Secure Logging for Irrefutable Administration

Francesco Bergadano[1], Davide Cavagnino[1], Paolo Dal Checco[1],
Pasquale Andrea Nesta[1], Michele Miraglia[1], and Pier Luigi Zaccone[2]
(Corresponding author: Francesco Bergadano)

Dipartimento di Informatica, Universita' degli Studi di Torino[1]
Corso Svizzera 185 - 10149 Torino - Italy. (E-mail: dalche@di.unito.it)
Telecom Italia S.p.A.[2]
Via Reiss Romoli 274 - 10148 Torino - Italy

(Received Oct. 07, 2005; revised and accepted Dec. 31, 2005)

Abstract

This paper presents a method that allows for securely saving a temporal sequence of data (log lines) in a file. Log lines are signed by an authority, and are thus unalterable without detection. Data is also encrypted in the file, and may be accessed with the granularity of a single log line with the possession of a decryption key. Also, it is possible that for some lines data must be accessed by a group of cooperating users.

Keywords: Auditing, computer forensics, logging, modification detection, privacy, secret sharing

1 Introduction

In many applications there are contexts in which it is necessary to check and verify the operations performed by some entity (possibly an administrator) on another entity (possibly a computer system). For example, in industrial environments some jobs are left in outsourcing to external companies; the operations performed by the external personnel should be controlled in some way, and at the same time, the privacy of the workers must be guaranteed, with the ability, in case of need, to verify and link the operations performed with the person who made them.

The system proposed in the present paper is related to the previously discussed context, considering system and network administrators, administered systems and networks, with the objective of giving a secure and reliable auditing system. The main characteristics of this system are the ability of logging all the operations that occur in a complex environment, linking these operations to the entities involved (namely, the administrator of the system and the system itself), with the guarantees that:

- The log does not immediately disclose its content (for privacy reasons), i.e. it is encrypted;
- The log's content may be examined only by the entities having the rights to perform this operation (i.e. only the authorized people, administrators or third parties, may decrypt the log entries content, with some defined modes);
- Log entries cannot be directly related to the entities that are involved in the activity described in the log entry itself;
- The log cannot be modified without detection (i.e., if the log is modified this can be discovered by the auditors that will eventually check the content).

The ideas presented in this paper have many fields of application, where the main problem to be solved is the logging of some activity for a subsequent control.

The paper is organized as follows: Section 2 presents some works related to the argument we deal with, Section 3 presents the terms of the problem and foresees some solutions, deeply discussed in Section 4. Section 5 shows some characteristics of the solution we propose, whilst Section 6 deals with the specific problem of permitting a set of users to access a log line. Finally, Section 7 presents some conclusions.

2 State of the Art

The commonly accepted definition of log is given in [13], as: "a plain file where data are stored sequentially as they arrive, by appending them to the end of the file. When a problem arises in the system (e.g. a fault or an intrusion), the log is reread to find its source and/or to correct its consequences.".

Ruffin, in his detailed discussion, does not even examine the possible requirement that logs should not be manipulated by malicious users or intruders. In modern systems, log files may contain important and/or classified information, and are stored on machines that should not be considered completely secure. Then, it is necessary that the logs themselves be made intrinsically secure, pre-

venting an attacker from accessing and reading these logs and in case modifying them.

Some approaches have been developed to authenticate or certify the content of different types of log files, possibly of large size. [2] and [3] present some possible methods.

A first hypothesis on the possible detection of log file manipulation has been made in [5], where it is proposed to authenticate every entry of the log file by means of a Message Authentication Code (MAC). This system guarantees that if an attacker gains control over a machine (containing the log files) at time t, then he will not be able to modify (without detection) the content of the log files generated at a time before t. This is made possible using a key for the MACs that changes over time, with no information kept on the system about previous keys. To accomplish this, a first key that must be kept secret is generated; this key is used to authenticate a first set of entries in a time interval. When the time interval is elapsed, a new key is generated applying a pseudo-random non-invertible function to the previous key. In this system only the first key of the chain, and the one used in the current time interval, have to be maintained. An attacker having control over the system will not be able to undetectably modify the entries because he is not able to obtain the keys used to generate the MACs. To guarantee that entries are not added or deleted, the various lines are sequentially numbered.

