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

(19) World Intellectual Property Organization
International Bureau

(43) International Publication Date
10 May 2007 (10.05.2007)

PCT

(10) International Publication Number
WO 2007/053295 A1

(51) International Patent Classification:
G06F 17/00 (2006.01)

(21) International Application Number:
PCT/US2006/040538

(22) International Filing Date: 16 October 2006 (16.10.2006)

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
11/263,701 1 November 2005 (01.11.2005) US

(71) Applicant *(for all designated States except US)*: **MICROSOFT CORPORATION** [US/US]; One Microsoft Way, Redmond, Washington 98052-6399 (US).

(72) Inventors: **LAUTER, Kristin E.**; One Microsoft Way, Redmond, Washington 98052-6399 (US). **CHARLES, Denis X.**; One Microsoft Way, Redmond, WA 98052 (US). **GOREN, Eyal Zvi**; One Microsoft Way, Redmond, WA 98052 (US).

(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, GT, HN, HR, HU, ID, IL, IN, IS, JP, KE, KG, KM, KN, KP, KR, KZ, LA, LC, LK, LR, LS, LT, LU, LV, LY, MA, MD, MG, MK, MN, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PG, PH, PL, PT, RO, RS, RU, SC, SD, SE, SG, SK, SL, SM, SV, SY, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, ZA, ZM, ZW.

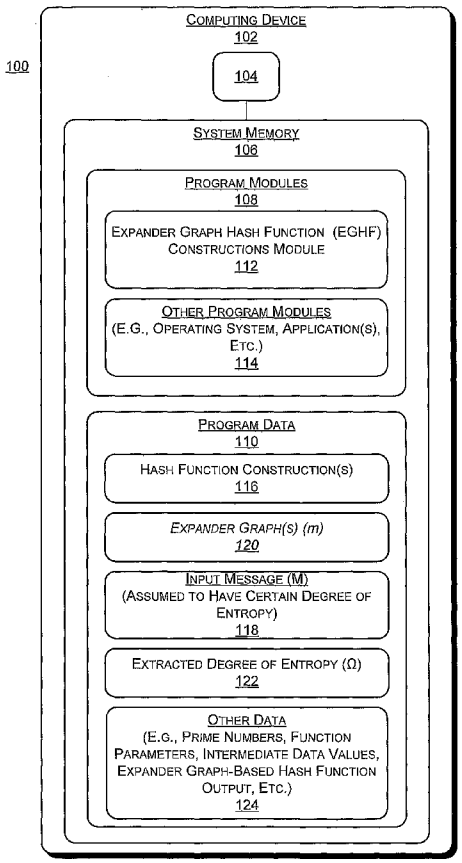
(84) Designated States *(unless otherwise indicated, for every kind of regional protection available)*: ARIPO (BW, GH, GM, KE, LS, MW, MZ, NA, SD, SL, SZ, TZ, UG, ZM, ZW), Eurasian (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), European (AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HU, IE, IS, IT, LT, LU, LV, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Declarations under Rule 4.17:

— *as to applicant's entitlement to apply for and be granted a patent (Rule 4.17(ii))*

[Continued on next page]

(54) Title: HASH FUNCTION CONSTRUCTIONS FROM EXPANDER GRAPHS



(57) Abstract: Hash function constructions from expander graphs are described. In one aspect, an expander graph is walked to compute a hash function. The expander graph is walked using respective subsets of an input message. A label of a last vertex walked is an output of the hash function.

WO 2007/053295 A1

— *as to the applicant's entitlement to claim the priority of the earlier application (Rule 4.17(iii))*

Published:

— *with international search report*

— *before the expiration of the time limit for amending the claims and to be republished in the event of receipt of amendments*

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.

Hash Function Constructions from Expander Graphs

BACKGROUND

[0001] Hash functions constructions are used in many algorithms and cryptographic protocols. They are functions f: U → S with |U| ≥ |S| that distribute their image "uniformly". In other words for most

$$x \in U,\ |\{y \in U \mid f(x) = y\}| \text{ is close to } \frac{|U|}{|S|}.$$

[0002] Hash functions that minimize the number of colliding pairs i.e., pairs (x, y) such that $f(x) = f(y)$ are very useful. For cryptographic applications of hash functions, it is typically desired for the problem of engineering collisions to be hard. This means the task of finding distinct elements x and y such that $f(x) = f(y)$ is computationally hard. Often, there is interest in the following weaker property: Given x finding another y such that $f(x) = f(y)$ is hard.

SUMMARY

[0003] This Summary is provided to introduce a selection of concepts in a simplified form that are further described below in the detailed description. This Summary is not intended to identify key features or essential features of the claimed subject matter, nor is it intended to be used as an aid in determining the scope of the claimed subject matter.

[0004] In view of the above, hash function constructions from expander graphs are described. In one aspect, an expander graph is walked as input to a hash function. The expander graph is walked using respective subsets of an input message. The output of the hash function is the label of the last vertex walked.

BRIEF DESCRIPTION OF THE DRAWINGS

[0005] In the Figures, the left-most digit of a component reference number identifies the particular Figure in which the component first appears.

[0006] Fig. 1 illustrates an exemplary system for hash function constructions from expander graphs, according to one embodiment.

[0007] Fig. 2 shows an exemplary procedure for hash function constructions from expander graphs, according to one embodiment.

[0008] Fig. 3 shows an exemplary procedure for hash function constructions from expander graphs, according to one embodiment.

[0009] Fig. 4 illustrates an example of a suitable computing environment in which hash function constructions from expander graphs may be fully or partially implemented.

DETAILED DESCRIPTION

Overview

[0010] Systems (e.g., systems, apparatus, computer-readable media, etc.) and methods for hash function constructions from expander graphs are described below in reference to Figs. 1 through 4. A hash function is constructed by taking walks on specific expander graphs. A random walk on an expander graph mixes very fast, so the hash function output is generally uniform when the input message is uniformly random. In one implementation, the systems and methods use extractors in conjunction with expander graphs to produce hash functions. In this implementation, input messages have a certain lower bound on the min-entropy. For example, cryptographically signing a message (which is done by hashing) is done after adding a "random pad" to the message. (This process injects entropy into the signature). Under the assumption that the input messages have some small amount of entropy, an extractor is utilized to extract this randomness and then execute a walk according to the output of the extractor.

[0011] These and other aspects of the systems and methods for hash function construction from expander graphs are now described in greater detail.

An Exemplary System

[0012] Although not required, the systems and methods for hash function constructions from expander graphs are described in the general context of computer-executable instructions (program modules) being executed by a computing device such as a personal computer. Program modules generally include routines, programs, objects, components, data structures, etc., that

perform particular tasks or implement particular abstract data types. While the systems and methods are described in the foregoing context, acts and operations described hereinafter may also be implemented in hardware.

