16th European Signal Processing Conference (EUSIPCO 2008), Lausanne, Switzerland, August 25-29, 2008, copyright by EURASIP

COMPARISON OF DIFFERENT FFT IMPLEMENTATIONS IN THE ENCRYPTED DOMAIN

Tiziano Bianchi[1]*, Alessandro Piva*[1]*, and Mauro Barni*[2]

Dipartimento di Elettronica e Telecomunicazioni (1)
Università di Firenze
Via S. Marta 3, 50139, Firenze, Italy
e-mail: {bianchi, piva}@lci.det.unifi.it

Dipartimento di Ingegneria dell'Informazione (2)
Università di Siena
Via Roma 56, 53100, Siena, Italy
e-mail: barni@dii.unisi.it

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 compare different implementations of the discrete Fourier transform (DFT) in the encrypted domain. Both radix-2 and radix-4 fast Fourier transforms (FFTs) will be defined using the homomorphic properties of the underlying cryptosystem. We derive the maximum size of the sequence that can be transformed by using the different implementations and we provide computational complexity analyses and comparisons. The results show that the radix-4 FFT is best suited for an encrypted domain implementation.

1. INTRODUCTION

A large variety of new applications, ranging from multimedia content distribution to advanced healthcare systems for continuous health monitoring, have been made possible by recent advances in signal processing. 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 simply build a secure cryptographic layer on top of the signal processing modules, by assuming that the involved parties or devices trust each other: the cryptography layer is used only to protect the data against third parties or to provide authenticity. Unfortunately, this may not be sufficient for the envisaged applications, since the owner of the data may not trust the processing devices.

The availability of signal processing modules that work directly on the encrypted data would be a powerful tool in application scenarios where "valuable" signals must be produced, processed or exchanged in digital format. Some recent examples regards zero-knowlegde watermark detection [1, 2], where a watermark detector reveals the presence of a watermark without learning anything about the detection process or the watermark itself, or buyer-seller protocols [3, 4], where a watermark is embedded in such a way that the seller does not have access to the watermarked copy of a content sold to a buyer.

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.

Surely, if a signal processing in the encrypted domain (s.p.e.d.) framework has to be defined, one of the fundamental modules should be 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. Among the appealing properties of the above transform one is that it can be implemented via fast algorithms, noted as fast Fourier transforms (FFTs).

The aim of our work is to provide and compare different s.p.e.d. implementations of the above expression. We will consider a scenario in which the transform processor is fed with a sample-wise encrypted version of the input vector. In order to make possible linear computations on encrypted values, we 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 should not be possible to decide if they conceal the same value. This is fundamental, since the alphabeth 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 requirements is the Paillier cryptosystem [5], for which the operator $\phi(\cdot,\cdot)$ is a modular multiplication.

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, the homomorphic property permits only a very limited set of operations to be performed on encrypted values. If we consider a fixed point representation, the result of each multiplication should be either rounded or truncated. In the encrypted domain, such operations are not possible relying only

on homomorphic computations, but require the use of interactive and computationally expensive protocols [6]. Since the s.p.e.d. DFT is a basic building block, it would be preferable to devise solutions that avoid interaction, so as to keep the s.p.e.d. DFT as simple as possible. Finally, FFT like algorithms should be applicable also in the encrypted domain, in order to achieve similar computational savings. As to the security of the s.p.e.d. DFT, this follows immediately from the security of the underlying cryptosystem: the proposed approach does not modify the underlying cryptosystem, but merely exploits some of its properties.

In this paper, a convenient signal model for s.p.e.d. will be proposed, allowing us to define both a radix-2 and a radix-4 s.p.e.d. FFT that require no interation between the involved parties. We will analyze the maximum size of the sequence that can be transformed by using the proposed s.p.e.d. implementations and we will provide a computational complexity analysis, taking into account the requirements of different possible s.p.e.d. scenarios.

2. SIGNAL MODEL FOR THE ENCRYPTED DOMAIN

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 $|x(n)| \leq 1$, 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

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

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. 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 $s_{R,I}(n)$ and $s_{R,I}(n) \mod N$. This can be achieved by imposing $N \geq 2Q_1 + 1$.

