3. Martin M, Lam MS (2008) Automatic generation of XSS and SQL injection attacks with goal-directed model checking. In: Seventeenth USENIX security symposium, San Jose
4. Rassadko N (2006) Policy classes and query rewriting algorithm for XML security views. In: Damiani E, Liu P (eds) Data and applications security. Lecture Notes in Computer Science, vol 4721. Springer, Heidelberg
5. Shields C (2002) What do we mean by network denial of service? In: Proceedings of the 2002 IEEE workshop on information assurance and security, West Point, 17–19 June 2002

HEC Acronym is Often Used for Hyper Elliptic Curves

►Hyperelliptic Curves

Heredity

►DNA

High Assurance Evaluation Methods

►Formal Methods in Certification and Evaluation

Higher Order Derivative Attack

►Cube Attack

Hippocratic Database

Tyrone Grandison[1], Kristen LeFevre[2]
[1]Intelligent Information Systems, IBM Services Research, Hawthorne, NY, USA
[2]Department of Electrical Engineering and Computer Science, University of Michigan, Ann Arbor, MI, USA

Synonyms

Privacy-aware database; Privacy-enabled database

Related Concepts

►Data Retention

Definition

A database system that has privacy (enablement) as its fundamental goal.

Background

The term "Hippocratic Database" (HDB for short) was coined in 2002 in a paper by Agrawal, Kiernan, Srikant, and Xu at the Very Large Databases (VLDB) conference [1]. They observed that technology trends, including the World Wide Web, were leading to a marked increase in electronic collection and storage of private and personal information. The HDB vision (Fig. 1) provided insight into a new class of data systems – one that required the redesign of current systems and that enabled the data system to automatically manage sensitive information without impeding information flow.

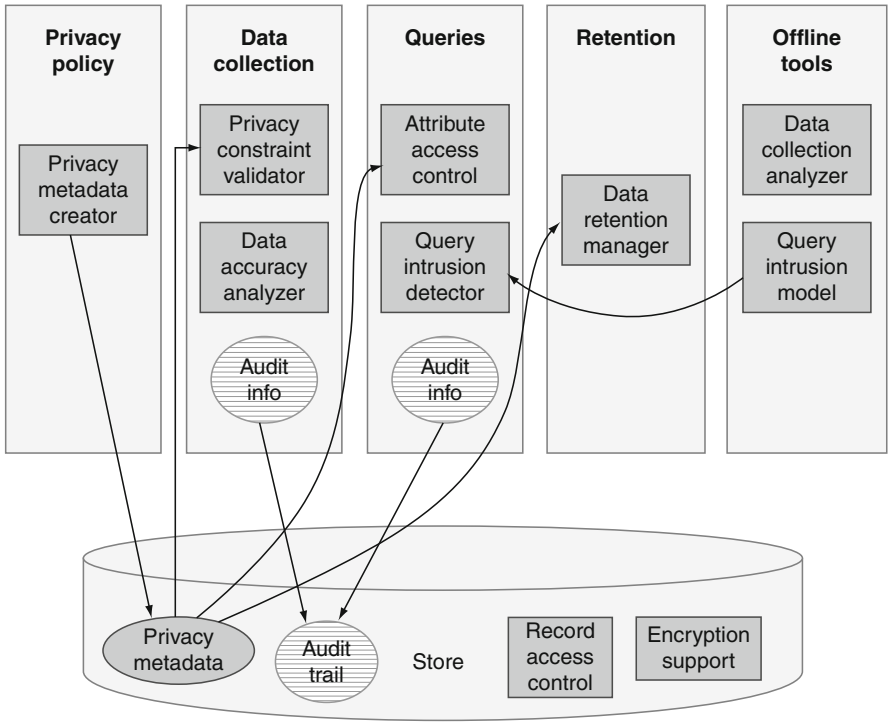
Theory

HDB is inspired by the privacy provision of the Hippocratic Oath, which states that:

> ... about whatever I may see or hear in treatment, or even without treatment, in the life of human beings – things that should not ever be blurted outside – I will remain silent, holding such things to be unutterable [2].