[0013] Fig. 1 illustrates an exemplary system 100 for hash function constructions from expander graphs, according to one embodiment. System 100 includes computing device 102, which includes one or more processing units 104 coupled to a system memory 106. Processor 104 fetches and executes computer-program instructions from program modules 108, and fetches and stores data to/from program data 110 portion of system memory 106. Program modules 108 include, for example, expander graph hash function construction module ("EGHF construction module") 112 and other program modules 114. Other program modules 114 include, for example, an operating system and one or more applications that utilize expander graph-based hash function constructions 116 generated by module 112. There are many applications for which such hash function constructions 116 are useful. For example, such constructions may be utilized in one or more applications implementing cryptography, hash tables, error correction, audio identification, Rabin-Karp string search algorithms, etc.

[0014] EGHF construction module 112 generates hash function constructions 116 from an input message 118 and an expander graph 120 of n vertices. Expander graph 118 is a sparse graph with high vertex or edge expansion, or in other words highly connected. In one implementation, expander graph 118 is a Ramanujan graph. In one implementation, the input message 118 has a degree of randomness (or entropy).

[0015] For example, in one implementation, expander graph 120 is determined as follows. Let p be a prime number and let ℓ ($\neq$ p) be another prime number. The expander graph G(p, ℓ) has as its vertex set V the set of supersingular j-invariants over the finite field F_q, $q=p^2$. There is an edge between the vertices j_1 and j_2 if there is an isogeny of degree ℓ between the supersingular elliptic curves whose j-invariants are j_1 and j_2. The graph G(p, ℓ) is known to be a ℓ+1 regular Ramanujan graph. The number of vertices of G(p, ℓ) is the class number of the quaternion algebra $B_{p,\infty}$ which is about p/12. G(p, ℓ) is the expander graph 120.

[0016] In another implementation, expander graph 120 is a Lubotzky-Phillips-Sarnak expander graph, as described below in the section titled "Alternate Embodiments".

[0017] To generate hash function constructions 116, expander graph hash function construction module 112 identifies a message 118. In one implementation, the message has a degree of entropy. EG HF construction module 112 assigns respective names, or labels to each vertex of the n vertices that comprise the expander graph 120. When the input message has a degree of entropy associated with it, EG HF construction module 112 extracts (determines) that degree of randomness with an extractor function. Exemplary such extraction functions and technique to extract randomness from such a message is described in greater detail below in the section titled "Extracting Randomness from the Input".

[0018] Construction module 112 identifies k-length bit segments of the input message 118 based either on the extracted degree of entropy (when present) or other objective criteria (described below), in view of a configurable vertex edge convention to identify vertices of the expander graph 120 to randomly walk (visit). Exemplary operations to walk and expander graph 120 are described in greater detail below in the section titled "Exemplary Procedure". A respective name / label associated with a last vertex of the vertices walked represents the output of the hash function construction 114.

Extracting Randomness from the Input

[0019] Min-Entropy: Let X be a random variable that takes values in $\{0, 1\}^n$. The min-entropy of X is defined to be the quantity

$$\min_{x\in\{0,1\}^n}\left(-\log\left(\Pr[X=x]\right)\right).$$

[0020] Closeness of distributions: Let X and Y be two distributions on $\{0, 1\}^d$. They are said to be ε-close (where ε is a real number) if

$$\max_{x\in\{0,1\}^d}\left|\Pr[X=x]-\Pr[Y=x]\right|\le\varepsilon.$$

[0021] Extractor: A function $Ext:\{0,1\}^n\times\{0,1\}^d\to\{0,1\}^m$ is called a (k,ε)-extractor if for any random variable X on $\{0, 1\}^n$ of min-entropy at least k and U_d the uniform distribution on $\{0,1\}^d$ the distribution $Ext(X,U_d)$ is ε-close to U_m.

[0022] Proposition: If $Ext:\{0,1\}^n \times \{0,1\}^d \rightarrow \{0,1\}^m$ is a (k,ε)-extractor. Then for most choices of the random seed $\sigma \in \{0,1\}^d$ the distribution $Ext(X,\sigma)$ is ε-close to U_m.

[0023] Proof: The distribution $Ext(X,U_d)$ can be described as choosing a distribution uniformly at random among the family X_d of distributions indexed by $\sigma \in \{0,1\}^d$ defined by $X_d = Ext(X,\sigma)$. The fact that Ext is an extractor implies that many of these distributions are ε-close to U_m. (End of proof).

[0024] Constructions of polynomial time extractors are known for any $k > n^\gamma$ $(\gamma < 1)$ and $\varepsilon > 0$ if d is at least $\log^2 n$ and $m = k^{1-\alpha}$ where α is any real number.

Construction of the Hash Function:

[0025] Random variable M (i.e., input message 118), which denotes the inputs to the hash function construction 116, has min-entropy at least $\log^{1+\beta} n$ where n is the number of vertices of G(p, ℓ) and $\beta > 0$. Let $\{0,1\}^N$ be the input space. To determine the degree of entropy 122 of M, construction module 112 implements an extractor function Ext and fixes the function $Ext:\{0,1\}^N \times \{0,1\}^d \rightarrow \{0,1\}^m$ with parameters $k = \log^{1+\beta} n$, ε very small and $m = \Theta(\log^{1+\alpha} n)$. For purposes of exemplary illustration, such parameters are shown as respective portions of "other data" 124. System 100 assumes that $N = k^{O(1)}$. Construction module 112 picks a uniformly at random from $\{0, 1\}^d$. Given an input $x \in \{0,1\}^N$, construction module 112 computes $\varpi = Ext(x,a)$ (i.e., degree of entropy 122). The result of this construction is a string of size m. Construction module 112 executes a walk on m starting at some fixed vertex v_0 following the directions given by ϖ and the output of the hash function 116 is the label of the final vertex in the walk.

[0026] For the expander graph whose nodes are supersingular elliptic curves modulo a prime p, and edges are isogenies of degree ℓ between elliptic curves, we can take steps of a walk around the graph as follows:

[0027] Beginning at a node corresponding to the elliptic curve E, first find generators P and Q of the ℓ-torsion of $E[\ell]$. To this end:

1. Let n be such that $F_q(E[\ell]) \subseteq F_{q^n}$.
2. Let $S = \text{éE}(F_{q^n})$; the number of F_{q^n} rational points on E. (Original)

3. Set $s = S/\ell^k$, where ℓ^k is the largest power of ℓ that divides S (note $k \geq 2$).
4. Pick two points P and Q at random from $E[\ell]$:
   (a) Pick two points U, V at random from $E(F_{q^n})$.
   (b) Set $P' = sU$ and $Q' = sV$, if either P' or Q' equals O then repeat step (i).
   (c) Find the smallest i_1, i_2 such that $\ell^{i_1} P' \neq O$ and $\ell^{i_2} Q' \neq O$ but $\ell^{i_1+1} P' = O$ and $\ell^{i_2+1} Q' = O$.
   (d) Set $P = \ell^{i_1} P'$ and $Q = \ell^{i_2} Q'$.
5. Using the well-known Shanks's Baby-steps-Giant-steps algorithm, determine if Q belongs to the group generated by P. If so, step (d) is repeated.

