The mathematical art of concentrating information

Stefano Leonesi

Published online: 8 August 2014
© Centro P.R.I.ST.EM, Università Commerciale Luigi Bocconi 2014

Abstract Every digital message sent runs the risk of being intercepted, so it is important that messages be encrypted to protect our privacy. The addition of a digital signature can help prove the authenticity of a message or document, but it could nevertheless be intercepted and altered. To prevent this, a message digest is created, forming a kind of fingerprint of the original message. Such fingerprints are mathematical functions that concentrate a message into a fixed-length representation. These are also useful in situations where information is to be kept secret until a later moment, but must be capable of being unmodified and authenticated when it is finally revealed. The aim of this paper is to show how cryptography, digital signatures and fingerprint functions may be linked with a famous curiosity of probability, the birthday paradox.

Keywords Public-key cryptography · Digital signatures · Bit-commitment · Hash function · One-way function · Fingerprint · Birthday paradox · Birthday attack · Square-root attack

1 Prediction and secrets

Imagine a situation where Magdalene wants to have a bet with Rudolph: she will correctly guess the subject of one of the essays of next state exam for high schools. However, she does not want to reveal it to him until the day of the exam, perhaps to keep the advantage to herself. So

Magdalene has to find a way to prove a posteriori to Rudolph that she was right and that she did not change her prediction along the way, after the publication of the subject. How is she to proceed, if she wants to commit herself to a prediction or a secret, without revealing it until a suitable time, but making it possible to check its correctness later on? Mathematics and computer science help us with this task.

2 Honey, I shrunk the messages

Magdalene might hit on the idea of committing to a solution with Rudolph by sending him, rather than the whole message m (her prediction), an abridged and condensed version containing only some fundamental features, say the number of letters in message m , or of vowels and consonants, of full stops and commas, the frequency of a specific letter, the 15th symbol and so on. In any case, she should share the rules with Rudolph. In this way, we get concise but essential key data about the message that do not give away its content but determine it with a very low probability of confusion. To understand better the situation, consider some lines from the fourth Canto of Dante's *Inferno*:

*Gran duol mi prese al cor quando lo'ntesi,
però che gente di molto valore
conobbi che'n quel limbo eran sospesi.
«Dimmi, maestro mio, dimmi, signore»,
comincia' io per voler esser certo
di quella fede che vince ogni errore:
«uscicci mai alcuno, o per suo merto
o per altrui, che poi fosse beato?».*

(I thought how many worthy souls there were/suspended in that Limbo, and a weight/closed on my heart for what

S. Leonesi (✉)
Department of Mathematics and Computer Science,
University of Camerino, Via Madonna delle Carceri, 9,
62032 Camerino, MC, Italy
e-mail: stefano.leonesi@unicam.it

the noblest suffer./“Instruct me, Master and most noble Sir,”/I prayed him then, “better to understand/the perfect creed that conquers every error:/has any, by his own or another’s merit,/gone ever from this place to blessedness?”—English translation by John Ciardi).

Assume we take as distinctive information the following (from the Italian text):

- number of characters (excluding spaces) = 246;
- number of words = 54;
- numbers of each vowel (ordered a, e, i, o, u) = 12, 34, 21, 28, 8;
- number of rs = 19;
- the 15th symbol (excluding spaces) = e .

The condensed version of Dante’s text, according to these criteria, becomes: 246, 54, 12, 34, 21, 28, 8, 19, e .

Such a synthesis of the message is called a *fingerprint function*, and represents a kind of DNA sequence that distinguishes concisely and almost uniquely the message itself.

If Magdalene, while betting on and committing to the “prediction message” m , had sent this sequence, Rudolph would have not any way to reconstruct its content. However, in the later phase of disclosure of the prediction, if Magdalene changed (even a little) the content of the original message to conform it to what would be better for her at that point, it is likely that the condensed version would vary too. So, when Rudolph finally gets from Magdalene an altered form of the whole message, he could find out about the alteration by computing on his own the fingerprint, which would unlikely coincide with the one he got in advance.

