8

A SECRET SPLITTING METHOD FOR ASSURING THE CONFIDENTIALITY OF ELECTRONIC RECORDS

Andrew Po-Jung Ho

Abstract Large databases are increasingly used to manage sensitive data; however, they are vulnerable to abuse by insiders and intruders. We describe a novel method that protects electronic records from intruders and even the most powerful of insiders, the system administrators. Confidential data are partitioned and distributed after asymmetric encryption for management over multiple databases. Data splitting and distributed management prevent unauthorized and covert access by any single party. Confidential systems using this design can work synergistically with existing security measures and are suitable for health, genomic, and financial records.

Keywords: Asymmetric cryptography, confidential, distributed database, secret splitting, separation of duty

Introduction

Knowledge is power and large databases together with new data mining technologies are increasingly used to advance personal services and scientific discoveries. The public, however, remains distrustful of large databases especially because of privacy concerns. This distrust delays the implementation of promising technology and leads to economic and social costs. A recent example is the opposition to Iceland's national health and genomic database [1, 4].

According to a 1993 U.S. Congress Office of Technology Assessment report, the greatest threat to privacy of stored records is by persons within the system [15]. This assessment is supported by publicly known incidents of abuse by insiders [6, 5, 16]. How to allow workers to perform their tasks while preventing them from abusing these privileges is the most urgent subject for

The original version of this chapter was revised: The copyright line was incorrect. This has been corrected. The Erratum to this chapter is available at DOI: 10.1007/978-0-387-35508-5_22

V. Atluri et al. (eds.), *Research Advances in Database and Information Systems Security*

the managers of these databases. It is difficult to allow insiders to have the privileges needed to perform their tasks while preventing the abuse of these same powers. This is especially true for the powerful insiders, such as system administrators, who are typically able to retrieve confidential information without detection. Although the system administrators are not commonly perceived as the group that is most likely to perpetrate abuse, this vulnerability will increasingly gain significance as the size and value of the database increase, and other vulnerabilities are better addressed: Powerful insiders will increasingly become the weakest link. An important and related threat comes from intruders who obtain the system administrator's privileges. Methods that can prevent the abuse of the system administrators' privileges will also function as the last line of defense against the intruders.

Current security measures such as access control, trail audit [9], and anonymizing/encryption offer little protection against powerful insiders. Knowledgeable insiders and intruders are commonly able to bypass the access control and trail audit measures. Insiders who have access to the codebook, encryption algorithm, or the un-anonymized/un-encrypted data entering or leaving the system can defeat the anonymizing/encryption systems. For example, if a codebook is used to replace the names of the subjects with pseudonyms, the procedure can be reversed to determine the names of the subjects. If cryptographic algorithms (such as one-way hash or asymmetric algorithms) are used, insiders with access to the cryptographic algorithms can re-identify the records through a trial encryption attack even if the decryption keys are not accessible to the attacker [2, 3]. Furthermore, insiders typically have access to un-anonymized or un-encrypted data when the system performs the anonymizing and encryption procedure. This is a weakness even when the encryption algorithm is secured by tamper-proof hardware. It is especially significant when a single coding unit is used since the insiders at that unit have access to all secrets entering the system. An example of a system with this vulnerability is the German cancer registry that uses a central coding unit to assign pseudonyms to patient records [7, 10]. It implements separation of duty between the coding unit and three other units. Insiders at the coding unit have access to all un-anonymized records entering this system. Similarly, an U.S. Air Force design that relies on the distribution of sensitive data over multiple units suffers the same vulnerability [?]. The Air Force design uses a front-end unit to decompose queries into sub-queries, which are sent to individual back-end databases. The back-end databases return their output to the front-end unit where they are recombined and transmitted to the user. While this design prevents the insiders at the back-end units from discovering the secrets, the insiders at the front-end unit have access to the complete contents during both the input and output phases of the transaction.

An alternative to a central anonymizing/encryption unit is a system where the users perform the anonymizing and encryption. A useful scheme that relies on distributed anonymizing or encryption must solve the problem of key management. A useful multi-user system must also provide centralized access control and decryption key management. At the same time, the powerful insiders charged with the access control and key management must not be able to abuse their privileges. To our knowledge, no existing design can achieve these goals.

Another limitation of anonymizing/encryption databases is that obviously identifiable personal data such as names, addresses, and phone numbers cannot be maintained by the system. More importantly, many other potentially identifiable data also cannot be maintained because they can be used to infer the identity of records [14]. In some applications, the absence of these data fields is acceptable; however, the "scrubbing" of all potentially identifying data such as date and place of birth inevitably reduces the usefulness of the system. For databases that maintain longitudinal health records or genetic information, it is effectively impossible to construct a sufficiently anonymous database [1].

