Processing in the Encrypted Domain using a Composite Signal Representation

Tiziano Bianchi[1], Thijs Veugen[2], Alessandro Piva[1], and Mauro Barni[3]*

[1] Dipartimento di Elettronica e Telecomunicazioni, Università di Firenze
Via S. Marta 3, 50139, Firenze, Italy
tiziano.bianchi@.unifi.it, alessandro.piva@.unifi.it
[2] TNO Information and Communication Technology
P.O. Box 5050, 2600 GB Delft, The Netherlands, and
Information and Communication Theory Group, Delft University of Technology
P.O. Box 5, 2600 AA Delft, The Netherlands
thijs.veugen@tno.nl
[3] Dipartimento di Ingegneria dell'Informazione, Università di Siena
Via Roma 56, 53100, Siena, Italy
barni@dii.unisi.it

Abstract. The current solutions for secure processing in the encrypted domain are usually based on homomorphic cryptosystems operating on very large algebraic structures. Recently, a composite signal representation has been proposed that allows to speed up linear operations on encrypted signals via parallel processing and to reduce the size of the encrypted signals. Though many of the most common signal processing operations can be applied to composite signals, some operations require to process the signal samples independently from each other, thus requiring an unpacking of the composite signals. In this paper, we will address the above issues, showing both merits and limits of the composite signal representation when applied in practical scenarios. A secure protocol for converting an encrypted composite representation into the encryptions of the single signal samples will be introduced. Two case studies clearly highlights pros and cons of using the composite signal representation in the proposed scenarios.

1 Introduction

Signal processing in the encrypted domain (hereafter referred to as s.p.e.d.), i.e., the availability of signal processing tools that work directly on encrypted data,

* The work described in this paper has been partially supported by the European Commission through the IST Programme under Contract no 034238 - SPEED and by the Italian Research Project (PRIN 2007): "Privacy aware processing of encrypted signals for treating sensitive information". The information in this document reflects only the author's views, is provided as is and no guarantee or warranty is given that the information is fit for any particular purpose. The user thereof uses the information at its sole risk and liability.

represents a valid solution for all the applications where valuable or sensitive signals must be processed by non trusted entities. This new field of research is receiving an increasing attention from the cryptographic and signal processing communities, which is justified by the list of applications that would benefit from the availability of s.p.e.d. tools [1]: access to a database containing encrypted data or signals [2], database access by means of encrypted queries [3], remote processing of private data, like medical recordings or biometric signals, by non-trusted parties [4], transcoding of encrypted contents [5], buyer-seller watermarking protocols [6].

From a technical point of view, processing of encrypted signals is feasible by relying on probabilistic homomorphic encryption [7,8] and secure multiparty computation (MPC) [9,10]. In this paper we focus on techniques based on homomorphic encryption, since they permit to avoid interaction among the parties involved in the computation in the case of linear processing.

Additively homomorphic cryptosystems play a central role in s.p.e.d. theory. For such systems we have:

$$D[E[m_1] \cdot E[m_2]] = m_1 + m_2 \tag{1}$$

$$D[E[m_1]^c] = c \cdot m_1 \tag{2}$$

where D and E indicate, respectively, the decryption and encryption operators and c is a constant factor. In other words, an addition in the plain domain corresponds to a multiplication in the encrypted domain, hence allowing the application of many basic signal processing tools directly in the encrypted domain. This is the case of linear operators like the Discrete Fourier Transform (DFT), FIR filters, correlation and simple operations among two or more signals like componentwise signal addition.

A problem with the use of homomorphic encryption is that signals need to be encrypted sample-wise. 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. In [11], a compositite representation of signals that permits to greatly reduce the expansion factor due to encryption, while still allowing the exploitation of the homomorphic properties of the underlying cryptosystem to process signals in the encrypted domain, has been proposed. In addition to limiting the storage requirement, this representation allows the parallel processing of different samples, thus providing a considerable reduction of computational complexity in terms of operations between encrypted messages.

Though many of the most common signal processing operations, including block-wise linear transforms, FIR convolution, linear filtering, can be easily applied to composite signals, the use of composite representation in complex processing chains may present some problems. As a matter of fact, some operations may require to process the signal samples independently from each other, thus making impossible the processing of their composite representation.

In this paper, the above issues will be addressed, trying to show both merits and limits of the composite signal representation when applied in practical scenarios. In order to improve the flexibility of the proposed representation tech-

Fig. 1. Graphical representation of a M-polyphase composite representation having order R. 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.

nique, we will introduce a secure protocol for converting an encrypted composite representation into the encrytions of the single signal samples. Such a protocol can be used in situations where it is convenient to apply simultaneously processing techniques based on composite signal representation and algorithms requiring a samplewise encryption. The proposed techniques will be applied in two case studies, secure content retrieval and watermark embedding in the encrypted domain, showing the potentialities of the composite representation of signals when applied to complex s.p.e.d. scenarios.