In practice, it would not be safe to use such trivial fingerprints, like the one in the example about Dante’s lines, since the text could be easily altered by Magdalene, by using just a little cunning and care. Consider a bet about the result of a football match, foreseen to end 3–1. If the actual score were 1–3, Magdalene should only exchange two digits in her message (or the two words spelling out them) before sending it to Rudolph, without modifying the fingerprint: this permutation does not change the total number of characters, of words nor the other data about the text, at least if 1 and 3 do not occur in the 15th position. In this way, Magdalene would have been able to adapt her wrong prediction to reality (and to her advantage), while Rudolph would not be able to detect anything, since the two messages, the original and the altered one, would have the same fingerprint: in this case we talk about a *collision* in the fingerprints.

3 From naïve attempts to serious stuff

Let us now show how behind these first, simple stratagems, which feel more like games than serious and useful

techniques, lie the theoretical principles of protocols and strategies that can possibly provide for extremely modern issues. We live in an era characterised by information freely circulating through computer networks; that information—money, messages, web pages—is codified in *bits* flowing in the labyrinth of the Internet. Those bits could fall prey to unsavoury people hoping to intercept, decrypt or alter them, perhaps to have money unlawfully paid to them using a stolen credit card number. In this regard, let us recall three principles of security of digital communication:

1. *secrecy*: information has to be kept secret from anybody but the intended addressee;
2. *data integrity*: we must ensure that the original data are not tampered with;
3. *authentication and checking of the data source*: the receiver must be able to check whether the data having A as their sender were actually sent by A , rather than by someone impersonating A .

We will just skim over the first point, strictly connected with cryptography (but see [5]). In fact, we shall assume that the reader has some familiarity with the general principles of public-key ciphering/deciphering (for more about this, we refer the reader to, for instance, to [1–3]). Here it will suffice to assume that E_A and D_A (and E_B, D_B) represent, respectively, the encoding function, which depends on the public key, and the inverse, private decoding one, used by A (by B) within a public key encryption system. Assume further that, in order to derive D_A (and D_B), it is necessary to know the private key of A (of B), who for this reason is careful to keep it absolutely secret. In short, given a message m , we have $E_A(D_A(m)) = m$ and $D_A(E_A(m)) = m$.

So, let us dwell on the last two points. As to data security, it suffices to translate and suitably adapt what we have previously seen in a naïve way, about predictions and commitments (see [6, 7]). It is necessary to have an impartial *Certificate Authority*, recognised by all, which prescribes and lets all users know the fingerprint function f to be used, usually of constant, not too large length, and such that it is unlikely for two different messages to take two equal values. In next paragraph we shall see this notion and the techniques involved in greater detail. The procedure is as follows. Let m be the message to be sent:

- (a) the phase of the *commitment* of the message by A amounts to sending B the fingerprint disguised using $E_B(D_A(f(m)))$, which can only be done by A , who is in possession of his own private key;
- (b) B is then able to *verify* the commitment, and hence the integrity of the data in m , as follows: using first using first his own private key and then A ’s public

one, he decodes the fingerprint $f(m)$ as $E_A(D_B(E_B(D_A(f(m)))))) = f(m)$; next, as soon as A reveals the complete message $E_B(m)$, encoded for security reasons, B decodes with his private key the message $m = D_B(E_B(m))$ and separately computes the fingerprint $f(m)$, according to the rules given by the certificate authority. If this agrees with the fingerprint $f(m)$ committed to in phase (a), then B may conclude that A 's message was not tampered with.

When the procedure takes place in the binary system, by transmitting bit strings, it is also referred to as a *bit-commitment*.

This technique is well suited to satisfy requirement 3 of a secure communication, that is, the possibility of checking the data source. Indeed, even in a digital world, we would like to be able to sign our email or web messages in such a way that no one could impersonate us. So, when A sends a message, he could sign it digitally by including his encoded signature $E_B D_A(A)$ in the message m he sends B : to this end, A includes and condenses his personal data and any other information making it possible to prove his identity (for more about digital signatures, see [4]).

As we have seen, such an action by A is possible and, moreover, quite reasonable, in order for him to be identified by B using his own private key and B 's public one. Indeed (omitting from now on the use of parentheses, for the sake of clarity), we have $E_A D_B E_B D_A(A) = A$. Unfortunately, such a strategy is guilty of naïveté: if the signature were only related to A 's identity (including, usually, his first name, surname, address and other data concerning him, and so quite easy to trace even within an encoded message), it could be found by an interloper, should he intercept some messages and check recurring sequences. The signature $E_B D_A(A)$ is sure to be among such sequences, and this would allow him to gather precious information about D_A , useful to decrypt the message, or even impersonate A effortlessly, by simply signing $E_B D_A(A)$, with dangerous, easy to imagine consequences.

