

# We are IntechOpen, the world's leading publisher of Open Access books Built by scientists, for scientists

4,800

Open access books available

122,000

International authors and editors

135M

Downloads

Our authors are among the

154

Countries delivered to

TOP 1%

most cited scientists

12.2%

Contributors from top 500 universities



WEB OF SCIENCE™

Selection of our books indexed in the Book Citation Index  
in Web of Science™ Core Collection (BKCI)

Interested in publishing with us?  
Contact [book.department@intechopen.com](mailto:book.department@intechopen.com)

Numbers displayed above are based on latest data collected.  
For more information visit [www.intechopen.com](http://www.intechopen.com)



# Numerical Problem Encryption for High-Performance Computing Applications

Riccardo Bernardini

## Abstract

Recent years witnessed the diffusion of cloud-based services. Cloud services have the interesting advantage that they can provide resources (CPU, disk space, etc.) that would be too expensive to deploy and maintain in-house. A major drawback of cloud-based services is the problem of handling private data and—possibly—intellectual property to a third party. With some service (e.g., data storage), cryptography can provide a solution; however, there are some services that are more difficult to protect. An example of such services is the renting of CPU to carry out numerical computation such as differential equation solving. In this chapter, we discuss the problem of encrypting *numerical problems* so that their solution can be safely outsourced. The idea is to transform (*encrypt*) a given numerical problem into a different one whose solution can be mapped back to the solution of the original problem if the *key* used at the encryption stage is known.

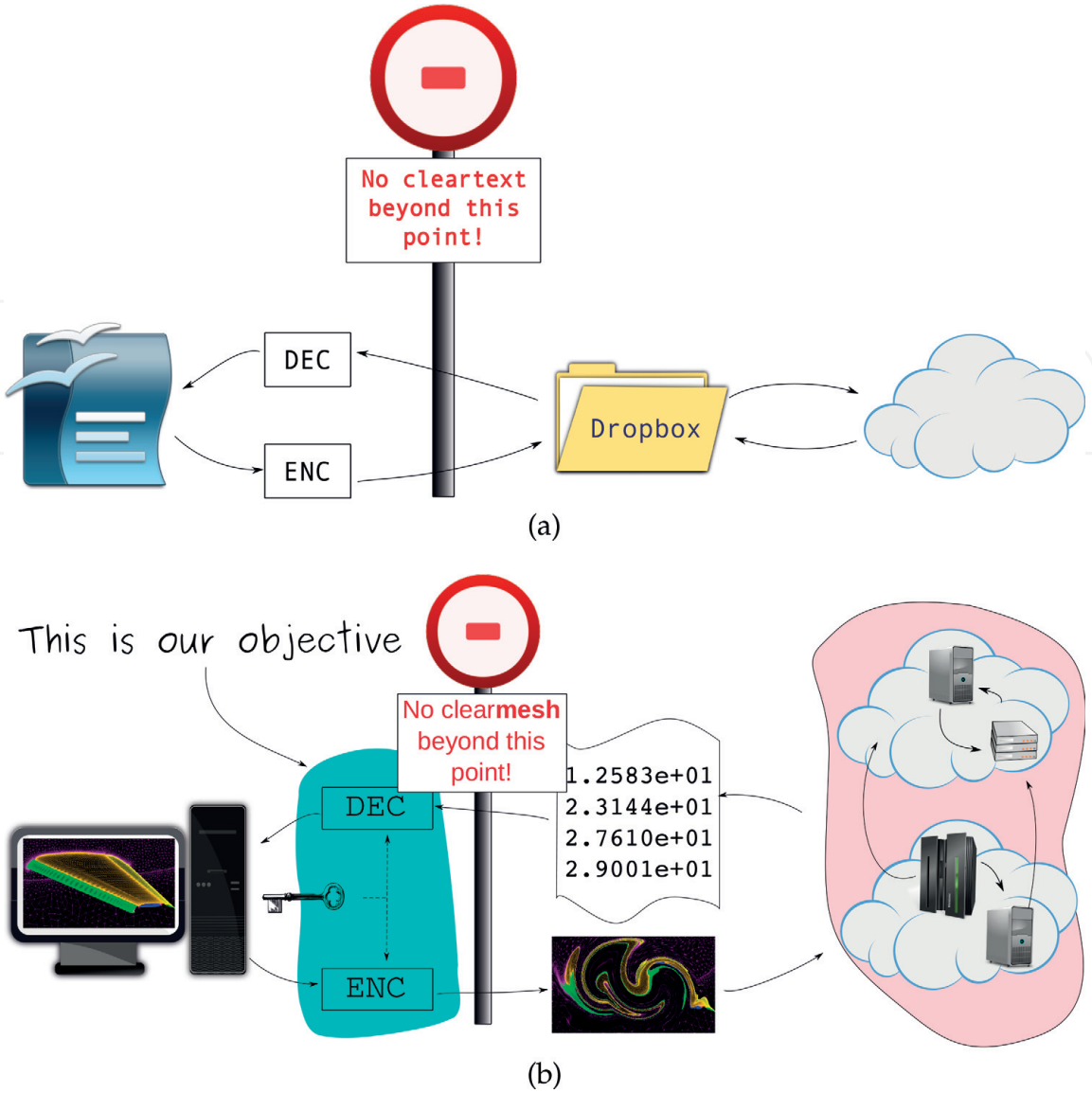
**Keywords:** HPC, numerical analysis, security, cloud

