Secure Signal Processing in the Cloud

Juan Ramón Troncoso-Pastoriza *Member, IEEE* and F. Pérez-Gonzalez *Senior Member, IEEE*

Abstract

In recent years, the paradigm of Cloud Computing has become a very appealing concept both for providers, that can benefit from hiring out their extra computation and storage resources, and for users, that can avoid the initial investment on resources by outsourcing their processes and data to a cloud.

This game changer concept of outsourcing has also arrived at multimedia processing, and cloud technology has become an idoneous platform to deliver multimedia processing services. Nevertheless, privacy-aware multimedia applications dealing with sensitive signals find an insurmountable barrier in the privacy issues derived from the fuzzy nature of processing and location of an untrusted environment like a cloud.

This paper provides a comprehensive overview that takes a look at the role of Signal Processing in the Encrypted Domain in providing a practical solution for privacy-preserving multimedia cloud processing, highlighting prototypical outsourced applications, like secure adaptive filtering and private eHealth. We cover the theoretical framework for privacy, involving privacy measures, and also practical approaches, with a special emphasis on noninteractive solutions based on fully homomorphic encryption, discussing their actual applicability, tradeoffs and practicality and foreseeing the future advances that will revolutionize the field of noninteractive private outsourcing of multimedia processes.

Index Terms

Multimedia Cloud, Cloud Privacy, Signal Processing in the Encrypted Domain

I. INTRODUCTION

In recent years, the paradigm of *Cloud Computing* has gained an increasing interest from the academic community and from the commercial point of view. The Cloud is a very appealing concept both for the providers, that can benefit from hiring out their extra computation and storage resources, and for the users, that can avoid the initial investment on resources by outsourcing their processes and data to a cloud.

This game changer concept of outsourcing has also arrived at multimedia processing; cloud technology has become an idoneous platform to deliver multimedia processing services [1]. Implementing these services in a cloud is a challenging task due to heterogeneity issues and QoS requirements, but multimedia processing is extremely amenable to distributed parallel processing and it is a perfect fit for computing over grids, content delivery networks, server based computing and P2P multimedia computing; all of these services can be efficiently supported by a *multimedia-aware cloud* whose architecture is adapted to the provision of multimedia-oriented services; furthermore, we are also witnessing the reciprocal behavior as new *cloud-aware multimedia* processing

J. R. Troncoso-Pastoriza and F. Pérez-Gonzalez are with the Signal Theory and Communications Dept., University of Vigo, Spain. E-mail: {troncoso,fperez}@gts.uvigo.es.

F. Pérez-Gonzalez is with Gradiant, Vigo, Spain, and with the Electrical and Computer Engineering Dept., University of New Mexico, USA

applications, that take into account and effectively profit from the benefits that the Cloud may offer to them, keep constantly arising [1].

There are currently some technological challenges that *multimedia clouds* still need to tackle in order to be fully operational; those have become very active research topics. But the most important issues that can hold back the widespread adoption of an outsourced scenario like a cloud are actually security and privacy. Both concepts are very close to each other in the Cloud, as there can be no privacy without security. Nevertheless, privacy is a more specific requirement, and it is related only to sensitive data and/or processes. While many research efforts are devoted nowadays to guaranteeing security [2] in clouds, dealing with aspects such as authentication through federated identities or basic encryption of the managed data, the issue of preserving data privacy and addressing the different data protection legislations remains open. The privacy problem in the Cloud is a severe concern, mainly because data and signals in a cloud can be distributed among different servers and even different countries with their own data protection legislation. This fuzzy nature of processing and location in clouds can negatively affect the trust that users put on these systems, as they face the risk of losing control over their data and processes when they are outsourced. In fact, many cases of privacy invasion by some cloud services have been reported by the press; they are mainly related to location services (like Street View) and mail automated processing for personalized advertising. These privacy breaches have triggered legal decisions against the abusive use of private data by cloud providers. Hence, it has become evident that privacy issues can constitute a severe barrier for cloud adoption[1], but they can be addressed through the use of technological solutions developed within the emergent field of *Signal Processing in the Encrypted Domain* (SPED) [3].

This paper shows the issues and challenges that *multimedia clouds* face with respect to privacy, and explains, as a comprehensive overview, how they are being progressively tackled through the application of SPED techniques, focusing on exemplifying use cases. We will present the most crucial aspects of a SPED privacy-preserving multimedia cloud, ranging from the theoretical level to implementation problems and practical techniques developed within the SPED field. Special emphasis will be put on noninteractive solutions based on homomorphic encryption, discussing their actual applicability, tradeoffs and practicality and foreseeing the future advances that will revolutionize the field of noninteractive private outsourcing of multimedia processes.

II. CLOUD AND PRACTICAL SCENARIOS

Cloud Computing [4] comprises the provision of computing and storage services to users through the Internet, commonly supported by heterogeneous infrastructures. A public cloud provides determined services to individuals and organizations that want to take advantage of either an ubiquitous access to their resources or make use of a huge computing power without having to invest in the needed hardware acquisition or maintenance. Typically, cloud services are presented in three layers (Fig. 1):

- Infrastructure as a Service (IaaS): The lower architectural layer, representing the *raw* cloud hardware resources put at the service of the customer, providing mainly virtualized storage and processing capabilities. Examples of IaaS are EC2 (Elastic Compute Cloud) from Amazon, or Azure from Microsoft.

[1] In 2010, the European Commission published a report on "The future of Cloud Computing. Opportunities for European Cloud Computing beyond 2010", in which the opportunities and challenges of Cloud Computing are highlighted.

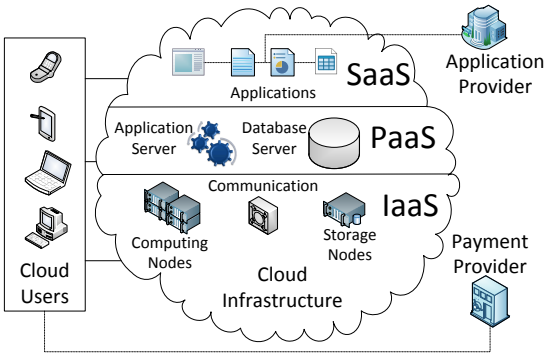


Fig. 1. Service levels and stakeholders in a cloud-based scenario.

- Platform as a Service (PaaS): The second layer, usually providing an application server and a database in which the customer can develop and run his/her own applications, coded on top of the provided Application Programming Interface (API). Google App Engine is an example of PaaS.
- Software as a Service (SaaS): This is the most extended use of the Cloud as such, where a series of applications (typically office packages) are provided to the final user, that can access them without any installation or maintenance. Examples are Google Docs, Zoho or Office365.

Each of these layers comprises three essential functionalities that summarize the purpose of a cloud: *storage*, *transmission* and *computation*. For all of them, privacy is a must when dealing with sensitive data in a public (untrusted) cloud; hence, mechanisms that enforce privacy guarantees over the stored, transferred and processed data are needed. Secure transmission and storage encompass many technical cryptographic approaches that already provide the desired protection; these two problems are not specific for the Cloud, and they have been studied and effectively addressed for a long time. Nevertheless, protecting the involved data and signals *while they are being processed* is a relatively new concept, specific to outsourced or distributed systems where untrusted environments come into play; this is the crucial point in which we will focus our discussion. Before describing what trust is and how it is distributed in a cloud scenario, let us present the involved stakeholders. We will abstract ourselves from some of the design problems typical of cloud architectures not directly related to privacy (i.e., network, device and service heterogeneity, load balancing and elasticity), and present the functional elements/roles played by the stakeholders of a cloud system from a privacy-aware perspective:

- *Cloud Infrastructure*: The company that owns the HW (datacenters) supporting the cloud services provided in upper layers. These companies can be dedicated IaaS providers or organizations with a surplus of computing power and unused resources that they hire to external customers. All the storage and processing outsourced to the cloud takes place at the cloud infrastructure.
- *Software/Application provider*: Software developers that produce applications run on top of the cloud infrastructure/platform and offered as a SaaS to end-users.
- *Payment provider*: Party in charge of billing for the consumed cloud resources.
- *Customers/End-users*: Parties that contract and make use of a service on the cloud infrastructure, supplied

by the application provider; they may also outsource data and/or signals to the cloud infrastructure so that the contracted application can process them and provide the desired results.