Here is where the fingerprint functions help us: the idea is to sign every message in a way that also depends on its content and perhaps on the addressee too, in such a way that the result is always different and not liable to vulnerable patterns and repetitions. In practice, A signs the encoded message not using $E_B D_A(A)$, but using $E_B D_A(f(m))$, where f is a fingerprint function fixed in advance and distributed to users by the certification authority. The receiver B first decodes the fingerprint $f(m)$ as above: $f(m) = E_A D_B E_B D_A(f(m))$; when, later, he gets the whole encoded message $E_B(m)$, B decodes it by $m = D_B E_B(m)$ and computes independently its fingerprint: if the two fingerprints coincide, then B may assume that A is the

actual sender of m . Most importantly, now A has not let out unintended clues useful for undesired future violations.

4 Identikit of a digital fingerprint

Till now, we have hinted at some applications and trivial examples of fingerprint functions, without going into their theory and the properties they should have. This is a good moment to do so.

Formally, a *fingerprint* (or *hash*) is a function f that, given an input m of arbitrary length n (for instance, a finite binary sequence), yields a binary output $f(m)$ of a fixed, not too large length l . Furthermore, f must possess the following properties:

1. *efficiency*: f has to be computationally fast and easy;
2. *one-way*: f has to be one-way, in the sense that it has to be very hard and slow to reconstruct m when $f(m)$ is known;
3. *resistance*: given an input m , it has to be highly improbable that another input m' gives the same fingerprint, that is, that a collision $f(m) = f(m')$ occurs.

To clarify the situation by means of a metaphor, as a model of a fingerprint function we can picture the shattering of a crystal vase: it is very easy and quick to do, it is extremely improbable that it happen twice in the same way, and it is very difficult to reverse the process putting back together the pieces of the vase. But let us get back to the theoretical viewpoint.

About the first requirement, by “computationally fast and easy” we mean that there exist algorithms with an at most polynomial running time $t(n)$ (see [8]). More precisely, a function $t: N \rightarrow R^+$ is (at most) *polynomial*, and is denoted by $t(n) = O(n^k)$, for k non-negative integer, if

$$\lim_{n \rightarrow +\infty} \frac{t(n)}{n^k} < \infty.$$

Compare this with those “intrinsically hard” problems for which any algorithm solving them takes an *at least exponential* time as a function of the length n of the input: in more formal terms, this happens when the running time $t(n)$ of the algorithm to solve the problem satisfies

$$\lim_{n \rightarrow +\infty} \frac{t(n)}{a^n} = s$$

with $s \in R - \{0\}$ for some real $a > 1$, or $s = +\infty$ for every $a > 1$.

Now, the extreme “hardness” and “slowness” of computing m from the fingerprint $f(m)$ in the one-way requirement (property 2) can be described as an at least exponential running time, which means that it is in practice an infeasible and intractable computation.

Let us now discuss the last notion, *resistance*. In principle, it is clearly impossible to avoid collisions, at least when dealing with messages whose size n is greater than the fixed size l of the fingerprint f : in this case we have 2^n messages and 2^l possible fingerprints; hence there are necessarily multiple messages sharing a single fingerprint. The key point lies in making it difficult to generate or detect two messages giving rise to a collision. There are two main possibilities:

1. *weak resistance* to collisions: for a given m , and hence for a fixed fingerprint $f(m)$, it is computationally infeasible—i.e., an at least exponential time is required—to determine another m' such that $f(m) = f(m')$;
2. *strong resistance* to collisions: it is computationally infeasible to find a pair m, m' of messages such that $f(m) = f(m')$, regardless of the fingerprint.