## 1. Introduction

The rise of *cloud computing* has made it possible for SMEs to procure, on a pay-per-use basis, resources that until few years ago they had to acquire themselves. Although cloud computing offers interesting opportunities, it has some drawbacks too. One important drawback is the lack of *data privacy*: as soon as the SME hands its data to the cloud provider, there is the risk that the data could be exposed to third parties. Privacy protection in cloud services is indeed one of the key challenges highlighted by the *Public Consultation on Cloud Computing and Software* [1].

In some cases, the SME can take some simple countermeasures to mitigate privacy risks. For example, the SME can send to the provider an encrypted version of the data in the case of storage service (see **Figure 1a**). This is possible since the cloud provider can store the data just as a “binary blob” without the need to understand them.

However, there are some services where simple solutions are not feasible, for example, the procurement of high-performance computing (HPC) resources on the cloud. It is a fact that many potential users are not eager to employ cloud HPC services because of the risk of data disclosure. In fact, simple data encryption mechanisms are not sufficient to deal with security and privacy protection in data centric environments, and even “the right to be forgotten” does not cover indirect security and privacy aspects [2].



**Figure 1.**  
(a) Using safely storage on networks and (b) renting safely CPU on network.

Ideally, we would like a solution reminiscent of what is done for storage: encrypt the computation before sending it to the cloud and decrypt what we receive from the cloud. **Figure 1b** shows a graphical representation of this idea: on the left hand side, we see a user that needs to compute the airflow around a new wing in an aerodynamic application. In order to outsource that aerodynamic problem, the user processes it with a secret key in the block ENC with the aim of transforming it into a *different numerical* problem that is uploaded to the cloud. A number of HPC providers collaborating on the cloud may be needed to solve this computation-intensive problem. The encrypted solution is then synthesized and sent back to the user (wing designer) that processes it with DEC in order to get the final answer. Standard cryptography techniques (e.g., RSA, AES, etc.) in this context provide only limited result. The question is if the user uploads their data in encrypted form to the cloud, how can the cloud process it?

At a first glance, this seems an unsolvable problem, but recent developments in the field of theoretical cryptography can address this apparent paradox using a set of secure techniques belonging to the family of secure multiparty computation (SMPC). Some efficient SMPC solutions, such as additive homomorphic encryption or garbled circuit, allow evaluating particular functions on encrypted data, but unfortunately they require interaction with the user, often making the cooperation

more expensive than the direct solution. On the other hand, fully homomorphic encryption (FHE) allows evaluating functions on encrypted binary data, without decrypting it nor interacting with the client. This implies that the cloud can compute any function of the data uploaded by the user (without learning anything about the plain-text data) and return the encrypted answer.

Even more recent developments in the field of cryptography show that even cryptographic obfuscation can be achieved (under strong but plausible assumptions): using obfuscation, one can upload a piece of software to the cloud, which now can be run on any input data without learning anything about the proprietary code—think of a researcher who find a new algorithm to diagnose some diseases. Now the researcher can sell diagnosis as an external service on a cloud provider, without fearing that the cloud can steal or leak his proprietary methods.

While SMPC and FHE provide wonderful theoretical results, they will have little or no practical impact in the way users work with the cloud. The computational overhead introduced by FHE is huge, and an obfuscated version of even a simple function (a point function with few bit inputs) requires gigabytes of memory.

