



(22) Date de dépôt/Filing Date: 2007/11/21

(41) Mise à la disp. pub./Open to Public Insp.: 2008/06/05

(62) Demande originale/Original Application: 2 707 531

(30) Priorités/Priorities: 2006/12/01 (GB0624053.5);
2007/05/22 (GB0709759.5)

(51) Cl.Int./Int.Cl. *G06F 21/24* (2006.01)

(71) Demandeur/Applicant:
IRVINE, DAVID, GB

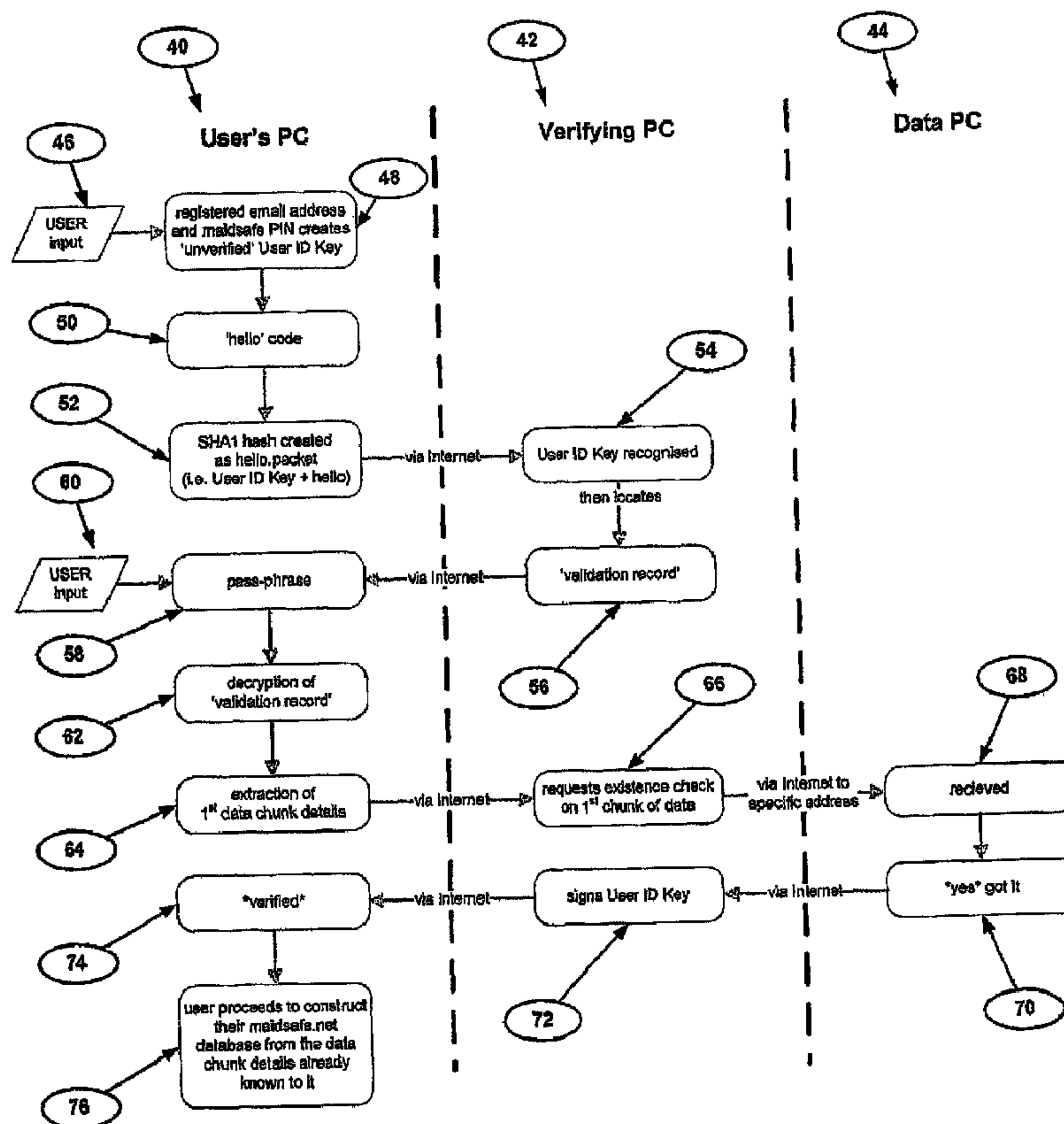
(72) Inventeur/Inventor:
IRVINE, DAVID, GB

(74) Agent: GASTLE AND ASSOCIATES

(54) Titre : AUTOCHIFFREMENT

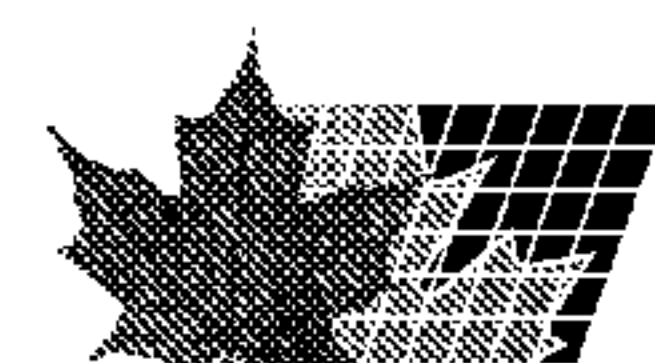
(54) Title: SELF ENCRYPTION

Authentication Flowchart



(57) Abrégé/Abstract:

This invention is a network that is defined by its novel approach to privacy, security and freedom for its users. Privacy by allowing access anonymously, security by encrypting and obfuscating resources and freedom by allowing users to anonymously and



(57) Abrégé(suite)/Abstract(continued):

irrefutably be seen as genuine individuals on the network and to communicate with other users with total security and to securely access resources that are both their own and those that are shared by others with them. Further, this invention comprises a system of self healing data, secure messaging and a voting system to allow users to dictate the direction of development of the network, whereby adoption or denial of proposed add-ons to the network will be decided. System incompatibilities and security breaches on networks and the Internet are addressed by this invention where disparity and tangents of development have had an undue influence. The functional mechanisms that this invention provides will restore open communications and worry-free access in a manner that is very difficult to infect with viruses or cripple through denial of service attacks and spam messaging, plus, it will provide a foundation where vendor lock-in need not be an issue.

ABSTRACT

This invention is a network that is defined by its novel approach to privacy, security and freedom for its users. Privacy by allowing access anonymously, security by encrypting and obfuscating resources and freedom by allowing users to anonymously and irrefutably be seen as genuine individuals on the network and to communicate with other users with total security and to securely access resources that are both their own and those that are shared by others with them. Further, this invention comprises a system of self healing data, secure messaging and a voting system to allow users to dictate the direction of development of the network, whereby adoption or denial of proposed add-ons to the network will be decided. System incompatibilities and security breaches on networks and the Internet are addressed by this invention where disparity and tangents of development have had an undue influence. The functional mechanisms that this invention provides will restore open communications and worry-free access in a manner that is very difficult to infect with viruses or cripple through denial of service attacks and spam messaging, plus, it will provide a foundation where vendor lock-in need not be an issue.

WO 2008/065341

1

PCT/GB2007/004421

TITLE: SELF ENCRYPTION**STATEMENT OF INVENTION:**

1 An issue with today's networks is a combination of vendor lock in,
2 imposed vendor based controls and lack of standards. The present
3 invention allows users to take charge of a new global network in a
4 manner that will maintain effectiveness and promote the setting and
5 attaining of common goals.

6 Another issue with today's networks is the security and privacy of data,
7 this invention allows a secure private and free network where users can
8 enjoy an efficiently managed working environment that presents a
9 guaranteed level of private and securely protected activity.

10 Also today, many computer resources are underutilised to a great
11 degree, including disk space, memory, processing power and any other
12 attached resources, this is inefficient and environmentally detrimental.
13 The present invention seeks to maximise these resources and share
14 them globally to people who purchase them or to people or
15 organisations who are deemed appropriate to benefit from them, such
16 as children in poorer countries, science labs etc. Allocation from these
17 resource pools, together with other resources, will be decided by the
18 users of the system.

BACKGROUND:

19 Digital data is often stored on the hard disks of individual PCs which
20 invariably have memory and operational overhead restrictions. Storage

21 on distributed systems such as the internet is also possible but requires
22 specific storage servers to be available. In addition to these physical
23 systems, data management elements such as security, repair,
24 encryption, authentication, anonymity and mapping etc. are required to
25 ensure successful data transactions and management via the Internet.
26 Systems of messaging and voting exist today but do not allow either
27 authentication on what was voted for or on line anonymity. There have
28 been some attempts as listed below, but none of these systems operate
29 as maidsafe.net does.