[0028] The **j**-invariants in F_{p^2} of the $\ell+1$ elliptic curves that are isogenous to E are $j_1, \cdots, j_{\ell+1}$. To find them:

(a) Let $G_1 = \langle Q \rangle$ and $G_{1+i} = \langle P + (i-1)*Q \rangle$ for $1 \leq i \leq \ell$.

(b) For each i, $1 \leq i \leq \ell+1$ compute the **j**-invariant of the elliptic curve E/G_i using Vélu's formulas.

[0029] If we use the graph of supersingular elliptic curves with 2-isogenies, for example, we can take a random walk in the following explicit way: at each step, after finding the 3 non-trivial 2-torsion points of E, order them in terms of their x-coordinates in a pre-specified manner. Then use the input bits to the hash function to determine which point to choose to quotient the elliptic curve by to get to the next node in the walk.

Proof That Output of Hash Function Is Almost Uniform

[0030] By the Proposition the output of the extractor function implemented by expander graph hash function constructions module 112 is close to uniform and the walk we take on the expander graph 120 is very close to being a random walk. (The walk being random just means that being at some vertex v on the graph, we are equally likely to be at any of its neighbors at the next step). Now since the graph G(p, ℓ) has n vertices, and $m = \Omega(\log^{1+\alpha} n)$ the walk mixes

rapidly and the output vertex is very close to uniform. Next, we make the above statements precise. One way to state that a random walk of $O(\log n)$ steps on a d-regular graph G (say) of n vertices mixes rapidly is to say that

$$\left\| \left(\frac{1}{d} A \right)^{O(\log n)} \cdot v - \frac{1}{n} \vec{1} \right\| \leq \varepsilon ,$$

where ε is small, A is the adjacency matrix of G, v may be taken as any of the standard unit vectors and $\vec{1}$ is the vector (1, 1, ..., 1). The matrix

$$\frac{1}{d} A$$

can be thought of as the transition matrix of a uniformly random Markov chain on the graph 120. In this implementation, system 100 implements an almost random walk on the graph 120. This can be thought of as using a matrix B as the transition matrix such that

$$\left\| \frac{1}{d} A - B \right\| \leq \delta$$

and δ is a small real number (where the symbol $\| \ \|$ refers to the matrix norm). In other words, construction module 112 perturbs the random walk a small amount. The following proposition shows that this new random walk mixes quickly if δ can be taken small enough.

[0031] Proposition: Let A and B be two sub-stochastic matrices, then $\left\| A^k - B^k \right\| \leq k \left\| A - B \right\|$.

[0032] Proof: One can write the difference $A^k - B^k$ as

$$\sum_{0 \leq i \leq k-1} A^{k-i-1} (A - B) B^i .$$

Taking norms on both sides and using the fact that $\|A\| = \|B\| = 1$ (as they are sub-stochastic matrices) one gets the result. (End of Proof).

[0033] Since the length of the random walk that we take is O(log n). If we can arrange the parameter δ to be as follows:

$$O\left(\frac{1}{\log^2 n} \right),$$

the resulting approximate random walk will also mix rapidly. This can be arranged by setting the parameter ε of the extractor to be equal to the following:

$$O\left(\frac{1}{\log^2 n} \right).$$

Collision Resistance

[0034] Explicitly finding a collision under this hash function 116 is equivalent to finding two isogenies between a pair of supersingular elliptic curves of the same ℓ-power degree. If the graph $G(p, \ell)$ does not have small cycles then this problem is very hard, since constructing isogenies of high degree between curves is a well-known computationally hard problem.

Alternative Embodiments

[0035] As an alternative to using the graph $G(p, \ell)$ described above, system 100 utilizes the Lubotzky-Phillips-Sarnak expander graph 120. Let ℓ and p be two distinct primes, with ℓ a small prime and p relatively large. We also assume that p and ℓ are $\equiv 1 \bmod 4$ and the ℓ is a quadratic residue mod p (this is the case if $\ell^{(p-1)/2} \equiv 1 \bmod p$). We denote the LPS graph, with parameters ℓ and p, by $X_{\ell,p}$. We define the vertices and edges that make up the graph $X_{\ell,p}$ next. The vertices of $X_{\ell,p}$ are the matrices in $PSL(2,F_p)$, i.e. the invertible 2x2 matrices with entries in F_p that have determinant 1 together with the equivalence relation $A = -A$ for any matrix A. Given a 2x2 matrix A with determinant 1, a name for the vertex will be the 4-tuple of entries of A or those of $-A$ depending on which is lexicographically smaller in the usual ordering of the set $\{0,\ldots,p-1\}^4$. We describe the edges that make up the graph next. A matrix A is connected to the matrices g_iA where the g_i's are the following explicitly defined matrices. Let i be an integer satisfying $i^2 \equiv -1 \bmod p$. There are exactly $8(\ell+1)$ solutions $g = (g_0, g_1, g_2, g_3)$ to the equation $g_0^2 + g_1^2 + g_2^2 + g_3^2 = \ell$. Among these there are exactly $\ell+1$ with $g_0 > 0$ and odd and g_j, for $j = 1, 2, 3$ is even. To each such g associate the matrix

$$\begin{pmatrix} g_0 + ig_1 & g_2 + ig_3 \\ -g_2 + ig_3 & g_0 - ig_1 \end{pmatrix}.$$

[0036] This gives us a set S of $\ell+1$ matrices in $PSL(2,F_p)$. The $\mathbf{g}_i$'s are the matrices in this set S. It is a fact that if $\mathbf{g}$ is in S then so is $\mathbf{g}^{-1}$. Furthermore, since ℓ is small the set of matrices is S can be found by exhaustive search very quickly.

An Exemplary Procedure

[0037] Fig. 2 shows an exemplary procedure 200 for hash function constructions from expander graphs, according to one embodiment. For purposes of exemplary description, the operations of procedure 200 are described with respect to components of system 100 of Fig. 1. The leftmost numeral of a component reference number indicates the particular figure where the component is first described.

