



(19) **United States**

(12) **Patent Application Publication**
Leggette et al.

(10) **Pub. No.: US 2015/0169897 A1**
(43) **Pub. Date: Jun. 18, 2015**

(54) **EFFICIENT AND SECURE DATA STORAGE UTILIZING A DISPERSED DATA STORAGE SYSTEM**

Publication Classification

(71) Applicant: **CLEVERSAFE, INC., CHICAGO, IL (US)**

(51) **Int. Cl.**
G06F 21/62 (2006.01)
H04L 9/06 (2006.01)
(52) **U.S. Cl.**
CPC **G06F 21/6227** (2013.01); **H04L 9/0618** (2013.01)

(72) Inventors: **Wesley Leggette**, Chicago, IL (US);
Jason K. Resch, Chicago, IL (US)

(73) Assignee: **CLEVERSAFE, INC., CHICAGO, IL (US)**

(57) **ABSTRACT**

(21) Appl. No.: **14/633,315**

(22) Filed: **Feb. 27, 2015**

Related U.S. Application Data

(62) Division of application No. 12/426,727, filed on Apr. 20, 2009.

A method of securely storing data to a dispersed data storage system is disclosed. A data segment is arranged along the columns or rows of an appropriately sized matrix. Data slices are then created based on either the columns or the rows so that no consecutive data is stored in a data slice. Each data slice is then stored in a separate storage node.