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

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

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

3. 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 new pair of coefficients is obtained as a linear combination of the corresponding old pair of coefficients, using the well-known *butterfly* structure [7]. A s.p.e.d. butterfly can be obtained by applying the proposed model and the homomorphic property. First, an integer valued butterfly is obtained as

$$S^{(m+1)}(p) = Q_2 S^{(m)}(p) + C(r) S^{(m)}(q) \qquad (5)$$

$$S^{(m+1)}(q) = Q_2 S^{(m)}(p) - C(r) S^{(m)}(q). \qquad (6)$$

Note that in the proposed model a unit coefficient is quantized as Q_2. Hence, the integer implementation of the FFT algorithm requires $M/2$ additional integer multiplications at each stage.

The computation of the radix-2 FFT using (5)-(6) requires two problems to be tackled with. The first one is that there will be a scaling factor between $S^{(\nu)}(k) = S(k)$ and the desired value $X(k)$. The second one is that, if a cryptosystem which encrypts integers modulo N is used, one must ensure that there is a one-to-one mapping between $S(k) \mod N$ and $S(k)$. Hence, according to the proposed model, one has to find an *upper bound* on $S(k)$ such that $|S_{R,I}(k)| \leq Q_S$, and verify that $N \geq 2Q_S + 1$.

Let us express $s(n) = Q_1 x(n) + \varepsilon_s(n)$ and $C(r) = Q_2 W^r + \varepsilon_W(r)$, where $\varepsilon_s(n)$ and $\varepsilon_W(r)$ denote the quantization errors on the input signal and the transform coefficients. We will show that the integer FFT output $S(k)$ can be always expressed as

$$S(k) = KX(k) + \varepsilon_S(k) \qquad (7)$$

where K is an overall scale factor and $\varepsilon_S(k)$ models the propagation of the quantization errors. Based on the above equation, the desired DFT output can be estimated as $\tilde{X}(k) = S(k)/K$, and the upper bound is

$$Q_S = MK + \varepsilon_{S,max}. \qquad (8)$$

Since the two branches of the butterfly are equivalent, we can consider without loss of generality the first branch. If we express $S^{(m)}(p) = K^{(m)} X^{(m)}(p) + \varepsilon_S^{(m)}(p)$, then we have

$$\begin{aligned} S^{(m+1)}(p) =& Q_2 K^{(m)} \left(X^{(m)}(p) + W^r X^{(m)}(q) \right) \\ &+ Q_2 \left(\varepsilon_S^{(m)}(p) + W^r \varepsilon_S^{(m)}(q) \right) \\ &+ K^{(m)} X^{(m)}(q) \varepsilon_W(r) + \varepsilon_S^{(m)}(q) \varepsilon_W(r) \end{aligned} \qquad (9)$$

from which we derive the following recursive relations

$$K^{(m+1)} = Q_2 K^{(m)} \qquad (10)$$

$$\begin{aligned} \varepsilon_S^{(m+1)}(p) =& Q_2 \left(\varepsilon_S^{(m)}(p) + W^r \varepsilon_S^{(m)}(q) \right) \\ &+ K^{(m)} X^{(m)}(q) \varepsilon_W(r) + \varepsilon_S^{(m)}(q) \varepsilon_W(r). \end{aligned} \qquad (11)$$

At the first stage $S^{(0)}(n) = s(\tilde{n}) = Q_1 x(\tilde{n}) + \varepsilon_s(\tilde{n})$, where $\tilde{n}$ indicates n in bit reverse order, so that the recursion starts with $K^{(0)} = Q_1$ and $\varepsilon_S^{(0)}(p) = \varepsilon_s(\tilde{p})$. Since at the first two stages $W^r \in \{1, j\}$, 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 (10), 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, given as

$$|\varepsilon_S^{(m+1)}| \leq \left(2Q_2 + \frac{1}{\sqrt{2}} \right) |\varepsilon_S^{(m)}| + \frac{2^m}{\sqrt{2}} K^{(m)}. \qquad (12)$$

