Efficient Remote Data Possession Checking in Critical Information Infrastructures

Francesc Sebé, Josep Domingo-Ferrer, *Senior Member*, *IEEE*, Antoni Martínez-Ballesté, Yves Deswarte, *Member*, *IEEE*, and Jean-Jacques Quisquater, *Member*, *IEEE*

Abstract—Checking data possession in networked information systems such as those related to critical infrastructures (power facilities, airports, data vaults, defense systems, and so forth) is a matter of crucial importance. Remote data possession checking protocols permit checking that a remote server can access an uncorrupted file in such a way that the verifier does not need to know beforehand the entire file that is being verified. Unfortunately, current protocols only allow a limited number of successive verifications or are impractical from the computational point of view. In this paper, we present a new remote data possession checking protocol such that 1) it allows an unlimited number of file integrity verifications and 2) its maximum running time can be chosen at set-up time and traded off against storage at the verifier.

Index Terms—Infrastructure protection, database management, management of computing and information systems.

1 Introduction

The protection of critical infrastructures (for example, airports, power plants, financial data vaults, hospitals, defense systems, industrial sensor networks, and so forth) is a priority for governments and companies. The development of the information society has caused a vast majority of such infrastructures to critically depend on the correct operation of the underlying information systems—that is why such information systems are often referred to as *critical information infrastructures.* Indeed, any service interruption, malfunction or, even worse, partial or total destruction of those information systems as a consequence of an accident or a terrorist attack can result in huge material or even human casualties. This reality supports the claim that one of the biggest challenges in infrastructure protection is the underlying information security problem: information systems should be dependable.

The following examples show how remote data possession checking is relevant to critical infrastructures:

- Data vaulting systems [14], [1], [9] are increasingly being used to store off-site copies or backups of critical data (for example, financial, government, or sensor data). If backup data stored in the data vault become corrupted without knowledge of the data owner (the data vault customer), no backup recovery is possible in case of loss of the primary critical data. Using a remote data possession checking protocol, the data vault customer might be able to periodically verify that the data vault provider is storing a *current* and complete copy or backup of the critical files. Any corruption will be noticed by the data owner who will be able to take immediate action (such as making another backup or using another data vault provider).
- Remote data possession checking is an important component of intrusion detection systems (IDS) used to detect server corruption. However, if the application is a different service than the backup itself and the server can be corrupted/malicious, a remote data possession checking protocol alone is not enough: the server could back up original files and access them to properly run the protocol while using the corrupted versions to provide the service. In the Dependable Intrusion Tolerance architecture (DIT, [13]), integrity check is just one among the various building blocks used to detect corruption of remote data. Several complementary mechanisms exist (including comparison of several server behaviors, runtime model checking, network-based and host-based intrusion detection, and so forth), so that it should be extremely difficult for an attacker to cause a security failure.

Note 1. Conventional integrity checks (for example, cyclic redundancy checks (CRCs), checksums [15]) are useful to detect accidental integrity loss but not malicious integrity attacks. To illustrate this point, consider a CRC with polynomial P of degree d with binary coefficients used to check the integrity of files stored in an off-site data vault: given an original file m of length $|m| \gg d$, it is easy to construct $m' \neq m$ such that both m' and m yield

- *F. Sebé, J. Domingo-Ferrer, and A. Martínez-Ballesté are with the Department of Computer Engineering and Maths, UNESCO Chair in Data Privacy, Rovira i Virgili University, Av. Països Catalans 26, E-43007 Tarragona, Catalonia.*
  E-mail: {francesc.sebe, josep.domingo, antoni.martinez}@urv.cat.
- *Y. Deswarte is with LAAS-CNRS, Université de Toulouse, 7 Avenue du Colonel Roche, F-31077 Toulouse Cedex 4, France.*
  E-mail: deswarte@laas.fr.
- *J.-J. Quisquater is with the Laboratory for Microelectronics, Department of Electrical Engineering, Université Catholique de Louvain, Place du Levant 3, B-1348 Louvain-la-Neuve, Belgium.*
  E-mail: quisquater@dice.ucl.ac.be.