[0038] At block 202, EG HF constructions module 112 (Fig. 1) divides an input message 118 into segments. For example, input message has a length N. Given that there are n vertices in a k-regular the expander graph 120 (each vertex having a name / label), the name of each edge coming out of any one vertex will have log k bits. The input message 118 is broken up into chunks of length log k. At block 204, EG HF constructions module 112 walks the expander graph 120 as input to a hash function. The walk is determined as follows: Suppose we are at some vertex v, the next vertex in the walk is determined by reading off the next chunk of log k bits from the input to determine the edge we will traverse out of vertex v, the other end point of this edge will be the next vertex on the walk. For example, EG HF constructions module 112 starts the random walk of edges in the expander graph 120 from a first vertex specified by the first k-bits (segment / chunk) of the input message 118. The next vertex walked in the expander graph 120 is specified by the next chunk of log k-bits. These operations are iteratively performed in view of a convention that specifies how the name of an edge corresponds to the vertices in the expander graph 120. An exemplary such convention is that for each vertex v, there is a function $f_v : \{1,\ldots,k\} \to E$. Thus $f_v(1)$ is the first edge out of v, $f_v(2)$ is the second edge out of v, etc.

[0039] At block 206, EG HF constructions module 112 determines a label of a last vertex walked. At block 208, EG HF constructions module 112 outputs the label as a result of the hash function.

[0040] Fig. 3 shows an exemplary procedure for hash function constructions from expander graphs, according to one embodiment. For purposes of exemplary description, the operations of procedure 300 are described with respect to components of system 100 of Fig. 1. At

block 302, expander graph hash function constructions module ("EGHF constructions module") 112 (Fig. 1), identifies a message 118 with a degree of entropy. At block 304, EGHF constructions module 112 assigns respective labels to each vertex in an expander graph 120. At block 306, EGHF constructions module 112 uses an extractor function to determine the degree of entropy in the input message 118. This determined degree is shown as the extracted degree of entropy 122. At block 308, EGHF constructions module walks the expander graph 120 based on the extracted degree of entropy 122. At block 310, EGHF constructions module 112 outputs a label associated with a last vertex walked and the expander graph 120 as a result of the hash function construction 116. That is, the operations of blocks 302 through 310 correspond to operations of hash function construction 116.

An Exemplary Operating Environment

[0041] Fig. 4 illustrates an example of a suitable computing environment in which hash function constructions from expander graphs may be fully or partially implemented. Exemplary computing environment 400 is only one example of a suitable computing environment for the exemplary system of Fig. 1 and exemplary operations of Figs. 2 and 3, and is not intended to suggest any limitation as to the scope of use or functionality of systems and methods the described herein. Neither should computing environment 400 be interpreted as having any dependency or requirement relating to any one or combination of components illustrated in computing environment 400.

[0042] The methods and systems described herein are operational with numerous other general purpose or special purpose computing system, environments or configurations. Examples of well-known computing systems, environments, and/or configurations that may be suitable for use include, but are not limited to, personal computers, server computers, multiprocessor systems, microprocessor-based systems, network PCs, minicomputers, mainframe computers, distributed computing environments that include any of the above systems or devices, and so on. Compact or subset versions of the framework may also be implemented in clients of limited resources, such as handheld computers, or other computing devices. The invention is practiced in a distributed

computing environment where tasks are performed by remote processing devices that are linked through a communications network. In a distributed computing environment, program modules may be located in both local and remote memory storage devices.

[0043] With reference to Fig. 4, an exemplary system for hash function constructions from expander graphs includes a general purpose computing device in the form of a computer 410 implementing, for example, system 100 of Fig. 1. The following described aspects of computer 410 are exemplary implementations of computing devices 102 of Fig. 1. Components of computer 410 may include, but are not limited to, processing unit(s) 420, a system memory 430, and a system bus 421 that couples various system components including the system memory to the processing unit 420. The system bus 421 may be any of several types of bus structures including a memory bus or memory controller, a peripheral bus, and a local bus using any of a variety of bus architectures. By way of example and not limitation, such architectures may include Industry Standard Architecture (ISA) bus, Micro Channel Architecture (MCA) bus, Enhanced ISA (EISA) bus, Video Electronics Standards Association (VESA) local bus, and Peripheral Component Interconnect (PCI) bus also known as Mezzanine bus.

[0044] A computer 410 typically includes a variety of computer-readable media. Computer-readable media can be any available media that can be accessed by computer 410 and includes both volatile and nonvolatile media, removable and non-removable media. By way of example, and not limitation, computer-readable media may comprise computer storage media and communication media. Computer storage media includes volatile and nonvolatile, removable and non-removable media implemented in any method or technology for storage of information such as computer-readable instructions, data structures, program modules or other data. Computer storage media includes, but is not limited to, RAM, ROM, EEPROM, flash memory or other memory technology, CD-ROM, digital versatile disks (DVD) or other optical disk storage, magnetic cassettes, magnetic tape, magnetic disk storage or other magnetic storage devices, or any other medium which can be used to store the desired information and which can be accessed by computer 410.

[0045] Communication media typically embodies computer-readable instructions, data structures, program modules or other data in a modulated data signal such as a carrier wave or other transport mechanism, and includes any information delivery media. The term "modulated data signal" means a signal that has one or more of its characteristics set or changed in such a manner as to encode information in the signal. By way of example and not limitation, communication media includes wired media such as a wired network or a direct-wired connection, and wireless media such as acoustic, RF, infrared and other wireless media. Combinations of the any of the above should also be included within the scope of computer-readable media.

[0046] System memory 430 includes computer storage media in the form of volatile and/or nonvolatile memory such as read only memory (ROM) 431 and random access memory (RAM) 432. A basic input/output system 433 (BIOS), containing the basic routines that help to transfer information between elements within computer 410, such as during start-up, is typically stored in ROM 431. RAM 432 typically contains data and/or program modules that are immediately accessible to and/or presently being operated on by processing unit 420. By way of example and not limitation, Fig. 4 illustrates operating system 434, application programs 433, other program modules 436, and program data 437.

[0047] The computer 410 may also include other removable/non-removable, volatile/nonvolatile computer storage media. By way of example only, Figure 4 illustrates a hard disk drive 441 that reads from or writes to non-removable, nonvolatile magnetic media, a magnetic disk drive 431 that reads from or writes to a removable, nonvolatile magnetic disk 432, and an optical disk drive 433 that reads from or writes to a removable, nonvolatile optical disk 436 such as a CD ROM or other optical media. Other removable/non-removable, volatile/nonvolatile computer storage media that can be used in the exemplary operating environment include, but are not limited to, magnetic tape cassettes, flash memory cards, digital versatile disks, digital video tape, solid state RAM, solid state ROM, and the like. The hard disk drive 441 is typically connected to the system bus 421 through a non-removable memory interface

such as interface 440, and magnetic disk drive 431 and optical disk drive 433 are typically connected to the system bus 421 by a removable memory interface, such as interface 430.

