(12) INTERNATIONAL APPLICATION PUBLISHED UNDER THE PATENT COOPERATION TREATY (PCT)

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
29 November 2007 (29.11.2007)

PCT

(10) International Publication Number
WO 2007/136346 A1

(51) International Patent Classification:
G06F 17/30 (2006.01)

(21) International Application Number:
PCT/SG2006/000126

(22) International Filing Date: 17 May 2006 (17.05.2006)

(25) Filing Language: English

(26) Publication Language: English

(71) Applicant *(for all designated States except US)*: **AGENCY FOR SCIENCE, TECHNOLOGY AND RESEARCH** [SG/SG]; 20 Biopolis Way, #07-01 Centros, Singapore, 138668 (SG).

(72) Inventors; and

(75) Inventors/Applicants *(for US only)*: **CHUA, Chun Yong** [SG/SG]; c/o Institute for Infocomm Research, 21 Heng Mui Keng Terrace, Singapore, 119613 (SG). **WU, Yong Dong** [CN/SG]; c/o Institute for Infocomm Research, 21 Heng Mui Keng Terrace,, Singapore, 119613 (SG). **ANANTHARAMAN, Lakshminarayanan** [IN/SG]; c/o Institute for Infocomm Research, 21 Heng Mui Keng Terrace,, Singapore, 119613 (SG).

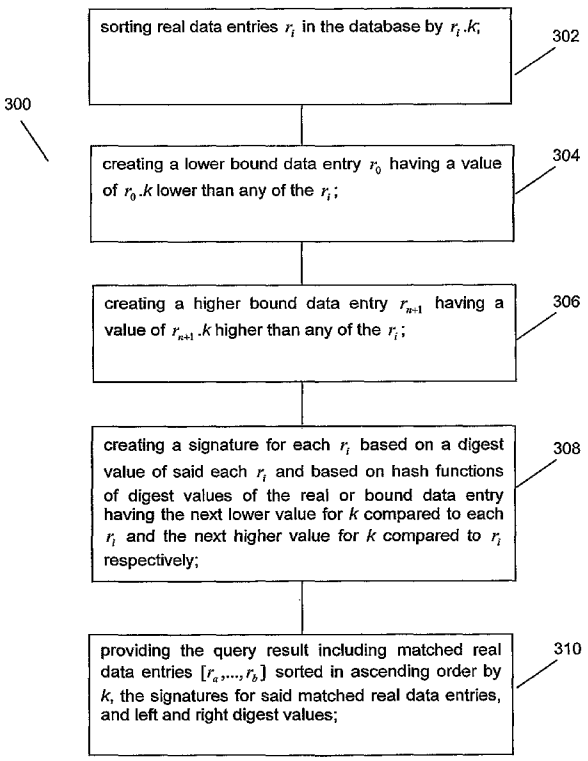
(74) Agent: **ELLA CHEONG SPRUSON & FERGUSON (SINGAPORE) PTE LTD**; P.O. Box 1531, Robinson Road Post Office,, Singapore, 903031 (SG).

(81) Designated States *(unless otherwise indicated, for every kind of national protection available)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BW, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DE, DK, DM, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RU, SC, SD, SE, SG, SK, SL, SM, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, YU, ZA, ZM, ZW.

(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT,

[Continued on next page]

(54) Title: METHOD AND SYSTEM FOR PROVIDING A QUERY RESULT IN RESPONSE TO A DATABASE QUERY



(57) Abstract: A method is provided for an owner of data to make data available to a user by transmitting it to an untrusted third party publisher. The user is able to query the data for records within particular ranges on certain attributes and verify that the publisher is sending all relevant results without omission, insertion or modification by the publisher. This is done by the owner transmitting in addition to the data, upper and lower bound values and signatures for all records obtained by combining the record and each adjacent record hashed a number of times equivalent to the difference between the record and the adjacent records when sorted by a particular attribute in relation to which queries are to be processed. The method is also adapted to provide for queries returning no valid results.

WO 2007/136346 A1

RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Published:

— *with international search report*

For two-letter codes and other abbreviations, refer to the "Guidance Notes on Codes and Abbreviations" appearing at the beginning of each regular issue of the PCT Gazette.

Method And System For Providing A Query Result In Response To A Database Query

FIELD OF INVENTION

The present invention relates broadly to a method and system for providing a query result in response to a database query, a method of manipulating data in a database for deposition of the database with a publisher processor unit, to a method of verifying a query result in response to a database query associated with a range (α, β) of an attribute k, and to respective data storage media for instructing a computer system to execute the respective methods.

BACKGROUND

Databases are often hosted in an insecure environment (e.g., the Internet), where they are susceptible to attacks which can compromise the databases integrity. Even in a protected environment (e.g., in a corporate LAN), exclusion from insider attacks may not be guaranteed.

Where data outsourcing is concerned, the problem is compounded as the owner delegates the task of satisfying user queries to a third-party publisher. With the publisher servers outside of the owner's administrative domain, and in fact the publisher servers may reside on poorly secured platforms, the query results that the publisher servers generate cannot be accepted at face value, especially where they are used as the basis for making critical decisions.

Instead, there should be provisions for the user to check the *correctness* of a query result, in terms of:

- Authenticity – All records in the results originated from the data owner. No spurious records have been added or values of valid records altered.

- Completeness – All records that satisfy the query conditions are included in the result, i.e., no qualified records have been omitted.

Currently, there are several methods available for verifying the authenticity and completeness of query results produced by untrusted third-party publishers. For example, the method by Devanbu et al [P. Devanbu, M. Gertz, C. Martel, and S. G. Stubblebine. Authentic Data Publication over the Internet. In *Journal of Computer of Computer Security*, vol. 11, pp. 291-314, 2003] is among the first solutions proposed. However, this method requires the publisher to release unqualified records to users for completeness verification which may contradict the access control policies of the database.

Another method is disclosed in Pang et al [H. Pang, A. Jain, K. Ramamritham, and K.-L. Tan. Verifying Completeness of Relational Query Results in Data Publishing. In *Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data*, September 2005]. However, the method described in Pang et al. has been identified as being vulnerable against dictionary attacks, and as being computationally expensive.

A need therefore exists to provide an alternative method and system for verifying the authenticity and completeness of query results that seek to address one or more of the above problems.

SUMMARY

In accordance with a first aspect of the present invention there is provided a method of providing a query result in response to a database query associated with a range (α, β) of an attribute k (*i.e.* $\alpha < k < \beta$), the method comprising the steps of sorting real data entries r_i in the database by $r_i.k$; creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i; creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i; creating a signature for

each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for k compared to each r_i and the next higher value for k compared to r_i respectively; providing the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k, the signatures for said matched real data entries, and left and right digest values; wherein the left digest value comprises a hash function of the real or bound data entry r_{a-1} having the next lower value for k compared to r_a, and the right digest value comprises a hash function of the real or bound data entry r_{b+1} having the next higher value for k compared to r_b.

The digest values of the real or bound data entries may be based on attribute values of the respective real or bound data entry and a random value attribute assigned to each real or bound data entry.

The signature may comprise $sig(r_i) = s(h(h^{r_i.k-r_{i-1}.k}(r_{i-1}) \mid h^0(r_i) \mid h^{r_{i+1}.k-r_i.k}(r_{i+1})))$, the left digest value comprises $h^{\alpha-r_{a-1}.k}(r_{a-1})$, the right digest value comprises $h^{r_{b+1}.k-\beta}(r_{b+1})$, and the method may further comprise verifying the query result, the verifying comprising the steps of hashing the digest left value by $(r_a.k-\alpha)$ times; hashing the digest right value by $(\beta-r_b.k)$ times; and verifying the signatures provided utilising a public key.

The method may further comprise creating a unique shadow data entry t_i for each pair of consecutive real data entries (r_i, r_{i+1}), each t_i having a value of $t_i.k$, with $r_i.k < t_i.k < r_{i+1}.k$; and creating a signature for each t_i, where $sig(t_i) = s(h(h^{t_i.k-r_i.k}(r_i) \mid t_i.k \mid h^{r_{i+1}.k-t_i.k}(r_{i+1})))$; and wherein, if no real data entry matches the query, the method further comprises selecting the shadow data entry t_J, where $r_J.k \le \alpha, \beta \le r_{J+1}.k$, and returning the query result including the signature $sig(t_J)$; if $\alpha < t_J.k$, the left digest value equal to $h^{\alpha-r_J.k}(r_J)$, else the left digest value

comprising $h^{t_J.k-r_J.k}(r_J)$; if $\beta > t_J.k$, the right digest value equal to $h^{r_{J+1}.k-\beta}(r_{J+1})$, else the right digest value equal to $h^{r_{J+1}.k-t_J.k}(r_J)$.

The method may further comprise verifying the query result, the verifying comprising the steps of if $t_J.k > \alpha$, hashing the left digest value by $(t_J.k-\alpha)$ times, else using the left digest value provided; if $t_J.k < \beta$, hashing the right digest value by $(\beta - t_J.k)$ times, else using the right digest value provided; and verify the signature provided utilising a public key.

The method may further comprise re-writing a given query condition in the form of the database query associated with the range (α, β) of the attribute *k*.

In accordance with a second aspect of the present invention there is provided a system for providing a query result in response to a database query associated with a range (α, β) of an attribute *k*, the system comprising a first processor unit for sorting real data entries r_i in the database by $r_i.k$, for creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i, for creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i, and for creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for *k* compared to each r_i and the next higher value for *k* compared to r_i respectively; a second processor unit for providing the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*, the signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function of the real or bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of the real or bound data entry r_{b+1} having the next higher value for *k* compared to r_b.

The digest values of the real or bound data entries may be based on attribute values of the respective real or bound data entry and a random value attribute assigned to each real or bound data entry.

The signature may comprise $sig(r_i) = s(h(h^{r_i.k-r_{i-1}.k}(r_{i-1}) \mid h^0(r_i) \mid h^{r_{i+1}.k-r_i.k}(r_{i+1})))$, the left digest value comprises $h^{\alpha-r_{a-1}.k}(r_{a-1})$, the right digest value comprises $h^{r_{b+1}.k-\beta}(r_{b+1})$, and the system may further comprise a third processor unit for verifying the query result, the verifying comprising the steps of hashing the digest left value by $(r_a.k-\alpha)$ times; hashing the digest right value by $(\beta - r_b.k)$ times; and verifying the signatures provided utilising a public key.