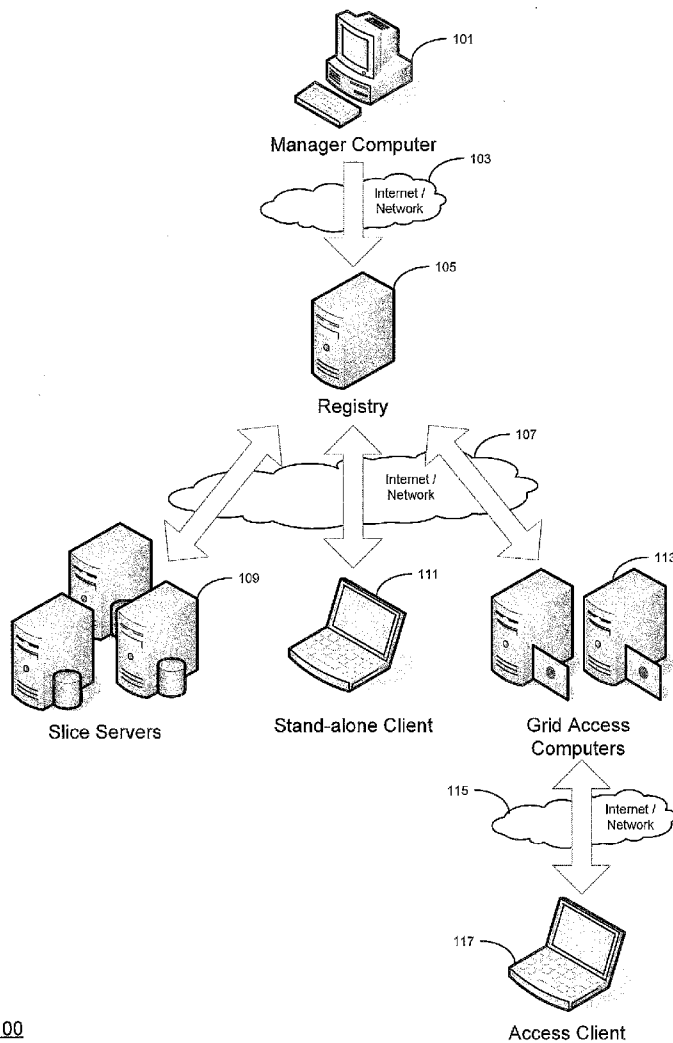


Figure 1

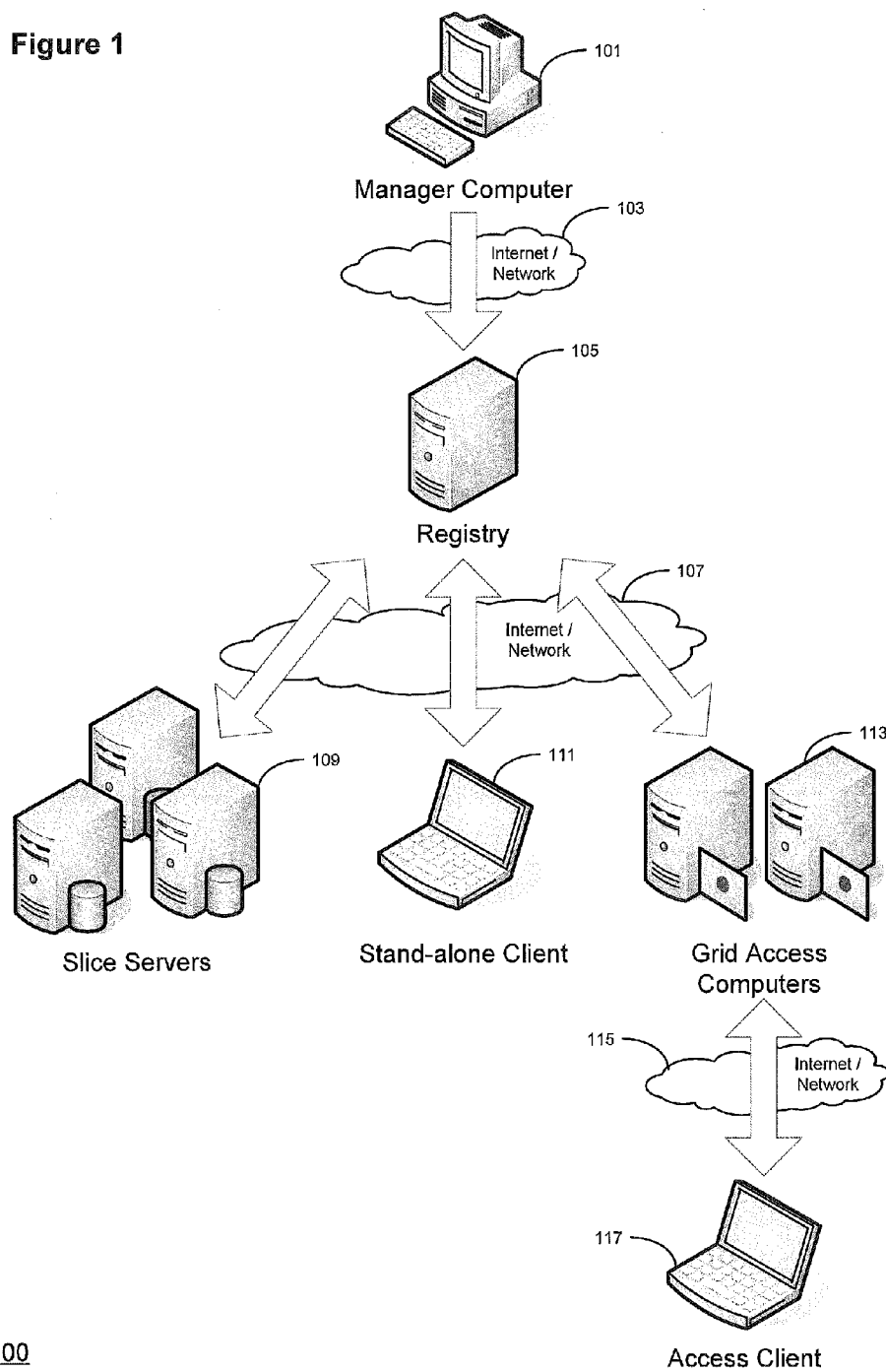


Figure 2

Character:	<u>Q</u>	<u>u</u>	<u>i</u>	<u>c</u>	<u>k</u>	<u>_</u>	<u>b</u>	<u>r</u>	<u>o</u>	<u>w</u>	<u>n</u>	<u>_</u>	<u>f</u>	<u>o</u>	<u>x</u>	<u>!</u>
Position:	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15

Break into slices with a threshold of 8

		floor (Position / 8)		
		0	1	
Position modulus 8	0	Q	o	Data Slice 1
	1	u	w	Data Slice 2
	2	i	n	Data Slice 3
	3	c		Data Slice 4
	4	k	f	Data Slice 5
	5		o	Data Slice 6
	6	b	x	Data Slice 7
	7	r	!	Data Slice 8