[0048] The drives and their associated computer storage media discussed above and illustrated in Figure 4, provide storage of computer-readable instructions, data structures, program modules and other data for the computer 410. In Figure 4, for example, hard disk drive 441 is illustrated as storing operating system 444, application programs 443, other program modules 446, and program data 447. Note that these components can either be the same as or different from operating system 434, application programs 433, other program modules 436, and program data 437. Application programs 433 includes, for example program modules 108 of computing device 102 of Fig. 1. Program data 437 includes, for example, program data 110 of computing device 102 of Fig. 1. Operating system 444, application programs 443, other program modules 446, and program data 447 are given different numbers here to illustrate that they are at least different copies.

[0049] A user may enter commands and information into the computer 410 through input devices such as a keyboard 462 and pointing device 461, commonly referred to as a mouse, trackball or touch pad. Other input devices (not shown) may include a microphone, joystick, game pad, satellite dish, scanner, or the like. These and other input devices are often connected to the processing unit 420 through a user input interface 460 that is coupled to the system bus 421, but may be connected by other interface and bus structures, such as a parallel port, game port or a universal serial bus (USB).

[0050] A monitor 491 or other type of display device is also connected to the system bus 421 via an interface, such as a video interface 490. In addition to the monitor, computers may also include other peripheral output devices such as printer 496 and audio device(s) 497, which may be connected through an output peripheral interface 493.

[0051] The computer 410 may operate in a networked environment using logical connections to one or more remote computers, such as a remote computer 480. In one implementation, remote computer 480 represents computing device 102 or networked

computer 104 of Fig. 1. The remote computer 480 may be a personal computer, a server, a router, a network PC, a peer device or other common network node, and as a function of its particular implementation, may include many or all of the elements described above relative to the computer 410, although only a memory storage device 481 has been illustrated in Figure 4. The logical connections depicted in Figure 4 include a local area network (LAN) 471 and a wide area network (WAN) 473, but may also include other networks. Such networking environments are commonplace in offices, enterprise-wide computer networks, intranets and the Internet.

[0052] When used in a LAN networking environment, the computer 410 is connected to the LAN 471 through a network interface or adapter 470. When used in a WAN networking environment, the computer 410 typically includes a modem 472 or other means for establishing communications over the WAN 473, such as the Internet. The modem 472, which may be internal or external, may be connected to the system bus 421 via the user input interface 460, or other appropriate mechanism. In a networked environment, program modules depicted relative to the computer 410, or portions thereof, may be stored in the remote memory storage device. By way of example and not limitation, Figure 4 illustrates remote application programs 483 as residing on memory device 481. The network connections shown are exemplary and other means of establishing a communications link between the computers may be used.

Conclusion

[0053] Although the systems and methods for hash function construction from expander graphs have been described in language specific to structural features and/or methodological operations or actions, it is understood that the implementations defined in the appended claims are not necessarily limited to the specific features or actions described. Rather, the specific features and operations of system 100 are disclosed as exemplary forms of implementing the claimed subject matter.

CLAIMS

1. A computer-implemented method comprising:

walking an expander graph according to input to a hash function, the expander graph being walked using respective subsets of an input message;

determining a label of a last vertex walked; and

outputting the label as a result of the hash function.

2. The method of claim 1, wherein the expander graph is a Ramanujan graph.

3. The method of claim 1, wherein the expander graph is a Lubotzky-Phillips-Sarnak expander graph.

4. The method of claim 1, wherein the expander graph is the graph of supersingular elliptic curves over a finite field of characteristic p.

5. The method of claim 1, wherein the result is a cryptographic hash.

6. The method of claim 1, wherein finding collisions for the hash function is computationally hard.

7. The method of claim 1, wherein the input message has a certain degree of entropy, and wherein the hash function is collision resistant.

8. The method of claim 1, wherein walking further comprises:

dividing the input message into segments; and

determining, for at least a subset of these segments, a path to a next respective vertex in the expander graph based on aspects of a particular segment of a subset.

9. The method of claim 1, wherein the expander graph comprises *n* vertices, wherein the input message has a degree of entropy, and wherein the method further comprises:

assigning a respective label to vertices of the graph;

determining the degree of entropy;

wherein walking further comprises walking the *n* vertices using the degree of entropy to identify completely random vertex output; and

wherein the output is a respective assigned label of a last vertex of the *n* vertices walked.

10. The method of claim 9, wherein determining the degree of entropy further comprises using an extractor function to determine a degree of randomness associated with the input message.

11. A computer-readable medium comprising computer-programmed instructions executable by a processor for:

dividing a message into segments;

walking an expander graph according to input to a hash function, the expander graph being walked using respective ones of the segments to determine a path to a next vertex of n vertices in the expander graph;

determining a label of a last vertex walked; and

outputting the label as a result of the hash function.

12. The computer-readable medium of claim 11, wherein the expander graph is a Ramanujan graph or a Lubotzky-Phillips-Sarnak expander graph.

13. The computer-readable medium of claim 11, wherein the result is a cryptographic hash.

14. The computer-readable medium of claim 11, wherein finding collisions for the hash function is computationally hard.

15. The computer-readable medium of claim 11, wherein the message is divided into the segments based on a degree of entropy extracted from the message.

16. The computer-readable medium of claim 11, wherein the expander graph comprises *n* vertices, wherein the message has a degree of entropy, and wherein the computer-program instructions further comprising structures for:

assigning a respective label to vertices of the graph;

determining the degree of entropy;

wherein walking further comprises walking the *n* vertices using the degree of entropy to identify completely random vertex output; and

wherein the output is a respective assigned label of a last vertex of the *n* vertices walked.

17. The computer-readable medium of claim 11, wherein the computer-programmed instructions for determining the degree of entropy further comprises instructions for using an extractor function to determine a degree of randomness associated with the message.

18. A computing device comprising:

a processor; and

a memory coupled to the processor, the memory comprising computer-program instructions executable by the processor for:

assigning a respective label to respective ones of n vertices in an expander graph;

determining randomness of an input message;

walking the expander graph as input to a hash function, vertices in the expander graph being visited based on the randomness;

determining a label of a last vertex of the vertices walked; and

outputting the label as a result of the hash function.

19. The computing device of claim 18, wherein the expander graph is a Ramanujan graph or a Lubotzky-Phillips-Sarnak expander graph.