Manuscript received 29 Apr. 2006; revised 5 July 2007; accepted 17 July 2007; published online 6 Aug. 2007.
For information on obtaining reprints of this article, please send e-mail to: tkde@computer.org, and reference IEEECS Log Number TKDESI-0203-0406. Digital Object Identifier no. 10.1109/TKDE.2007.190647.

the same d-bit CRC. Since $CRC(m) = m(\mathrm{mod}P)$, for any file m'', take $m' := m'' + (m(\mathrm{mod}P)) - (m''(\mathrm{mod}P))$ to get $CRC(m') = CRC(m)$. If the attacker has enough control of the data vault to replace m by m' (this is the case if the data vault provider herself tries to cheat and save storage by keeping an m' much shorter than m), the substitution will not be detected by the data vault customer who compares against the CRC of m. If the attacker cannot control the data vault but can control the communication channel (man in the middle), she can replay the CRC of a previous version of m, different from the current one; this replay attack also applies if a cryptographic hash is used instead of a CRC. □

Remote data possession checking protocols have been proposed in the last few years [3], [6]. Using one of such protocols, a prover can convince a verifier that the prover has access to a complete and uncorrupted version of a data file m. The following requirements ought to be satisfied for a remote data possession checking protocol to be of practical use:

- **R1.** The verifier should *not* be required to keep an entire copy of the file(s) to be checked. It would be impractical for a verifier to replicate the content of all provers to be verified. Storing a reduced-size digest of the data at the verifier should be enough.
- **R2.** The protocol has to stay secure even if the prover is malicious. A malicious prover is *interested* in proving knowledge of some data that she does not entirely know; security means that such a prover ought to fail in convincing the verifier.
- **R3.** The amount of communication required by the protocol should be low.
- **R4.** The protocol should be efficient in terms of computation.
- **R5.** It ought to be possible to run the verification an unlimited number of times.

When the first version of this paper was submitted (April 2006), no contribution in the literature met *all* the above requirements. The trivial approach of the prover sending the entire data file to the verifier to show possession clearly violates R3. As discussed above, having the prover send a CRC or a cryptographic hash of the file to the verifier fails to meet R2; this is why the commercial products (for example, Tripwire [12], [8]) following this approach assume that the verification is conducted *locally* by rebooting the server from a secure medium. In [3], two remote verification protocols are proposed: the first one violates R5 (limited number of verifications); the second one (also proposed in [6]) fails to meet R4 (the computational cost for the prover is extremely high).

At the time of preparing the final version of this paper (July 2007), two relevant unpublished preprints have just been posted on the Internet, coauthored by people from Google and RSA Labs, respectively. The preprint [2] uses ideas from an earlier internal report version of this paper [10]; their probabilistic optimization can be easily added to our scheme, but it does not fulfill requirement R2 with 100 percent probability. On the other hand, the preprint from that in [7] presents a system to prove backup possession, which requires modifying and encrypting stored data, so that it is not well suited as an IDS component (the IDS needs files in the clear).

1.1 Contribution and Plan of this Paper

In this paper, we present a protocol whereby a prover can show data possession, that is, that she has access to a complete and uncorrupted version of a given file. The new protocol meets all the above requirements: the verifier is not required to store the complete file but simply a digest of it (R1); the protocol stays secure (R2) in the presence of a cheating prover (who tries to pass the verification without having access to the complete file) or an untrusted channel between prover and verifier (with a man in the middle); a trade-off between storage requirements at the verifier and computing time at the prover is possible (R3); communication is low (R4); unlimited verifications are possible (R5).

The rest of this paper is organized as follows: The new data possession checking protocol is described in Section 2. The security of the new protocol is analyzed in Section 3, and its performance is examined in Section 4. Section 5 is a conclusion. The Appendix contains security proofs.

2 A Data Possession Checking Protocol

The proposal described next follows the ideas of the Diffie-Hellman-based approach in [3] and [6] in that it seeks to permit an arbitrary number of verifications. A clear advantage of our proposal is that it satisfies requirement R4, because the computing time of verification can be reduced at the set-up stage by trading off the computing time required at the prover against the storage required at the verifier.