30 Listed below is some prior art for these individual elements, of which we
31 have analysed and rejected as true prior art, where necessary we
32 indicate why it is not prior art for our invention:

33 ***PERPETUAL DATA***

34 Most perpetual data generation is allocated with time & calendar etc.
35 (US62669563, JP2001100633). This is not related to this current
36 invention as we have no relation to calendaring, which demonstrates
37 perpetual generation time related data. However, External devices as
38 communication terminal (JP2005057392) (this is a hardware device not
39 related to this present invention) have been used for plurality of packet
40 switching to allow perpetual hand-off of roaming data between networks
41 and battery pack (EP0944232) has been used to around-the-clock
42 accessibility of customer premises equipment interconnected to a
43 broadband network is enhanced by perpetual mode operation of a
44 broadband network interface. In addition, perpetual data storage and
45 retrieval in reliable manner in peer to peer or distributed network The
46 only link here is these devices are connected to Internet connections
47 but otherwise presents no prior art.

48 ***DATABASES & DATA STORAGE METHODS***

49 Patents W09637837, TW223167B, US6760756 and US7099898
50 describe methods of data replication and retention of data during failure.

WO 2008/065341

PCT/GB2007/004421

3

51 Patent WO200505060625 discloses method of secure interconnection
52 when failure occurs.

53 **AUTHENTICATION**

54 Authentication servers are for user and data transaction authentication
55 e.g. JP2005311545 which describe a system wherein the application of
56 'a digital seal' to electronic documents conforms to the Electronic
57 Signature Act. This is similar to the case of signing paper documents
58 but uses the application of an electronic signature through an electronic
59 seal authentication system. The system includes: client computers, to
60 each of which a graphics tablet is connected; an electronic seal
61 authentication server and a PKI authentication server, plus the
62 electronic seal authentication server. US2004254894 discloses an
63 automated system for the confirmed efficient authentication of an
64 anonymous subscriber's profile data in this case.

65 JP2005339247 describes a server based one time ID system and uses
66 a portable terminal. US2006136317 discloses bank drop down boxes
67 and suggests stronger protection by not transmitting any passwords or
68 IDs. Patent US2006126848 discloses a server centric and deals with a
69 one time password or authentication phrase and is not for use on a
70 distributed network. Patent US2002194484 discloses a distributed
71 networks where all chunks are not individually verified and where the
72 manifest is only re-computed after updates to files and hashes are
73 applied and are for validation only.

74 **SELF-AUTHENTICATION**

75 This is mostly used in biometric (WO2006069158). System for
76 generating a patch file from an old version of data which consists of a
77 series of elements and a new version of data which also consists of a
78 series of elements US2006136514). Authentication servers (therefore
79 not a distributed networking principle as per this invention) are
80 commonly used (JP2006107316, US2005273603, EP1548979).

81 However, server and client exchange valid certificates can be used
82 (US2004255037). Instead of server, uses of information exchange
83 system (semantic information) by participant for authentication can be
84 used (JP2004355358), again this semantic information is stored and
85 referenced unlike this present invention.

86 Concepts of identity-based cryptography and threshold secret sharing
87 provides for a distributed key management and authentication. Without
88 any assumption of pre-fixed trust relationship between nodes, the ad
89 hoc network works in a self-organizing way to provide the key
90 generation and key management service, which effectively solves the
91 problem of single point of failure in the traditional public key
92 infrastructure (PKI)-supported system (US2006023887). Authenticating
93 involves encryption keys for validation (WO2005055162) These are
94 validated against known users unlike the present invention. Also, for
95 authentication external housing are used (WO2005034009). All of
96 these systems require a lost or (whether distributed or not) record of
97 authorised users and pass phrases or certificates and therefore do not
98 represent prior art.

99 Ranking, hashing for authentication can be implemented step-by-step
100 and empirical authentication of devices upon digital authentication
101 among a plurality of devices. Each of a plurality of authentication
102 devices can unidirectionally generate a hash value of a low experience
103 rank from a hash value of a high experience rank, and receive a set of
104 high experience rank and hash value in accordance with an experience.
105 In this way, the authentication devices authenticate each other's
106 experience ranks (US2004019788). This is a system of hashing access
107 against known identities and providing a mechanism of effort based
108 access. This present invention does not rely or use such mechanisms.

109 **QUICK ENCIPHERING**

110 This is another method for authentication (JP2001308845). Self-
111 verifying certificate for computer system, uses private and public keys –
112 no chunking but for trusted hardware subsystems (US2002080973) this
113 is a mechanism of self signing certificates for authentication, again
114 useful for effort based computing but not used in this present invention.
115 Other authentication modes are, device for exchanging packets of
116 information (JP2001186186), open key certificate management data
117 (JP10285156), and certification for authentication (WO96139210).
118 Authentication for Peer to Peer system is demonstrated by digital rights
119 management (US2003120928). Digital rights management and CSC
120 (part of that patent s a DRM container) issues which are based on
121 ability to use rather than gaining access to network or resources and
122 therefore not prior art.

123 Known self-healing techniques are divided broadly into two classes.
124 One is a centralized control system that provides overall rerouting
125 control from the central location of a network. In this approach, the
126 rerouting algorithm and the establishing of alarm collection times
127 become increasingly complex as the number of failed channels
128 increases, and a substantial amount of time will be taken to collect
129 alarm signals and to transfer rerouting information should a large
130 number of channels of a multiplexed transmission system fail. The other
131 is a distributed approach in which the rerouting functions are provided
132 by distributed points of the network. The following papers on distributed
133 rerouting-approach have been published: (these are all related to self
134 healing but from a network pathway perspective and therefore are not
135 prior art for this invention which deals with data or data chunks self
136 healing mechanisms.

137 Document 1: W. D. Grover, "The Selfhealing Network", Proceedings of
138 Grobecom '87, November 1987.

139 Document 2: H. C. Yang and S. Hasegawa, "Fitness: Failure
140 Immunization Technology For Network Service Survivability",
141 Proceedings of Globecom '88, December 1988.

142 Document 3: H. R. Amirazizi, "Controlling Synchronous Networks With
143 Digital Cross-Connect Systems", Proceedings of Globecom '88,
144 December 1988.

145 Document 1 is concerned with a restoration technique for failures in a
146 single transmission system, and Document 2 relates to a "multiple-
147 wave" approach in which route-finding packets are broadcast in multiple
148 wave fashion in search of a maximum bandwidth until alternate routes
149 having the necessary bandwidth are established. One shortcoming of
150 this multiple wave approach is that it takes a long recovery time.
151 Document 3 also relates to fault recovery for single transmission
152 systems and has a disadvantage in that route-finding packets tend to
153 form a loop and hence a delay is likely to be encountered.

154 **SELF-HEALING**

155 This is demonstrated by a system and method of secure and
156 tamperproof remote files over distributed system, redirects integrity
157 check fail data to install module for repairing (WO20566133) This
158 discloser relies on testing data from a central location and not
159 distributed chunking as with the present invention. It also does not allow
160 for multiple access and sharing of the testing and ownership of chunks.
161 Server are used for self-healing (US2004177156), effectively removing
162 these from a prior art claim. Self-repairing is conducted by data overlay
163 is built as a data structure on top of a logical space defined by a
164 distributed hash table (DHT) in a peer-to-peer (P2P) network
165 environment (US2005187946) This Microsoft patent is a patent to DT
166 networks which is peculiar as these exist in some quantity and have
167 done for many years, however there is no claim made to self repair data
168 as is in this present invention but to self repair data storage locations

169 (i.e. in p2p terms find nearest node). This is not self healing data but
170 merely a description of a typical DHT and the availability of routes to
171 data and providing multiple routes. This is not prior art for this present
172 inventions but very likely not enforceable as there are many cases of
173 prior art against this Microsoft patent.

174 Identical communicating node elements are used for power delivery
175 network for self-repairing (US2005043858). Self-healing also relates to
176 distributed data systems and, in particular, to providing high availability
177 during performance of a cluster topology self-healing process within a
178 distributed data system cluster. A cluster topology self-healing process
179 may be performed in response to a node failure in order to replicate a
180 data set stored on a failed node from a first node storing another copy
181 of the data set to a second non-failed node (US2004066741). An
182 apparatus and method for self-healing of software may rely on a
183 distribution object in a directory services of a network to provide data for
184 controlling distribution of software and installation of files associated
185 therewith (US6023586). A technique for the substantially instantaneous
186 self-healing of digital communications networks. Digital data streams
187 from each of N nearby sources are combined and encoded to produce
188 N+M coded data streams using a coding algorithm. The N+M coded
189 data streams are then each transmitted over a separate long haul
190 communications link to a decoder where any N of the N+M coded data
191 streams can be decoded uniquely to produce the original N data streams
192 (EP0420648). To provide a self-healing communications network which
193 can be recovered from a failure in a short period of time even if the
194 failure has occurred in a multiplexed transmission line (US5235599)
195 The above patents and inventions are based on clustering technology
196 and not distributed computing or Internet based computing. The cluster
197 is simply many machines connected to create a larger machine. It is
198 treated as a single machine with known user access etc. and not prior
199 art to this present invention. The N + M coding schemes discussed are
200 patents based on digital communications and reception links and are

201 not related to this present invention although at first glance they appear
202 to have the same language in areas.