Many cloud-based services have been rapidly deployed in recent years, and they are becoming completely ubiquitous. But they are often provided with no privacy guarantees. The service level agreement that an end-user signs with the cloud provider involves the acceptance that the latter will have total access to any data/signal that the user outsources unencrypted to the cloud; this means that the user must "blindly" trust the cloud provider and expect that the data will be used only for a correct purpose inside the cloud infrastructure (*data/signals privacy*). Additionally, for application providers that deploy their software on top of a PaaS, a mutual trust relationship is established: the PaaS provider must trust that the code it will execute is not malicious, while the application provider must trust that the PaaS will use the software in a correct way, managing the appropriate licenses and limiting the execution to the authorized users; this software may also be subject of intellectual property protection (*process privacy*). Besides the secure authentication needs that these trust relationships rise, it is a must (and a legal requirement in some countries) that blind trust be substituted by privacy-preserving technical means that enforce the privacy rights of the customers. We are concerned here with *signals privacy*: outsourced sensitive data from the customers must never be accessed by the cloud infrastructure. It must be noted that there are also further problems, like billing for cloud usage, that involve privacy constraints and require secure protocols that fall out of the scope of this paper.

Let us depict here three exemplifying scenarios of signal processing applications that showcase the privacy problem in the Cloud (Fig. 2):

a) Outsourced Biometric Recognition: Biometric signals coming from faces, iris, fingerprints, voice,... are inherently sensitive signals, as they hold information that can uniquely identify their owner. In a biometric recognition system, the collected biometric information of an individual is contrasted against the templates from a database stored at a server (or distributed among several collaborative servers) in order to determine whether the individual is recognized by the system. Outsourcing the storage of biometric databases to a cloud yields many advantages (easy parallelization of the recognition logic and matching process), but the outsourced signals must be protected whenever the cloud is untrusted. That is, the presented biometric sample must be kept encrypted while it is processed in the server, and the database templates should also be kept encrypted within the cloud infrastructure [5]. An exemplifying scenario of outsourced biometric recognition is that of a CCTV system where the faces of the recorded citizens are matched against a database of potential criminals. The faces of innocent citizens must not be indiscriminately disclosed to the recognition logic.

b) e-Health: e-Health is the paradigmatic scenario where sensitive signals (DNA [6], Electrocardiograms, Magnetic Resonance Images,...) are involved; hence, privacy is a crucial aspect. Cloud adoption would be highly beneficial for health institutions, allowing for the outsourcing of the storage and processing needs that their own systems may not be able to cope with (e.g. *HealthGrids*). Many collaborative studies and statistical retrospective analyses for specific diseases could take advantage of cloudified medical databases: one or several institutions that own a possibly distributed private patient database can release it (with patient data conveniently protected) so that research groups and labs can perform accurate statistical analyses on those non-synthetic data and obtain the desired results.

Nevertheless, whenever the used cloud does not belong to the Health institution, automated cloud outsourcing

for e-Health systems is completely infeasible from a legal perspective. Privacy-preserving technological solutions that enforce signal protection and enable these outsourced scenarios are thus a must.

c) Outsourced adaptive or collaborative filtering: Filtering itself is an essential block of signal processing, present in any system we can think of, from voice processing in a smartphone to complex volume rendering in the production of a synthetic 3D movie. More specifically, adaptive filters are the only effective filters in nonstationary environments; they are present in *multiuser communications* scenarios dealing with sensitive signals, where the privacy of the users must be protected from each other and from the central processing server [7], whenever there is one (in the Cloud). *Model-Reference Adaptive Control* (MRAC) is an example application dealing with adaptive controllers in many industrial contexts like robot manipulation, ship steering, aircraft control or metallurgical/chemical process control; when the controller is outsourced to a cloud, the signals coming from the plant have to be sent to the cloud; if they are sensitive (e.g., in the case of an industrial plant whose processes have to be kept secret for protecting the industrial property), a privacy-preserving solution must be applied.

Additionally, collaborative filtering, involving data from multiple sources, has found application in recommender systems, similarity-based searches and prediction in social networks or e-commerce portals. In this scenario, the similarity between two users is commonly calculated as a normalized correlation between their respective preference vectors; a recommender system will find the most similar users to provide them with a list of items that have been highly rated by those similar users. Hence, the recommender system, that may be implemented on a cloud, has access to all the sensitive data comprised in the user preferences database; these data must be protected.

Privacy preservation in the presented scenarios faces many challenges, related to the development of secure protocols that efficiently provide the desired functionalities without hindering the provider's capabilities to normally develop its activities and deploy its services. In order to effectively guarantee privacy and evaluate the impact of a given privacy preserving protocol on the utility provision, a unified privacy framework for multimedia clouds is needed, as a means to measure privacy and formalize the privacy settings.

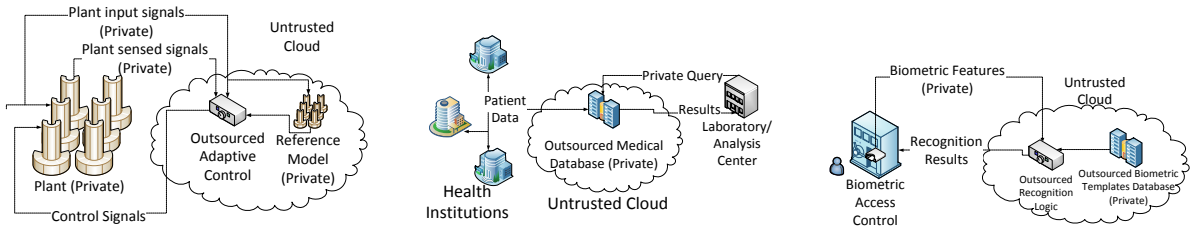


Fig. 2. Representative scenarios of Privacy-aware signal processing in the Cloud (MRAC, e-Health and Biometric verification).

III. FORMALIZING THE PRIVACY PROBLEM: MEASURING PRIVACY

The trust relationships between the Cloud stakeholders introduced in the previous section depict and define a trust model. Within this model, the behavior of those parties considered "untrusted" has to be established. All the participants in a secure protocol interchange and obtain a series of intermediate messages until the

protocol finalizes and the result is provided to the appropriate parties; the set of all these messages conforms the *transcript* of the protocol; untrusted parties may use it for dishonest purposes. *Semi-honest* parties follow the predetermined protocol without any deviation, but they might be curious and try to infer additional information from the transcript, while *malicious* parties will deviate from the protocol and either introduce fake data or forge some of the intermediate protocol messages in order to gain some advantage for inferring extra private information. Typically, privacy-preserving protocols tackle the semi-honest case, and present further extensions for dealing with the more realistic malicious case.

Once the adversary model is defined, a proper privacy framework should clearly establish means to quantitatively measure the *privacy*, or, conversely, the *private information leakage*. The evaluation of the leakage that a given protocol produces determines its suitability for a set of privacy requirements, that fix a determined privacy level for a given cloud application dealing with sensitive signals. It is worth noting that typical *cryptographic* measures for security and secrecy (besides Shannon's perfect secrecy) *usually* rely on complexity theory and hardness assumptions for computationally bounded adversaries; contrarily, signal processing measures for the conveyed information in a signal are based on fundamental information-theoretic magnitudes that do not take into account any computational limitation. As SPED joins together these two disciplines, there has been some controversy on the definition of a universal "privacy measure".

From the cryptographic point of view, in the semi-honest adversary model, a simulator argument is typically used to prove the statistical (or computational) indistinguishability of a correct protocol transcript and the output of a random simulator that impersonates the honest parties. This means that a (polynomial-time constrained) adversary will not be able to correctly guess with non-negligible probability whether the transcript comes from the honest party or from the random simulator, not being able to infer any information from the true transcript.

