Priv-Ware
Privacy aware processing of encrypted signals for treating sensitive information

Signal processing primitives

Tiziano Bianchi, Alessandro Piva
University of Firenze

Sponsored by
MIUR
Contract no. 2007JXH7ET

Deliverable 3.1
1 September 2010
Abstract

This deliverable summarizes the activities regarding the study of signal processing primitives in the encrypted domain. The first section deals with the problem of signal representation in the encrypted domain. The next three sections analyze three important building blocks to be used in the design of more complex protocols, namely the encrypted domain Discrete Fourier Transform, the composite representation, and the additive/multiplicative blinding.

1 Introduction

This deliverable summarizes the main achievements in the field of signal processing in the encrypted domain (s.p.e.d.) concerning the identification of suitable signal processing primitives and the analysis of their properties.

Section 2 analyzes the problem of signal representation. As a matter of fact, signals are usually represented by real numbers and stored and processed as floating point numbers. However, many useful cryptographic tools, like homomorphic cryptosystems, work on messages belonging to finite integer rings, hence raising the problem of defining a suitable way of representing signals by avoiding to resort to floating point arithmetic. Several solutions are investigated, and the best candidate for the implementation of the subsequent low level signal processing building blocks is selected.

The next three sections present the most significant low level processing primitives built on top of the chosen representation. In section 3, we present a solution for the implementation of the Discrete Fourier Transform in the encrypted domain, together with a general theoretical framework permitting to analyze the trade-off between representation accuracy and computational complexity.

Section 4 is devoted to another fundamental aspect of signal representation, i.e. the necessity to limit the expansion factor accompanying the componentwise encryption of signal samples. Specifically a composite representation of signals is presented that on one side allows to greatly reduce the expansion factor and on the other side it allows the parallel processing of several signal samples thus ensuring significant storage and computation savings.

Finally, we consider the possibility of carrying out a limited yet important set of non-linear operations by means of signal obfuscation (blinding). In brief, a party that wants to apply a non-linear function to an encrypted value first blinds it either by multiplying or adding to it a random factor, then it passes the blinded value to the party knowing the encryption key, who decrypts the blinded value, applies the non linear function, re-encrypts the result and sends it back to the first party, who in turn should be able to recover the true encrypted value by unblinding the result (still working in the encrypted domain). We have first derived the conditions that ensure that the above procedure is possible, then we proposed an information theoretic framework to analyze the security of a blinding scheme. As opposed to other security analysis, we proposed a soft security measure that fits better the imprecise nature of signals for which some limited (and measurable) leakage of information may be allowed. Finally we have applied the newly introduced measure to assess the security of additive and multiplicative blinding, and to find the optimal probability density function of the blinding factor in the two cases. The theory of additive and multiplicative blinding is described in section 5.

2 Signal representation

The first theoretical problem that the developers of s.p.e.d. techniques have to face with is the definition of a proper signal representation model. This includes finding a way to represent signals by using the finite size ring arithmetic made available by all practical cryptosystems, and limiting the expansion factor accompanying the sample-wise encryption at the basis of any s.p.e.d. operation relying on homomorphic encryption.

In this section, we discuss some possible ways to represent signals and we introduce the trade-offs that need to be considered when choosing a proper representation model. The considered trade-offs include: storage requirements, representation accuracy (linked to the quantization error made when passing from real to integer numbers), computational complexity.
2.1 Approximating real numbers on finite rings

Signals are customarily represented by real-valued sequences. Real numbers have dynamic structures, such that the length of their representation strings generally increase during the computations. On the other hand, traditional encryption schemes work with strings of limited lengths, where these lengths are governed by security parameters. This means that if the traditional encryption schemes, which are based on ring of integers or polynomials, are used to perform exact homomorphic operations on encryption of real numbers, the security parameter should be infinitely large.

The problem of representing real numbers by strings of limited length is not specific to cryptography. Traditional computing systems use truncations of real numbers for storing and representation purposes. The results of computations on encrypted values must also be truncated, in order that further computation on these numbers be possible. In the following we describe several approaches to represent real values in computing systems and analyze their adaption to homomorphic encryption schemes.

2.1.1 Floating Point Representation

A wide range of real numbers can be represented by few bytes, when the floating point representation is used. In the basis $\beta$, the floating point representation of the number $x$ is the tuple $(s_x, e_x)$, where $s_x$ and $e_x$ are called significand and exponent, respectively, such that:

$$x \approx s_x \beta^{e_x}$$

The significand and exponent are selected in such a way that the approximation error in (1) is as small as possible. Floating point representations use bit-strings of lengths $l_s$ and $l_e$ to store the significands and exponents, respectively. We denote such a floating point representation by $\text{FLP}(l_s, l_e)$. These values are then interpreted as the signed integer representation for $e_x$, whereas that of $s_x$ is a decimal value such that, per agreement, the decimal point is at the leftmost position. For example if $\beta = 10$ the value 0.0125 is shown by $s_{0.0125} = 125$ and $e_{0.0125} = -1$ since we have:

$$0.0125 = 0.125 \times 10^{-1}$$

It can be shown that when the value $x$ is represented by $(s_x, e_x)$ in $\text{FLP}(l_s, l_e)$, then the absolute value of the error of approximation in (1) can be bounded by

$$\beta^{e_x - (l_s - 1)}$$

and hence to decrease the error, it is desirable to make $e_x$ as small as possible. This results in the normalized representation of floating point for which the leftmost digit of the significand is not zero.

Multiplication of floating point numbers is straightforward: multiply the significand, add the exponents, and normalize the result. Addition is, on the other hand, more complicated. To do addition we have to find the maximum of the two exponents, adjust the significands in such a way that both of them have this common exponent, and add the significands together. Finally normalize the result.

To perform these operations using homomorphic encryptions, we can use a multiplicative homomorphic encryption for the significands and an additive one for the exponents. In this way the multiplication will be easier, but still requires truncation of significands during the normalization and checking if the leftmost digit is zero. Addition of two values will need comparison of two encrypted values and, generally, truncating the results to bring them into the larger exponent and finally normalization.

Given the difficulties of implementing the above operations in the encrypted domain without resorting to interactivity, we can conclude that the floating point representation is not a suitable model for s.p.e.d. computation.

2.1.2 Fixed Point Representation

A less space-efficient (but simpler) approach to represent real numbers is to use fixed point representation. This method, in its original form, is characterized by the scaling factor

$$R = \beta^p$$

4
which is multiplied by the rational number $x$ to compute its fixed point representation, denoted by $X$, i.e.,

$$x \approx X/R$$  \hspace{1cm} (4)

Here $\beta$ is the basis used to represent numbers and $p$ is the number of digits on the right side of the floating point. The integer $X$ is computed from the rational number $xR$, either by taking the integer part, or by truncation according to the system used. It is easy to show that the approximation error in (4) can be bounded by $R^{-1}$. In contrast to (3) we let $R$ to be any integer and denote the fixed point representation by the scaling factor by $\text{FIP}(R)$. Again here, the values are represented by a limited number of digits. We assume that the absolute value of integer representation of numbers in $\text{FIP}(R)$ are smaller than $R^2$, whose length we denote by representation length.

Computation using fixed point representation is easier than in floating point. Addition of values is done by adding corresponding integers. Multiplication is also multiplication of integers followed by truncation. The truncation stage can be explained as below. Let $X$ and $Y$ be the representations of $x$ and $y$ in $\text{FIP}(R)$, i.e., $X \approx xR$ and $Y \approx yR$ and hence,

$$XY \approx xyR^2$$  \hspace{1cm} (5)

whereas the valid representation of $xy$ is an integer $Z$ such that $Z = xyR$. It is not difficult to see that a suitable candidate for $Z$ can be computed by dividing $XY$ by $R$ and taking the quotient. Although even this simple task does not seem to be easily possible in encrypted domain, it seems that fixed point representation is more suitable for computations in encrypted domain.

Indeed, one of the strongest arguments against fixed point representations is the long representations when the same accuracy as floating point is going to be achieved. We argue here, that this argumentation cannot be used in encrypted domain, since here, we already need long representations to achieve high security. As one of the most famous examples for additive homomorphic encryptions we mention the Paillier encryption scheme. One of the most important parameters in this system is the integer $N$, where the arithmetic is done in the ring $\mathbb{Z}_{N^2}$ and the plaintext $x$ is assumed to be smaller than $N$. $N$ is generally called the modulo number in Paillier encryption scheme. On the other hand, according to (5) we see that the length of $N$ must be larger than two times representation length, i.e., length of $R^2$. These and the fact that fixed point arithmetic consists of arithmetic on integers show that $\text{FIP}(R)$ can be used together with the Paillier encryption with the modulo number $N$ as long as

$$R^4 < N$$  \hspace{1cm} (6)

Assuming $R^4 = N$ and the fact that the maximum error of $\text{FIP}(R)$ is $R^{-1}$ shows that using fixed point representation and the Paillier scheme with the modulo number $N$ the error can be reduced to $N^{-1/4}$. Table 1 shows different standard floating point representations from [16] and the values of suitable length for $N$ which must be used to achieve maximum accuracy of these representations.

<table>
<thead>
<tr>
<th>Format</th>
<th>best accuracy</th>
<th>$\log_2 N$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Single</td>
<td>$2^{-151}$</td>
<td>604</td>
</tr>
<tr>
<td>Single extended</td>
<td>$2^{-1055}$</td>
<td>4220</td>
</tr>
<tr>
<td>Double</td>
<td>$2^{-1076}$</td>
<td>4304</td>
</tr>
<tr>
<td>Double extended</td>
<td>smaller than $-16383$</td>
<td>very large</td>
</tr>
</tbody>
</table>

Table 1: The parameters of single and double precision formats of IEEE-754 standard and appropriate length of $N$ to achieve the same highest accuracy using Paillier encryption and fixed point representations.

As can be seen from Table 1 the values of $N$ for the accuracies of the first three standard representations are not far from reality using the security assumption that the factorization of $N$ must be difficult.

### 2.1.3 Fixed point representation without truncation

As we saw in Section 2.1.2 the fixed point representation provides a reasonable method to perform arithmetic on real numbers. But after each multiplication a division with remainder should be performed. We
are not aware of any method to perform this in encrypted form. One possibility is to select the modulo number so large that the required number of multiplications can be performed without any truncation. The result is then truncated after being decrypted. This results of enormous large values of $N$ which are not practical.

To solve this problem we observe that the values suggested in Table 1 are obtained for the best accuracies of the floating point representations. The accuracy of floating point representations vary over their range, as can be observed from (2), whereas accuracy of fixed point representations is fixed over their whole range of definition. This effect is shown in Figure 1.

This means that the values of Table 1 are very conservative and the values of $N$ can be smaller for the same arithmetic accuracy. This smaller values of $N$ are appropriate for special problems and must be selected according to the exact arithmetic circuit which is used to solve the problem.

\[
\begin{array}{cccc}
0 & e_x^\text{Maximum} & e_x^\text{Effective} & e_x^\text{Accuracy} \\
\text{Average} & & & \\
\end{array}
\]

(a)

\[
\begin{array}{cccc}
0 & e_x^\text{Average} & & \\
\text{Accuracy} & & & \\
\end{array}
\]

(b)

Figure 1: (a) The accuracy of floating point representation varies over the range of definition, whereas (b) the accuracy of fixed point representations is constant.

### 2.1.4 Rational representation of signals

Another method to represent real numbers is to approximate them by means of rational number, that, in turn, are represented by integers. Such an approach is described in [14] where the homomorphic properties of the Paillier encryption scheme are exploited. In this method the rational number $s/t$, for $s, t < N$, is denoted by $r \cdot t^{-1}$, where $t^{-1}$ is the inverse of $t$ in $\mathbb{Z}_N$. To convert the number $u' = s't^{-1}$ to the rational representation $s'/t'$ the Gauss method is used to find a short vector in the lattice generated by the vectors $v_1 = (N, 0)$ and $v_2 = (u', 1)$. This method guarantees to give the unique solution if the values $s'$ and $t'$ are in the intervals $-S_{\text{lattice}} < s' < S_{\text{lattice}}$ and $-T_{\text{lattice}} < t' < T_{\text{lattice}}$ and $2S_{\text{lattice}}T_{\text{lattice}} < N$.

Adding and multiplying two rational numbers which are represented in this method can be done simply by adding and multiplying, respectively, the corresponding integer representations in $\mathbb{Z}_N$. In addition it can be shown that this method has a better accuracy than the fixed point representation. The only problem is that in this method even addition of two values can cause denominators to grow quickly and hence before any further computations, the values must be truncated, a highly non-linear operation that can not be performed without interaction.

### 2.1.5 Algebraic integers

An alternative representation for signals in the encrypted domain could be obtained by approximating a real or complex value over a ring of algebraic integers [15]. Algebraic integers offer some advantages with respect to either integer or rational representations: for example, they are dense over $\mathbb{C}$, meaning that the quantization does not require any scaling factor, and the dynamic range of the quantized values is dramatically reduced for a given error tolerance. In the literature, the computation of a linear transform over a ring of algebraic integers was first proposed in [13], aiming at reducing the dynamic range of the computations over a residue number system.