203 Attempts to moving towards attaining some limited aspects of self-
204 encryption are demonstrated by

205 (a) US2003053053625 discloser shows limitation of asymmetrical and
206 symmetrical encryption algorithms, and particularly not requiring
207 generating a key stream from symmetric keys, nor requiring any time
208 synchronising, with minimal computational complexity and capable of
209 operated at high speed. A serial data stream to be securely transmitted
210 is first demultiplexed into a plurality N of encryptor input data stream.
211 The input data slices are created which have cascade of stages, include
212 mapping & delay function to generate output slices. These are
213 transmitted though a transmission channel. Decryptor applies inverse
214 step of cascade of stages, equalizing delay function and mapping to
215 generate output data slices. The output data streams are multiplexed.
216 The encryptor and decryptor require no synchronizing or timing and
217 operate in simple stream fashion. N:N mapping does not require
218 expensive arithmetic and implemented in table lookup. This provides
219 robust security and efficiency. A significant difference between this
220 approach and prior cipher method is that the session key is used to
221 derive processing parameters (tables and delays) of the encryptor and
222 decryptor in advance of data transmission. Instead of being used to
223 generate a key stream at real-time rates. Algorithm for generating
224 parameters from a session key is disclosed This patent is based on
225 data communications and encrypting data in transit automatically and
226 decrypting automatically at the remote end, this is not related to this
227 present invention.

228 (b) US2002184485 discloser addresses secure communication, by
229 encryption of message (SSDO-self signing document objects), such
230 that only known recipient in possession of a secret key can read the

231 message and verification of message, such that text and origin of
232 message can be verified. Both capabilities and built into message that
233 can be transmitted over internet and decrypted or verified by computer
234 implementing a document representation language that supports
235 dynamic content e.g. any standard web browser, such that elaborate
236 procedures to ensure transmitting and receiving computers have same
237 software are no longer necessary. Encrypted message or one encoded
238 for verification can carry within itself all information needed to specify
239 the algorithm needed for decryption. This is a patent describing a key
240 pair encryption and validation of same software. This is not used by the
241 present invention where key pairs are used for asymmetric encryption
242 of some data but this is used with the RSA (now out of patent)
243 encryption ciphers and not in the manner described above which is
244 more for validation.

245 A range of limited methods for self-encryption have been developed
246 e.g. system for radomisation-encryption of digital data sequence with
247 freely selectable (EP1182777) (this is a key generating patent and not
248 self encryption as this current invention shows), use of code key
249 calculation encryption mode but using server (CN1658553), uses self-
250 test mode (US6028527), encryption system for randomising data signal
251 for transmission (not storing) and reproducing information at a receiver
252 (US4760598), uses private encryption keys into components and
253 sending them to trusted agents (rather than self encryption as per this
254 present invention) (JP2005328574), cryptographic system with key
255 escrow feature, rather than self encryption as described in this present
256 invention (US6009177), steps of first encoding one set of message
257 signal with first keyed transformation (US6385316), self-modifying fail-
258 safe password system (US6370649), time-based encrypting method
259 involves splitting voice signal into time intervals, random permutations
260 etc. (RU2120700), uses hardware decryption module (HDM)
261 (US2003046568), realizing data security storage and algorithm storage
262 by means of semiconductor memory device (US2006149972), use

263 certificate from certificate server (US20020428080), use certificates for
264 encryption of communications (EP1422865), use self-service terminal
265 for encryption and transmission of data (US2006020788), method for
266 implementing security communication by encryption algorithm
267 (US2005047597), method of data encryption – block encryption variable
268 length (BEVL) encoding, overcomes weakness of CMEA algorithm)
269 (US2004190712), encrypted cipher code for secure data transmission
270 (CN1627681) method and system for encrypting streamed data
271 employing fast set-up single use key and self-synchronising
272 (US2005232424) and for security, generate MAC for data integrity,
273 placing electronic signature, use TREM software module
274 (US2004199768)

275 None of the above systems utilise self encryption as per the present
276 invention and are related to voice and data transmissions, or include
277 hardware controllers or servers.

278 ***PRIVATE SHARED FILES***

279 US6859812 discloses a system and method for differentiating private
280 and shared files, where clustered computers share a common storage
281 resource, Network-Attached Storage (NAS) and Storage Area Network
282 (SAN), therefore not distributed as in this present invention. US5313646
283 has a system which provides a copy-on-write feature which protects the
284 integrity of the shared files by automatically copying a shared file into
285 user's private layer when the user attempts to modify a shared file in a
286 back layer, this is a different technology again and relies on user
287 knowledge – not anonymous. WO02095545 discloses a system using a
288 server for private file sharing which is not anonymous.

289 ***DISTRIBUTED NETWORK SHARED MAPS***

290 A computer system having plural nodes interconnected by a common
291 broadcast bus is disclosed by US5117350. US5423034 shows how
292 each file and level in the directory structure has network access

293 privileges. The file directory structure generator and retrieval tool have a
294 document locator module that maps the directory structure of the files
295 stored in the memory to a real world hierarchical file structure of files.
296 Therefore not distributed across public networks or anonymous or self
297 encrypting, the present inventions does not use broadcasting in this
298 manner.

299 **SECURITY**

300 Today systems secure transactions through encryption technologies
301 such as Secure Sockets Layer (SSL), Digital Certificates, and Public
302 Key Encryption technologies. The systems today address the hackers
303 through technologies such as Firewalls and Intrusion Detection
304 systems. The merchant certification programs are designed to ensure
305 the merchant has adequate inbuilt security to reasonably assure the
306 consumer their transaction will be secure. These systems also ensure
307 that the vendor will not incur a charge back by attempting to verify the
308 consumer through secondary validation systems such as password
309 protection and eventually, Smart Card technology.

310 Network firewalls are typically based on packet filtering which is limited
311 in principle, since the rules that judge which packets to accept or reject
312 are based on subjective decisions. Even VPNs (Virtual Private
313 Networks) and other forms of data encryption, including digital
314 signatures, are not really safe because the information can be stolen
315 before the encryption process, as default programs are allowed to do
316 whatever they like to other programs or to their data files or to critical
317 files of the operating system. This is done by (CA247150) automatically
318 creating an unlimited number of Virtual Environments (VEs) with virtual
319 sharing of resources, so that the programs in each VE think that they
320 are alone on the computer. The present invention takes a totally
321 different approach to security and obviates the requirement of much of
322 the above particularly CA2471505.

323 US6185316 discloses security via fingerprint imaging testing bit of code
324 using close false images to deter fraudulent copying, this is different
325 from the present invention in that we store no images at all and certainly
326 not in a database.

327 **SECURITY & STORAGE SYSTEMS**

328 There are currently several types of centralised file storage systems
329 that are used in business environments. One such system is a server-
330 tethered storage system that communicates with the end users over a
331 local area network, or LAN. The end users send requests for the
332 storage and retrieval of files over the LAN to a file server, which
333 responds by controlling the storage and/or retrieval operations to
334 provide or store the requested files. While such a system works well for
335 smaller networks, there is a potential bottleneck at the interface
336 between the LAN and the file storage system.

337 Another type of centralised storage system is a storage area network,
338 which is a shared, dedicated high-speed network for connecting storage
339 resources to the servers. While the storage area networks are generally
340 more flexible and scalable in terms of providing end user connectivity to
341 different server-storage environments, the systems are also more
342 complex. The systems require hardware, such as gateways, routers,
343 switches, and are thus costly in terms of hardware and associated
344 software acquisition.

345 Yet another type of storage system is a network attached storage
346 system in which one or more special-purpose servers handle file
347 storage over the LAN.

348 Another file storage system utilizes distributed storage resources
349 resident on various nodes, or computers, operating on the system,
350 rather than a dedicated centralised storage system. These are
351 distributed systems, with the clients communicating peer-to-peer to

352 determine which storage resources to allocate to particular files,
353 directories and so forth. These systems are organized as global file
354 stores that are physically distributed over the computers on the system.
355 A global file store is a monolithic file system that is indexed over the
356 system as, for example, a hierarchical directory. The nodes in the
357 systems use Byzantine agreements to manage file replications, which
358 are used to promote file availability and/or reliability. The Byzantine
359 agreements require rather lengthy exchanges of messages and thus
360 are inefficient and even impractical for use in a system in which many
361 modifications to files are anticipated. US200211434 shows a peer-to-
362 peer storage system which describes a storage coordinator that
363 centrally manages distributed storage resources. The difference here is
364 the requirement of a storage broker, making this not fully distributed.
365 The present invention also differs in that the present invention has no
366 central resources for any of the system and we also encrypt data for
367 security as well as the self healing aspect of our system which is again
368 distributed.

369 US7010532 discloses improved access to information stored on a
370 storage device. A plurality of first nodes and a second node are coupled
371 to one another over a communications pathway, the second node being
372 coupled to the storage device for determining meta data including block
373 address maps to file data in the storage device.

374 ~~JP2003273860 discloses a method of enhancing the security level~~
375 ~~during access of an encrypted document including encrypted content. A~~
376 ~~document access key for decrypting an encrypted content within an~~
377 ~~encrypted document is stored in a management device, and a user~~
378 ~~device wishing to access the encrypted document transmits its user ID~~
379 ~~and a document identification key for the encrypted document, which~~
380 ~~are encrypted by a private key, together with a public key to the~~
381 ~~management device to request transmission of the document access~~
382 ~~key. Differing from this invention in that it never transmit user id or login~~

383 in the network at all. Also it does not require management devices of
384 any form.