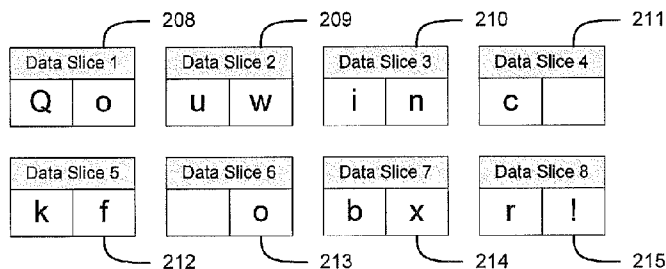


Figure 3

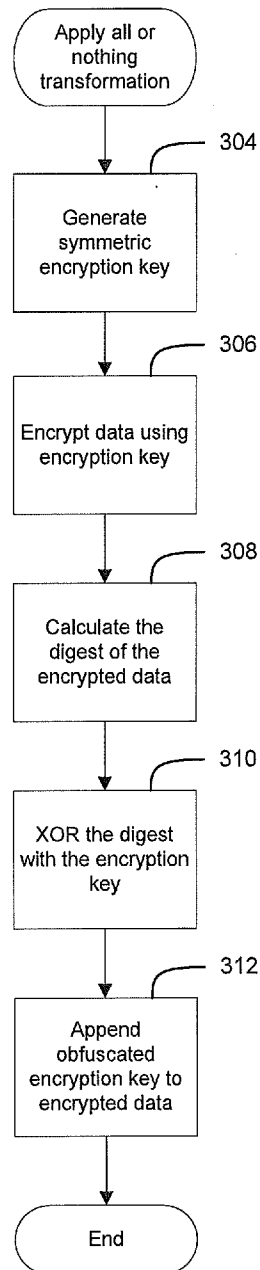


Figure 4

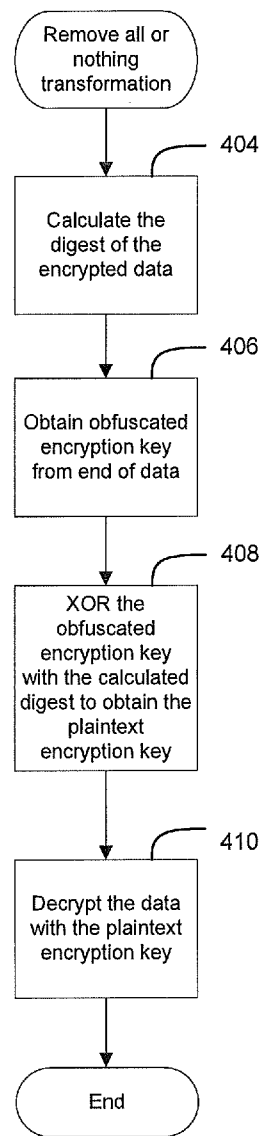


Figure 5

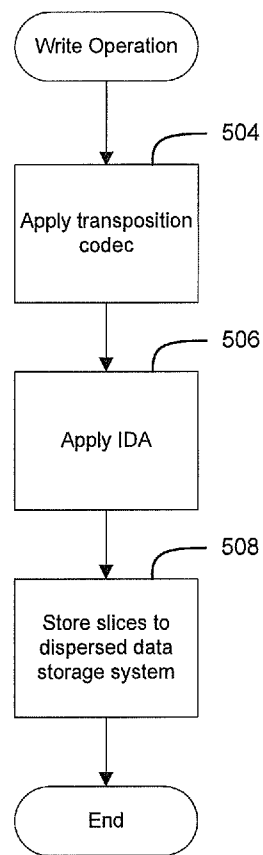


Figure 6

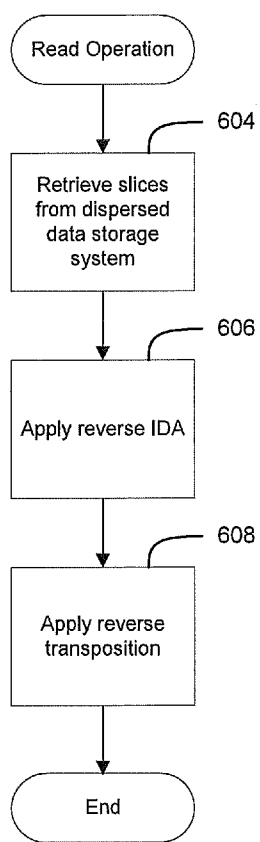


Figure 7

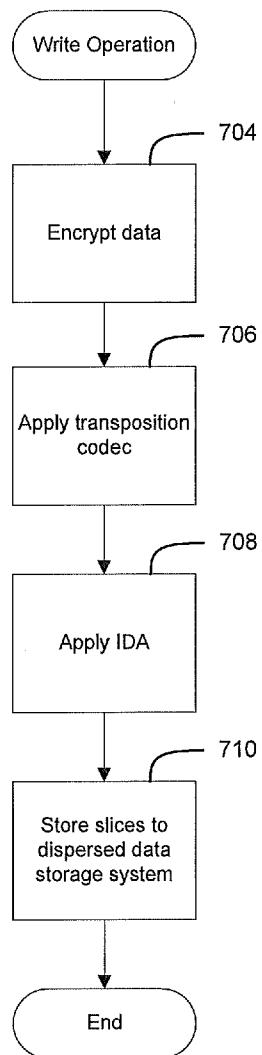


Figure 8

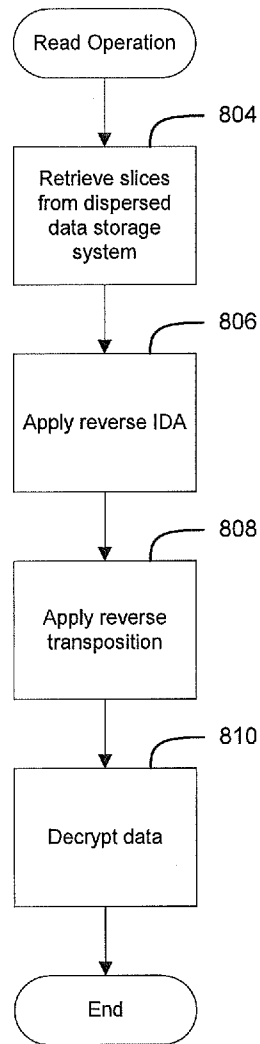


Figure 9

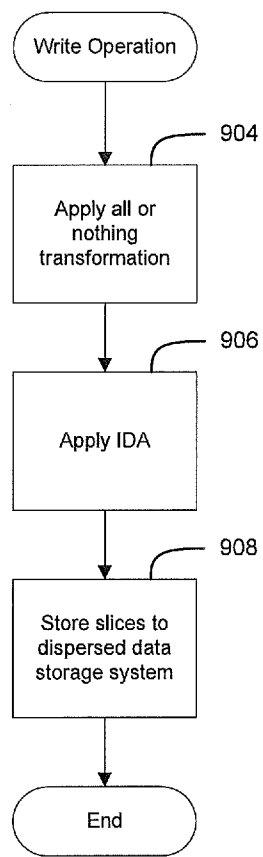


Figure 10

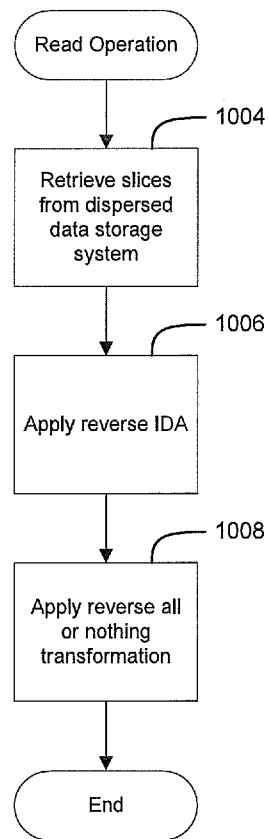


Figure 11

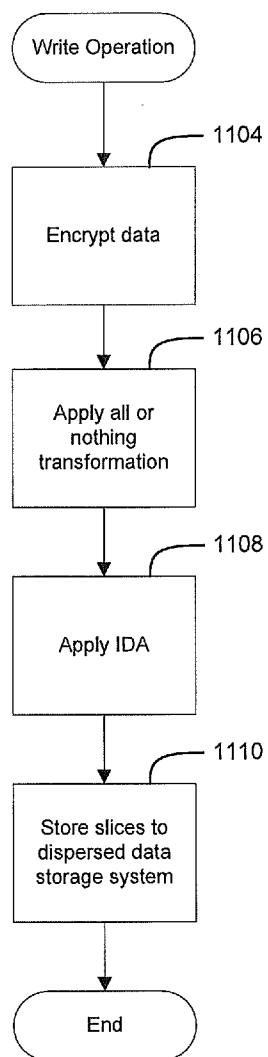
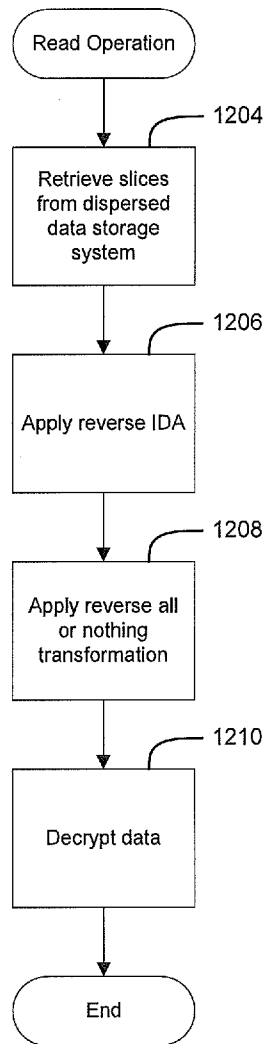