Let $\omega = e^{2\pi i/R}$, $R = 2^\mu$, $\mu \geq 2$ be a primitive $R$th root of unity. The subring of the field of complex
numbers \( \mathbb{C} \) generated by \( \omega \) over the integers \( \mathbb{Z} \) is defined as [15, 13]

\[
\mathbb{Z}[\omega] = \left\{ \sum_{i=0}^{R/2-1} \alpha_i \omega^i \ \middle| \ \alpha_i \in \mathbb{Z} \right\}.
\]  

(7)

The set \( \mathbb{Z}[\omega] \) is usually referred to as the ring of algebraic integers. Both addition and multiplication can be defined over \( \mathbb{Z}[\omega] \) as combinations of the integer coefficients \( \alpha_i [13] \).

If \( \mu = 2 \), then \( \mathbb{Z}[\omega] \equiv \mathbb{Z}[j] \) is the ring of Gaussian integers, i.e., it contains numbers in the form \( a + jb \), \( a, b \in \mathbb{Z} \). Hence, approximating over \( \mathbb{Z}[j] \) is equivalent to rounding both the real and imaginary part to the nearest integer.

If \( \mu \geq 3 \), then \( \mathbb{Z}[\omega] \) is dense in \( \mathbb{C} \) [15, 9], meaning that any complex number can be arbitrarily approximated by a number in \( \mathbb{Z}[\omega] \) without requiring the multiplication by a scaling factor. In the following, we will refer to \( \mathbb{Z}[\omega] \) assuming that \( \mu \geq 3 \), which is sometimes referred to as the ring of cyclotomic integers [13]. Note that a real value can be approximated using only \( R/4 \) integer coefficients, since \( \Re\{\omega^{2-1}\} = -\Re\{\omega^{R-1}\} \).

In practical applications, we are interested only in those elements of \( \mathbb{Z}[\omega] \) which can be represented by bounded integer coefficients. These numbers will be indicated by the set

\[
\mathbb{Z}[\omega]_K = \left\{ \sum_{i=0}^{R/2-1} \alpha_i \omega^i \ \middle| \ \alpha_i \in \mathbb{Z}, \ |\alpha_i| \leq K \right\}.
\]

(8)

Note that each number in \( \mathbb{Z}[\omega]_K \) can be correctly represented by coefficients modulo \( N \) if \( N \geq 2K + 1 \).

Let us consider a simple s.p.e.d. application. Let \( \mathbf{x} \in \mathbb{C}^M \) be a vector of \( M \) complex data, \( \mathbf{x} = (x_1, x_2, \ldots, x_M)^T \), and \( \mathbf{T} \in \mathbb{C}^{M \times M} \) an \( M \times M \) matrix of complex coefficients. A generic linear transform can be defined as

\[
\mathbf{y} = \mathbf{T}\mathbf{x}.
\]

(9)

In order to compute the above transform over \( \mathbb{Z}[\omega] \), we need an approximation function \( \psi_Q : \mathbb{C} \rightarrow \mathbb{Z}[\omega]_Q \) such that

\[
s_k = \psi_Q(x_k), \quad ||s_k - x_k|| \leq \delta
\]

(10)

where \( \delta \) is the given tolerance. In [13], it has been shown that if \( \delta = 2^{-b} \) (that is, \( b \) bit precision is required), then

\[
Q \approx \frac{1}{2} (CR 2^{2b-2})^{1/2} \approx 2^{b/R}.
\]

(11)

where \( C_R = R \sin(2\pi/R)/(\pi(1 - \cos(2\pi/R))) \). Hence, with respect to an integer quantization requiring \( Q_1 \approx 2^b \), the dynamic range is reduced as \( Q \approx Q_1^{1/R} \).

By using the above approximation function, each element of \( \mathbf{x} \) and \( \mathbf{T} \) can be represented as a vector of \( R/2 \) bounded integer coefficients, that is, it is suitable for computations in the encrypted domain.

Let us consider the algebraic integer representation of both the inputs and the transform coefficients, defined as \( [s]_k = \psi_{Q_1}([X]_k) \) and \( [\mathbf{T}]_{km} = \psi_{Q_1}([\mathbf{T}]_{km}) \). An algebraic integer implementation can be obtained as

\[
\mathbf{u} = \Gamma s
\]

(12)

where all operations are carried out in \( \mathbb{Z}[\omega] \). It can be demonstrated that \( \mathbf{u} \in \mathbb{Z}[\omega]_K \), where \( K \leq Q_1(M R Q_\alpha /4) \) [13]. Hence, the above implementation is feasible in the encrypted domain if \( N \geq 2Q_1(M R Q_\alpha /4) + 1 \).

The above results can be compared with the requirements of an integer quantization system, showing that the algebraic integer representation allows us to perform a greater number of operations in the encrypted domain. As a drawback, such a representation requires to encrypt \( R/2 \) values for each complex sample. Since the number of bits of the encrypted representation is fixed due to security requirements, this means that the algebraic integer representation would require \( R/4 \) times the bits of an integer representation when translated into the encrypted domain.
2.1.6 Remarks

Given the above discussion, it is evident that fixed point representation without truncation and rescaling is the most suitable representation model if homomorphic encryption has to be used. For this reason in the following we will always assume that signals are represented as sequences whose values are represented by fixed point integers. Once the signal representation model has been fixed, it is necessary to choose its working parameters, namely the number of bits used to represent a signal value and the quantization step used to pass from a real to an integer representation. The issues associated to such a choice are described in the next section by considering some of the most common linear signal processing operations.

3 Encrypted domain Discrete Fourier Transform

To start our analysis of the implementation of linear processing modules in the encrypted domain we observe that since practical homomorphic cryptosystems (e.g., Paillier) are based on modular operations on a finite field/ring, both the inputs and the outputs of the computation need to be correctly represented as integers on an appropriate finite field, where, 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 \( x \) we should have \( |x| \leq (N - 1)/2 \); otherwise, its magnitude will be lost due to the modulo \( N \) operations. Hence, the size of \( N \) imposes a trade-off between the accuracy of encrypted domain computations and the number of operations that can be performed without resorting to interactive protocols.

This trade-off can be investigated according to different points of view. One of the main aspects in a s.p.e.d. implementation is its feasibility. 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 processing task must rely on an feasible implementation, i.e., an implementation which relies on homomorphic computations only.

Another important aspect of an encrypted domain implementation is its computational complexity. When several implementations are feasible, 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 [10, 19], a further question to be answered is whether a fast algorithm requiring a higher modulus size can be less efficient than a naive implementation requiring a lower modulus size.

In the sequel, we will present a theoretical framework wherein the above issues can be cast and solved. The proposed framework will be presented by considering a practical case study, the implementation of the Discrete Fourier Transform (DFT) and its fast version (FFT) in the encrypted domain [3, 6]. 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. The final aim is to provide design criteria which take into account the different issues raised by the s.p.e.d. DFT. To do so, we will first 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 theoretical framework introduced previously will be exploited since it permits to give general rules for the choice of the s.p.e.d. DFT parameters, which will apply also in more general scenarios [1, 5].

3.1 Signal Model for the Encrypted Domain

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

\[
X(k) = \sum_{n=0}^{M-1} x(n)W^{nk}, \quad k = 0, 1, \ldots, M - 1
\]
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).

First of all, let us consider a signal $x(n) \in \mathbb{R}$, $n = 0, \ldots, M - 1$. In the following, we will assume that the signal has been properly scaled so that $|x(n)| \leq 1$. In order to process $x(n)$ in the encrypted domain, its values have to be represented as integer numbers belonging to $\mathbb{Z}_N$. This is accomplished by first defining an integer version of $x(n)$ as

$$s(n) = \lfloor Q_1 x(n) \rfloor$$

where $\lfloor \cdot \rfloor$ is the rounding function and $Q_1$ is a suitable scaling factor and then encrypting the modulo $N$ representation of $s(n)$, i.e., $E[s(n)] \triangleq E[s(n) \mod N]$.

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

$$\hat{x}(n) = \begin{cases} 
\frac{D[E[s(n)]]}{Q_1} & \text{if } D[E[s(n)]] < \frac{N}{2} \\
\frac{D[E[s(n)]]-N}{Q_1} & \text{if } D[E[s(n)]] > \frac{N}{2} 
\end{cases}$$

When considering the DFT, the above signal model can be easily extended to complex signals. We will consider a signal $x(n) \in \mathbb{C}$, with $x(n) = x_R(n) + jx_I(n)$, $x_R, I \in \mathbb{R}$, where we will assume $|x(n)| \leq 1$, from which $|x(R, I)(n)| \leq 1$.

The quantization process simply becomes

$$s(n) = [Q_1 x(n)] = [Q_1 x_R(n)] + j[Q_1 x_I(n)] = s_R(n) + js_I(n)$$

Based on the properties of $x(n)$, the quantized signal will satisfy $-Q_1 \leq s_{R, I}(n) \leq Q_1$. Hence, we can consider the encryption of $s(n)$ as the separate encryption of both $s_R(n)$ and $s_I(n)$, i.e., $E[s(n)] = \{E[s_R(n)], E[s_I(n)]\}$, and the original value can be obtained by applying eq. (15) to both real and imaginary parts.

The coefficients $W_{nk}$ in (13) can be quantized using the same strategy as above. In particular, we define

$$C(u) = [Q_2 W^u] = [Q_2 \cos(2\pi u/M)] - j[Q_2 \sin(2\pi u/M)] = C_R(u) + jC_I(u)$$

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

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

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 [28], either requiring four real multiplications and two real additions, $sC = \{s_RC_R - s_IC_I, s_RC_I + s_IC_R\}$, or three real multiplications and three real additions, $sC = \{(s_R + s_I)C_R - s_I(C_R + C_I), (s_R + s_I)C_I - s_R(C_R - C_I)\}$. If the inputs are encrypted with the Paillier cryptosystem, when implemented in the encrypted domain such implementations become

$$E[s]^{C(1)} \triangleq \{E[s_R]^{C_R} E[s_I]^{C_I}, E[s_R]^{C_I} E[s_I]^{C_R}\}$$

$$E[s]^{C(2)} \triangleq \{E[Z]E[s_R]^{C_R}, E[Z]E[s_I]^{C_I}\}$$
where \( E[Z] = (E[s_R]E[s_I])^C_n \), \( C_+ = C_R + C_I \), \( C_- = C_R - C_I \) and all computations are carried out modulo \( N^2 \). For example, if we use (19) the DFT in the encrypted domain can be evaluated as

\[
E[S(k)] = \prod_{n=0}^{M-1} E[s(n)]^{C(nk)} = \left\{ \prod_{n=0}^{M-1} E[s_R(n)]^{C(nk)} E[s_I(n)]^{-C_I(nk)}, \prod_{n=0}^{M-1} E[s_R(n)]^{C_I(nk)} E[s_I(n)]^{C(nk)} \right\}
\]

(21) for \( k = 0, 1, \ldots, M - 1 \).

### 3.2 Upper Bounds and Magnitude Requirements

The computation of the DFT using (18) 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 (18) using a cryptosystem which encrypts integers modulo \( N \), one must ensure that the one-to-one mapping defined by (14)-(15) 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 \( |S_{R,L}(k)| \leq Q_S \), and verify that \( N \geq 2Q_S + 1 \).

In the following we will see, irrespective of the DFT implementation, \( S(k) \) can always be expressed as

\[
S(k) = KX(k) + \varepsilon_S(k)
\]

(22) where \( K \) is a scale factor depending on the particular implementation, whereas \( \varepsilon_S(k) \) 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 \(|S(k)| \leq K|X(k)| + |\varepsilon_S(k)| \). Hence, for all \( k \) we obtain

\[
|S(k)| \leq |MK + \varepsilon_{S,max}| \triangleq Q_S
\]

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

The value of \( K \) and \( \varepsilon_{S,max} \) 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.

#### 3.2.1 Direct Computation

Let us express the quantized samples in (16) and the quantized coefficients in (17) as \( s(n) = Q_1x(n) + \varepsilon_s(n) \) and \( C(u) = Q_2W^n + \varepsilon_W(u) \), respectively, where \( \varepsilon_s(n) \) and \( \varepsilon_W(u) \) are the quantization errors. If the DFT is computed directly by applying (18), then we have

\[
S(k) = Q_1Q_2X(k) + \sum_{n=0}^{M-1} \left[ Q_1x(n)\varepsilon_W(nk) + Q_2\varepsilon_s(n)W^{nk} + \varepsilon_s(n)\varepsilon_W(nk) \right].
\]

(24)

The scaling factor in (22) is then \( K = Q_1Q_2 \). As to the upper bound \( Q_S \) given by equation (23), due to the properties of the rounding function, \(|\varepsilon_sW(n)\leq 1/\sqrt{2}\). Hence \(|s(n)| \leq Q_1 + 1/\sqrt{2}, |C(u)| \leq Q_2 + 1/\sqrt{2}\) and, after simple manipulations,

\[
|S(k)| \leq M \left( Q_1Q_2 + \frac{Q_1}{\sqrt{2}} + \frac{Q_2}{\sqrt{2}} + \frac{1}{2} \right)
\]

(25)

from which we derive \( Q_S = MQ_1Q_2 + |Q_1/\sqrt{2} + Q_2/\sqrt{2} + 1/2| \).

#### 3.2.2 Real-Valued Signals

In the case of the DFT of a real-valued signal, the direct DFT computation can be expressed as

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

(26)
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 \(|S_{R,I}(k)| \leq Q_S M Q_1 Q_2\). 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 [27], since this leads to a sensible reduction of the complexity. Hence, in the following only the complex-valued case will be considered.