20. The computing device of claim 18, wherein the result is a cryptographic hash.

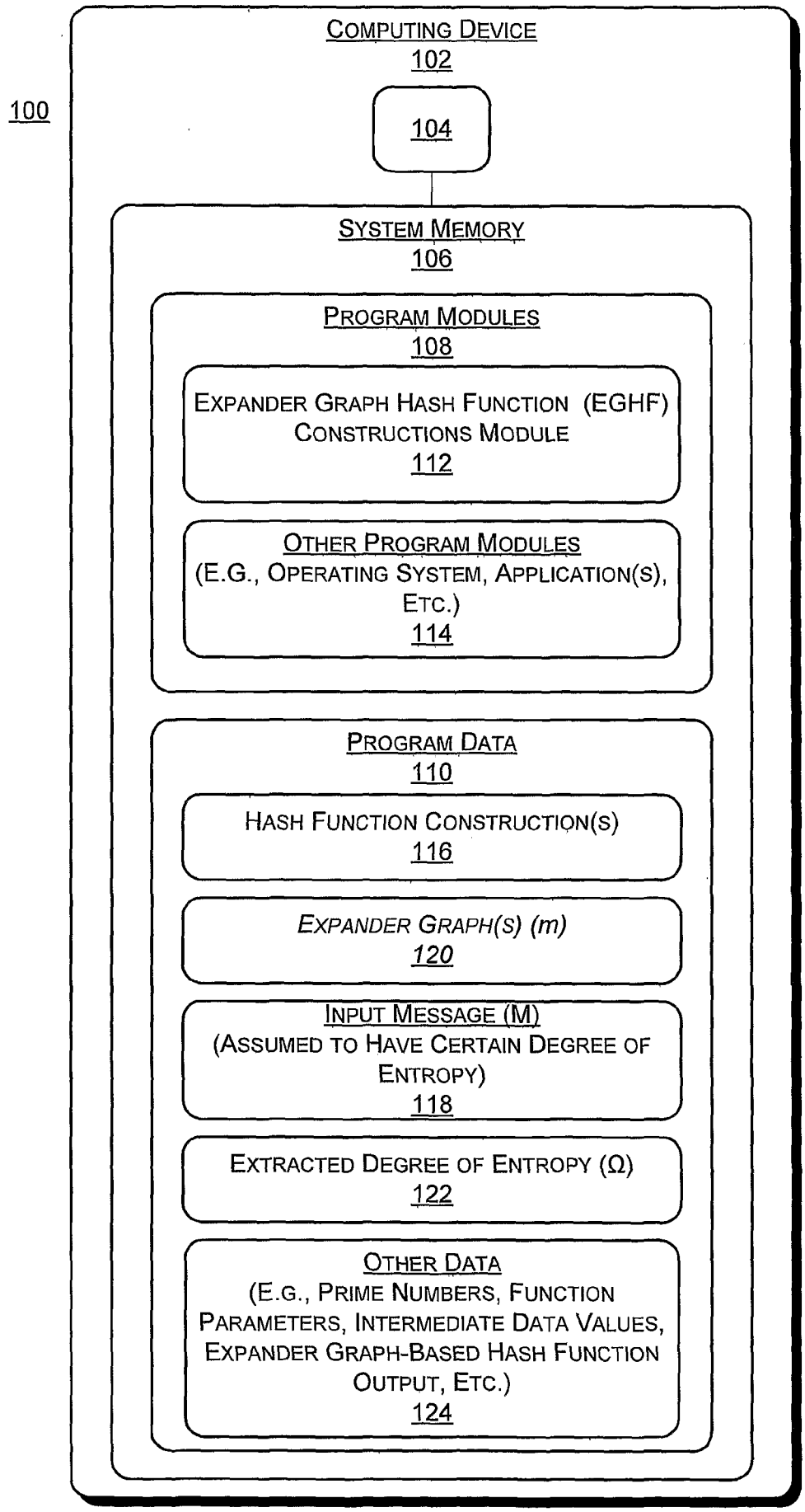

100
COMPUTING DEVICE
102
104
SYSTEM MEMORY
106
PROGRAM MODULES
108
EXPANDER GRAPH HASH FUNCTION (EGHF) CONSTRUCTIONS MODULE
112
OTHER PROGRAM MODULES
(E.G., OPERATING SYSTEM, APPLICATION(S), ETC.)
114
PROGRAM DATA
110
HASH FUNCTION CONSTRUCTION(S)
116
EXPANDER GRAPH(S) (m)
120
INPUT MESSAGE (M)
(ASSUMED TO HAVE CERTAIN DEGREE OF ENTROPY)
118
EXTRACTED DEGREE OF ENTROPY (Ω)
122
OTHER DATA
(E.G., PRIME NUMBERS, FUNCTION PARAMETERS, INTERMEDIATE DATA VALUES, EXPANDER GRAPH-BASED HASH FUNCTION OUTPUT, ETC.)
124