Figure 12



EFFICIENT AND SECURE DATA STORAGE UTILIZING A DISPERSED DATA STORAGE SYSTEM

CROSS-REFERENCE TO RELATED APPLICATIONS

[0001] The present U.S. Utility patent application claims priority pursuant to 35 U.S.C. §121 as a divisional of U.S. Utility application Ser. No. 12/426,727, entitled “EFFICIENT AND SECURE DATA STORAGE UTILIZING A DISPERSED DATA STORAGE SYSTEM”, filed Apr. 20, 2009, which is hereby incorporated herein by reference in its entirety and made part of the present U.S. Utility patent application for all purposes.

FIELD OF THE INVENTION

[0002] The present invention relates generally to systems, apparatus, and methods for securely storing data, and more particularly to systems, apparatus, and methods for secure distributed data storage using an information dispersal algorithm so that no one location will store an entire copy of stored data.

DESCRIPTION OF THE PRIOR ART

[0003] Storing data in digital form is a well-known problem associated with all computer systems, and numerous solutions to this problem are known in the art. The simplest solution involves merely storing digital data in a single location, such as a punch film, hard drive, or FLASH memory device. However, storage of data in a single location is inherently unreliable. The device storing the data can malfunction or be destroyed through natural disasters, such as a flood, or through a malicious act, such as arson. In addition, digital data is generally stored in a usable file, such as a document that can be opened with the appropriate word processing software, or a financial ledger that can be opened with the appropriate spreadsheet software. Storing an entire usable file in a single location is also inherently insecure as a malicious hacker only need compromise that one location to obtain access to the usable file.

[0004] To address reliability concerns, digital data is often “backed-up,” i.e., an additional copy of the digital data is made and maintained in a separate physical location. For example, a backup tape of all network drives may be made by a small office and maintained at the home of a trusted employee. When a backup of digital data exists, the destruction of either the original device holding the digital data or the backup will not compromise the digital data. However, the existence of the backup exacerbates the security problem, as a malicious hacker can choose between two locations from which to obtain the digital data. Further, the site where the backup is stored may be far less secure than the original location of the digital data, such as in the case when an employee stores the tape in her home.

[0005] Another method used to address reliability and performance concerns is the use of a Redundant Array of Independent Drives (“RAID”). RAID refers to a collection of data storage schemes that divide and replicate data among multiple storage units. Different configurations of RAID provide increased performance, improved reliability, or both increased performance and improved reliability. In certain configurations of RAID, when digital data is stored, it is split into multiple stripes, each of which is stored on a separate

drive. Data striping is performed in an algorithmically certain way so that the data can be reconstructed. While certain RAID configurations can improve reliability, RAID does nothing to address security concerns associated with digital data storage.

[0006] One method that prior art solutions have addressed security concerns is through the use of encryption. Encrypted data is mathematically coded so that only users with access to a certain key can decrypt and use the data. While modern encryption methods are difficult to break, numerous instances of successful attacks are known, some of which have resulted in valuable data being compromised. Furthermore, if a malicious hacker should gain access to the encryption key associated with the encrypted data, the entirety of the data is recoverable.

[0007] While modern encryption tends to utilize block ciphers, such as, for example, 3-way, AES, Anubis, Blowfish, BMGL, CAST, CRYPTON, CS-Cipher, DEAL, DES, DESede, DESX, DFC, DFCv2, Diamond2, E2, FROG, GOST, HPC-1, HPC-2, ICE, IDEA, ISAAC, JEROBOAM, LEVIATHAN, LOKI91, LOKI97, MAGENTA, MARS, MDC, MISTY1, MISTY2, Noekeon, Noekeon Direct, Panama, Rainbow, RC2, RC4, RC4-drop, RC5, Rijndael, SAFER-K, SAFER-SK, SAFER+, SAFER++, SAFER++-64, Sapphire-II, Scream, Scream-F, SEAL-3.0, Serpent, SHARK, SKIPJACK, SNOW, SOBER, SPEED, Square, TEA, Twofish, WAKE-CFB, WiderWake4+1, WiderWake4+3, PBE-PKCS5, PBE-PKCS12, etc., other methods have been used in the past. One early form of encoding is transposition. Transposition involves the deterministic swapping of members within a set. For example, if a five member set X is defined as $X=\{a,b,c,d,e\}$, a transposition function σ may be defined as follows:

- $\sigma(0)=a$
- $\sigma(1)=e$
- $\sigma(2)=c$
- $\sigma(3)=d$
- $\sigma(4)=b$

[0008] Therefore, the application of the transposition function to the entire set X would yield a new set $X'=\{a, e, c, d, b\}$.

[0009] By transposing information transmitted in a message, the usability of the transposed information is reduced or eliminated. However, transposition schemes are easily broken by modern computers.

