# IMPLEMENTING THE DISCRETE FOURIER TRANSFORM IN THE ENCRYPTED DOMAIN

*Tiziano Bianchi, Alessandro Piva*[*]

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

*Mauro Barni*

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

## ABSTRACT

Signal processing modules working directly on the encrypted data could provide an elegant solution to application scenarios where valuable signals should 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 both the DFT and radix-2 fast Fourier transform, including the error analysis and the maximum size of the sequence that can be transformed.

***Index Terms***— Discrete Fourier transforms, cryptography, error analysis, homomorphic encryption, signal processing in the encrypted domain

## 1. INTRODUCTION

Signal processing in the encrypted domain (s.p.e.d.), i.e, the availability of signal processing modules that work directly on the encrypted data, would be of great help for application scenarios where "valuable" signals must be processed. Two recent examples regard zero-knowlegde watermark detection [1], or privacy preserving analysis of personal data [2].

From the signal processing point of view, one of the most extensively used modules is the discrete Fourier transform (DFT), defined as

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

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

The aim of our work is to provide a s.p.e.d. implementation of the above expression. We will consider a scenario in which the transform device is fed with a sample-wise encrypted version of the input vector. 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

$$D[\phi(E[a], E[b])] = a + b \qquad (2)$$

where $E[\cdot]$ and $D[\cdot]$ denote the encryption and decryption operators. With such a cryptosystem it is indeed possible to add two encrypted values without first decrypting them. Moreover, it is possible to multiply an encrypted value by a public integer value by repeatedly applying the operator $\phi(\cdot,\cdot)$.

Another property of the above cryptosystem we assume is that it is *probabilistic*, that is, given two encrypted values it is not possible to decide if they conceal the same value. This is fundamental, since the alphabet to which the input samples belong is usually limited, and a non-probabilistic 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 [3], for which the operator $\phi(\cdot,\cdot)$ is a modular multiplication[1].

Since the DFT transform coefficients are public, the expression in (1) can be computed with an encrypted input vector by relying on the homomorphic property. However, some issues need to be addressed.

First of all, both the input samples and the DFT coefficients need to be represented as integer values. Secondly, FFT like algorithms should be applicable also in the encrypted domain. Finally, encrypted values can not be scaled relying on homomorphic computations. Since many practical homomorphic cryptosystems are based on modular operations on a finite field/ring, we must ensure that both input and output values do not wrap around in the modular representation.

In this paper, we will provide solutions to the above issues. A convenient signal model 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. Particular attention will be devoted to the quantization error

[*]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.

[1]With the Paillier cryptosystem, the following s.p.e.d. operations are defined: $D[E[a] \cdot E[b]] = a + b$; $D[E[a] \cdot E[b]^{-1}] = a - b$; $D[E[a]^b] = ab$.

introduced by the s.p.e.d. implementation and to the maximum size of the sequence that can be transformed.

## 2. SIGNAL MODEL FOR S.P.E.D.

Let us consider a signal $x(n) \in \mathbb{C}$, $n = 0, \ldots, M-1$, with $x(n) = x_R(n) + jx_I(n)$, $x_{R,I} \in \mathbb{R}$. In the following, we will assume $|x(n)| \le 1$, from which $|x_{R,I}(n)| \le 1$.

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

$$s(n) = \lceil Q_1 x_R(n) \rfloor + j \lceil Q_1 x_I(n) \rfloor = s_R(n) + js_I(n) \quad (3)$$

where $\lceil \cdot \rfloor$ is the rounding function and $Q_1$ is a suitable scaling factor. In the following, we will assume that the quantization scaling factor is an integer. Based on the properties of $x(n)$, the quantized signal will satisfy $-Q_1 \le s_{R,I}(n) \le Q_1$. If the cryptosystem encrypts integers modulo $N$, this means that the scaling factor must satisfy $N \ge 2Q_1 + 1$, so that there is a one-to-one mapping between $s_{R,I}(n) \mod N$ and $s_{R,I}(n)$.