### 3.2.3 Bounds on Complex Multiplications

The bound in (25) does not take into account the intermediate computations of a complex multiplication. This is not a problem in the case of (19), since \(|s_R + s_I| \leq |s| \sqrt{2}\) 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 (25) should be multiplied by a factor \( \sqrt{2} \) whenever (20) is used. For the sake of simplicity, the upper bounds in the following sections will be derived under the hypothesis that (19) is used. The corresponding upper bounds in the case of (20) can be obtained by multiplying by \( \sqrt{2} \).

### 3.2.4 Decimation in Time Radix-2 FFT

This algorithm is applied when \( M = 2^\nu \) and allows the DFT to be computed in \( \nu \) stages each requiring \( \frac{M}{2} \) 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

\[
X^{(t+1)}(k_0) = X^{(t)}(k_0) + W^u X^{(t)}(k_1)
\]

\[
X^{(t+1)}(k_1) = X^{(t)}(k_0) - W^u X^{(t)}(k_1)
\]

where the indexes \( k_0, k_1 \) and the exponent \( u \) depend on the particular stage [21]. The computation of the above butterfly can be performed in the encrypted domain by applying the proposed model, yielding

\[
S^{(t+1)}(k_0) = Q_2 S^{(t)}(k_0) + C(u) S^{(t)}(k_1)
\]

\[
S^{(t+1)}(k_1) = Q_2 S^{(t)}(k_0) - C(u) S^{(t)}(k_1).
\]

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

\[
S^{(t+1)}(k_0) = Q_2 K^{(t)} \left(X^{(t)}(k_0) + W^u X^{(t)}(k_1)\right) + Q_2 \left(\epsilon_S^{(t)}(k_0) + W^u \epsilon_S^{(t)}(k_1)\right) + K^{(t)} X^{(t)}(k_1) \epsilon_W(u) + \epsilon_S^{(t)}(k_1) \epsilon_W(u)
\]

from which we derive the following recursive relations

\[
K^{(t+1)} = Q_2 K^{(t)}
\]

\[
\epsilon_S^{(t+1)}(k_0) = Q_2 \left(\epsilon_S^{(t)}(k_0) + W^u \epsilon_S^{(t)}(k_1)\right) + K^{(t)} X^{(t)}(k_1) \epsilon_W(u) + \epsilon_S^{(t)}(k_1) \epsilon_W(u).
\]

At the first stage \( S^{(0)}(n) = s(\tilde{n}) = Q_1 \sigma(\tilde{n}) + \epsilon_S^{(0)}(k_0) \), where \( \tilde{n} \) indicates \( n \) in bit reverse order, so that the recursion starts with \( K^{(0)} = Q_1 \) and \( \epsilon_S^{(0)}(k_0) = \epsilon_S(k_0) \). By using the initial conditions in (33), 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 (33), given as

$$|\epsilon_S^{(t+1)}| \leq \left( 2Q_2 + \frac{1}{\sqrt{2}} \right) |\epsilon_S^{(t)}| + \frac{2^t}{\sqrt{2}} K^{(t)}.$$  \hspace{1cm} (34)

By using the initial condition $|\epsilon_S^{(0)}| \leq \frac{1}{\sqrt{2}}$ in (34), the upper bound on the final quantization error can be expressed as

$$|\epsilon_S^{(\nu)}| \leq \frac{1}{\sqrt{2}} \left( 2Q_2 + \frac{1}{\sqrt{2}} \right)^\nu + \sum_{l=0}^{\nu-1} \frac{2^{\nu-1-l}}{\sqrt{2}} Q_1 Q_2^{\nu-1-l} \left( 2Q_2 + \frac{1}{\sqrt{2}} \right)^l.$$  \hspace{1cm} (35)

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 (33), 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 (34) 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 (34) (since $|\epsilon_S^{(0)}| \leq 1/\sqrt{2}$), the upper bound on the quantization error can be expressed as

$$|\epsilon_S^{(\nu)}| \leq \frac{4}{\sqrt{2}} \left( 2Q_2 + \frac{1}{\sqrt{2}} \right)^{\nu-2} + \sum_{l=0}^{\nu-3} \frac{2^{\nu-1-l}}{\sqrt{2}} Q_1 Q_2^{\nu-3-l} \left( 2Q_2 + \frac{1}{\sqrt{2}} \right)^l = \epsilon_S^{(\nu)}_{R2,\max}, \hspace{0.5cm} \nu > 2$$  \hspace{1cm} (36)

from which we derive the final upper bound on $S(k)$ as $Q_S = M Q_1 Q_2^{\nu-2} + |\epsilon_S^{(\nu)}_{R2,\max}|$.

### 3.2.5 Decimation in Time Radix-4 FFT

This algorithm can be employed when $M = 4^u$ and allows the DFT to be computed in $u$ 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* [23]

$$X^{(t+1)}(k_t) = \sum_{i=0}^{3} X^{(t)}(k_i) W^{ui} (-j)^l, \hspace{0.5cm} l = 0, \ldots, 3.$$  \hspace{1cm} (37)

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

$$S^{(t+1)}(k_0) = Q_2 S^{(t)}(k_0) + C(u) S^{(t)}(k_1) + C(2u) S^{(t)}(k_2) + C(3u) S^{(t)}(k_3).$$  \hspace{1cm} (38)

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

$$K^{(t+1)} = Q_2 K^{(t)}.$$  \hspace{1cm} (39)

$$|\epsilon_S^{(t+1)}| \leq 4Q_2 + \frac{3}{\sqrt{2}} |\epsilon_S^{(t)}| + 3 \frac{4^t}{\sqrt{2}} K^{(t)}.$$  \hspace{1cm} (40)

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 $K = K^{(\nu)} = Q_1 Q_2^{\nu-1}$, whereas the
The effects of finite precision arithmetic on DFT/FFT computation have been extensively investigated in the literature [29, 22, 17]. However, the encrypted domain implementation of the DFT introduces some noise sources in fixed point implementations. Hence, the error introduced by the proposed DFT/FFT is quantified by relying on equation (33) as

\[ \eta = \frac{\sigma^2}{Q^2z} \]

where \( \sigma^2 \) indicates the signal power.

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

\[ \tilde{\eta} = \frac{\sigma^2_{\epsilon S}}{K^2M\sigma^2_{\tilde{h}}} \]

where \( \sigma^2_{\epsilon S} \) is the variance of the output error (\( \epsilon_S(k) \)). In the case of the direct DFT computation, by relying on equation (24) and neglecting the terms \( \epsilon_s(n)\epsilon_W(nk) \), we can estimate

\[ \sigma^2_{\epsilon S} \approx MQ_1^2\sigma^2_{\epsilon} \]

Hence, it is easy to derive

\[ \tilde{\eta}_D = \eta + \frac{\sigma^2_{W,q}}{Q^2}. \]

In the case of a radix-2 FFT, the variance of the error at the \( t \)th stage can be recursively approximated by relying on equation (33) as

\[ \sigma^2_{\epsilon S(t+1)} \approx 2Q_2^2\sigma^2_{\epsilon S(t)} + 2^t\sigma^2_{W,q}(K^t)^2. \]
If we set the initial conditions $\sigma^2_{\epsilon^{(i)}} = 4\sigma^2_\epsilon$ and $K^{(2)} = Q_1$, then we have $\sigma^2_{\epsilon^{(2)}} = 2^\nu Q^2_2(\nu-2)\sigma^2_\epsilon + (\nu-2)2^{\nu-1}\sigma^2_{W,q}Q^2_2Q^2_2(\nu-3)$. Therefore, the NSR can be expressed as

$$\tilde{\eta}_{R2} = \eta + \frac{\nu - 2}{2} \frac{\sigma^2_{W,q}}{Q^2_2}. \quad (46)$$

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

$$\sigma^2_{\epsilon^{(t+1)}} \approx 4Q^2_2\sigma^2_{\epsilon^{(t)}(2)} + 3 \cdot 4^t \sigma^2_{W,q}(K^{(t)})^2 \quad (47)$$

and, since the initial conditions are $\sigma^2_{\epsilon^{(1)}} = 4\sigma^2_\epsilon$ and $K^{(1)} = Q_1$, the error at the last stage is $\sigma^2_{\epsilon_s} = 4^\nu Q^2_2(\mu-1)\sigma^2_\epsilon + 3(\mu - 1)4^{\nu-1}\sigma^2_{W,q}Q^2_2Q^2_2$. Therefore, the NSR is given by

$$\tilde{\eta}_{R4} = \eta + \frac{3(\mu - 1)\sigma^2_{W,q}}{4} Q^2_2 \quad (48)$$

### 3.3.1 Comparison with Plaintext Implementation

Since the value of $Q_1$, and hence $\eta$, 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:

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 $\tilde{\eta} - \eta \leq 4M \cdot 2^{-2b_2}/3\sigma^2_\epsilon$ [21], where $\sigma^2_\epsilon$ is the power of the input signal, assumed white. If we assume a uniformly distributed quantization error on the DFT coefficients, i.e., $\sigma^2_{W,q} = 1/6$, we have

$$Q_2 \geq \begin{cases} 
\sigma_\epsilon \cdot 2^{b_2 - \nu/2 - 3/2} & \text{DFT} \\
\sigma_\epsilon \sqrt{\nu - 2} \cdot 2^{b_2 - \nu/2 - 2} & \text{radix-2 FFT} \\
\sigma_\epsilon \sqrt{3\nu/2 - 3} \cdot 2^{b_2 - \nu/2 - 5/2} & \text{radix-4 FFT} 
\end{cases} \quad (49)$$

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 $\tilde{\eta} - \eta \leq 2\nu \cdot 2^{-3f_2/3}$ [21]. Hence

$$Q_2 \geq \begin{cases} 
\sqrt{1/4\nu} \cdot 2^{f_2} & \text{DFT} \\
\sqrt{(\nu - 2)/8\nu} \cdot 2^{f_2} & \text{radix-2 FFT} \\
\sqrt{(3\nu/2 - 3)/16\nu} \cdot 2^{f_2} & \text{radix-4 FFT} 
\end{cases} \quad (50)$$

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.
3.4 Feasibility Analysis

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 a feasible implementation, i.e., an implementation satisfying the requirements on the modulus.

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 require $n_p \geq 1024$.

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 $\pm 2^{-2^{e_1-1} - 2}$ to $\pm 2^{2^{e_1-1}}$, where $e_1$ is the number of bit of the exponent. Unless some information about the properties of the input signal are known, this requires $n_1 = 2^{e_1} - 2$;

- **Fixed point implementation**: by using $\sigma_x \leq 1$ and $\nu \geq 3$, a choice satisfying (49) 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 $n_2 = \lfloor b_2 - \nu/2 + 1/2 \rfloor$;

- **Floating point implementation**: a choice satisfying (50) for all implementations and all values of $\nu$ is $Q_2 \geq 2^{f_2}$, from which $n_2 = f_2 - 1$;

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

$$N \geq 2 \left(2^\nu Q_1 Q_2^\alpha + \xi \right) + 1 \quad (51)$$

where we can have $\alpha = 1$ (DFT), $\alpha = \nu - 2$ (radix-2) or $\alpha = \nu/2 - 1$ (radix-4), and $\xi$ can be obtained from equation (36), (41) or (25). 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

$$n_p \geq \nu + n_1 + \alpha n_2 + 3. \quad (52)$$

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 $\nu$.

3.5 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 cyphertext to a plaintext. Moreover, an encrypted subtraction requires a modular division, which is usually more complex than a modular multiplication [24, 18]. 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 \((a^{-e})^{-1} \mod n\). 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 MM, while the complexity of a complex subtraction is two MM and two MI. As to a complex multiplication, we consider the two implementations in (19)-(20), either requiring four MEs, two MM and one MI, or three ME, three MM and two MI. Moreover, if we assume that the sign of the multipliers is uniformly distributed, either two additional MIs or one and a half additional MI 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, MM, and MIs are summarized in Tables 3 and 4.

| Table 3: Computational complexity of s.p.e.d. FFT algorithms: a complex multiplication is implemented through four real multiplications. |
|-----------------|-----------------|-----------------|
| | radix-2 | radix-4 | DFT |
| ME | \(3M \log_2 M - 6M\) | \(\frac{7}{2}M \log_2 M - \frac{3}{2}M\) | \(4M^2\) |
| MM | \(3M \log_2 M - 2M\) | \(\frac{11}{2}M \log_2 M - \frac{7}{2}M\) | \(4M^2 - 2M\) |
| MI | \(3M \log_2 M - 4M\) | \(\frac{9}{2}M \log_2 M - 2M\) | \(2M^2\) |

| Table 4: Computational complexity of s.p.e.d. FFT algorithms: a complex multiplication is implemented through three real multiplications. |
|-----------------|-----------------|-----------------|
| | radix-2 | radix-4 | DFT |
| ME | \(\frac{5}{2}M \log_2 M - 5M\) | \(\frac{11}{4}M \log_2 M - \frac{11}{4}M\) | \(3M^2\) |
| MM | \(\frac{7}{4}M \log_2 M - 3M\) | \(\frac{11}{4}M \log_2 M - \frac{3}{2}M\) | \(5M^2 - 2M\) |
| MI | \(\frac{11}{4}M \log_2 M - \frac{7}{2}M\) | \(\frac{9}{4}M \log_2 M - \frac{23}{4}M\) | \(\frac{9}{2}M^2\) |

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

An interesting scenario to be taken into consideration is that 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 [10, 19], 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 can make 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 \( N_{\text{min}}^2 = 2^{n_{p,\text{min}}} \)) is modeled as \( C_E = 1.5n_2(2n_{p,\text{min}})^2\kappa \) [20], where \( \kappa \) 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

\[
C_{\text{DFT}} = \gamma_{\text{DFT}} 2^{2\nu} n_2^2 n_{p,\text{min,DFT}} \text{ bit ops}
\]
\[
C_{R2} = \gamma_{R2}(\nu - 2) 2^\nu n_2^2 n_{p,\text{min,R2}} \text{ bit ops}
\]
\[
C_{R4} = \gamma_{R4}(\nu - 2) 2^\nu n_2^2 n_{p,\text{min,R4}} \text{ bit ops}
\]

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

\[
C_{R4} < C_{R2} < C_{\text{DFT}} \quad \forall \nu \geq 3; \quad \forall n_1, n_2 \geq 0;
\]

irrespective of the implementation of the complex multiplications. A detailed proof is given in [6].