[0010] In 1979, two researchers independently developed a method for splitting data among multiple recipients called “secret sharing.” One of the characteristics of secret sharing is that a piece of data may be split among n recipients, but cannot be known unless at least t recipients share their data, where $n \geq t$. For example, a trivial form of secret sharing can be implemented by assigning a single random byte to every recipient but one, who would receive the actual data byte after it had been bitwise exclusive orred with the random bytes. In other words, for a group of four recipients, three of the recipients would be given random bytes, and the fourth would be given a byte calculated by the following formula:

$$s'=s \oplus r_a \oplus r_b \oplus r_c,$$

where s is the original source data, r_a , r_b , and r_c are random bytes given to three of the four recipients, and s' is the encoded

byte given to the fourth recipient. The original bytes can be recovered by bitwise exclusive-oring all four bytes together.

[0011] A cryptosystem, such as secret sharing, is called information-theoretically secure if its security derives purely from information theory; meaning that its security can be proven even if an adversary has access to unlimited computing power. As a secret sharing scheme can guarantee that no usable information can be recovered unless an attacker gains access to a threshold number of shares, secret sharing is information-theoretically secure. However, each data share is of equal size as the original data, so secret sharing makes for an inefficient storage mechanism.

[0012] All-or-nothing encryption is a recent development in cryptography, with the property that the entire cyphertext must be known before even a portion of the original data can be recovered. The original motivation behind all-or-nothing encryption was to increase the time required by brute force attacks to successfully compromise an encrypted cyphertext by a factor equal to the number of message blocks within the cyphertext. All-or-nothing encryption is described in "All-Or-Nothing Encryption and the Package Transform," by Ronald L. Rivest, which is hereby incorporated by reference. Additional properties of all-or-nothing encryption are described in "Exposure-Resilient Functions and All-Or-Nothing Transforms," by Ran Canetti, Yevgeniy Dodis, Shai Halevi, Eyal Kushilevitz, and Amit Sahai, as well as "On the Security Properties of OAEP as an All-Or-Nothing Transform," by Victor Boyko, both of which are hereby incorporated by reference.

[0013] Dispersed data storage systems involved utilizing an information dispersal algorithm to slice data. Schemes for implementing dispersed data storage systems, such as dispersed data storage networks ("DDSNs"), are also known in the art. For example, U.S. Pat. No. 5,485,474, issued to Michael O. Rabin, describes a system for splitting a segment of digital information into n data slices, which are stored in separate devices. When the data segment must be retrieved, only m of the original data slices are required to reconstruct the data segment, where $n > m$.

[0014] Generally, dispersed data storage systems provide some level of security, as each data slice will contain less information than the original digital information. Furthermore, as each slice is stored on a separate computer, it will generally be harder for a malicious hacker to break into m computers and gather enough data slices to reconstruct the original information. However, depending on the information dispersal algorithm utilized, each data slice will contain up to $1/m$ part of the original data. Generally, the information will be retained in the data slice as it existed in the original digital information. Accordingly, by compromising a storage node, a malicious hacker could access up to $1/m$ part of the original data.

OBJECTS OF THE INVENTION

[0015] Accordingly, it is an object of this invention to provide a system, apparatus, and method for efficiently implementing a secure dispersed data storage system.

[0016] Another object of the invention is to provide a system, apparatus, and method for implementing a secure dispersed data storage system without significantly increasing the amount of stored data.

[0017] Another object of the invention is to provide a system, apparatus, and method for implementing a secure dis-

persed data storage system with a level of security that approaches information theoretic security.

[0018] Another object of the invention is to provide a system, apparatus, and method for implementing a secure dispersed data storage system so that an attacker can gain no information about a stored data segment unless the attacker is able to compromise a threshold number (m) data slices.

[0019] Other advantages of the disclosed invention will be clear to a person of ordinary skill in the art. It should be understood, however, that a system, method, or apparatus could practice the disclosed invention while not achieving all of the enumerated advantages, and that the protected invention is defined by the claims.

SUMMARY OF THE INVENTION

[0020] The disclosed invention achieves its objectives by providing a method for securely storing data to a multi-node storage system, the method comprising the steps of arranging a data segment comprised of a plurality of data units in accordance with a matrix having j rows and k columns so that each data unit of the data segment is associated with a position of the matrix. The arranged data segment is then sliced in accordance with either the rows or the columns of the matrix to produce a plurality of data slices so that each data slice does not contain consecutive data units as they were arranged in the original data segment. Each of the data slices is then stored to a separate storage node.

[0021] In a further embodiment, a data string is padded with pad data prior to the step of arrangement as described above.

[0022] In an alternative embodiment, a method for securely storing data using a multi-node storage system is described, the method starting with the step of encrypting a data segment comprised of a plurality of data units using a suitable block cipher. The encrypted data segment is then arranged in accordance with a matrix having j rows and k columns so that each data unit of the encrypted data segment is associated with a position of the matrix. The encrypted arranged data segment is then sliced in accordance with either the rows or the columns of the matrix to produce a plurality of data slices so that each data slice does not contain consecutive data units as they were arranged in the original data segment. Each of the data slices is then stored to a separate storage node.

[0023] In a further embodiment of the disclosed invention, a method for securely storing data using a multi-node storage system comprises the steps of applying an all-or-nothing transformation to a data segment, slicing the all-or-nothing encrypted data segment using an information dispersal algorithm, and storing each of the resultant plurality of data slices on a separate storage node.

[0024] In a still further embodiment of the disclosed invention, a method of securely storing data using a multi-node storage system comprises the steps of encrypting a data segment using a first block cipher, applying an all-or-nothing transform to the encrypted data segment, slicing the all-or-nothing encrypted data segment using an information dispersal algorithm, and storing each of the resultant data slices on a separate storage node.

[0025] The application of an all or nothing transform can comprise the steps of generating an encryption key using, for example, a random key generator, and encrypting a data segment using the generated encryption key. The digest of the encrypted data is then calculated using an appropriate hashing algorithm, and the encryption key is exclusive-ored with the digest to produce an obfuscated encryption key. The

obfuscated encryption key is then packaged with the data segment by, for example, appending the encryption key to the end of the data segment.