## 2. The setup

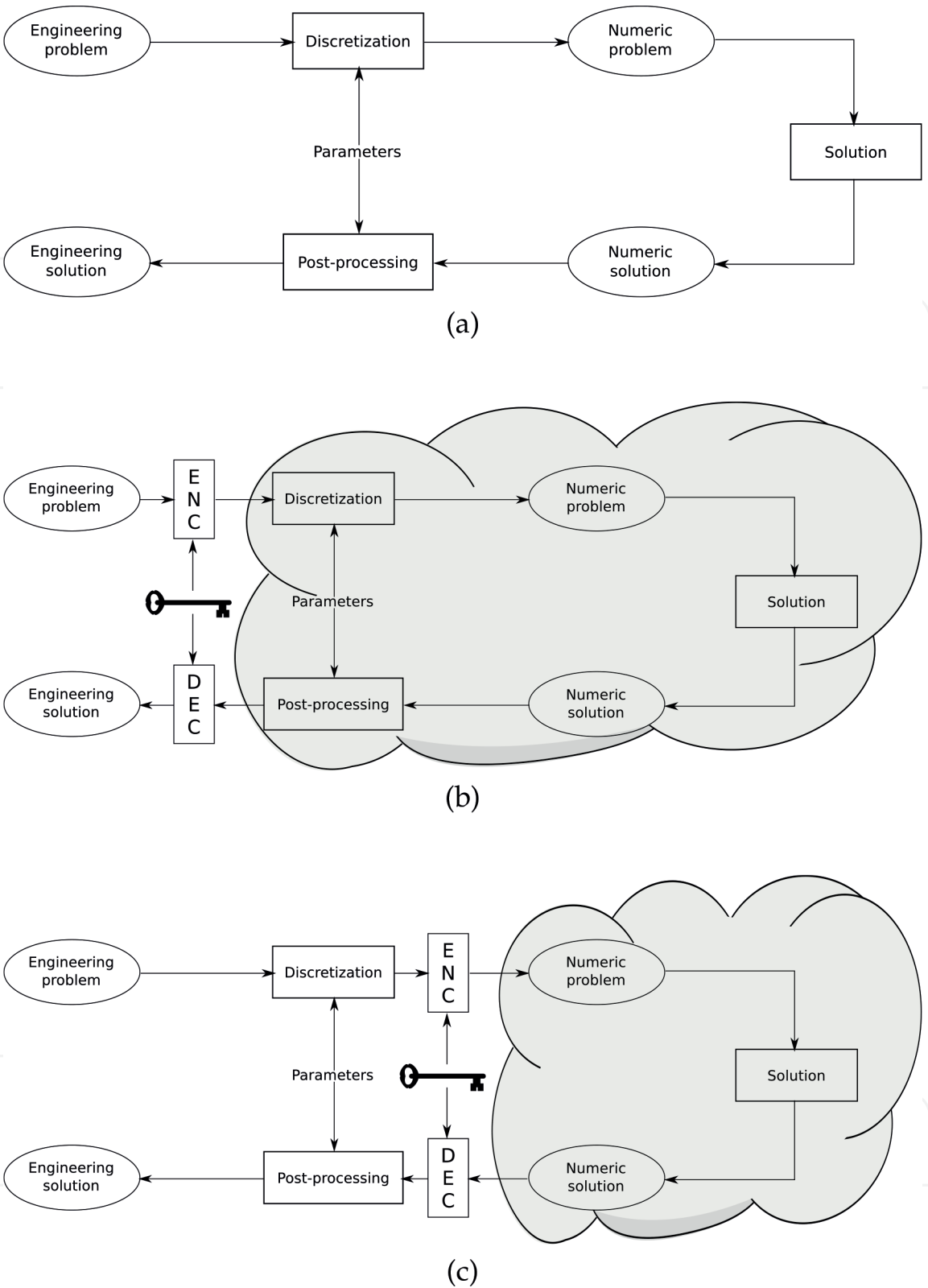
### 2.1 The basic setup

**Figure 2** shows a more detailed setup with two possible scenarios. We have, with reference to **Figure 2a**, an original problem that we will call the *engineering problem*; for example, determine drag and lift of a new innovative wing profile or the electronic configuration of a new molecule. The first step in the solution of the engineering problem is its transformation into a *numerical problem* (typically a linear system or an eigenvalue problem) that is solved by means of a well-known numerical algorithm in order to obtain a *numerical solution*. The final step is to use the numerical solution to obtain the *engineering solution*, that is, the values that are of interest for us (e.g., drag and lift in the wing problem). It is worth to discuss in some details the three steps.

#### 2.1.1 From the engineering problem to the numerical problem

Many engineering problems require the solution of differential equations that when discretized give rise, usually, to a linear system or an eigenvalue problem. The matrices obtained with the discretization can be quite large but (usually) sparse. Note that instead the engineering problem has usually a relatively compact description; in the example at hand, it would be a description of the geometry of the wing (by means of 3D model format) together with a description of the air flow [3–5]; in the case of the molecule, it could be a description of the configuration of the molecule (e.g., by means of its structural formula).

Although intuitively one can imagine that the solution of the numerical problem is the heaviest part, even the discretization step can be nontrivial. For example, some discretization technique—e.g., the popular finite element method (FEM)—requires to partition the space using a grid whose generation can be fairly complex [6]. Finally, it is worth observing that with large problems, not only the required CPU time can be a problem but also the amount of memory required that can become readily prohibitive because of the “dimension curse.” Indeed many engineering problems involve at least four dimensions: one for time and three for space. This suggests that working with very sparse matrices (and preserving their sparse



**Figure 2.**  
(a) Typical scenario: an engineering problem is converted to a numeric problem that is solved to derive the desired engineering solution, (b) outsourcing by encrypting the engineering problem and (c) outsourcing by encrypting the numerical problem.

structure) is of utmost importance. This will be important in the following when talking about some proposals found in the literature.