During setup, the following values are set. Let $N = pq$ be an RSA modulus, created by the verifier, with two prime factors; this value is public. Let $\phi(N) = (p-1)(q-1)$ be secret and private (only known by the verifier). Let l be an integer properly chosen to trade off the storage requirements at the verifier and the computational cost at the prover (see Section 4 for guidance on the choice of l). Let t be a security parameter and $PRNG$ be a pseudorandom number generator generating t-bit integer values as output.

The *verifier* precomputes the digest of data m in the following way:

1. Split the data m into l-bit pieces. Let
$$m_1, m_2, \cdots m_n (n = \lceil |m|/l \rceil)$$
be the integer values corresponding to fragments of m (the last fragment is padded with 0s in the most significant bit positions if its length is less than l).
2. For each fragment m_i, compute and store $M_i = m_i \bmod \phi(N)$.

The challenge-response verification protocol is as follows:

1. The verifier
   - generates a random seed S and a random element $a \in \mathbb{Z}_N \setminus \{1, N-1\}$, and
   - sends the challenge (a, S) to the prover.
2. Upon reception, the prover
   - generates n pseudorandom values $c_i \in [1, 2^t]$, for $i = 1$ to n, using $PRNG$ seeded by S,

- computes $r = \sum_{i=1}^{n} c_i m_i$ and $R = a^r \bmod N$, and
- sends R to the verifier.

3. The verifier
   - regenerates the n pseudorandom values $c_i \in [1, 2^t]$, for $i = 1$ to n, using $PRNG$ seeded by S,
   - computes
   $$r' = \sum_{i=1}^{n} c_i M_i \bmod \phi(N)$$
   and $R' = a^{r'} \bmod N$, and
   - checks whether $R \stackrel{?}{=} R'$.

The second protocol in [3] is a special case of our protocol: taking $n = 1$ in the latter yields (essentially) the former.

3 Correctness and Security Analysis of the Protocol

Correctness in a remote data possession checking protocol means that a prover knowing the complete version of data m should be able to successfully pass the verification. Security means that the prover should not be able to pass the verification unless it has access to the *complete* unaltered version of m. The following theorem proven in the Appendix characterizes the correctness and the security of our protocol:

Theorem 1 (Correctness and security). *Assuming that the prover does not know the factorization of $N = pq$ and assuming that the RSA and the Diffie-Hellman problems are hard over $\mathbb{Z}_N$, the prover can successfully respond to the challenge in our protocol with non-negligible probability if and only if she has non-negligible probability of accessing the complete file m.*

4 Performance Analysis of the Protocol

In this section, we present a performance analysis of our protocol.

4.1 Storage, Communication, and Computational Cost

Next, we detail the storage required by the prover and the verifier:

- **Prover side.** In line with the purpose of the protocol, the prover has to store the complete data m, whose bitlength is $|m|$ bits.
- **Verifier side.** The storage requirements for the verifier are $|N|n = |N|\lceil |m|/l \rceil$ bits (in front of $|N|$ bits stored by the verifier in the second protocol in [3]). The size of the stored data is proportional to the number of fragments n. The factor between what the verifier stores and what the prover stores is $|N|/l$, so the protocol makes sense only if $l \gg |N|$.

The communication cost consists of the challenge sent by the verifier to the prover, with constant bitlength $|N| + |S|$, and the response sent by the prover to the verifier, with constant bitlength $|N|$.

The computational cost can be assessed as follows:

- **Prover side.** During verification, the prover generates n pseudorandom t-bits integers c_i. Then, it computes the value $r = \sum_{i=1}^{n} c_i m_i$. The computation of each $c_i m_i$ corresponds to the product of two integers being t and l bits long, respectively. The cost of this operation is upper bounded by that of $t - 1$ additions of $(t + l)$-bit integers. Once the values $c_i m_i$ are computed, r is obtained by computing $n - 1$ additions of $(t + l)$-bit integers. As the resulting r has a bitlength of at most $|n| + t + l$ bits, the cost of this operation is upper bounded by that of computing $n - 1$ additions of $(|n| + t + l)$-bit integers. Next, the prover computes $R = a^r \bmod N$.