2 Composite representation of signals

We now briefly review 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

$$a_C(k) = \sum_{i=0}^{R-1} a_i(k) B^i, \quad k = 0, 1, \ldots, M-1 \tag{3}$$

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. Several ways of partitioning $a(n)$ exist. A useful one is obtained by choosing $a_i(k) = a(iM + k)$ [11]. In this case, each composite word contains R samples which are spaced M samples apart in the original sequence, i.e., belonging to one of the Mth order polyphase components of signal $a(n)$: this representation is referred to as M-polyphase composite representation (M-PCR). A graphical interpretation of M-PCR is provided in Fig. 1.

While the composite representation may seem a trivial one, its use 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* $|a(n)| < Q \quad \forall n$, $B > 2Q$, *and* $B^R \leq N$ *where* N *is a positive integer, and let* $a_C(k)$ *be defined as in equation (3). Then, the following holds:*

$$0 \leq a_C(k) + \omega_Q < N \tag{4}$$

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

$$a_i(k) = \left\{ \left[(a_C(k) + \omega_Q) \div B^i \right] \bmod B \right\} - Q. \tag{5}$$

The proof of the theorem is given in [11]. To give a hint of the proof, just consider that $a_C(k) + \omega_Q$ can be considered as a positive base-B integer whose digits are given by $a_i(k) + Q$.

3 Merits and Limits of Composite Representation

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 (3), and viceversa that the original signal values can be correctly computed from the composite representation according to (5). 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 representation, if we work with a semantically secure cryptosystem like the one proposed in [12] the security is automatically achieved.

Let us now consider the case where the original signal samples $a(n)$ have been encrypted samplewise by using an additive homomorphic cryptosystem whose private key is owned by a party P_1; in such a case, the encryption of the composite representation can be performed directly in the encrypted domain by a second party P_2, by applying (3) 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 owner of the private key P_1, or performed by means of a properly designed interactive protocol involving P_1 and P_2.

In [11], 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 can not 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. In the following, we will 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.

4 Unpacking the Composite Representation in the Encrypted Domain

The party P_2 has the encrypted number $[x] = [a_C(k) + \omega_Q]$ presented as a B-ary integer. The number x contains R integers $x_i = a_i(k) + Q$, $0 \leq i < R$, each from the interval $[0, B)$ such that $\sum_i x_i B^i = x$. P_2 would like to compute the encrypted integers $[x_i]$ from $[x]$. The party P_1 knows the secret key of the cryptosystem. The parameters B and R are assumed known by both parties. Neither P_1 nor P_2 should gain any knowledge about x or the integers x_i at the end of the protocol. The number N equals the modulus of the homomorphic encryption system, e.g. Paillier [12]. We use [.] do denote encrypted numbers, and $\div$ to denote integer division, i.e. $x \div B = (x - x \bmod B)/B$.

Summary:	
Publicly known	the Pallier modulus N and its public key, the parameters B and R
Only known to P_1	the private Pallier key
Only known to P_2	The encrypted number $[x]$
Output for P_2 (not for P_1)	the encrypted numbers $[x_i]$ for $i = 0, \ldots, R-1$
The relation $x = \sum_{i=0}^{R-1} x_i B^i$ holds. During the protocol P_1 and P_2 are not allowed to learn anything about x (or the numbers x_i).	

We define ξ_i as the B-ary presentation of the integers x_i up to x_{R-1}, i.e. $\xi_i = \sum_{j=0}^{R-(i+1)} x_{j+i} B^j$. Then the main protocol looks like:
$\{x = \xi_0\}$
for $i := 0$ to $R-1$ do: { compute x_i from x }

1. $\{x = \xi_i\}$
   P_1 and P_2 jointly compute $[x \bmod B]$ from $[x]$ without revealing x or $x \bmod B$ to P_1 or P_2. $\{x \bmod B = x_i\}$
2. $x := x \div B \{x := \xi_{i+1}\}$

The second step could be computed in the encrypted domain by P_2, but the division by B requires a modular exponentiation to B^{-1}, which is an integer of size N. To avoid this costly exponentiation, we would like P_1 to compute this division in the plain domain. The problem is that P_1 is not allowed to learn x or $x \bmod B$. Therefore, P_2 will first blind x by adding a sufficiently large random number r before sending it to P_1. The size (number of bits) of r should be equal to size(x) + K, the parameter K being the security parameter which is usually something like 80. The addition $z = x + r$ can be computed by P_2 due to the homomorphic property of the encryption system: $[z] := [x] * [r] \bmod N$.

The blinding of x is also necessary for jointly computing $[x \bmod B]$ in an efficient and secure way, as is shown in subsection 4.1. In this protocol the encrypted bit $[s]$ is securely computed which indicates whether a carry-over occured in the addition of $x \bmod B$ and $r \bmod B$, i.e. $\{s = 1\} \equiv \{x \bmod B + r \bmod B \geq B\}$. This bit s is also used in the computation of $x \div B$ in step 2.