Fig. 1

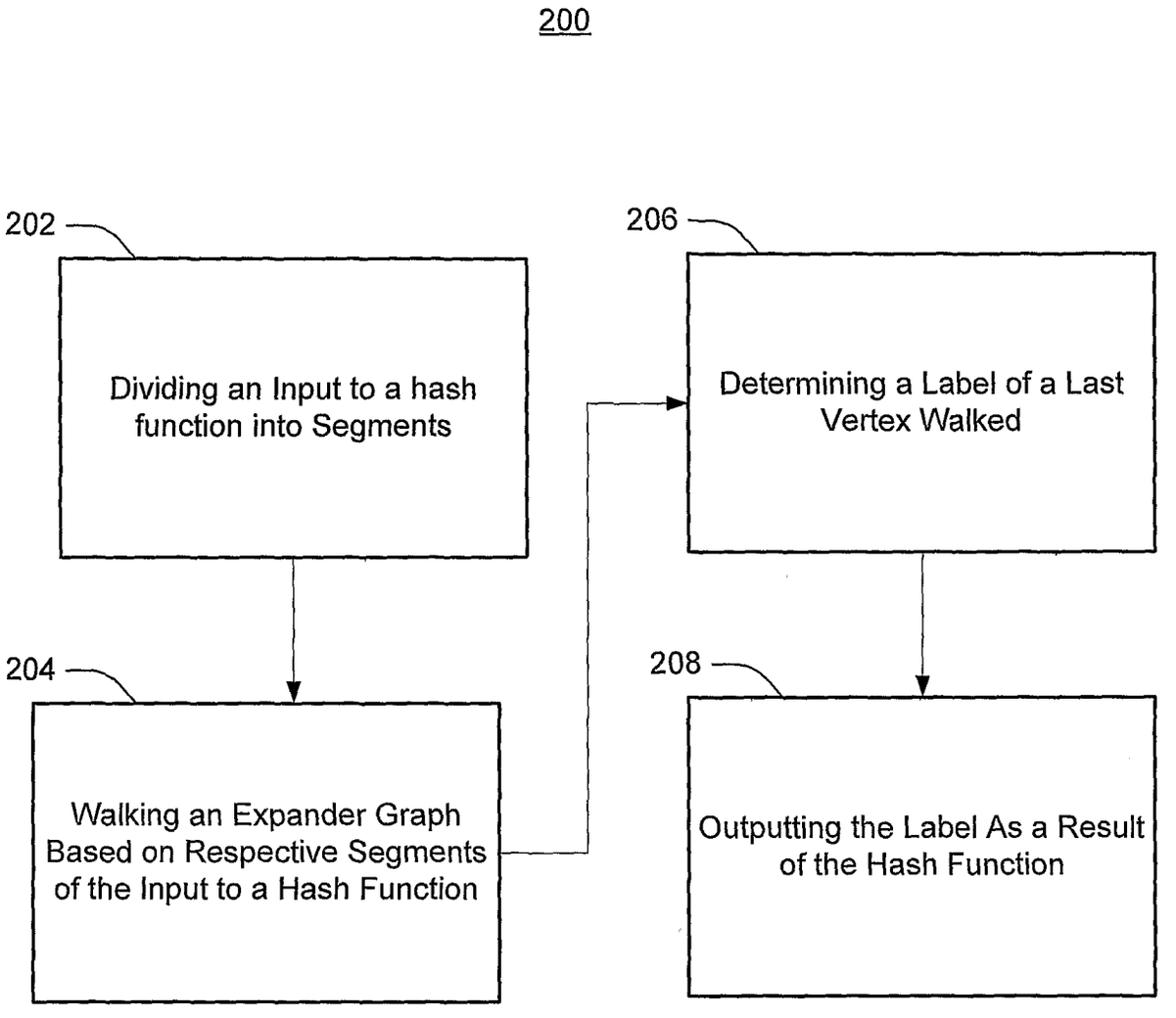

200
202
Dividing an Input to a hash function into Segments
204
Walking an Expander Graph Based on Respective Segments of the Input to a Hash Function
206
Determining a Label of a Last Vertex Walked
208
Outputting the Label As a Result of the Hash Function

Fig. 2

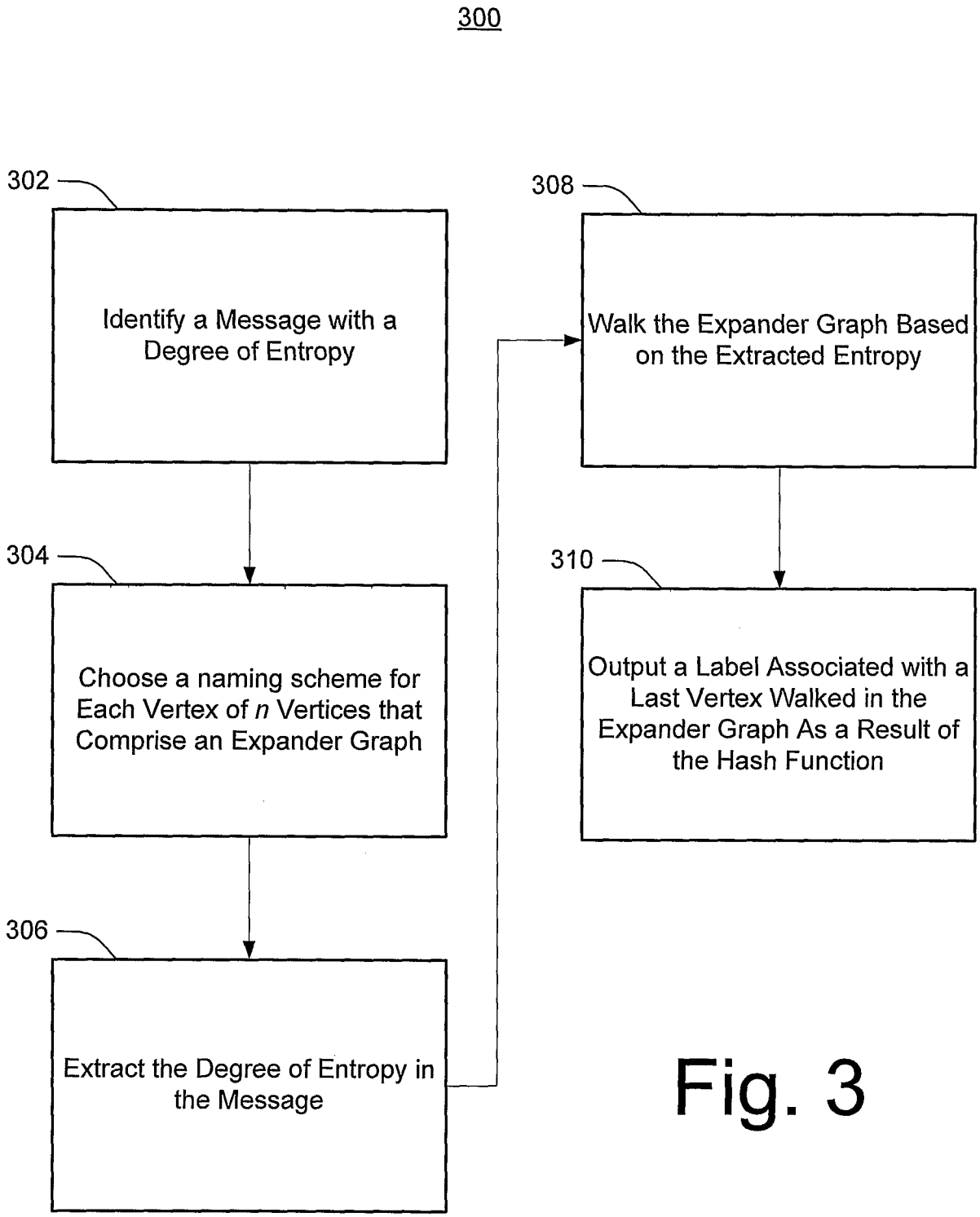

300
302
Identify a Message with a Degree of Entropy
304
Choose a naming scheme for Each Vertex of n Vertices that Comprise an Expander Graph
306
Extract the Degree of Entropy in the Message
308
Walk the Expander Graph Based on the Extracted Entropy
310
Output a Label Associated with a Last Vertex Walked in the Expander Graph As a Result of the Hash Function