Schneier and Kelsey have developed a method for keeping the log files secret and for detecting in a fast way any log file modification [15, 16]. Their system involves an untrusted machine U and a trusted secure system T; with a small amount of data exchange between these two machines it is possible to keep the log file inaccessible (i.e. unreadable and not modifiable with respect to the entries logged before the attack) to an attacker that takes control of U. As in the previous system, also this solution uses authentication and encryption keys that change for every log file entry; this is obtained by means of a hash function applied to the previous keys. To avoid the insertion or the deletion of entries, these are sequentially concatenated in a hash chain that is authenticated. Schneier and Kelsey accurately describe the system, explaining how to create and close the log file, and how to deal with sudden machine shutdowns. Every log file entry belongs to a class. This class is used to specify the kind of access each auditor has on that entry, allowing for different privileges of different auditors on an entry. On the latter characteristic [17] describes how to minimize the bandwidth necessary to verify the log file content. [18] is a patent on the system developed by Schneier and Kelsey, that also introduces the possibility of using asymmetric encryption.

A possible implementation of the system by Schneier and Kelsey is described in [6], where a tamper-resistant hardware (iButton) is used to keep the secrets, having the role of maintaining information that should not be kept on an untrusted system.

Recently Waters et al. [19] made a new study on secure audit log. The problem they want to solve is to create an encrypted log that could be searched using some keywords. The approach followed to encrypt and link the log entries is very similar to the method proposed by Schneier and Kelsey; the main difference is that this method extracts some keywords from the data to be logged before they'll be encrypted. An auditor willing to search for some data has to send a keyword to a centralized trusted element; on the basis of the element's authorization policy such element may grant him the necessary information to reconstruct a key to be used for decrypting the data itself. In this paper it is not discussed how keywords could be extracted from raw data.

3 Preliminary Considerations

In our approach we consider a log file as a journal in which information coming from various activities is stored in a set of lines; each line refers to a particular event of interest in each activity. We do not consider the physical implementation, and refer to the definition given in [13].

The environment of our system is composed by administrators that perform activities on objects: these activities are logged by an entity. The job of this entity is to ensure that the content of parts of the log file is available only to the authorized people (auditors) and that this content cannot be modified without detection. We want to propose a method where there is no need for a centralized element that authorizes any new people to access a previously produced log entry. The set of people which is authorized to access stored data has to remain the same as it was when a log entry was produced. In the following we discuss techniques that can be used to obtain these objectives.

The first consideration relates to data encryption. The objective of data encryption is to allow the access to particular data only to set of users. This set may change for every logged line. Moreover, it must be possible to allow access to groups of users in which the presence of at least n users over N is required to reveal the content of a line. The chosen approach is to encrypt each line with a different key. This key is generated automatically by the system, and access to this key is given according to the kind of access we want for each user, in a exclusion/elusion strategy for the log file auditing that will be presented later.

Another goal of the system we propose is to ensure that the file cannot be altered without possible successive detection by a verifier. Thus, one objective is to avoid data forging. A solution to this requirement is to use a hash chain. A hash chain keeps the log lines linked, in the same order they were originally written, and prevents the insertion of a line between two other lines. One of the first proposals of the use of chains to connect a sequence of data is presented in [7]. [1] uses a hash chain to link a set of data transmitted in streaming; in that paper the point that remains to be solved is how to make sure that the last element of the chain is not modified.

4 Possible Approaches

As previously seen, for privacy reasons the data section of the log line is encrypted with an randomly generated key. To record this random key for later use in auditing, we consider two possible approaches:

A. Each auditor has his own symmetric secret key; the system encrypts the random key for each auditor with his secret key.

B. Each auditor has his own pair of asymmetric public/private keys; the system encrypts the random key for each auditor with his public key. The auditor will use his private key to decrypt the random key and access the log line data.

The approach of directly encrypting the log line data with the auditor's public keys was taken into consideration, but discarded due to the computational complexity of the asymmetric encryption and the amount of data that were required to be encrypted and stored. Let's see the structure of the log lines in the two cases A and B. The index i runs over the log lines; k is an index that runs over the identifiers of the entities involved in the logged transaction, and j indexes the various auditors. (In this example of line structure, auditors from 0 to $j-1$ have access to the line content, auditors from j to n have not. This will become clearer throughout the rest of the paper.)