The idea of computing $x := x \div B$ securely in the plain domain, is mainly implemented by letting P_1 compute $z := z \div B$, and letting P_2 compute $r := r \div B$. As long as the relation $z = x + r$ is maintained, the next B-ary value x_i can be computed. We know that $(x + r) \bmod B = x \bmod B + r \bmod B - s * B$, since there is at most one reduction modulo B, which occurs when $s = 1$. Therefore,

$$\begin{aligned} z \div B &= \frac{z - z \bmod B}{B} \\ &= \frac{x + r - (x + r) \bmod B}{B} \\ &= \frac{x + r - x \bmod B - r \bmod B + s * B}{B} \\ &= x \div B + r \div B + s \end{aligned}$$

So the relation $z = x+r$ can be maintained, by letting P_1 substract the encrypted bit s from $z \div B$, which is actually the correction of the carry-over that occurred during the B-ary addition of x and r. However, P_1 is not supposed to learn whether a carry-over occurred, because that would leak information about x. Therefore, P_2 will not send the encrypted bit s to P_1, but a blinded version of this bit: $s' = r' - s$, where r' is a sufficiently large random number of $K+1$ bits. The number s' is added to $z \div B$ by P_1, and the blinding factor r' is added by P_2 to $r \div B$, such that the relation $z = x + r$ is maintained.

The above derivation leads to the following unpacking protocol, using variables $[z]$ and r, where r is an unencrypted random variable only known to P_2, and $[z]$ is an encrypted variable whose content is blinded for P_1:
$\{x = \xi_0\}$

1. P_2 chooses a random number r of size(x)+K bits, and computes $[z] := [x+r]$ by computing $[z] := [x] * [r] \bmod N$
   $\{$ P_2 blinds x for $P_1\}$
2. P_2 sends $[z]$ to P_1 who decrypts it to z
3. for $i := 0$ to $R - 1$ do: $\{$ compute x_i from x $\}$

(a) $\{z = x + r; x = \xi_i\}$
P_1 and P_2 jointly compute $[x \bmod B]$ from $[z]$ without revealing x or $x \bmod B$ to P_1 or P_2
$\{x \bmod B = x_i\}$
P_2 will obtain the encrypted bit $[s]$ such that
$\{s = 1\} \equiv \{x \bmod B + r \bmod B \geq B\}$
(see subsection 4.1 for more details)

(b) P_2 chooses a random variable r' of $K+1$ bits, and computes $[s'] := [r' - s]$ by computing $[s'] := [r'] * [s]^{-1} \bmod N$
$\{$ P_2 blinds s for $P_1\}$

(c) P_2 sends $[s']$ to P_1 whe decrypts it to s'

(d) $\{$ Compute $x := \xi_{i+1}\}$
P_1 computes $z := z \div B + s'$ and P_2 computes $r := r \div B + r'$
$\{z := z \div B + s' = (x \div B + r \div B + s) + (r' - s) = \xi_{i+1} + r\}$

4.1 Computing $[x \bmod B]$

Summary:	
Publicly known	the Pallier modulus N and its public key, the parameters B and R
Only known to P_1	the private Pallier key, the integer z
Only known to P_2	the random number r
Output for P_2 (not for P_1)	the encrypted number $[x \bmod B]$
The relation $z = x + r \bmod N$ holds (and $x = \xi_i$). During the protocol P_1 and P_2 are not allowed to learn anything about x (or the number $x \bmod B$).	

To compute $[x \bmod B]$ from $[z]$ we have the following protocol between the owner (P_2) of (the encrypted) x and the owner (P_1) of the private key for decryption:

1. P_2 asks P_1 for $[c]$ with its encrypted bits $[c_i]$ such that $c = z \bmod B$ and $c = \sum_i 2^i c_i$.
2. Perform the bitwise comparison protocol with $[c]$ and $d = r \bmod B$. At the end of the protocol P_2 receives from P_1 the encrypted bit $[s]$ such that $\{s = 1\} \equiv \{c < d\}$.
3. P_2 computes $[y] = [x \bmod B]$ by computing $[y] := [c] * [d]^{-1} * [s]^B \bmod N$ $\{y := c - d + B * s\}$. This works because $x \bmod B = (z - r) \bmod B = (c - d) \bmod B$. So y is either $c - d + B$ or $c - d$ depending on whether $c < d$ or not.

To see that the encrypted bit $[s]$ indeed satisfies $\{s = 1\} \equiv \{x \bmod B + r \bmod B \geq B\}$ observe that $c = z \bmod B = (x + r) \bmod B = (x \bmod B + d) \bmod B$, so c can only be smaller than $d = r \bmod B$ when the addition of $x \bmod B$ and d leads to a reduction modulo B.

The bitwise comparison protocol (step 2) has the largest computational complexity. A good comparison protocol to use is by Damgård, Geisler and M. Krøigaard (DGK) [13]. This protocol uses an encryption system that is fine-tuned to small numbers and since B is relatively small compared to N, this advantage can be effectively exploited. To that end in step 1, P_1 will have to provide the bits c_i encrypted by this modified encryption system, and in step 2, P_1 is asked to switch back to the original encryption system.