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

$$\begin{aligned} C(r) =& \left\lceil Q_2 \cos\left(\frac{2\pi r}{M}\right) \right\rfloor - j \left\lceil Q_2 \sin\left(\frac{2\pi r}{M}\right) \right\rfloor \\ =& C_R(r) + jC_I(r) \end{aligned} \quad (4)$$

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

Based on the above model, the s.p.e.d. DFT is defined as

$$S(k) = \sum_{n=0}^{M-1} C(nk)s(n), \;\; k = 0, \ldots, M-1. \quad (5)$$

Since all computations are between integers, the expression above can be evaluated in the encrypted domain by relying on the homomorphic properties. If the inputs are encrypted with the Paillier cryptosystem, the s.p.e.d. DFT is given as

$$E[S(k)] = \prod_{n=0}^{M-1} E[s(n)]^{C(nk)}, \;\; k = 0, \ldots, M-1 \quad (6)$$

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

## 3. MAGNITUDE REQUIREMENTS

The computation of the DFT using (5) requires two problems to be tackled with. 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 (5) using a cryptosystem which encrypts integers modulo $N$, one must ensure that $S(k)$ is always recoverable from $S(k) \mod N$. Hence, according to the proposed model, one has to find an *upper bound* on $S(k)$ such that $|S_{R,I}(k)| \le Q_S$, and verify that $N \ge 2Q_S + 1$.

In general, $S(k)$ can be expressed as

$$S(k) = KX(k) + \epsilon_S(k). \quad (7)$$

Based on the above equation, the desired DFT output can be estimated as $\tilde{X}(k) = S(k)/K$, and the upper bound is

$$Q_S = \lfloor MK + \epsilon_{S,max} \rfloor . \quad (8)$$

The value of both $K$ and $\epsilon_{S,max}$ will depend 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.

### 3.1. Direct Computation

Let us express $s(n) = Q_1 x(n) + \epsilon_s(n)$ and $C(r) = Q_2 W^r + \epsilon_W(r)$. If the DFT is directly computed by applying (5), then

$$\begin{aligned} S(k) =& Q_1 Q_2 X(k) + \sum_{n=0}^{M-1} Q_1 x(n) \epsilon_W(nk) \\ &+ \sum_{n=0}^{M-1} Q_2 \epsilon_s(n) W^{nk} + \sum_{n=0}^{M-1} \epsilon_s(n) \epsilon_W(nk). \end{aligned} \quad (9)$$

The scaling factor is $K = Q_1 Q_2$. As to the upper bound, due to the properties of the rounding function, $|\epsilon_{s,W}(n)| \le 1/\sqrt{2}$. Hence $|s(n)| \le Q_1 + 1/\sqrt{2}$, $|C(r)| \le Q_2 + 1/\sqrt{2}$ and, after simple manupulations,

$$|S(k)| \le M \left( Q_1 Q_2 + \frac{Q_1}{\sqrt{2}} + \frac{Q_2}{\sqrt{2}} + \frac{1}{2} \right) \quad (10)$$

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

### 3.2. Decimation in Time Radix-2 FFT

This algorithm is applied when $M = 2^\nu$ and allows the DFT to be computed in $\nu$ stages. At each stage, a new pair of coefficients is obtained as a linear combination of the corresponding old pair of coefficients, using a so-called *butterfly* structure [4]. By applying the proposed model, the s.p.e.d. radix-2 butterfly can be obtained as

$$S^{(m+1)}(p) = Q_2 S^{(m)}(p) + C(r) S^{(m)}(q) \quad (11)$$
$$S^{(m+1)}(q) = Q_2 S^{(m)}(p) - C(r) S^{(m)}(q). \quad (12)$$

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.