Fig. 3

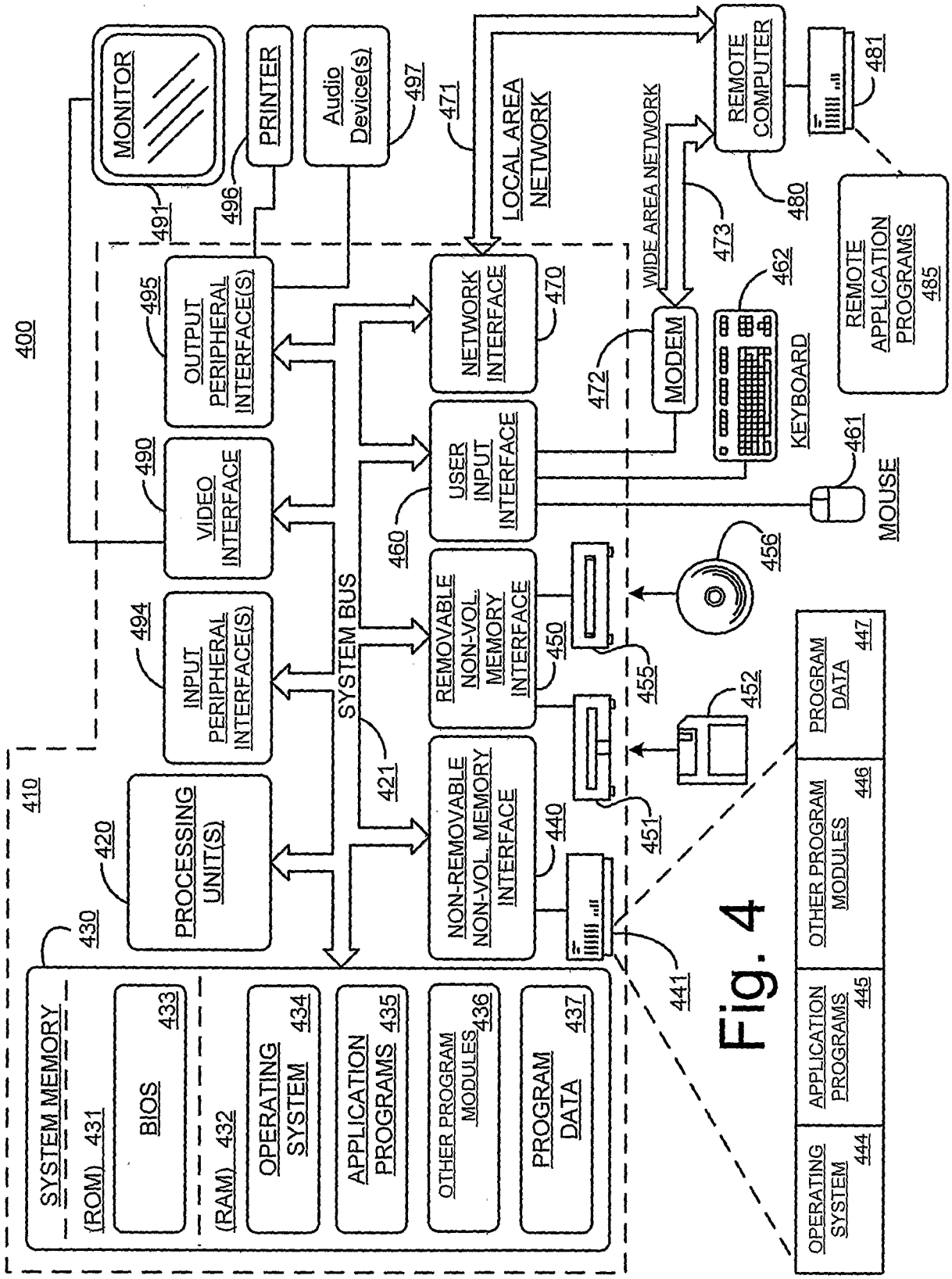

400
410
SYSTEM MEMORY
(ROM) 431
BIOS 433
(RAM) 432
OPERATING SYSTEM 434
APPLICATION PROGRAMS 435
OTHER PROGRAM MODULES 436
PROGRAM DATA 437
430
PROCESSING UNIT(S)
420
INPUT PERIPHERAL INTERFACE(S)
494
VIDEO INTERFACE
490
OUTPUT PERIPHERAL INTERFACE(S)
495
SYSTEM BUS
421
NON-REMOVABLE NON-VOL. MEMORY INTERFACE
440
REMOVABLE NON-VOL. MEMORY INTERFACE
450
USER INPUT INTERFACE
460
NETWORK INTERFACE
470
MONITOR
491
PRINTER
496
Audio Device(s)
497
LOCAL AREA NETWORK
471
WIDE AREA NETWORK
473
MODEM
472
KEYBOARD
462
MOUSE
461
REMOTE COMPUTER
480
481
REMOTE APPLICATION PROGRAMS 485
441
451
455
452
456
OPERATING SYSTEM 444
APPLICATION PROGRAMS 445
OTHER PROGRAM MODULES 446
PROGRAM DATA 447

Fig. 4

INTERNATIONAL SEARCH REPORT

International application No.
PCT/US2006/040538

A. CLASSIFICATION OF SUBJECT MATTER

G06F 17/00(2006.01)i

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)
IPC8 G06F 1/02; G06F 17/30; H04L 9/00; G06F 7/00

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched
Korean patents and applications for inventions since 1975.
Korean utility models and applictions for utility models since 1975.
Japanese utility models and application for utility models since 1975.

Electronic data base consulted during the internationl search (name of data base and, where practicable, search terms used)
e-KIPASS "HASH, EXPANDER, GRAPH"

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
A	US 6 104 811 A (AIELLO, W. A. et al.) 15 August 2000 See abstract; fig. 1; claim 3; claim 4	1-20
A	US 6 757 686 B1 (SYEDA-MAHMOOD, T. F. et al) 29 June 2004 See abstract	1-20
A	US 2005/0175176 A1 (VENKATESAN, R.) 11 August 2005 See abstract; claim 1	1-20
A	US 2005/0071335 A1 (KADATCH, A. V.) 31 March 2005 See abstract	1-20

☐ 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 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
26 FEBRUARY 2007 (26.02.2007)	**26 FEBRUARY 2007 (26.02.2007)**
Name and mailing address of the ISA/KR Korean Intellectual Property Office 920 Dunsan-dong, Seo-gu, Daejeon 302-701, Republic of Korea Facsimile No. 82-42-472-7140	Authorized officer YUK, SEONG WON Telephone No. 82-42-481-8213

Form PCT/ISA/210 (second sheet) (April 2005)

INTERNATIONAL SEARCH REPORT

Information on patent family members

International application No.
PCT/US2006/040538

Patent document cited in search report	Publication date	Patent family member(s)	Publication date
US06104811A	15.08.2000	US6104811A	15.08.2000
US06757686B1	29.06.2004	US6757686BA	29.06.2004
US2005175176A1	11.08.2005	US2005175176AA	11.08.2005
US2005071335A1	31.03.2005	US2005071335AA US6988180BB	31.03.2005 17.01.2006