Finally, it is worth observing that there is a good degree of arbitrary in the discretization step: if a grid is used, it is not uniquely determined; in finite difference method (FDM), the ordering used to map the grid points to matrix coordinates is arbitrary; in methods based on function space discretization (e.g., Galerkin-like



methods), there is a wide freedom in the choice of basis functions. This implies that while it is known how to go from an engineering problem to its numerical counterpart, it is not clear how (or “if”) it is possible to go in the other direction. This is important in our context since we can expect that the opponent will be interested in the engineering problem rather than in the numerical one.

### 2.1.2 Numerical solution

In this step the numerical problem (typically, a linear system or an eigenvalue problem) is solved to give the numerical solution (typically, a vector). If the problem is very large, this step can be very CPU-consuming. Algorithms used for the solution typically exploit the very sparse structure of the matrices involved. For example, in the solution of a linear system, usually an iterative procedure is preferred, since the iterative procedure exploits more easily any sparsity (the cost of a product matrix-vector is proportional to the number of non-zero elements), while the inverse of a sparse matrix is not necessarily sparse, requiring much more memory and CPU time to be handled.

This suggests that if the encryption is applied at the numerical problem level (see Section 3 in the following), care must be exercised in order not to spoil the sparsity, although this goes against with the requirement—sometimes stated—of hiding the number of zeros (see Section 4.1.1).

### 2.1.3 From the numerical solution to the engineering solution

This step usually is not as computationally demanding as the previous ones. It amounts to extract from the numerical solution the values of interest (e.g., drag and lift in the wing example) or producing suitable visualizations of the data (e.g., in the electronic distribution of a molecule example).

## 3. Encrypting the problem

There are two possibilities, shown in **Figure 2**, of encrypting the problem: in a case (see **Figure 2b**) the original problem is directly encrypted and outsourced, leaving both discretization and numerical solution to the cloud; we will call this solution the *engineering problem encryption (EPE)*. In the other case (see in **Figure 2c**), the discretization step is done on the client computer, and only the numerical problem (e.g., solution of linear system) is encrypted and outsourced; we will call this solution the *numerical problem encryption (NPE)*. Pros and cons of the two solutions are as follows.

*EPE*: From the point of view of the work to be done on the client, this solution is preferable since the possibly large cost of the discretization step is outsourced to the cloud. This solution, however, requires a different encryption technique for every problem (e.g., an encryption technique suitable for aerodynamic problems cannot be used for computational chemistry problems). To the best of our knowledge, there is currently no proposal working at the EPE level.

*NPE*: On the one hand, this solution requires that the client does the discretization step and the computational cost of this step cannot be negligible; on the other hand, many engineering problems reduce to a just few numerical problems when discretized. This means that a procedure to encrypt, say, a linear system can be applied in many engineering problems that require the solution of differential equations.

Note that in both cases the key remains in the client, similarly to what happens in the case of remote storage. This implies that the client can employ very long keys, maybe as long as the data to be encrypted, since there is not a problem of distributing the key.

### 3.1 Numerical issues

Since we are interested in numerical problems, the issue of the impact of the encryption/decryption on the precision of the result needs to be taken into account. For example, consider the following protocol, similar to many techniques proposed for protecting a linear system [7–9]:

$$\mathbf{Ax} = \mathbf{b} \quad (1)$$

with  $\mathbf{A} \in \mathbb{R}^{n \times n}$  and  $\mathbf{x}, \mathbf{b} \in \mathbb{R}^n$ . System (1) is protected by randomly generating two permutation matrices  $\mathbf{P}, \mathbf{Q} \in \mathbb{R}^{n \times n}$  and a diagonal matrix  $\mathbf{D} \in \mathbb{R}^{n \times n}$  and replacing  $\mathbf{A}$  and  $\mathbf{b}$  with, respectively,

$$\hat{\mathbf{A}} = \mathbf{PAQD} \quad (2)$$

$$\hat{\mathbf{b}} = \mathbf{Pb} \quad (3)$$

and sending the problem

$$\hat{\mathbf{A}}\hat{\mathbf{x}} = \hat{\mathbf{b}} \quad (4)$$

to the cloud. It is immediate to check that the solutions of (1) and (4) are related by

