (plaintext) implementation of the DFT, different scenarios may arise, since both the inputs and the twiddle factors in the DFT/FFT implementation may be either fixed point or floating point numbers. In our analysis, we will consider the following cases:

- *Fixed point inputs*: if the input signal is quantized using b_1 bits, its values can be directly mapped onto integer values in the interval $[-2^{b_1-1}, 2^{b_1-1}-1]$, so that we can assume $n_1 = b_1 - 1$;
- *Floating point inputs*: in order to preserve the whole dynamic of the normalized floating point representation, one should be able to represent values from $\pm 2^{-2^{c_1-1}-2}$ to $\pm 2^{2^{c_1-1}}$, where c_1 is the number of bit of the exponent. Unless some information about the properties of the input signal are known, this requires $n_1 = 2^{c_1} - 2$;
- *Fixed point implementation*: by using $\sigma_x \leq 1$ and $\nu \geq 3$, a choice satisfying (42) for all implementations is $Q_2 \geq \sqrt{2\nu} \cdot 2^{b_2 - \nu/2 - 5/2}$ or, equivalently, $n_2 \geq b_2 - \nu/2 + (\log_2 \nu)/2 - 2$. If we assume $\nu \leq 32$, we can set $n_2 = \lceil b_2 - \nu/2 + 1/2 \rceil$;
- *Floating point implementation*: a choice satisfying (43) for all implementations and all values of ν is $Q_2 \geq \frac{1}{2}2^{f_2}$, from which $n_2 = f_2 - 1$;

A. Implementation Constraints

One of the main problems of an implementation in the encrypted domain is its *feasibility*. Consider a scenario in which a set of encrypted signals must undergo different processing tasks. It is not realistic to adapt the parameters of the cryptosystem according to the processing task, mainly because this would produce a huge amount of encrypted data, and encrypting data with an homomorphic cryptosystem is usually an expensive procedure. In such a scenario, it is reasonable to assume that the signals are encrypted once, and that each processing task employs the same set of encrypted data. Therefore, each processing task must rely on an feasible implementation, i.e., an implementation satisfying the requirements on the modulus.

Given $M = 2^\nu$, Q_1, and Q_2, in order to ensure that no wrap-around occurs in the internal computations the modulus of the cryptosystem must satisfy

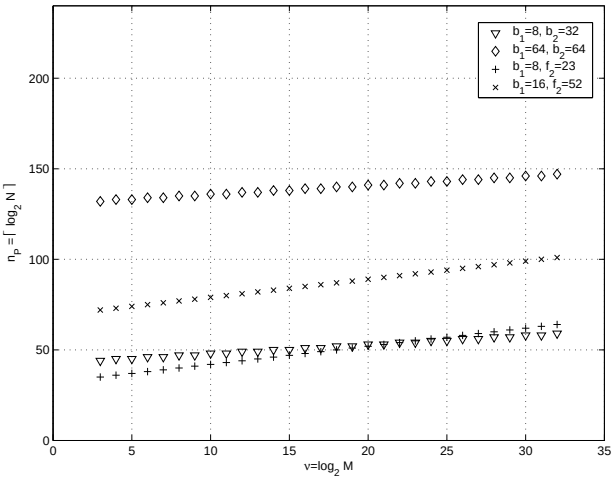
$$N \geq 2\left(2^\nu Q_1 Q_2^\alpha + \xi\right) + 1 \tag{44}$$

where we can have $\alpha = 1$ (DFT), $\alpha = \nu - 2$ (radix-2) or $\alpha = \nu/2 - 1$ (radix-4), and ξ can be obtained from equation (29), (34) or (18). Considering practical choices of Q_1 and Q_2, it is safe to assume $\xi < 2^\nu Q_1 Q_2^\alpha - 1/2$, so that the above bound is satisfied by requiring