\section{Composite representation}

A problem with the use of homomorphic encryption is that signals need to be encrypted sample-wise, as clearly comes out from the previous examples. Samplewise encryption of signals poses some severe complexity problems since it introduces a huge expansion factor between the original signal sample and the encrypted one.

To fix the ideas, let us assume that the Paillier cryptosystem is used; in this case each encrypted sample is an element of \( \mathbb{Z}_{N^2} \), i.e. the set of integer numbers modulo \( N^2 \) with \( N \) being at least 1024 bit long, that is each encrypted sample needs at least 2048 bits to be represented. By considering that plain signal samples are usually represented by a few bits (e.g. 8 bits for images or 16 bits for ECG signals [11]), we conclude that due to encryption, signals are expanded by a factor ranging from 125 to 250. For instance, the size of a grey level 1000 \( \times \) 1000 image will pass from 1Mbyte in the clear to 250 Mbytes in the encrypted domain. This huge expansion factor is clearly not affordable in many practical applications.

In order to solve these problems, we developed an alternative representation of signals that permits to greatly reduce the expansion factor introduced by encryption, while still allowing the exploitation of the homomorphic properties of the underlying cryptosystem to process signals in the encrypted domain [2, 7]. In addition to limiting the storage requirement, the proposed representation allows the parallel processing of different samples, thus providing a considerable reduction of computational complexity in terms of operations between encrypted messages [4].

The main idea behind the representation we propose is to pad multiple data samples to form a composite encrypted message. To be specific, let \( \mathcal{M} \) be the message space and \( \mathcal{C} \) the cypher space and let signal samples be \( l \)-bit long. We propose to bundle \( R \) \( l \)-bit messages \( m_1, \ldots, m_R \) within a single composite message \( x \) as follows:

\[
x = m_1 \cdot 2^0 + m_2 \cdot 2^l + \ldots + m_R \cdot 2^{l(R-1)}.
\]

If \( L \) is larger than \( l \), samples will remain distinct in the composite representation; moreover, if \( L \) is sufficiently large, adding two composite messages will result in the addition of the single messages composing
them, and multiplying the composite message by a constant factor, will be equivalent to multiplying each single message by the same factor.

Though apparently simple, the composite representation of signals suggested in equation (57), presents a number of problems that need to be tackled with. First of all, the security aspects must be considered, given that the particular structure of the plain messages could leak some information about the secret keys or the plain message itself. Then the possibility of processing encrypted composite signals by relying on homomorphic encryption must be investigated; this is not a trivial task if we want to allow addition and multiplication between numbers with sign. Finally, the intrinsic parallelism provided by the composite signal representation poses some challenges and offers some opportunities that need to be carefully investigated.

In the sequel, all the above issues will be discussed, thus showing the great potentialities offered by the composite representation of signals. As a matter of fact, many of the most common signal processing operations, including block-wise linear transforms, FIR convolution, linear filtering, can be easily applied to composite signals, thus opening the way towards the development of fast and storage-efficient tools to process signals in the encrypted domain.

### 4.1 Mathematical model

We now introduce the composite representation of signals. Let us consider an integer valued signal $a(n) \in \mathbb{Z}$, satisfying $|a(n)| \leq Q$, where $Q$ is a positive integer. Given a pair of positive integers $B, R$, we define the composite representation of $a(n)$ of order $R$ and base $B$ as

$$a_C(k) = \sum_{i=0}^{R-1} a_i(k)B^i, \quad k = 0, 1, \ldots, M - 1$$

(58)

where $a_i(k), i = 0, 1, \ldots, R - 1$ indicate $R$ disjoint subsequences of the signal $a(n)$.

The $k$-th element of the composite signal $a_C(k)$, represents a word where we can pack $R$ samples of the original signal, chosen by partitioning the original signal samples $a(n)$ into $M$ sets of $R$ samples each. In the following, we will consider two ways of partitioning $a(n)$: i) $a_i(k) = a(iM + k)$; ii) $a_i(k) = a(kR + i)$. 

In the first case, each composite word will contain $R$ samples which are spaced $M$ samples apart in the original sequence, i.e., belonging to one of the $M$th order polyphase components of signal $a(n)$; this representation will be referred to as $M$-polyphase composite representation ($M$-PCR). In the second case, each composite word will contain $R$ consecutive samples of the original signal, which is equivalent to a partitioning of $a(n)$ into adjacent blocks having size $R$: this representation will be referred to as block composite representation (BCR). A graphical interpretation of $M$-PCR and BCR is provided in Fig. 2.

While the composite representation may seem a trivial one, its use for for the parallel processing of an encrypted signal is not straightforward, especially if we want to represent and process negative values. To do so, we must first establish some properties. These are given by the following theorem:

**Theorem 1** Let us assume that

$$|a(n)| < Q \quad \forall n$$

(59)

$$B > 2Q$$

(60)

$$B^R \leq N$$

(61)

where $N$ is a positive integer, and let $a_C(k)$ be defined as in equation (58). Then, the following holds:

$$0 \leq a_C(k) + \omega_Q < N$$

(62)

where $\omega_Q = Q \sum_{i=0}^{R-1} B^i - QB^{R-1}$. Moreover, the original samples can be obtained from the composite representation as

$$a_i(k) = \left\{ \left( \left( a_C(k) + \omega_Q \right) \div B^i \right) \mod B \right\} - Q.$$  

(63)
Figure 2: Graphical representation of a composite representation having order \( R \): (a) \( M \)-polyphase composite representation; (b) block composite representation. The values inside the small boxes indicate the indexes of the samples of \( a(n) \). Identically shaded boxes indicate values belonging to the same composite word.

**Proof**: let us express

\[
a_C(k) + \omega Q = \sum_{j=0}^{R-1} [a_j(k) + Q] B^j. \tag{64}
\]

Thanks to (59) and (60), we have \( 0 \leq a_j(k) + Q \leq 2Q \leq B - 1 \). Hence, \( a_C(k) + \omega Q \) can be considered as a positive base-\( B \) integer whose digits are given by \( a_j(k) + Q \). Moreover, since \( a_C(k) + \omega Q \) has \( R \) digits, it is bounded by

\[
a_C(k) + \omega Q \leq \sum_{j=0}^{R-1} (B - 1)B^j = B^R - 1 < N \tag{65}
\]

where the last inequality comes from (61).

As to the second part of the theorem, for each \( i \) we have

\[
a_C(k) + \omega Q = B^i \sum_{j=i}^{R-1} [a_j(k) + Q] B^{j-i} + \sum_{j=0}^{i-1} [a_j(k) + Q] B^j. \tag{66}
\]
Thanks to the properties of \( a_j(k) + Q \), we have \( \sum_{j=0}^{i-1} [a_j(k) + Q]B^j \leq B^i - 1 \). Hence

\[
[a_C(k) + \omega_Q] \div B^i = \sum_{j=i}^{R-1} [a_j(k) + Q]B^{j-i}
\]

\[
= B \sum_{j=i+1}^{R-1} [a_j(k) + Q]B^{j-i-1} + a_i(k) + Q
\]

(67)

from which (63) follows hence completing the proof.

4.1.1 Packing and unpacking operations

Let us now analyze the possibility to go from the samplewise representation to the composite one and back both when the plain signal and when the encrypted signal is available. To do so we will follow the notation we used in the introduction, that is we assume that the signal is owned by a party \( P_1 \) who owns the decryption key, and it is processed by a second, non-trusted, party \( P_2 \).

When working on plain data, the previous analysis ensures that given the original signal samples \( a(n) \), it is possible to compute the composite representation according to (58), and vice versa that the original signal values can be correctly computed from the composite representation according to (63).

When dealing with encrypted data, the first part of the previous theorem demonstrates, first of all, that the composite representation can be safely encrypted by using a homomorphic cryptosystem defined on modulo \( N \) arithmetic: in fact, as long as the hypotheses of the theorem hold, the composite data \( a_C(k) \) takes no more than \( N \) distinct values, so the values of the composite signal can be represented modulo \( N \) without loss of information. Concerning the security of the composite signal encryption, since we work with the Paillier cryptosystem, which is semantically secure, the security is automatically achieved.

Let us now consider the case where the original signal samples \( a(n) \) have been encrypted samplewise by \( P_1 \) by using an additive (semantically secure) homomorphic cryptosystem; in such a case, the encryption of the composite representation can be performed directly in the encrypted domain by the party \( P_2 \), by applying (58) and exploiting the homomorphic properties of the cryptosystem.

Going from the composite to the samplewise representation however is not possible in the encrypted domain by means of homomorphic computations only, since such a conversion requires rounding and division. Then unpacking has to be carried out by the data owner \( P_1 \), or performed by means of a properly designed interactive protocol involving \( P_1 \) and \( P_2 \).

4.2 Processing encrypted composite signals

4.2.1 Sample-wise operations between signals

The composite representation can be used to speed up sample-wise operations on encrypted signal via parallel processing. In the following, we will apply the proposed representation for signal scaling and for adding or subtracting two signals.

Signal scaling is defined as the multiplication of the samples of the signal by a constant scaling factor \( C \), i.e.

\[
u(n) = Ca(n).
\]

(68)

In the following, we will assume \( C \in \mathbb{Z} \).

If we apply the same operation to the composite signal we obtain a scaled version of \( a_C(k) \), that is

\[
u_C(k) = Ca_C(k).
\]

(69)

**Proposition 2** If \( B > 2CQ \), then the samples \( u(n) \) can be exactly computed from the modulo \( N \) representation of \( u_C(n) \).

**Proof**: let us express \( u_C(n) = \sum_{i=0}^{R-1} [Ca_i(k)]B^i \). It suffices to note that \( |Ca_i(k)| \leq CQ \) and replace \( Q \) with \( CQ \) in the proof of Theorem 1.
With sum/difference of two signals we denote either the sum or the difference of two signals operated sample by sample, i.e.

\[ u(n) = a(n) \pm b(n). \]  

(70)

In the following, we will assume \(|a(n)| \leq Q\) and \(|b(n)| \leq Q\). The composite sum/difference of two signals is defined as

\[ u_C(k) = a_C(k) \pm b_C(k) \]  

(71)

where \(b_C(k)\) is the composite representation of the signal samples \(b(n)\) obtained by using the same partitioning rule used for \(a_C(k)\).

**Proposition 3** If \(B > 4Q\), then \(u(n)\) can be exactly computed from the modulo \(N\) representation of \(u_C(n)\).

**Proof:** let us express \(u_C(n) = \sum_{i=0}^{R-1} (a_i(k) + b_i(k))B^i\). It suffices to note that \(a_i(k)\) and \(b_i(k)\) correspond to the same index \(n\) in the original sequences (due to the use of the same partitioning rule), \(|a_i(k) + b_i(k)| \leq 2Q\) and then replace \(Q\) with \(2Q\) in the proof of Theorem 1.

### 4.2.2 Block-wise linear transforms

Given a finite length signal having size \(W\), a linear transform can be defined by the following relationship

\[ y(r) = \sum_{n=0}^{W-1} T(n, r)a(n) \quad r = 0, 1, \ldots, W - 1 \]  

(72)

where \(T(n, r)\) are a set of coefficients defining the particular transform.

In the following, we will assume that the transform coefficients have been quantized according to some rule, i.e., \(T(n, r) \in \mathbb{Z}\), and that \(|T(n, r)| < QR\).

If the transform is implemented according to (72), the transformed signal can be bounded as \(|y(r)| \leq WQR\). However, several practical transforms can be factorized by relying on the properties of \(T(n, r)\)’s (see, for example, the DFT). Such factorizations usually lead to a faster implementation, permitting to compute the whole transform as a series of smaller and very simple elementary transforms linked together by suitable scaling factors. Depending on the quantization of the intermediate steps, the final bound on the transformed signal is usually larger than that obtained for the direct implementation. Hence, in the following we will assume \(|y(r)| \leq QS\), where \(QS\) is an upper bound that should be computed according to the particular implementation of the transform.

Usually, a transform is applied to the whole signal. However, when the size of a signal is not specified a-priori or it is very large, it is customary to partition the signal into adjacent blocks having some predetermined size and apply the transform to each block separately. This is the case, for example, of audio and image coding.