The first processor unit may create a unique shadow data entry t_i for each pair of consecutive real data entries (r_i, r_{i+1}), each t_i having a value of $t_i.k$, with $r_i.k < t_i.k < r_{i+1}.k$; and creates a signature for each t_i, where $sig(t_i) = s(h(h^{t_i.k-r_i.k}(r_i) \mid t_i.k \mid h^{r_{i+1}.k-t_i.k}(r_{i+1})))$.

If no real data entry matches the query, the second processor unit may select the shadow data entry t_J, where $r_J.k \le \alpha, \beta \le r_{J+1}.k$, and returns the query result including the signature $sig(t_J)$; if $\alpha < t_J.k$, the left digest value equal to $h^{\alpha-r_J.k}(r_J)$, else the left digest value comprising $h^{t_J.k-r_J.k}(r_J)$; if $\beta > t_J.k$, the right digest value equal to $h^{r_{J+1}.k-\beta}(r_{J+1})$, else the right digest value equal to $h^{r_{J+1}.k-t_J.k}(r_J)$.

The system may further comprising a third processor unit for verifying the query result, the verifying comprising the steps of if $t_J.k > \alpha$, hashing the left digest value by $(t_J.k-\alpha)$ times, else using the left digest value provided; if $t_J.k < \beta$, hashing the right digest value by $(\beta - t_J.k)$ times, else using the right digest value provided; and verify the signature provided utilising a public key.

The second processor unit may re-write a given query condition in the form of the database query associated with the range (α, β) of the attribute k for use by the respective processor units.

In accordance with a third aspect of the present invention there is provided a method of manipulating data in a database for deposition of the database with a publisher processor unit, the method comprising the steps of sorting real data entries r_i in the database by $r_i.k$, where k is an attribute; creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i; creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i; and creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for k compared to each r_i and the next higher value for k compared to r_i respectively.

The method may further comprise creating a unique shadow data entry t_i for each pair of consecutive real data entries (r_i, r_{i+1}), each t_i having a value of $t_i.k$, with $r_i.k < t_i.k < r_{i+1}.k$; and creating a signature for each t_i.

In accordance with a fourth aspect of the present invention there is provided a method of providing a query result in response to a database query associated with a range (α, β) of an attribute k, the method comprising the steps of providing matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k, providing signatures for said matched real data entries, and providing left and right digest values; wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for k compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for k compared to r_b.

If no real data entry matches the query, the method may comprise selecting a shadow data entry t_J, where $r_J.k \leq \alpha, \beta \leq r_{J+1}.k$, and returning the query result including a signature $sig(t_J)$; if $\alpha < t_J.k$, the left digest value equal to $h^{\alpha - r_J.k}(r_J)$, else the left digest value comprising $h^{t_J.k - r_J.k}(r_J)$; if $\beta > t_J.k$, the right digest value equal to $h^{r_{J+1}.k - \beta}(r_{J+1})$, else the right digest value equal to $h^{r_{J+1}.k - t_J.k}(r_J)$.

In accordance with a fifth aspect of the present invention there is provided a method of verifying a query result in response to a database query associated with a range (α, β) of an attribute *k*, the method comprising the steps of receiving the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*, signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for *k* compared to r_b; hashing the digest left value; hashing the digest right value; and verifying the signatures provided utilising a public key.

If no real data entry matches the query, the query result may include a signature $sig(t_J)$, where t_J is a shadow data entry with $r_J.k \leq \alpha, \beta \leq r_{J+1}.k$; if $\alpha < t_J.k$, the left digest value equal to $h^{\alpha - r_J.k}(r_J)$, else the left digest value comprising $h^{t_J.k - r_J.k}(r_J)$; if $\beta > t_J.k$, the right digest value equal to $h^{r_{J+1}.k - \beta}(r_{J+1})$, else the right digest value equal to $h^{r_{J+1}.k - t_J.k}(r_J)$; and the method comprises, if $t_J.k > \alpha$, hashing the left digest value by $(t_J.k - \alpha)$ times, else using the left digest value provided; if $t_J.k < \beta$, hashing the right digest value by $(\beta - t_J.k)$ times, else using the right digest value provided; and verifying the signature provided utilising the public key.

In accordance with a sixth aspect of the present invention there is provided a data storage medium having stored thereon computer code means for instructing a computer system to execute a method of providing a query result in response to a database query associated with a range (α, β) of an attribute k, the method comprising the steps of sorting real data entries r_i in the database by $r_i.k$; creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i; creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i; creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for k compared to each r_i and the next higher value for k compared to r_i respectively; providing the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k, the signatures for said matched real data entries, and left and right digest values; wherein the left digest value comprises a hash function of the real or bound data entry r_{a-1} having the next lower value for k compared to r_a, and the right digest value comprises a hash function of the real or bound data entry r_{b+1} having the next higher value for k compared to r_b.

In accordance with a seventh aspect of the present invention there is provided a data storage medium having stored thereon computer code means for instructing a computer system to execute a method of manipulating data in a database for deposition of the database with a publisher processor unit, the method comprising the steps of sorting real data entries r_i in the database by $r_i.k$, where k is an attribute; creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i; creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i; and creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for k compared to each r_i and the next higher value for k compared to r_i respectively.

In accordance with an eighth aspect of the present invention there is provided a data storage medium having stored thereon computer code means for instructing a computer system to execute a method of providing a query result in response to a database query associated with a range (α, β) of an attribute *k*, the method comprising the steps of providing matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*, providing signatures for said matched real data entries, and providing left and right digest values; wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for *k* compared to r_b.

In accordance with a ninth aspect of the present invention there is provided a data storage medium having stored thereon computer code means for instructing a computer system to execute a method of verifying a query result in response to a database query associated with a range (α, β) of an attribute *k*, the method comprising the steps of receiving the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*, signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for *k* compared to r_b; hashing the digest left value by $(r_a.k-\alpha)$ times; hashing the digest right value by $(\beta-r_b.k)$ times; and verifying the signatures provided utilising a public key.

BRIEF DESCRIPTION OF THE DRAWINGS

Embodiments of the invention will be better understood and readily apparent to one of ordinary skill in the art from the following written description, by way of example only, and in conjunction with the drawings, in which:

Figure 1 is a schematic drawings illustrating a system 100 for providing a query result in response to a database query associated with a range (α, β) of an attribute *k*.

Figure 2 is a schematic diagram illustrating a computer system for implementing components of the method and system of Figure 1.

Figure 3 shows a flow chart illustrating a method of providing a query result in response to a database query associated with a range {α, β} of an attribute *k*.

Figure 4 shows a flow chart illustrating a method of manipulating data in a database for deposition of the database with a publisher processor unit.

Figure 5 shows a flow chart illustrating a method of providing a query result in response to a database query associated with a range {α, β} of an attribute *k*.

Figure 6 shows a flow chart illustrating a method of verifying a query result in response to a database query associated with a range {α, β} of an attribute k.

DETAILED DESCRIPTION

The embodiment described can provide a method and system for verifying the authenticity and completeness of query results produced by untrusted third-party publishers, that avoid releasing unqualified records to users by employing the used of hash iteration to embed attribute values of records into the signature of neighbouring records. The described embodiment is further secured against dictionary attacks, and can yield better performance compared with existing techniques.

For the purpose of this description, the following terms are briefly defined for clarity. However, the given definitions are not intended to limit the scope of the present invention, but rather are intended only for better clarity of the description of the example embodiment.

Owner: Refers to the creator of the relational data. Note that the owner must be trusted by the user since an attempt on his part to generate incorrect data cannot be detected by the user.

User: Someone who issues relational queries based on the data generated by the creator. There may be some access control policy in place that restricts the user's access to the data. Hence, there is a need to prevent the release of unqualified data to the user.

Third-party publisher: A third-party who has been delegated the task to handle user queries. Our scheme removes the need for the publisher to be a party trusted by the user. Any attempt on his part to generate incorrect answer will be detected by the user. On the other hand, the publisher can also be a trusted party (can even be the owner himself). In this case, our scheme would help guard against attempts by adversaries to comprise the system or intercept and alter the answer generated by the publisher before the answer reaches the user.

Figure 1 is a schematic drawings illustrating a system 100 for providing a query result in response to a database query associated with a range (α, β) of an attribute k. The system comprises an owner processor unit 102 coupled to a database 101 for sorting real data entries r_i in the database 101 by $r_i.k$, for creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i, for creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i, and for creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry

having the next lower value for k compared to each r_i and the next higher value for k compared to r_i respectively. The sorted real data entries r_i in the database 101 together with the lower and upper bound data entries r_0 , r_{n+1} and with the signatures for each r_i are deposited in a database 105 coupled to a publisher processor unit 104 in a one-time process as part of a data publishing arrangement between the owner and the publisher.

The publisher processor unit 104 provides the query results in response to a query received from a user utilising a user processor unit 106. Each query result includes matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k, the signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function $h^{\alpha-r_{a-1}.k}(r_{a-1})$ of the real or bound data entry r_{a-1} having the next lower value for k compared to r_a, and the right digest value comprises a hash function $h^{r_{b+1}.k-\beta}(r_{b+1})$ of the real or bound data entry r_{b+1} having the next higher value for k compared to r_b.

The user processor unit 106 verifies the query result. The verifying comprises the steps of hashing the digest left value by $(r_a.k-\alpha)$ times; hashing the digest right value by $(\beta-r_b.k)$ times; and verifying the signatures provided utilising a public key of the owner. In the example embodiment, the processor units 102, 104, 106 are each connected to the Internet 108, and implemented as web-based application programs.

The processor units 102, 104, and 106 can each be implemented on a computer system 200, schematically shown in Figure 2. They may be implemented as software, such as a computer program being executed within the computer system 200, and instructing the computer system 200 to conduct the described steps.

The computer system 200 comprises a computer module 202, input modules such as a keyboard 204 and mouse 306 and a plurality of output devices such as a display 208, and printer 210.

The computer module 202 is connected to a computer network 212 via a suitable transceiver device 214, to enable access to e.g. the Internet or other network systems such as Local Area Network (LAN) or Wide Area Network (WAN).

The computer module 202 in the example includes a processor 218, a Random Access Memory (RAM) 220 and a Read Only Memory (ROM) 222. The computer module 202 also includes a number of Input/Output (I/O) interfaces, for example I/O interface 224 to the display 208, and I/O interface 226 to the keyboard 204.

The components of the computer module 202 typically communicate via an interconnected bus 228 and in a manner known to the person skilled in the relevant art.

