Computers & Security, 13 (1994) 573-580

Database authentication revisited

Thomas Hardjono[1,2], Yuliang Zheng[1] and Jennifer Seberry[1]

[1] *Centre for Computer Security Research, Department of Computer Science, University of Wollongong, Wollongong, NSW 2522, Australia*
[2] *Department of Computing & Information Systems, University of Western Sydney at Macarthur, Campbelltown, NSW 2560, Australia*

Database authentication via cryptographic checksums represents an important approach to achieving an affordable safeguard of the integrity of data in publicly accessible database systems against illegal manipulations. This paper revisits the issue of database integrity and offers a new method of safeguarding the authenticity of data in database systems. The method is based on the recent development of pseudo-random function families and sibling intractable function families, rather than on the traditional use of cryptosystems. The database authentication scheme can be applied to records or fields.

The advantage of the scheme lies in the fact that each record can be associated with one checksum, while each data element in the record can be verified using the checksum independently of the other data elements in the record. The security of the scheme depends on the difficulty of predicting the outputs of pseudo-random functions and on inverting the sibling intractable function family. The same approach can also be applied to the generation of encipherment keys for databases.

Keywords: Information security, Database security, Database integrity, Authentication, Cryptography, Database systems, Sibling intractable function family.

1. Introduction

The problem of providing integrity to data stored in database systems has been a subject of interest among researchers for a number of years. In many cases the need to maintain the integrity of the data carried a higher priority than the need for information secrecy. Examples of such situations range from public medical information to statistical results from public events (e.g. elections) which are publicly readable, but which can be modified only by authorized users.

The most common method of ensuring integrity of data in a database is to use cryptographic *checksums*. A checksum is typically calculated on a piece of data (such as a database field entry or a record) as a function of some secret parameter which is available only to the authorized users.

The notion of a cryptographic checksum has been embodied in the past within the concept of a database *filter* which is to be located between the user and the database management system. From the point of view of the development of database security technology, the concept of a filter was perhaps one of the earliest to appear due to its simplicity. Given a database system to be protected, it was only natural to think initially of an intermediary between the user and the database system, in the form of a filter that simply screens out data according to some policy for labelling data.

One of the earliest realizations of the idea of a filter was the *integrity lock* approach, which was suggested initially by the US Air Force Summer Study on Multilevel Data Management Security in 1982 [1]. The notion of a 'spray paint' to label elements in

0167-4048/94/$7.00

the database system was also suggested by the study. The integrity lock approach applied a checksum function to the contents of each record, and maintained this checksum for each record to detect illegal tampering by opponents who by-passed the filter.

Ideally, the checksum should be a cryptographic hash function or encryption algorithm which is resistant to attacks based on cryptanalysis. The checksum is calculated whenever data is to be stored in the database system, and it is recomputed and compared with the stored checksum to detect illegal changes since the last modification of the data. The data in the records are not encrypted, to allow record processing by the database system, and the correct labelling of data remains the task of the filter. These checksums provide only error detection, not error correction.

The use of checksums for data in database records has received attention, notably in the works of Denning [2–4] and Graubart *et al.* [5–7]. The work by Denning in [2] is significant because it identifies the granularity of the data to be protected, namely whole records, whole attributes or individual data elements.

The other major work on the integrity lock approach was by Graubart [5], where it was applied to a commercial 'off-the-shelf' database management system. The components of the integrity lock design in [5] are the *Untrusted Front End* (UTFE), the *Trusted Front End* (TFE) and the untrusted database management system. The UTFE performs query parsing and the formatting of output to the user. The TFE performs tasks such as user authentication, tuple formatting, projections of data, and the calculations and verification of the checksums. The untrusted database system performs the usual tasks of record searching, tuple selection, insertion and deletion, and also database reconfiguration. The tuple in the database is left as plaintext for performance reasons, while the label and checksum are encrypted. As expected, the use of encryption expands the storage requirements of the database. The implementation of the integrity lock design was done on the MISTRESS database management system running on the Unix operating system [7]. A description of the operating system support environment for the integrity lock approach is given by Graubart and Kramer [6].