In our case, the signal \(a(n)\) is first partitioned into adjacent blocks then each block is transformed separately as:

\[ u_i(r) = \sum_{k=0}^{M-1} T(k, r)a(iM + k) \quad r = 0, 1, \ldots, M - 1, \]  

(73)

where \(i\) indicates the \(i\)-th block being transformed. Since the same processing is applied to all the blocks, it is suitable for a parallel implementation relying on the composite representation of \(a(n)\).

To be specific, we define the equivalent parallel blockwise transform as

\[ u_C(r) = \sum_{k=0}^{M-1} T(k, r)a_C(k) \quad r = 0, 1, \ldots, M - 1. \]  

(74)

where \(a_C(k)\) is the \(M\)-PCR of \(a(n)\). In the following, we will assume that the length of \(a(n)\) is a multiple of \(RM\), so that the length of the corresponding \(M\)-PCR is a multiple of \(M\).

**Proposition 4** If \(B > 2QS\), then \(u_i(r), i = 0, 1, \ldots, R - 1\), can be exactly computed from the modulo \(N\) representation of \(u_C(r)\).
Proof: let us consider the following equalities:

\[ u_C(r) = \sum_{k=0}^{M-1} T(k, r) \sum_{i=0}^{R-1} a(iM + k)B^i \]
\[ = \sum_{i=0}^{R-1} \left[ \sum_{k=0}^{M-1} T(k, r)a(iM + k) \right] B^i \]
\[ = \sum_{i=0}^{R-1} u_i(r)B^i. \]  

(75)

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

Blockwise transforms are very common in the case of 2D signals (e.g. the block DCT is the basis for the JPEG compression standard), hence in this section we extend the results of the previous section to the 2D case. Given \(r_1, r_2 = 0, 1, \ldots, M - 1\), a blockwise 2D transform can be defined as

\[ u_i(r_1, r_2) = \sum_{n_1=0}^{M-1} \sum_{n_2=0}^{M-1} T(n_1, r_1, n_2, r_2)a(p_iM + n_1, q_iM + n_2) \]  

(76)

where we assumed a square \(M \times M\) tiling of the original 2D signal and \(p_i, q_i\) define an indexing rule for the tiles.

The parallel block transform can be easily extended to the 2D case as

\[ u_C(r_1, r_2) = \sum_{k_1=0}^{M-1} \sum_{k_2=0}^{M-1} T(n_1, r_1, n_2, r_2)a_C(k_1, k_2) \]  

(77)

where \(a_C(k_1, k_2) = \sum_{r=0}^{R-1} a(p_iM + k_1, q_iM + k_2)B^i\). Given \(u_C(r_1, r_2)\), we can safely compute \(u_i(r_1, r_2)\) provided that \(B > 2Q_{S,2D}\). The demonstration follows the same lines as Proposition 4 and it is omitted for sake of brevity.

4.2.3 Convolution and linear filtering

The output of a linear filter having impulse response \(h(n)\) when the input is the sequence \(a(n)\) is given by the convolution of two sequences, defined as

\[ y(n) = \sum_{r=-\infty}^{\infty} h(r)a(n-r). \]  

(78)

In practical applications, the convolution algorithm is useful when the impulse response of the filter is finite. In our application, we consider a finite impulse response (FIR) filter of length \(L\) and we will assume that the input sequence \(a(n)\) has length \(P = RM\). Hence, we will consider the finite convolution

\[ u(n) = \sum_{r=0}^{L-1} h(r)a(n-r) \quad n = 0, 1, \ldots, P + L - 2 \]  

(79)

where we assume \(h(r) \in \mathbb{Z}\), and \(a(n) = 0\) for \(n \geq P\). Due to the properties of \(a(n)\), we can bound the output as \(|u(n)| \leq Q \sum_{r=0}^{L-1} |h(r)| = Q_F\). The proposed block convolution is defined as

\[ u_C(k) = \sum_{r=0}^{L-1} h(r)\hat{a}_C(k-r) \quad k = 0, 1, \ldots, M + L - 2 \]  

(80)

where we define

\[ \hat{a}_C(k) = \begin{cases} \hat{a}_C(k) & 0 \leq k < M \\ a_C(k-M) & M \leq k < M + L - 1 \\ 0 & \text{elsewhere} \end{cases} \]  

(81)
and \( a_C(k) \) is the \( M \)-PCR of \( a(n) \). A graphical representation of the disposition of the elements of \( a(n) \) within \( \tilde{a}_C(k) \) is given in Fig. 3. As it can be seen, the particular concatenation of the composite words (rows) properly extends each of the \( R \) parallel subsequences (columns) at the boundaries.

**Proposition 5** If \( B > 2Q_F \) and \( B^{R+1} \leq N \), then \( u(n) \) can be exactly computed from the modulo \( N \) representation of \( u_C(k) \), \( L - 1 \leq k < M + L - 1 \), as

\[
u(iM + k) = \left\{ \left( u_C(k) + \omega_{Q_F} \right) \div B^{i+1} \right\} \mod B - Q_F.
\]

(82)

where \( \omega_{Q_F} = Q_F \sum_{i=0}^{R} B^i \).

**Proof**: let us consider \( u_C(k) \). For \( L - 1 \leq k < M \) we have

\[
u_C(k) = \sum_{s=k-L+1}^{k} h(k-s)\tilde{a}_C(s) = \sum_{s=k-L+1}^{k} h(k-s)Ba_C(s)
\]

(83)

\[
u_C(k) = \sum_{s=k-L+1}^{k} h(k-s) \sum_{j=0}^{R-1} a(jM + s)B^{j+1}
\]
Figure 4: Graphical representation of $u_C(k)$. The values inside the boxes indicate the index of the samples within the composite representation. Blank boxes indicate valid convolution output values. Shaded boxes indicate convolution outputs at the boundaries. Crossed boxes indicate values that have to be discarded.

by letting $i = j + 1$ and $r = k - s$, we obtain:

$$u_C(k) = \sum_{s=k-L+1}^{k} h(k-s) \sum_{i=1}^{R} a(iM-M+s)B^i$$

$$= \sum_{i=1}^{R} \left[ \sum_{s=k-L+1}^{k} h(k-s)a(iM-M+s) \right] B^i$$

$$= \sum_{i=1}^{R} \left[ \sum_{r=0}^{L-1} h(r)a(iM-M+k-r) \right] B^i$$

$$= \sum_{i=1}^{R} u(iM-M+k)B^i$$

whereas for $M \leq k < M + L - 1$ we have

$$u_C(k) = \sum_{s=k-L+1}^{k} h(k-s)\tilde{a}_C(s) =$$

$$= \sum_{s=k-L+1}^{M-1} h(k-s)Ba_C(s) + \sum_{s=M}^{k} h(k-s)a_C(s-M)$$

$$= \sum_{s=k-L+1}^{M-1} h(k-s) \sum_{j=0}^{R-1} a(jM+s)B^{i+1} + \sum_{s=M}^{k} h(k-s) \sum_{i=0}^{R-1} a(iM-M+s)B^i$$

(84) (85)
Proposition 6

If \( B > 2Q_{F,2D} \) and \( B^{R+1} \leq N \), then \( u(n_1, n_2) \) can be exactly computed from the modulo \( N \) representation of \( u_C(k_1, n_2) \).
Proof: Let us rewrite \( u_C(k_1, n_2) \) by letting \( s = k_1 - r_1 \):

\[
\begin{align*}
    u_C(k_1, n_2) &= \sum_{r_2=0}^{L-1} \sum_{s=k_1-L+1}^{k_1} h(k_1-s, r_2) a_C(s, n_2 - r_2).
\end{align*}
\]

(92)

By following the same derivations as in (83)-(85), it is easy to show that for \( L - 1 \leq k_1 < M \) we have

\[
\begin{align*}
    u_C(k_1, n_2) &= \sum_{i=1}^{R} u(iM - M + k_1, n_2) B^i.
\end{align*}
\]

(93)

whereas for \( M \leq k_1 < M + L - 1 \) we have

\[
\begin{align*}
    u_C(k_1, n_2) &= \sum_{r_2=0}^{L-1} \sum_{s=M}^{k_1} h(k_1-s, r_2) a(-M + s, n_2 - r_2) + \sum_{i=1}^{R} u(iM - M + k_1, n_2) B^i.
\end{align*}
\]

(94)

Then, the demonstration is analogous to that of Proposition 5 by noticing that \( \sum_{r_2=0}^{L-1} \sum_{s=M}^{k_1} h(k_1-s, r_2) a(-M + s, n_2 - r_2) \leq Q_{F,2D}, |u(n_1, n_2)| \leq Q_{F,2D} \), and replacing \( Q_F \) with \( Q_{F,2D} \).

4.2.4 Practical Considerations

The proposed composite representation permits to reduce both the computational complexity and the bandwidth usage in s.p.e.d. applications. As to block scaling, block sum/difference, and blockwise transforms, the composite representation allows to process \( R \) samples in parallel by using a single s.p.e.d. operation. As to the block convolution, by computing a single output of (83) we obtain \( R \) output values of the original convolution. Therefore, the complexity of a block operation is reduced by a factor \( R \) with respect to that of the corresponding operation implemented sample-by-sample. Also the bandwidth usage is reduced by the same factor, since in all cases we pack \( R \) signal samples into a single cyphertext.

It is important to note that the constraints deriving from different applications usually lead to different choices of \( R \). Let us consider an estimate of the number of samples that can be safely packed into a single word. A safe implementation requires at least \( B = 2Q_Z + 1 \), where \( Q_Z \) is a bound on the output of the computation. Since we must have \( B < \sqrt{N} \), this leads to

\[
R_{\text{max}} = \left\lfloor \frac{\log_2 N}{\log_2 (2Q_Z + 1)} \right\rfloor \approx \left\lfloor \frac{\log_2 N}{\log_2 Q_Z + 1} \right\rfloor = R_{U,Z}.
\]

(95)

Moreover, the block convolution imposes a smaller \( B \) with respect to the other applications: since \( R + 1 \) samples must be packed in a single composite word, we have \( R_{\text{conv}} = R_{U,Z} - 1 \).

4.3 Unpacking the Composite Representation

In the previous sections, it has been shown that several basic signal processing operations can be performed on the encrypted composite representation. As a matter of fact, every kind of processing which shows a certain degree of parallelism, that is, where different signal samples undergo similar operations, is amenable to be processed in composite form. This is the case of very basic signal processing building blocks like linear filtering (convolution of two sequences) and block-by-block linear transforms.

However, several examples can be found where signals cannot be processed in composite form. A simple case is when each signal sample should be scaled by a different value, like in a masking or windowing operation. Another case occurs when the values of each sample should be added together, for example to estimate the average value of a signal. In both cases, it is easy to verify that there is no way to perform the above operations by manipulating the composite representation, unless the original samples are first extracted from it.

Since the extraction of the encrypted signal samples from the encrypted composite representation requires an interactive protocol, when and how using the composite representation should be decided according to the specific processing chain. We can consider two possible application scenarios:
1. Signal samples are encrypted samplewise and processed samplewise. Composite representation is used at the end of the processing chain to save bandwidth and/or limit storage requirements.

2. Composite representation is applied at the beginning of the processing chain (if applicable, before encryption). When (and if) needed, an encrypted samplewise representation can be computed using the following interactive protocol.

Up to now, there is no general theory which allows us to make the right choice. In order to decide which one of the two scenarios is the more convenient, the complexity of the specific unpacking protocol should be evaluated. To this end, several protocols may be devised. When $B$ is a power of two, the unpacking protocol can be implemented using the protocol in [25] for converting an encrypted composite value into the encryption of the bits. Alternatively, an efficient solution could be obtained relying on a garbled circuit, since the unpacking circuit is just a reordering of the wires. It is also possible to design ad-hoc solutions to deal with arbitrary $B$, which could permit the optimization of the number of samples that can be packed together and may prove more efficient in particular scenarios [8].

5 Additive and multiplicative blinding

The use of homomorphic encryption allows the computation of linear functions on encrypted signals. Whenever a non-linear function must be calculated, it is necessary to resort to some form of interaction between the client and the server. In general, it is necessary to resort to Multiparty Computation or Secure Function Evaluation techniques, which introduce a rather high overhead on the computation. A simplest way to use interaction between the client and the server to apply a non-linear function to private data is through obfuscation (sometimes referred to as blinding). In this chapter we discuss the opportunities offered by and the problems associated with obfuscation techniques, with particular attention to multiplicative blinding. Specifically, we will show the different level of security achievable by means of additive and multiplicative blinding in an information theoretic sense, concluding that the security level offered by multiplicative blinding should be considered non-sufficient according to a classical security analysis. Such a security level, though, may still be appropriate in particular application scenarios, wherein the use of multiplicative blinding may lead to a significant simplification of the involved protocols.

5.1 Problem statement

Consider a scenario in which Alice encrypts some data and sends them to Bob, who, at some point in the processing, wants to perform some non-linear operations over the encrypted data. A very simple protocol could be devised in the following two steps: 1) Bob obfuscates the intermediate result and sends it back to Alice; 2) Alice decrypts it, performs the operation, re-encrypts the result, and sends it to Bob. The above protocol requires the following requirements to be met in order to work (at least in the semi-honest scenario):

1. the obfuscation should be possible on encrypted values;
2. the obfuscation should preserve the meaning of the non-linear operation;
3. the obfuscation should be secure, i.e., Alice should be able to infer from the obfuscated value neither the true value of the input nor the true result of the non-linear operation.