  In summary, the cost of computing R is upper bounded by the cost of generating n pseudorandom t-bit integers, plus the cost of computing one exponentiation of a number in $\mathbb{Z}_N$ to an $(|n| + t + l)$-bit exponent, plus the cost of $n(t - 1)$ additions of $(t + l)$-bit integers, plus the cost of $n - 1$ additions of $(|n| + t + l)$-bit integers. Since $time_{add}(t + l) < time_{add}(|n| + t + l)$, and $n = \lceil |m|/l \rceil$, we obtain the following upper bound on the prover's computation time:
  $$\lceil |m|/l \rceil time_{prng}(t) + time_{exp}(|\lceil |m|/l \rceil| + t + l, N) + t\lceil |m|/l \rceil time_{add}(|\lceil |m|/l \rceil| + t + l).$$

  The above time is to be compared to the computing time $time_{exp}(|m|, N)$ required by the second protocol in [3].
- **Verifier side.** Except for two additional pseudorandom number generations corresponding to the challenge, the cost analysis for computing R' is similar to that on the prover side but replacing l-bit operations by $|N|$-bit operations. Therefore, the verifier computation time is upper bounded by
  $$(2 + \lceil |m|/l \rceil) time_{prng}(t) + time_{exp}(|\lceil |m|/l \rceil| + t + |N|, N) + t\lceil |m|/l \rceil time_{add}(|\lceil |m|/l \rceil| + t + |N|).$$

  Since $|N| \ll l$, the time to compute R' is smaller than the time to compute R. Note that, in our protocol, R and R' can be computed in parallel, so that the time to compute R' does not influence the overall protocol execution time. The latter is dominated by the time to compute R.

4.2 Addition, Exponentiation, and Pseudorandom Generation Times

In order to complete the performance analysis, we have measured the average times required for 1) exponentiating an element in $\mathbb{Z}_N$, with N being 1,024 bits long, to an l-bit long exponent; 2) adding two l-bit long integers; 3) given $t \leq 160$, generating a nonzero t-bit pseudorandom integer c_i as the t least significant bits of $SHA(S\|i)$, where $SHA(\cdot)$ is the SHA-1 secure hash algorithm [5], and S is a seed. (If the t least significant bits of $SHA(S\|i)$ are all zeroes, then $j = 1, 2, \cdots$ is tried until the first j is found such that *not all*

TABLE 1
For a 1,024-Bit Long N and Several Values of l, Average Time to Exponentiate to an l-Bit Exponent over $\mathbb{Z}_N$ and Average Time to Add Two l-Bit Integers

Bitlength l	$time_{exp}(l, N)$ (ms)	$time_{add}(l)$ (ms)
32,768 (2^{15})	770	0.0052
65,536 (2^{16})	1,560	0.0128
131,072 (2^{17})	3,130	0.0224
262,144 (2^{18})	6,240	0.0448
524,288 (2^{19})	12,500	0.0919

t least significant bits of $SHA(j\|S\|i)$ are zero. These are taken to be c_i.)

Exponentiation and addition times have been measured with a C++ program using the NTL [11] library running on a PC with an Intel Pentium 4 processor clocked at 3 GHz and a Linux Debian operating system. They are summarized in Table 1.

Regarding pseudorandom number generation using SHA-1, we measured an average time of 5.55 ms to run the SHA-1 algorithm on the aforementioned computer platform. The average was computed to be over 1,000 runs of SHA-1 using the test values suggested by the National Institute of Standards and Technology (NIST) and the source code in [4].

4.3 An Example

We next give a numerical example to illustrate the feasibility and the performance improvement of our proposal. Computing times shown in Table 1 are used in the example.

Let m be a 2-Mbyte (2^{24} bits) data file stored in an offsite data vault. The bitlength of the RSA modulus N (and of $\phi(N)$) is taken to be 1,024 bits. Let parameter t be 128. Let us assume that the time constraints of the prover (data vault provider) require her not to spend more than 4.5 sec in her computation. Then, we can choose the fragment size parameter l to be 131,072 (2^{17}) bits. In this way, the prover would spend a time at most

$$\lceil |m|/l \rceil time_{prng}(t) + time_{exp}(|\lceil |m|/l \rceil| + t + l, N) + t\lceil |m|/l \rceil time_{add}(|\lceil |m|/l \rceil| + t + l)$$