HDB was not intended to be a fixed technology set. Instead, it expressed a vision for a class of database systems, tools, and applications that are designed with the primary goal of storing and managing personal and private data. In [1], they outlined ten key principles that should be incorporated into the design of future database systems. The following represents our interpretation of these ten principles, and examples of instances where the principles have been applied.

1. *Purpose Specification*: When personal information is collected, the purpose(s) for which the data is collected should be permanently associated with that information. For example, if an individual's address is collected for the purposes of shipping an order, this information should be maintained by default (e.g., as a form of meta data). This principle has been applied, for example, in the development of purpose based access control [3, 5].
2. *Consent*: Similarly, the user should provide consent for each purpose for which her data is collected. For example, if the user's address is collected for the purpose of shipping, then additional consent should be obtained before using this information for another



Hippocratic Database. Fig. 1 The vision of a Hippocratic Database (Diagram from [1])

purpose. Several techniques for enabling database consent management have been proposed [6, 7]. This issue is likely to persist as the debate over enabling the *secondary use of data* gains prominence, especially in sectors like health care.

3. *Limited Collection*: The personal data collected should be the minimum necessary for the required tasks [8].
4. *Limited Use*: Similar to consent, personal data should only be used for the purpose for which it was collected. Ideas on enforcing limited use are emerging in the field. For example, Angela C. Duta and Ken Barker [9] provide an example of how it could be done for XML data.
5. *Limited Disclosure*: Personal information should not be communicated outside the database for purposes other than those for which there is consent. From a technical perspective, this is one of the more heavily studied principles. Interestingly, the idea of limited disclosure can be interpreted in two distinct ways. On one hand, it can be interpreted literally (i.e., as a form of access control [4, 5]). Alternatively, the idea of limited disclosure can also be equated with statistical disclosure prevention [10–12], in which personal data can be used for computing aggregate statistics, as long as the underlying data cannot be inferred.
6. *Limited Retention*: Personal data should not be retained beyond the period necessary for fulfilling the purpose(s) for which it was collected. From a technical perspective, securely deleting data from a database can be a nontrivial task [13, 14]; it is important to guarantee that the data is securely removed from all parts of the system.
7. *Accuracy*: Personal data stored in the database should be accurate and up to date. Individuals should have the opportunity to correct inaccuracies.
8. *Safety*: This principle is most closely connected to conventional notions of security. Essentially, data stored in the database should be protected from theft and other security vulnerabilities.
9. *Openness*: Related to accuracy and compliance, an individual should be able to access his or her personal data stored in the database. The idea of openness can also be interpreted to include the idea that an individual should be able to find out how her data has been used (e.g., [15–17]).
10. *Compliance*: An individual who contributes personal data to a database should be able to verify all of the above principles. From a regulatory perspective, an authorized auditor should also be able to verify compliance with legal regulations pertaining to data use [18], e.g., HIPAA, Sarbanes-Oxley, etc.

Applications

HDB was initially applied to SQL (Structured Query Language) statements in the context of relational databases. Further work has implemented HDB for non-relational systems and to DML (Data Manipulation Language) statements. The technology has been prototyped in the Health Care, Finance, Government, and Scientific Research domains.

Open Problems

1. Improved Policy Specification – For HDB controls to be completely effective, policies must accurately capture the data usage practices of enterprises and the preference and choices of individuals concerning the use and disclosure of their personal information. The policy language must be fine grained enough to allow enterprises to collect, use, and disclose the minimum necessary information to accomplish their intended purposes. It must also be simple enough that technically unsophisticated individuals can understand the consequences of their decisions to provide personal information.
2. Enforcement After Extraction – Current implementations of HDB are adept at limiting disclosure of information contained within the database, but does not exert any control or safeguards over information that is legitimately extracted and transferred outside of the database.
3. Filter and Deny Semantics – HDB policy enforcement uses query predicates to filter results in compliance with the applicable policy rules. The system transforms the query so that the database only returns information that is compliant with the user's authorization, the enterprise's privacy policy, and any individual choices. Prohibited values that are sought by the query are returned as null values. However, in some circumstances, this type of filtering may not be desirable because it may mislead the user into thinking that the prohibited values do not actually exist.
4. Query Intrusion Detection – It is desirable to have the ability to detect illegitimate access by comparing a query access pattern to the usual and expected access pattern for that particular user and purpose. This capability advances the HDB safety principle by preventing inappropriate access through legitimate channels.
5. Data Integrity – An HDB system should provide guarantees on the soundness of the data that it contains. Individuals should have access to view and verify the accuracy of their information. Data cleansing can be used to identify and correct erroneous data. Maintaining the provenance of information can also provide an indication of the reliability of the information.