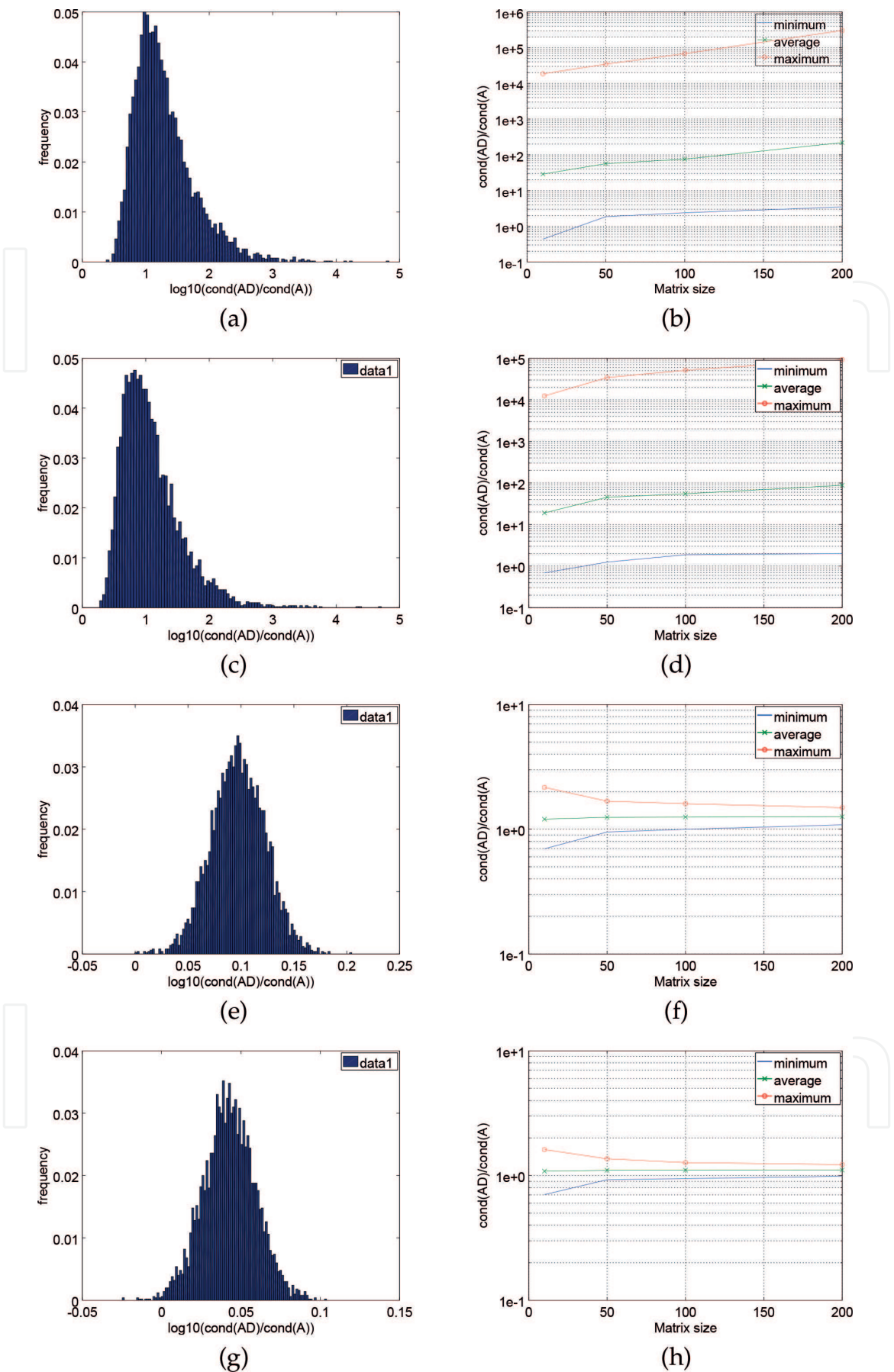
$$\mathbf{x} = \mathbf{QD}\hat{\mathbf{x}} \quad (5)$$

Eq. (5) can be considered as the decryption algorithm. This looks fine, but the value returned for  $\hat{\mathbf{x}}$  usually has some error  $\epsilon$ . It is immediate to verify that this induces an error  $\mathbf{QD}\epsilon$  on the decrypted value. If  $\mathbf{D}$  has some large entries, this will amplify the noise that affects the decrypted value. Moreover, right multiplication by  $\mathbf{D}$  can affect the condition number of  $\mathbf{A}$ , worsening the conditioning of the system. **Figure 3** shows the result of a numerical experiment demonstrating this problem: we generated 5000 symmetric matrices  $\mathbf{A}$  and 5000 diagonal matrices  $\mathbf{D}$  of sizes ranging from  $10 \times 10$  to  $200 \times 200$ , and for every iteration, we computed the condition number amplification

$$\rho = \text{cond}(\mathbf{AD})/\text{cond}(\mathbf{A}) \quad (6)$$

where  $\text{cond}(\mathbf{A}) = \|\mathbf{A}\|_2 \|\mathbf{A}^{-1}\|_2$  is the 2-norm condition number of  $\mathbf{A}$ . Entries of  $\mathbf{A}$  were Gaussian with zero mean and unit variance, while entries of  $\mathbf{D}$  have been generated using a variety of distributions: Gaussian as the entries of  $\mathbf{A}$ , uniform in  $[0, 1]$ , uniform in  $[1/2, 3/2]$ , and uniform in  $[1, 2]$ .