In the case of the first two stages the above expression simplifies as $|\varepsilon_S^{(m+1)}| \leq 2|\varepsilon_S^{(m)}|$, since $\varepsilon_W = 0$. Hence, by using as

initial condition $|\varepsilon_S^{(2)}| \leq 4/\sqrt{2}$ in (12) (since $|\varepsilon_S^{(0)}| \leq 1/\sqrt{2}$), the final upper bound can be expressed as

$$|\varepsilon_S^{(\nu)}| \leq \frac{4}{\sqrt{2}}\left(2Q_2+\frac{1}{\sqrt{2}}\right)^{\nu-2} + \sum_{k=0}^{\nu-3}\frac{2^{\nu-1-k}}{\sqrt{2}}Q_1Q_2^{\nu-3-k}\left(2Q_2+\frac{1}{\sqrt{2}}\right)^{k} = \varepsilon_{S,R2,max} \tag{13}$$

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

4. 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. At each stage, four new coefficients are obtained as a linear combination of four old coefficients using the following *radix-4 butterfly* [7]

$$X^{(m+1)}(p_k) = \sum_{i=0}^{3} X^{(m)}(p_i)W^{ri}(-j)^{ik}, \quad k = 0,\ldots,3. \tag{14}$$

Without loss of generality, the upper bound can be evaluated considering the integer version of the first branch:

$$\begin{aligned} S^{(m+1)}(p_0) =& Q_2S^{(m)}(p_0) + C(r)S^{(m)}(p_1) \\ &+ C(2r)S^{(m)}(p_2) + C(3r)S^{(m)}(p_3). \end{aligned} \tag{15}$$

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

$$K^{(m+1)} = Q_2K^{(m)} \tag{16}$$

$$|\varepsilon_S^{(m+1)}| \leq \left(4Q_2+\frac{3}{\sqrt{2}}\right)|\varepsilon_S^{(m)}| + 3\frac{4^m}{\sqrt{2}}K^{(m)}. \tag{17}$$

Moreover, at the first stage of the radix-4 FFT algorithm $W^r = 1$, so that the recursion begins with $K^{(1)} = Q_1$ and $|\varepsilon_S^{(1)}| \leq 4/\sqrt{2}$. The error upper bound for radix-4 FFT is

$$|\varepsilon_S^{(\mu)}| \leq \frac{4}{\sqrt{2}}\left(4Q_2+\frac{3}{\sqrt{2}}\right)^{\mu-1} + \sum_{k=0}^{\mu-2}\frac{4^{\mu-1-k}}{\sqrt{2}}Q_1Q_2^{\mu-2-k}\left(4Q_2+\frac{3}{\sqrt{2}}\right)^{k} = \varepsilon_{S,R4,max} \tag{18}$$

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

5. DIRECT IMPLEMENTATION

To make comparisons, we can also consider a direct computation according to the DFT definition. By applying the proposed model, it is easy to obtain

$$\begin{aligned} S(k) =& Q_1Q_2X(k) + \sum_{n=0}^{M-1} Q_1x(n)\varepsilon_W(nk) \\ &+ \sum_{n=0}^{M-1} Q_2\varepsilon_s(n)W^{nk} + \sum_{n=0}^{M-1}\varepsilon_s(n)\varepsilon_W(nk). \end{aligned} \tag{19}$$

The scaling factor is $K = Q_1Q_2$. As to the upper bound, after simple manupulations we have

$$|\varepsilon_S| \leq M\frac{Q_1}{\sqrt{2}} + M\frac{Q_2}{\sqrt{2}} + \frac{M}{2} = \varepsilon_{S,DFT,max} \tag{20}$$

from which we derive $Q_S = MQ_1Q_2 + \varepsilon_{S,DFT,max}$.

6. COMPLEXITY ANALYSIS

The complexity of the proposed FFT implementation in the encrypted domain depends on several parameters, including the used cryptosystem, its homomorphic properties, and the scaling factor Q_2. In this paper we 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, usually implemented as $ab^{-1} \mod n$. The same holds for exponentiations to negative exponents, usually implemented as $(a^{-e})^{-1} \mod n$. 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 by a s.p.e.d. algorithm.