Recommended Reading

1. Agrawal R, Kiernan J, Srikant R, Xu Y (2002) Hippocratic databases. In: Proceedings of the international conference on very large data bases (VLDB), Hong Kong, 2002
2. Von Staden H (trans) (1966) In a pure and holy way: personal and professional conduct in the hippocratic oath. J His Med Appl Sci 51:406–408
3. Byun J-W, Bertino E, Li N (2005) Purpose based access control for complex data for privacy protection. In: Proceedings of the ACM symposium on access control models and technologies. ACM press, New York, pp 102–110
4. Ghazinour K, Majedi M, Barker K (2009) A lattice-based privacy aware access control model. In: Proceedings of the 2009 international conference on computational science and engineering. IEEE Computer Society, Washington, DC, pp 154–159
5. Al-Fedaghi S (2007) Beyond purpose-based privacy access control. In: Proceedings of the 18th Australasian database conference, Ballarat, Australia, January 29–February 2, 2007
6. LeFevre K, Agrawal R, Ercegovac V, Ramakrishnan R, Xu Y, DeWitt D (2004) Limiting disclosure in Hippocratic Database. In: Proceedings of the international conference on very large data bases (VLDB), Toronto, 2004
7. Agrawal R, Bird P, Grandison T, Kiernan J, Logan S, Rjaibi W (2005) Extending relational database systems to automatically enforce privacy policies. In: Proceedings of the 21st international conference on data engineering, Tokyo, 2005
8. Crépin L, Vercouter L, Jaquenet F, Demazeau Y, Boissier O (2008) Hippocratic multi-agent systems. In: Proceedings of the 10th international conference of enterprise information systems (ICEIS), Springer, Barcelona, pp 301–308
9. Duta AC, Barker K (2008) P4A: a new privacy model for XML. In: Proceedings of the 2008 data and applications security XXII. Springer, pp 65–80
10. Dwork C (2006) Differential privacy. In: Proceedings of 33rd international colloquium on automata, languages and programming, http://research.microsoft.com/en-us/projects/databaseprivacy/dwork.pdf, pp 1–12
11. Adam N, Wortmann J (1989) Security-control methods for statistical databases: a comparative study. ACM Comput Surv 21(4):515–556
12. Chen B-C, Kifer D, LeFevre K, Machanavajjhala A (2009) Privacy-preserving data publishing. Found Trends Databases 2(1–2):1–167
13. Stahlberg P, Miklau G, Levine B (2007) Threats to privacy in the forensic analysis for database systems. In: Proceedings of the ACM SIGMOD international conference on management of data, Beijing, 2007
14. Ananthanarayanan R, Gupta A, Mohania M (2008) Towards automated privacy compliance in the information Life cycle. In: Proceedings of the 2009 IFIP advances in web semantics, pp 247–259
15. Agrawal R, Bayardo R, Faloutsos C, Kiernan J, Rantzau R, Srikant R (2004) Auditing compliance in a Hippocratic Database. In: Proceedings of the international conference on very large data bases (VLDB), Toronto, 2004
16. Agrawal R, Evfimievski A, Kiernan J, Velu R (2007) Auditing disclosure by relevance ranking. In: Proceedings of the ACM