385 JP2002185444 discloses improves security in networks and the
386 certainty for satisfying processing requests. In the case of user
387 registration, a print server forms a secret key and a public key, and
388 delivers the public key to a user terminal, which forms a user ID, a
389 secret key and a public key, encrypts the user ID and the public key by
390 using the public key, and delivers them to the print server. This is not
391 linked at all to this invention and is a system for a PKI infrastructure for
392 certificate access to network nodes.

393 The private and public keys of users are used in US6925182, and are
394 encrypted with a symmetric algorithm by using individual user
395 identifying keys and are stored on a network server making it a different
396 proposition from a distributed network

397 US2005091234 describes data chunking system which divides data into
398 predominantly fixed-sized chunks such that duplicate data may be
399 identified. This is associated with storing and transmitting data for
400 distributed network. US2006206547 discloses a centralised storage
401 system, whilst US2005004947 discloses a new PC based file system.
402 US2005256881 discloses data storage in a place defined by a path
403 algorithm. This is a server based duplicate removal and not necessarily
404 encrypting data, unlike the present invention which does both and
405 requires no servers.

406 **SECURITY & ENCRYPTION**

407 Common email communications of sensitive information is in plain text
408 and is subject to being read by unauthorized code on the senders
409 system, during transit and by unauthorized code on the receiver's
410 system. Where there is a high degree of confidentiality required, a
411 combination of hardware and software secures data.

412 A high degree of security to a computer or several computers
413 connected to the Internet or a LAN as disclosed in US2002099666.
414 Hardware system is used which consists of a processor module, a
415 redundant non-volatile memory system, such as dual disk drives, and
416 multiple communications interfaces. This type of security system must
417 be unlocked by a pass phrase to access data, and all data is
418 transparently encrypted, stored, archived and available for encrypted
419 backup. A system for maintaining secure communications, file transfer
420 and document signing with PKI, and a system for intrusion monitoring
421 and system integrity checks are provided, logged and selectively
422 alarmed in a tamper-proof, time-certain manner.

423 **ENCRYPTION**

424 WO2005093582 discloses method of encryption where data is secured
425 in the receiving node via private tag for anonymous network browsing.
426 However, other numerous encryption methods are also available such
427 as (i) implantation of Reed Solomon algorithm (WO02052787), which
428 ensures data is coded in parabolic fashion for self-repairing and
429 storage, (ii) storage involves incremental backup (WO02052787), (ii)
430 uses stenographic (US2006177094), (iv) use cipher keys (CN1620005),
431 encryption for non text (US2006107048) and US2005108240 discloses
432 user keys and randomly generated leaf node keys. The present
433 invention uses none of these methods of encryption and in particular
434 ensures all chunks are unique and do not point to another for security
435 (an issue with Reed Solomon and N + K implementations of parabolic
436 coding)

437 **ENCRYPTED DOCUMENT SIGNING**

438 WO2005060152 discloses a digital watermark representing the one-
439 way hash is embedded in a signature document is used for electronic
440 signing. Mostly encrypted document signing is associated with legal
441 documents, e.g. on-line notary etc. e.g. US2006161781, signature

442 verification (US6381344). WO0182036 discloses a system and method
443 for signing, storing, and authenticating electronic documents using
444 public key cryptography. The system comprises a document service
445 computer cluster connected to user computers, document owner server
446 computers, and registration computers via a network such as for
447 example, the internet or the world wide web. WO0013368 discloses
448 both the data object and the signature data are encrypted. None of
449 these systems are designed or allow for distributed signing networks
450 unlike the present invention.

451 US6912660 discloses a method for parallel approval of an electronic
452 document. A document authentication code (DAC 0) is generated,
453 linked to the original document. Subsequent approvals of the document
454 generate a DAC x related to that specific approval. This is not linked to
455 the present invention as it's a document approval system – i.e. one
456 which allows a document to have multiple signatories to authenticate
457 approval, the present invention does not do this at all.

458 US6098056 discloses a system and method for controlling access
459 rights to and security of digital content in a distributed information
460 system, e.g., Internet. The network includes at least one server coupled
461 to a storage device for storing the limited access digital content
462 encrypted using a random-generated key, known as a Document
463 Encryption Key (DEK). The DEK is further encrypted with the server's
464 public key, using a public/private key pair algorithm and placed in a
465 digital container stored in a storage device and including as a part of the
466 meta-information which is in the container. The client's workstation is
467 coupled to the server (one of the many difference's from the present
468 invention) for acquiring the limited access digital content under the
469 authorized condition. A Trusted Information Handler (TIH) is validated
470 by the server after the handler provides a data signature and type of
471 signing algorithm to transaction data descriptive of the purchase
472 agreement between the client and the owner. After the handler has

473 authenticated, the server decrypts the encrypted DEK with its private
474 key and re-encrypts the DEK with the handler's public key ensuring that
475 only the information handler can process the information. The encrypted
476 DEK is further encrypted with the client's public key personalizing the
477 digital content to the client. The client's program decrypts the DEK with
478 his private key and passes it along with the encrypted content to the
479 handler which decrypts the DEK with his private key and proceeds to
480 decrypt the content for displaying to the client.

481 US5436972 discloses a method for preventing inadvertent betrayal by a
482 trustee of escrowed digital secrets. After unique identification data
483 describing a user has been entered into a computer system, the user is
484 asked to select a password to protect the system. US5557518 discloses
485 a system to open electronic commerce using trusted agents.
486 US5557765 discloses a system and method for data recovery. An
487 encrypting user encrypts a method using a secret storage key (KS) and
488 attaches a Data Recovery Field (DRF), including an Access Rule Index
489 (ARI) and the KS to the encrypted message.

490 US5590199, discloses a system for authenticating and authorizing a
491 user to access services on a heterogeneous computer network. The
492 system includes at least one workstation and one authorization server
493 connected to each other through a network.

494 US2006123227 and WO0221409 describe trust based effort measuring
495 techniques to validate signatures without the requirement for a central
496 body or central messaging entity. This is an interesting new concept but
497 not used in the current invention.

498 **SELF-ENCRYPTION**

499 Attempts to moving towards attaining some limited aspects of self-
500 encryption are demonstrated by:

501 (a) US2003053053625 discloses limitation of asymmetrical and
502 symmetrical encryption algorithms, and particularly not requiring
503 generation of a key stream from symmetric keys, nor requiring any time
504 synchronizing, with minimal computational complexity and capable of
505 operating at high speed. A serial data stream to be securely transmitted
506 is first demultiplexed into a plurality N of encryptor input data stream.
507 The input data slices are created which have a cascade of stages,
508 include mapping & delay functions to generate output slices. These are
509 transmitted through a transmission channel. Decryptor applies inverse
510 step of cascade of stages, equalizing delay function and mapping to
511 generate output data slices. The output data streams are multiplexed.
512 The encryptor and decryptor require no synchronizing or timing and
513 operate in simple stream fashion. N:N mapping does not require
514 expensive arithmetic and implemented in table lookup. This provides
515 robust security and efficiency. A significant difference between this
516 approach and prior cipher method is that the session key is used to
517 derive processing parameters (tables and delays) of the encryptor and
518 decryptor in advance of data transmission. Instead of being used to
519 generate a key stream at real-time rates. Algorithm for generating
520 parameters from a session key is disclosed. This is a data
521 communications network and not related to current invention.

522 (b) US2002184485 addresses secure communication, by encryption of
523 message (SSDO-self signing document objects), such that only known
524 recipient in possession of a secret key can read the message and
525 verification of message, such that text and origin of message can be
526 verified. Both capabilities are built into message that can be transmitted
527 over internet and decrypted or verified by computer implementing a
528 document representation language that supports dynamic content e.g.
529 any standard web browser, such that elaborate procedures to ensure
530 transmitting and receiving computers have same software are no longer
531 necessary. Encrypted message or one encoded for verification can

532 carry within itself all information needed to specify the algorithm needed
533 for decryption.

534 ***ANONYMOUS TRANSACTIONS & INTERFACES***

535 US2004117303 discloses an anonymous payment system and is
536 designed to enable users of the Internet and other networks to
537 exchange cash for electronic currency that may be used to conduct
538 commercial transactions world-wide through public networks.
539 US2005289086 discloses an anonymity for web registration which
540 allows payment system. US2002073318 describe use of servers where
541 the system is effort based trust on combination of anonymous keys to
542 transact and public key to buy non anonymous credits. Each of these is
543 a centrally controlled system and do not provide a mechanism to
544 transfer credits or cash to anonymous accounts. Many of these actually
545 require user registration on a web site.