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, we consider an implementation requiring four real multiplications and two real additions, i.e., $sC = \{s_RC_R - s_IC_I, s_RC_I + s_IC_R\}$.

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 have four MEs, two MMs and one MI. Moreover, if we assume that the sign of the multipliers is uniformly distributed, two 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 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 are summarized in Table 1.

7. COMPARISON OF DIFFERENT IMPLEMENTATIONS

Let us consider a scenario in which a set of encrypted data must undergo different processing tasks. In such a scenario, 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 task must rely on an implementation satisfying the requirement on the modulus.

Table 1: Computational complexity of s.p.e.d. FFT algorithms.

	radix-2	radix-4	DFT
ME	$3M\log_2 M - 6M$	$\frac{7}{4}M\log_2 M - \frac{7}{2}M$	$4M^2$
MM	$3M\log_2 M - 2M$	$\frac{11}{4}M\log_2 M - \frac{3}{2}M$	$4M^2 - 2M$
MI	$3M\log_2 M - 4M$	$\frac{9}{4}M\log_2 M - \frac{5}{2}M$	$2M^2$

In the following, we will assume that both Q_1 and Q_2 are powers of two, i.e., $Q_1 = 2^{n_1}$ and $Q_2 = 2^{n_2}$. For security reasons, we will assume that the minimum modulus used by Paillier satisfies $n_P = \lceil \log_2 N \rceil = 1024$.

In order to ensure that no wrap-around occurs in the internal computations the modulus of the cryptosystem must satisfy

$$N \geq 2\left(2^{\nu} Q_1 Q_2^{\alpha} + \xi\right) + 1 \tag{21}$$

where we can have $\alpha = 1$ (DFT), $\alpha = \nu - 2$ (radix-2) or $\alpha = \nu/2 - 1$ (radix-4), and ξ can be deduced from equation (13), (18) or (20). 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

$$n_P \geq \nu + n_1 + \alpha n_2 + 3. \tag{22}$$

As to n_1, n_2, different scenarios can be considered, according to whether the inputs and the twiddle factors of the plaintext FFT are either fixed point or floating point values:

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 $n_1 = b_1 - 1$;

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 $n_1 = 2^{c_1} - 2$;

Fixed point coefficients: as in the case of fixed point inputs, if the W^{nk}s are quantized using b_2 bits we have $n_2 = b_2 - 1$;

Floating point coefficients: considering that the dynamic range of DFT coefficients is $[\sin(2\pi/M), 1] \approx [\pi 2^{-\nu+1}, 1]$, keeping the same precision of a floating point representation requires $n_2 = f_2 + \nu - 2$, where f_2 is the number of bits of the fractional part.

As an alternative way to choose n_2, one could consider the variance of the error introduced by coefficient scaling and rounding. In the case of the radix-2 implementation, an error analysis has been provided in [8], showing that the approximation error introduced by a s.p.e.d. implementation is well below that of a plaintext fixed point implementation and comparable to that of a plaintext floating point implemementation. Such analysis can be easily extended to the radix-4 case, leading to similar results.

Given the number of bits of Paillier and the number of points of the FFT, the disequalities implied by (22) can be compared in order to assess which implementation is feasible. An example is given in Fig. 1, where both fixed point and floating point cases are considered. In the floating point case, we considered IEEE 754 single precision inputs ($c_1 = 8$) and double precision coefficients ($f_2 = 52$). As can be seen, the radix-4 implementation can be used with $n_P = 1024$ even if the number of DFT points grows very large. However, in the floating point scenario even the radix-4 may not be feasible for high DFT sizes. Hence, if some processing requires a higher size DFT one has to resort to an alternative implementation, e.g. the direct DFT.