Case A:

$$\begin{aligned} L_i &= \{TS_i, U_k, l_i, E(A_i/D_i), E(K_0/A_i), \\ &\quad \cdots, E(K_{j-1}/A_i), E(K_j/H(A_i)), \cdots, \\ &\quad E(K_n/H(A_i)), HC_i, S_i\} \end{aligned}$$

Case B:

$$\begin{aligned} L_i &= \{TS_i, U_k, l_i, E(A_i/D_i), \alpha(K_0^+/(A_i, R_0)), \\ &\quad \cdots, \alpha(K_{j-1}^+/(A_i, R_{j-1})), \alpha(K_j^+/R_j), \\ &\quad \cdots, \alpha(K_n^+/R_n), HC_i, S_i\}. \end{aligned}$$

Note, in the preceding lines, the parts that are underlined, whose meaning will be discussed deeply. In the following the meaning of the various parts is presented.

- TS_i: It is the timestamp issued by a Time Stamping Authority[1] or it is a timestamp assigned by the system. If a Time Stamping Authority comes into play, then TSi is calculated on the result of a hash function (e.g. like SHA-1 [11]) applied to S_{i-1}[2]concatenated with all of the data in L_i except TS_i, HC_i and S_i. It may express the time of logging or the time of the reception of the line. Both are possible approaches. Even if the data contained in the log line already contains a timestamp, TS_i may be useful for some cross checks on the data.

- U_k: It is a set of data related to the log entry; in our environment it represents the identifier of the user (administrator) that generated the data in the log line, along with an identifier of the administered system. For the responses from the systems, this may be an identifier of the system and of the user whose this response is sent. In order to enforce the secrecy of this field the method proposed in [19] could be used.

- l_i: It represents the length of data in cryptographic blocks.

- D_i are the data to be logged for the i-th line.

- A_i is the symmetric key, randomly generated, used to encrypt the data of the i-th log line.

- $K_0, \cdots, K_n$ are the auditor's secret keys, used in Approach A that uses symmetric encryption of A_i.

- $K_0^+, \cdots, K_n^+$ are the auditor's public keys, used in Approach B that uses asymmetric encryption of A_i.

- $R_0, \cdots, R_n$ are random values used to preserve the elusion property we will discuss in a following section.

- $E(x/y)$ represents a symmetric encryption function that uses the key x to encrypt data y; it returns the encrypted data. A good candidate function could be AES [9].

- $\alpha(x^+/y)$ represents an asymmetric encryption function that uses the key x^+ to encrypt data y; it returns the encrypted data. A function that may be used is RSA [12].

- $H(x)$ is a *one-way hash* function (like SHA-1 [11]).

- HC_i is the element of the hash chain for the i-th log line (see below).

- S_i is the signature of the element of the hash chain, that is, it corresponds to Sign(B^-/HC_i), that is the function of digital signing HC_i with the logging system private key (B^-); it returns the signature. Functions that may be used are, for example, RSA [12] or DSA [10].

Let's see how the element HC_i of the hash chain is computed. It is the hash of the previous log line hash (i.e. HC_{i-1}) concatenated with all the elements of the current line, except HC_i and S_i (obviously, because the first one is what we are computing, and the second one will depend on the first one). In formulas, we may write that (for both proposals):

[1]The decision about whether and how often a Time Stamping Authority has to be involved must be made according to the effectiveness of the vulnerability and threats associated with the system.

[2]This field prevents an attacker that obtains B^- at a certain point in time from being able to successfully forge any previously stored log lines.

Proposal A:

$$\begin{aligned} HC_i &= H(HC_{i-1}, TS_i, U_k, l_i, E(A_i/D_i), \\ &\quad E(K_0/A_i), \cdots, E(K_{j-1}/A_i), \\ &\quad E(K_j/H(A_i)), \cdots, E(K_n/H(A_i)). \end{aligned}$$

Proposal B:

$$\begin{aligned} HC_i &= H(HC_{i-1}, TS_i, U_k, l_i, E(A_i/D_i), \\ &\quad \alpha(K_0^+/(A_i, R_0)), \cdots, \\ &\quad \alpha(K_{j-1}^+/(A_i, R_{j-1})), \alpha(K^+{}_j/R_j), \\ &\quad \cdots, \alpha(K_n^+/R_n)). \end{aligned}$$

The first element of the hash chain, namely HC_1, is computed using as previous element a fixed and known value for HC_0 which may be recorded, without encryption, in the beginning of the log file. When a line is verified, the hash of the previous line should be trusted, thus a verification of the signature of the previous line should be performed.