The first possibility corresponds to the requirement to make it hard for anyone, given a message m , to construct another message m' whose fingerprint is identical to that of m : however, as we have seen, the existence of such an m' is certain when $n > l$. In this regard, we want even small changes to a message to give rise to large variations in the corresponding fingerprint. The ideal, safest situation is when, for random sequences of input binary messages, the corresponding fingerprints of length l behave as a random variable taking all integer values from 0 to $2^l - 1$ with uniform probability distribution. Under this hypothesis, the probability of finding a message having the same fingerprint is 2^{-l} , hence the probability of not being able to do so is $1 - 2^{-l}$ for any given attempt, and $(1 - 2^{-l})^k$ for k independent attempts. It follows that the probability of getting at least one success after k brute force attempts equals 1 minus the probability of getting none:

$$P(\text{at least one success in } k \text{ attempts}) = 1 - (1 - 2^{-l})^k.$$

By the binomial theorem it follows that

$$\begin{aligned} 1 - (1 - 2^{-l})^k &= 1 - \sum_{h=0}^k \binom{k}{h} (-2^{-l})^h \\ &= k \cdot 2^{-l} - \frac{k \cdot (k-1) \cdot 2^{-2l}}{2} \\ &\quad + \frac{k \cdot (k-1) \cdot (k-2) \cdot 2^{-3l}}{6} - \dots \\ &\quad + (-1)^{k+1} \cdot 2^{-kl}, \end{aligned}$$

which asymptotically equals $k \cdot 2^{-l}$, in the sense that $\lim_{n \rightarrow \infty} \frac{P}{k \cdot 2^{-l}} = 1$; we write this as $P \sim k \cdot 2^{-l}$.

This means that, for l large enough, we may ignore the terms after $k \cdot 2^{-l}$, since they are extremely small. In short, to obtain a reasonable, fixed probability p to crack a weak resistance, we need a number k of attempts in the order of

$p \cdot 2^l$, which is proportional to 2^l , and hence exponential as a function of the length l of the fingerprint: in conclusion, it is infeasible according to the standards of theoretical computer science.

5 The birthday paradox and devious digital fingerprints

If now we want it to be difficult to find any two messages m and m' giving the same fingerprint, independently of its value, we fall within the scope of strong resistance. Indeed, a dishonest sender might want to find in advance two messages with the same fingerprint, and hence with the same signature—think about contracts, bets, predictions—in order to be able to claim later that he sent the one most advantageous for him.

Let us discuss this issue, which surprisingly connects with an amusing question about birthdays and which has, as we shall show, a distinctly different computational complexity. So, let us compare the brute force used for weak resistance to the light-heartedness of a party, and let us ask how many guests are required in order to have more than an equal chance that at least two of them share the same birthday. The perhaps surprising and counterintuitive answer is that as few as 23 are enough, and is known as the *birthday paradox*. In fact, even if there are $N = 365$ possible distinct birthdays (let us ignore leap years), we have fixed no precise day: so the key factor is the number of all possible pairs of k guests, a much larger number than k . To be precise, the probability P of finding at least two matching birthdays equals 1 minus the probability that all birthdays are different. So a moment's thought or simple combinatorial techniques confirm that if $k = 23$:

$$\begin{aligned} P(\text{matching birthdays}) &= 1 - \frac{365 \cdot 364 \cdots (365 - 23 + 1)}{365^{23}} \\ &\approx 50.7\%, \end{aligned}$$

which can also be interpreted as:

$$P(\text{matching birthdays}) = 1 - \frac{P_{365,23}}{P_{365,23}^r},$$

where by $P_{n,k}$ and $P_{n,k}^r$ we denote the number of k -permutations of n elements without and with repetitions allowed, respectively.

The following graph shows how the probability of having matching birthdays grows quickly when the number of guests increases: with 50 guests, it is greater than 97 %! (Fig. 1).

But where is the connection between all of this and cryptography, hash functions and collisions? This paradox confirms and justifies intuitively an unexpected property: if

Fig. 1 The birthday paradox. Image: Rajkiran g/Wikimedia Commons

an element may randomly take N distinct values, it is likely (i.e., the probability is $>50\%$) that the first collision, that is, the situation where it takes again a previously taken value, already takes place after about $\sqrt{N}$ draws. The interested reader may find the details of the proof in “Appendix 1” below.

As an example, a dishonest correspondent B might use this fact to try to find in a feasible time two messages m, m' [the first a legitimate one and the second a fraudulent one, having the same fingerprint $f(m) = f(m')$ and hence the same signature] have A sign the legitimate one using $E_{B-D_A(f(m))}$, then substituting m' for m and claim that the m' was the one signed by A . In all of the above, he does not need A 's private key. Indeed, finding two matching binary fingerprints of length l requires a number of attempts proportional to $\sqrt{2^l} = 2^{l/2}$, which grows exponentially (and thus is not an effortless task), but quite more reasonable, as computational complexity, than something proportional to 2^l . To picture this, consider the case $l = 20$: while $2^{20} = 1,048,576$, the number 2^{10} is “just” 1,024.