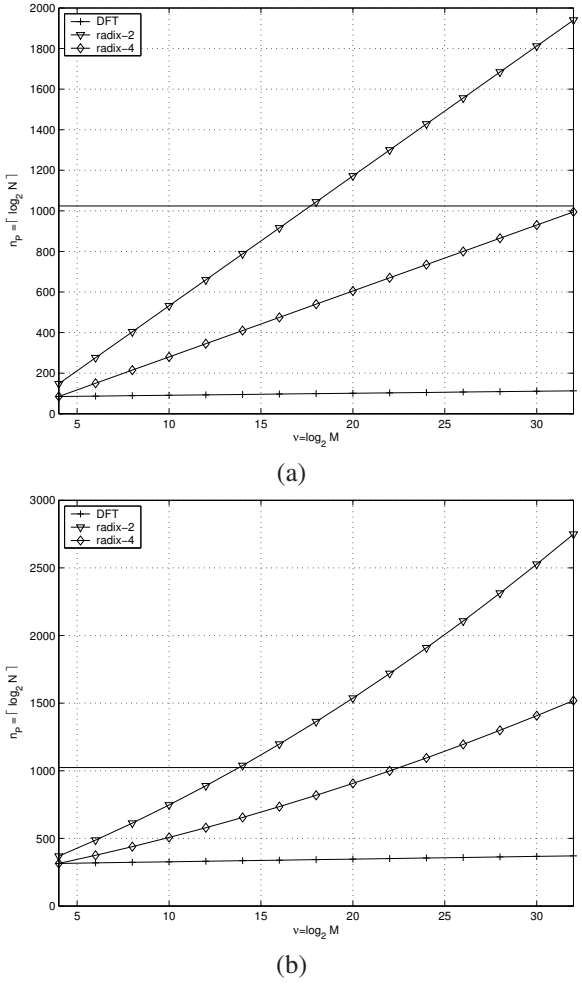
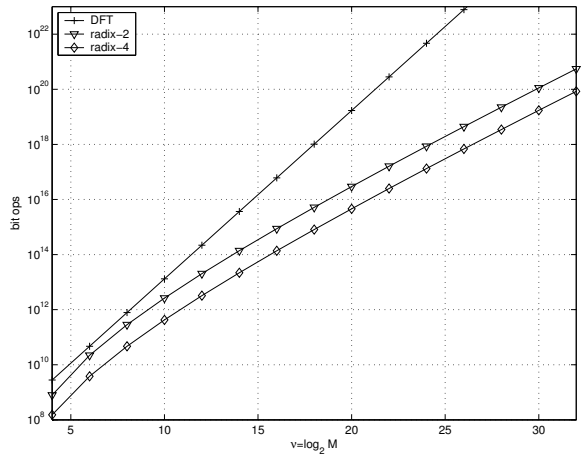


Figure 1: Minimum value of $n_P = \lceil \log_2 N \rceil$ as a function of $\nu = \log_2 M$ for three different s.p.e.d. DFT implementations. a) $n_1 = 15$, $n_2 = 63$; b) $n_1 = 254$, $n_2 = 50 + \nu$. The straight line corresponds to $n_P = 1024$.

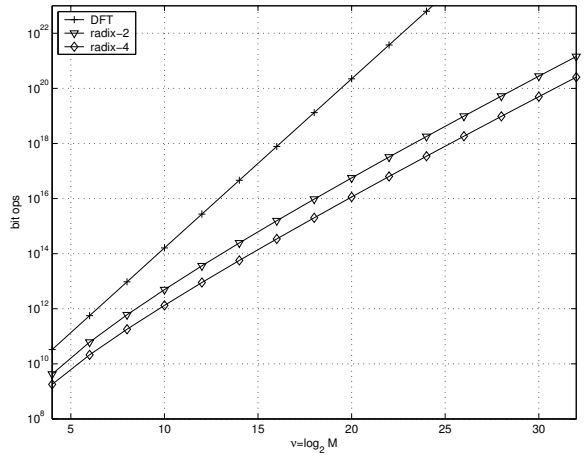
Also a different scenario can be taken into consideration, in which the modulus of the cryptosystem is set to the minimum value required by a particular FFT implementation. Since the cost of a modular operation depends on the modulus size [9], a natural question is whether a fast algorithm requiring a higher modulus size (e.g., the FFT) can be less efficient than a naïve implementation requiring a lower modulus size (e.g., the direct DFT).