5 Encryption and Exclusion/Elusion

The objective of this section is to introduce the intrinsic security of the log file when it is stored on any device. In fact, it has to be taken into account that the security of the log file should not change even if it is saved and copied for backup purposes. That is, the log file content should not be alterable (by anyone) and should not be visible by non-authorized people.

To avoid the disclosure of the content to non-authorized people we already introduced the idea of encrypting the data with a random key A_i (that changes for every line): thus, this key is used to encipher the data; afterwards, the key A_i is made available to the various auditors encrypting it with the personal key of every auditor that should have access to that data. When the key has been encrypted, then it is destroyed, and only the authorized auditors will be able to reconstruct the original data. If there are auditors that should not have access to a particular log line, then A_i is not encrypted for them; instead:

- $H(A_i)$: A one-way hash of the key, is encrypted in case of Approach A.
- a random value R_r[3] (different for every auditor) is encrypted with the public key in case of Approach B.

This is done for the following two reasons:

Exclusion: It is easy to exclude one or more auditors from accessing the log line data, simply giving them a fake key: a random number in Approach B or obtained from the right key, but through a non-invertible function, in Approach A. In the literature are presented many one-way hash functions easy to compute. The use of a different A_i for every line allows for a fine granularity in giving access to every log line only to a subset of auditors. Thus the exclusion is local to every log line. We used a one-way hash function in Approach A because we considered it as an efficient source of randomness. Due to the elusion property discussed below, we had to use a different random number for every auditor in Approach B.

Elusion: It is easy to see that simply looking at the log file it is not possible to understand which auditors have access to which log lines. This is due to the fact that we encrypt the key A_i for every auditor. At this point we distinguish the two Approaches A and B. previously introduced.

A. Access to a line depends on the possession of A_i, useful to decrypt the line, or $H(A_i)$, that does not allow access to the line. But, for the properties of symmetric encryption, it is impossible to deduce which case (if A_i or $H(A_i)$) has been encrypted for an auditor. Note that it is important to use $H(A_i)$ that changes for every line. In fact, suppose to use a constant value for those auditors that should not have access to a line. Then, encrypting a constant value using a fixed key (the secret key of the auditor) will disclose the lines that are not accessible to an auditor, simply by inspection of the log file looking for a repeated value for an auditor. Note also that the use of $H(A_i)$ has the only objective to create a random number in an efficient manner; that is, instead it could be used a random number B_i different from A_i.

B. From the properties of asymmetric encryption it is impossible to deduce which auditor is able to decrypt the key A_i. Note the use of the random values R_r to ensure the elusion property. For those auditors having rights to access to the log line, then the key A_i is encrypted along with a random number (different for every auditor) to ensure that an auditor decrypting the key A_i is not able, through asymmetric encryption using the other auditors' public key, to deduce which of them has access to the log line. At the same time, for auditors that do not have access to a line, a random value (also in this case, different for every auditor) is encrypted with the public key of each auditor, thus the resulting value is undistinguishable from the encryption of the correct key A_i and a random value.

[3] In some embodiments, in place of Rr the concatenation of H(Ai) and a random number Rr (different for every auditor) can be used.

6 Group Auditing

One of the objectives of the system is to give different access modes to different auditors. We have already presented a method for allowing or not the access to a line. In this section we present how to give access to a single line to a group of auditors. For example, some log lines should be decrypted only when a set of at least three auditors out of five agree on looking at its content; with this approach it is possible to access the data even if not all the auditors belonging to one group are available.

6.1 State of the Art on Secret Sharing

We give a brief introduction on some proposals for secret sharing found in the literature.

[4] presents a model for sharing a secret; the objective of the paper is to develop a method to keep copies of cryptographic keys distributing the secret among many people. The idea behind is based on n-dimensional geometry. Let's see an example of this method from [8]: suppose to distribute a secret among m parts, where at least 3 of them are required to reconstruct the secret. Suppose that the secret is represented as a point into the three dimensional space. Construct m planes such that any three of them intersect into the point representing the secret, but any two of them defines a line. Obviously, only knowing at least three of these planes, then it is possible to unambiguously identify the single point in the space.