The simulator argument suffices to determine the validity of a privacy-preserving interactive protocol; but from the signal processing point of view, it does not measure how much private information is leaked from the final output of the protocol, whenever that output is disclosed to the cloud or to other parties (e.g., in the statistical analysis eHealth scenario). This privacy leakage is not easily quantified, as it generally depends on the specific process or algorithm that the input data undergoes at the cloud. A cryptographic approach that has recently consolidated as a widely used measure of privacy is the so called *Differential Privacy* [8], initially proposed for private database statistical calculations (see Box II). Informally, the concept of differential privacy in a statistical database states that any possible outcome of an analysis should be "almost" equally likely for two databases that differ in just one record. Hence, the performed statistical analyses will not disclose significant information about one individual of the database. More formally, for any two databases D_1 and D_2 that differ in at most one element, and a query f run on any of these databases with results in $\mathbb{R}^k$, an ϵ-differentialy private mechanism ($\epsilon > 0$) builds a mechanism comprising a randomized function[2] $\mathcal{K}$ that is called instead of f, such that the probability of any output set $S \subseteq \mathbb{R}^k$ is not significantly higher (within a multiplicative factor e^{ϵ}) for D_1 than for D_2.

Differential privacy has not been extensively applied to privacy-preserving Cloud Computing yet, but it is closely related to the exemplifying scenarios that we have presented: in all of them, there is a private database,

[2]A randomized function $\mathcal{K}$ yields a non-deterministic result with a probability distribution parameterized by the input arguments of $\mathcal{K}$.

whose data are subject to a process in the cloud; the result of this process should not disclose information about the sensitive inputs. The cloud scenario introduces the particularity of an untrusted environment that will require further protection on the stored, communicated and processed database, and not only on the output values of the analyses. Nevertheless, it is evident that differential privacy can be applied to quantify how much information (bounded by the parameter ϵ) is leaked about an individual when disclosing a result of a query masked by $\mathcal{K}$, that can be any function or process, like robust statistics or predictive filtering. Conversely, it is also crucial to measure and evaluate the needed random distortion that $\mathcal{K}$ introduces for achieving a required ϵ-differential privacy level (see Box II). We will further discuss this essential privacy-utility distortion tradeoff in Section V.

IV. GETTING PRACTICAL: PRIVACY TOOLS FROM SPED

The theoretical framework for determining privacy measures and utility tradeoffs discussed in the previous section has to be translated into the implementation of efficient primitives that can be practically used in actual privacy-aware environments like the cloud-based scenarios that we are focusing on. The most important cryptographic tools available for the development of such primitives are briefly explained in the following paragraphs. For an extensive description of these techniques and their main applications we refer the reader to the tutorial article [9].

d) Homomorphic Processing: Some cryptosystems present homomorphisms between the clear-text and cipher-text groups, that allow for the execution of a given operation directly on encrypted values, without the need for further decryption.

Paillier [10], with its variants, is the most widely used homomorphic cryptosystem in privacy-preserving protocols, due to its semantic security and its additive homomorphism, as it provides a good trade-off between efficiency, encryption size and cipher expansion. In its most common form, a Paillier encryption of a number $x \in \mathbb{Z}_n$ is $E_P[x] = (1 + x \cdot n) \cdot r^n \mod n^2$, where n is the product of two safe primes, and r is chosen at random in $\mathbb{Z}_n^*$. The additive homomorphism of Paillier allows for computing the addition of two encrypted numbers and the product of an encrypted number and a known integer:

$$E_P[x + y] = E_P[x] \cdot E_P[y] \mod n^2, \quad E_P[x \cdot k] = E_P[x]^k \mod n^2. \tag{1}$$

Since Gentry's seminal paper [11] in 2009, there have been many proposals of fully homomorphic cryptosystems (FHE), able to execute any operation in the encrypted domain without the need for decryption. While their applications are really promising, at this point they are still not totally practical, due to the big size of the encryptions and the need of costly *bootstrapping* operations on the ciphers to achieve the full homomorphism (see Box III).

e) Searchable Encryption and Private Information Retrieval (PIR): Searchable encryption has been identified by DARPA as one of the technical advances that can be used to balance the need for both privacy and national security in information aggregation systems[3]. It is a cryptographic primitive that allows for checking whether a given pattern has a match inside some encrypted data. It is used for keyword searches in encrypted

[3]In 2002 the Information Science and Technology Study Group from DARPA published the report "Privacy with Security".

databases, and presents the advantage of protecting also the performed queries; as a counterpart, it is not very flexible, and it has also scalability issues. Additionally, PIR methods solve a somehow dual privacy problem, as they allow for private queries on a database, in which the contents are not necessarily encrypted, concealing just the query and its result from the database owner.

f) Secure Multiparty Computation (SMC) and garbled circuits: Secure Two-Party Computation was born in 1982 with Yao's Millionaires' problem. The problem was cast as a binary comparison of two quantities in possession of their respective owners, who are not willing to disclose to the other the exact quantity they own. The proposed solution was based on *garbled circuits*, in which both parties evaluate a given circuit, gate by gate, without knowing the output of each gate.

While homomorphic computation is very efficient for implementing arithmetic operations, circuit evaluation [12] is still more efficient when dealing with binary tests, comparisons and thresholding. Traditionally, the search for efficient solutions has led to proposals for changing between integer and binary representations in order to efficiently implement both arithmetic and binary operations.

g) Secure (approximate) interactive protocols: SMC is a generic solution to almost any privacy-preserving problem but its huge communication overhead makes it unpractical. Furthermore, in a cloud scenario, garbled circuits would need that the user itself generate the circuit offline and send it to the Cloud to be executed on the private inputs. Unless a trusted third party generates the circuit, the customer cannot deal with such computational cost, that surpasses that of executing the circuit in the clear. Hence, garbled circuits get (in general) partially invalidated for their use in the Cloud.

Specific interactive protocols developed for privately executing a certain function are an efficient workaround for this problem. The needs of communication and interaction rounds can be reduced with respect to a generic garbled circuit. Furthermore, it is sometimes possible to let the private function introduce a certain power-bounded error to the outputs; then, either a simpler approximate function can be found, or it may be possible to optimize the cost of a secure implementation by representing the original function as a linear and a non-linear part: the former is performed homomorphically in the Cloud, while all the interaction is limited only to those non-linear parts that will be run as secure protocols.

h) Obfuscation mechanisms (differentially private): Obfuscation consists in adding some kind of (random) noise to the sensitive signals in order to partially or totally conceal them [13], in such a way that some relatively complex operations can be performed on them, provided that the original meaning of the data is preserved. These methods were initially proposed as the only protection mechanism for private data, as they have the advantage of allowing complex operations with a relatively low computational and communication burden. The counterpart is accuracy, for the induced error, and privacy, as some information about the obfuscated data can be inferred from the intermediate steps of the process. Nevertheless, with the advent of differential privacy, obfuscation mechanisms that provide a differentially private output have been combined with homomorphic encryption that keeps the intermediate values secret. In fact, this is an appropriate solution for private access to encrypted databases in a cloud.

A. Practical limitations of privacy tools

Endowing any signal processing application with privacy protection through any of the previous techniques will have the cost of a negative impact on the efficiency of the algorithms. There are two main magnitudes that determine the efficiency of a secure protocol in any scenario: its computational overhead and the communication it needs in terms of used bandwidth and number of interaction rounds.

The restrictions and peculiarities of a cloud scenario essentially limit two parameters: a) the bandwidth of the customer-cloud link, that cannot be continuously active, and b) the computational overhead for the customer: there is a very asymmetric relationship with the server, the latter having a much higher computational power and parallelizable computing resources. Nevertheless, a cloud has the advantage of being able to absorb the additional computation load and distribute it among several nodes, playing with the spatio-temporal overhead and providing almost transparent response times to the customer. Hence, the fundamental bottleneck for privacy-preserving implementations in cloud is communication. It must be noted, though, that the use of more cloud resources (either in nodes or in time) will always mean a higher bill for the customer: that is the price to pay for privacy.

The presence of many users can also pose challenges to the key distribution and key management in a privacy-preserving cloud computing architecture. In order to be combined, all the aforementioned privacy-preserving techniques must work with a unique key: e.g., homomorphic addition is only possible between values encrypted under the same key. There is the possibility of counting on trusted third parties that perform reencryption, either having the corresponding keys or through user delegation using a proxy-reencryption mechanism; nevertheless, resorting to trusted third parties is a solution that should be avoided, as it implies shifting the trust from the cloud to those third parties, but it cannot enforce privacy in the absence of trust.