As to the upper bound analysis, the two branches of the butterfly are equivalent. Let us consider the first branch. If we express $S^{(m)}(p) = K^{(m)} X^{(m)}(p) + \epsilon_S^{(m)}(p)$, then

$$\begin{aligned} S^{(m+1)}(p) =& Q_2 K^{(m)} \left( X^{(m)}(p) + W^r X^{(m)}(q) \right) \\ &+ Q_2 \left( \epsilon_S^{(m)}(p) + W^r \epsilon_S^{(m)}(q) \right) \\ &+ K^{(m)} X^{(m)}(q) \epsilon_W(r) + \epsilon_S^{(m)}(q) \epsilon_W(r) \end{aligned} \quad (13)$$

from which the following recursions can be derived

$$K^{(m+1)} = Q_2 K^{(m)} \tag{14}$$

$$\begin{aligned}\epsilon_S^{(m+1)}(p) = {} & Q_2\left(\epsilon_S^{(m)}(p) + W^r \epsilon_S^{(m)}(q)\right) \\ & + K^{(m)} X^{(m)}(q)\epsilon_W(r) + \epsilon_S^{(m)}(q)\epsilon_W(r).\end{aligned} \tag{15}$$

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 $\epsilon_S^{(0)}(p) = \epsilon_s(\tilde{p})$. Moreover, up to the second stage we have $W^r \in \{1, j\}$. Hence, the multiplication by $W_r$ does not require any integer multiplication at all, so that no scaling factor is introduced and $\epsilon_W = 0$. Therefore, $K^{(2)} = Q_1$ and by using (14), it is easy to derive the scale factor as $K = K^{(\nu)} = Q_1 Q_2^{\nu-2}$.

As to the upper bound, we consider an equivalent recursive relation on an upper bound of the quantization error

$$|\epsilon_S^{(m+1)}| \le \left(2Q_2 + \frac{1}{\sqrt{2}}\right)|\epsilon_S^{(m)}| + \frac{2^m}{\sqrt{2}} K^{(m)}. \tag{16}$$

Note that for the first two stages the above expression simplifies as $|\epsilon_S^{(m+1)}| \le 2|\epsilon_S^{(m)}|$, since $\epsilon_W = 0$. Hence, by using as initial condition $|\epsilon_S^{(2)}| \le 4/\sqrt{2}$ in (16) , the final upper bound can be expressed as

$$\begin{aligned}|\epsilon_S^{(\nu)}| \le {} & \frac{4}{\sqrt{2}}\left(2Q_2 + \frac{1}{\sqrt{2}}\right)^{\nu-2} \\ & + \sum_{k=0}^{\nu-3} \frac{2^{\nu-1-k}}{\sqrt{2}} Q_1 Q_2^{\nu-3-k}\left(2Q_2 + \frac{1}{\sqrt{2}}\right)^k = \epsilon_{S,max}\end{aligned} \tag{17}$$

from which we derive the upper bound on $S(k)$ as $Q_S = M Q_1 Q_2^{\nu-2} + \epsilon_{S,max}$.

## 4. ERROR ANALYSIS

Since in the s.p.e.d. DFT there is no rescaling after multiplication, there is no computational noise, and the error is due only 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 $\epsilon_W(r)$ are i.i.d. variables with zero mean and variance $\sigma_q^2$ and $\sigma_{W,q}^2$, respectively. Although $\epsilon_W(r)$ is deterministic, this will produce a simple estimate of the quantization noise.