As to the first requirement, a simple solution is to use linear obfuscation functions and an additive homomorphic cryptosystem.

As to the second requirement, we can devise several linear obfuscation techniques which preserve the meaning of simple non-linear operations. Let $x$ and $y$ be the value to be obfuscated and the obfuscated value, respectively. A few examples are:

1. scaling: $y = x + b$, where $b$ is a suitable random variable; $x/C = y/C - b/C$;
2. square: $y = x + b ; x^2 = y^2 - 2xb - b^2$;
3. \textit{sign}: \( y = ax \), where \( a \neq 0 \) is a suitable random variable; \( \text{sign}(x) = \text{sign}(y) \cdot \text{sign}(a) \).

The above examples suggest a more formal (and general) definition for the second requirement. Let us consider a non-linear operation \( \phi(x) \). We say that an obfuscation \( y = ax + b \) preserves the meaning of \( \phi(x) \) if there exists a corresponding (non-linear) operation \( \psi(y) \) such that

\[
\phi(x) = \alpha(a, b)\psi(y) + \beta(a, b)x + \gamma(a, b)
\]

where \( \alpha, \beta, \gamma \) can be arbitrary functions of \( a, b \). The rationale of the above definition is that Bob, who knows \( a, b \) and receives the encryption of both \( x \) and \( \psi(y) \), should be able to compute the encryption of \( \phi(x) \) relying on homomorphic properties.

As to the third requirement, a possible approach is to consider the security in an information theoretic sense. Let us consider the mutual information between the true value \( x \) and the obfuscated value \( y \), given as [12]

\[
I(X;Y) = \sum_{x,y} p_{x,y}(x,y) \log \frac{p_{x,y}(x,y)}{p_x(x)p_y(y)}.
\]

Perfect or unconditional security is obtained if there exists a choice of \( a, b \) such that \( I(X;Y) = 0 \) [26]. Perfect security ensures that Alice, by observing either \( y \) or \( \psi(y) \), does not discover anything about \( x \). Moreover, it also ensures that nothing can be inferred about \( \phi(x) \). This can be verified by noting that \( \phi(x) \rightarrow x \rightarrow y \rightarrow \psi(y) \) forms a Markov chain and by applying the data processing inequality for mutual information [12].

In the proposed setting, perfect security can not in general be achieved. However, it would be interesting to quantify the information leakage on both \( x \) and \( \phi(x) \), so as to define appropriate security measures on the obfuscation of \( x \). In the following we propose a theoretical framework to measure the secure leakage of a particular blinding strategy and apply it to evaluate the security of additive and multiplicative blinding.

5.2 A soft security measure

The proposed framework to analyze the security of a blinding scheme consists of two actors, Alice and Bob (see Figure 5). Alice holds some secret signal value \( x \in \mathbb{R} \), and obfuscates it by applying an obfuscation function which yields the obfuscated value \( y \in \mathbb{R} \). The obfuscated value is then sent to Bob (the attacker) who tries to discover the true value \( x \) by by applying an estimation function producing an estimate \( \hat{x} \).

The objective is to tell how close can Bob’s estimate be with respect to the true value \( x \). The signal \( x \) is assumed to have mean \( \mu \) and variance \( \sigma^2 \). Since we are dealing with signals, a suitable distortion measure is the \textit{squared distance}, defined as \( d(\hat{x}, x) = (\hat{x} - x)^2 \). The squared distance is an absolute distance, which
does not take into account the energy of $x$. In practical application, we are interested in the ratio between the error and the signal energy. Hence, we will consider the signal-to-noise ratio (SNR) of the estimate, defined as the ratio of the signal energy versus the expected squared distance between the estimate and the signal, i.e.,

$$
\eta \triangleq \frac{E[X^2]}{E[d(X, X)]]} = \frac{\sigma^2}{E[(X - \bar{X})^2]},
$$

(98)

where we have assumed that $X$ is a zero mean for simplicity.

**Theorem 7** Let us define the function $\eta()$ as

$$
\eta(R) \triangleq \frac{\sigma^2}{D(R)}.
$$

(99)

where $D(R)$ is the distortion-rate function [12]. The SNR of Bob’s estimate has the following upper bound

$$
\eta \leq \eta(I(X; Y))
$$

(100)

where $I(X; Y)$ is the mutual information between $x$ and $y$.

**Proof**: Consider the definition of the distortion-rate function:

$$
D(R) = \inf_{p_{X|X}(\hat{x}|x): I(\hat{x}; X) \leq R} E[d(\hat{X}, X)].
$$

(101)

If $I(\hat{X}; X) \leq R'$, it follows that $\eta \leq \eta(R')$. Then, the proof follows from the data processing inequality[12], which states $I(\hat{X}; X) \leq I(Y; X)$. □

The above theorem links the maximum SNR obtainable by Bob to the mutual information between the obfuscated signal and the true signal. If we define the security requirements in terms of the maximum achievable SNR, then the obfuscation technique should be designed so that $I(Y; X)$ fulfills such requirements.

The main problem in using the above theorem is that it requires to know $D(R)$. Unfortunately, closed form expressions for $D(R)$ are known only for very special cases, e.g., a memoryless Gaussian source $x$.

In order to have a similar bound for arbitrary memoryless sources, we can use the following:

**Corollary 8** The function $\eta()$ has the following upper bound

$$
\eta(I(X; Y)) \leq \frac{2\pi e \sigma^2}{2^{2h(X)}} \cdot 2^{2I(X; Y)} = 2\pi e \sigma^2 \cdot 2^{-2h(X|Y)}
$$

(102)

where $h(X)$ and $h(X|Y)$ are the differential entropy of $x$ and the conditional differential entropy of $x$ given $y$ (expressed in bits). The above bound holds with equality in the case of a Gaussian source.

**Proof**: Consider the Shannon lower bound on the rate-distortion function for squared error distortion

$$
R(D) \geq h(X) - \frac{1}{2} \log(2\pi e D).
$$

(103)

After few manipulations, we have

$$
D(R) \geq \frac{1}{2\pi e} e^{2h(X) - 2R}
$$

(104)

which substituted in (100) gives (102). Furthermore, the above bound holds with equality when $x$ is Gaussian. □

By using the above theorems, we are able to introduce a definition of security which is better suited to signals than the classical definitions in the literature. A signal is not a message, i.e., a unique value, but is a representation of a physical quantity that can be approximated up to some precision. Hence, we can relate the security of an obfuscated signal to the maximum precision with which it is possible to estimate the original signal. This can be thought of as a “soft” security definition, because it does not make a hard distinction between what is secure and what is not secure, but characterizes the security as a smooth quantity.
**Definition 9** An obfuscated signal is said to satisfy $K$-SNR security if

$$\eta(I(X;Y)) \leq K.$$  \hfill (105)

According to Theorem 7, from an $y$ satisfying $K$-SNR security it is not possible to obtain an estimate of $x$ having better SNR than $K$. The SNR security can be linked to unconditional security by observing that 0db-SNR security is equivalent to perfect security. This can be intuitively justified by considering that if $y$ does not carry any information regarding $x$, then the estimator must rely only on prior knowledge of $x$, i.e., the best estimator is the expected value of $x$. Hence, the expected distortion will be equal to the signal power.

The above definition considers a single observation of an obfuscated value. In a practical setting, we can suppose that an attacker will try to access multiple and possibly independent obfuscations of $x$. For example, considering the scenario in which $x$ is the result of some homomorphic computation in the encrypted domain depending on Bob’s encrypted input, a malicious Bob could force the same $x$ by repeatedly providing the same input. Hence, we also need a security definition which depends on the number of observations.

First of all, we can note that increasing the number of observations always increases the achievable SNR.

**Lemma 10** Let us consider $N+1$ independent obfuscations $y_1, y_2, \ldots, y_{N+1}$ of the same signal value $x$. Then

$$\eta(I(X;Y_1, Y_2, \ldots, Y_N)) \leq \eta(I(X;Y_1, Y_2, \ldots, Y_{N+1})).$$ \hfill (106)

**Proof:** Since $D(R)$ is a nonincreasing function, $\eta(R)$ is nondecreasing. Hence, the proof follows from $I(X;Y_1, Y_2, \ldots, Y_N) \leq I(X;Y_1, Y_2, \ldots, Y_{N+1})$. \hfill □

**Definition 11** An obfuscated signal is said to satisfy $N$-observation $K$-SNR security if

$$\eta(I(X;Y_1, Y_2, \ldots, Y_N)) \leq K.$$ \hfill (107)

We note that $N$-observation $K$-SNR security is stronger than simple $K$-SNR security, because it means that an attacker who observes up to $N$ independent obfuscations of $x$ can not obtain an estimate having better SNR than $K$. Moreover, perfect security implies $\infty$-observation $K$-SNR security.

In order to have $N$ independent obfuscations of $x$, Bob should iterate a protocol between him and Alice $N$ times. In practice, we can assume that there is a minimum time interval $\Delta t$ between two protocol executions (for example, because it is imposed by Alice). Hence, if $y$ satisfies $N$-observation $K$-SNR security, obtaining an estimate having better SNR than $K$ will require a time greater than $N\Delta t$. In such a sense, we can link $N$-observation $K$-SNR security to the “computational” effort which is required to obtain a $K$-SNR estimate of $x$.

### 5.3 Additive Blinding

The simplest blinding technique we can think of is the addition of a random variable to the signal values. Such a technique can be used in an interactive protocol for secure scaling of an encrypted value. In the following, we will investigate the security of additive obfuscation under a single observation and multiple observations.

The security depends on both the distribution of the signal values and the distribution of the blinding factor. In the following, we will consider the class of signals with bounded variance, defined by the set $\mathcal{B}_V(V_{max}) = \{X | E[(X - E[X])^2] \leq V_{max} \}$.

#### 5.3.1 Single Observation

The obfuscation model under a single observation is

$$y = b + x$$ \hfill (108)
where $b$ is a real blinding value. We will assume $x, b$ statistically independent.

In this case, the mutual information between $X$ and $Y$ can be upper bounded as

$$I(X;Y) = h(Y) - h(Y|X)$$
$$= h(X + B) - h(X + B|X)$$
$$= h(X + B) - h(B)$$
$$\leq \frac{1}{2} \log 2\pi e (\sigma_X^2 + \sigma_B^2) - h(B)$$

(109)

where the bound holds with equality if $Y$ is Gaussian. In order to minimize the upper bound, the blinding factor should be chosen so as to maximize $h(B)$. A natural choice is $B \sim N(0, \sigma_B^2)$, which leads to

$$I(X;Y) \leq \frac{1}{2} \log \left(1 + \frac{\sigma_X^2}{\sigma_B^2}\right).$$

(110)

The right hand side of the previous equation is the capacity of an additive white Gaussian noise channel for power limited inputs. It is known [12] that the above value is optimum according to a game theoretic approach, that is, it is a saddle point in a game where a transmitter ($X$) try to transmit with a power $\sigma_X^2$ and a jammer ($B$) try to interfere with a power $\sigma_B^2$. In other words, in the absence of any information about the distribution of $X$, the optimal additive blinding strategy in a power limited setting is given by a Gaussian distribution.

### 5.3.2 Multiple Observations

The obfuscation model under multiple observations is

$$y = b + x1$$

(111)

where $b = [b_1, \ldots, b_N]^T$ is a vector of blinding values and $1 = [1, \ldots, 1]^T$. We will assume i.i.d. blinding values independent of $x$.

In this case, the mutual information between $X$ and $Y$ can be upper bounded as

$$I(X;Y) = h(Y) - h(Y|X)$$
$$= h(Y) - h(X1 + Y|X)$$
$$= h(Y) - h(B)$$
$$\leq \frac{1}{2} \log 2\pi e \det(R_Y) - h(B)$$

(112)

The optimal blinding values are given by a sequence of i.i.d. Gaussian variables, $B \sim N(0, \sigma_B^2 I)$, and the corresponding upper bound is

$$I(X;Y) \leq \frac{1}{2} \log \left(1 + N\sigma_X^2 \sigma_B^2\right).$$

(113)

The above quantity is the capacity of the AWGN single-input-multiple-output channel.

### 5.4 Multiplicative blinding

Another useful blinding technique consists of multiplicating the signal values by a random factor. Such a technique can be used in an interactive protocol for determining the sign of a signal or for signal scaling. It can also be used to compute the logarithm of a value $x$, it only needs that Alice who holds the encrypted version of $x$ multiplies it by a known blinding factor $a$ by exploiting the additive homomorphic properties of the underlying cryptosystem. Then it passes the encrypted value of $ax$ to Bob who in turn decrypts $ax$, computes $\log(ax)$, encrypts it again and sends it back to Alice. At this point Alice can obtain the encrypted value of $\log(x)$ by computing it as $\log(ax) - \log(a)$ in the encrypted domain.