546 US2003163413 discloses a method of conducting anonymous
547 transactions over the Internet to protect consumers from identity fraud.
548 The process involves the formation of a Secure Anonymous
549 Transaction Engine to enable any consumer operating over an open
550 network, such as the Internet to browse, collect information, research,
551 shop, and purchase anonymously. The Secure Anonymous Transaction
552 Engine components provide a highly secure connection between the
553 consumer and the provider of goods or services over the Internet by
554 emulating an in store anonymous cash transaction although conducted
555 over the Internet. This again is server based and requires user
556 registration.

557 With regard to cash transfers, a truly anonymous purchase is one in
558 which the purchaser and seller are unknown to each other, the
559 purchase process is not witnessed by any other person, and the
560 exchange medium is cash. Such transactions are not the norm. Even
561 cash transactions in a place of business are typically witnessed by

WO 2008/065341

PCT/GB2007/004421

20

562 salespersons and other customers or bystanders, if not recorded on
563 videotape as a routine security measure. On the other hand, common
564 transaction media such as payment by personal check or credit card
565 represent a clear loss of anonymity, since the purchaser's identity as
566 well as other personal information is attached to the transaction (e. g.,
567 driver's license number, address, telephone number, and any
568 information attached to the name, credit card, or driver's license
569 number). Thus, although a cash transaction is not a truly anonymous
570 purchase, it provides a considerably higher degree of purchase
571 anonymity than a transaction involving a personal check or credit card,
572 and affords perhaps the highest degree of purchase anonymity
573 achievable in the present. The use of cash, however, has limitations,
574 especially in the context of electronic commerce.

575 WO0203293 discloses methods, systems, and devices for performing
576 transactions via a communications network such as the Internet while
577 preserving the anonymity of at least one of the parties. A transaction
578 device is linked to an anonymous account to allow a party to preserve
579 an equivalent level of anonymity as the use of cash when making a
580 transaction at a traditional brick-and-mortar business as well as in the
581 virtual world of electronic commerce. As such, the transaction device
582 may be considered equivalent to a flexible and versatile cash wallet. In
583 this way, combines the desirable features of cash (anonymity, security,
584 and acceptance) and of electronic commerce (speed, ease, and
585 convenience). This like the next invention requires a hardware based
586 device unlike the present invention.

587 EP0924667 is based on a distributed payment system for cash-free
588 payment with purse chip cards using the Net. The system consists of a
589 client system which is, for example, installed at the customer site and a
590 server system which is, for example, installed at the dealer.

591 US6299062 discloses an electronic cash system for performing an
592 electronic transaction using an electronic cash, comprises at least one
593 user apparatus each capable of using the electronic cash; an
594 authentication centre apparatus, for receiving a user identity
595 information, a corresponding public key along with a certificate issue
596 request from one of the user apparatus and for issuing a certificate for
597 the user apparatus's public key after confirming the identity of the
598 corresponding user. This again requires hardware and user registration
599 to the system

600 US2004172539 discloses method for generating an electronic receipt in
601 a communication system providing a public key infrastructure,
602 comprising the steps of receiving by a second party a request message
603 from a first party, the request message comprising a transaction request
604 and a first public key based on a secret owned by the first party and
605 wherein the secret is associated with at least the secret of a further
606 public key of the first party. (server based)

607 WO0219075 discloses publicly-accessible, independent, and secure
608 host internet site that provides a downloadable agent program to any
609 anonymous client PC, with the agent program generating within the
610 client PC a registration checksum based upon the document to be
611 registered.

612 ***ANONYMOUS VOTING***

613 US2003159032 discloses automatically generating unique, one-way
614 compact and mnemonic voter credentials that support privacy and
615 security services. Discloses any voting system, voting organization, or
616 voting game wherein participants need to be anonymous and/or must
617 exchange secrets and/or make collective decisions. US2002077887
618 (requires registration and initial knowledge of the person who receives
619 the ballot, and requires a server) discloses an architecture that enables
620 anonymous electronic voting over the Internet using public key

621 technologies. Using a separate public key/private key pair, the voting
622 mediator validates the voting ballot request. (Hardware device)
623 DE10325491 discloses that the voting method has an electronic ballot
624 box for collecting encoded electronic voting slips and an electronic box
625 for collecting the decoded voting slips. The voter fills out his voting slip
626 at a computer and authenticates his vote with an anonymous signature
627 setting unit.

628 US2004024635 (hardware based, requiring servers) discloses a
629 distributed network voting system; a server for processing votes cast
630 over a distributed computing network. The server includes memory
631 storage, data identification, an interested party and a processor in
632 communication with the memory. The processor operates to present an
633 issue to a user of a client computer, receive a vote on the issue from
634 the user, and transmit data relating to the vote to the interested party
635 based upon the data identifying the interested party stored in the
636 memory. The processor further operates to generate a vote status
637 cookie when the user submits the vote, transmit the vote status cookie
638 to the client for storage, and transmit data to the user that prompts the
639 user to provide authentication data relating to the user, who then
640 receives authentication data relating to the user and authenticate the
641 user based on the authentication data.

642 WO03098172 discloses modular monitoring and protection system with
643 distributed voting logic.

644 **MAPPING**

645 US2006112243 discloses a hard disk mapping where the data is copied
646 locally and then the machine decides it can use either copy and
647 whether or not update the other one. EP1049291 discloses a remote
648 device monitoring using pre-calculated maps of equipment locations.
649 These are hardware based data mapping systems and not related.

WO 2008/065341

PCT/GB2007/004421

23

650 As above prior art highlights separate existence of elements such as
651 storage, security, repairing, encryption, authentication, anonymity,
652 voting and mapping etc. for data transaction and storage via internet.
653 There is some limited linkage between a few of the individual elements,
654 but none are inter-linked to provide comprehensive solution for secure
655 data storage and transmittance via internet utilisation. The inventions
656 below list solutions to address the vacuum and provide an inexpensive
657 solution for secure internet data storage and transmittance with other
658 added benefits.

Summary of Invention

659 The main embodiments of this invention are as follows:

660 A system of sharing access to private files which has the functional
661 elements of:

- 662 1. Perpetual Data
- 663 2. Self encryption
- 664 3. Data Maps
- 665 4. Anonymous Authentication
- 666 5. Shared access to Private files
- 667 6. ms Messenger
- 668 7. Cyber Cash
- 669 8. Worldwide Voting System

670 ... with the additionally linked functional elements of:

- 671 1. Peer Ranking
- 672 2. Self Healing
- 673 3. Security Availability
- 674 4. Storage and Retrieval
- 675 5. Duplicate Removal
- 676 6. Storing Files
- 677 ~~7. Chunking~~
- 678 8. Encryption / Decryption
- 679 9. Identify Chunks
- 680 10. Revision Control
- 681 11. Identify Data with Very Small File
- 682 12. Logon
- 683 13. Provide Key Pairs
- 684 14. Validation
- 685 15. Create Map of Maps

WO 2008/065341

PCT/GB2007/004421

25

- 686 16. Share Map
- 687 17. Provide Public ID
- 688 18. Encrypted Communications
- 689 19. Document Signing
- 690 20. Contract Conversations
- 691 21. Counterfeit Protection
- 692 22. Allow Selling of Machine Resources
- 693 23. Interface with Non-Anonymous Systems
- 694 24. Anonymous Transactions
- 695 25. Anonymity
- 696 26. Proven Individual
- 697 27. Validation of Vote Being Used
- 698 28. Distributed Controlled Voting

699 A distributed network system and product which provides:

- 700 a. secure communications
- 701 b. store data & share resources
- 702 c. anonymous backing and restoring data
- 703 d. share private files & secure data without using server
- 704 e. anonymous authentication of users
- 705 f. approve transaction based on digital currency
- 706 g. CPU sharing via anonymous voting system

707 ~~A method of allowing users to securely store data and share resources~~
708 ~~across a distributed network by utilising anonymously shared computer~~
709 ~~resources.~~

710 A method to allow secure communications between users by utilising
711 public ID's linked to anonymous ID's to authenticate users as well as
712 allowing contract signed conversations.

WO 2008/065341

PCT/GB2007/004421

26

- 713 A method to allow sharing and allocation of resources globally by
714 utilising effort based testing and anonymously authenticated users in a
715 global distributed network.
- 716 A method specifically to backup and restore data anonymously in a
717 distributed network with guarantees on integrity and recovery times..
- 718 A method to share private and secured data without the use of file
719 servers or any controlling body or centralised resource.
- 720 A method to approve the exchange of resources and other transactions
721 based on a digital currency which utilises links with non anonymous
722 payment systems.
- 723 A method to allow data to be described decoded and identified using
724 very small data map files.
- 725 A method to allow anonymous authentication of users on a network.
- 726 A method of above to allow sharing of CPU power globally and to
727 contribute to systems based on users input from a worldwide secure
728 and anonymous voting system.
- 729 A method where a person's computer operating system and related
730 computer program may be held on a removable disk (such as a USB
731 stick optionally with biometric recognition to evade keyloggers) and
732 used to boot any compatible computer with a known virus / trojan horse
733 free system to access their data remotely and securely without worrying
734 about the integrity of host machine they are using.
- 735 At least one computer program comprising instructions for causing at
736 least one computer to perform the method, system and product
737 according to any of above.