4.2 Security analysis

The main requirement is that both P_1 and P_2 are not allowed to learn anything about the number $x = \xi_0$ (or any part of it). We assume that both P_1 and P_2 honestly follow the steps of the unpacking protocol. However, they can be curious, so they are allowed to do various computations with the values they obtain. We informally proof that our main security requirement holds.

Especially P_1, who owns the private key, is allowed to decrypt any value it receives. For this reason, we chose to blind any sensitive (encrypted) value that is sent from P_2 to P_1, by adding a large enough random number. The security that is obtained for P_1 is therefore statistically: the probability of P_1 guessing the value of x (or some part of it) is negligible, i.e. smaller than 2^{-K}, K being the security parameter. The values that are obtained by P_1 are z (which is blinded by r) and R times the value of s' (which is blinded by r'). In each round the value of z is modified by using the current values of z and s'. Since the current values are blinded, the new value of z can again be considered as a blinded value. In the DGK comparison protocol $\log B$ extra values are sent from P_2 to P_1 but these are fully blinded as well as the comparison result s (see [13] for more details and a security proof of the comparison protocol). Since all these values are indeed blinded, it's statistically infeasible for P_1 to learn anything about $x = \xi_0$.

Party P_2 will not learn anything about the value of x (or some part of it), because P_2 only obtains encrypted versions and both the Paillier and the DGK systems are semantically secure. This means that the security for P_2 is computationally: given the amount of computational power that is nowadays available, it is infeasable for P_2 to compute any plain part of the encrypted values he obtains. Due to the semantical security of the used encryption systems (Pallier and DGK), it is even impossible for P_2 to decide whether two encrypted values contain the same plain value or not. The values that are obtained by P_2 are:

- The encrypted variable $[x]$, where $x = \xi_0$.
- The encrypted variable $[z]$, where $z = \xi_0 + r \bmod N$ and r is a random variable chosen by P_2.
- The R encrypted variables $[c]$, where $c = (\xi_i + r) \bmod B$ and r is each time computed from r and r'.
- The $R \cdot \log B$ encrypted bits $[c_i]$, where $c = \sum_i c_i 2^i$.
- The R encrypted bits $[s]$, where $\{s = 1\} \equiv \{c < d\}$ and $d = r \bmod B$.
- The R encrypted variables $[s']$, where $s' = s + r'$ and r' is a random variable chosen by P_2.
- The R encrypted variables $[y]$, where $y = x \bmod B$, $x = \xi_i$ and thus $y = x_i$.

The only unencrypted values that P_2 obtains (r, d and r') are computed from random numbers that where generated by P_2 itself, and since all other values are indeed encrypted, it's computationally infeasible for P_2 to learn anything about $x = \xi_0$.

4.3 Final remarks

The exponentiation $[s]^B$ in step 3 of the $[x \bmod B]$ protocol can be avoided when P_1 gives back both $[s * B]$ and $[s]$, at the expense of an increased communication complexity.

It is important that $\xi_0 + r$ is smaller than N, the size of the encryption system, because carry-overs modulo N lead to miscomputations. This means that there could be a problem when computing z in the first step in the main unpacking algorithm. There are two solutions:

1. Let the modulus of the encryption system be at least $\text{size}(x) + 2^K$ bits long, so we take a buffer of K bits to avoid carry-over at the cost of including less signals in one encrypted number. This is the solution used in the case study.
2. Use a different comparison protocol for the first steps that can cope with large encrypted numbers. They exist but are computationally harder. See e.g. the protocol of Schoenmakers and Tuyls [14]. Our main unpacking algorithm becomes more complicated but the overall complexity, especially for large R, will however not be influenced much.

We point out that when B is a power of two the unpacking protocol can be implemented using the protocol in [14] for converting an encrypted composite value into the encryption of the bits. However, such a solution would be computationally more expensive than the specifically tailored protocol we propose. If B is a power of two, an efficient solution could be obtained relying on a garbled circuit, since the unpacking circuit is just a reordering of the wires. Nevertheless, our solution can deal with arbitrary B, which can be useful in optimizing the number of samples that can be packed together.

5 Two Case Studies

The feasibility of the proposed solutions in a practical scenario is verified by considering two cases studies, the retrieval of a digital content in an encrypted database [15] and the secure embedding of a watermark as described in [16]. In the following, we will analyze the behavior of both cases considering the two application scenarios described in section 3.

5.1 Secure Content Retrieval

In the secure content retrieval scenario, a client downloads from a server a set of encrypted feature vectors representing some digital contents and matches them with his own query vector using a secure matching protocol. In our case study, we assume that the secure matching protocol consists of a correlation (inner product) between feature vectors followed by a comparison with a threshold.

The schemes we consider are depicted in Fig. 2. The first solution employs a samplewise representation encrypted with the Paillier cryptosystem until the end of the processing chain. The second solution performs feature vector encryption and correlation using a composite representation, but switches back to

(a)

(b)

Fig. 2. Secure content retrieval scenarios: (a) samplewise representation; (b) composite representation. The dashed boxes indicate the blocks which differ between the two solutions.

a samplewise representation for comparing the result of each correlation with the threshold. Thanks to the properties of the composite representation, this allows us to compute R correlations using a single correlation between the query vector and the encrypted composite vectors [11]. In our analysis, we will concentrate on the blocks which differ between the two solutions, namely the encryption, correlation, and unpacking. We also point out that the encryption of the feature vectors is performed only once when they are added to the database, so its complexity is kept separated from correlation and unpacking. The complexity of packing in the plain domain has been assumed negligible.