The right hand column of **Figure 3** shows the histogram of  $\rho$  in logarithmic scale for  $100 \times 100$  matrices; **Figure 3b** shows the minimum, maximum, and average values of  $\rho$  as function of the matrix size. Different rows of **Figure 3** are relative to different ways of generating  $\mathbf{D}$ . Although from **Figure 3** one can see that sometimes the conditioning can improve ( $\rho < 1$ ), it is clear that there is a non-negligible probability of worsening the condition number of several orders of magnitude, especially if the entries of  $\mathbf{D}$  can go near to zero (first two rows of **Figure 3** relative to the cases Gaussian and uniform in  $[0, 1]$ ).



**Figure 3.** (a) Histogram of the condition number amplification  $\rho = \text{cond}(\mathbf{AD})/\text{cond}(\mathbf{A})$  in logarithmic scale for  $100 \times 100$  matrices; matrix entries of both  $\mathbf{A}$  and  $\mathbf{D}$  are Gaussian with mean zero and unitary variance; (b) minimum, maximum, and average values of  $\rho$  as a function of the matrix size; (c) and (d) like (a) and (b), but the entries of  $\mathbf{D}$  are uniformly distributed in  $[0,1]$ ; (e) and (f) like (c) and (d), but the entries of  $\mathbf{D}$  are uniformly distributed in  $[1/2, 3/2]$ ; (g) and (h) like (c) and (d), but the entries of  $\mathbf{D}$  are uniformly distributed in  $[1,2]$ .



## 4. The adversary model

In the literature, two kinds of “bad behavior” can be present together or not in a specific opponent:

*Honest but curious:* In this case all the parties follow correctly the protocol, and the computed result is correct, but the cloud tries to learn the protected secret. Note that it is reasonable to expect that the opponent is interested in the *engineering problem* or the corresponding solution rather than in the numerical counterparts. The numerical problem/solution can be seen as a mean to find out the engineering problem/solution. Note that the freedom that one has in the discretization step can potentially be used to make it more difficult to invert it.

*Malicious:* The cloud does not follow correctly the protocol and, in particular, could try to “cut some corners” returning results that are not correct. In many cases, this attack can be easily counteracted by checking the result returned by the cloud; this is possible since many problems can be computationally heavy to solve, but it is fairly easy to check if a solution is correct; think, for example, the case of a linear system.

In some cases, even a full check can be quite demanding; in those cases, a random selection of checks to be done can give a fairly good assurance that the result is correct. For example, in the case of a linear system, a random subset of the equations can be checked. The probability of a false positive (an uncorrected result is accepted) decreases exponentially with the number of checks, while the computational cost grows only linearly. Indeed, many check techniques proposed in the literature can be reduced to this idea.

### Remark 4.1

It is worth observing that since we are talking about floating point computation, we need to take into account numerical noise that results from computation. For example, if a numerical system is solved with an iterative approach, the returned solution will differ from the “true” solution by a, hopefully, small error. This needs to be taken into account when checking if the solution is correct.

In a typical context the adversary is interested in getting the original problem or the solution. It is worth observing that while in the classical cryptography setup, the opponent recovers completely the message or nothing at all; in this case, there is the possibility that the adversary recovers an approximate version of the problem/solution. Observe also that the setup of **Figure 2c** makes the problem for the adversary more difficult since after recovering the numerical problem there is the problem of doing the inverse of discretization.

Finally, it is worth observing that the encryption/decryption steps could amplify the numerical noise introduced by the solution algorithm.

### 4.1 Security criterion

A major difference between traditional cryptography and cryptography of numerical problems is the information that an opponent can gain about the secret. In the most typical case, in a practical application of classical cryptography, two cases are possible: (i) the opponent succeeds in the attack and learns the whole secret and (ii) the attack is unsuccessful and very little, or nothing is learned about the secret; the case where a partial success can be achieved is quite uncommon. Suitable security criterion in this kind of application is information-theoretical criterion (where the adversary has unlimited computational power) or based on computational indistinguishability (where we admit that the computational power of the adversary can grow only polynomially).

In the context of encryption of numerical problems instead, it is possible that the opponent gains some partial or *low-resolution* information. For example, in the case

of a new wing, the adversary is interested in learning the profile of the wing; the adversary has a strong *a priori* information about the profile, and only some detail information to integrate such a priori are needed. Depending on the specific encryption technique, it could be possible for the adversary to find said details but only up to a resolution. The ambiguity could be “unsolvable” even with infinite computational power (more or less the equivalent of information-theoretical security), or maybe it could be that the amount of computation required to improve the resolution increases more than the polynomial with the required resolution (this would be the equivalent of classical computational indistinguishably).