From the available privacy-preserving techniques, homomorphic computation is the most promising one. Currently, additive homomorphisms are widely used in private implementations, and they can efficiently cope with any linear filter or transform. As an example, they have been used to implement Fast Fourier Transforms [14]. This is especially relevant for the Cloud, as any operation that can be performed homomorphically can be run unattended in a cloud without any extra communication needs. Unfortunately, an additive homomorphism is not enough for many practical applications dealing with non-linear functions such as comparisons, thresholding or binarization. Furthermore, the blow-up problem (see Box I) is also present even in linear functions, requiring a certain amount of communication to reduce the accumulated scaling factors before the cipher gets useless.

Let us depict here two cases of secure filtering in which the blow-up problem is readily visible:

Discrete Fourier Transform (DFT): The well known definition of the M-point DFT of a sequence x_n is the following:

$$X_k = \sum_{n=0}^{M-1} x_n W^{nk},\ k = 0, 1, \ldots, M-1,$$

with $W = \exp(-j2\pi/M)$. The encryption of a complex signal x_n with quantized real part $\hat{x}_{R,n}$ and imaginary part $\hat{x}_{I,n}$ (see Box I), can be denoted [14] as $E[\hat{x}_n] = \{E[\hat{x}_{R,n}], E[\hat{x}_{I,n}]\}$, using a quantization scale Δ. The *twiddle factors* W^{nk} undergo the same quantization process $C(u) = \hat{W}^u = C_R(u) + jC_I(u) = \lceil \Delta_2 \cos\left(\frac{2\pi u}{M}\right) \rfloor - j\lceil \Delta_2 \sin\left(\frac{2\pi u}{M}\right) \rfloor$, with a possibly different scale Δ_2. Bianchi *et al.* [14] propose a privacy-preserving implementation of the DFT using Paillier encryptions, for which clear-text additions correspond to

products between encryptions. They devise two encrypted implementations: a direct one and a radix-2^k one. The former takes the form

$$E[\hat{X}_k] = \left\{ \prod_{n=0}^{M-1} \left(E[\hat{x}_{R,n}]^{C_R(nk)} E[\hat{x}_{I,n}]^{-C_I(nk)} \right), \right. \\ \left. \prod_{n=0}^{M-1} \left(E[\hat{x}_{R,n}]^{C_I(nk)} E[\hat{x}_{I,n}]^{C_R(nk)} \right) \right\}, \; k = 0, 1, \ldots, M-1.$$

If the quantized integer inputs are bounded by Δ, the outputs are upper bounded by $M(\Delta\Delta_2 + \frac{\Delta+\Delta_2}{\sqrt{2}} + \frac{1}{2})$. Hence, the modulus n of the cipher must be larger than this bound so that the operations do not wrap-around (cipher blow-up, see Box I). The radix-2 implementation is analogous to the direct one, by applying the recursive relationships given by the binary FFT butterflies; this radix-2 implementation implies recursive multiplications of the $\hat{W}^u$ at each stage, such that the outputs magnitude will grow with $M\Delta\Delta_2^{\log_2 M-1}$; $\log_2 M$ is the number of stages for the radix-2 implementation. A radix-2 implementation in the clear is a much more efficient construction than a direct one; nevertheless, its encrypted version imposes a much stronger constraint on the size of the cipher (exponential in the number of levels) in order to be privately implementable without blowups.

Adaptive Filtering: We pick a simple (yet powerful) adaptive filtering algorithm like the LMS (Least Mean Squares) to picture the blow-up problem. The LMS comprises two processes that jointly form a feedback loop: 1) a transversal filter $\boldsymbol{w}_n$ with N_E coefficients applied to the input sequence u_n, and 2) an update process of the coefficients of the transversal filter, based on the instantaneous estimation error e_n between the output of the filter y_n and a desired response d_n. For real signals, these two processes are expressed as

$$y_n = \boldsymbol{w}_n^T \boldsymbol{u}_n, \quad \boldsymbol{w}_{n+1} = \boldsymbol{w}_n + \mu \boldsymbol{u}_n \underbrace{(d_n - y_n)}_{e_n},$$

where μ is the step size and $.^T$ denotes transpose.

When outsourced, the filtering and update equations of the LMS will be executed at the cloud; let us assume that the cloud has access to the input signal u_n, and the customer sends the encrypted reference d_n, obtaining as output the encrypted desired signal y_n. The values of the filter coefficients $\boldsymbol{w}_n$ will be kept encrypted in the cloud. It must be noted that if the cloud does not know u_n in the clear, a solution based solely on additive homomorphic encryption will not be enough, as the product between $\boldsymbol{w}_n$ and $\boldsymbol{u}_n$, if both are encrypted, will require an interactive multiplication protocol.

The real inputs $u_n, d_n, \boldsymbol{w}_o, \mu$ have to be quantized prior to being either encrypted or operated with other encryptions, so the system will be working with $\hat{u}_n, \hat{d}_n, \hat{\boldsymbol{w}}_0, \hat{\mu}$ instead, all of them with a scale factor Δ. After the first iteration, the obtained $\hat{y}_0 = \hat{\boldsymbol{w}}_0^T \hat{\boldsymbol{u}}_0$ and $\boldsymbol{w}_1$ will accumulate a scale factor Δ^2. After k iterations, the accumulated scale factor in $\hat{y}_k$ will be Δ^{k+2}.

If the real inputs are all in the range $(-1, 1)$, after roughly $\lfloor \frac{\log n}{\log \Delta} \rfloor - 2$ iterations, the cipher will not be able to accommodate the increased plaintext size, and the results will wrap-around: we have a blow-up. In the case of adaptive filtering (in general, for any iterative algorithm with recursive updates), the blow-up problem is far more serious than for the DFT case, as it imposes a bound on the number of iterations that can be run; this completely invalidates the sole homomorphic encryption solution for adaptive filtering, unless other techniques are used, as we will see in the next section.

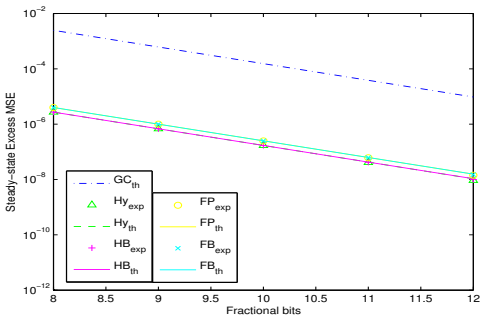
V. MAPPING COMPLEX TO REAL SOLUTIONS

There are many challenges that SPED must face in order to provide efficient privacy-preserving solutions in a cloud scenario. Specifically, we have named several tradeoffs that have to be optimized: for a given privacy-preserving solution, a balance must be found among the following four magnitudes: computational load, communication (bandwidth and interaction rounds), accuracy (error propagation), and privacy level (differential privacy). At the same time, the technical limitations of some solutions (like the cipher blow-up problem) have to be worked out, with the corresponding impact on the previous tradeoffs. In general, the appropriateness of a privacy-preserving solution could be measured as a cost function that gives weights to each of the four magnitudes, depending on the scenario for which it is devised.

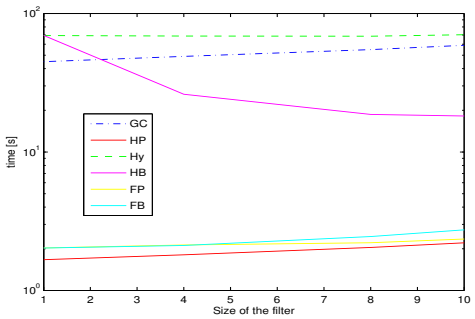
Privacy vs Accuracy: As mentioned before, the privacy level can be established through a given privacy metric; if we choose ϵ-differential privacy, we can determine the optimal randomization mechanism (noise added to mask the output function) in order to achieve a certain privacy level [15], that is to say, define the optimal utility masked function for a given level of differential privacy [16].