In order to estimate the complexity of the proposed solutions, we quickly review correlation in the encrypted domain. If we assume that the vectors have M components, the correlation between an encrypted vector a and a plaintext one x is

$$[\rho] = \prod_{k=0}^{M-1} [a_k]^{x_k}, \tag{6}$$

where $[a_k]$ and x_k are the components of the encrypted and plaintext vectors, respectively. In the following, we will assume that both a_k and x_k are rounded to q-bit integers, where $q = \log_2 Q$. If we consider a database of L vectors, the above implementation will require ML Paillier encryptions of q-bit numbers, ML exponentiations to q-bit numbers, and $(M-1)L$ multiplications.

When using the composite representation, since R features are encrypted in a single word, we have only ML/R Paillier encryptions. Moreover, if we encode the kth component of R vectors in a composite component, with a single correlation between the query vector and a composite vector we are able to compute R correlations:

$$[\rho_C] = \prod_{k=0}^{M-1} [a_{C,k}]^{x_k} = \prod_{k=0}^{M-1} [\sum_{i=0}^{R-1} (a_k^{(i)})^{B^i}]^{x_i} = [\sum_{i=0}^{R-1} (\rho^{(i)})^{B^i}]. \tag{7}$$

Hence, on average we have ML/R exponentiations and $(M-1)L/R$ multiplications.

As to the unpacking protocol, required to obtain the R correlations to be compared to a threshold, the complexity can be estimated as follows. In our scenario, we have to unpack L/R encrypted values. For each value, the client (P_2) needs to perform one encryption of a $\log_2 N$-bit number ($[r]$) and one multiplication ($[z] = [x] * [r]$) followed by R repetitions of the inner loop. Considering

Table 1. Computational complexities of the building blocks in the two SCR scenarios. Values indicate the estimated number of multiplications modulo N.

samplewise scenario	
Encrypt	$6qML$
Correlate	$(6q+4)ML - 4L$
composite scenario	
Encrypt	$\frac{6}{R}qML$
Unpack (P_1)	$(\frac{3}{2}\log_2 N + \frac{3}{2}\ell t + \frac{13}{2}\ell)L$
Correlate	$\frac{1}{R}(6qM + 4M - 4)L$
Unpack (P_2)	$(\frac{6}{R}(\log_2 N - K) + 6K + \frac{3}{4}\ell^2 + 16\ell + 12)L$

the loop, for each cycle P_2 needs to perform three multiplications, one Paillier encryption of a ℓ-bit number ($[d]$), one Paillier encryption of a K-bit number ($[r']$), and one exponentiation to a ℓ-bit number ($[t]^B$). Moreover, the DGK comparison protocol requires that P_2 performs $\ell/2$ ($m_i \oplus x_i$ in [13]) plus 2ℓ (c_i in [13]) multiplications and $\ell/2$ exponentiations to $\log_2 u$-bit numbers (used for blinding $[x \bmod B]$ towards P_1) over the field of DGK cryptosystem, where $\mathbb{Z}_u$ is the plaintext space of DGK and $\log_2 u \approx \ell + 2$ [13]. During each cycle of the loop, the server (P_1) needs to perform one Paillier decryption and one Paillier encryption of a ℓ-bit number. Moreover, the DGK comparison protocol requires that P_1 also performs ℓ DGK encryptions of bits and checks ℓ times whether a DGK encrypted number is zero.

The overall complexity has been summarized in Table 1 in terms of number of multiplications modulo N (MM), where N is the modulus of the Paillier cryptosystem, assuming that one multiplication modulo N^2 costs as four MMs. We also assume that the size of the field used by the DGK cryptosystem is the same as N. The following further computational estimates are used: Paillier encryptions of a n-bit number cost $6n$ MMs (we assume r^N terms are precomputed); Paillier decryptions cost $3\log_2 N/2$ MMs (see [12]); exponentiations to n-bit numbers cost $3n/2$ multiplications over the underlying ring; DGK encryptions of bits cost $1/2$ MMs (we assume h^r are precomputed); checking whether a DGK encrypted number is zero costs $3t/2$ MMs, where t is a security parameter of DGK [13].

As to the communication complexity, in the loop P_2 sends to P_1 one Paillier encryption and ℓ DGK encryptions, whereas P_1 sends to P_2 two Paillier encryptions plus ℓ DGK encryptions. This should be executed for each correlation, resulting in $L(6+2\ell)\log_2 N$ bits. We must also consider that in the samplewise case P_2 receives from P_1 ML encryptions, resulting in $2ML\log_2 N$ bits, whereas in the composite case P_2 receives from P_1 ML/R encryptions, resulting in $2ML\log_2 N/R$ bits. As to the round complexity, the loop of the unpacking protocol runs in two rounds. Since it is serially repeated R times for each

(a)

(b)

Fig. 3. Computational complexity of the SCR scenarios at different required precisions: (a) number of MMs for each correlation; (b) number of MMs for each encrypted vector.