[0026] The disclosed invention can be operated by different computers. For example, in one embodiment of the disclosed invention, an access computer comprises a network port and a processor. The processor receives a data segment from the network port, arranges the data segment in accordance with a matrix having j rows and k columns, and slices the arranged data segment in accordance with the rows or columns of the matrix. The processor then sends each data slice to a separate storage node. In a separate embodiment, a stand-alone client comprises a network port and a processor. The processor retrieves a data segment and arranges the data segment in accordance with a matrix having j rows and k columns. The processor then slices the arranged data segment in accordance with the rows or columns of the matrix, and sends each data slice to a separate storage node.

BRIEF DESCRIPTION OF THE DRAWINGS

[0027] Although the characteristic features of this invention will be particularly pointed out in the claims, the invention itself, and the manner in which it may be made and used, may be better understood by referring to the following description taken in connection with the accompanying drawings forming a part hereof, wherein like reference numerals refer to like parts throughout the several views and in which:

[0028] FIG. 1 is a network diagram of a dispersed data storage system utilizing the disclosed security schemes;

[0029] FIG. 2 is an illustration of the principles of transposition as applied to a dispersed data storage system;

[0030] FIG. 3 is a flowchart illustrating the application of an all-or-nothing transformation to a data segment;

[0031] FIG. 4 is a flowchart illustrating the removal of an all-or-nothing transformation from a data segment;

[0032] FIG. 5 is a flow chart illustrating a write operation to a dispersed data system utilizing a columnar transposition cipher and an information dispersal algorithm;

[0033] FIG. 6 is a flow chart illustrating a read operation from a dispersed data storage system utilizing a columnar transposition cipher and an information dispersal algorithm;

[0034] FIG. 7 is a flow chart illustrating a write operation to a dispersed data storage system utilizing encryption, transposition, and an information dispersal algorithm;

[0035] FIG. 8 is a flow chart illustrating a read operation from a dispersed data storage system utilizing encryption, transposition, and an information dispersal algorithm;

[0036] FIG. 9 is a flow chart illustrating a write operation to a dispersed data storage system utilizing an all-or-nothing transformation and an information dispersal algorithm;

[0037] FIG. 10 is a flow chart illustrating a read operation from a dispersed data storage system utilizing an all-or-nothing transformation and an information dispersal algorithm;

[0038] FIG. 11 is a flow chart illustrating a write operation to a dispersed data storage system utilizing encryption, an all-or-nothing transformation, and an information dispersal algorithm; and

[0039] FIG. 12 is a flow chart illustrating a read operation from a dispersed data storage system utilizing encryption, an all-or-nothing transformation, and an information dispersal algorithm.

DETAILED DESCRIPTION OF THE ILLUSTRATED EMBODIMENT

[0040] Turning to the Figures, and to FIG. 1 in particular, a dispersed data storage system **100** is shown. An arbitrary number of storage nodes, such as slice servers **109** store data slices sent to them by source computers **111**, **117**. Storage nodes **109** may be networked slice servers as illustrated, or may merely be a collection of drives. In a networked implementation, some number of grid access computers **113** may serve access clients **117** in providing access to the storage nodes **109**. Alternatively, the source computers may include the software required to access the storage nodes **109** directly, such as stand-alone client **111**. All of the computers may be general purpose computers, comprised of, for example, a housing containing a processor, fast memory, such as dynamic RAM, one or more storage drives, such as rotating media drives or flash drives, a plurality of input/output ports, such as USB ports or Firewire ports, one or more network ports, such as Ethernet ports or 802.11 ports, as well as an external display, and one or more input/output devices, such as a keyboard for data entry, and a mouse or touchpad for cursor control. Alternatively, access computers and storage nodes may be thinner devices. For example, an access computer may be comprised of a housing containing a processor, fast memory, and one or more network ports. Further by way of example, a storage node could be comprised of a housing containing a processor, fast memory, one or more network ports, and one or more storage drives.

[0041] FIG. 2 presents an overview of a process to secure a data segment in accordance with an embodiment of the disclosed invention. A data segment **202** is comprised of 16 bytes of ASCII data expressing the phrase "Quick brown fox!" It should be noted that the type of encoding of the data, as well as the size of a data unit, is irrelevant to the invention. Therefore, Unicode instead of ASCII could be used to encode the phrase, and the size of a data unit could be set to word (16 bits), double word (32 bits), or any other size. Furthermore, a data segment could be 32 bytes, 48 bytes, etc.

[0042] Where a higher level of security is required, the data segment **202** can be encrypted using an appropriate block cipher, such as DES or AES. While the use of encryption will increase security for the data storage system, it will also increase processor load on computers accessing the storage system. Accordingly, the performance of the system will be lowered, as computers writing data to the dispersed data storage system will need to encrypt data prior to writing it out, and systems reading data will likewise need to decrypt data. For those systems where high performance is more important than high security, the system administrator can elect to disable encryption. In this case, a moderate level of security is still attained by the disclosed system through the use of transposition, as explained below.

[0043] Regardless of whether the data segment is encrypted or not, the data within the data segment is arranged in a matrix **206**. The matrix **206** is sized so that (1) every element of the data segment is assigned to a particular matrix entry, and (2) the number of data slices created is a multiple of the number of data slices created per data segment. In the depicted example, which assumes that eight data slices are created per data segment, an 8×2 matrix is used to fit the 16 data unit data segment, with the data segment arranged sequentially along the columns.

[0044] The data is then dispersed into data slices **208-215**, each containing one row of data. As depicted, each data slice **208-215** contains entirely non-consecutive data from the original data segment.