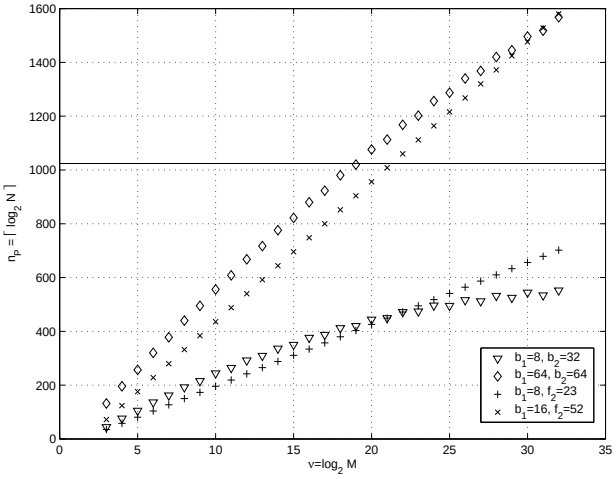
$$n_P \geq \nu + n_1 + \alpha n_2 + 3. \tag{45}$$

By using the above relationship, it is easy to assess whether a particular FFT can be implemented by relying on the minimum modulus (and, hence, on a standard Paillier implementation) or it requires an ad-hoc cryptosystem, e.g. as a function of ν.

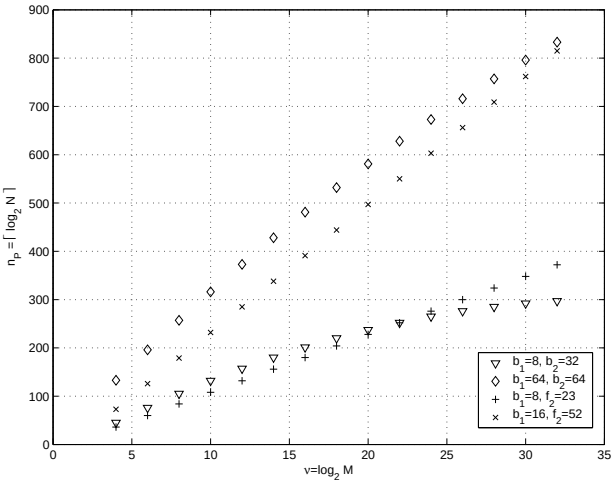
According to the considered scenario, one can choose the convenient values of n_1 and n_2 and substitute them into (45) in order to assess which implementation is feasible. In Fig. 1, we show the minimum n required by four different scenarios characterized by fixed point inputs. If the number of FFT points is not very high (above 2^{19}), in all the scenarios the encrypted FFT can be implemented relying on the given



(a)

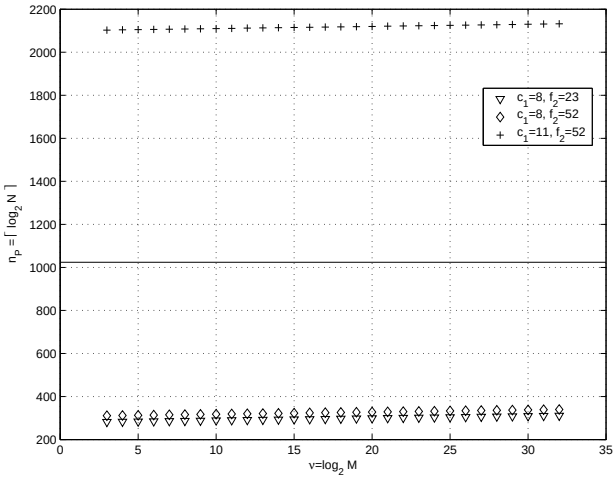


(b)

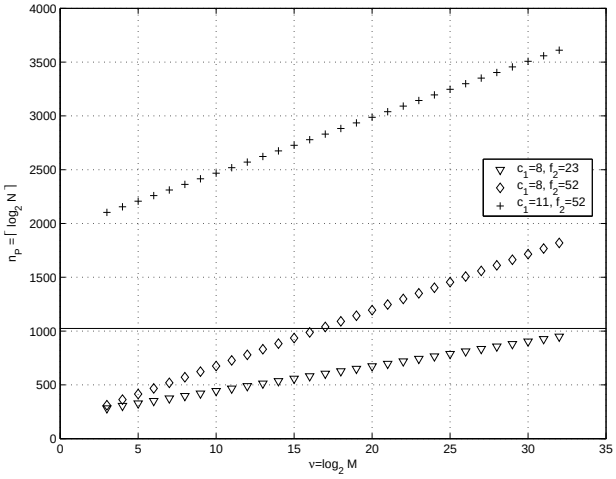


(c)