1. BASIC DESIGN

We describe a novel approach that uses data partitioning, asymmetric encryption, and distributed processing to limit the power of each powerful insider. We show that this method can be used to implement centralized access control and maintain identifiable personal and sensitive data within an enforceable check-and-balance system of administration.

Unlike the German cancer registry and the Air Force database described previously, each database unit in our system has access to only the data needed to perform its assigned storage and processing tasks; each unit does not have access to any other part of the submitted data. This strict, "need-to-know" data isolation is maintained by cryptography throughout the data handling procedure. Separation of duty is achieved through separate administration of each database unit. By partitioning the sensitive data and distributing their storage and retrieval over different databases, sensitive information is protected from improper access by any one powerful insider. Furthermore, once fragmented into arbitrarily small units and separated from the rest of the confidential record, all personal identifiers can be managed as any other elements of information.

Figure 8.1 shows an example of how the secret splitting method can be applied to a database containing personally identifiable data. In Step 1, the User inputs a query into a client program. The Client Program collects information to identify and authenticate the user (subject), the client program, the record

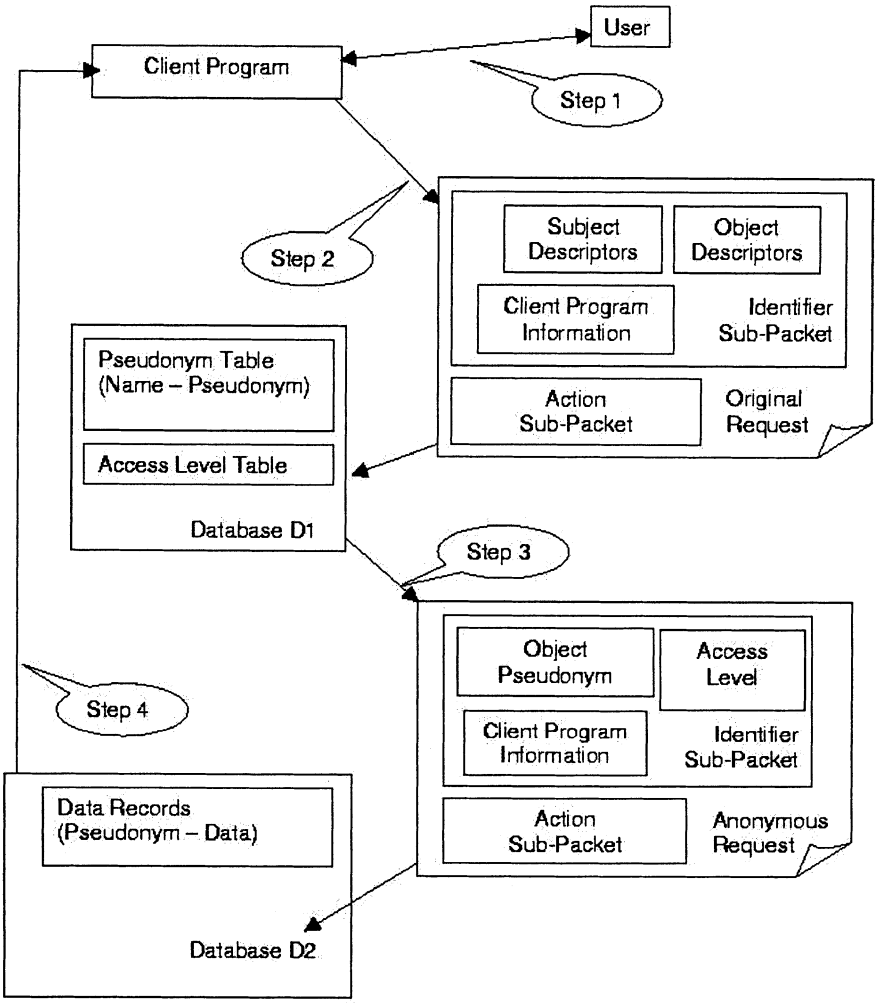

Figure 8.1 Basic design.

of interest (object), and the action to be performed on the record. The Client Program divides the data to create 1) Identifier sub-packet that conveys information needed to identify and authenticate the subject, object, and the client program; 2) Action sub-packet that contains the description of the operation(s) to be performed including any new data to be added. In Step 2, the Client Program first encrypts the Action sub-packet with a code that is decodable only by Database D2; then, the encrypted Action sub-packet is combined with the Iden-

tifier sub-packet and encrypted with a code that is decodable only by Database D1. The output is called the Original Request. In systems where there are more than 2 database units, additional sub-packets can be the nested within the Action sub-packet. A variety of encryption algorithms can be used so long as the sub-packets can only be decoded by each of the intended databases. For example, a symmetric algorithm can first be used to encrypt the sub-packet and then an asymmetric algorithm used to encrypt the symmetric key (using the public keys of Database D1 and D2 as appropriate). Cryptographic algorithms and key exchange protocols have been extensively reviewed [12].