[14] defines a method to distribute a secret among n entities where at least k of them are necessary to reconstruct the original secret. It is based on polynomial interpolation. The idea is to construct a polynomial $q(x)$ of degree $k-1$ having random coefficients except for the coefficient of x^0 that must be equal to the secret to share. Distributing the evaluation of the polynomial into n different points, then it is possible to calculate the k coefficients (thus the shared secret also) only when at least k evaluations of the polynomial are available. This can be done interpolating the polynomial in the k points (note that there is only *one* polynomial of degree $k-1$ that fits into k points, and there are *infinite* polynomials of the same degree that fit into $k-1$ points).

6.2 Group Access to a Log Line

In our application, we use the method from [14], with the following constraints:

- Each auditor should be able to access the content of a log line both alone (if he has the rights) or with the cooperation of other auditors (if he belongs to a group of auditors that should have access to the line);
- When a group of auditors has used a secret to disclose the content of a line, then this secret must be useless if used to disclose the content of other lines; the reason lies in the fact that when a group of auditors agree in looking at the content of a line, then some of them may not agree in disclosing the content of other lines to the members of the same group;
- Each auditor may belong to any number of groups (also none).

We obtain the previous results by distributing to the auditors that need a group access to a log line, a share to determine the secret A_i. That is, instead of encrypting the secret A_i for an auditor, we encrypt a part that allows the reconstruction of the complete secret A_i. This implies that to decrypt a line there may be:

- Users that have access to the line as alone entities, i.e. they have $E(K_j/A_i)$ or $\alpha(K_j^+/(A_i,R_j))$[4];
- Users that do not have access to the line as alone entities, i.e. they have $E(K_j/H(A_i))$ or $\alpha(K_j^+/(R_j))$[4];
- Users that have access to the line only with the collaboration of at least k users, i.e. they have $E(K_j/\sum_{A_i})$ or $\alpha(K_j^+/SA_i)^4$, where $\sum_{A_i}$ is the share of a secret that allows disclosing A_i with the collaboration of other $k-1$ users. Users may belong to many groups, thus having many shares of the secret (obviously, the various shares will be related to different polynomials).

Note that the three sets of users may be not disjoint (the first two are obviously disjoint). Thus, our system allows for users that may access a log line by themselves, or in collaboration with other users also, or only when other group members agree in disclosing the content of a line.

Let's see which data is saved for every auditor that potentially has access to a line:

$$E(K_j/[H(A_i), ID_{group'}, \sum{}'_{A_i}, ID_{group''}, \sum{}''_{A_i}, \cdots])$$
$$\alpha(K_j^+/[R_j, ID_{group'}, \sum{}'_{A_i}, ID_{group''}, \sum{}''_{A_i}, \cdots])^4$$
or, for some embodiments :
$$\alpha(K_j^+/[H(A_i), R_j, ID_{group'}, \sum{}'_{A_i}, ID_{group''}, \sum{}''_{A_i}, \cdots]R_j)^4.$$

In this example the j-th auditor has not access as individual, but only as belonging to some groups. If a user does not belong to a group (or a group does not have access to the line) then $\sum$ may be left as a set of zeroes of the right size (using a proper encryption function, all this data will preserve the elusion property).

To add an auditor to the group of auditors, it is sufficient to give him a new share based on the polynomial, encrypting this share with the auditor's key. To exclude an auditor from a group it is sufficient not to give him his share anymore.

To modify the minimum number of auditors necessary to disclose a log line, a different polynomial should be used, according to [14].

[4] If each auditor has his own pair of asymmetric public/private keys.

To work properly and to be able to decrypt correctly a log line for a group, the system requires at least the following information for each group:

- A group identifier;
- The minimum number of auditors that are required to disclose the secret;
- The identifiers of all the auditors belonging to the group.

6.3 Observations on Multiple Groups

A question that may arise on the security of the method applied on multiple groups is: what happens if shares of different groups on the secret A_i are joined together? Do these parts allow the determination of A_i? That is, let's suppose the worst case. Imagine $m'-1$ auditors of a group (requiring m' auditors to compute A_i) colluding with $m''-1$ auditors of another group (requiring m'' auditors to compute A_i). Moreover, note that the two groups may overlap.