which, since in our example

$$|\lceil |m|/l \rceil| + t + l = 7 + 128 + 131,072,$$

is

$$128 \cdot 5.5 + 3,133 + 128 \cdot 2^{24}/2^{17} \cdot 0.0225 = 4,206 \text{ ms} < 4.5 \text{ s}$$

the information stored by the verifier (data vault customer) is $|N|\lceil |m|/l \rceil = 2^{10} \cdot 2^{24}/2^{17} = 2^{17}$ bits (16 Kbytes). Assuming that seeds S are 128 bits long, challenges would have a bitlength $|N| + |S| = 1,024 + 128 = 1,152$ bits. Responses have a bitlength $|N| = 1,024$ bits.

TABLE 2
Trade-Off between Storage at the Verifier and Computation Time at the Prover

l	Storage at verifier (KBytes)	Max. time at prover (ms)
2^{15}	64	3,931
2^{16}	32	3,391
2^{17}	16	4,206
2^{18}	8	6,962
2^{19}	4	13,055

The same verification using the second protocol in [3] (also described in [6]) would take 402 sec. This time corresponds to the time required to exponentiate an element in $\mathbb{Z}_N$, with N being 1,024 bits long, to a 2^{24}-bit long exponent. Let us now examine the trade-off between computing time and storage. Our protocol reduces that computing time by a factor of 100 (roughly $\lceil |m|/l \rceil$); the price paid is that the verifier storage in our protocol (16 Kbytes) is also $\lceil |m|/l \rceil = 128$ times larger than in the second protocol in [3] (1 Kbit).

Table 2 shows the trade-off between storage requirements and computation time for different choices of parameter l.

In general, larger values for l lead to longer computing times. The reason is that the prover's computing time is dominated by the exponentiation operation. Smaller values for l cause this exponentiation time to decrease down to some point in which further decreasing l increases computing time. This is because, at this point, the computation time begins to be dominated by the computation of pseudorandom values.

If very short computing times are desired, a very fast pseudorandom number generator for values c_i must be used.

Note 2. In a real deployment of a verification system using our protocol or any remote integrity checking protocol, the time to access files should be taken into account when evaluating performance. However, this is unlikely to have substantial impact, as a state-of-the-art hard drive technology allows as much as 1 Mbyte to be read in as little time as a few nanoseconds.

5 Concluding Remarks

The first practical protocol for remote file integrity checking allowing an infinite number of verifications has been presented. As mentioned above, the probabilistic optimization described in [2] can be easily added to our scheme in order to obtain $O(1)$ block access cost for the prover (a cost independent of the file size); the price paid is that the proof of possession obtained in this way is probabilistic, that is, requirement R2 above is not met with 100 percent probability.

The generalization of our protocol to check the integrity of a set of files in a single verification round is straightforward. To do this, an ordering or a structure between the set of files should be defined, so that the set of files can be regarded as a superfile. Once the superfile is defined, its integrity can be checked using our protocol without any modification.

APPENDIX

To prove Theorem 1, we first give two instrumental results (Lemma 1 and Corollary 1).

Lemma 1. *Given two integer values x and N, computing a value y such that $y \neq x$ and $x \equiv y \pmod{\phi(N)}$ is as hard as computing a value y' such that $x \cdot y' \equiv 1 \pmod{\phi(N)}$.*

Proof. Let us assume that there exists an algorithm $\mathcal{A}(x, N)$ that returns an integer value y such that $y \neq x$ and $x \equiv y \pmod{\phi(N)}$.

Assuming, without loss of generality, that $x > y$, this means $(x - y) = k\phi(N)$ for some integer k. In this way, we can compute $y' \equiv x^{-1} \pmod{(x-y)}$. Thus, $x \cdot y' = 1 + k'(x - y)$ for some k', so that $x \cdot y' = 1 + k'k\phi(N)$. Finally, we conclude that $y' \equiv x^{-1} \pmod{\phi(N)}$. □

Corollary 1. *Given an RSA modulus $N = pq$ and an integer value x, obtaining a value y such that $y \neq x$ and $x \equiv y \pmod{\phi(N)}$ is as hard as obtaining an RSA private key $x^{-1} \pmod{\phi(N)}$ from the corresponding public key (x, N).*

The proof of Theorem 1 can now be stated.