In this paper we follow the direction taken by Denning [2] and Graubart [5] in the use of cryptographic techniques to achieve record authentication. However, unlike these approaches which use symmetric or asymmetric cryptosystems to generate a checksum, our approach is based on the application of the concept of the sibling intractable function family (SIFF) which was first introduced by the work in [8].

In the previous approaches based on the use of cryptosystems a choice had to be made between a checksum for the entire record and a checksum for each data element in the record. The first method was advantageous in terms of space requirements, but resulted in the need to involve every data element in the record when the intention was to authenticate only one data element. The second method remedied this difficulty by creating a separate checksum for each data element in the record. In this way each data element could be authenticated independently of the other data elements, but the space requirements would be more than those for the first method. However, with today's rapidly decreasing cost of secondary storage the second method is becoming less intolerable.

Our approach in the checksum calculation for plaintext (or enciphered) records is based on having a single checksum for each record. However, our approach allows each data element in the record to be authenticated independently of the others using the same record checksum. Another advantage lies in the flexibility of placing the description of the instances of SIFF associated with each record in the same storage as the records. This removes the need to have specialized secure storage which, in general, is several magnitudes higher in cost than the

ordinary secondary storage media. Finally, our approach allows the authentication of data elements without necessarily needing any secret cryptographic information. This compares favourably to Denning's approach whereby a secret encryption key must be used before any data element can be authenticated.

The organization of this paper is as follows. In Section 2 the necessary definitions of the sibling intractable function family and of the pseudo-random function family will be presented. This is followed by the use of the sibling intractable function family for record authentication in Section 3, and its further use for the generation of cryptographic keys in the case of the encipherment of the database in Section 4. The paper is closed by some remarks and conclusion in Section 5.

2. Background in cryptography

This section introduces two basic constructs for our database authentication scheme, namely, *pseudo-random function families* and *sibling intractable function families* (SIFF).

Denote by $\mathcal{N}$ the set of all positive integers, n a security parameter, Σ the alphabet $\{0,1\}$ and $\#S$ the number of elements in a set S. By $x \in_R S$ we mean that x is chosen randomly and uniformly from the set S. The composition of two functions f and g is defined as $f \circ g(x) = f(g(x))$. Throughout the paper l and m will be used to denote polynomials from $\mathcal{N}$ to $\mathcal{N}$.

Let $F = \{F_n | n \in \mathcal{N}\}$ be an infinite family of functions, where $F_n = \{f | f: \Sigma^{l(n)} \to \Sigma^{m(n)}\}$. We call F a function family mapping $l(n)$-bit input to $m(n)$-bit output string. F is *polynomial time computable* if there is a polynomial time algorithm (in n) computing all $f \in F$, and *samplable* if there is a probabilistic polynomial time algorithm that, on input $n \in \mathcal{N}$, outputs uniformly at random a description of $f \in F_n$.

Now, we introduce the definition of pseudo-random functions [9] which will be applied in Section 3. Intuitively, $F = \{F_n | n \in \mathcal{N}\}$ is a pseudo-random function family if, to a probabilistic polynomial time algorithm, the output of a function f chosen randomly and uniformly from F_n, whose description is unknown to the algorithm, appears to be totally uncorrelated to the input of f, even if the algorithm can choose input for f. The formal definition is described in terms of *(uniform) statistical tests for functions*. A (uniform) statistical test for functions is a probabilistic polynomial time algorithm A that, given n as input and access to an oracle $\mathcal{O}_f$ for a function $f: \Sigma^{l(n)} \to \Sigma^{m(n)}$, outputs a bit 0 or 1. A can query the oracle only by writing on a special tape some $y \in \Sigma^{l(n)}$ and will read the oracle answer $f(y)$ on a separate answer-tape. The oracle prints its answer in one step.