[0045] A variety of sizes of matrices can be used to achieve many of the advantages of the disclosed system. For example, for a 16 byte data segment being stored on a system that slices data into 4 data slices, a 4x4 matrix could be used; data could be arranged along either the rows or columns, with the other serving as the basis for the data slices. However, while such an arrangement would increase security for the stored information, as no consecutive data units would be stored in a single slice, the optimal increase in security is achieved by sizing the matrix so that one dimension of the matrix, rows or columns, is equal to the threshold of the dispersed data storage network. For example, in system with a threshold m of eight and the data segment size is set to 16 bytes, an 8x2 matrix could be used as described above. It should be noted that additional, code slices containing redundant data would be generated if the total slice number n for the system is above eight. In this case, if a malicious hacker should recover two consecutive slices, a minimal number of consecutive data units will be recovered, i.e., two strips of data, each two data units in length.

[0046] Persons of skill in the art will realize that the decision to arrange data along the columns of the matrix is an arbitrary decision. For example, a 2x8 matrix could be used, and data could be arranged along the rows, instead of the columns. The data slices would then be made from the columns.

[0047] FIG. 3 depicts a method for applying an all-or-nothing transformation to a data segment. In a first step **304**, a symmetric encryption key is generated. In step **306**, the data segment is encrypted using the generated encryption key. In step **308**, the digest of the encrypted data is calculated, by applying a hashing algorithm to the data segment; suitable hashing algorithms include MD5, SHA-1, SHA-2, and any other secure cryptographic hashing algorithm. The digest is then XOR-ed with the encryption key in step **310**, and the obfuscated encryption key is appended to the data segment in step **312**. From this process, it is apparent that the encryption key generated in step **304** is not "secret information," as it will be appended to the data with trivial protection.

[0048] FIG. 4 depicts a method for removing an all-or-nothing transformation from a data segment. In step **404**, the digest of the encrypted data is calculated; note that the obfuscated digest placed at the end of the data segment in the method of FIG. 3 is not included in this calculation. It should also be noted that the plaintext encryption key is never stored, but rather, is calculated in memory, and obfuscated before it is appended to the data segment. In step **406**, the obfuscated encryption key is read into a memory location, or otherwise obtained, and in step **408**, the digest is XOR-ed with the obfuscated encryption key to obtain the plaintext encryption key. Finally, in step **410** data segment is decrypted with the encryption key.

[0049] FIG. 5 depicts the steps required to write data from an access computer or an integrated client to a dispersed data storage system in accordance with a first embodiment of the disclosed invention. In step **502** a write operation is initiated. The initiation of the write operation involves accepting a data string of arbitrary size, and then, if necessary, padding the string to the dispersed data storage system's data segment size. In step **504** a transposition cipher is applied to the data

segment. While the transposition cipher is trivially reversible if a malicious hacker should gain access to a threshold number of slices, the compromise of a single slice will not yield any consecutive information. In step **506** an information dispersal algorithm is applied to the transposed data segment, and the data slices are then written to different storage nodes of the dispersed data storage system in step **508**.

[0050] FIG. 6 depicts the steps required to read data from a dispersed data storage system in accordance with a first embodiment of the disclosed invention. In step **602** a read operation is initiated. In step **604**, a threshold number of data slices are retrieved from the dispersed data storage system, where the threshold for a given dispersed data storage system is the minimum number of slices required to reconstruct a stored data segment. In step **406** a reverse information dispersal algorithm is applied to obtain a transposed data segment, and, in step **408** the transposition cipher is reversed to produce a usable data segment.

[0051] FIG. 7 depicts the steps required to write data from an access computer or an integrated client to a dispersed data storage system in accordance with a second embodiment of the disclosed invention. In step **702** a write operation is initiated. The initiation of the write operation involves accepting a data string of arbitrary size, and then, if necessary, padding the string to the dispersed data storage system's data segment size. In step **704**, data is encrypted using any suitable block cipher, such as those mentioned earlier in this specification. In step **706** a transposition cipher is applied to the encrypted data segment. The use of the transposition cipher will guarantee that no consecutive data will be stored in any slice, and therefore, even if a malicious hacker should compromise the encryption key, she would still have to assemble a number of slices equal to the dispersed data storage system's threshold prior to gaining access to any usable information.

[0052] In step **708** the encrypted and transposed data segment is dispersed using a suitable information dispersal algorithm, such as Cauchy-Reed Solomon. The slices are then stored to different nodes of the dispersed data storage system, such as, for example, slice servers in step **710**.

[0053] FIG. 8 depicts the steps required to read data from a dispersed data storage system in accordance with a second embodiment of the disclosed invention. In step **802** a read operation is initiated. In step **804**, a threshold number of data slices are retrieved from the dispersed data storage system, and in step **806** a reverse information dispersal algorithm is applied to obtain a transposed encrypted data segment. In step **808** the transposition cipher is reversed to produce an encrypted data segment, and in step **810**, decryption is applied to produce a usable data segment.

[0054] The first two embodiments of the disclosed invention involved the use of transposition to improve the security of data stored by a dispersed data storage network. The final two embodiments of the disclosed invention involve the use of all-or-nothing encryption to improve the security of data stored by a dispersed data storage network. Specifically, the use of all-or-nothing encryption, when combined with additional encryption, can provide the advantage of an attacker gaining access to no usable information unless the attacker is able to compromise a threshold m number of data slices. This approaches information theoretic security without the disadvantage of greatly increasing storage size.

[0055] FIG. 9 depicts the steps required to write data to a dispersed data storage system in accordance with a third embodiment of the disclosed invention. In step **904**, an all-

nothing-transformation is applied to a data segment to be stored, thereby producing an all-or-nothing encrypted data segment. The all-or-nothing transformation could be that described earlier in this document, or some other all-or-nothing transformation. In step 906 an information dispersal algorithm is applied to the all-or-nothing encrypted data segment to produce a plurality of data slices, and in step 908, the plurality of data slices is stored to a plurality of storage nodes.