The interested reader may find more details in “Appendix 2”.

6 A final look at current events

In order to prevent all these risks, the fingerprint functions customarily used in practice are carefully designed in such a way that even tiny changes in a message give rise to large distortions in the corresponding fingerprint, and that it is very difficult to intentionally generate collisions. Moreover, it is highly desirable for them to be quite fast and easy to compute, to keep the procedures light and the methods serviceable.

Without going into details and technicalities about fingerprint functions and hash algorithms, which would be

beyond the scope of these short remarks, we shall confine ourselves to the most usual general principles inspiring them:

- in general, the message is divided into blocks of the same length (if necessary, some bits can be added to pad out a block);
- each block is repeatedly compressed according to a fingerprint function resistant to collisions that has as among its inputs the result of the previous iteration, using a random value for the first iteration;
- the fingerprint functions generally include bitwise mappings, such as shifts, rotations, *and*, *or*, *xor* operators and their compositions.

Since 2002, National Institute of Standards and Technology (NIST), the US agency that fixes the standards for computer security, has recommended adopting several standard hash algorithms, called *SHA* for *Secure Hash Algorithm*: in particular, the “old” *SHA-1* with its 160-bit fingerprint, the more modern *SHA-256*, *SHA-384*, and *SHA-512*—all belonging to *SHA-2* family—structurally similar to *SHA-1*, but more resistant to collisions, due to their 256-, 384-, and 512-bit fingerprint output, respectively. In conclusion, let us recall the algorithm *Keccak* (pronounced *catch-ack*), devised by Guido Bertoni, Joan Daemen, and Gilles Van Assche of STMicroelectronics and Michaël Peeters of NXP Semiconductors, which was the winner of a 5-year long competition organised in 2007 by NIST; 64 proposals of hash algorithms took part in the competition. As the winning proposal, *Keccak* will become the *SHA-3* algorithm of NIST.

Appendix 1

As in the case of the weak resistance, assume we are dealing with fingerprint functions with binary output of length l that return, to random input values, any of the 2^l possible output values with the same probability. Rephrasing the birthday paradox, let us compute the probability P to obtain at least two equal fingerprints, that is, a collision, when randomly picking $0 < k \leq 2^l$ input messages of length n . Again, combinatorics allows us to find that all possible ordered sequences of k fingerprints of messages are $P_{2^l, k}^r = 2^{l \cdot k}$ and that those consisting of distinct fingerprints are

$$P_{2^l, k} = \frac{2^l!}{(2^l - k)!}.$$

It follows that

$$\begin{aligned}
P &= 1 - \frac{P_{2^{-l},k}}{P_{2^{-l},k}'} = 1 - \frac{2^l!}{(2^l - k)!} \cdot \frac{1}{2^{lk}} \\
&= 1 - \frac{2^l \cdot (2^l - 1) \cdot (2^l - 2) \cdots (2^l - k + 1)}{2^{lk}} \\
&= 1 - \left(1 - \frac{1}{2^l}\right) \cdot \left(1 - \frac{2}{2^l}\right) \cdots \left(1 - \frac{k-1}{2^l}\right) \\
&= 1 - (1 - 2^{-l}) \cdot (1 - 2 \cdot 2^{-l}) \cdots (1 - (k-1) \cdot 2^{-l}).
\end{aligned}$$

Let us now recall the MacLaurin series $e^{-x} = 1 - x + x^2/2 + o(x^2)$ with $o(x^2)$ approaching zero faster than x^2 . From $x^2/2 + o(x^2) \geq 0$ we obtain that $e^{-x} \geq 1 - x$ for x close enough to zero. So, if l is sufficiently large, the terms of the form $1 - h \cdot 2^{-l}$ for $h = 1, \dots, k-1$ in the last formula for P can play the role of x in MacLaurin series; hence we can write

$$\begin{aligned}
P &\leq 1 - e^{-2^{-l}} \cdot e^{-2 \cdot 2^{-l}} \cdots e^{-(k-1) \cdot 2^{-l}} \\
&= 1 - e^{-(1+2+\cdots+(k-1))2^{-l}} = 1 - e^{\frac{k(k-1)}{2} \cdot 2^{-l}} < 1 - e^{\frac{k^2}{2} \cdot 2^{-l}}.
\end{aligned}$$