Proof (Theorem 1). Let the file m be composed of fragments $m_1, \ldots, m_n$. The prover correctly responds to the challenge in our protocol if she can obtain a value R satisfying $R \equiv a^{\sum_{i=1}^{n} c_i M_i} \pmod{N}$, where $M_i = m_i \bmod \phi(N)$:

1. *Correctness.* If the prover knows all fragments, then she can compute $r = \sum_{i=1}^{n} c_i m_i$ and $R = a^r \bmod N$, which is a valid response.
2. *Security.* Without loss of generality, let us assume a prover who has negligible probability of knowing the first k fragments $m_1, \ldots, m_k$ and who does not know the factorization of N.

   Let us assume that this prover is nonetheless able to compute a valid response R returned by a polynomial-time algorithm $\mathcal{A}(a, c_1, \ldots, c_n, \hat{m})$, where a and $(c_1, \ldots, c_n)$ correspond to the current challenge (a, S), and $\hat{m}$ is the information possessed by the prover on m.

   Using knowledge on $m_{k+1}, \ldots, m_n$, the prover can obtain $a^{\sum_{i=1}^{k} c_i M_i} \pmod{N}$ computed as $R\left(a^{\sum_{i=k+1}^{n} c_i m_i}\right)^{-1} \pmod{N}$. From the Diffie-Hellman assumption, for any random a, we have that $a^{\sum_{i=1}^{k} c_i M_i} \pmod{N}$ can only be obtained with non-negligible probability if a value $X \equiv \sum_{i=1}^{k} c_i M_i \pmod{\phi(N)}$ is known. Therefore, at some part of the computation, the value X can be obtained.

   Using algorithm $\mathcal{A}$, the prover can compute

$$T_1 = \mathcal{A}(a, c'_1, \ldots, c'_n, \hat{m})\left(a^{\sum_{i=k+1}^{n} c'_i m_i}\right)^{-1} \pmod{N}$$
$$\ldots$$
$$T_k = \mathcal{A}(a, c_1^{(k)}, \ldots, c_n^{(k)}, \hat{m})\left(a^{\sum_{i=k+1}^{n} c_i^{(k)} m_i}\right)^{-1} \pmod{N}$$

   for some vectors $(c'_1, \ldots, c'_n), \ldots, (c_1^{(k)}, \ldots, c_n^{(k)})$.

   As said before, during the computation of T_1, a value $X_1 = \sum_{i=1}^{k} c'_i m_i \bmod \phi(N)$ can be obtained. In the same way, during the computation of T_2, the prover can obtain $X_2 = \sum_{i=1}^{k} c''_i m_i \bmod \phi(N)$ and so on until T_k is computed, from which $X_k = \sum_{i=1}^{k} c_i^{(k)} m_i \bmod \phi(N)$ is obtained.

   If vectors $(c'_1, \ldots, c'_n), \ldots, (c_1^{(k)}, \ldots, c_n^{(k)})$ are linearly independent, the prover can obtain values $m'_1, \ldots, m'_k$ by solving the following equation system:

$$\begin{cases} X_1 = \sum_{i=1}^{k} c'_i m'_i \\ \quad \ldots \\ X_k = \sum_{i=1}^{k} c_i^{(k)} m'_i. \end{cases}$$

   These values satisfy that $m'_1 \equiv m_1 \bmod \phi(N)$, $m'_2 \equiv m_2 \bmod \phi(N), \ldots, m'_k \equiv m_k \bmod \phi(N)$. From Corollary 1, assuming that the RSA cryptosystem is secure, the prover cannot find any value $m'_i \neq m_i$ satisfying $m'_i \equiv m_i \bmod \phi(N)$ for any fragment m_i of file m. Therefore, the only possibility is that $m'_1 = m_1$, $m'_2 = m_2, \ldots, m'_k = m_k$. From the initial assumption on the negligible probability of knowing $m_1, m_2, \ldots, m_k$, such an algorithm $\mathcal{A}$ working properly with non-negligible probability for arbitrary linearly independent vectors $(c_1, \ldots, c_n)$ cannot exist.