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

Given the above hypotheses, the complexity of the differ-



(a)



(b)

Figure 2: Number of bit operations for three different s.p.e.d. DFT implementations, according to (23)-(25). a) $n_1 = 15$, $n_2 = 63$; b) $n_1 = 254$, $n_2 = 50 + \nu$.

ent implementations can be expressed as

$$C_{DFT} = 24 \cdot 2^{2\nu} n_2 (\nu + n_1 + n_2 + 3)^2 \quad \text{bit ops} \tag{23}$$

$$C_{R2} = 18(\nu-2)2^{\nu} n_2 (\nu + n_1 + n_2(\nu-2) + 3)^2 \quad \text{bit ops} \tag{24}$$

$$C_{R4} = 10.5(\nu-2)2^{\nu} n_2 (\nu + n_1 + n_2(\nu-2)/2 + 3)^2 \quad \text{bit ops} \tag{25}$$

The complexity of the proposed FFTs is compared in Fig. 2. As can be seen, even if both FFTs are impaired by 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.

It is also interesting to compare the complexity of the s.p.e.d. DFT with that of encryption/decryption operations. In the case of the Paillier cryptosystem, both operations require one exponentiation each [5]. However, such exponentiations are more complex than those involved in s.p.e.d. operations, because the exponents usually have n_P significant bits (whereas s.p.e.d. requires n_2 bits exponentiations). If we model the cost of an encryption/decryption operation as $C_{\mathscr{E}\mathscr{D}} = 6n_P^3\kappa$, then the cost of encrypting and decrypting a vector of $M = 2^{\nu}$ samples will be $C_{ENC/DEC} = 12 \cdot 2^{\nu}(n_P)^3$ bit ops. Such cost can be compared to (23)-(25). For example, in the case of the radix-4 FFT, as long as $10.5(\nu-2)n_2 < 12n_P$ the cost of the FFT will be less than the cost of encryption plus decryption. One of the consequences of this behavior is that a processing chain performing decryption, plaintext FFT, and encryption, apart from not guaranteeing confidentiality, can be more expensive than directly processing encrypted data.

8. CONCLUDING REMARKS

We have investigated different FFT implementations on a vector of encrypted samples, which are made possible thanks to the homomorphic properties of the underlying cryptosystem. The relations between the maximum allowable FFT size and the modulus of the cryptosystem, the FFT implementation, and the required precision have been derived. 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. Our approach demonstrates that the radix-4 FFT is best suited for both scenarios, giving useful design criteria for the implementation of s.p.e.d. modules.

REFERENCES

[1] A. Adelsbach and A.-R. Sadeghi. Zero-knowledge watermark detection and proof of ownership. In *Proc. 4th Int. Work. on Information Hiding, IH'01*, pages 273–288, 2001.

[2] M. Malkin and T. Kalker. A cryptographic method for secure watermark detection. In *Proc. 8th Int. Work. on Information Hiding, IH'06*, Old Town Alexandria, Virginia, USA, 10-12 July 2006. Springer Verlag.

[3] N. Memon and P. Wong. A buyer-seller watermarking protocol. *IEEE Trans. on Image Proc.*, 10(4):643–649, Apr. 2001.

[4] M. Kuribayashi and H. Tanaka. Fingerprinting protocol for images based on additive homomorphic property. *IEEE Transactions on Image Processing*, 14(12):2129–2139, Dec. 2005.

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

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

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

[8] T. Bianchi, A. Piva, and M. Barni. Implementing the discrete Fourier transform in the encrypted domain. In *Proc. of ICASSP 2008*, 30 Mar.–4 Apr. 2008.

[9] Ç. K. Koç, T. Acar, and B. S. Kalinski. Analyzing and comparing Montgomery multiplication algorithms. *IEEE Micro*, 16(3):26–33, June 1996.