Definition 1 *Let* $F = \{F_n | n \in \mathcal{N}\}$ *be an infinite family of functions, where* $F_n = \{f | f: \Sigma^{l(n)} \to \Sigma^{m(n)}\}$. *Assume that* F *is both polynomial time computable and samplable.* F *is a* pseudo-random function family *iff, for any statistical test* A, *for any polynomial* Q, *and for all sufficiently large* n,

$$|p_n^f - p_n^r| < 1/Q(n)$$

where p_n^f *denotes the probability that* A *outputs 1 on input* n *and access to an oracle* $\mathcal{O}_f$ *for* $f \in_R F_n$ *and* p_n^r *the probability that* A *outputs 1 on input* n *and access to an oracle* $\mathcal{O}_r$ *for a function* r *chosen randomly and uniformly from the set of all functions from* $\Sigma^{l(n)}$ *to* $\Sigma^{m(n)}$. *The probabilities are computed over all the possible choices of* f, r *and the internal coin tosses of* A.

In [9], it has been shown that pseudo-random function families can be constructed from any pseudo-random string generators. By the result of [10, 11], the existence of any one-way functions is sufficient for the construction of pseudo-random function families.

The following definition of the *collision accessibility property* is presented because of its importance in the definition of sibling intractable function families.

Definition 2. *Let* $U = U_n U_n$ *be a family of functions that is polynomial time computable, samplable, and maps*

l(n)-*bit input into* m(n)-*bit output strings. Let* k *be a fixed positive integer.* U *has the* collision accessibility property *if, for all* n *and for all* $1 \le j \le k$, *given any set* $X = \{x_1, x_2, \ldots, x_j\}$ *of* j *initial strings in* $\Sigma^{l(n)}$, *it is possible in probabilistic polynomial time to select randomly and uniformly functions from* U_n^X, *where* $U_n^X \subset U_n$ *is the set of all functions in* U_n *that map* $x_1, x_2, \ldots,$ *and* x_j *to the same strings in* $\Sigma^{m(n)}$.

Now we are ready to introduce the notion of the sibling intractable function family. Let $k = k(n)$ be a polynomial with $k(n) \ge 1$ and $H = \{H_n | n \in \mathcal{N}\}$, where $H_n = \{h | h: \Sigma^{l(n)} \to \Sigma^{m(n)}\}$, be an infinite family of functions that is polynomial time computable, samplable and has the collision accessibility property. Also let $X = \{x_1, x_2, \ldots, x_i\}$ be a set of i initial strings in $\Sigma^{l(n)}$, where $1 \le i \le k$, and h be a function in H_n that maps $x_1, x_2, \ldots, x_i$ to the same string. Let F, called a *sibling finder*, be a probabilistic polynomial time algorithm that, on input X and h, outputs either "?" ("I cannot find") or a string $x' \in \Sigma^{l(n)}$ such that $x' \notin X$ and $h(x') = h(x_1) = h(x_2) = \cdots = h(x_i)$. Informally, H is a k-*sibling intractable function family*, or k-SIFF for short, if, for any $1 \le i \le k$, for any sibling finder F, the probability that F outputs an x' is negligible. More precisely:

Definition 3. *Let* $k = k(n)$ *be a polynomial with* $k(n) \ge 1$ *and* $H = \{H_n | n \in \mathcal{N}\}$ *be a family of functions that is polynomial time computable, samplable, has the collision accessibility property and maps* l(n)-*bit input into* m(n)-*bit output strings. Let* $X = \{x_1, x_2, \ldots, x_i\}$ *be any set of* i *initial strings, where* $1 \le i \le k$. H *is a* k-sibling intractable function family, *or simply* k-*SIFF, if, for each* $1 \le i \le k$, *for each sibling finder* F, *for each polynomial* Q, *and for all sufficiently large* n,

$$Pr\{F(X, h) \neq ?\} < 1/Q(n)$$

where h *is chosen randomly and uniformly from* $H_n^X \subset H_n$, *the set of all functions in* H_n *that map* $x_1, x_2, \ldots,$ *and* x_i *to the same strings in* $\Sigma^{m(n)}$, *and the probability* $Pr\{F(X, h) \neq ?\}$ *is computed over* H_n^X *and the sample space of all finite strings of coin flips that* F *could have tossed.*

In [8] an explicit construction of SIFF from any one-way function was given. The reader is directed to [8] for other applications of SIFF.