   Since in our protocol values, c_i are t-bit long random integers, any two random vectors $(c_1^{(i)}, \ldots, c_k^{(i)})$, $(c_1^{(j)}, \ldots, c_k^{(j)})$ will be linearly independent with probability $1 - 2^{-t(k-1)}$. This probability can be made overwhelmingly high if the security parameter t is chosen large enough. □

ACKNOWLEDGMENTS

The first three authors are partly supported by the Government of Catalonia and the Spanish Ministry of Science and Education through projects 2005 SGR 00446, CSD2007-00004 CONSOLIDER INGENIO 2010 "ARES," TSI2007-65406-C03-01, and SEG2004-04352-C04-01. They are with the UNESCO Chair in Data Privacy, but their views do not necessarily coincide with UNESCO nor commit that organization.

REFERENCES

[1] Allmydata Inc., Unlimited Online Storage and Backup, http://allmydata.com, 2007.

[2] G. Ateniese, R. Burns, R. Curtmola, J. Herring, L. Kissner, Z. Peterson, and D. Song, "Provable Data Possession at Untrusted Stores," *Cryptology ePrint Archive,* Report 2007/202, http://eprint.iacr.org/, May 2007.

[3] Y. Deswarte, J.-J. Quisquater, and A. Saidane, "Remote Integrity Checking," *Integrity and Internal Control in Information Systems VI,* pp. 1-11. Kluwer Academic Publishers, Nov. 2003.

[4] C. Devine, "SHA-1 Source Code," http://www.cr0.net:8040/code/crypto/sha1/, 2004.
[5] FIPS-PUB-180-1, "Secure Hash Standard,"Technical Report FIPS PUB 180-1, Nat'l Inst. of Standards and Technology, http://csrc.ncsl.nist.gov/fips/fip180-1.txt, 1995.
[6] D.L. Gazzoni-Filho and P.S. Licciardi-Messeder-Barreto, "Demonstrating Data Possession and Uncheatable Data Transfer," *Cryptology ePrint Archive,* Report 2006/150, http://eprint.iacr.org/, 2006.
[7] A. Juels and B.S. Kaliski Jr., "PORs: Proofs of Retrievability for Large Files," *Cryptology ePrint Archive,* Report 2007/243, http://eprint.iacr.org/, June 2007.
[8] G.H. Kim and E.H. Spafford, "The Design and Implementation of Tripwire: A File System Integrity Checker," *Proc. ACM Conf. Computer and Comm. Security,* pp. 18-29, 1994.
[9] Network Technology Group, "DataVault Offsite Data Backup to Completely Secure Critical Computer Data," http://www.ntg.com/datavault.asp, 2007.
[10] F. Sebé, A. Martínez-Ballesté, Y. Deswarte, J. Domingo-Ferrer, and J.-J. Quisquater, "Time-Bounded Remote File Integrity Checking," Technical Report 04429, LAAS-CNRS, July 2004.
[11] V. Shoup, "NTL: A Library for Doing Number Theory," http://www.shoup.net/ntl, 2004.
[12] Tripwire Inc., Tripwire Checking Software, http://www.tripwire.com, 2007.
[13] A. Valdes, M. Almgren, S. Cheung, Y. Deswarte, B. Dutertre, J. Levy, H. Saïdi, V. Stavridou, and T.E. Uribe, "An Architecture for Adaptive Intrusion-Tolerant Server," *Proc. Security Protocols Workshop 2002,* pp. 158-178, 2003.
[14] Webopedia, Data Vaulting, http://www.webopedia.com/TERM/D/data_vaulting.html, 2007.
[15] Wikipedia, Cyclic Redundancy Check, http://en.wikipedia.org/wiki/Cyclic_redundancy_check, 2007.

Francesc Sebé received the MSc degree in computer engineering from Rovira i Virgili University in 2001 and the PhD degree in telematics engineering from the Polytechnical University of Catalonia, Barcelona, in 2003. He is an associate professor in telematics engineering at Rovira i Virgili University of Tarragona, Catalonia, where he is currently a member of the UNESCO Chair in Data Privacy. He has participated in several European- and Spanish-funded research projects. He has authored more than 40 international publications. His research interests include cryptography and information privacy. He was a corecipient of a research prize from the Association of Telecom Engineers of Catalonia in 2003. He was a visiting researcher at LAAS-CNRS, Toulouse, France, in 2004.