The application program is typically supplied to the user of the computer system 200 encoded on a data storage medium such as a CD-ROM or flash memory carrier and read utilising a corresponding data storage medium drive of a data storage device 230. The application program is read and controlled in its execution by the processor 218. Intermediate storage of program data maybe accomplished using RAM 220.

In the following, examples of data sets and digest functions will be described, together with respective characteristics in terms of vulnerability to dictionary attacks. An owner has a set of n records $\lambda = [r_1, ..., r_i, r_{i+1}, ..., r_n]$ sorted in order of ascending k values, i.e., $r_i.k \leq r_{i+1}.k$, where k is an arbitrary record attribute with an exclusive lower bound L and upper bound U, i.e., $L < k < U$. A user issues a query for all the records whose k values lie between α and β, i.e., $\alpha < k < \beta$. With λ arranged in order of ascending k values, the answer to the query occupies a contiguous range in λ,

$\lambda'=[r_a,...,r_b]$, where r_a and r_b denotes the first and last record respectively in λ to satisfy the query condition. The untrusted publisher, with whom the owner would have deposited the set of n records and who generates the query answer on behalf of the owner, needs to prove to the user that λ' is authentic and complete without exposing records before r_a and after r_b.

	Name	Other Attributes	Salary
r_1	Alice	...	800
r_2	Bob	...	2100
r_3	Cat	...	3500
r_4	Dan	...	4200
r_5	Emma	...	5400
r_6	Felix	...	6000

Table 1

The data in Table 1 is used to denote a record set which contains information of 6 employees in a company. An example query is for the list of employees whose salary (k) is greater than 1500 (α) but less than 5500 (β). In this example, the correct result is $\lambda'=[r_2, r_3, r_4, r_5]$. The publisher should prove to the user that λ' is authentic and complete without exposing any information about r_1 and r_6.

In the described system and method, prior to depositing the data with the publisher for query processing, the owner generates a set of signatures for λ utilising an application program executed on the owner processor unit 102 (Figure 1) in the following manner:

- First, the owner defines the values for L and U based on the domain of k. These two values are made known to the publisher and the user.

- Next, the owner creates two boundary records r_0 and r_{n+1}, whose k values are L and U respectively, and inserts them into λ to obtain $[r_0, r_1, ..., r_n, r_{n+1}]$. The purpose of this is to simplify the process of generating the signatures.
- The owner then creates a signature for each record (except for the boundary records) using the following equation:

$$sig(r_i) = s(h(h^{r_i.k-r_{i-1}.k}(r_{i-1}) \mid h^0(r_i) \mid h^{r_{i+1}.k-r_i.k}(r_{i+1})))$$

Where | represents concatenation, s() is a signature function using the publisher's private key and $h^x(r_i)$ applies a collision-resistant hash function h on r_i iteratively for x times, i.e., :

$$h^{x+1}(r_i) = h(h^x(r_i)), \text{ for } x \geq 0,$$

and $h^0(r_i)$ denotes the digest value of r_i.

- The owner deposits λ and the corresponding signature set $S = [sig(r_1), ..., sig(r_n)]$ with the publisher.

The publisher generates query answer λ' and *correctness proof* from λ and S respectively.

- The owner makes available his public key to the user through a secure channel. This can be an offline process. The public key shall be used by the user to verify the authenticity of the records and correctness returned by the publisher.

	Name	Other Attributes	Salary	Signature, $sig(r_i)$
r_0	L	...	0	Undefined
r_1	Alice	...	800	$s(h(h^{800}(r_0) \mid h^0(r_1) \mid h^{1300}(r_2)))$
r_2	Bob	...	2100	$s(h(h^{1300}(r_1) \mid h^0(r_2) \mid h^{1400}(r_3)))$
r_3	Cat	...	3500	$s(h(h^{1400}(r_2) \mid h^0(r_3) \mid h^{700}(r_4)))$
r_4	Dan	...	4200	$s(h(h^{700}(r_3) \mid h^0(r_4) \mid h^{1200}(r_5)))$

r_5	Emma	...	5400	$s(h(h^{1200}(r_4) \mid h^0(r_5) \mid h^{600}(r_6)))$
r_6	Felix	...	6000	$s(h(h^{600}(r_5) \mid h^0(r_6) \mid h^{4000}(r_7)))$
r_7	U	...	10000	Undefined

Table 2

Table 2 shows the corresponding signatures of the records. Here, $L = 0$ and $U = 10000$. Each signature is a function of its corresponding record, and its immediately preceding and following records. For example, $sig(r_4) = s(h(h^{700}(r_3) \mid h^0(r_4) \mid h^{1200}(r_5)))$ is a function of r_3, r_4 and r_5, and is also dependent on the difference in their salary values. 700 reflects the difference between the salary values of r_3 and r_4 while 1200 reflects that of r_4 and r_5. This is synonymous to saying that the salary of previous record is 700 less that than of r_4 and the salary of the next record is 1200 more than that of r_4.

With respect to each query received, the publisher computes and returns the following items, utilising an application program executed on the publisher processor unit 104 (Figure 1):

- Answer $\lambda' = [r_a, ..., r_b]$,
- Signature subset $S' = [sig(r_a), ..., sig(r_b)]$,
- Digest $left = h^{\alpha - r_{a-1}.k}(r_{a-1})$, and
- Digest $right = h^{r_{b+1}.k - \beta}(r_{b+1})$.

In the salary example, this would be:

- Answer $\lambda' = [r_2, r_3, r_4, r_5]$,
- Signature subset $S' = [sig(r_2), sig(r_3), sig(r_4), sig(r_5)]$
- Digest $left = h^{700}(r_1)$, since $\alpha = 1500$, $r_1.salary = 800$,
- Digest $right = h^{500}(r_6)$, since $\beta = 5500$, $r_6.salary = 6000$.

Based on the objects returned, the user verifies the correctness of λ' in the following manner, utilising an application program executed on the user processor unit 106 (Figure 1):

- The user hashes digest *left* by $(r_a.k-\alpha)$ times to get $h^{r_a.k-r_{a-1}.k}(r_{a-1})$. This will be used in the signature verification for $sig(r_a)$.
- The user hashes digest *right* by $(\beta-r_b.k)$ times to get $h^{r_{b+1}.k-r_b.k}(r_{b+1})$. This will be used in the signature verification for $sig(r_b)$.
- The user verifies λ' against S'. For each record r_i in λ', the user computes the following:

$$h(h^{r_i.k-r_{i-1}.k}(r_{i-1}) \mid h^0(r_i) \mid h^{r_{i+1}.k-r_i.k}(r_{i+1}))$$

and performs signature verification against its corresponding signature $sig(r_i)$ in S' using the owner's public key.

In the given example, the user thus:

- hashes digest $left = h^{700}(r_1)$ by $(2100-1500)$ times to get $h^{1300}(r_1)$.
- hashes digest $right = h^{500}(r_6)$ by $(5500-5400)$ times to get $h^{600}(r_6)$.
- computes $h(h^{1300}(r_1) \mid h^0(r_2) \mid h^{1400}(r_3))$ from r_2 and r_3 (note that $h^{1300}(r_1)$ was computed above), and verifies against $sig(r_2)$.
- computes $h(h^{1400}(r_2) \mid h^0(r_3) \mid h^{700}(r_4))$ from r_2, r_3 and r_4, and verifies it against $sig(r_3)$.
- computes $h(h^{700}(r_3) \mid h^0(r_4) \mid h^{1200}(r_5))$ from r_3, r_4 and r_5, and verifies it against $sig(r_4)$.
- computes $h(h^{1200}(r_4) \mid h^0(r_5) \mid h^{600}(r_6))$ from r_4 and r_5 (note that $h^{600}(r_6)$ was computed above), and verifies against $sig(r_5)$.

Given that the owner's secret key has not been compromised and all the signature verifications are successful, the user is confident that:

- All the records in λ' are authentic since it is computationally infeasible for an adversary to create valid digital signatures for spurious records or records whose attributes have been tampered with.

For example, if the publisher tries to return $\lambda'=[r_2, r_3, r_s, r_4, r_5]$, where r_s is a spurious record. The publisher needs to generate a valid signature $sig(r_s)$ based on (r_3, r_s, r_4). Also, the publisher needs to generate valid signatures for r_3 and r_4 based on (r_2, r_3, r_s) and (r_s, r_4, r_5) respectively. Without the owner's secret key, it is computationally infeasible for him to do so.

If the publisher tries to return $\lambda'=[r_2, r_3', r_4, r_5]$ where r_3' is a valid record but having one of its attributes changed (can be any attribute beside salary), the publisher must generate a valid signature for $sig(r_3')$. New signatures must be generated for r_2 and r_4 as well. Again, this is computationally infeasible.

- λ' occupies a contiguous range in λ. Signature $sig(r_i)$ is a function of the record r_i, the preceding record r_{i-1} and trailing record r_{i+1}, when the records are sorted in ascending order of k values. An adversary cannot omit an intermediate record as it is computationally infeasible for him to create valid record signatures for the incomplete list of records.

If the publisher tries to return $\lambda'=[r_2, r_4, r_5]$, the publisher needs to generate a new signature for r_2 based on (r_1, r_2, r_4). Likewise, the publisher needs to generate a new signature for r_4 based on (r_2, r_4, r_5). Without the owner's secret key, it is computationally infeasible for the publisher to do so.

- $r_{a-1}.k \leq \alpha$. During the process of verifying r_a against $sig(r_a)$, the user needs to generate

$$h(h^{r_a.k-r_{a-1}.k}(r_{a-1}) \mid h^0(r_a) \mid h^{r_{a+1}.k-r_a.k}(r_{a+1}))$$

The user can generate components $h^0(a_i)$ and $h^{r_{a+1}.k-r_a.k}(r_{a+1})$ directly from r_a and r_{a+1} since they are returned as part of λ'. To assist the user in generating component $h^{r_a.k-r_{a-1}.k}(r_{a-1})$, the publisher could return r_{a-1}. However, this would violate the access control policy, since r_{a-1} does not satisfy the query condition. Rather the publisher returns digest $left = h^{\alpha-r_{a-1}.k}(r_{a-1})$. The user can derive the required component by hashing $left$ $(r_a.k-\alpha)$ times. If $sig(r_a)$ is verified to be correct, the user is sure that $h^{r_a.k-r_{a-1}.k}(r_{a-1})$ is derived correctly. While the user does not know the exact value of $r_{a-1}.k$, he is sure that $r_a.k-r_{a-1}.k \geq r_a.k-\alpha$ and hence $r_{a-1}.k \leq \alpha$.