Computation vs Communication vs Accuracy: Let us come back to the adaptive filtering scenario [7] to showcase the interrelations among these three magnitudes, together with a solution to the cipher blow-up problem. As we have seen, private adaptive filtering cannot be addressed only with the use of additive homomorphic computation without incurring in cipher blow-up. Hence, a secure primitive that handles the rescaling or requantization of intermediate values is needed; how this requantization is performed, and how much error can this process allow will determine the efficiency of the chosen solution. In fact, [7] shows that it is possible to lower both the computation and communication complexity by playing with the output error power. This is achieved through an approximate interactive rounding protocol that minimizes the interaction rounds and the computational complexity for both client and server, while slightly increasing the error that is induced on the outputs. Fig. 3 shows the comparison between the proposed solution in [7] and other combinations of homomorphic encryptions and garbled circuits or exact interactive protocols for rounding; the Figure plots the three relevant magnitudes (output error, computation and communication), as a function of input accuracy (fractional bits used for representing the input values), filter size for a fixed number of iterations (50), and performed iterations with a fixed filter size (5 taps). The plots show that the trade-off achieved by the fast approximate solution is much better than for the others. In fact, the computation and communication complexity are greatly reduced and they are close to those of a protocol based solely on homomorphic encryption, plotted as a reference minimum complexity obtained without tackling the blow-up problem. Regarding the computation needs, Fig. 3 shows a large time complexity for a relatively small number of iterations; this is due to a sequential implementation that tests the total needed resources. If we take into account the availability of computing resources in a cloud architecture, parallelization is a natural option for trading time complexity by resource utilization. SPED primitives are very amenable to be parallelized, and privacy-preserving implementations can take advantage of this fact to balance the latency of the secure cloud and keep the responses within a reasonable time scale. Similarly, for some applications it is possible to achieve a reduction on the complexity through packing techniques (see Box IV).

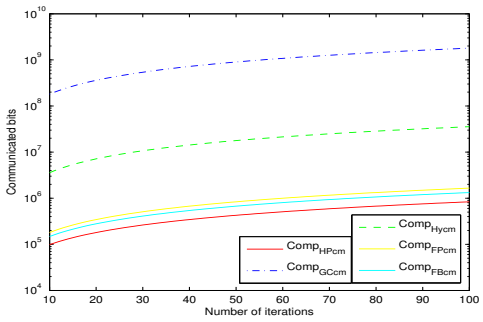
Communication in the Cloud Cost Function: If we look into cloud requirements and limitations, as presented in previous sections, the cost function that determines the optimal tradeoff for the Cloud will give the



(a) Accuracy Loss (Mean Square Error)



(b) Computation complexity (time)



(c) Communication Complexity (bits)

Fig. 3. Comparison of efficiency and accuracy loss for different privacy-preserving approaches to the Block LMS protocol: GC - Garbled Circuits, HP - Homomorphic Processing, Hy - Hybrid (HP with GC requantization), HB - Hybrid with packing, FP - Approximate Protocol, FB - FP with packing.

largest weight to communication overheads. The optimal case consists in limiting the communication between the customer and the cloud to only transmitting the encrypted input values and receiving the final results. For a generic function, this can only be achieved if the so called *somewhat homomorphic cryptosystems* or fully homomorphic ones are used (see Box III). As an example of this fully noninteractive processing, we can recall the outsourced biometric scenario, where the cloud holds the biometric database of (encrypted) templates, and the client sends a fresh biometric features vector, also encrypted. Troncoso and Pérez-González [17] present a solution based on a extension of a fully homomorphic cryptosystem that copes with low cardinality non-binary plaintexts, and applies it to face verification with optimally quantized Gabor coefficients. The cloud executes a second order comparison function on 4000-dimensional feature vectors to obtain a score value whose sign determines the matching of the presented face. Communication is strictly limited to the encrypted features and the encrypted result. The system can perform an unattended verification in around ten seconds with a sequential implementation, for which the client load is 30 times lower than the server load. When implemented in a cloud, massive parallelization allows for a real-time response.

Privacy as the Key Factor: For some outsourced scenarios, privacy is an absolute requirement: independently of the efficiency factors, the system is not feasible if a solution that addresses privacy protection is not available. This is the case of the e-Health data analysis scenario, where a database of private information about some individuals is made available to organizations and parties wishing to perform statistical analyses on the

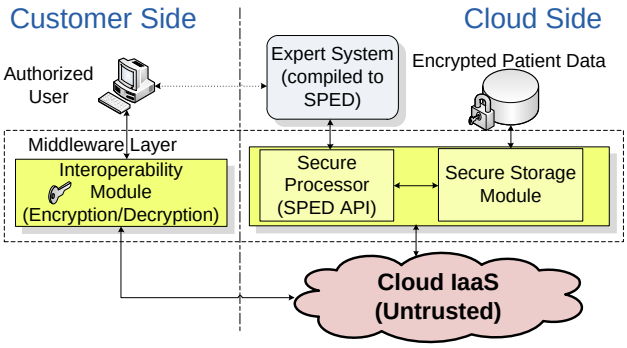


Fig. 4. Generic architecture for a SPED-privacy protected Health Cloud.

database records. The privacy of those individuals is the main concern of this system. In quantifiable terms, a certain ϵ-differential privacy has to be guaranteed for any statistical analysis that is allowed on the data. Additionally, this scenario poses further challenges mostly related to the heterogeneous structure of medical databases, which mix qualitative and numerical data, and may have errors and missing values; these anomalies are really difficult to cope with when there is no direct clear-text access to individual data [18]. Nevertheless, for structured databases, like genomic ones, that are thoroughly checked before they are released, these difficulties are not present. There are recent approaches for addressing these problems like Uhler *et al*'s [19] who propose differentially private methods for releasing minor allele frequencies (MAF), χ^2 statistics and p-values; they also provide a differentially private method for finding genome-wide associations through a penalized logistic regression. Unfortunately, most of the existing approaches assume that the noise is added to the released statistics directly by the database owner, so this is not applicable if the database has to be released through an untrusted cloud. In that case, additional encryption mechanisms for the individual records stored in the cloud have to be devised. Ideally, a privacy-preserving eHealth cloudified scenario should conform to the generic architecture depicted in Fig. 4.

Other limitations: Generally, homomorphic encryption is not able to combine data encrypted by multiple private keys from different users in the cloud. López-Alt *et al.* [20] propose the so called *on-the-fly multiparty computation*. It consists in cloud-aided Secure Multiparty Computation, for which the cloud can non-interactively perform arbitrary, dynamically chosen computations on data belonging to arbitrary sets of users selected on-the-fly. This approach also minimizes the need for interaction, as each user is only involved in two steps: initial encryption of the inputs and final decryption of the outputs. This is achieved thanks to a new type of encryption called *multikey FHE* (Fully Homomorphic Encryption). In this construction, a ciphertext resulting from a multikey evaluation can be jointly decrypted using the secret keys of all the users involved in the computation; a relinearization step is performed prior to decryption, so that the latter does not depend on the specific function that outputs the to-be-decrypted result. Such cryptosystem is based on the NTRU (stemming from "N-th degree TRUncated polynomial ring", a special case of GGH, Goldreich-Goldwasser-Halevi, using

convolutional modular lattices), that is turned into a fully homomorphic multikey encryption, with ciphertext and key size polynomial (with large constants) on the number of users.

In this setting, each user can encrypt his/her data (with his/her own key) and upload them encrypted to the cloud. Thanks to the fully homomorphic capacity of the cryptosystem, the cloud itself can execute a chosen function on the inputs from several users. The obtained result is in an encrypted form; in order to decrypt it, the cloud needs the cooperation of all the users whose encrypted data were used as inputs to the function. This is a very nice setting, in the sense that if certain data from a user are employed in a calculation, nobody can decrypt the result until that user agrees to it, by cooperating in the decryption step. The drawbacks of this solution, as with any other solution based on current FHE cryptosystems, are the inefficient homomorphic operations (computation) and the huge sizes of the ciphertexts (communication).

VI. OTHER APPROACHES AND FUTURE LINES

Throughout this paper we have motivated the need for privacy when outsourcing processes to cloud environments, answering to the legal and moral rights to privacy of the customers accessing a cloud. These rights get materialized in the technological need of a generic enough noninteractive solution for private process outsourcing, for which Cloud Computing is a paradigmatic case.