Josep Domingo-Ferrer received the MSc degree (with honors) and the PhD degree (Outstanding Graduation Award) in computer science from the Autonomous University of Barcelona in 1988 and 1991, respectively, and also the MSc degree in mathematics. He is a full professor of computer science at Rovira i Virgili University of Tarragona, Catalonia, where he is the chairholder of the UNESCO Chair in Data Privacy. His research interests include data privacy, data security, and cryptographic protocols. He was a corecipient of a research prize from the Association of Telecom Engineers of Catalonia. He received The Outstanding Young Persons (TOYP) 2004 Award from the Junior Chambers of Catalonia. He is the holder of three patents and more than 180 publications, one of which became an ISI highly cited paper in early 2005. He is the coordinator of the CONSOLIDER "ARES" team on security and privacy, one of the 34 strongest scientific teams in Spain. He was the coordinator of the EU FP5 project COORTHOGONAL and of several Spanish-funded and US funded research projects. He has chaired or cochaired nine international conferences on security and privacy and is an associate editor of three international journals. In 2004, he was a visiting fellow at Princeton University. He has also visited Siemens AG Corporate R+D (Munich), the University of Wisconsin, and Katholieke Universiteit Leuven.

Antoni Martínez-Ballesté received the MSc degree in computer engineering from Rovira i Virgili University of Tarragona (URV), Catalonia, in 2002 and the PhD degree (with honors) in telematics engineering from the Technical University of Catalonia (UPC) in 2004. He is a tenured assistant professor at URV, where he is currently a member of the UNESCO Chair in Data Privacy. His research interests include security in computer networks and privacy in information and communication systems. He was a visiting researcher at the Laboratoire d'Analyse et d'Architectures des Systèmes-Centre National de la Recherche Scientifique (LAAS-CNRS), Toulouse, France, in 2004. He has authored more than 25 articles in journals and conferences. He is also currently working on innovative learning techniques in engineering and the European Higher Education Space.

Yves Deswarte is currently a research director of the Centre National de la Recherche Scientifique (CNRS), member of the "Dependable Computing and Fault Tolerance" Research Group, LAAS-CNRS, Université de Toulouse, France. Successively, at Compagnie Internationale pour l'Informatique (CII), Compagnie pour l'Informatique Militaire, Spatiale et Aéronautique (CIMSA), l'Institut National de Recherche en Informatique et en Automatique (INRIA), and LAAS, his research work has dealt mainly with fault-tolerance and security in distributed computing systems. Recently, his main research interests include intrusion tolerance, quantitative security evaluation, dependability evaluation criteria, protection of safety-critical systems with multiple levels of integrity, flexible security policies, and privacy-preserving authorization schemes, and he is the author or coauthor of more than 100 international publications in these areas. He has been a consultant for several organizations in France and for SRI International, USA. He has been a member of many international conference program committees and has chaired several of them. He is a senior member of the Societe de l'Electricite, de l'Electronique, et des Technologies de l'Information et de la Communication (SEE), a member of the IEEE Computer Society Technical Committee on Security and Privacy, and a member of the ACM Special Interest Group on Security, Audit and Control (SIGSAC). He is representing the IEEE Computer Society at the International Federation for Information Processing Technical Committee on Security and Protection in Information Processing Systems (IFIP TC-11). He is a member of the IEEE.

Jean-Jacques Quisquater is a professor of cryptography and multimedia security in the Laboratory for Microelectronics (DICE), Department of Electrical Engineering, Catholic University of Louvain (UCL), Louvain-la-Neuve, Belgium, where he is responsible for the UCL Crypto Group, working on research projects related to smart cards (hardware and software), RFIDs, secure protocols for communications, digital signatures, payTV, protection of copyrights (DRM), and security tools for electronic commerce. He is the inventor and designer of the first cryptographic coprocessor for smart cards. He has published about 200 scientific papers and is the holder of 20 patents including the so-called Guillou-Quisquater protocol. He is a member of the IEEE.