In the salary example, $r_1.salary = 800$, $\alpha = 1500$, $r_2.salary = 2100$ and $r_3.salary = 3500$. The publisher cannot disclose $r_1.salary$ to the user. But if the publisher can prove that $r_1.salary$ is less than $r_2.salary$ by at least 600, then that is sufficient. The publisher computes and returns $left = h^{700}(r_1)$. Because the user can generate the component correctly by hashing $left$ 600 times, he knows that $r_2.salary - r_1.salary \geq 600$. Hence, $r_1.salary \leq 1500$. Also, by revealing only $left = h^{700}(r_1)$, the publisher is sure that the user is not able to derive r_1 or any of its attributes. A hash function has the property of a one-way function, i.e., given $y = h(x)$, it is computationally infeasible to derive x when given y. This property can be extended to a hash iteration function, i.e., given $h^{i+1}(x) = h(h^i(x))$, it is computationally infeasible to derive $h^i(x)$ when given $h^{i+1}(x)$. Given $left = h^{700}(r_1)$, the user is not able to derive $h^{699}(r_1)$, $h^{698}(r_1)$, ..., $h^0(r_1)$. Furthermore, the user does not know where is the start of the chain, i.e., he does not know if he has obtained $h^0(r_1)$.

If the publisher returns $\lambda' = [r_3, r_4, r_5]$, as part of the signature verification process, the user generates

$$h(h^{1400}(r_2) \mid h^0(r_3) \mid h^{700}(r_4))$$

Since $r_3.salary = 3500$, the user expects $r_2.salary$ to be less than $r_3.salary$ by at least 2000. Else r_2 would have qualified as part of λ'. Hence, the user would hash *left* iteratively by 2000 times. As mentioned above, it is computationally infeasible for the publisher to generate a new but valid signature for r_3. In order for the user to correctly derive $h^{1400}(r_2)$, the publisher must return $left = h^{-600}(r_2)$. The publisher knows the correct value of $h^0(r_2)$. But due to the one-way property of the hash function, it is computationally infeasible for him to derive $h^{-600}(r_2)$ from it.

- $r_{b+1}.k \geq \beta$. During the process of verifying r_b against $sig(r_b)$, the user generates

$$h(h^{r_b.k-r_{b-1}.k}(r_{b-1}) \mid h^0(r_b) \mid h^{r_{b+1}.k-r_b.k}(r_{b+1}))$$

The user can generate components $h^{r_b.k-r_{b-1}.k}(r_{b-1})$ and $h^0(r_b)$ from r_{b-1} and r_b, which are returned as part of λ'. To assist the user in generating component $h^{r_{b+1}.k-r_b.k}(r_{b+1})$, the publisher could return r_{b+1}. However, this would violate the access control policy, since r_{b+1} does not satisfy the query condition. Rather, the publisher returns digest $right = h^{r_{b+1}.k-\beta}(r_{b+1})$. The user can derive the required component by hashing *right* $(\beta - r_b.k)$ times. If $sig(r_b)$ is verified to be correct, the user is sure that $h^{r_{b+1}.k-r_b.k}(r_{b+1})$ is derived correctly. While the user does not know the exact value of $r_{b+1}.k$, he is sure that $r_{b+1}.k - r_b.k \geq \beta - r_b.k$ and hence $r_{b+1}.k \geq \beta$.

In the example, $r_4.salary = 4200$, $r_5.salary = 5400$, $\alpha = 5500$ and $r_6.salary = 6000$. The publisher cannot disclose $r_6.salary$ to the user. But if he can prove that $r_6.salary$ is greater than $r_5.salary$ by at least 100, then that is sufficient. So the publisher computes and returns $right = h^{500}(r_6)$. Because the user can generate the correct component by hashing *right* 100 times, he knows that $r_6.salary - r_5.salary \geq 100$. Hence, $r_6.salary \geq 5500$.

If the publisher returns $\lambda' = [r_2, r_3, r_4]$. As part of the signature verification process, the user generates

$$h(h^{700}(r_3) \mid h^0(r_4) \mid h^{1200}(r_5))$$

Since $r_4.salary = 4200$, the user expects $r_5.salary$ to be greater than $r_4.salary$ by at least 1300. Hence, the user would hash *right* iteratively by 1300 times. In order for the user to correctly derive $h^{1200}(r_5)$, the publisher must return $right = h^{-100}(r_5)$. However, it is computationally infeasible for him to do so.

Based on the above, the user is sure that the records in λ' are authentic. The user is also sure that λ' occupies a contiguous range in λ. While the user does not know the exact values of $r_{a-1}.k$ and $r_{b+1}.k$, the user is sure that both values do not satisfy the query condition (i.e., $\alpha < k < \beta$). Hence, the user can conclude that the answer is both authentic and complete.

In the described example, $h^0(r_i)$ refers to the digest value of r_i. The present disclosure is not limited to a particular digest (hash) function. However, it may be desirable to take into consideration the structure of the data set when deciding on which digest function to use. In the following, examples of data sets and digest functions will be described, together with respective characteristics in terms of vulnerability to dictionary attacks.

	Name	Salary	Signature, $sig(r_i)$
r_0	L	0	Undefined
r_1	Alice	800	$s(h(h^{800}(r_0) \mid h^0(r_1) \mid h^{1300}(r_2)))$
r_2	Bob	2100	$s(h(h^{1300}(r_1) \mid h^0(r_2) \mid h^{1400}(r_3)))$
r_3	Cat	3500	$s(h(h^{1400}(r_2) \mid h^0(r_3) \mid h^{700}(r_4)))$

r_4	Dan	4200	$s(h(h^{700}(r_3) \mid h^0(r_4) \mid h^{1200}(r_5)))$
r_5	Emma	5400	$s(h(h^{1200}(r_4) \mid h^0(r_5) \mid h^{600}(r_6)))$
r_6	Felix	6000	$s(h(h^{600}(r_5) \mid h^0(r_6) \mid h^{4000}(r_7)))$
r_7	U	10000	Undefined

Table 3

To demonstrate the vulnerability, a record set as shown in Table 3 is first considered. The data set is similar to the record set in Table 2 above, except that there are only two attributes per record (name and salary). If the digest function is defined as:

$$h^0(r_i) = h(r_i.name \mid r_i.salary),$$

in response to the same query example where $\alpha = 1500$ and $\beta = 5500$, the publisher returns $left = h^{700}(r_1)$. If the user has the list of employee's names, given λ', the user knows that $r_{a-1}.name$ is either Alice of Felix, as the remaining names will be identified in the provided results.

Now, the user can try to find out $r_{a-1}.name$ and $r_{a-1}.salary$ by generating $h^{\alpha-x}(Alice \mid x)$ and $h^{\alpha-x}(Felix \mid x)$, for $0 \le x \le \alpha$. The one that tallies with $left$ will be the correct values of r_{a-1}.

One possible way to defeat this vulnerability is by introducing an additional attribute to each record, in the form of a sufficiently long random byte array and make $h^0(r_i)$ dependent on this attribute. For example, $h^0(r_i) = h(r_i.name \mid r_i.salary \mid r_i.random)$. Now, the user does not know $r_{a-1}.random$. He must guess correctly its value before conducting the above-mentioned attack. If $r_{a-1}.random$ is 40-bit length, he must try, on the average 2^{39} times to guess its value correctly.

A modification of the previously described example to handle queries with null query results will now be described.

For each pair of consecutive records (r_i, r_{i+1}), where $0 \le i \le n$, if $r_i.k$ and $r_{i+1}.k$ are neither equal nor consecutive values in the domain of k, the owner creates:

- A unique shadow record t_i, where $t_i.k$ is a randomly chosen value such that $r_i.k < t_i.k < r_{i+1}.k$,
- And its associated signature

$$sig(t_i) = s(h(h^{t_i.k - r_i.k}(r_i) \mid t_i.k \mid h^{r_{i+1}.k - t_i.k}(r_{i+1})))$$

Example shadow records t_i for the data set of Table 2 is shown in Table 4.

	Salary	Signature, $sig(t_i)$
t_0	300	$s(h(h^{300}(r_0) \mid t_0.k \mid h^{500}(r_1)))$
t_1	1300	$s(h(h^{500}(r_1) \mid t_1.k \mid h^{800}(r_2)))$
t_2	2500	$s(h(h^{400}(r_2) \mid t_2.k \mid h^{1000}(r_3)))$
t_3	3900	$s(h(h^{400}(r_3) \mid t_3.k \mid h^{300}(r_4)))$
t_4	4800	$s(h(h^{600}(r_4) \mid t_4.k \mid h^{600}(r_5)))$
t_5	5500	$s(h(h^{100}(r_5) \mid t_5.k \mid h^{500}(r_6)))$
t_6	8000	$s(h(h^{2000}(r_6) \mid t_6.k \mid h^{2000}(r_7)))$

Table 4

If a user submits a query for all the records which satisfy $\alpha < k < \beta$ and there are no records which satisfy the query condition, the publisher returns the following:

- The shadow record t_J, where $t_J \in T = \{ t_0, \ldots, t_i \}$ and $r_J.k \le \alpha, \beta \le r_{J+1}.k$,

- Its associated signature $sig(t_J)$,
- If $\alpha < t_J.k$, return $left = h^{\alpha - r_J.k}(r_J)$, else return $left = h^{t_J.k - r_J.k}(r_J)$,
- If $\beta > t_J.k$, return $right = h^{r_{J+1}.k - \beta}(r_{J+1})$, else return $right = h^{r_{J+1}.k - t_J.k}(r_{J+1})$.

Based on the objects returned, the user verifies t_J against $sig(t_J)$. To do so, the user

- Constructs $h(h^{t_J.k - r_J.k}(r_J) \mid t_J.k \mid h^{r_{J+1}.k - t_J.k}(r_{J+1}))$
- To generate component $h^{t_J.k - r_J.k}(r_J)$, the user checks the value of $t_J.k$. If $t_J.k > \alpha$, then the user hashes $left = h^{\alpha - r_J.k}(r_J)$ by $(t_J.k - \alpha)$ times to obtain $h^{t_J.k - r_J.k}(r_J)$. Otherwise, $left = h^{t_J.k - r_J.k}(r_J)$ is already returned by the publisher.
- To generate $h^{r_{J+1}.k - t_J.k}(r_{J+1})$, the user checks the value of $t_J.k$. If $t_J.k < \beta$, then the user hashes $right = h^{r_{J+1}.k - \beta}(r_{J+1})$ by $(\beta - t_J.k)$ times to obtain $h^{r_{J+1}.k - t_J.k}(r_{J+1})$. Otherwise, $right = h^{r_{J+1}.k - t_J.k}(r_{J+1})$ is already returned by the publisher.