WO 2008/065341

PCT/GB2007/004421

27

738

That at least one computer program of above embodied on a recording

739

medium or read-only memory, store.

Detailed Description:

740 (References to IDs used in descriptions of the system's functionality)

741 **MID** – this is the base ID and is mainly used to store and forget files.
742 Each of these operations will require a signed request. Restoring may
743 simply require a request with an ID attached.

744 **PMID** – This is the proxy mid which is used to manage the receiving of
745 instructions to the node from any network node such as get/ put / forget
746 etc. This is a key pair which is stored on the node – if stolen the key pair
747 can be regenerated simply disabling the thief's stolen PMID – although
748 there's not much can be done with a PMID key pair.

749 **CID** – Chunk Identifier, this is simply the chunkid.KID message on the
750 net.

751 **TMID** – This is today's ID a one time ID as opposed to a one time
752 password. This is to further disguise users and also ensure that their MID
753 stays as secret as possible.

754 **MPID** – The maidsafe.net public ID. This is the ID to which users can add
755 ~~their own name and actual data if required. This is the ID for messenger,~~
756 sharing, non anonymous voting and any other method that requires we
757 know the user.

758 **MAID** – this is basically the hash of and actual public key of the MID. this
759 ID is used to identify the user actions such as put / forget / get on the
760 maidsafe.net network. This allows a distributed PKI infrastructure to exist
761 and be automatically checked.

762 **KID** – Kademlia ID this can be randomly generated or derived from
763 known and preferably anonymous information such as an anonymous
764 public key hash as with the MAID.. In this case we use kademlia as the
765 example overlay network although this can be almost any network
766 environment at all.

767 **MSID** – maidsafe.net Share ID, an ID and key pair specifically created for
768 each share to allow users to interact with shares using a unique key not
769 related to their MID which should always be anonymous and separate.

Anonymous Authentication Description

770 Anonymous authentication relates to system authentication and, in
771 particular, authentication of users for accessing resources stored on a
772 distributed or peer-to-peer file system. Its aim is to preserve the
773 anonymity of the users and to provide secure and private storage of data
774 and shared resources for users on a distributed system. It is a method of
775 authenticating access to a distributed system comprising the steps of;

- 776 • Receiving a user identifier;
- 777 • Retrieving an encrypted validation record identified by the user
778 identifier;
- 779 • Decrypting the encrypted validation record so as to provide
780 decrypted information; and ...
- 781 • Authenticating access to data in the distributed system using the
782 decrypted information.

783 Receiving, retrieving and authenticating may be performed on a node in
784 the distributed system preferably separate from a node performing the
785 step of decrypting. The method further comprises the step of generating
786 the user identifier using a hash. Therefore, the user identifier may be
787 considered unique (and altered if a collision occurs) and suitable for

788 identifying unique validation records. The step of authenticating access
789 may preferably further comprise the step of digitally signing the user
790 identifier. This provides authentication that can be validated against
791 trusted authorities. The method further comprises the step of using the
792 signed user identifier as a session passport to authenticate a plurality of
793 accesses to the distributed system. This allows persistence of the
794 authentication for an extended session.

795 The step of decrypting preferably comprises decrypting an address in the
796 distributed system of a first chunk of data and the step of authenticating
797 access further comprises the step of determining the existence of the first
798 chunk at the address, or providing the location and names of specific
799 data elements in the network in the form of a data map as previously
800 describe. This efficiently combines the tasks of authentication and
801 starting to retrieve the data from the system. The method preferably
802 further comprises the step of using the content of the first chunk to obtain
803 further chunks from the distributed system. Additionally the decrypted
804 data from the additional chunks may contain a key pair allowing the user
805 at that stage to sign a packet sent to the network to validate them or
806 additionally may preferable self sign their own id.

807 Therefore, there is no need to have a potentially vulnerable record of the
808 file structure persisting in one place on the distributed system, as the
809 user's node constructs its database of file locations after logging onto the
810 system.

811 There is provided a distributed system comprising;

- 812 • a storage module adapted to store an encrypted validation record;
- 813 • a client node comprising a decryption module adapted to decrypt an
814 encrypted validation record so as to provide decrypted information;
815 and
- 816 • a verifying node comprising:

- 817 • a receiving module adapted to receive a user identifier;
818 • a retrieving module adapted to retrieve from the storage module an
819 encrypted validation record identified by the user identifier;
820 • a transmitting module adapted to transmit the encrypted validation
821 record to the client node; and
822 • an authentication module adapted to authenticate access to data in
823 the distributed file system using the decrypted information from the
824 client node.

825 The client node is further adapted to generate the user identifier using a
826 hash. The authentication module is further adapted to authenticate
827 access by digitally sign the user identifier. The signed user identifier is
828 used as a session passport to authenticate a plurality of accesses by the
829 client node to the distributed system. The decryption module is further
830 adapted to decrypt an address in the distributed system of a first chunk of
831 data from the validation record and the authentication module is further
832 adapted to authenticate access by determining the existence of the first
833 chunk at the address. The client node is further adapted to use the
834 content of the first chunk to obtain further authentication chunks from the
835 distributed system.

836 There is provided at least one computer program comprising program
837 instructions for causing at least one computer to perform. One computer
838 program is embodied on a recording medium or read-only memory,
839 stored in at least one computer memory, or carried on an electrical
840 carrier signal.

841 Additionally there is a check on the system to ensure the user is login
842 into a valid node (software package). This will preferably include the
843 ability of the system to check validity of the running maidsafe.net
844 software by running content hashing or preferably certificate checking of
845 the node and also the code itself.

Linked elements for maidsafe.net (Figure 1)

846 The maidsafe.net product invention consists of 8 individual inventions,
847 which collectively have 28 inter-linked functional elements, these are:

848 The individual inventions are:
849 PT1 – Perpetual Data
850 PT2 – Self encryption
851 PT3 – Data Maps
852 PT4 – Anonymous Authentication
853 PT5 – Shared access to Private files
854 PT6 – ms Messenger
855 PT7 – Cyber Cash
856 PT8 – Worldwide Voting System

857 The inter-linked functional elements are:
858 P1 – Peer Ranking
859 P2 – Self Healing
860 P3 – Security Availability
861 P4 – Storage and Retrieval
862 P5 – Duplicate Removal
863 P6 – Storing Files
864 ~~P7 – Chunking~~
865 P8 – Encryption / Decryption
866 P9 – Identify Chunks
867 P10 – Revision Control
868 P11 – Identify Data with Very Small File
869 P12 – Logon
870 P13 – Provide Key Pairs
871 P14 – Validation
872 P15 – Create Map of Maps

873 P16 – Share Map
 874 P17 – Provide Public ID
 875 P18 – Encrypted Communications
 876 P19 – Document Signing
 877 P20 – Contract Conversations
 878 P21 – Counterfeit Prevention
 879 P22 – Allow Selling of Machine Resources
 880 P23 – Interface with Non-Anonymous Systems
 881 P24 – Anonymous Transactions
 882 P25 - Anonymity
 883 P26 – Proven Individual
 884 P27 – Validation of Vote Being Used
 885 P28 – Distributed Controlled Voting
 886
 887 (description of figure 1 here ****)
 888

Self Authentication Detail (Figure 2)

- 889 1. A computer program consisting of a user interface and a chunk server (a
 890 system to process anonymous chunks of data) should be running, if not
 891 they are started when user selects an icon or other means of starting the
 892 program.
- 893 2. A user will input some data known to them such as a userid (random ID)
 894 and PIN number in this case. These pieces of information may be
 895 concatenated together and hashed to create a unique (which may be
 896 confirmed via a search) identifier. In this case this is called the **MID**
 897 (maidsafe.net ID)
- 898 3. A TMID (Today's MID) is retrieved from the network, the TMID is then
 899 calculated as follows:

900 The TMID is a single use or single day ID that is constantly changed.
 901 This allows maidsafe.net to calculate the hash based on the user ID pin
 902 and another known variable which is calculable. For this variable we use
 903 a day variable for now and this is the number of days since epoch
 904 (01/01/1970). This allows for a new ID daily, which assists in maintaining
 905 the anonymity of the user. This TMID will create a temporary key pair to
 906 sign the database chunks and accept a challenge response from the
 907 holder of these db chunks. After retrieval and generation of a new key
 908 pair the db is put again in new locations – rendering everything that was
 909 contained in the TMID chunk useless. The TMID CANNOT be signed by
 910 anyone (therefore hackers can't BAN an unsigned user from retrieving
 911 this – in a DOS attack)– it is a special chunk where the data hash does
 912 NOT match the name of the chunk (as the name is a random number
 913 calculated by hashing other information (i.e. its a hash of the TMID as
 914 described below)

- 915 • take dave as user ID and 1267 as pin.
- 916 • dave + (pin) 1267 = dave1267 Hash of this becomes MID
- 917 • day variable (say today is 13416 since epoch) = 13416
- 918 • so take pin, and for example add the number in where the pin states
- 919 i.e.
- 920 • 613dav41e1267
- 921 • (6 at beginning is going round pin again)
- 922 • so this is done by taking 1st pin 1 - so put first day value at position 1
- 923 • then next pin number 2 - so day value 2 at position 2
- 924 • then next pin number 6 so day value 3 at position 6
- 925 • then next pin number 7 so day value 4 at position 7
- 926 • then next pin number is 1 so day value 5 at position 1 (again)
- 927 • so TMID is hash of 613dav41e1267 and the MID is simply a hash of
- 928 dave1267