3. Using SIFF to authenticate records

Following [2], we will denote record i as having a record identifier R_i and field (or attribute) j as having the field identifier F_j. The actual value of field j in record i is denoted as X_{ij}. For simplicity, we assume that the security labels will be applied at the record level, incorporated into the record identifier R_i. Hence, R_i should be unique for each record and is assumed to contain enough information to determine the security classification of the record. If security classification is to be applied at the data element level, then it is assumed that for each field F_j an additional field F_j' exists in the record which contains the security classification of data element X_{ij} (see [2, 3]). We also assume that a key K_{db} for the whole database exists, and is stored in a tamper-free condition. In the remainder of this paper we will use the term *trusted party* (TP) to denote a trusted agent which holds the secret cryptographic information necessary for the checksum generation and verification, and for the encipherment and decipherment of data in the records. The trusted party can be an intermediary between the user and the database, or it can be a separate function in the database system. However, we will not be concerned any further with the actual architecture of the system that incorporates the trusted party.

The use of SIFF to calculate checksums for fields provides an alternative method for data integrity. The calculation of record checksums using SIFF has the major advantage of field checksums, namely that it allows a single checksum value to be associated with each record yet allows each data item to be authenticated independently of other data items in the record.

3.1 Mathematical description

Consider an $(\alpha + 1)$-SIFF, where α is an integer denoting the number of fields in each record i

including the record identifier R_i and its security classification. We assume that every record follows this arrangement uniformly. The trusted party holds a secret information s_{db}. In addition, it generates the key K_{db} which is publicly readable in a tamper-free state. The public state of K_{db} allows any party, trusted or otherwise, to verify that the data in the database is authentic.

Let $H=\{H_n|n\in\mathcal{N}\}$ be an $(\alpha+1)$-SIFF mapping n-bit input to n-bit output strings. Furthermore, assume that $F=\{F_n|n\in\mathcal{N}\}$ is a pseudo-random function family, where $F_n=\{f_K|f_K:\Sigma^{l(n)}\to\Sigma^n, K\in\Sigma^n\}$ and each function $f_K\in F_n$ is specified by an n-bit string K. Here we must have that $|X_{ij}|+|R_i|+|F_j|\le l(n)$, and padding (such as in [2]) can be used.

For each record i we choose uniformly and randomly from H_n an instance of SIFF h_i such that:

$$h_i(f_{s_{db}}(R_i\|s_{db}))=h_i(f_{K_{db}}(X_{i1}\|R_i\|F_1))$$
$$=\cdots=h_i(f_{K_{db}}(X_{i\alpha}\|R_i\|F_\alpha))=S_i \qquad (1)$$

where S_i is a randomly chosen n-bit string, the checksum for record i. Here '$\|$' denotes concatenation. In this way the data element $X_{ij}(1\le j\le\alpha)$ can be authenticated when

$$h_i(f_{K_{db}}(X_{ij}\|R_i\|F_j))=S_i$$

is satisfied. This process is shown in Fig. 1.

3.2 Checking instances of SIFF

Ideally, the description of the instances of SIFF h_i should be placed in secure storage within the security perimeter of the trusted party. However, due to the large size of the description of these instances of SIFF a more manageable approach would be to store them in a 'shadow' database or together with the actual data in the database. In any case, the description of the instances h_i of SIFF can be placed in a publicly readable storage since any modifications to them or to the checksums S_i can be detected through the use of s_{db}. That is, if h_i of record i is modified illegally into h_i', then using the false h_i' will not yield the correct checksum value, as shown in the following:

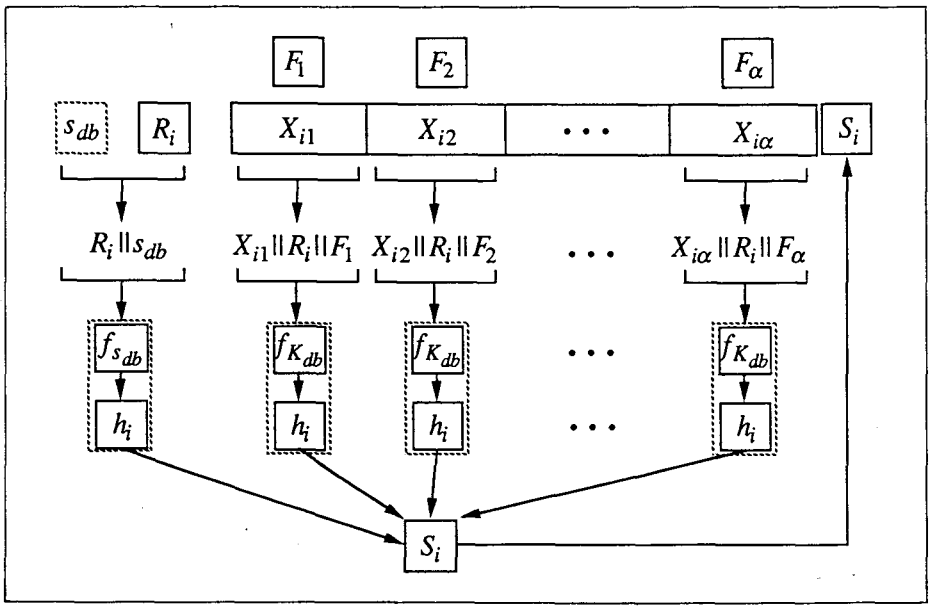

Fig. 1. Using SIFF for database authentication.

$$h'_i(f_{s_{db}}(R_i \| s_{db})) \neq S_i$$

due to the fact that s_{db} is secret. This is true even if the false h'_i yields the correct checksum for all the fields $F_1, F_2, \ldots, F_\alpha$:

$$h'_i(f_{K_{db}}(X_{i1} \| R_i \| F_1)) = h'_i(f_{K_{db}}(X_{i2} \| R_i \| F_2))$$
$$= \ldots = h'_i(f_{K_{db}}(X_{i\alpha} \| R_i \| F_\alpha)) = S_i$$

If S_i is modified illegally into S'_i then the output of h_i will not match the false S'_i. This is shown in the following:

$$h_i(f_{s_{db}}(R_i \| s_{db})) \neq S'_i$$

Suppose that both h_i and S_i are illegally modified into h'_i and S'_i respectively; then, due to the participation of the secret s_{db}, we still have that

$$h'_i(f_{s_{db}}(R_i \| s_{db})) \neq S'_i$$

which indicates that the illegal modifications have occurred.

The sibling intractable property of SIFF allows the detection by the trusted party of any illegal modification to either or both of h_i and S_i. The secrecy of s_{db} is necessary to ensure that only the trusted party can create S_i through the selection of a suitable instance of SIFF h_i for record i.

3.3 Field authentication

The idea of maintaining a checksum for whole fields (or attributes) was also suggested by Denning [2, 3]. Similar to record checksums, an instance h_j of a $(\beta + 1)$-SIFF can be used for field checksums as follows:

$$h_j(f_{s_{db}}(F_j \| s_{db})) = h_j(f_{K_{db}}(X_{1j} \| R_1 \| F_j))$$
$$= \ldots = h_j(f_{K_{db}}(X_{\beta j} \| R_\beta \| F_j)) = S_j \qquad (2)$$

where β is the number of records in the database.

This approach, however, is impractical because the update of a value X_{ij} $(1 \leq i \leq \beta, 1 \leq j \leq \alpha)$ requires the involvement of all the field values $X_{1j}, X_{2j}, \ldots, X_{\beta j}$ in the recalculation of the checksum S_j. This approach is more suitable to be applied to fields which are rarely changed, such as an employee's name and birth date in the case of a company database.

3.4 Security of the database authentication scheme

The record authentication scheme in eq. (1) can be considered secure if it is computationally difficult for an illegal user (or a trojan horse) to find the secret value s_{db} even when the user knows K_{db}, h_i and S_i. We will assume that the computational power of an illegal user or a trojan horse is bounded by probabilistic polynomial time.