In short, $P < 1 - e^{-\frac{k^2}{2} \cdot 2^{-l}}$; from this we may conclude that, in order to get a collision probability $P = 1/2$, the number of necessary attempts can be found from $1/2 < 1 - e^{-\frac{k^2}{2} \cdot 2^{-l}}$ and in particular we get $k > \sqrt{2 \cdot \ln 2} \cdot \sqrt{2^l} = \sqrt{2 \cdot \ln 2} \cdot 2^{l/2}$, that is, it is bounded by a function approximately equal to $\sqrt{2^l}$.

Indeed, coming back to the birthday problem, if we put 365 in the place of 2^l , we find $k > 22.49$.

Appendix 2

The so-called *birthday attack* or *square-root attack*, performed by a dishonest correspondent B who wants to find two messages with the same fingerprint, consists of the following steps:

1. B generates $2^{l/2}$ variations of the legitimate message m , obtained through small changes that do not significantly modify its content, so to be basically equivalent.
2. Next, B computes and records the fingerprint of each variation of m in a direct-access archive (a *hash table*): there are well-known search (*hashing*) algorithms, allowing one to access records of an archive in a direct way, using arithmetic operations transforming the fingerprints into the addresses of the records. In this way, accessing any of the fingerprints requires an approximately constant time.

3. Now B creates a second, fraudulent message m' , only slightly different from the legitimate one in its overall formal structure, but completely different in its meaning, conforming to different premises that B intends to unlawfully use to his advantage. For instance, he might modify or exchange the digits of some amount of money in a contract. This is the message he aims at substituting for the original one in order to swindle A . B computes the fingerprint and checks whether it is present in the archive, using for each attempt a constant time which does not substantially modify the running time:

- (a) if the fingerprint is in the archive, the untrustworthy B has found the desired collision, he can easily find out which variation of the legitimate message gave the same fingerprint, and this is the message he will send A .
- (b) Otherwise, step 3 is iterated.

By the birthday paradox, we may expect a collision to be found after as few as $2^{l/2}$ variations.

The long and the short of this is that in order to achieve a strong resistance, we have to ask for fingerprint functions twice as long as those giving a security level suitable for weak resistance, the ones needed to prevent brute-force detection of a collision for a fixed value of the fingerprint.

Translated from the Italian by Daniele A. Gewurz.

References

1. Baldoni, M.W., Ciliberto, C., Piacentini Cattaneo, G.M.: Elementary Number Theory, Cryptography and Codes. D.A. Gewurz (trans.) Berlin, Heidelberg, Springer (2009)
2. Ferragina, P., Luccio, F.: Crittografia. Principi, Algoritmi, Applicazioni. Bollati Boringhieri, Torino (2001)
3. Leonesi, S., Toffalori, C.: Numeri e Crittografia. Springer, Milan (2006)
4. Leonesi, S.: Una passeggiata nella crittografia applicata: le firme digitali. Lettera Matematica PRISTEM 73 (2009)
5. Menezes, A.J., Van Oorschot P.C., Vanstone, S.A.: Handbook of Applied Cryptography, CRC Press, p 816 (1997)
6. Schneier, B.: Secrets and Lies: Digital Security in a Networked World New York. Wiley, New York (2000)
7. Stallings, W.: Cryptography and Network Security: Principles and Practice, 5th ed. McGraw-Hill, New York (2010)
8. Toffalori, C., Corradini, F., Leonesi, S., Mancini, S.: Teoria della Computabilità e della Complessità. McGraw-Hill, Milan (2005)

Stefano Leonesi has a degree in Mathematics from the University of Camerino. He holds a Ph.D. in mathematical logic and theoretical computer science from the University of Siena and a Ph.D. in epistemology of computer science and social changes from the University of Teramo. Currently he teaches didactics and history of mathematics at the University of Camerino. He is also author of several books on computability and complexity, algebra and cryptography, and of books of popular science such as *Matematica, Miracoli e Paradossi* (Bruno Mondadori edi-

tor, 2007) and *L'Arte di Uccidere i Draghi* (Università Commerciale Bocconi), written with C. Toffalori. He is also interested in meteorology.