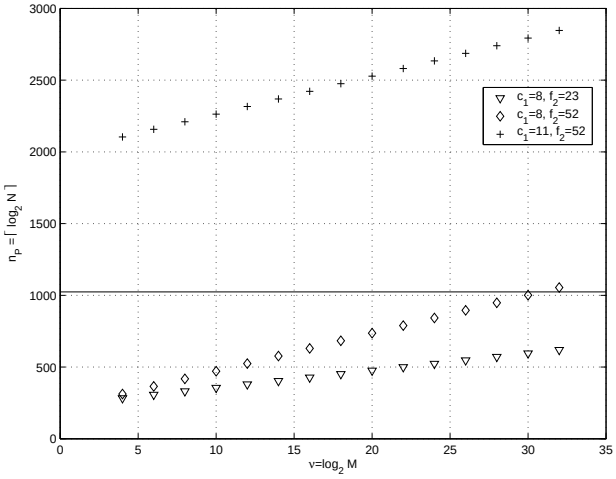
Fig. 1. Minimum value of $n_P = \lceil \log_2 N \rceil$ as a function of $\nu = \log_2 M$ for four different scenarios considering fixed point inputs: a) DFT; b) radix-2 FFT; c) radix-4 FFT. The straight line corresponds to $n_P = 1024$. $f_2 = 23$ corresponds to IEEE 754 single precision, $f_2 = 52$ corresponds to IEEE 754 double precision.



(a)



(b)



(c)

Fig. 2. Minimum value of $n_P = \lceil \log_2 N \rceil$ as a function of $\nu = \log_2 M$ for three different scenarios considering floating point inputs: a) DFT; b) radix-2 FFT; c) radix-4 FFT. The straight line corresponds to $n_P = 1024$. $c_1 = 8$ and $f_2 = 23$ correspond to IEEE 754 single precision, $c_1 = 11$ and $f_2 = 52$ correspond to IEEE 754 double precision.

modulus. The only exception is given by the radix-2 implementations using high precision coefficients, which require an extended modulus in order to cope with a number of points greater than 2^{19}. In Fig. 2, we show the minimum n_P required by three different scenarios characterized by floating point inputs. As in the previous example, the radix-2 implementation with single precision inputs requires an extended modulus only when using double precision coefficients and $M > 2^{16}$. The situation is quite different in the case of double precision inputs: due to the very high number of bits used to represent the input samples each implementation would require an extended modulus. Note that the radix-4 implementation can be used with $n_P = 1024$ even if the number of DFT points grows very large. However, above $M = 2^{30}$ even the radix-4 implementation becomes no more feasible. Hence, if some processing is required with $M > 2^{30}$ (a possible example is the processing of multidimensional signals) one has to resort to an alternative implementation (for example, the direct DFT) at the cost of a greater complexity.

B. Complexity Comparisons

Another important aspect of an encrypted domain implementation is its computational complexity. When all the implementations can be used relying on the same modulus, one can simply compare the number of modular operations required by the different approaches. However, also a different scenario can be taken into consideration, in which the modulus of the cryptosystem is set to the minimum value required by a particular implementation. Since the cost of a modular operation depends on the modulus size [22], [23], a natural question is whether a fast algorithm requiring a higher modulus size (i.e., the FFT) can be less efficient than a naive implementation requiring a lower modulus size (i.e., the direct DFT).

In order to make a complexity comparison, we made the following simplifying assumptions: 1) the cost of the algorithm is dominated by the number of exponentiations; 2) the cost of a modular exponentiation (modulo $N_{min}^2 = 2^{2n_{P,min}}$) is modeled as $C_{\mathcal{E}} = 1.5 n_2 (2n_{P,min})^2 \kappa$ [24], where κ can be interpreted as the cost of a bit operation (bit op).