In the following, we will investigate the security of multiplicative blinding under a single observation. The case of multiple observations can be derived in a similar manner as in the additive blinding example.
5.4.1 Positive Signals

First, we will consider the case in which both the signal and the blinding factor are positive quantities. This case is useful to assess the amount of information that the observed value leaks about the magnitude of the secret value. Moreover, the general case can be derived from the results of the positive signal case. In the scalar case the model is

\[ y = bx \]  

(114)

where \( b, x > 0 \) and \( b, x \) are mutually independent. This case can be reconducted to the additive one by applying the logarithm to the involved quantities. If we define \( z = \ln y \), then

\[ z = w + v \]  

(115)

where \( w = \ln b \) and \( v = \ln x \). Note that in order to use the results we obtained for the additive case, we must assume that the variance of \( v \) is bounded, i.e., \( x \) must belong to the set \( \mathcal{B}_{L}(V_{\max}) = \{ X | E[(\ln X - E[\ln X])^2] \leq V_{\max} \} \). By applying the results of the preceding sections, we know which is the optimal distribution of \( W \) in order to minimize the upper bound on \( I(V; Z) \). The problem is to relate \( I(V; Z) \) to \( I(X; Y) \). The answer to this question is given by the following lemma.

**Lemma 12** If \( E[V] \) and \( E[W] \) exist and are finite, then \( I(V; Z) = I(X; Y) \).

**Proof:** Let us consider the following mutual informations:

\[
I(Y; X) = h(Y) - h(Y|X) \\
= h(Y) - \int_0^{+\infty} h(Y|X = x) p_X(x) dx \\
= h(Y) - \int_0^{+\infty} h(xB)p_X(x) dx \\
= h(Y) - \int_0^{+\infty} [h(B) + \ln x] p_X(x) dx \\
= h(Y) - h(B) - \int_0^{+\infty} \ln xp_X(x) dx \\
= h(Y) - h(B) - E[\ln X] \\
= h(Y) - h(B) - E[\ln Y] \\
(116)
\]

and

\[
I(Z; V) = h(Z) - h(Z|V) \\
= h(Z) - h(W + V|V) \\
= h(Z) - h(W|V) \\
= h(Z) - h(W). \\
(117)
\]

Let us consider the relationship between \( h(Y) \) and \( h(Z) \). We have

\[
h(Y) = - \int p_Y(y) \ln p_Y(y) dy \\
= - \int \frac{1}{y} p_Z(\ln y) \ln \frac{1}{y} p_Z(\ln y) dy \\
= \int \frac{1}{y} p_Z(\ln y) \ln y dy - \int \frac{1}{y} p_Z(\ln y) \ln p_Z(\ln y) dy \\
= \int zp_Z(z) dz - \int p_Z(z) \ln p_Z(z) dz \\
= E[Z] + h(Z). \\
(118)
\]
In a similar way we prove that $h(B) = E[W] + h(W)$. Hence

$$I(Y; X) = E[Z] + h(Z) - E[W] - h(W) - E[V]$$

$$= I(Z; W) + E[Z] - E[W] - E[V]$$

$$= I(Z; W) \quad (119)$$


According to the results for the additive blinding, the best distribution for $W$ is $W \sim \mathcal{N}(0, \sigma_W^2)$, which leads to $B \sim \mathcal{LN}(0, \sigma_W^2)$. Hence, the optimal distribution of the multiplicative blinding factor for the positive case is Log-Normal and the corresponding upper bound on the mutual information is

$$I(X; Y) \leq \frac{1}{2} \log \left( 1 + \frac{\sigma_Y^2}{\sigma_W^2} \right). \quad (120)$$

### 5.4.2 Signals with sign

When a signal having arbitrary sign is considered, the equivalent logarithmic model can not be used as it is. According to the relationship between the sign and the magnitude of the signal values, different cases are possible. In order to analyze the general case, we will split $I(X; Y)$ into a term taking into account the leakage regarding the sign of $X$ and a term taking into account the leakage regarding the magnitude of $X$. This is possible thanks to the following theorem.

**Theorem 13** Let the auxiliary random variable $S_X, M_X$ be defined as the sign and the magnitude of $X$. Then

$$I(X; Y) = I(S_X; Y) + I(M_X; Y|S_X). \quad (121)$$

**Proof**: It is suffices to consider the following information inequalities:

$$I(X; Y) \leq I(S_X, M_X; Y)$$

$$= I(S_X; Y) + I(M_X; Y|S_X) \quad (122)$$

where the inequality follows from the fact that $X$ is a function of $S_X$ and $M_X$,

$$I(X; Y) = I(S_X; Y) + I(X; Y|S_X) - I(S_X; Y|X)$$

$$= I(S_X; Y) + I(X; Y|S_X)$$

$$\geq I(S_X; Y) + I(M_X; Y|S_X) \quad (123)$$

since $Y \rightarrow X \rightarrow S_X$ is a Markov chain and $M_X$ is a function of $X$. □

The term $I(S_X; Y)$ quantifies the amount of information about $S_X$ that can be deduced from the observation of $Y$. The term $I(M_X; Y|S_X)$ quantifies the amount of information about $M_X$ that can be deduced from the observation of $Y$ given $S_X$. In order to derive the distribution of $B$ which minimizes both terms, we can exploit the following theorems.

**Theorem 14** Let the auxiliary random variable $S_B, M_B$ be defined as the sign and the magnitude of $B$. Then $I(S_X; Y)$ is minimized when $S_B$ and $M_B$ are mutually independent and $H(S_B) = 1$ (that is, the signs of $B$ are equiprobable). Moreover, in this case we have

$$I(S_X; Y) = I(S_X; M_Y) \quad (124)$$

where $M_Y$ is the magnitude of $Y$.

**Proof**: Thanks to the data processing inequality, $I(S_X; Y) \geq I(S_X; M_Y)$. First, we demonstrate that the equality holds when the hypotheses of the theorem are satisfied. This follows from the following
inequalities:

\[
I(S_X; Y) \leq I(S_X; S_Y, M_Y) \\
= I(S_X; M_Y) + I(S_X; S_Y | M_Y) \\
= I(S_X; M_Y) + H(S_Y | M_Y) - H(S_Y | S_X, M_Y) \\
= I(S_X; M_Y) + H(S_Y | M_Y) - H(S_X \oplus S_B | S_X, M_Y) \\
\leq I(S_X; M_Y) + 1 - H(S_B | M_Y) \\
= I(S_X; M_Y) + 1 - H(S_B) \\
= I(S_X; M_Y)
\]

where we exploited the fact that \( S_B \) and \( M_Y \) are independent. Then, we prove that \( I(S_X; Y) \) achieves its minimum when \( I(S_X; Y) = I(S_X; M_Y) \). Let us suppose that the minimum is achieved for a distribution \( B \) which does not satisfy the hypotheses of the theorem. Given \( B \), we can always construct a distribution \( B' \) such that \( M_{B'} = M_B \) and the sign of \( B' \) is equiprobable and independent of \( M_B \), by taking the magnitude of \( B \) and multiplying it by a variable taking values \( \{-1, 1\} \) with equal probability. Clearly, \( M_{Y'} = M_X M_{B'} = M_X M_B = M_Y \). Hence, thanks to the previous result and the data processing inequality \( I(S_X; Y) \geq I(S_X; M_Y) = I(S_X; M_{Y'}) = I(S_X; Y') \), which contradicts the hypothesis that \( B \) achieves the minimum. \( \square \)

**Theorem 15** Let the auxiliary random variables \( S_B, M_B \) be defined as the sign and the magnitude of \( B \). Then \( I(M_X; Y | S_X) \) is minimized when \( S_B \) and \( M_B \) are mutually independent. Moreover, in this case we have

\[
I(M_X; Y | S_X) = I(M_X; M_Y | S_X)
\]

where \( M_Y \) is the magnitude of \( Y \).

*Proof:* Thanks to the data processing inequality, \( I(M_X; Y | S_X) \geq I(M_X; M_Y | S_X) \). First, we demonstrate that the equality holds when the hypotheses of the theorem are satisfied. This follows from the following inequalities:

\[
I(M_X; Y | S_X) \leq I(M_X; S_Y, M_Y | S_X) \\
= I(M_X; M_Y | S_X) + I(M_X; S_Y | S_X, M_Y) \\
= I(M_X; M_Y | S_X) + H(S_Y | S_X, M_Y) - H(S_Y | S_X, M_Y, M_X) \\
= I(M_X; M_Y | S_X) + H(S_X \oplus S_B | S_X, M_Y) - H(S_X \oplus S_B | S_X, M_Y, M_X) \\
= I(M_X; M_Y | S_X) + H(S_B | M_Y) - H(S_B | M_X) \\
= I(M_X; M_Y | S_X) + H(S_B) - H(S_B) \\
= I(M_X; M_Y | S_X)
\]

where we exploited the fact that \( S_B \) is independent from \( M_Y \) and \( M_X \). Then, we prove that \( I(M_X; Y | S_X) \) achieves its minimum when \( I(M_X; Y | S_X) = I(M_X; M_Y | S_X) \). Let us suppose that the minimum is achieved for a distribution \( B \) which does not satisfy the hypotheses of the theorem. Given \( B \), we can always construct a distribution \( B' \) such that \( M_{B'} = M_B \) and the sign of \( B' \) is independent of \( M_{B'} \), by taking the magnitude of \( B \) and multiplying it by a random variable taking values \( \{-1, 1\} \). Clearly, \( M_{Y'} = M_X M_{B'} = M_X M_B = M_Y \). Hence, thanks to the previous result and the data processing inequality \( I(M_X; Y | S_X) \geq I(M_X; M_Y | S_X) = I(M_X; M_{Y'} | S_X) = I(M_X; Y' | S_X) \), which contradicts the hypothesis that \( B \) achieves the minimum. \( \square \)

The above theorems show that in order to minimize \( I(X; Y) \) the probability density function of \( B \) should be chosen so that \( S_B \) and \( M_B \) are independent and the sign of \( B \) is equiprobable. It is easy to show that this is equivalent to \( p_B(b) = p_B(-b) \), i.e., the pdf of \( B \) must be an even function.

As to the optimal shape of \( p_B(b) \), we can consider the following theorem.

**Theorem 16** If \( P(X = 0) = 0 \), \( S_B \) and \( M_B \) are independent, and the sign of \( B \) is equiprobable the mutual information between \( X \) and \( Y \) is

\[
I(X; Y) = I(V; Z)
\]
where we define $V = \ln M_X$ and $Z = \ln M_Y$. Moreover, the optimal distribution for the blinding factor is $M_B \sim \mathcal{LN}(0, \sigma_W^2)$. In this case, $I(X; Y)$ can be upper bounded as

$$I(X; Y) \leq \frac{1}{2} \log \left(1 + \frac{\sigma_Y^2}{\sigma_W^2}\right).$$  \hspace{1cm} (129)

**Proof:** Let us define $W = \ln M_B$. Thanks to the preceding theorems, we have the following chain of equalities

$$I(X; Y) = I(S_X; M_Y) + I(M_X; M_Y | S_X)
= I(S_X; M_X; M_Y)
\overset{(a)}{=} I(S_X, \ln M_X; \ln M_Y)
= I(S_X; V; Z)
= h(Z) - h(Z | S_X, V)
= h(Z) - h(V + W | S_X, V)
= h(Z) - h(W | S_X)
= h(Z) - h(W)
= h(Z) - h(V + W | V)$$

where (a) follows from the fact that $v = \ln x$ and $z = \ln y$ are invertible with probability equal to one and we exploited the fact that $W$ is independent from both $S_X$ and $V$. Hence, the first part of the theorem is proved. The proof of the second part follows from the optimal solution for additive blinding applied to $I(V; Z)$. □

When the pdf of $M_B$ is an even function, the above theorem says that the mutual information between $X$ and $Y$ is equal to the mutual information between their magnitudes, i.e., the general case can be reconverted to the positive case. One may wonder if this means that the sign of $X$ is perfectly hidden by the chosen blinding strategy. The answer in general is negative, i.e., in some cases there can be a sign leakage. However, an increase in the amount of information regarding the sign corresponds to an equal decrease in the amount of information regarding the magnitude\(^1\). This is shown by the following lemmas.

**Lemma 17** Let us define the auxiliary random variables $W = \ln M_B$, $V^+ = \ln X | X > 0$, and $V^- = \ln(-X) | X < 0$. If $P(X = 0) = 0$, then $I(M_X; M_Y | S_X)$ can be expressed as

$$I(M_X; M_Y | S_X) = p_{S_X} I(V^+; V^+ + W) + (1 - p_{S_X}) I(V^-; V^- + W)$$ \hspace{1cm} (131)

where $p_{S_X} = P(X > 0)$. Moreover, if $M_B \sim \mathcal{LN}(0, \sigma_W^2)$, then $I(M_X; M_Y | S_X)$ can be upper bounded as

$$I(M_X; M_Y | S_X) \leq \frac{p_{S_X}}{2} \log \left(1 + \frac{\sigma_Y^2}{\sigma_W^2}\right) + \frac{1 - p_{S_X}}{2} \log \left(1 + \frac{\sigma_Y^2}{\sigma_W^2}\right)$$ \hspace{1cm} (132)

The equality holds when $V^+$ and $V^-$ are Gaussian variables.

**Proof:** The first part is proved by the following chain of inequalities

$$I(M_X; M_Y | S_X) \overset{(a)}{=} I(\ln M_X; \ln M_Y | S_X)
= I(\ln M_X; \ln M_X + \ln M_B | S_X)
= p_{S_X} I(\ln M_X; \ln M_X + W | X > 0)
+ (1 - p_{S_X}) I(\ln M_X; \ln M_X + W | X < 0)
= p_{S_X} I(V^+; V^+ + W) + (1 - p_{S_X}) I(V^-; V^- + W)$$

where (a) follows from the fact that $v = \ln x$ and $z = \ln y$ are invertible with probability equal to one. The second part of the proof follows from the optimal solution for additive blinding applied to $I(V^+; V^+ + W)$ and $I(V^-; V^- + W)$. □

---

\(^1\) A possible explanation is that, when we have sign leakage, the sign and the magnitude of $X$ are correlated. Hence, the information regarding the sign corresponds to information common to both $S_X$ and $M_X$.  