If the verification is correct, the shadow record is authentic. Based on the intermediate digests, the user is sure that there exist two consecutive records $r_J.k$ and $r_{J+1}.k$ where $r_J.k \le \alpha, \beta \le r_{J+1}.k$.

For example, a user issues a query where $\alpha = 1000$ and $\beta = 2000$. Since $r_1.k = 800$ and $r_2.k = 2100$, there is no record that satisfy the query. In this case, the publisher return t_1, where $t_1.k = 1300$. The user generates the following:

$$h(h^{500}(r_1) \mid t_J.k \mid h^{800}(r_2))$$

Since $\alpha < t_1.k$ and $t_1.k - \alpha = 300$, the user expects $r_1.k$ to be less than $t_1.k$ by at least 300. Thus the publisher generates and returns $left = h^{200}(r_1)$. The user can

generate the component $h^{t_1.k-r_1.k}(r_1)$ by hashing $left = h^{200}(r_1)$ 300 times. If the signature verification is correct, the user is sure that $r_1.k$ is less than $t_1.k$ by at least 300.

If $\alpha = 1600$ and β = 2000, which is greater than $t_1.k$, publisher returns $t_1.k = 1300$. As the user therefore knows that $r_1.k < t_1.k$, he is certain that $r_1.k < 1300$. The publisher returns $left = h^{500}(r_1)$.

Since $\beta > t_1.k$ and $t_1.k - \beta = 700$, the user expects $r_2.k$ to be greater than $t_1.k$ by at least 700. The publisher generates and returns $right = h^{100}(r_2)$. The user can generate component $h^{r_2.k-t_1.k}(r_2)$ by hashing $right = h^{100}(r_2)$ 700 times. If the signature verification is correct, the user is sure that $r_2.k$ is greater than $t_1.k$ by at least 700.

If α = 1000 and suppose $\beta = 1200$, β lesser than $t_1.k$, the publisher returns $t_1.k = 1300$. Since the user does knows that $r_2.k > t_1.k$, the user is certain that $r_2.k > 1300$. The publisher returns $right = h^{800}(r_2)$, and the user can perform the signature verification.

While the example embodiment has been described in the context of range queries with condition $\alpha < k < \beta$, it will be appreciated by a person skilled in the art that the example embodiment can also handle conditions such as $\alpha \le k \le \beta$, $\alpha < k$, $k < \beta$, $\alpha \le k$, $k \le \beta$ where they can be written into the form $\alpha' < k < \beta'$. Likewise, point query can also be covered.

Figure 3 shows a flow chart 300 illustrating a method of providing a query result in response to a database query associated with a range (α, β) of an attribute *k*. At step 302, real data entries r_i are sorted in the database by $r_i.k$. At step 304, a lower bound data entry r_0 is created having a value of $r_0.k$ lower than any of the r_i. At step 306, an upper bound data entry r_{n+1} is created having a value of $r_{n+1}.k$ higher than any of the r_i. At step 308, a signature is created for each r_i based on a

digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for k compared to each r_i and the next higher value for k compared to r_i respectively.

At step 310, the query result is provided including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k, the signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function of the real or bound data entry r_{a-1} having the next lower value for k compared to r_a, and the right digest value comprises a hash function of the real or bound data entry r_{b+1} having the next higher value for k compared to r_b.

Figure 4 shows a flow chart 400 illustrating a method of manipulating data in a database for deposition of the database with a publisher processor unit. At step 402, real data entries r_i in the database are sorted by $r_i.k$, where k is an attribute. At step 404, a lower bound data entry r_0 is created having a value of $r_0.k$ lower than any of the r_i. At step 406, an upper bound data entry r_{n+1} is created having a value of $r_{n+1}.k$ higher than any of the r_i. At step 408, a signature is created for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for k compared to each r_i and the next higher value for k compared to r_i respectively;

Figure 5 shows a flow chart 500 illustrating a method of providing a query result in response to a database query associated with a range (α, β) of an attribute k. At step 502, matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k are provided. At step 504, signatures for said matched real data entries are provided. At step 506, left and right digest values are provided; wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1}

having the next lower value for k compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for k compared to r_b.

Figure 6 shows a flow chart 600 illustrating a method of verifying a query result in response to a database query associated with a range (α, β) of an attribute k. At step 602, the query result is received including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k, signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for k compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for k compared to r_b. At step 602; the digest left value is hashed. At step 604, the digest right value is hashed. At step 606, the signatures provided are verified utilising a public key.

The disclosed methods and systems can provide the following features and advantages.

- Verifying the correctness of a relational database query result. We consider a query result to be correct only if it is authentic and complete.
- Pre-processing of the records, which involves generating a record signature for each of them, before sending both the records and signatures to the publisher.
- The publisher proves to the user that the query result is authentic and complete by returning the qualified records along with their associated signatures. In addition, the publisher computes and returns two intermediate hashes based on the boundary records.
- After receiving the records, the associated signatures and the intermediate hashes, the user performs a set of computation to verify the authenticity and completeness of the result.

- Unqualified records are not required to be exposed to the user. Based on the intermediate hashes, the user can verify that the boundary records indeed do not satisfy the query conditions but is not able to derive their actual values.

It will be appreciated by a person skilled in the art that numerous variations and/or modifications may be made to the present invention as shown in the specific embodiments without departing from the spirit or scope of the invention as broadly described. The present embodiments are, therefore, to be considered in all respects to be illustrative and not restrictive.

CLAIMS

1. A method of providing a query result in response to a database query associated with a range (α, β) of an attribute *k*, the method comprising the steps of

sorting real data entries r_i in the database by $r_i.k$;

creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i;

creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i;

creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for *k* compared to each r_i and the next higher value for *k* compared to r_i respectively;

providing the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*, the signatures for said matched real data entries, and left and right digest values;

wherein the left digest value comprises a hash function of the real or bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of the real or bound data entry r_{b+1} having the next higher value for *k* compared to r_b.

2. The method as claimed in claim 1, wherein the digest values of the real or bound data entries is based on attribute values of the respective real or bound data entry and a random value attribute assigned to each real or bound data entry.

3. The method as claimed in claims 1 or 2, wherein the signature comprises $sig(r_i) = s(h(h^{r_i.k-r_{i-1}.k}(r_{i-1}) \mid h^0(r_i) \mid h^{r_{i+1}.k-r_i.k}(r_{i+1})))$, the left digest value comprises $h^{\alpha-r_{a-1}.k}(r_{a-1})$, the right digest value comprises $h^{r_{b+1}.k-\beta}(r_{b+1})$, and the

method further comprises verifying the query result, the verifying comprising the steps of:

hashing the digest left value by $(r_a.k-\alpha)$ times;

hashing the digest right value by $(\beta-r_b.k)$ times;

and verifying the signatures provided utilising a public key.

4. The method as claimed in claims 1 or 2, further comprising creating a unique shadow data entry t_i for each pair of consecutive real data entries (r_i, r_{i+1}), each t_i having a value of $t_i.k$, with $r_i.k < t_i.k < r_{i+1}.k$; and creating a signature for each t_i, where $sig(t_i) = s(h(h^{t_i.k-r_i.k}(r_i) \mid t_i.k \mid h^{r_{i+1}.k-t_i.k}(r_{i+1})))$; and

wherein, if no real data entry matches the query, the method further comprises selecting the shadow data entry t_J, where $r_J.k \le \alpha, \beta \le r_{J+1}.k$, and returning the query result including

the signature $sig(t_J)$;

if $\alpha < t_J.k$, the left digest value equal to $h^{\alpha-r_J.k}(r_J)$, else the left digest value comprising $h^{t_J.k-r_J.k}(r_J)$;

if $\beta > t_J.k$, the right digest value equal to $h^{r_{J+1}.k-\beta}(r_{J+1})$, else the right digest value equal to $h^{r_{J+1}.k-t_J.k}(r_J)$.

5. The method as claimed in claim 4, further comprising verifying the query result, the verifying comprising the steps of:

if $t_J.k > \alpha$, hashing the left digest value by $(t_J.k-\alpha)$ times, else using the left digest value provided;

if $t_J.k < \beta$, hashing the right digest value by $(\beta-t_J.k)$ times, else using the right digest value provided;

and verify the signature provided utilising a public key.

6. The method as claimed in any one of the preceding claims, further comprising re-writing a given query condition in the form of the database query associated with the range (α, β) of the attribute *k*.

7. A system for providing a query result in response to a database query associated with a range (α, β) of an attribute *k*, the system comprising

a first processor unit for sorting real data entries r_i in the database by $r_i.k$, for creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i, for creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i, and for creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for *k* compared to each r_i and the next higher value for *k* compared to r_i respectively;

a second processor unit for providing the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*, the signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function of the real or bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of the real or bound data entry r_{b+1} having the next higher value for *k* compared to r_b.

8. The system as claimed in claim 7, wherein the digest values of the real or bound data entries is based on attribute values of the respective real or bound data entry and a random value attribute assigned to each real or bound data entry.

9. The system as claimed in claims 7 or 8, wherein the signature comprises $sig(r_i) = s(h(h^{r_i.k-r_{i-1}.k}(r_{i-1}) \mid h^0(r_i) \mid h^{r_{i+1}.k-r_i.k}(r_{i+1})))$, the left digest value comprises $h^{\alpha-r_{a-1}.k}(r_{a-1})$, the right digest value comprises $h^{r_{b+1}.k-\beta}(r_{b+1})$, and the

system further comprisies a third processor unit for verifying the query result, the verifying comprising the steps of:

hashing the digest left value by $(r_a.k-\alpha)$ times;

hashing the digest right value by $(\beta-r_b.k)$ times;

and verifying the signatures provided utilising a public key.

10. The system as claimed in claims 7 or 8, wherein the first processor unit creates a unique shadow data entry t_i for each pair of consecutive real data entries (r_i, r_{i+1}), each t_i having a value of $t_i.k$, with $r_i.k < t_i.k < r_{i+1}.k$; and creates a signature for each t_i, where $sig(t_i) = s(h(h^{t_i.k-r_i.k}(r_i) \mid t_i.k \mid h^{r_{i+1}.k-t_i.k}(r_{i+1})))$.