- 929 (This is an example algorithm and many more can be used to enforce
930 further security.)
- 931 4. From the TMID chunk the map of the user's database (or list of files
932 maps) is identified. The database is recovered from the net which
933 includes the data maps for the user and any keys passwords etc.. The
934 database chunks are stored in another location immediately and the old
935 chunks forgotten. This can be done now as the MID key pair is also in
936 the database and can now be used to manipulate user's data.
- 937 5. The maidsafe.net application can now authenticate itself as acting for
938 this MID and put get or forget data chunks belonging to the user.
- 939 6. The watcher process and Chunk server always have access to the PMID
940 key pair as they are stored on the machine itself, so can start and
941 receive and authenticate anonymous put / get / forget commands.
- 942 7. A DHT ID is required for a node in a DHT network this may be randomly
943 generated or in fact we can use the hash of the PMID public key to
944 identify the node.
- 945 8. When the users successfully logged in he can check his authentication
946 validation records exist on the network. These may be as follows:

MAID (maidsafe.net anonymous ID)

- 947 1. This is a data element stored on net and preferably named with the hash
948 of the MID public Key.
- 949 2. It contains the MID public key + any PMID public keys associated with
950 this user.

- 951 3. This is digitally signed with the MID private key to prevent forgery.
- 952 4. Using this mechanism this allows validation of MID signatures by
- 953 allowing any users access to this data element and checking the
- 954 signature of it against any challenge response from any node pertaining
- 955 to be this MID (as only the MID owner has the private key that signs this
- 956 MID) Any crook could not create the private key to match to the public
- 957 key to digitally sign so forgery is made impossible given today's
- 958 computer resources.
- 959 5. This mechanism also allows a user to add or remove PMIDS (or chunk
- 960 servers acting on their behalf like a proxy) at will and replace PMID's at
- 961 any time in case of the PMID machine becoming compromised.
- 962 Therefore this can be seen as the PMID authentication element.

PMID (Proxy MID)

- 963 1. This is a data element stored on the network and preferably named with
- 964 the hash of the PMID public key.
- 965 2. It contains the PMID public key and the MID ID (i.e. the hash of the MID
- 966 public key) and is signed by the MID private key (authenticated).
- 967 3. This allows a machine to act as a repository for anonymous chunks and
- 968 supply resources to the net for a MID.
- 969 4. When answering challenge responses any other machine will confirm the
- 970 PMID by seeking and checking the MIAD for the PMID and making sure
- 971 the PMID is mentioned in the MAID bit – otherwise the PMID is
- 972 considered rouge.

- 973 5. The key pair is stored on the machine itself and may be encoded or
974 encrypted against a password that has to be entered upon start-up
975 (optionally) in the case of a proxy provider who wishes to further
976 enhance PMID security.
- 977 6. The design allows for recovery from attack and theft of the PMID key pair
978 as the MAID data element can simply remove the PMID ID from the
979 MAID rendering it unauthenticated.

980 Figure 3 illustrates, in schematic form, a peer-to-peer network in
981 accordance with an embodiment of the invention; and

982 Figure 4 illustrates a flow chart of the authentication, in accordance with
983 a preferred embodiment of the present invention.

984 With reference to Figure 3, a peer-to-peer network 2 is shown with nodes
985 4 to 12 connected by a communication network 14. The nodes may be
986 Personal Computers (PCs) or any other device that can perform the
987 processing, communication and/or storage operations required to
988 operate the invention. The file system will typically have many more
989 nodes of all types than shown in Figure 3 and a PC may act as one or
990 many types of node described herein. Data nodes 4 and 6 store chunks
991 16 of files in the distributed system. The validation record node 8 has a
992 storage module 18 for storing encrypted validation records identified by a
993 user-identifier.

994 The client node 10 has a module 20 for input and generation of user
995 identifiers. It also has a decryption module 22 for decrypting an encrypted
996 validation record so as to provide decrypted information, a database or
997 data map of chunk locations 24 and storage 26 for retrieved chunks and
998 files assembled from the retrieved chunks.

999 The verifying node 12 has a receiving module 28 for receiving a user
 1000 identifier from the client node. The retrieving module 30 is configured to
 1001 retrieve from the data node an encrypted validation record identified by
 1002 the user identifier. Alternatively, in the preferred embodiment, the
 1003 validation record node 8 is the same node as the verifying node 12, i.e.
 1004 the storage module 18 is part of the verifying node 12 (not as shown in
 1005 Figure 3). The transmitting module 32 sends the encrypted validation
 1006 record to the client node. The authentication module 34 authenticates
 1007 access to chunks of data distributed across the data nodes using the
 1008 decrypted information.

1009 With reference to Figure 4, a more detailed flow of the operation of the
 1010 present invention is shown laid out on the diagram with the steps being
 1011 performed at the User's PC (client node) on the left 40, those of the
 1012 verifying PC (node) in the centre 42 and those of the data PC (node) on
 1013 the right 44.

1014 A login box is presented 46 that requires the user's name or other detail
 1015 Preferably email address (the same one used in the client node software
 1016 installation and registration process) or simply name (i.e. nickname) and
 1017 the user's unique number, preferably PIN number. If the user is a 'main
 1018 user' then some details may already be stored on the PC. If the user is a
 1019 visitor, then the login box appears.

1020 A content hashed number such as SHA (Secure Hash Algorithm),
 1021 Preferably 160 bits in length, is created 48 from these two items of data.
 1022 This 'hash' is now known as the 'User ID Key' (MID), which at this point is
 1023 classed as 'unverified' within the system. This is stored on the network as
 1024 the MAID and is simply the hash of the public key containing an
 1025 unencrypted version of the public key for later validation by any other
 1026 node. This obviates the requirement for a validation authority

1027 The software on the user's PC then combines this MID with a standard
1028 'hello' code element 50, to create 52 a 'hello.packet'. This hello.packet is
1029 then transmitted with a timed validity on the Internet.

1030 The hello.packet will be picked up by the first node (for this description,
1031 now called the 'verifying node') that recognises 54 the User ID Key
1032 element of the hello.packet as matching a stored, encrypted validation
1033 record file 56 that it has in its storage area. A login attempt monitoring
1034 system ensures a maximum of three responses. Upon to many attempts,
1035 the verifying PC creates a 'black list' for transmission to peers.
1036 Optionally, an alert is returned to the user if a 'black list' entry is found
1037 and the user may be asked to proceed or perform a virus check.

1038 The verifying node then returns this encrypted validation record file to the
1039 user via the internet. The user's pass phrase 58 is requested by a dialog
1040 box 60, which then will allow decryption of this validation record file.

1041 When the validation record file is decrypted 62, the first data chunk
1042 details, including a 'decrypted address', are extracted 64 and the user PC
1043 sends back a request 66 to the verifying node for it to initiate a query for
1044 the first 'file-chunk ID' at the 'decrypted address' that it has extracted
1045 from the decrypted validation record file, or preferably the data map of
1046 the database chunks to recreate the database and provide access to the
1047 key pair associated with this MID.

1048 The verifying node then acts as a 'relay node' and initiates a 'notify only'
1049 query for this 'file-chunk ID' at the 'decrypted address'.

1050 Given that some other node (for this embodiment, called the 'data node')
1051 has recognised 68 this request and has sent back a valid 'notification
1052 only' message 70 that a 'file-chunk ID' corresponding to the request sent
1053 by the verifying node does indeed exist, the verifying node then digitally
1054 signs 72 the initial User ID Key, which is then sent back to the user.

1055 On reception by the user 74, this verified User ID Key is used as the
1056 user's session passport. The user's PC proceeds to construct 76 the
1057 database of the file system as backed up by the user onto the network.
1058 This database describes the location of all chunks that make up the
1059 user's file system. Preferably the ID Key will contain irrefutable evidence
1060 such as a public/private key pair to allow signing onto the network as
1061 authorised users, preferably this is a case of self signing his or her own
1062 ID – in which case the ID Key is decrypted and user is valid – self
1063 validating.

1064 Further details of the embodiment will now be described. A 'proxy-
1065 controlled' handshake routine is employed through an encrypted point-to-
1066 point channel, to ensure only authorised access by the legal owner to the
1067 system, then to the user's file storage database, then to the files therein.
1068 The handshaking check is initiated from the PC that a user logs on to
1069 (the 'User PC'), by generating the 'unverified encrypted hash' known as
1070 the 'User ID Key', this preferably being created from the user's
1071 information preferably email address and their PIN number. This 'hash'
1072 is transmitted as a 'hello.packet' on the Internet, to be picked up by any
1073 system that recognises the User ID as being associated with specific
1074 data that it holds. This PC then becomes the 'verifying PC' and will
1075 initially act as the User PC's 'gateway' into the system during the
1076 authentication process. The encrypted item of data held by the verifying
1077 PC will temporarily be used as a 'validation record', it being directly
1078 associated with the user's identity and holding the specific address of a
1079 number of data chunks belonging to the user and which are located
1080 elsewhere in the peer-to-peer distributed file system. This 'validation
1081 record' is returned to the User PC for decryption, with the expectation
1082 that only the legal user can supply the specific information that will allow
1083 its accurate decryption.