[0056] FIG. 10 depicts the steps required to read data from a dispersed data storage system in accordance with a third embodiment of the disclosed invention. In step 1004, a plurality of data slices corresponding to a stored data segment are retrieved from a plurality of storage nodes, and a reverse information dispersal algorithm is applied in step 1006. In step 1008, the all-or-nothing transformation is removed by using, for example, the method described earlier in this document, or some other method appropriate to the all-or-nothing transformation to be removed.

[0057] FIG. 11 depicts the steps required to write data from a dispersed data storage system in accordance with a fourth embodiment of the disclosed invention. In step 1104, a data segment to be stored is encrypted using a block cipher, such as AES, RC4, or any of the block ciphers discussed earlier in this document, thereby producing an encrypted data segment. In step 1106, an all-nothing-transformation is applied to the encrypted data segment to, thereby producing an all-or-nothing encrypted data segment. The all-or-nothing transformation could be that described earlier in this document, or some other all-or-nothing transformation. In step 1108, an information dispersal algorithm is applied to the all-or-nothing encrypted data segment, producing a plurality of data slices, which are stored to a plurality of storage nodes in step 1110.

[0058] FIG. 12 depicts the steps required to read data from a dispersed data storage system in accordance with a fourth embodiment of the disclosed invention. In step 1204, a plurality of data slices corresponding to a stored data segment are retrieved from a plurality of storage nodes. In step 1206, a reverse information dispersal algorithm is applied, and in step 1208, the all-or-nothing transformation is removed by using, for example, the method described earlier in this document or another appropriate method. Finally, in step 1210, the read data segment is decrypted.

[0059] The foregoing description of the invention has been presented for purposes of illustration and description, and is not intended to be exhaustive or to limit the invention to the precise form disclosed. The description was selected to best explain the principles of the invention and practical application of these principles to enable others skilled in the art to best utilize the invention in various embodiments and various modifications as are suited to the particular use contemplated. It is intended that the scope of the invention not be limited by the specification, but be defined by the claims set forth below.

What is claimed is:

1. A method operating on a computer, the method comprises:

- applying an all-or-nothing transformation to a data segment to produce an all-or-nothing data segment;
- encoding, using an information dispersal algorithm, the all-or-nothing data segment to produce a set of encoded data slices; and
- generating a set of write command to write the set of encoded data slices to storage units of a dispersed storage network.

2. The method of claim 1 wherein the applying an all-or-nothing transformation comprises:

- generating an encryption key;
- encrypting the data segment with said encryption key to produce an encrypted data segment;
- calculating a digest corresponding to the encrypted data segment;
- obfuscating the encryption key by exclusive ORing said encryption key with the digest to produce an obfuscated encryption key; and
- packaging the obfuscated encryption key with the encrypted data segment to produce the all-or-nothing data segment.

3. The method of claim 1 further comprises:

- encrypting the data segment using a first block cipher prior to the applying of the all-or-nothing transformation.

4. The method of claim 1 further comprises:

- retrieving, by a second computer, at least a decode threshold number of encoded data slices of the set of encoded data slices from the storage units;
- decoding, by the second computer using the information dispersal algorithm, the at least the decode threshold number of encoded data slices to recover the all-or-nothing data segment; and
- reversing, by the second computer, the all-or-nothing transformation on the recovered all-or-nothing data segment to recover the data segment.

5. The method of claim 1 further comprises:

- retrieving at least a decode threshold number of encoded data slices of the set of encoded data slices from the storage units;
- decoding, using the information dispersal algorithm, the at least the decode threshold number of encoded data slices to recover the all-or-nothing data segment; and
- reversing the all-or-nothing transformation on the recovered all-or-nothing data segment to recover the data segment.

6. The method of claim 5, wherein the reversing the all-or-nothing transformation comprises:

- separating the all-or-nothing data segment into an encrypted data segment and an obfuscated encryption key;
- calculating a digest based on the encrypted data segment;
- exclusive ORing the obfuscated encryption key with the digest to recover an encryption key; and
- decrypting the encrypted data segment using the encryption key to recover the data segment.

7. A computer comprising:

- a network port adapted to couple with a network and receive a data segment; and
- a processor coupled to the network port wherein the processor:
 - applies an all-or-nothing transformation to the data segment to produce an all-or-nothing data segment;
 - encodes, using an information dispersal algorithm, the all-or-nothing data segment to produce a set of encoded data slices; and
 - generates a set of write commands to write the set of encoded data slices to storage nodes of a dispersed storage network.

8. The computer of claim 7, wherein the processor further functions to apply the all-or-nothing transformation by:

- generating an encryption key;
- encrypting said data segment with said encryption key to produce an encrypted data segment;
- calculating a digest corresponding to said encrypted data segment;
- obfuscating said encryption key by exclusive-ORing said encryption key with said digest to produce an obfuscated encryption key; and
- packaging said obfuscated encryption key with said encrypted data segment to produce the all-or-nothing encrypted data segment.

9. The computer of claim 7, wherein the processor further functions to:

- encrypt the data segment using a first block cipher prior to the applying of the all-or-nothing transformation.

10. The computer of claim 7, wherein the processor further functions to:

retrieve, via the network port, at least a decode threshold number of encoded data slices of the set of encoded data slices from the storage nodes;

decode, using the information dispersal algorithm, the at least the decode threshold number of encoded data slices to recover the all-or-nothing data segment; and
reverse the all-or-nothing transformation on the recovered all-or-nothing data segment to recover the data segment.

11. The computer of claim 10, wherein the processor further functions to reverse the all-or-nothing transformation by: separating the all-or-nothing data segment into an encrypted data segment and an obfuscated encryption key;

calculating a digest based on the encrypted data segment; exclusive ORing the obfuscated encryption key with the digest to recover an encryption key; and
decrypting the encrypted data segment using the encryption key to recover the data segment.

* * * * *