Let's write the two polynomials we want to determine:

$$\begin{aligned} y &= \alpha_{m'-1}x^{m'-1} + \alpha_{m'-2}x^{m'-2} + \cdots + \alpha_1 x + A_i \\ y &= \beta_{m''-1}x^{m''-1} + \beta_{m''-2}x^{m''-2} + \cdots + \beta_1 x + A_i. \end{aligned}$$

The target is to determine the α values, the β values and A_i; that is, in all $m'+m''-1$ values. The colluding auditors have $m'+m''-2$ points (possibly not distinct), $m'-1$ from one polynomial, and $m''-1$ from the other polynomial. This allows to write a system of $m'+m''-2$ equations in $m'+m''-1$ variables. The target may not be reached because the system of equations is undetermined if we make the assumption that a single polynomial of degree m is undetermined if only $m-1$ points are available. But, to discover the shared key, it is sufficient to determine A_i: we show that this is not possible. We will call the set of equations coming from the first polynomial $\prod$ and the set of equations coming from the second polynomial Θ.

Given that A_i cannot be determined from Π ([14]), then reducing this set should bring to an equation of this kind:

$$c_1\alpha_j + c_2 A_i = b_1.$$

For the same reason, reducing Θ will lead to

$$c_3\beta_k + c_4 A_i = b_2,$$

where the $c_m s$ and $b_n s$ are constant values.

The system of these two equations does not allow to determine A_i because α_j and β_k are different unknowns (they are coefficients from different polynomials). Thus, to answer the question we posed in the beginning of this section, even if different auditors from different groups collude to determine the shared key, they will not be able to get it unless the required number of auditors in one of the groups is reached.

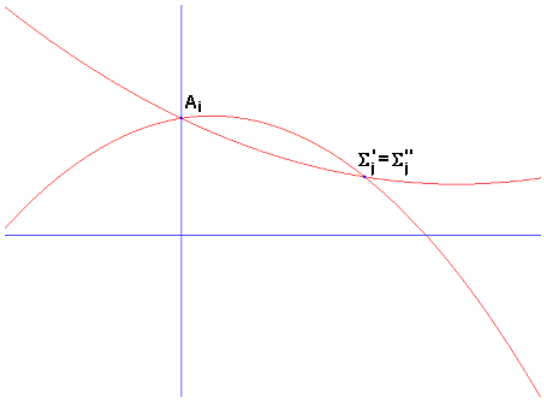

Figure 1: Common share of different polynomials

The same demonstration works also in the case two auditors belonging to different groups own the same share (i.e. the same point in the plane, where two distinct polynomials intersect).

7 Conclusions

In this paper we dealt with the problem of keeping the content of a log file both unalterable (without detection) and private.

The first objective is obtained through the use of signatures and a hash chain that links all the log lines together.

The privacy of the content is obtained by means of encryption. Log lines are encrypted with a key that is successively encrypted with the keys of the entities that should have access to the line; moreover, the key may be distributed (encrypted) among a set of entities, if these entities should have access to the line only together. The solution proposed allows also for privacy in the access to a line: this was obtained with an exclusion/elusion property of the method. That is, the data to be used for accessing a line is encrypted in a way that it is impossible, if not in possession of the decrypting key, to decide if the data is useful or not for disclosing the log line content. The consequence of this is that no one is able to decide whether an auditor has access or not to a line (except the auditor itself).

The proposed method is efficient in the sense that uses only one key to encrypt a line, and distributes this key to the auditors with the modes previously discussed. Nonetheless this key changes for each line, leaving a fine granularity in giving or not the possibility to various auditors to access a log line content.

The method is especially suitable to scenarios where a log entry size is considerable.

References

[1] F. Bergadano, D. Cavagnino, and B. Crispo, "Chained stream authentication," in *Proceedings of*

Selected Areas in Cryptography 2000, LNCS 2012, pp. 144-157, Springer-Verlag, 2000.

[2] F. Bergadano, D. Cavagnino, and L. Egidi, "Partially sighted signatures on large documents," in *Proceedings of International Network Conference*, Sherwell Conference Centre, University of Plymouth, UK, pp. 373-380, 2002.

[3] F. Bergadano, D. Cavagnino, and P. A. Nesta, "Certification of web access statistics," in *Proceedings of e-2001, E-work and E-commerce*, vol. 1, IOS Press, pp. 326-332, Oct. 2001.