11. The system as claimed in claim 10, wherein, if no real data entry matches the query, the second processor unit selects the shadow data entry t_J, where $r_J.k \le \alpha, \beta \le r_{J+1}.k$, and returns the query result including

the signature $sig(t_J)$;

if $\alpha < t_J.k$, the left digest value equal to $h^{\alpha-r_J.k}(r_J)$, else the left digest value comprising $h^{t_J.k-r_J.k}(r_J)$;

if $\beta > t_J.k$, the right digest value equal to $h^{r_{J+1}.k-\beta}(r_{J+1})$, else the right digest value equal to $h^{r_{J+1}.k-t_J.k}(r_J)$.

12. The system as claimed in claim 11, further comprising a third processor unit for verifying the query result, the verifying comprising the steps of:

if $t_J.k > \alpha$, hashing the left digest value by $(t_J.k-\alpha)$ times, else using the left digest value provided;

if $t_J.k < \beta$, hashing the right digest value by $(\beta-t_J.k)$ times, else using the right digest value provided;

and verify the signature provided utilising a public key.

13. The system as claimed in any one of claims 7 to 12, wherein the second processor unit re-writes a given query condition in the form of the database query associated with the range (α, β) of the attribute *k* for use by the respective processor units.

14. A method of manipulating data in a database for deposition of the database with a publisher processor unit, the method comprising the steps of

sorting real data entries r_i in the database by $r_i.k$, where *k* is an attribute;

creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i;

creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i; and

creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for *k* compared to each r_i and the next higher value for *k* compared to r_i respectively.

15. The method as claimed in claim 14, further comprising creating a unique shadow data entry t_i for each pair of consecutive real data entries (r_i, r_{i+1}), each t_i having a value of $t_i.k$, with $r_i.k < t_i.k < r_{i+1}.k$; and creating a signature for each t_i..

16. A method of providing a query result in response to a database query associated with a range (α, β) of an attribute *k*, the method comprising the steps of

providing matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*,

providing signatures for said matched real data entries, and

providing left and right digest values; wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of

the real or a second bound data entry r_{b+1} having the next higher value for k compared to r_b.

17. The method as claimed in claim 16, wherein the signature comprises $sig(r_i) = s(h(h^{r_i.k-r_{i-1}.k}(r_{i-1}) \mid h^0(r_i) \mid h^{r_{i+1}.k-r_i.k}(r_{i+1})))$, the left digest value comprises $h^{\alpha-r_{a-1}.k}(r_{a-1})$, the right digest value comprises $h^{r_{b+1}.k-\beta}(r_{b+1})$, and wherein, if no real data entry matches the query, the method comprises selecting a shadow data entry t_J, where $r_J.k \le \alpha, \beta \le r_{J+1}.k$, and returning the query result including

a signature $sig(t_J)$;

if $\alpha < t_J.k$, the left digest value equal to $h^{\alpha-r_J.k}(r_J)$, else the left digest value comprising $h^{t_J.k-r_J.k}(r_J)$;

if $\beta > t_J.k$, the right digest value equal to $h^{r_{J+1}.k-\beta}(r_{J+1})$, else the right digest value equal to $h^{r_{J+1}.k-t_J.k}(r_J)$.

18. A method of verifying a query result in response to a database query associated with a range (α, β) of an attribute *k*, the method comprising the steps of

receiving the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*, signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for *k* compared to r_b;

hashing the digest left value;

hashing the digest right value; and

verifying the signatures provided utilising a public key.

19. The method as claimed in claim 18, wherein, if no real data entry matches the query, the query result includes:

a signature $sig(t_J)$, where t_J is a shadow data entry with $r_J.k \le \alpha, \beta \le r_{J+1}.k$;

if $\alpha < t_J.k$, the left digest value equal to $h^{\alpha - r_J.k}(r_J)$, else the left digest value comprising $h^{t_J.k - r_J.k}(r_J)$;

if $\beta > t_J.k$, the right digest value equal to $h^{r_{J+1}.k - \beta}(r_{J+1})$, else the right digest value equal to $h^{r_{J+1}.k - t_J.k}(r_J)$; and the method comprises,

if $t_J.k > \alpha$, hashing the left digest value by $(t_J.k - \alpha)$ times, else using the left digest value provided;

if $t_J.k < \beta$, hashing the right digest value by $(\beta - t_J.k)$ times, else using the right digest value provided;

and verifying the signature provided utilising the public key.

20. A data storage medium having stored thereon computer code means for instructing a computer system to execute a method of providing a query result in response to a database query associated with a range (α, β) of an attribute *k*, the method comprising the steps of:

sorting real data entries r_i in the database by $r_i.k$;

creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i;

creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i;

creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for *k* compared to each r_i and the next higher value for *k* compared to r_i respectively;

providing the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*, the signatures for said matched real data entries, and left and right digest values;

wherein the left digest value comprises a hash function of the real or bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of the real or bound data entry r_{b+1} having the next higher value for *k* compared to r_b.

21. A data storage medium having stored thereon computer code means for instructing a computer system to execute a method of manipulating data in a database for deposition of the database with a publisher processor unit, the method comprising the steps of

sorting real data entries r_i in the database by $r_i.k$, where *k* is an attribute;

creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i;

creating an upper bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i; and

creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for *k* compared to each r_i and the next higher value for *k* compared to r_i respectively.

22. A data storage medium having stored thereon computer code means for instructing a computer system to execute a method of providing a query result in response to a database query associated with a range (α, β) of an attribute *k*, the method comprising the steps of

providing matched real data entries $[r_a,...,r_b]$ sorted in ascending order by *k*,

providing signatures for said matched real data entries, and

providing left and right digest values; wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for *k* compared to r_a, and the right digest value comprises a hash function of

the real or a second bound data entry r_{b+1} having the next higher value for k compared to r_b.

23. A data storage medium having stored thereon computer code means for instructing a computer system to execute a method of verifying a query result in response to a database query associated with a range (α, β) of an attribute k, the method comprising the steps of:

receiving the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k, signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for k compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for k compared to r_b;

hashing the digest left value ;

hashing the digest right value ; and

verifying the signatures provided utilising a public key.