35
Lemma 18 Let us define the auxiliary random variables $V = \ln M_X$, $W = \ln M_B$, $V^+ = \ln X|X > 0$, and $V^- = \ln(-X)|X < 0$. If $P(X = 0) = 0$ and $P(B = 0) = 0$, then

$$I(S_X; M_Y) = I(V; V + W) - p_{S_X}I(V^+; V^+ + W) + (1 - p_{S_X})I(V^-; V^- + W)$$

where $p_{S_X} = P(X > 0)$. Moreover, if $p_{V^+} = p_{V^-}$, then $I(S_X; M_Y) = 0$. When $M_B \sim \mathcal{L}N(0, \sigma^2_X)$, $I(S_X; M_Y)$ can be upper bounded as

$$I(S_X; M_Y) \leq \frac{1}{2} \log \left(\frac{\sigma^2_V + \sigma^2_W}{(\sigma^2_{V^+} + \sigma^2_{V^-})^{p_{S_X}}(\sigma^2_{V^+} + \sigma^2_{V^-})^{1-p_{S_X}}}\right)$$

where we define $\sigma^2_{V^+} = 2^{2h(V^+)/2\pi e}$ and $\sigma^2_{V^-} = 2^{2h(V^-)/2\pi e}$. The equality holds when $V$, $W$, $V^+$, and $V^-$ are Gaussian variables.

Proof: The first part is trivial, since from Theorem 16 we have $I(S_X; M_Y) = I(V; Z) - I(M_X; M_Y | S_X)$. As to the second part, let us consider the following equalities

$$I(S_X; M_Y) = I(V; V + W) - p_{S_X}I(V^+; V^+ + W) + (1 - p_{S_X})I(V^-; V^- + W)$$

$$= h(V + W) - h(W) - p_{S_X}(h(V^+ + W) - h(W))$$

$$- (1 - p_{S_X})(h(V^- + W) - h(W))$$

$$= h(V + W) - p_{S_X}h(V^+ + W) - (1 - p_{S_X})h(V^- + W).$$

We have $p_{V^+} = p_{S_X}p_{V^+} + (1 - p_{S_X})p_{V^-}$. Hence, it follows $p_{V^+} = p_{V^-} = p_{V^-}$. We substitute in (136) gives $I(S_X; M_Y) = 0$. Finally, the upper bound can be proved as follows

$$I(S_X; M_Y) = h(V + W) - p_{S_X}h(V^+ + W) - (1 - p_{S_X})h(V^- + W)$$

$$\leq \frac{1}{2} \log 2\pi e \left(\sigma^2_{V^+} + \sigma^2_{V^-}\right) - p_{S_X}h(V^+ + W) - (1 - p_{S_X})h(V^- + W)$$

$$\leq \frac{1}{2} \log \left(2^{2h(V^+)/2\pi e} + 2^{2h(W)/2}\right)$$

where (a) follows from the entropy power inequality. The bound in (135) is derived by using $h(W) = \frac{1}{2} \log 2\pi e \sigma^2_W$ in (137) and doing some straightforward manipulations. \(\square\)

The second part of the lemma gives an interesting result: when the distribution of $X|X > 0$ is identical to the distribution of $-X|X < 0$ (i.e., the positive part of $p_X(x)$ is identical to a scaled version of the negative part of $X$) the sign of $X$ is perfectly blinded by the chosen $B$. This is intuitively justified by the fact that in this case the sign of $X$ is statistically independent from the magnitude of $X$.

In summary, the best distribution for the multiplicative blinding factor $B$ among all log-power limited distributions (i.e., so that $Var[\ln M_B] \leq \sigma^2_W$) is symmetric around zero and satisfy $M_B \sim \mathcal{L}N(0, \sigma^2_W)$. The worst blinding case among all log-power limited secret signals $X$ happens when $M_X \sim \mathcal{L}N(0, \sigma^2_V)$.

It is interesting to consider two limit cases:

- $p_{V^+} = p_{V^-}$: in this case, the sign of $X$ is perfectly hidden. The only information that an attacker can obtain observing $Y$ regards the magnitude of $X$.

- $\sigma^2_{V^+} = \sigma^2_{V^-} = 0$ but $\sigma^2_{V^+} \neq 0$. This means that $p_X(x)$ puts all its mass in two distinct values, one positive and one negative, having different magnitude. In this case, we have $I(M_X; M_Y | S_X) = 0$. Hence, the only information that an attacker can obtain from $Y$ regards the sign of $X$. However, this does not mean that the magnitude of $X$ is perfectly hidden. As a matter of fact, given the particular distribution of $X$, the knowledge of $S_X$ immediately discloses the value of $M_X$. This is justified by the fact that $I(M_X; Y | S_X) = 0$: that is, the mutual information between $M_X$ and $Y$ is zero only if we condition to the observation of $S_X$.
5.4.3 Multiple observations

In the multiplicative case, the blinding model under multiple observations is

\[ y = b x \]  \hspace{1cm} (138)

where \( b = [b_1, \ldots, b_N]^T \) is a vector of i.i.d. blinding values independent of \( x \).

When we can assume to deal with positive quantities, a model analogous to the additive one can be obtained by applying the logarithm to each variable, i.e.,

\[ z = w v \]  \hspace{1cm} (139)

where \( z = [\ln y_1, \ldots, \ln y_N]^T \) and \( w = [\ln b_1, \ldots, \ln b_N]^T \). Similarly to the case of \( I(X; Y) \) and \( I(V; Z) \), the relationship between \( I(X; Y) \) and \( I(V; Z) \) can be established thanks to the following lemma.

Lemma 19 If \( E[V] \) and \( E[W_i] \), \( i = 1, \ldots, N \), exist and are finite and \( W \) are i.i.d., then \( I(V; Z) = I(X; Y) \).

**Proof**: Let us consider the following mutual informations

\[ I(Y; X) = h(Y) - h(Y|X) \]
\[ = h(Y) - N h(B) - E[V] \]  \hspace{1cm} (140)

and

\[ I(Z; V) = h(Z) - h(Z|V) \]
\[ = h(Z) - N h(W) \]  \hspace{1cm} (141)

where we used \( p_Y(y|x) = p_B(y/x) = \prod_i p_B(y_i/x) \) \( (p_Z(z|v) = p_W(z - v) = \prod_i p_W(z_i - v)) \) and \( h(Y|X) = h(B) - E[V] \). Now we have

\[ h(Y) = -\int p_Y(y) \ln p_Y(y) \]
\[ = \int \prod_{i=1}^N \frac{1}{y_i} p_Z(\ln y) \ln y_i dy \]
\[ - \int \prod_{i=1}^N \frac{1}{y_i} p_Z(\ln y) \ln p_Z(\ln y) dy \]  \hspace{1cm} (142)

\[ = \int \sum_{i=1}^N z_i p_Z(z) dz - p_Z(z) \ln p_Z(z) dz \]
\[ = E[\sum_{i=1}^N Z_i] + h(Z) \]
\[ = NE[W] + NE[V] + h(Z). \]

Hence,

\[ I(Y; X) = h(Y) - N h(B) - E[V] \]
\[ = NE[W] + NE[V] + h(Z) - NE[W] - N h(W) - NE[V] \]  \hspace{1cm} (143)
\[ = I(Z; V). \]

where we used \( h(B) = E[W] + h(W) \). □
If we assume the blinding values as a sequence of i.i.d. log-normal variables, i.e., \( B_i \sim \mathcal{LN}(0, \sigma_{B_i}^2) \), then the mutual information can be upper bounded as

\[
I(X;Y) \leq \frac{1}{2} \log \left( 1 + N \frac{\sigma_Y^2}{\sigma_W^2} \right).
\] (144)

By using similar arguments as in the previous section, it is possible to demonstrate that the above upper bound holds also in the case of signed signals, by choosing as blinding values a sequence given by the product of i.i.d. log-normal variables and i.i.d. Rademacher’s variables, i.e., variables taking values \( \{1, -1\} \) with equal probability. The demonstration is omitted for the sake of brevity.

\section{Security of Blinding}

The analysis performed in the above sections permits to evaluate the security achievable by different blinding techniques. Here, an interesting point is the amount of resources required to achieve a given security level. From (102), it is evident that a generic blinding scheme achieves at least \( K \)-SNR security, where \( K = \frac{2N\sigma^2}{W} \cdot 2^{I(X;Y)} \). Hence, in order to characterize the security of a given scheme we can substitute the value of \( I(X;Y) \) obtained when using the corresponding optimal blinding strategy (i.e., i.i.d. Gaussian values for additive blinding and i.i.d. log-normal magnitudes - equiprobable signs for multiplicative blinding).

In the following, we will assume that the signal to be blinded \( x \) is normally distributed with mean \( \mu_X \) and variance \( \sigma_X^2 \). Moreover, we will assume \( \mu_X \gg \sigma_X \), so that we can approximate \( \sigma_X^2 \triangleq Var[\ln X] \approx \sigma_X^2/\mu_X^2 \). The different schemes will be compared using the same blinding variance \( \sigma_B^2 \). Such a variance can be directly related to the number of bits required to represent the blinded values. In the case of log-normal blinding, a log-normal variable having variance \( \sigma_B^2 \) is assumed distributed according to \( \mathcal{LN}(0, \sigma_W^2) \), where

\[
\sigma_W^2 = \frac{\ln(1+4\sigma_B^2)}{2}.
\]

According to the different cases we analyzed, we can define the following quantities depending on the security parameter \( \sigma_B \):

\[
K_{A}(\sigma_B) = 1 + \frac{\sigma_X^2}{\sigma_B^2}, \quad (145)
\]

\[
K_{M}(\sigma_B) = 1 + \frac{\sigma_X^2}{\mu_X^2 \ln(1 + \sqrt{1 + 4\sigma_B^2})/2}, \quad (146)
\]

\[
K_{A}(\sigma_B, N) = 1 + \frac{N\sigma_X^2}{\sigma_B^2}, \quad (147)
\]

\[
K_{M}(\sigma_B, N) = 1 + \frac{N\sigma_X^2}{\mu_X^2 \ln(1 + \sqrt{1 + 4\sigma_B^2})/2}. \quad (148)
\]

An additive (resp. multiplicative) blinding scheme using the optimal strategy will achieve \( K_{A}(\sigma_B) \)-SNR (resp. \( K_{M}(\sigma_B) \)-SNR) and \( N \)-observations \( K_{A}(\sigma_B, N) \)-SNR (resp. \( K_{M}(\sigma_B, N) \)-SNR) security.

The first thing we can notice is that multiplicative blinding requires in general a much higher \( \sigma_B \) in order to achieve the same security level. In the case of additive blinding, \( K \) goes to one (remember \( K = 1 \) or \( K = 0 \) dB means perfect security) as \( 1/\sigma_B^2 \), however in the case of multiplicative blinding \( K \) goes to one only as \( 1/\sigma_X^2 \), which grows logarithmically with \( \sigma_B^2 \).

In order to show this behavior, we plot both \( K_{A} \) and \( K_{M} \) for different values of \( \sigma_B \) (Figure 6) and different values of \( N \) at a fixed \( \sigma_B \) (Figure 7). In this example, we have \( \mu_X = 4\sigma_X \). As it can be seen in Fig. 6, which refers to a single observation, for low values of \( \sigma_B \) multiplicative blinding is even more “secure” than additive blinding: however, additive blinding only needs a small increment of \( \sigma_B \) in order to achieve higher security. A similar behavior is observed in Fig. 7, where we compare the security of the different blinding schemes for an increasing number of observations, assuming \( \sigma_B^2/\sigma_X^2 = 30 \) dB and \( \sigma_B^2/\sigma_X^2 = 60 \) dB. If we set a \( K \)-SNR security level of 10 dB, in the additive case doubling the standard deviation (i.e., doubling the bits used for blinding) causes the number of observations needed to break the
Figure 6: $K$-SNR security of additive and multiplicative blinding at different $\sigma_B^2/\sigma_X^2$ values.

Figure 7: $N$-observations $K$-SNR security of additive and multiplicative blinding at $\sigma_B^2/\sigma_X^2 = 30$ dB and $\sigma_B^2/\sigma_X^2 = 60$ dB.
system to grow from $10^4$ to about $10^7$, whereas in the multiplicative case such a number grows more or less from 50 to about 100.

While in the additive case the security of the system grows exponentially with the number of bits of the security parameters, in the multiplicative case the security of the system grows only linearly with the number of bits of the blinding value. This is not enough to state multiplicative blinding secure according to the standard cryptographic definitions. Nevertheless, there can be practical cases in which the security offered by multiplicative blinding can still be exploited.

6 Conclusion

In this deliverable we have presented the main achievements in the field of s.p.e.d. regarding the development of signal processing primitives. The most important primitive turned out to be the proper signal representation in order to encode a signal within a ciphertext. Based on a suitable signal representation for the encrypted domain, we have proposed and studied three processing primitives, namely the encrypted domain Discrete Fourier Transform, the composite representation of signals and the additive/multiplicative blinding, which can be successfully used as building blocks to design more complex protocols.

References