Given the above hypotheses, the complexity of the different implementations can be expressed as

$$C_{DFT} = \gamma_{DFT} 2^{2\nu} n_2 n_{p,min,DFT}^2 \quad \text{bit ops} \qquad (46)$$

$$C_{R2} = \gamma_{R2} (\nu - 2) 2^{\nu} n_2 n_{p,min,R2}^2 \quad \text{bit ops} \qquad (47)$$

$$C_{R4} = \gamma_{R4} (\nu - 2) 2^{\nu} n_2 n_{p,min,R4}^2 \quad \text{bit ops} \qquad (48)$$

where $n_{p,min,DFT} = (\nu + n_1 + n_2 + 3)$, $n_{p,min,R2} = (\nu + n_1 + n_2(\nu - 2) + 3)$ and $n_{p,min,R4} = (\nu + n_1 + n_2(\nu - 2)/2 + 3)$, and the coefficients γ_{DFT}, γ_{R2}, γ_{R4}, depend on the implementation of the complex multiplications. It can be demonstrated that

$$C_{R4} < C_{R2} < C_{DFT} \quad \forall \nu \geq 3;\ \forall n_1, n_2 \geq 0; \qquad (49)$$

irrespective of the implementation of the complex multiplications. A detailed proof is given in Appendix A.

As an example in the case of a practical implementation, the complexity of the proposed FFTs is compared in Fig. 3, assuming four real multiplications for each complex multiplication. Even if both FFTs are obliged to use a large modulus size, their complexity is always well below that of a direct DFT. In this scenario, the radix-4 algorithm is always the best one for what concerns the complexity, irrespective of the other parameters.

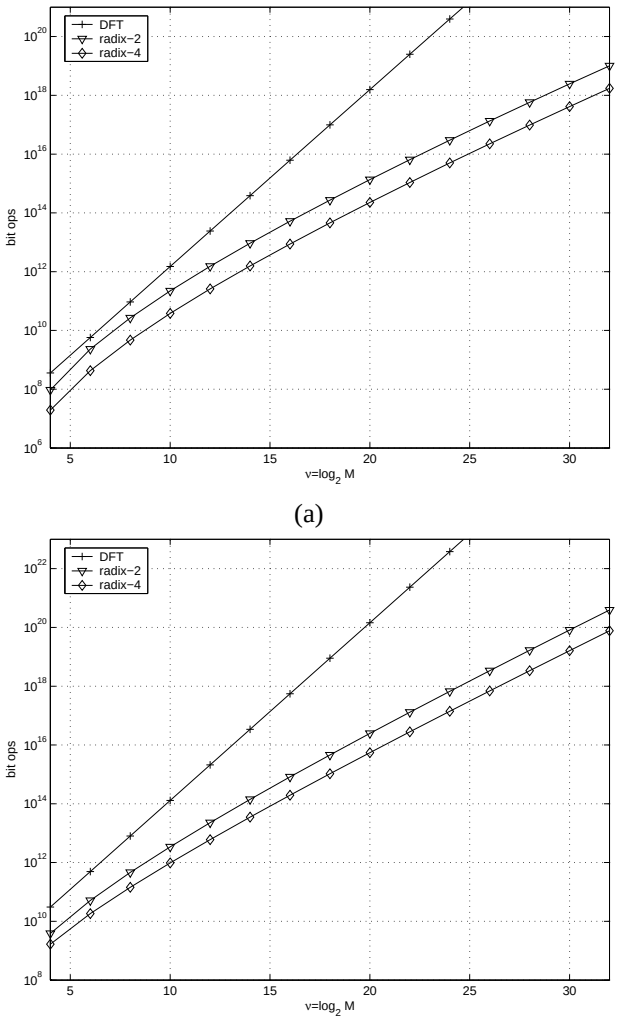


Fig. 3. Number of bit operations for three different s.p.e.d. DFT implementations, according to (46)-(48): a) $n_1 = 7$, $n_2 = \lceil 31 - \nu/2 + 1/2 \rceil$ (corresponding to 8-bit fixed point inputs and 32-bit fixed point coefficients); b) $n_1 = 254$, $n_2 = 51$ (corresponding to single precision floating point inputs and double precision floating point coefficients).