In more definite terms, we can say that for a database with a total of β records, each having a different instance h_i of SIFF associated with it, the record authentication scheme is secure if, for any data element X_{ij} in the database, for any polynomial Q, and for all sufficiently large n, the probability that an illegal user or a trojan horse can find s_{db} using K_{db} is less than $1/Q(n)$. This also holds true when every record i in the database is given a different key K_{db_i} $(i = 1, \ldots, \beta)$ which are all known to the illegal user or the trojan horse.

Now, the ability of an illegal user or a trojan horse to calculate s_{db} is equivalent to that of predicting outputs of a pseudo-random function. This is a contradiction to the definition of pseudo-random function families. In fact, the ability of an illegal user or a trojan horse to obtain even the input string $f_{s_{db}}(R_i \| s_{db})$ (or $f_{s_{db}}(F_j \| s_{db})$) represents a further contradiction. Such an ability implies that the user or the trojan horse is able to invert or to find a collision string for the sibling intractable function family, both cases of which have a negligible probability of happening. Hence, the database authentication scheme in eqs. (1) and (2) is secure provided that s_{db} remains secret.

4. Using SIFF to generate encipherment keys

Independent, but related to the issue of database authentication, is the issue of protection of the

database from illegal access. One possible method of preventing illegal access to records in the database is by way of encipherment techniques. In simple terms, this involves the encipherment of records in the database using a cryptosystem that requires an encipherment key as one of its parameters. The database then consists of enciphered records which can be read or updated only through the use of the decipherment key (which may be different, but related, to the encipherment key).

In the trusted party concept it is intended that a user should interact with the database through the trusted party, which on behalf of the user accesses the database, deciphers the retrieved records and presents the resulting plaintext records to the user. Records with fields of higher security classification than the user's security clearance can be filtered out, or the whole record can be suppressed from the user. Hence it is the task of a user to present some form of identification information to the trusted party, which then authenticates the user and retrieves the required data from the database depending on the user's security clearance.

The simplest use of a SIFF in this situation is for it to derive the decipherment key by using the user's secret key or password as input to the instance of SIFF. Assume that there are P users having the secret keys $K_{u1}, \ldots, K_{uP}$ respectively. Furthermore, assume that K_T is the secret key of the trusted party, h_{acc} is the instance of SIFF maintained as a secret by the trusted party, and K_{dec} is the decipherment key. The trusted party must choose uniformly and randomly from H_n an instance h_{acc} of a P-SIFF such that:

$$h_{acc}(f_{K_T}(K_T)) = h_{acc}(f_{K_T}(K_{u_1})) = \ldots = h_{acc}(f_{K_T}(K_{uP})) = K_{dec} \quad (3)$$

Note that here K_{dec} should never be visible or accessible to the users. Hence its derivation must be a guarded privilege of the trusted party. The enciphered records are then retrieved by the trusted party and deciphered using K_{dec} within the security bounds of the trusted party.

This simple idea using SIFF, as expressed in eq. (3), can be extended further to the multilevel case where users are grouped according to their security clearances, each group having access to a subset of the database depending on the security clearance.

5. Conclusion

An alternative approach to the authentication of databases has been the topic of research in this paper. The approach is realized in a scheme which is based on pseudo-random functions and the sibling intractable function family (SIFF). The security of the scheme has been shown to be equivalent to predicting the output of pseudo-random functions and inverting the sibling intractable function family, both of which have a small probability of occurring.

The scheme, which has been discussed in the context of the concept of the trusted party that acts as an intermediary between the user and the database, allows each record to be associated with one checksum which can be used to verify the authenticity of one data element within the record independently from other data elements. The scheme also has the advantage that it requires only a small amount of information to be maintained secret, which is something affordable for the trusted party. Related to this is the advantage that the description of the instances of SIFF can be placed in the same storage area as the records of the database and their associated checksums. In this way no secure storage needs to be assigned for the maintenance of the instances of SIFF and the checksums. The scheme is also suitable for the generation of encipherment (and decipherment) keys in enciphered databases to allow only legal access to the database.

Acknowledgements

This work was supported in part by TELECOM Australia under the contract number 7027 and by