The first and most fundamental challenge for this scenario deals with the definition and quantification of privacy in the Cloud. The range of cloud applications is rich and varied, from very simple spreadsheet applications to rendering of synthetic video scenes or finding the solution to complex optimization problems. While differential privacy can provide a framework for quantifying the privacy level of the (noisy) disclosed values, it has not yet been extensively applied to cloud, and there is a real challenge in determining and optimizing the tradeoff between the added noise-power (the degradation of the output) and the ϵ-differential privacy that can be achieved, especially when the calculated function is not a simple aggregation, but a complex function of the private input values. Another promising approach is the evaluation of which transformations and operations on the input data preserve differential privacy so that privacy analyses can be easily extrapolated.

The communication burden of most privacy-preserving approaches conflict with the essential limitation of the cloud with respect to the link with the customer. We can foresee that the most promising research lines that address and can potentially solve the open issues of private multimedia clouds are essentially related to unattended private processing without the need of collaboration or communication with the client. The development of efficient fully homomorphic encryption (FHE) that allow for the practical use of noninteractive homomorphic processing is the main research line that can lead to feasible solutions. There are several further challenges for FHE, like: a) the efficient private execution of non-linear functions; full homomorphisms commonly deal with a ring with addition and multiplication, but there is no direct mapping of nonlinear operations in the real numbers (like logarithms or thresholding) to these finite rings, so they have to be approximated somehow; and b) the efficient combination of inputs from different customers, each holding his/her own private key, a especially relevant issue in a multi-tenant and multi-user environment like a cloud. Meanwhile, privacy in the Cloud can be progressively addressed through the combination of additive homomorphic processing and interactive protocols that can effectively protect the sensitive signals in the cloud, at the cost of a computation and communication burden with respect to a non-privacy-preserving execution.

We have focused on the use of the computation resources of the Cloud, but the mere use of a cloud as a database also poses interesting challenges. There are very recent proposals that can deal with encrypted cloud searches in a privacy preserving fashion. As an example, PRISM [21] is a secure version of a MapReduce system, that consists in a cloud outsourced database in which encrypted information is uploaded for subsequent word searches. The proposed system runs parallel private queries using PIR mechanisms and then linearly combines the results to produce the output. The net privacy overhead for the queries multiplies by ten the computation time and introduces communication rounds with the client; thanks to the parallelization, the cloud reduces the perceived overhead in the response time to 11%. It must be pointed out also that the output of this protocol has a (low) probability of being erroneous. Again, also for the database approaches, communication and accuracy are fundamental parameters that still have to be optimized in future proposals.

Finally, there is another aspect of cloud privacy that we have not covered: resource usage computation and billing. Disclosing the individual usage of cloud resources can lead to very efficient attacks that can accurately estimate the behavior of a customer and the specific operations and processes that he/she performed in the Cloud. This is another notion of privacy not directly related to the outsourced sensitive signals themselves, but to the usage and consumption patterns of the cloud user; it can also be addressed through SPED techniques.

BOX I: ACCURACY AND SIGNAL REPRESENTATION: THE BLOWUP PROBLEM

All the known secure cryptographic techniques work on finite rings. Hence, all their inputs must be mapped to the appropriate ring $(\mathbb{Z}_n, +, \cdot)$, in which $+$ and $\cdot$ are modulo n operations. For bounded discrete data this is not usually a problem, provided that n is sufficiently large to accommodate all the input and output domains for such data; then, the finite ring operations are equivalent to the clear-text addition and product, and the cipher never "wraps-around". Nevertheless, (multimedia) signals are real numbers that need a determined representation and computation accuracy in order to yield acceptable results. Applying a quantization operation is a must for using cryptographic techniques with signals: the inputs to a secure protocol must be quantized prior to encryption.

There have been several proposals (and controversy) in terms of an accurate signal representation/quantization that perform this mapping from $\mathbb{R}$ to the cryptosystem finite ring $\mathbb{Z}_n$. The straightforward and most widespread approach consists in applying a scale factor and working in fixed-point arithmetic, like many FPGAs and DSPs actually do; fixed-point implies keeping a quantization or scale factor Δ, for which an input $x \in \mathbb{R}$ gets represented as $\hat{x} = \lceil \frac{x}{\Delta} \rfloor$, where $\lceil . \rfloor$ stands for rounding to the nearest integer. The quantization noise $\varepsilon = x - \hat{x} \cdot \Delta$ will get propagated from the inputs to the outputs of the employed signal processing algorithm. Hence, numerical stability and numerical accuracy of the algorithms come into play.

The flexibility of floating point and the large dynamic range that it allows for is lost when using the previous approach. Hence, different encodings have also been proposed, trying to mimic the IEEE 754 floating point standard in the encrypted domain. An example is the logarithmic encoding presented in [22], for which a bounded real $x \in [-l, l]$, with $l > 0$, can be represented as a tuple: $(\rho_x, \sigma_x, \tau_x) \in \{0, 1\} \times \{-1, 1\} \times \{0, \ldots, 2^\kappa - 1\}$, where ρ_x indicates whether x is zero, σ_x is the sign of x, and $\tau_x = \lceil -S \cdot \log_B \left(\frac{|x|}{C} \right) \rfloor$ with κ bits. C, B and S are constants that must be adjusted for the range of the involved numbers. This strategy achieves an increased dynamic range and better accuracy in the representation of small quantities, but it comes at the price of increased complexity for additions and multiplications between tuples.

Finally, even when working with fixed point arithmetic, there is no direct mapping between the real division and inverses in $\mathbb{Z}_n$, so the scale factor Δ affecting all the encrypted values will increase with each multiplication. This effectively limits the number k of allowed consecutive products that an encrypted algorithm can perform to a bound given by the maximum mappable number in the finite ring $\mathbb{Z}_n$. After a number k of successive products, the result ($\Delta^k \prod_{i=0}^{k-1} |x_i| > n$) may "wrap-around" due to the modulo reduction, the subsequent operations are not equivalent to their real counterparts anymore, and it is said that the cipher *blows up* [7]. This is a critical limitation that imposes the need to applying secure requantization protocols at intermediate steps.

BOX II: PRIVACY METRICS AND PRIVACY NOTIONS

Privacy metrics have been related and deeply entangled with statistical databases from their origin. The first notion of "privacy metrics" comes from a "privacy desideratum" by Dalenius in 1977, who defined for the first time the probabilistic notion of "disclosure"; he stated that in order to avoid disclosures from a given database, "nothing about an individual that could not be learned without access to the database should be learnable from the database". This is the most stringent privacy requirement that can be issued on any database; fulfilling it would imply having a perfectly private database. Unfortunately, Dwork [8] showed later that no useful database can comply with this requirement; only a completely uninformative, and therefore, useless, database can fulfill it.

Hence, the concept of privacy and the measure of disclosure of private information had to be relaxed. Other notions of privacy arose, like the concept of *non-privacy* [23], occurring when a computationally-bounded adversary can expose a $1 - \epsilon$ fraction of the database entries for all constant $\epsilon > 0$; this concept excludes even the weakest notions of privacy, and was later restricted to *attribute non-privacy*, that excludes the secrecy of any attribute when all other attributes are known to an attacker. The most flexible and useful notion of privacy presented to date is *differential privacy* [8], for which a randomized mechanism provides sufficient privacy protection (ϵ-differential privacy) if the ratio between the probabilities that two datasets differing in one element give the same answer is bounded by e^{ϵ}.

Opposed to these privacy notions, there are also information-theoretic concepts related to privacy coming from the area of software systems, like *secure information flow* [24], that ideally seeks the lack of leakage (there called *non-interference*), and tries to quantify the amount of produced leakage using entropy-based measures. The most widespread form of entropy in this area is that of *min-entropy*, defined as minus the logarithm of the probability of guessing the true value of a random variable with the optimal "one-try" attack: the probability of the most likely element; then, the leakage is maximized when the prior distribution for the guessed variable is uniform. Finally, there are other privacy notions based on the use of minimizing Bayes' risk as a measure of information leakage [25].