There is also the possibility that the encryption technique leaks some *invariant* of the problem. For example, if linear system  $\mathbf{Ax} = \mathbf{b}$  is encrypted by applying two orthogonal matrices  $\mathbf{U}, \mathbf{V}$  as in

$$\underbrace{\mathbf{UAV}}_{\hat{\mathbf{A}}} \underbrace{\mathbf{V}^t}_{\hat{\mathbf{x}}} \mathbf{x} = \underbrace{\mathbf{Ub}}_{\hat{\mathbf{b}}}, \quad (7)$$

matrix  $\hat{\mathbf{A}}$  preserves the singular values of  $\mathbf{A}$ . Is this a problem? Most probably, it depends on the specific underlining engineering problem. As in another example, consider the encryption protocol described in (2). In this case,  $\hat{\mathbf{A}}$  preserves not the singular values of  $\mathbf{A}$  but its sparsity (i.e., the number of non-zero entries). Again, if this is a problem or not probably depends on the corresponding engineering problem.

To the best of our knowledge, most of the literature consider the classical cryptography criterion of computational indistinguishably, and more research about criterion specific for numerical problem could prove useful.

#### 4.1.1 About sparsity preservation

As said before, many discretization techniques produce matrices that are very sparse. This is very important from the viewpoint of computational efficiency since there are algorithms that are able to exploit the sparseness, working only with the non-zero entries of the matrix. This suggests that the encryption should preserve the sparseness or, at least, not reduce it in a significant way. However, some researchers raise the concern that even the fraction of non-zero entries can be private information. This would suggest that the encryption step should not preserve matrix sparsity.

It is also worth observing that in some application, the number of non-zero entries is known a priori with good precision. For example, in FDM/FEM discretization methods for differential equations, the fraction of non-zero entries depend on the structure of the grid employed which can be considered known with good precision. A quantitative and objective approach to the importance of preserving or hiding the sparsity can be an interesting field for future research activity.

## 5. Existing techniques

The field of encrypting numerical problems is relatively new, and there is not a huge variety of encryption algorithms proposed; many of them are just variations of some basic scheme. To the best of our knowledge, no algorithm found in the literature tries to encrypt the engineering problem; it rather tackles the numerical problem. Also, it is very difficult to find some discussion about the numerical stability of the proposed scheme.

## 5.1 Basic techniques

There are few basic techniques used in the literature to encrypt matrices and vectors.

*Addition:* To matrix  $\mathbf{A}$  a matrix  $\mathbf{K}$  is added to obtain  $\hat{\mathbf{A}} = \mathbf{A} + \mathbf{K}$ . This approach is not widely used, probably because sum does not “propagate” nicely in usual linear algebra operations (e.g., linear systems, eigenvalue problems, determinants, singular value decomposition). Moreover, in the case of linear systems, it is not obvious how to choose  $\mathbf{K}$  in order to have  $\hat{\mathbf{A}}$  invertible.

*Product by a diagonal matrix:* Matrix  $\mathbf{A}$  is multiplied (on the left, on the right, or both) by a diagonal matrix  $\mathbf{D}$ . The effect on problems like linear systems is quite easy to describe, and it suffices that all the diagonal entries are not zero in order to guarantee the invertibility of  $\hat{\mathbf{A}}$ , but, as shown in Section 3.1, if  $\mathbf{D}$  is randomly chosen, the condition number of  $\mathbf{A}$  can increase by several orders of magnitude.

*Product by a permutation:* Matrix  $\mathbf{A}$  is multiplied (on the left, on the right, or both) by a permutation matrix  $\mathbf{P}$ . This grants that  $\text{cond}(\hat{\mathbf{A}}) = \text{cond}(\mathbf{A})$  (therefore also invertibility is preserved). A product by a permutation requires no floating point multiplications, although the cost of data movement (especially if  $\mathbf{A}$  is very large) is not necessarily small. Sparsity is preserved, and this can be seen either as an advantage or as a drawback (see Section 4.1.1). Some characteristic values such as determinant and singular values are preserved.

*Product by a unitary matrix:* Matrix  $\mathbf{A}$  is multiplied (on the left, on the right, or both) by a matrix  $\mathbf{R}$  such that  $\mathbf{R}^t \mathbf{R} = \mathbf{I}$ . This grants that  $\text{cond}(\hat{\mathbf{A}}) = \text{cond}(\mathbf{A})$ .

Product by  $\mathbf{R}$  does not necessarily preserve sparsity. If  $\mathbf{R}$  is generated as a sequence of planar rotations, product by  $\mathbf{R}$  (stored as sequence of rotation, not as a full matrix) can be more efficient than usual matrix product. Singular values and determinant are preserved.