The Client Program sends the Original Request to Database D1 where Database D1 (D1) decodes the Identifier sub-packet. D1 does not have the decryption key necessary to decode the encrypted Action sub-packet; therefore, the content of the Action sub-packet is inaccessible to the system administrator of D1. D1 uses the information from the Identifier sub-packet to (1) authenticate the subject, (2) determine the access level of the subject in relation to the object, and (3) determine the pseudonym of the object from a Pseudonym Table. The pseudonym is a code that uniquely represents the object. In Step 3, D1 combines the access level, pseudonym of the object, client program identifier, and the encrypted Action sub-packet to form the Anonymous Request. Anonymous Request is encrypted such that it is only decodable by Database D2.

D1 sends the Anonymous Request to Database D2 where Database D2 (D2) decodes it. If the subject's access level permits the execution of the operation specified in the Action sub-packet on the specified object, D2 performs the operation and returns the results, if any, to the Client Program. The results can be encrypted for transmission to the Client Program using the public key of the Client Program, the User, or a special Session Key transmitted as part of the Action sub-packet.

In this potential application, Database D2 does not contain demographics, personal, or other potentially personally identifiable information; these identifiers are stored in Database D1. The fragmentation of data diminishes the likelihood of a successful statistical inference attack on D2 by unauthorized insiders while permitting retrieval of the identifying information by authorized users. Thus, although D2 contains the list of all patients with the diagnosis of AIDS, even the system administrator of D2 cannot discover the names of these patients. The names of the patients can only be revealed when the pseudonyms are linked to the names stored in D1.

2. ACCESS TRAIL AUDITING

Each database unit maintains an activity log to discourage system administrators from sending unauthorized queries to the system to try to decode the pseudonym or to perform unauthorized data access. The activity log at D1 records the time of transmission from D1 to D2 and the object of the query. The activity log at D2 records the time and object of the query and the destination to which the results were sent. When there is a question as to the integrity of the systems, a third party auditor can compare the two logs to determine whether there are irregularities. The preferred procedure for a third party audit utilizes a procedure to transform these logs to protect the confidential identity of the subject-object pairs.

Periodic reports can also be generated by D1 to disclose the identity of all subjects who accessed a given object. These reports can be sent directly to the owners of the objects or to an entity given the responsibility of oversight. Inappropriate access of records can thereby be identified in a timely manner and all participants held accountable for their activities.

3. DESIGN VARIATIONS

The basic design in Figure 8.1 shows the data flow from the Client Program to a first database D1, onward to D2, and then back to the Client Program. Other structural organizations are possible; for example, additional databases can be connected in multiple parallel and sequential layers to further vertically and/or horizontally partition the data. Various combinations and permutations of the design can be used to implement confidential systems with performance, data integrity, scalability, and security trade-offs. The issue of data integrity, for example, can be addressed with a design using redundant paths and subsets of databases that manage error correction codes. Although the protection of personal information is an important application of this technology, the same protection method can be applied to non-personal data.

4. DISCUSSION

As the reliability and transfer rate of networks and cryptographic engines continue to increase, it will be increasingly feasible to trade performance for increased security and confidentiality. This data management approach allows the secure and confidential storage of electronic data with little additional performance overhead since encryption during transmission is already a necessary practice for transmission security. The performance overhead is approximately one additional encryption-decryption cycle in the most basic design and any additional setup time related to the use of two rather than one encryp-

tion/decryption procedures. As hardware-based cryptographic engines continue to improve, this performance penalty will continue to diminish. The storage overhead is also manageable since it is directly proportional to the number of Subjects and Objects.

The fragmentation of data into partitions has long been used to protect sensitive information. Fundamentally, cryptographic algorithms can be viewed as methods of fragmentation (and recombination) such that the original content can be recovered only if the fragments are recombined. The data fragments can be in the form of a decryption key, share of secret, or cryptotext. Existing schemes uniformly aim to produce data fragments that confer no information when taken individually [12]. Our approach uses data fragments that confer partial information when taken individually. When these fragments are distributed to multiple units, it is possible to perform local processing using the partial information. The fragmentation process prevents successful inference attack and precludes the need for centralized decryption key management. In this way, a powerful insider who controls any given fragment is able to perform the assigned tasks but unable to infer information about any other fragment.