It would be difficult to choose one of the previous privacy notions over the others, as they all present advantages and shortcomings, but the sweet fact here is that all these measures are closely interrelated: they all measure the "privacy loss" produced by disclosing a function of some private inputs, and the needed noise that has to be added to this function. For a given ϵ-differential privacy level, it is indeed possible to find explicit relationships and tradeoffs between all these magnitudes, establishing quantitative bounds for either the distortion power [8], [26] or the mutual information between private inputs and noisy outputs [15].

BOX III: LATTICE CRYPTOGRAPHY AND FULLY HOMOMORPHIC ENCRYPTION (FHE)

Lattice cryptography [27] was born as a great promise for post-quantum cryptography, as the security of lattice-based constructions rely on worst-case assumptions, unlike "traditional" cryptography; but the turning point when lattices started to be a really appealing cryptographic concept can be set at the time Gentry presented his seminal work [11], proving the existence of a fully homomorphic cryptosystem with semantic security; it was based on ideal lattices. Many similarities can be found between the use of lattices in signal processing for source and channel coding and lattice cryptography; in fact, encrypting a value is essentially encoding it, with the peculiarities that the channel noise is induced on purpose, and the used lattice description is not the same for coding as for decoding (public-private key).

We will explain here the GGH-family of lattice cryptosystems (named after Goldreich, Goldwasser and Halevi), as it is based on simple concepts while being highly powerful and efficient for building fully homomorphic schemes. Choose a lattice $\mathcal{L}(\boldsymbol{B})$ generated by a basis formed by the columns of a matrix $\boldsymbol{B}$. The encryption of a value m in a GGH cryptosystem is a two-step encoding: first, a noise vector $\boldsymbol{n}[m]$ that encodes the message m and fits inside the Voronoi cell V of $\mathcal{L}(\boldsymbol{B})$ is generated; then, a random centroid $\boldsymbol{v}$ in the lattice is chosen; the encryption of m is produced by adding the noise to the centroid: $\boldsymbol{c} = E[m] = \boldsymbol{v} + \boldsymbol{n}[m]$. Decryption is also two-step: the error correcting capabilities of the lattice allow to obtain the centroid $\boldsymbol{v}$ ($\boldsymbol{v} = \boldsymbol{B}\lceil \boldsymbol{B}^{-1}\boldsymbol{c} \rfloor$) and subtract it from $\boldsymbol{c}$ to get the noise $\boldsymbol{n}[m]$, further decoded to recover the original message m. Normally, $\boldsymbol{n}[m]$ is a *structured* noise, chosen as a bounded nested lattice inside the Voronoi region V with nesting factor equal to the plaintext cardinality.

The secrecy of the encoding-decoding process comes from the design of the basis $\boldsymbol{B}$: a) the vectors of the lattice are chosen with random components, so that $\mathcal{L}$ is a random lattice, and b) its dimensionality is much higher than for the lattices usually employed in coding: several thousands or tens of thousands. The second fact makes that the problem of finding the shortest vector of the lattice (SVP, roughly equivalent to finding the lattice centroid closest to a given vector $\boldsymbol{c}$ through lattice reduction algorithms like LLL, Lenstra-Lenstra-Lovász), be computationally infeasible if the available basis is not a "good basis"; additionally, the two facts together (random basis and extremely high dimensionality) make that the Voronoi region defined by $\boldsymbol{B}$ be nearly hyperspherical, as the columns of $\boldsymbol{B}$ will be, with high probability, almost orthogonal (a "good basis"). Hence, for decryption to be infeasible it is necessary to find a "bad basis", that is normally chosen as the Hermite Normal Form $\boldsymbol{H} = HNF(\boldsymbol{B})$; this "bad basis" represents the public key of the cryptosystem, and the random vector $\boldsymbol{v}$ of an encryption is chosen such that $\boldsymbol{c} = \boldsymbol{v} + \boldsymbol{n}(m) = \boldsymbol{n}(m) \mod \boldsymbol{H}$; respectively, the "good basis" $\boldsymbol{B}$ is the secret key that allows for decryption.

This construction may have homomorphic properties: the encryption operation $\boldsymbol{v} + \boldsymbol{n}[m]$ is linear, and due to the linearity of the lattice (both the "coarse" lattice generated by $\boldsymbol{B}$ and the "fine" lattice given by the structured noise), addition of two lattice points ($\boldsymbol{v}_1$ and $\boldsymbol{v}_2$) will yield another lattice point ($\boldsymbol{v}_3$):

$$\boldsymbol{c}_1 + \boldsymbol{c}_2 = (\boldsymbol{v}_1 + \boldsymbol{n}[m_1]) + (\boldsymbol{v}_2 + \boldsymbol{n}[m_2]) = \boldsymbol{v}_3 + \boldsymbol{n}_s[m_1 + m_2].$$

With this construction, the cryptosystem presents an additive homomorphism; but this is only true whenever the resulting noise vector $\boldsymbol{n}_s[m_1] + \boldsymbol{n}_S[m_2]$ still lies inside V. Whenever two ciphertexts are added, the noise vectors also get added in the resulting ciphertext; if the added noise falls outside the Voronoi region of the

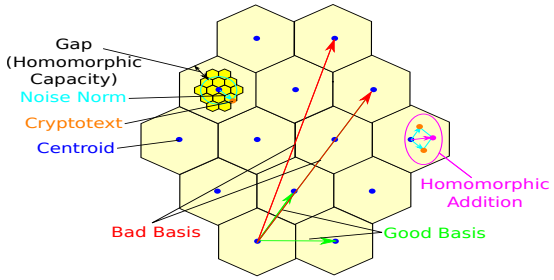


Fig. 5. Graphical 2D example of a lattice construction for a GGH cryptosystem.

lattice, then the error cannot be corrected anymore, and decryption may be incorrect. Consequently, in order to achieve a determined homomorphic capacity, there must be a *gap* between the noise norm ($||\boldsymbol{n}_s||$) of a new "fresh" encryption and the radius of the Voronoi region V (Fig. 5). When the product between two lattice points is also defined (e.g., when a convolutional modular lattice is used, and $\boldsymbol{B}$ is generated as a circulant matrix), the cryptosystem presents an algebraic homomorphism, limited by the *gap* (it is *somewhat homomorphic*). Gentry solved this limitation by reducing the order of the decryption circuit (*squashing*) and executing it homomorphically, in what he called a *bootstrapping* operation of a ciphertext; this "resets" the noise of an already operated ciphertext to a "fresh" encryption, thus achieving a *full homomorphism*. This step needs encryptions of "fragments" of the secret key so that anyone without that key can homomorphically execute the squashed decryption circuit.

The main drawbacks of current lattice-based fully homomorphic cryptosystems are the large size of both encryptions and keys, and the high computational complexity of the bootstrapping operation. These limitations have motivated recent advances either towards a) leveled fully homomorphic cryptosystems that do not need bootstrapping to execute a polynomial of bounded degree [28], and also towards b) reducing the cipher expansion [17] (ratio between the cipher size and the plaintext size).

BOX IV: DATA PACKING STRATEGIES; FILLING THE MODULUS GAP

The most widely used homomorphic cryptosystems (Paillier family) need a large modulus n for the hardness security assumptions to hold. This leaves a huge plaintext size of thousands of bits to encrypt typically small signals that can be represented in fixed-point arithmetic with tens of bits. Hence, the actual cipher expansion is much bigger for typical signals, and all that extra room is left unused. When the involved signals are bounded as $|\hat{x}_i| \leq 2^{n_b-1}$ at every stage of the protocol, it is possible to take advantage of this extra space that the cipher offers: a vector of $K = \lfloor \frac{\log_2 n}{n_b} \rfloor$ components can be packed into only one encryption as [29] (Fig. 6)

$$E\left[\hat{\boldsymbol{x}}_{\text{packed}}\right] = E\left[\sum_{i=0}^{K-1} (\hat{x}_i + 2^{n_b-1}) \cdot 2^{i \cdot n_b}\right],$$

being 2^{n_b-1} a shift factor for considering only positive numbers[4].

As with any parallelization strategy, encrypted vector packing needs a computation overhead for performing the packing and unpacking operations, and it may exist an optimum vector size for packing, where the complexity

[4]The shift factor fixes the sign convention between the bit representation ($-a \equiv 2^{n_b} - a$) and the modular arithmetic ($-a \equiv n - a$), working always in the range $[0, 2^{n_b})$, and avoiding errors in the conversion between both representations.