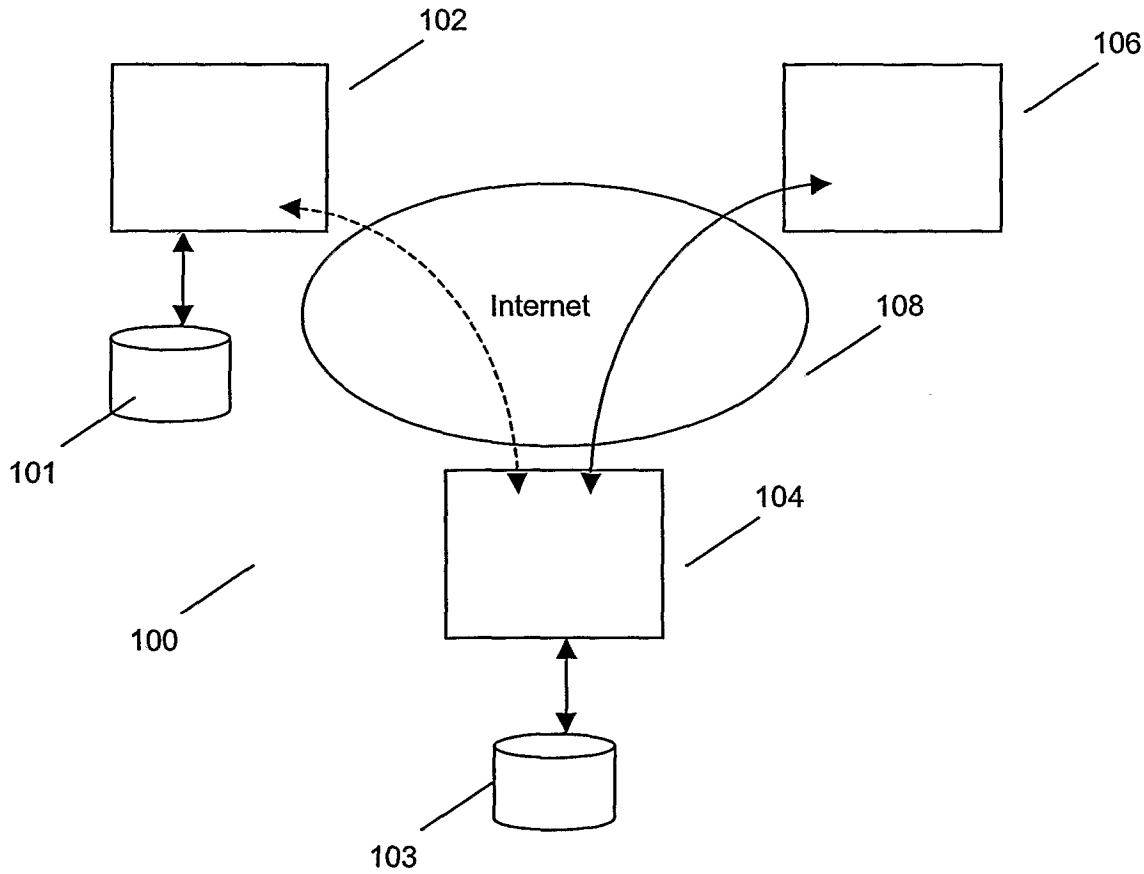


Figure 1

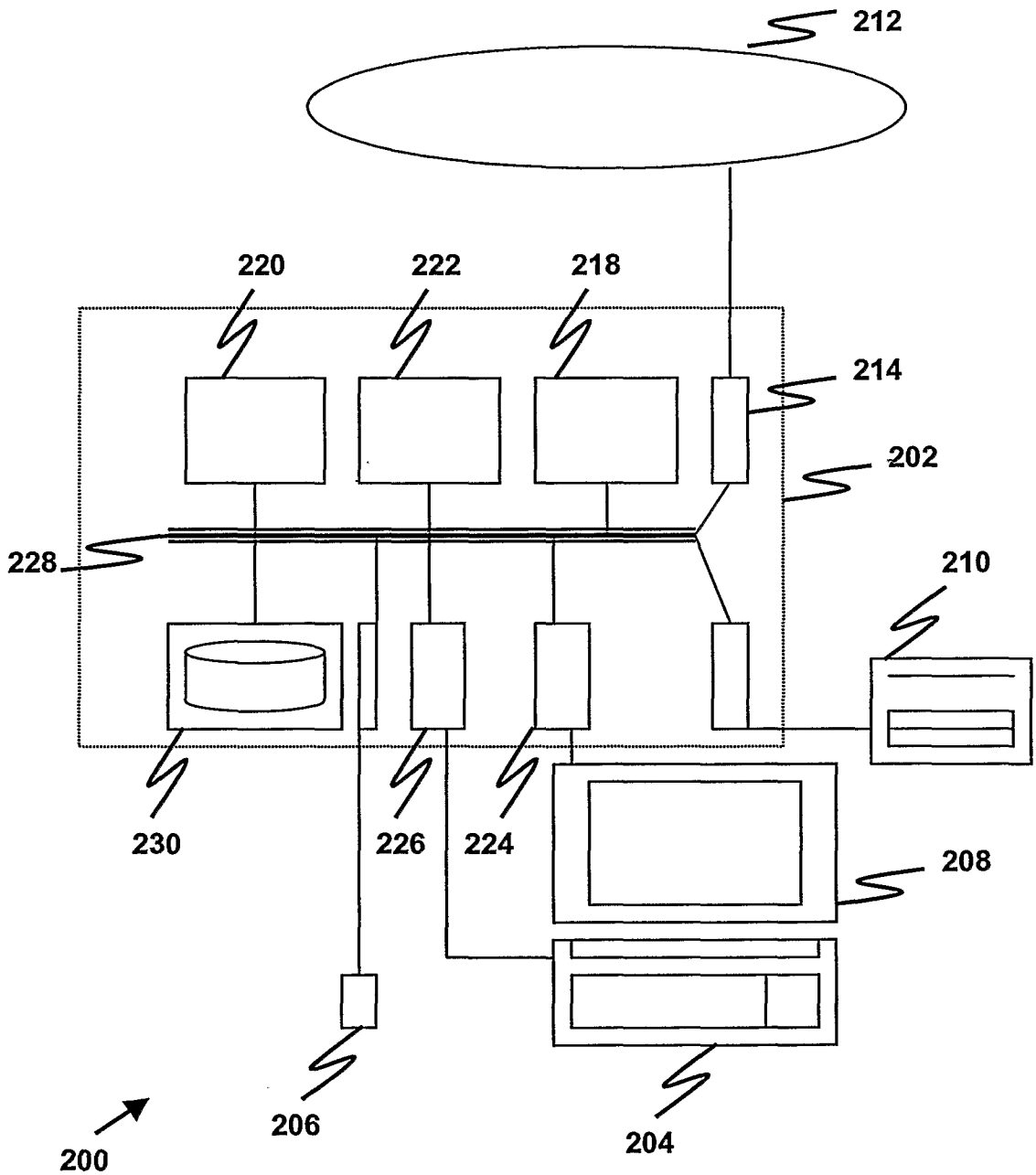


Figure 2

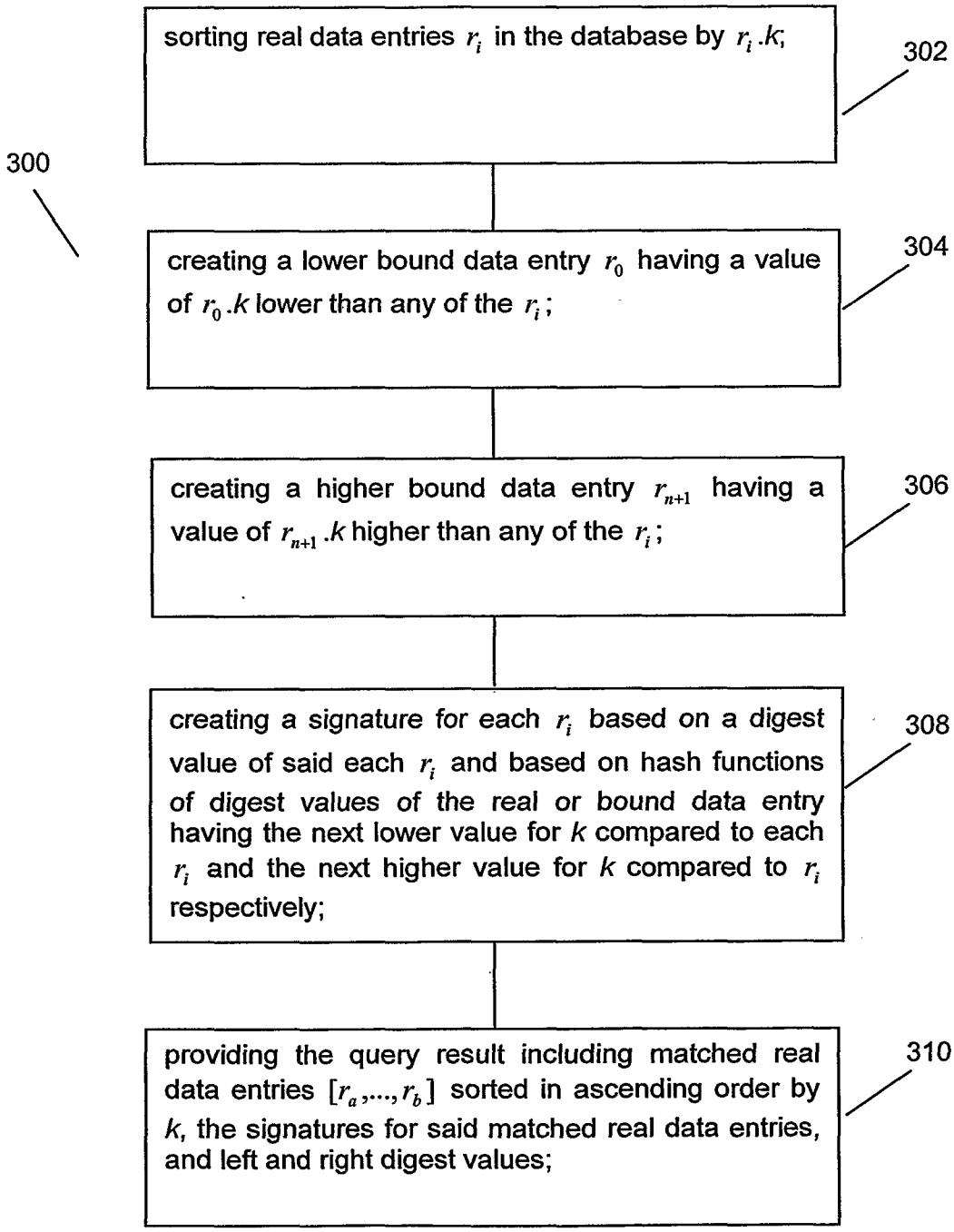

300
sorting real data entries r_i in the database by $r_i.k$;
302
creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i;
304
creating a higher bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i;
306
creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for k compared to each r_i and the next higher value for k compared to r_i respectively;
308
providing the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k, the signatures for said matched real data entries, and left and right digest values;
310

Figure 3

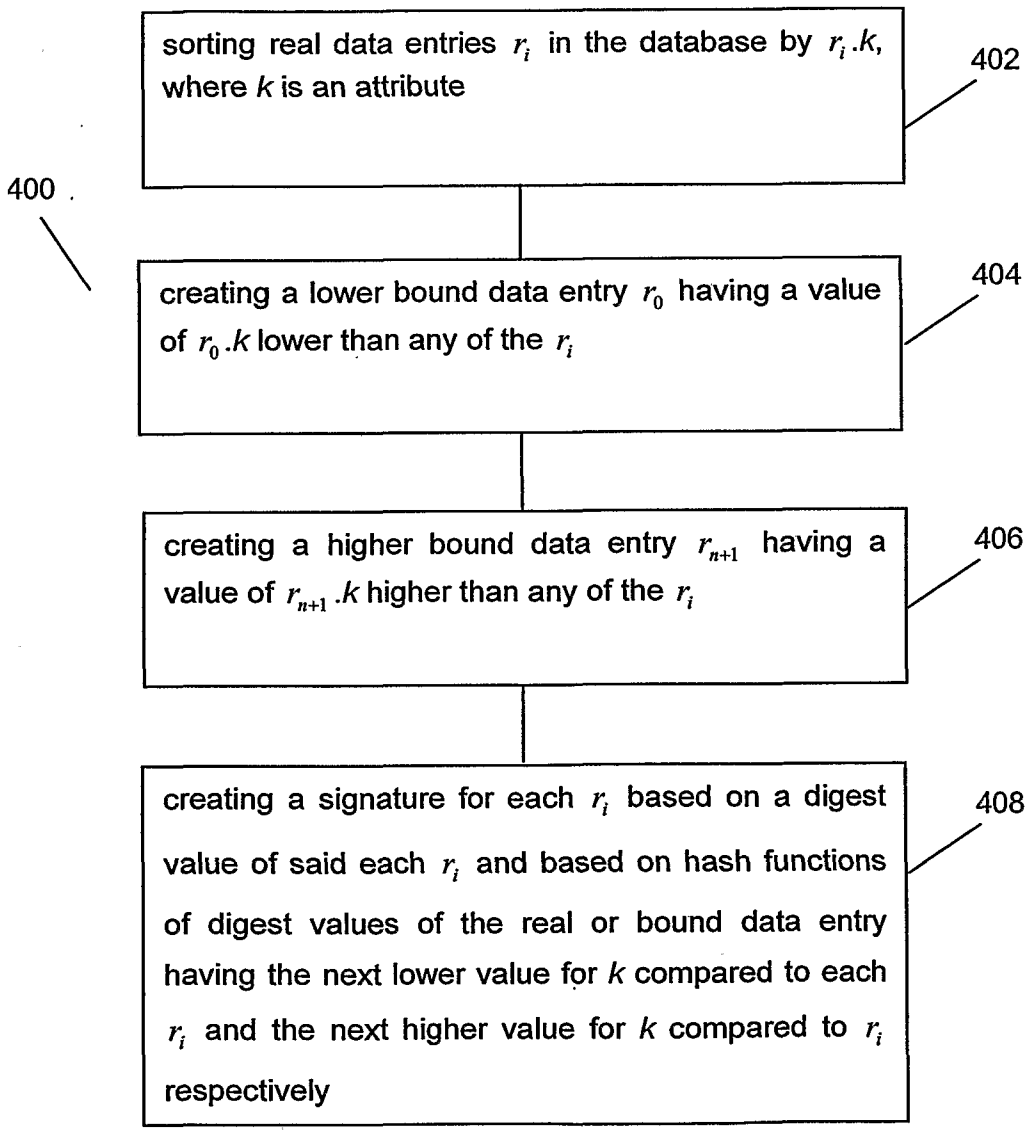

400
sorting real data entries r_i in the database by $r_i.k$, where k is an attribute
402
creating a lower bound data entry r_0 having a value of $r_0.k$ lower than any of the r_i
404
creating a higher bound data entry r_{n+1} having a value of $r_{n+1}.k$ higher than any of the r_i
406
creating a signature for each r_i based on a digest value of said each r_i and based on hash functions of digest values of the real or bound data entry having the next lower value for k compared to each r_i and the next higher value for k compared to r_i respectively
408