Fig. 4. Communication complexity of the SCR scenarios at different required precisions.

composite word, assuming to process composite words in parallel we have $2R$ rounds.

The computational and communication complexities of the two scenarios have been compared in Fig. 3 and Fig. 4, respectively, in terms of MMs per correlation/encrypted vector and kbits per correlation. The curves in Fig. 3-(a) take into account only the complexity of correlation and unpacking, i.e., the operations that should be performed during each access to the database, whereas the complexity of encrypting the feature vectors is compared in Fig. 3-(b). We have assumed $M = 256$. As to security parameters, we have chosen $\log_2 N = 1024$, $K = 100$, and $t = 160$. The complexity has been evaluated as a function of the required precision q, examining values of q ranging from 8 to 32 bits. As shown in [11], the values of ℓ and R can be related to the size of N and the required precision. Namely, we can assume $\ell = 2q + 1$ and $R = \lfloor \frac{\log_2 N - K}{\ell} \rfloor$.

As can be seen, the composite scenario when using the unpacking protocol has some advantages with respect to the samplewise scenario. The computational complexity of P_2 is reduced, even considering the overhead of the unpacking protocol. Furthermore, the communication complexity of the composite solution is well below that of the samplewise solution. On the other hand, the composite scenario requires P_1 to perform some extra computations at every query, which after few queries balances the gain obtained in encrypting the database, and the unpacking protocol requires some extra rounds. However, we point out that the composite solution permits to greatly reduce the storage requirements, which may compensate for the increased computational effort of P_1.

5.2 Watermark Embedding in the Encrypted Domain

In the secure watermark embedding (SWE) scenario, a seller receives the bits of the watermark encrypted with the public key of a buyer – the output of a previous protocol between him and the buyer – and embeds them into a set of features extracted from the digital content he owns. In our case study, we assume that the content is an image and that the features are obtained by applying a block 2-dimensional Discrete Cosine Transform (2D-DCT) to the pixel values. We also assume that the seller wants to perform the inverse DCT (IDCT) of the watermarked features in the encrypted domain, before sending them to the

(a)

(b)

Fig. 5. Secure watermark embedding scenarios: (a) pixelwise; (b) composite. The dashed boxes indicate the blocks which differ between the two solutions.

buyer. This is justified by the possibility of applying a perceptual mask [17] to the watermarked image.

The schemes we consider are summarized in Fig. 5. The first solution employs a pixelwise representation encrypted with Paillier cryptosystem until the end of the processing chain. The second solution performs image encryption and SWE using a composite representation of the image. In our analysis, we will concentrate only on the blocks which differ between the two solutions, namely the encryption, SWE, s.p.e.d. IDCT and unpacking. The complexity of packing in the plain domain has been assumed negligible.

In order to estimate the complexity of the proposed solutions, we quickly review quantization index modulation (QIM) watermarking in the encrypted domain. The image is divided into square blocks of $M \times M$ pixels and an $M \times M$ (I)DCT is applied to each block. The watermark encoder then chooses a subset $\mathcal{M}$ of the available DCT coefficients that will be used to carry the watermark. If we consider a generic $x \in \mathcal{M}$, QIM embedding can be expressed as

$$y = \mathcal{Q}^{2\Delta}(x) + w \cdot \mathrm{sgn}(x - \mathcal{Q}^{2\Delta}(x))\Delta = \tilde{x} + w \cdot \Delta_x \qquad (8)$$

where $w \in \{0, 1\}$ is the embedded bit, $\mathcal{Q}^{\Delta}(\cdot)$ is a uniform quantizer with step Δ, and $\mathrm{sgn}(\cdot)$ is the sign function. In the following, we will assume that both $\tilde{x}$ and Δ_x are rounded to q-bit integers, where $q = \log_2 Q$. In order to keep secret the exact set of features, the embedder outputs all the DCT coefficients of the image in encrypted form. The marked coefficients are obtained as in (8). The other coefficients are simply rounded to q-bit integers before encryption. Hence, s.p.e.d. QIM can be expressed as

$$[y] = \begin{cases} [\tilde{x}] * [w]^{\Delta_x} & x \in \mathcal{M} \\ [x] & x \notin \mathcal{M} \end{cases} . \qquad (9)$$

If we consider an image of size I and $|\mathcal{M}| = W$, the above implementation will require I Paillier encryptions of q-bit numbers, W exponentiations to q-bit numbers, and W multiplications. As to s.p.e.d. IDCT, we refer to a separable implementation by means of one dimensional M-point IDCTs: in order to keep the size of the output limited, we preferred a direct implementation instead of a fast one [18]. The complexity of a direct 1D-IDCT is M^2 multiplications and a separable implementation requires $2M$ 1D-IDCTs, hence the complexity of a s.p.e.d. 2D-IDCT is $2M^3$ exponentiations. Processing the entire image requires I/M^2 2D-DCTs, which results in $2MI$ exponentiations to q_T-bit numbers, where $q_T = \log_2 Q_T$ and Q_T is the scale factor used to quantize the cosine values [18].