As to the input signal, its NSR can be estimated as $\eta = \sigma_q^2 / Q_1^2 \sigma_x^2$ where $\sigma_x^2$ indicates the signal power. If we consider the DFT computation, the NSR can be estimated as $\tilde{\eta} = \sigma_{\epsilon_S}^2 / K^2 M \sigma_x^2$. In the case of the direct DFT computation, relying on equation (9) and neglecting the terms $\epsilon_s(n)\epsilon_W(nk)$, we can estimate $\sigma_{\epsilon_S}^2 \approx M Q_1^2 \sigma_x^2 \sigma_{W,q}^2 + M Q_2^2 \sigma_q^2$. Hence

$$\tilde{\eta}_D = \eta + \frac{\sigma_{W,q}^2}{Q_2^2}. \tag{18}$$

In the case of a radix-2 FFT, the variance of the error at the $m$th stage can be recursively approximated as

$$\sigma^2_{\epsilon_S^{(m+1)}} \approx 2Q_2^2 \sigma^2_{\epsilon_S^{(m)}} + 2^m \sigma_x^2 \sigma_{W,q}^2 Q_1^2 Q_2^{2(m-2)}. \tag{19}$$

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

$$\tilde{\eta}_{R2} = \eta + \frac{\nu-2}{2}\frac{\sigma_{W,q}^2}{Q_2^2}. \tag{20}$$

Since the value of $Q_1$, and hence $\eta$, is 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.

According to the way a plaintext FFT is implemented, two cases need to be analyzed: 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 $\tilde{\eta} \le 4M \cdot 2^{-2b_2}/3\sigma_x^2$ [4], where $\sigma_x^2$ is the power of the input signal, assumed white. If we assume $\sigma_{W,q}^2 = 1/6$, we have

$$Q_2 \ge \begin{cases} \sigma_x \cdot 2^{b_2 - \nu/2 - 3/2} & \text{DFT} \\ \sigma_x \sqrt{\nu-2} \cdot 2^{b_2 - \nu/2 - 2} & \text{FFT.} \end{cases} \tag{21}$$

As a consequence, the s.p.e.d. implementation will require less bits for quantizing the twiddle factors with respect to a classical fixed 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 $\tilde{\eta} \le 2\nu \cdot 2^{-2f_2}/3$ [4]. Hence

$$Q_2 \ge \begin{cases} \sqrt{1/4\nu} \cdot 2^{f_2} & \text{DFT} \\ \sqrt{(\nu-2)/8\nu} \cdot 2^{f_2} & \text{FFT.} \end{cases} \tag{22}$$

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

## 5. PRACTICAL EXAMPLES

Consider a scenario in which a set of encrypted data must undergo different processing tasks. It is reasonable to assume that the data are encrypted once, and that each processing task employs the same set of encrypted data. Therefore, each processing task must rely on an implementation satisfying the requirement on the modulus.

In the following, we will assume that each s.p.e.d. implementation fulfills the same requirements in terms of both

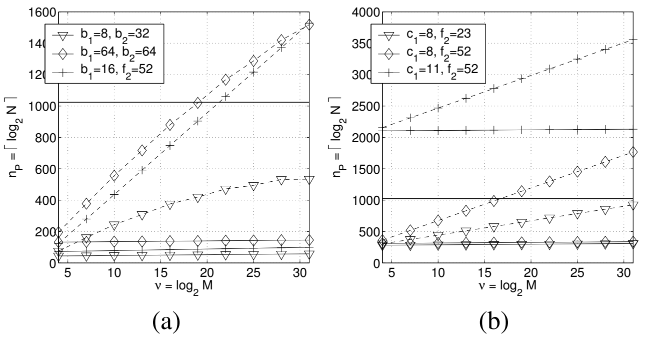

**Fig. 1**. Minimum value of $n_P = \lceil \log_2 N \rceil$ as a function of $\nu = \log_2 M$ for different scenarios: a) fixed point inputs; b) floating point inputs. Solid line is DFT. Dashed line is FFT. The straight line corresponds to $n_P = 1024$. $c_1 = 8$ and $f_2 = 23$ correspond to IEEE 754 single precision, $c_1 = 11$ and $f_2 = 52$ correspond to IEEE 754 double precision.

input NSR and output NSR as the plaintext version. Moreover, we will assume $Q_1 = 2^{n_1}$ and $Q_2 = 2^{n_2}$. Finally, for security reasons, we will assume that the modulus used by Paillier satisfies $n_P = \lceil \log_2 N \rceil = 1024$.