## 5.2 Brief literature review

Maybe the numerical problem most frequently addressed is the problem of solving linear systems. Lei et al. [7] proposed a scheme similar to (2) where  $\mathbf{A}$  is both left and right multiplied by a diagonal matrix and a permutation. The proposal of Wang et al. [10] is one of the few that proposes to use classical homomorphic encryption together with an iterative algorithm; the approach of Wang et al. requires to use matrices with integer entries (eventually by rescaling the system) and a continuous exchange between the cloud and the user. Chen et al. in [11] pointed out a weakness of [10] and proposed a variation to solve the problem. Another solution based on matrix pre- and post-multiplication is suggested in [8].

A problem similar to linear system is linear regression. Chen et al. in [9] used pre- and post-multiplication approaches with two diagonal matrices. Zhou et al. in [12] pointed out a possible weakness (in the hypothesis of an integer problem). Similar approaches, still based on matrix pre- and post-multiplication, can be found also for matrix product [13, 14], determinant computation [15], and singular value decomposition [16].

## 6. Conclusions

We analyzed the problem of outsourcing safely numerical and engineering problems to the cloud. It turns out that the field is still in an evolving phase. Many approaches are different instances of pre- and post-multiplication masking

techniques; security criterion is a reformulation of criterion from classical cryptography and does not address the peculiarities of numerical problem protections; finally, the problem of how the encryption/decryption impacts on issue such as numerical conditioning of the problem is usually not addressed. Future research directions will aim to cover such still open areas.

IntechOpen


IntechOpen

### **Author details**

Riccardo Bernardini  
University of Udine, Udine, Italy

\*Address all correspondence to: [riccardo.bernardin@uniud.it](mailto:riccardo.bernardin@uniud.it)

### **IntechOpen**

© 2019 The Author(s). Licensee IntechOpen. This chapter is distributed under the terms of the Creative Commons Attribution License (<http://creativecommons.org/licenses/by/3.0>), which permits unrestricted use, distribution, and reproduction in any medium, provided the original work is properly cited. 



## References

- [1] Griffin D. Report on the public consultation for H2020 Work Programme 2016–17: Cloud computing and software. Tech. Rep., DG CONNECT E.2. 2014
- [2] Jeffery K, Schubert L. Complete computing: Toward information, incentive and intention. Tech. rep., DG CONNECT E.2. 2014
- [3] Hsieh A. Direct numerical simulation of complex turbulence [PhD thesis]. University of Colorado; 2017
- [4] Alfonsi G. On direct numerical simulation of turbulent flows. *Applied Mechanics Reviews*. 2011;**64**:0802-0803
- [5] Landahl MT, Mollo-Christensen E. *Turbulence and Random Processes in Fluid Mechanics*. 2nd ed. Cambridge, UK: Cambridge University Press; 1992
- [6] Belytschko T, Rabczuk T, Huerta A, Fernández-Méndez S. *Meshfree Methods*, Ch. 10. Atlanta, GA, USA: American Cancer Society; 2004
- [7] Lei X, Liao X, Huang T, Li H, Hu C. Outsourcing large matrix inversion computation to a public cloud. *IEEE Transactions on Cloud Computing*. 2013;**1**(1):1
- [8] Kumar M, Meena J, and Vardhan M, Secure and efficient delegation of system of linear equation to a malicious cloud server. In: 2017 4th International Conference on Electronics and Communication Systems (ICECS); Feb 2017. pp. 12-17
- [9] Chen F, Xiang T, Lei X, Chen J. Highly efficient linear regression outsourcing to a cloud. *IEEE Transactions on Cloud Computing*. 2014;**2**:499-508
- [10] Wang C, Ren K, Wang J, Wang Q. Harnessing the cloud for securely outsourcing large-scale systems of linear equations. *IEEE Transactions on Parallel and Distributed Systems*. 2013;**24**: 1172-1181
- [11] Chen F, Xiang T, Yang Y. Privacy-preserving and verifiable protocols for scientific computation outsourcing to the cloud. *Journal of Parallel and Distributed Computing*. 2014;**74**(3): 2141-2151
- [12] Zhou L, Zhu Y, Choo K-KR. Efficiently and securely harnessing cloud to solve linear regression and other matrix operations. *Future Generation Computer Systems*. 2018;**81**: 404-413
- [13] Lei X, Liao X, Huang T, Heriniaina F. Achieving security, robust cheating resistance, and high-efficiency for outsourcing large matrix multiplication computation to a malicious cloud. *Information Sciences*. 2014;**280**:205-217
- [14] Kong S, Cai Y, Xue F, Yu H, Ditta A. Cloud outsourcing computing security protocol of matrix multiplication computation based on similarity transformation. *International Journal of Wireless and Mobile Computing*. 2018; **14**(1):90-96
- [15] Lei X, Liao X, Huang T, Li H. Cloud computing service: The case of large matrix determinant computation. *IEEE Transactions on Services Computing*. 2015;**8**:688-700
- [16] Pramkaew C, Ngamsuriyaroj S. Lightweight scheme of secure outsourcing svd of a large matrix on cloud. *Journal of Information Security and Applications*. 2018;**41**:92-102