Table 2. Computational complexities of the building blocks in the two SWE scenarios. Values indicate the estimated number of multiplications modulo N.

	pixelwise	composite
Encrypt	$6qI$	$\frac{1}{R}6qI$
SWE	$(6q+4)W$	$6qW + \frac{W}{R}[6(R-1)\ell + R]$
s.p.e.d. IDCT	$12q_T MI$	$\frac{1}{R}12q_T MI$
Unpack (P_2)	-	$(\frac{6}{R}(\log_2 N - K) + 6K + \frac{3}{4}\ell^2 + 16\ell + 12)I$
Unpack (P_1)	-	$(\frac{3}{2}\log_2 N + \frac{3}{2}\ell t + \frac{13}{2}\ell)I$

Fig. 6. Computational complexity of the SWE scenarios at different required precisions.

When using the composite representation, since R pixels are encrypted in a single word, we have only I/R Paillier encryptions. Moreover, also the complexity of 2D-DCT decreases to MI/R exponentiations [11]. However, in order to perform SWE we have to express the terms $[w]^{\Delta_x}$ according to the composite representation. This can be obtained as (s.p.e.d. packing)

$$[y_C(k)] = [\tilde{x}_C(k)] * \prod_{i=0}^{R-1} \left\{[w(iM+k)]^{\Delta_{x(iM+k)}}\right\}^{B^i} \tag{10}$$

and requires $R-1$ exponentiations to ℓ-bit numbers ($\ell = \log_2 B$) and $R-1$ multiplications. Hence, we have W exponentiations to q-bit numbers (to compute $[w]^{\Delta_x}$ terms) plus W/R s.p.e.d. packings and W/R multiplications.

As to the unpacking protocol, the complexity can be estimated as in the SCR scenario, considering that we have to unpack I/R encrypted values. The overall complexity in terms of number of multiplications modulo N has been summarized in Table 2.

The complexities of the two scenarios have been compared in Fig. 6, in terms of MMs per pixel. We have assumed $W = I/16$, i.e., we mark one DCT coefficient out of 16 and $M = 8$. As to security parameters, we have chosen $\log_2 N = 1024$, $K = 100$, and $t = 160$. The complexity has been evaluated as a function of the required precisions q and q_T. To this end, we assumed $q_T = q$ and we have examined values of q ranging from 8 to 32 bits. As shown in [11], the values of ℓ and R can be related to the size of N and the required precision. Namely, we can assume $\ell = 2q_T + q + 1$ and $R = \lfloor \frac{\log_2 N - K}{\ell} \rfloor$.

As to the communication complexity, the pixelwise scenario can be assumed to have no complexity, since all the processing chain is on the server's side. The communcation complexity of the composite scenario depends on the unpacking protocol and can be evaluated as in the SCR case. With the chosen parameters, the communication complexity of the unpacking protocol ranges from 57.3 kbit/pixel ($q = 8$) to 204.8 kbit/pixel ($q = 32$).

As can be seen, the complexity of the composite SWE scenario when using the unpacking protocol is far above that of the pixelwise scenario. Note that the complexity is dominated by the unpacking protocol, which has to be executed for each pixel of the watermarked image.

5.3 Discussion

In the SCR scenario, the main advantage is that a composite word is the result of parallel correlations between a set of encrypted vectors and a template. In such a case, the cost of encrypting the vectors component-by-component is higher than the cost of unpacking few composite words. We deem that also other scenarios may benefit from the use of a composite approach, for example when the cost of storing an encrypted samplewise representation for the entire life of the application is not acceptable. For example, an untrusted server could hold a large encrypted database, whose private key is owned by a third trusted authority, using the composite representation for storage efficiency. However, every time a query is made to that database it is likely that the matching algorithm will require a pixelwise representation. Such a representation can be obtained on the fly using the unpacking protocol between the server and the trusted authority. Nevertheless, if the number of encrypted composite words that have to be unpacked is equal to the original amount of encrypted data, then the use of the unpacking protocol may be too onerous with respect to a samplewise implementation. This is the case of the SWE scenario, where each pixel of the encrypted and watermarked image has to be extracted from the composite representation in order to apply a perceptual mask. In such cases, if we know in advance that some processing requiring a samplewise representation is needed, it may be worth using a samplewise representation in the whole processing chain.

The computational complexity of the proposed protocol can be further reduced by using some common optimizations. First of all, all random values needed by the protocol, that is $[r]$, $[r']$ and $[d]$, may be pre-computed. This will add a little to the storage requirements of the application. However, such values are needed only for a short time, since they can be computed when the processor is idle and discarded right after the end of the protocol. Hence, their effects on the storage requirements are different from those of the encrypted signals. Furthermore, in a practical implementation scalar products between vectors could be implemented relying on vector addition chains [19]. Such an optimization will affect in the same way both samplewise and composite solutions, however its use may be relevant for a comparison with an unencrypted version of the same protocols.