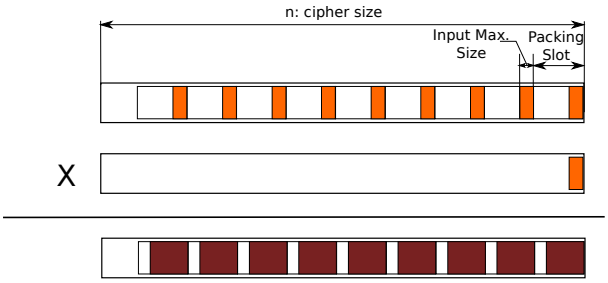


Fig. 6. Graphical representation of vector scalar product with packed data in a cipher.

reduction of the vectorial operations performed as scalar ones gets compensated by the complexity increase of the additional (un)packing protocols [29]. In fact, the packing formulation with powers of two can be generalized to an arbitrary base [30], but there are unpacking protocols that are optimized to work with binary decompositions of the encrypted values and impose no overhead only when the used base is a power of two.

There are applications we have shown, like filtering (either adaptive filtering [7] or frequency-domain filtering [14]), that can be easily parallelized when working in a block-by-block basis. The Block Least Mean Squares algorithm in [7] is an example where packing allows for a secure parallel block implementation. The computational and communication complexity of the whole protocol gets reduced by a factor of roughly K. This is possible because packing can be performed on the clear, with a negligible complexity compared to modular operations working with encryptions. Additionally, the latter get effectively reduced, with a packing strategy, by the same amount as the number of packed elements.

REFERENCES

[1] Wenwu Zhu, Chong Luo, Jianfeng Wang, and Shipeng Li, "Multimedia Cloud Computing," *Signal Processing Magazine, IEEE*, vol. 28, no. 3, pp. 59 –69, may 2011.

[2] M. Jensen, J.O. Schwenk, N. Gruschka, and L.L. Iacono, "On technical security issues in Cloud Computing," in *IEEE International Conference on Cloud Computing*, Bangalore, India, Sept. 2009, pp. 109–116.

[3] J.R. Troncoso-Pastoriza and F. Pérez-González, "CryptoDSPs for Cloud Privacy," in *CISE 2010*, Hong Kong, China, December 2010, vol. 6724 of *LNCS*.

[4] M. Armbrust, A. Fox, R. Griffith, A.D. Joseph, R.H. Katz, A. Konwinski, G. Lee, D.A. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "Above the clouds: A berkeley view of cloud computing," Tech. Rep. UCB/EECS-2009-28, EECS Department, Univ. of California, Berkeley, Feb 2009.

[5] Margarita Osadchy, Benny Pinkas, Ayman Jarrous, and Boaz Moskovich, "Scifi - a system for secure face identification," *Security and Privacy, IEEE Symposium on*, vol. 0, pp. 239–254, 2010.

[6] J.R. Troncoso-Pastoriza, S. Katzenbeisser, and M. Celik, "Privacy preserving error resilient DNA searching through oblivious automata," in *14th ACM CCS*, Alexandria, Virginia, USA, 2007, pp. 519–528.

[7] J.R. Troncoso-Pastoriza and F. Pérez-González, "Secure Adaptive Filtering," *IEEE Trans. on Information Forensics and Security*, vol. 6, no. 2, pp. 469–485, June 2011.

[8] Cynthia Dwork, "Differential privacy," in *Automata, Languages and Programming*, vol. 4052 of *LNCS*, pp. 1–12. Springer, 2006.

[9] R.L. Lagendijk, Z. Erkin, and M. Barni, "Encrypted signal processing for privacy protection," *IEEE Signal Processing Magazine*, 2013, to appear.

[10] P. Paillier, "Public-key cryptosystems based on composite degree residuosity classes," in *Advances in Cryptology - EUROCRYPT 1999*. 1999, vol. 1592 of *LNCS*, pp. 223–238, Springer.

[11] C. Gentry, "Fully homomorphic encryption using ideal lattices," in *ACM STOC'09*, Bethesda, MD, USA, May-June 2009, pp. 169–178.

[12] V. Kolesnikov and T. Schneider, "Improved garbled circuit: Free XOR gates and applications," in *ICALP'08*. 2008, vol. 5126 of *LNCS*, pp. 486–498, Springer.

[13] D.E. Bakken, R. Parameswaran, D.M. Blough, A.A. Franz, and T.J. Palmer, "Data obfuscation: Anonymity and desensitization of usable data sets," *IEEE Security and Privacy*, vol. 2, no. 6, pp. 34–41, 2004.

[14] T. Bianchi, A. Piva, and M. Barni, "On the Implementation of the Discrete Fourier Transform in the Encrypted Domain," *IEEE Transactions on Information Forensics and Security*, vol. 4, no. 1, pp. 86–97, 2009.

[15] M.S. Alvim, M.E. Andrés, K. Chatzikokolakis, P. Degano, and C. Palamidessi, "Differential privacy: on the trade-off between utility and information leakage," *CoRR*, vol. abs/1103.5188, 2011.

[16] A. Ghosh, T. Roughgarden, and M. Sundararajan, "Universally utility-maximizing privacy mechanisms," in *ACM STOC '09*. 2009, pp. 351–360, ACM.

[17] J.R. Troncoso-Pastoriza and F. Pérez-González, "Fully Homomorphic Faces," in *Accepted to IEEE ICIP 2012*, 2012, Online: http://www.gts.tsc.uvigo.es/~troncoso/troncoso-accepted_icip2012.pdf.

[18] Fida Kamal Dankar and Khaled El Emam, "The application of differential privacy to health data," in *Proceedings of the 2012 Joint EDBT/ICDT Workshops*, New York, NY, USA, 2012, EDBT-ICDT '12, pp. 158–166, ACM.

[19] Caroline Uhler, Aleksandra B. Slavkovic, and Stephen E. Fienberg, "Privacy-preserving data sharing for genome-wide association studies," *CoRR*, vol. abs/1205.0739, 2012.

[20] A. López-Alt, E. Tromer, and V. Vaikuntanathan, "On-the-Fly Multiparty Computation on the Cloud via Multikey Fully Homomorphic Encryption," in *STOC'12*, 2012.

[21] E.-O. Blass, R. Di Pietro, R. Molva, and M. Önen, "PRISM - privacy-preserving search in mapreduce," in *PETS'12*, Vigo, Spain, July 2012, vol. 7384 of *LNCS*, pp. 180–200, Springer.

[22] M. Franz, S. Katzenbeisser, S. Jha, K. Hamacher, H. Schroeder, and B. Deiseroth, "Secure computations on non-integer values," in *IEEE WIFS'10*, Seattle, USA, December 2010, IEEE.

[23] I. Dinur and K. Nissim, "Revealing information while preserving privacy," in *ACM PODS'03*. 2003, pp. 202–210, ACM.

[24] G. Smith, "On the foundations of quantitative information flow," in *FOSSACS'09*. 2009, pp. 288–302, Springer.

[25] K. Chatzikokolakis, C. Palamidessi, and P. Panangaden, "On the bayes risk in information-hiding protocols," *J. Comput. Secur.*, vol. 16, no. 5, pp. 531–571, Dec. 2008.

[26] Anindya De, "Lower bounds in differential privacy," *CoRR*, vol. abs/1107.2183, 2011.

[27] Daniele Micciancio and Oded Regev, *Post-quantum Cryptography*, chapter Lattice-based Cryptography, Springer, 2008.

[28] Z. Brakerski, C. Gentry, and V. Vaikuntanathan, "(leveled) fully homomorphic encryption without bootstrapping," in *ITCS*, 2012, pp. 309–325.

[29] J.R. Troncoso-Pastoriza, S. Katzenbeisser, M. Celik, and A. Lemma, "A secure multidimensional point inclusion protocol," in *ACM MMSEC'07*, Dallas, TX, USA, September 2007, pp. 109–120.

[30] T.Bianchi, A.Piva, and M.Barni, "Composite Signal Representation for Fast and Storage-Efficient Processing of Encrypted Signals," *IEEE Trans. on Information Forensics and Security*, vol. 5, no. 1, pp. 180–187, March 2010.