IX. Concluding Remarks

We have investigated the implementation of the DFT on a vector of encrypted samples relying on the homomorphic properties of the underlying cryptosystem. The relations between the maximum allowable DFT size and the modulus of the cryptosystem, the DFT/FFT implementation, and the required precision have been derived. The results have shown that the noise introduced by a s.p.e.d. implementation is usually smaller than in a classical fixed point implementation and comparable to a floating point one. Also the computational complexities of the different approaches have been derived and

compared, taking into account the constraints of the s.p.e.d. implementation. We considered a first scenario in which the available cryptosystem is fixed and a second scenario in which the parameters of the cryptosystem may be adapted to the requirements of the FFT. The results demonstrates that the radix-4 FFT is best suited for both scenarios.

Our approach gives useful design criteria for the implementation of s.p.e.d. modules and suggests several other issues to be addressed in future research on s.p.e.d. topics. For instance, an interesting open question is the tradeoff between feasibility and complexity, i.e., the comparison between feasible but less efficient implementations and efficient but sometimes unfeasible ones. Other topics needing further researcher are the analysis of s.p.e.d. FFT algorithms having radix greater than four, the analysis of mixed radix and split radix algorithms, and the hardware issues in practical implementations.

APPENDIX
PROOF OF EQUATION (49).

Consider the expressions of the complexity given in eqs. (46), (47), and (48). In the following, we will consider the case of four real multiplications for each complex multiplication, i.e., $\gamma_{DFT} = 24$, $\gamma_{R2} = 18$, and $\gamma_{R4} = 10.5$. The case of three real multiplications can be proved in a similar way. The following lemmas hold:

Lemma 1: $C_{R4} < C_{R2} \quad \forall \nu \geq 3;\ \forall n_1, n_2 \geq 0;$

Proof: This is immediately proved by inspecting equations (47)-(48). ■

Lemma 2: $C_{R2} < C_{DFT} \quad \forall \nu \geq 3;\ \forall n_1, n_2 \geq 0;$

Proof: Consider the functional

$$\phi(\nu, n_1, n_2) = \frac{C_{R2}}{C_{DFT}} \quad \nu \geq 3;\ n_1, n_2 \geq 0. \tag{50}$$

Clearly, $\phi(\nu, n_1, n_2) > 0$ on the given domain. Moreover, at the boundaries we have the following constraints:

$$\lim_{\nu \to \infty} \phi(\nu, n_1, n_2) = 0;$$

$$\lim_{n_1 \to \infty} \phi(\nu, n_1, n_2) = \frac{3(\nu-2)}{2^{\nu+2}} < 1, \quad \forall \nu \geq 3;$$

$$\lim_{n_2 \to \infty} \phi(\nu, n_1, n_2) = \frac{3(\nu-2)^3}{2^{\nu+2}} < 1, \quad \forall \nu \geq 3;$$

$$\phi(\nu, n_1, n_2)|_{\nu=3} = \frac{3}{2^5} < 1;$$

$$\phi(\nu, n_1, n_2)|_{n_2=0} = \frac{3(\nu-2)}{2^{\nu+2}} < 1, \quad \forall \nu \geq 3;$$

As to the hyperplane $n_1 = 0$, we have

$$\phi(\nu, n_1, n_2)|_{n_1=0} = \frac{3(\nu-2)}{2^{\nu+2}} \left(\frac{\nu + n_2(\nu-2) + 3}{\nu + n_2 + 3} \right)^2 = \psi(\nu, n_2).$$