6 Conclusions

In this paper we have highlighted both merits and limits of the recently proposed composite representation of signals, a tool for signal processing in the encrypted domain, when applied in practical scenarios. In order to improve the flexibility of the composite representation, we have introduced a secure protocol for converting an encrypted composite representation into the encrytions of the single signal samples. Such a protocol has been applied in two case studies, secure content retrieval and secure watermark embedding, where both composite and samplewise representations may be adopted according to the requirements of the processing tasks. The results show that in a secure content retrieval scenario the overhead due to the unpacking protocol is lower than the complexity of a processing chain entirely based on a samplewise representation. Conversely, in a secure watermark embedding scenario the samplewise representation may be preferable, especially when it is required to perform some processing task that would require the unpacking of the composite representation. In general, the unpacking protocol is useful when the number of encrypted composite words is much less than the original amount of encrypted data, or it is not convenient to store an encrypted samplewise representation for the entire life of the application. Further research will be devoted to the possible optimization of the proposed unpacking protocol and to comparisons with alternative solutions for the unpacking protocol (e.g., garbled circuits).

References

1. Erkin, Z., Piva, A., Katzenbeisser, S., Lagendijk, R.L., Shokrollahi, J., Neven, G., Barni, M.: Protection and retrieval of encrypted multimedia content: When cryptography meets signal processing. EURASIP Journal on Information Security **2007, Article ID 78943, 20 pages** (2007)
2. Agrawal, R., Srikant, R.: Privacy-preserving data mining. In: SIGMOD '00: Proceedings of the 2000 ACM SIGMOD international conference on Management of data. Volume 29(2)., ACM Press New York, NY, USA (2000) 439–450
3. Shashank, J., Kowshik, P., Srinathan, K., Jawahar, C.: Private content based image retrieval. In: IEEE Conference on Computer Vision and Pattern Recognition. (June 2008) 1–8
4. Canny, J.F.: Collaborative filtering with privacy. In: IEEE Symposium on Security and Privacy. (2002) 45–57
5. Johnson, M., Ishwar, P., Prabhakaran, V., Schonberg, D., Ramchandran, K.: On compressing encrypted data. IEEE Trans. on Signal Processing **52**(10) (October 2004) 2992–3006
6. Memon, N., Wong, P.: A buyer-seller watermarking protocol. IEEE Trans. on Image Proc. **10**(4) (Apr. 2001) 643–649
7. Rivest, R., Adleman, L., Dertouzos, M.: On data banks and privacy homomorphisms. In R.A. DeMillo et al., ed.: Foundations of Secure Computation, New York, Academic Press (1978) 169–179
8. Goldwasser, S., Micali, S.: Probabilistic encryption. Journal of Computer and System Sciences **28**(2) (1984) 270–299
9. Yao, A.C.: Protocols for secure computations. In: Proceedings of Twenty-third IEEE Symposium on Foundations of Computer Science, Chicago, Illinois (November 1982) 160–164
10. Cramer, R., Damgård, I., Nielsen, J.B.: Multiparty computation from threshold homomorphic encryption. Lecture Notes in Computer Science **2045** (2001) 280–299
11. Bianchi, T., Piva, A., Barni, M.: Efficient pointwise and blockwise encrypted operations. In: Proc. of ACM Multimedia & Security Workshop 2008, Oxford, UK (2008) 85–90

12. Paillier, P.: Public-key cryptosystems based on composite degree residuosity classes. In: Eurocrypt 1999. Volume 1592., Springer (1999) 223–238
13. Damgård, I., Geisler, M., Krøigaard, M.: Efficient and secure comparison for on-line auctions. In: Australasian Conference on Information Security and Privacy - A CSIP 2007. Volume 4586., Springer (July 2007) 416–430
14. Schoenmakers, B., Tuyls, P.: Efficient binary conversion for Paillier encrypted values. In: Advances in Cryptology - EUROCRYPT 2006, Springer (July 2006) 522–537
15. Lu, W., Swaminathan, A., Varna, A.L., Wu, M.: Enabling search over encrypted multimedia databases. In: Proc. of SPIE. Volume Media Forensics and Security. (January 2009)
16. Kuribayashi, M., Tanaka, H.: Fingerprinting protocol for images based on additive homomorphic property. IEEE Transactions on Image Processing **14**(12) (Dec. 2005) 2129–2139
17. Bartolini, F., Barni, M., Cappellini, V., Piva, A.: Mask building for perceptually hiding frequency embedded watermarks. In: Proc. 5th IEEE Int. Conf. on Image Processing, ICIP'98. Volume I., Chicago, IL, USA (October 1998) 450–454
18. Bianchi, T., Piva, A., Barni, M.: Discrete cosine transform of encrypted images. In: Proc. of ICIP 2008, San Diego, CA, USA (2008) 1668–1671
19. Pippenger, N.: On the evaluation of powers and related problems. In: Proc. 17th IEEE Symp. Foundation of Computer Science, Houston, TX, USA (1976) 258–263