To reduce the likelihood of improper access to the un-fragmented data, it is desirable to split the data as close to the time and place of their creation as possible. Each Client Program (Figure 8.1) splits sensitive data into fragments (sub-packets) where each fragment is encrypted such that only the intended database unit(s) can decode it. The data partitioning is achieved through asymmetric cryptography to eliminate the need for a secure channel for key exchange. Our approach differs from traditional distributed systems where the distributed nature of the architecture is hidden from the client program. With our design, the client program uses destination-specific information to encrypt and distribute the data fragments. Consequently, the client program in our system provides location and transaction transparency to the users of the system.

This work makes use of a simple but fundamental principle - asymmetric cryptographic algorithms allow the confidential delivery of data through insecure intermediaries without requiring a separate secure channel for key exchange. In this application, a portion of each query is "tunneled" through the set of first databases to reach the destination database without disclosing its confidential contents. In contrast with other "tunneling" protocols (such as in virtual private networking), our system requires the intermediate node(s) to be equal partners and share in the processing of the transaction. The use of intermediate nodes as data processing partners allows the system to separately protect the fragments and the links that connect them to each other across databases: a system can be designed where a database unit manages only the links between two other database units.

Our secret-splitting method differs from secret sharing schemes where each share of secret contains no information in isolation, and therefore each share of data cannot be used to respond to a query until the shares are fully reassembled [11]. In contrast, our approach permits the distributed and independent processing of the subqueries at each database unit without the need for reassembly. In our system, re-assembly, as the original disassembly, occurs in the client program.

The ability to prevent abuse by powerful insiders and intruders can be instrumental in enabling the implementation of large-scale data collection and information-based applications. The secret-splitting method is potentially suited to the protection of sensitive data such as biometrics, genetic, health, and financial records. The major vulnerability of this approach comes from collusion between powerful insiders of adjacent database units. Even collusion between a user and a powerful insider will only compromise the subset of objects accessible to that user. Further analysis and modeling will be needed to characterize the relationship between variations in the design and vulnerability to collusion between units.

References

[1] Anderson, R. (1998). The Decode proposal for an Icelandic health database. http://www.cl.cam.ac.uk/users/rja14/iceland/iceland.html.

[2] Bouzelat, M., Quantin, C. and Dusserre, L. (1996). Extraction and anonymity protocol of medical file. *Proceedings of the AMIA Annual Fall Symposium*, pp. 323–327.

[3] Dusserre, L. Quantin, C. and Bouzelat, M. (1995). A one way public key cryptosystem for the linkage of nominal files in epidemiological studies. *Medinfo*, **8(1)**, pp. 644–647.

[4] Enserink, M. (1998). Opponents criticize Iceland's database [news]. *Science*, **282(5390)**, p. 859.

[5] Ex-employee videotapes Stanford medical data security breach. *Inside Healthcare Computing*, May 27, 1996.

[6] How a 13-year-old breached HIS security, made phony AIDS calls. *Inside Healthcare Computing*, March 4, 1996.

[7] Michaelis, J., Miller, M., Pommerening, K. and Schmidtmann, I. (1995). A new concept to ensure data privacy and data security in cancer registries. *Medinfo*, **8(pt 1)**, pp. 661–665.

[8] O'Connor, J., Gray, J., McCollum, C. and Notargiacomo, L. (1992). *Research Directions in Database Security* (ed. T. Lunt), Springer-Verlag, pp. 41–62.

[9] Pangalos, G. (1996). Security of medical database systems for health care IT and security personnel. *Data Security for Health Care, Volume II: Technical Guidelines* (eds. The SEISMED Consortium), IOS Press, pp. 235–342.

[10] Pommerening, K., Miller, M., Schmidtmann, I. and Michaelis, J. (1996). Pseudonyms for cancer registries. *Methods Inf Med*, **35**, pp. 112–121.

[11] Rabin, M. (1989). Efficient dispersal of information for security, load balancing, and fault tolerance. *Journal of the ACM*, **36(2)**, pp. 335–348.

[12] Schneier, B. (1996). *Applied Cryptography, Second Edition*, John Wiley & Sons, pp. 461–482, pp. 561–596.

[13] Sweeney, L. (1996). Replacing personally-identifying information in medical records, the Scrub system. *Proceedings of the AMIA Annual Fall Symposium*, pp. 333–337.

[14] Sweeney, L. (1997). Guaranteeing anonymity when sharing medical data, the Datafly System. *Proceedings of the AMIA Annual Fall Symposium*, pp. 51–55.

[15] U.S. Congress, Office of Technology Assessment (1993). *Protecting Privacy in Computerized Medical Information*. Washington, DC: U.S. Government Printing Office.

[16] Zitner, A. (1997). Protecting the privacy of medical records. *The Boston Globe*.