Consider the partial derivative of $\psi(\nu, n_2)$ with respect to n_2: we have

$$\frac{\partial \psi(\nu, n_2)}{\partial n_2} = \underbrace{\frac{3(\nu-2)(\nu + n_2(\nu-2) + 3)}{2^{\nu+1}(\nu + n_2 + 3)^3}}_{\kappa(\nu, n_2)} (\nu^2 - 9) > 0, \quad \forall \nu > 3 \tag{51}$$

since $\kappa(\nu, n_2) > 0$ on the domain of $\phi(\nu, n_1, n_2)$. Hence, the maxima (and minima) of $\psi(\nu, n_2)$ lie on the boundary of the domain, from which $\phi(\nu, n_1, n_2)|_{n_1=0} < 1, \quad \forall \nu \geq 3;\ \forall n_1, n_2 \geq 0$.

As to the interior of the domain, let us consider the partial derivative of $\phi(\nu, n_1, n_2)$ with respect to n_1: we have

$$\frac{\partial \phi(\nu, n_1, n_2)}{\partial n_1} = \underbrace{\frac{3(\nu-2)(\nu + n_1 + n_2(\nu-2) + 3)}{2^{\nu+1}(\nu + n_1 + n_2 + 3)^3}}_{\kappa'(\nu, n_1, n_2)} n_2(3-\nu) < 0, \tag{52}$$

since $\kappa'(\nu, n_1, n_2) > 0$ on the given domain. Hence, also the extrema of $\phi(\nu, n_1, n_2)$ lie on the boundary of the domain. This implies

$$\phi(\nu, n_1, n_2) < 1, \quad \forall \nu \geq 3;\ \forall n_1, n_2 \geq 0$$

which demonstrates the lemma. ■

The proof of equation (49) follows from the above two lemmas.

REFERENCES

[1] Z. Erkin, A. Piva, S. Katzenbeisser, R. L. Lagendijk, J. Shokrollahi, G. Neven, and M. Barni, "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.

[2] R. Rivest, L. Adleman, and M. Dertouzos, "On data banks and privacy homomorphisms," in *Foundations of Secure Computation*, R. D. et al., Ed. New York: Academic Press, 1978, pp. 169–179.

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

[4] A. C. Yao, "Protocols for secure computations," in *Proceedings of Twenty-third IEEE Symposium on Foundations of Computer Science*, Chicago, Illinois, November 1982, pp. 160–164.

[5] O. Goldreich, S. Micali, and A. Wigderson, "How to play ANY mental game," in *Proceedings of the nineteenth annual ACM conference on Theory of computing*. New York, New York, United States: ACM, 1987, pp. 218–229.

[6] R. Cramer, I. Damgård, and J. B. Nielsen, "Multiparty computation from threshold homomorphic encryption," *Lecture Notes in Computer Science*, vol. 2045, pp. 280–300, 2001.

[7] B. Schoenmakers and P. Tuyls, "Efficient binary conversion for Paillier encrypted values," in *Advances in Cryptology - EUROCRYPT 2006*. Berlin / Heidelberg: Springer, July 2006, pp. 522–537.

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

[9] C.-L. Lei, P.-L. Yu, P.-L. Tsai, and M.-H. Chan, "An efficient and anonymous buyer-seller watermarking protocol," *IEEE Transactions on Image Processing*, vol. 13, no. 12, pp. 1618–1626, December 2004.

[10] 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, December 2005.

[11] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Eurocrypt '99: Advances in Cryptology, Lecture Notes in Computer Science Volume 1592*. Springer-Verlag, 1999, pp. 223–238.

[12] I. Damgård and M. Jurik, "A generalisation, a simplification and some applications of Paillier's probabilistic public-key system." in *Public Key Cryptography*, 2001, pp. 119–136.

[13] M. Vetterli and P. Duhamel, "Split-radix algorithms for length-p^m DFT's," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 37, no. 1, pp. 57–64, Jan. 1989.

[14] H. Sorensen, D. Jones, M. Heideman, and C. Burrus, "Real-valued fast Fourier transform algorithms," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. 35, no. 6, pp. 849–863, June 1987.