Given (10), a simple design strategy ensuring that no wrap around occurs in the internal computations of DFT is

$$N \geq 4\,(2^{\nu} Q_1 Q_2) \qquad (23)$$

which holds for every $Q_1, Q_2 \geq 2$. The above bound is satisfied by requiring $\lfloor \log_2 N \rfloor \geq \lceil \log_2 (2^{\nu+2} Q_1 Q_2) \rceil$ or

$$n_P \geq \nu + n_1 + n_2 + 3. \qquad (24)$$

As to the FFT, a similar bound can be derived as

$$N \geq 2\left(2^{\nu} Q_1 Q_2^{\nu-2} + \xi\right) + 1 \qquad (25)$$

where $\xi$ can be deduced from equation (17). Unless $Q_2$ is very small, it is safe to assume $\xi < 2^{\nu} Q_1 Q_2^{\nu-2} - 1/2$, so that the above bound is satisfied by requiring

$$n_P \geq \nu + n_1 + (\nu - 2) n_2 + 3. \qquad (26)$$

For a fixed $n_2$, both the above constraints depend linearly on $\nu$, i.e., $n_P \geq \alpha\nu + \beta$, with $\alpha = 1$ for the DFT and $\alpha = 1 + n_2$ for the FFT. Hence, the maximum allowable FFT size mainly depends on $n_2$ (Asymptotically, $\nu_{FFT,max} \approx n_P/n_2$).

We will consider different scenarios, according to whether the inputs and the twiddle factors of the original FFT are either fixed point or floating point values: *a) Fixed point inputs*: if the input is quantized using $b_1$ bits, its values can be mapped onto integer values in the interval $[-2^{b_1-1}, 2^{b_1-1} - 1]$, so that we can assume $n_1 = b_1 - 1$. *b) 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 bits of the exponent. Unless the properties of the input signal are known, this requires $n_1 = 2^{c_1} - 2$; *c) Fixed point coefficients*: based on (21), a convenient choice is $n_2 = \lceil b_2 - \nu/2 + 1/2 \rceil$; *d) Floating point coefficients*: based on (22), a convenient choice is $n_2 = f_2 - 1$.

According to the considered scenario, one can choose the convenient values of $n_1$ and $n_2$ and substitute them into (24)-(26) in order to verify the requirements of the cryptosystem. In Fig. 1, we show the minimum $n_P$ required by six different scenarios characterized by either fixed point or floating point inputs. If the number of FFT point is not very high, most of the scenarios can rely on the 1024-bit modulus. However, when either the twiddle factors or the inputs require a high precision quantization, the allowable FFT size can be quite limited. A simple solution could be to use a direct DFT implementation, which has less stringent requirements on the cryptosystem modulus, at the cost of a greater complexity.

## 6. 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. It has been shown how the maximum allowable DFT size depends on the modulus of the cryptosystem, on the DFT/FFT implementation, and on the required precision. The results have also 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. 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.

## 7. REFERENCES

[1] A. Adelsbach and A.-R. Sadeghi. Zero-knowledge watermark detection and proof of ownership. In *Information Hiding*, pages 273–288, 2001.

[2] M. Barni, C. Orlandi, and A. Piva. A privacy-preserving protocol for neural-network-based computation. In *MM&Sec '06: Proceeding of the 8th workshop on Multimedia and security*, pages 146–151, New York, NY, USA, 2006. ACM Press.

[3] P. Pailler. Public-key cryptosystems based on composite degree residuosity classes. In *Proceedings of Eurocrypt'99, Lecture Notes is Computer Science vol. 1592*, pages 223–238. Springer-Verlag, 1999.

[4] Alan V. Oppenheim and Ronald W. Schafer. *Digital Signal Processing*. Prentice-Hall International, Inc., 1975.