[4] G. R. Blakley, "Safeguarding cryptographic keys," in *Proceedings of AFIPS, NCC*, vol. 48, pp. 313-317, Arlington, Va., Jun. 1979.

[5] M. Bellare and B. S. Yee, *Forward Integrity for Secure Audit Logs*, Technical report, UC at San Diego, Dept. of Computer Science and Engineering, Nov. 1997.

[6] C. N. Chong, Z. Peng, and P. H. Hartel, "Secure audit logging with tamper-resistant hardware," in *18th IFIP TC11 International Conference on Information Security (IFIPSEC), Security and Privacy in the Age of Uncertainty*, pp. 73-84, 2003.

[7] L. Lamport, "Password authentication with insecure communication," *Communications of the ACM*, vol. 24, no. 11, no. 24, pp. 770-772, 1981.

[8] A. J. Menezes, P. C. van Oorschot, and S. A. Vanstone, *Handbook of Applied Cryptography*, CRC Press, 1996.

[9] National Institute of Standards and Technologies, NIST FIPS PUB 197, *Advanced Encryption Standard (AES)*, U.S. Department of Commerce, Nov. 2001.

[10] National Institute of Standards and Technologies, NIST FIPS PUB 186, *Digital Signature Standard*, U.S. Department of Commerce, May. 1994.

[11] National Institute of Standards and Technology, NIST FIPS PUB 180-1, *Secure Hash Standard*, U.S. Department of Commerce, Apr. 1995.

[12] R. Rivest, A. Shamir, and L. Adleman, "A method for obtaining digital signatures and public-key cryptosystems," *Communications of the ACM*, vol. 21, pp. 120-126, 1978.

[13] M. Ruffin, *A Survey of Logging Uses*, Technical Report, no. FIDE2-94-82, Dept. of Computer Science, University of Glasgow, Glasgow, Scotland, Feb. 1995.

[14] A. Shamir, "How to share a secret," *Communications of the ACM*, vol. 22, pp. 612- 613, 1979.

[15] B. Schneier and J. Kelsey, "Cryptographic support for secure logs on untrusted machines," in *The 7th USENIX Security Symposium Proceedings*, USENIX Press, pp. 53-62, Jan. 1998.

[16] B. Schneier and J. Kelsey, "Secure audit logs to support computer forensics," *ACM Transaction on Information and System Security*, vol. 2, issue 2, May. 1999.

[17] J. Kelsey and B. Schneier, "Minimizing bandwidth for remote access to cryptographically protected audit logs," in *Web proceedings of the 2nd International Workshop on Recent Advances in Intrusion Detection (RAID'99)*, 1999.

[18] B. Schneier and J. Kelsey, *Event Auditing System*, US Patent #5,978,475, Nov. 2, 1999.

[19] B. R. Waters, D. Balfanz, G. Durfee and D. K. Smetters, "Building an encrypted and searchable audit log," in *The 11th Annual Network and Distributed System Security Symposium)*, Feb. 2004.

Francesco Bergadano received his Ph.D. in Computer Science from the University of Torino in 1991. He is Full Professor at the Computer Science Department of the University of Torino. He has authored several papers in international journals and conferences. His main research interests are security and its applications, in particular World Wide Web security, system security, public key infrastructure and user identification.

Davide Cavagnino received his laurea degree in 1992 and his Ph.D. in 1998, both in Computer Science. Presently he is a researcher in the University of Torino. His research interests are network and security protocols, and their applications.

Paolo Dal Checco has received his Laurea degree in Computer Science in 2002 and completed his Ph. D. studies in Computer Science in 2005. He is currently a Short Term Researcher at the Department of Computer Science, University of Turin. His research interests are web analytics, security and networking.

Pasquale Andrea Nesta received his Laurea degree in Computer Science in 2000 from the University of Turin. He has been a Ph.D. student at the Computer Science Department of the University of Torino working on Web technology and security.

Michele Miraglia has received his Laurea degree in Computer Science in 2004. At the moment he is a Short Term Researcher at the Department of Computer Science, University of Turin. His research interests are web analytics, security and networking.

Pier Luigi Zaccone received a degree in Computer Science from the University of Torino, Italy, on June 1994. Since 1997 he has been working in an Information Security Unit at Telecom Italia Research Department. He has worked on Information Security Risk Analysis and System Management Security. He is currently leading a corporate project regarding non-repudiation for system management.