[15] A. V. Oppenheim and R. W. Schafer, *Digital Signal Processing*. Prentice-Hall International, Inc., 1975.

[16] L. R. Rabiner and B. Gold, *Theory and Application of Digital Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1975.

[17] P. D. Welch, "A fixed-point fast Fourier transform error analysis," *IEEE Trans. Audio Electroacoust.*, vol. AU-17, no. 2, pp. 151–157, June 1969.

[18] A. V. Oppenheim and C. J. Weinstein, "Effects of finite register length in digital filtering and the fast Fourier transform," *Proc. IEEE*, vol. 60, no. 8, pp. 957–976, Aug. 1972.
[19] D. V. James, "Quantization errors in the fast Fourier transform," *IEEE Trans. Acoust., Speech, Signal Process.*, vol. ASSP-23, no. 3, pp. 277–283, June 1975.
[20] E. Savas and Çetin Kaya Koç, "The Montgomery modular inverse – revisited," *IEEE Trans. Comput.*, vol. 49, no. 7, pp. 763–766, July 2000.
[21] M. E. Kaihara and N. Takagi, "A hardware algorithm for modular multiplication/division," *IEEE Trans. Comput.*, vol. 54, no. 1, pp. 12–21, Jan. 2005.
[22] Çetin Kaya Koç, T. Acar, and B. S. Kalinski, "Analyzing and comparing Montgomery multiplication algorithms," *IEEE Micro*, vol. 16, no. 3, pp. 26–33, June 1996.
[23] B. S. Kalinski, "The Montgomery inverse and its applications," *IEEE Trans. Comput.*, vol. 44, no. 8, pp. 1064–1065, Aug. 1995.
[24] D. E. Knuth, *The art of computer programming, volume 2 (3rd ed.): seminumerical algorithms*. Boston, MA, USA: Addison-Wesley Longman Publishing Co., Inc., 1997.

Tiziano Bianchi (S'04-M'06) was born in Prato, Italy, in 1976. He received the M.Sc. degree (Laurea) in electronic engineering in 2001 and the Ph.D. degree in information and telecommunication engineering in 2005, both from the University of Florence, Italy.

From March 2005 he is with the Department of Electronics and Telecommunications of the University of Florence as a Research Assistant. His research interests have involved applications of multirate systems, signal processing in communications, and ultra-wideband systems. Current research topics include signal processing in the encrypted domain and processing of SAR images.

Alessandro Piva graduated cum laude in Electronic Engineering from University of Florence on 1995. He obtained the Ph. D. degree in "Computer Science and Telecommunications Engineering" from the University of Florence on 1999. From 2002 until 2004 he was Research Scientist at the National Inter-University Consortium for Telecommunications. He is at present Assistant Professor at the University of Florence. His current research interests are the technologies for Multimedia content security, and image processing techniques for Cultural Heritage field. He is co-author of more than 100 papers published in international journals and conference proceedings. He holds 3 Italian patents and an International one regarding watermarking. He is Person responsible for University of Florence of the European Project SPEED.

Mauro Barni (SM'06) graduated in electronic engineering at the University of Florence in 1991. He received the PhD in informatics and telecommunications in October 1995. He has carried out his research activity for over 15 years first at the Department of Electronics and Telecommunication of the University of Florence, then at the Department of Information Engineering of the University of Siena where he works as associate Professor. During the last decade he has been studying the application of image processing techniques to copyright protection and authentication of multimedia (digital watermarking). He is author/co-author of about 180 papers published in international journals and conference proceedings, an holds three patents in the field of digital watermarking. He is co-author of the book "Watermarking Systems Engineering: Enabling Digital Assets Security and other Applications", published by Dekker Inc. in February 2004. He is the editor in chief of the EURASIP Journal on Information Security. He serves as associate editor of the IEEE Trans. on Circuits and system for Video Technology, the IET Proceedings on Information Security.

Prof. Barni is a member of the IEEE Information Forensic and Security technical Committee (IFS-TC) of the IEEE Signal Processing Society. He is a senior member of the IEEE and EURASIP.