Figure 4

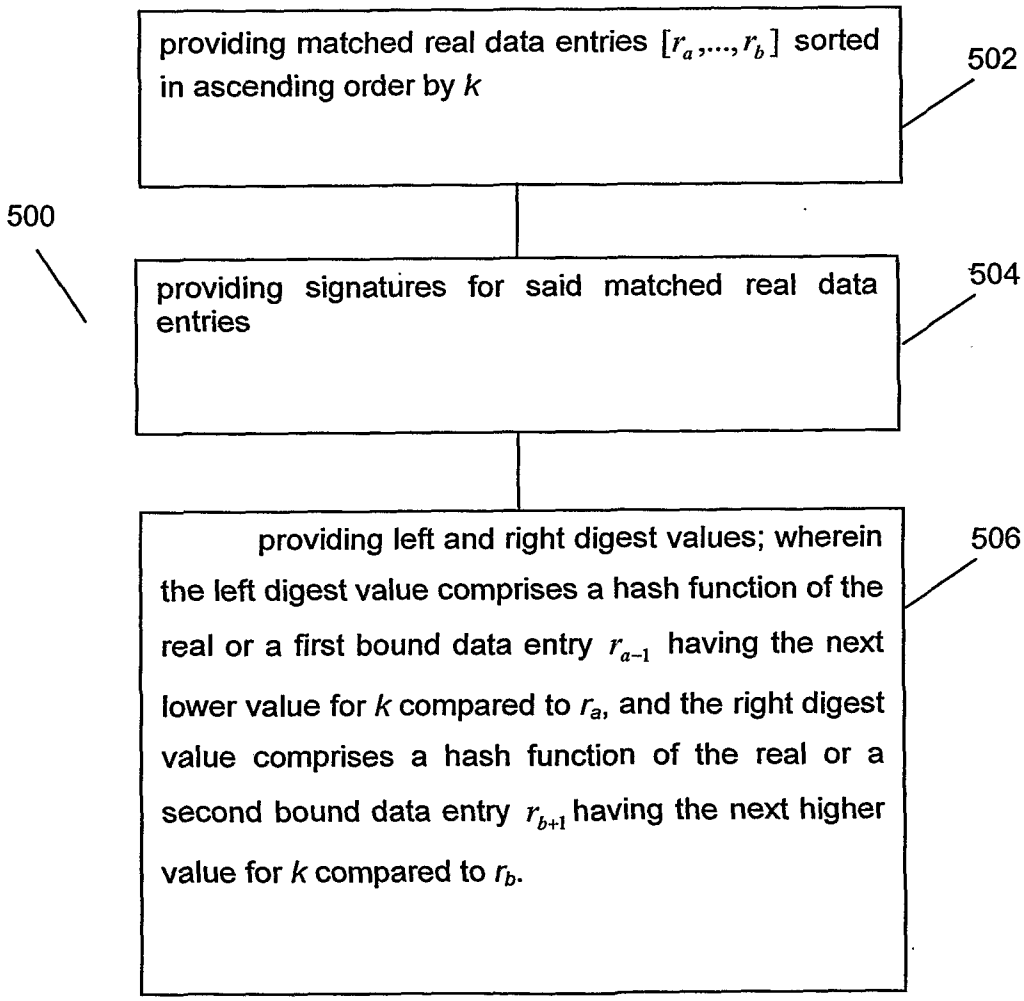

providing matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k
502
500
providing signatures for said matched real data entries
504
providing left and right digest values; wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for k compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for k compared to r_b.
506

Figure 5

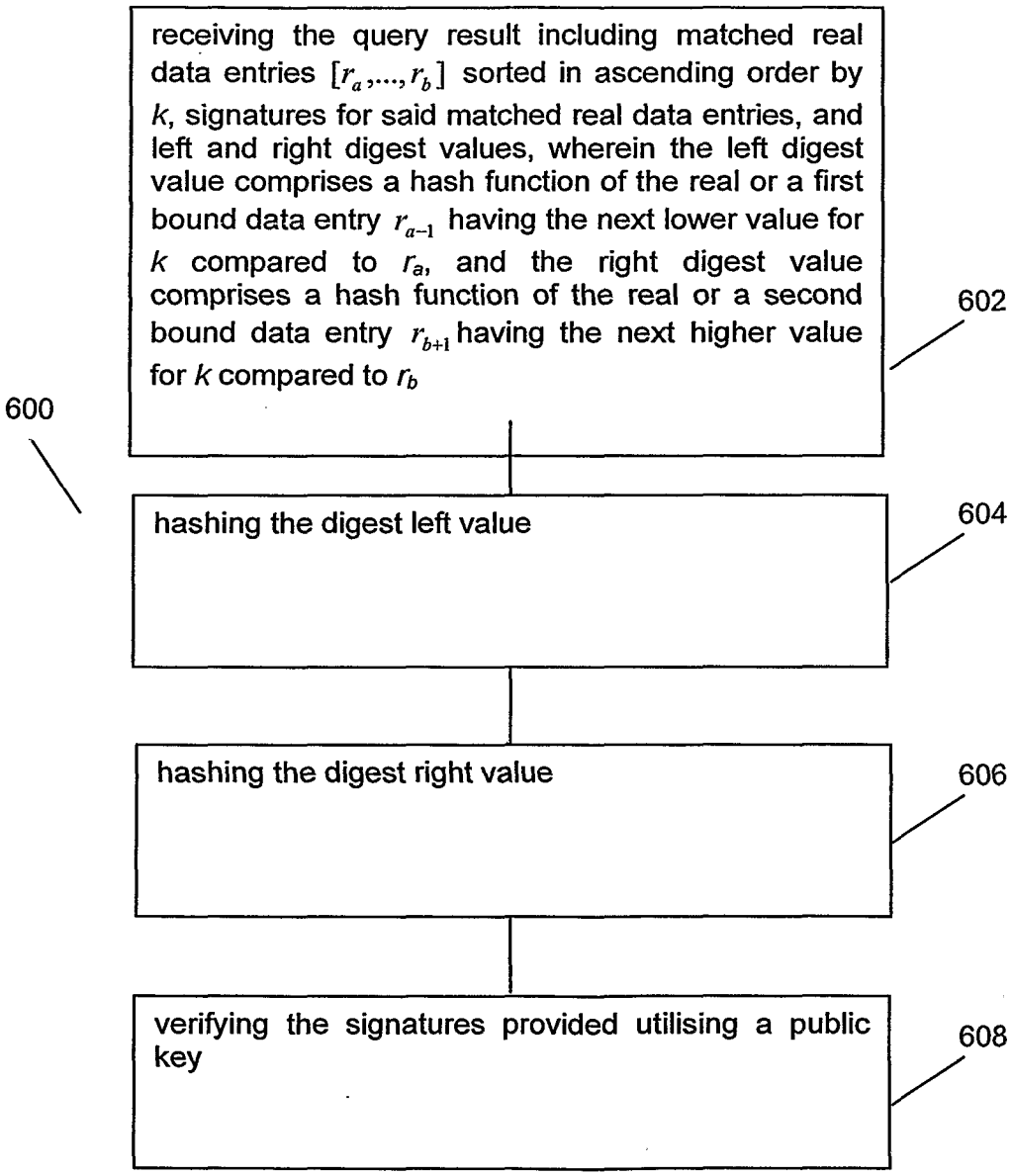

receiving the query result including matched real data entries $[r_a,...,r_b]$ sorted in ascending order by k, signatures for said matched real data entries, and left and right digest values, wherein the left digest value comprises a hash function of the real or a first bound data entry r_{a-1} having the next lower value for k compared to r_a, and the right digest value comprises a hash function of the real or a second bound data entry r_{b+1} having the next higher value for k compared to r_b
602
600
hashing the digest left value
604
hashing the digest right value
606
verifying the signatures provided utilising a public key
608

Figure 6

INTERNATIONAL SEARCH REPORT

International application No.
PCT/SG2006/000126

A. CLASSIFICATION OF SUBJECT MATTER

Int. Cl.

G06F 17/30 (2006.01)

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)
DWPI using keywords including database, hash, signature, verify, sequence and sequential

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	PANG ET AL. 'Verifying Completeness of Relational Query Results in Data Publishing'. In: Proceedings of the 2005 ACM SIGMOD International Conference on Management of Data, September 2005 See particularly sections 3.1 and 4.1	1, 2, 6 to 8, 13, 14, 16, 18 and 20 to 23
A	DEVANBU ET AL. 'Authentic data publication over the Internet'. Journal of Computer Security. 2003, vol. 11, pages 291-394 See whole document	1-23
A	US 2005/0234908 A1 (LOWRANCE ET AL) 20 October 2005 See whole document	1-23

☐ Further documents are listed in the continuation of Box C ☒ See patent family annex

* Special categories of cited documents:

"A" *document defining the general state of the art which is not considered to be of particular relevance*

"E" earlier application or patent but published on or after the international filing date

"L" document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

"O" document referring to an oral disclosure, use, exhibition or other means

"P" document published prior to the international filing date but later than the priority date claimed

"T" *later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention*

"X" document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

"Y" document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

"&" document member of the same patent family

Date of the actual completion of the international search	Date of mailing of the international search report
19 July 2006	04 AUG 2006
Name and mailing address of the ISA/AU AUSTRALIAN PATENT OFFICE PO BOX 200, WODEN ACT 2606, AUSTRALIA E-mail address: pct@ipaustralia.gov.au Facsimile No. (02) 6285 3929	Authorized officer **J.W. THOMSON** Telephone No : (02) 6283 2214

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/SG2006/000126

This Annex lists the known "A" publication level patent family members relating to the patent documents cited in the above-mentioned international search report. The Australian Patent Office is in no way liable for these particulars which are merely given for the purpose of information.

Patent Document Cited in Search Report		Patent Family Member	
US	2005234908	WO	2005098571

Due to data integration issues this family listing may not include 10 digit Australian applications filed since May 2001.

END OF ANNEX