SIGMOD international conference on management of data, Beijing, 2007
17. Motwani R, Nabar S, Thomas D (2008) Auditing SQL queries. In: Proceedings of the international conference on data engineering (ICDE), Istanbul, 2008
18. Johnson CM, Grandison T (2007) Compliance with data protection laws using Hippocratic Database active enforcement and auditing. IBM Syst J 46(2):255–264

History-Based Separation of Duties

►Separation of Duties

HMAC

Bart Preneel
Department of Electrical Engineering-ESAT/COSIC, Katholieke Universiteit Leuven and IBBT, Leuven-Heverlee, Belgium

Synonyms

Hash-based message authentication code

Related Concepts

►Hash Functions; ►MAC Algorithms

Definition

HMAC is a ►MAC algorithm that is computed by two calls to a ►hash function; the calls have the secret key of the MAC algorithm as part of the data input.

Background

HMAC was designed by Bellare, Canetti, and Krawczyk [2] in 1996. The HMAC construction became popular because in the mid to late 1990s no secure and efficient custom designed MAC algorithms were available and hash functions (such as ►MD5) offered a much better software performance than block ciphers; as an example, HMAC based on ►MD5 is about ten times faster than ►CBC-MAC based on ►DES.

Theory

HMAC is constructed starting from an iterated hash function h that processes inputs in blocks of n bits:

$$\mathrm{HMAC}_K(x) = h((K \oplus opad) \| h((K \oplus ipad) \| x)) .$$

Here K is the key of the MAC algorithm (padded with zeroes to a block of n bits) and $opad$ and $ipad$ are constant n-bit strings (obtained by repeating the hexadecimal values "36_x" and "$5c_x$," respectively). The resulting MAC value can (optionally) be truncated. In practice, one can apply the compression function f of the hash function to the strings $K \oplus ipad$ and $K \oplus opad$, respectively, and store the resulting values as initial values for the inner and outer hashing operations. This will reduce the number of operations of the compression function by two. HMAC offers the advantage that it can be implemented without making any modification to the code of the hash function itself.

A variant of HMAC is NMAC: NMAC replaces the initial value (IV) of the hash function by a secret key. Denote by h_K a hash function with the IV replaced by the key K, then

$$\mathrm{NMAC}_{K_1 \| K_2}(x) = h_{K_2}(h_{K_1}(x)) .$$

Bellare has improved the original security reduction of [2]; in [1] he showed that NMAC is a ►pseudo-random function and thus a secure MAC algorithm if the compression function is a ►pseudo-random function; for the security proof of HMAC with a single key as described above, pseudo-randomness under a related key attack is required. For NMAC the conditions can be further relaxed.

The best known generic attack on HMAC/NMAC is a forgery attack based on internal collisions [9]: it requires $2^{n/2}$ known text-MAC pairs (and a similar number of chosen texts if truncation is applied) and a single chosen text. A generic key recovery attack requires $2^{n/2}$ known text-MAC pairs and 2^{n+1} time, hence it is only meaningful for NMAC and for variants of HMAC that would use two different keys.

It turns out that the widely used hash functions ►MD4, ►MD5, and ►SHA-1 have weaknesses that result in more efficient attacks; the number of known or chosen text-MAC pairs is indicated as the data complexity. For HMAC-MD4 and NMAC-MD4, the following attacks are known: a forgery attack with data complexity 2^{58} [3] (compared to 2^{64} for a generic attack), and key recovery attacks with data complexity 2^{88} and time complexity 2^{95} [5] and data complexity 2^{72} and time complexity 2^{77} [11]. For NMAC-MD5, a forgery attack is known with data complexity 2^{47}; in the related key model, key recovery attacks exist with data complexity 2^{51} and time complexity 2^{100} [5, 10] and with data complexity 2^{75} and time complexity 2^{75} [11]; extending these attacks to HMAC is non-trivial. For HMAC-SHA-1 and NMAC-SHA-1 reduced to 34 out of 80 steps, a forgery with data complexity 2^{32} exists; a (partial) key recovery attack has been found for 53 out of 80 steps with a data complexity of $2^{97.5}$ [10]. Results are also known in a different attack model, in which one attempts to distinguish HMAC/NMAC based on a particular hash function from