1084 Preferably this data may be a signed response being given back to the
1085 validating node which is possible as the id chunk when decrypted
1086 (preferably symmetrically) contains the users public and private keys
1087 allowing non refutable signing of data packets.

1088 Preferably after successful decryption of the TMID packet (as described
1089 above) the machine will now have access to the data map of the
1090 database and public/private key pair allowing unfettered access to the
1091 system.

1092 It should be noted that in this embodiment, preferably no communication
1093 is carried out via any nodes without an encrypted channel such as TLS
1094 (Transport Layer Security) or SSL (Secure Sockets Layer) being set up
1095 first. A peer talks to another peer via an encrypted channel and the other
1096 peer (proxy) requests the information (e.g. for some space to save
1097 information on or for the retrieval of a file). An encrypted link is formed
1098 between all peers at each end of communications and also through the
1099 proxy during the authentication process. This effectively bans snoopers
1100 from detecting who is talking to whom and also what is being sent or
1101 retrieved. The initial handshake for self authentication is also over an
1102 encrypted link.

1103 Secure connection is provided via certificate passing nodes, in a manner
1104 that does not require intervention, with each node being validated by
1105 another, where any invalid event or data, for whatever reason (fraud
1106 detection, snooping from node or any invalid algorithms that catch the
1107 node) will invalidate the chain created by the node. This is all transparent
1108 to the user.

1109 Further modifications and improvements may be added without departing
1110 from the scope of the invention herein described.

- 1111 Figure 5 illustrates a flow chart of data assurance event sequence in
1112 accordance with first embodiment of this invention
- 1113 Figure 6 illustrates a flow chart of file chunking event sequence in
1114 accordance with second embodiment of this invention
- 1115 Figure 7 illustrates a schematic diagram of file chunking example
- 1116 Figure 8 illustrates a flow chart of self healing event sequence
- 1117 Figure 9 illustrates a flow chart of peer ranking event sequence
- 1118 Figure 10 illustrates a flow chart of duplicate removal event sequence
- 1119 With reference to Figure 5, guaranteed accessibility to user data by data
1120 assurance is demonstrated by flow chart. The data is copied to at least
1121 three disparate locations at step (10). The disparate locations store data
1122 with an appendix pointing to the other two locations by step (20) and is
1123 renamed with hash of contents. Preferably this action is managed by
1124 another node i.e. super node acting as an intermediary by step (30).
- 1125 Each local copy at user's PC is checked for validity by integrity test by
1126 step (40) and in addition validity checks by integrity test are made that
1127 the other 2 copies are also still ok by step (50).
- 1128 Any single node failure initiates a replacement copy of equivalent leaf
1129 node being made in another disparate location by step (60) and the other
1130 remaining copies are updated to reflect this change to reflect the newly
1131 added replacement leaf node by step (70).
- 1132 The steps of storing and retrieving are carried out via other network
1133 nodes to mask the initiator (30).

1134 The method further comprises the step of renaming all files with a hash
1135 of their contents.

1136 Therefore, each file can be checked for validity or tampering by running a
1137 content hashing algorithm such as (for example) MD5 or an SHA variant,
1138 the result of this being compared with the name of the file.

1139 With reference to Figure 6, provides a methodology to manageable sized
1140 data elements and to enable a complimentary data structure for and
1141 compression and encryption and the step is to file chunking. By user's
1142 pre-selection the nominated data elements (files are passed to chunking
1143 process. Each data element (file) is split into small chunks by step (80)
1144 and the data chunks are encrypted by step (90) to provide security for the
1145 data. The data chunks are stored locally at step (100) ready for network
1146 transfer of copies. Only the person or the group, to whom the overall data
1147 belongs, will know the location of these (100) or the other related but
1148 dissimilar chunks of data. All operations are conducted within the user's
1149 local system. No data is presented externally.

1150 Each of the above chunks does not contain location information for any
1151 other dissimilar chunks. This provides for, security of data content, a
1152 basis for integrity checking and redundancy.

1153 The method further comprises the step of only allowing the person (or
1154 group) to whom the data belongs, to have access to it, preferably via a
1155 shared encryption technique. This allows persistence of data.

1156 The checking of data or chunks of data between machines is carried out
1157 via any presence type protocol such as a distributed hash table network.

1158 On the occasion when all data chunks have been relocated (i.e. the user
1159 has not logged on for a while,) a redirection record is created and stored
1160 in the super node network, (a three copy process – similar to data)

1161 therefore when a user requests a check, the redirection record is given to
1162 the user to update their database.

1163 This efficiently allows data resilience in cases where network churn is a
1164 problem as in peer to peer or distributed networks.

1165 With reference to Figure 7 which illustrates flow chart example of file
1166 chunking. User's normal file has 5Mb document, which is chunked into
1167 smaller variable sizes e.g. 135kb, 512kb, 768kb in any order. All chunks
1168 may be compressed and encrypted by using Pass phrase. Next step is to
1169 individually hash chunks and given hashes as names. Then database
1170 record as a file is made from names of hashed chunks brought together
1171 e.g. in empty version of original file (C1#####t1,t2,t3:
1172 C2#####t1,t2,t3 etc), this file is then sent to transmission queue in
1173 storage space allocated to client application.

1174 With reference to Figure 8 provides a self healing event sequence
1175 methodology. Self healing is required to guarantee availability of accurate
1176 data. As data or chunks become invalid by failing integrity test by step
1177 (110). The location of failing data chunks is assessed as unreliable and
1178 further data from the leaf node is ignored from that location by step (120).
1179 A 'Good Copy' from the 'known good' data chunk is recreated in a new
1180 and equivalent leaf node. Data or chunks are recreated in a new and
1181 safer location by step (130). The leaf node with failing data chunks is
1182 marked as unreliable and the data therein as 'dirty' by step (140). Peer
1183 leaf nodes become aware of this unreliable leaf node and add its location
1184 to watch list by step (150). All operations conducted within the user's
1185 local system. No data is presented externally.

1186 Therefore, the introduction of viruses, worms etc. will be prevented and
1187 faulty machines/ equipment identified automatically.

1188 The network will use SSL or TLS type encryption to prevent unauthorised
1189 access or snooping.

1190 With reference to Figure 9, Peer Ranking id required to ensure consistent
1191 response and performance for the level of guaranteed interaction
1192 recorded for the user. For Peer Ranking each node (leaf node) monitors
1193 its own peer node's resources and availability in a scaleable manner,
1194 each leaf node is constantly monitored.

1195 Each data store (whether a network service, physical drive etc.) is
1196 monitored for availability. A qualified availability ranking is appended to
1197 the (leaf) storage node address by consensus of a monitoring super node
1198 group by step (160). A ranking figure will be appended by step (160) and
1199 signed by the supply of a key from the monitoring super node; this would
1200 preferably be agreed by more super nodes to establish a consensus for
1201 altering the ranking of the node. The new rank will preferably be
1202 appended to the node address or by a similar mechanism to allow the
1203 node to be managed preferably in terms of what is stored there and how
1204 many copies there has to be of the data for it to be seen as perpetual.

1205 Each piece of data is checked via a content hashing mechanism for data
1206 integrity, which is carried out by the storage node itself by step (170) or
1207 by its partner nodes via super nodes by step (180) or by instigating node
1208 via super nodes by step (190) by retrieval and running the hashing
~~209 algorithm against that piece of data. The data checking cycle repeats~~
1210 itself.

1211 As a peer (whether an instigating node or a partner peer (i.e. one that
1212 has same chunk)) checks the data, the super node querying the storage
1213 peer will respond with the result of the integrity check and update this
1214 status on the storage peer. The instigating node or partner peer will
1215 decide to forget this data and will replicate it in a more suitable location.

WO 2008/065341

PCT/GB2007/004421

46

1216 If data fails the integrity check the node itself will be marked as 'dirty' by
1217 step (200) and 'dirty' status appended to leaf node address to mark it as
1218 requiring further checks on the integrity of the data it holds by step (210).
1219 Additional checks are carried out on data stored on the leaf node marked
1220 as 'dirty' by step (220). If pre-determined percentage of data found to be
1221 'dirty' node is removed from the network except for message traffic by
1222 step (230). A certain percentage of dirty data being established may
1223 conclude that this node is compromised or otherwise damaged and the
1224 network would be informed of this. At that point the node will be removed
1225 from the network except for the purpose of sending it warning messages
1226 by step (230).

1227 This allows either having data stored on nodes of equivalent availability
1228 and efficiency or dictating the number of copies of data required to
1229 maintain reliability.

1230 Further modifications and improvements may be added without departing
1231 from the scope of the invention herein described.

1232 With reference to Figure 10, duplicate data is removed to maximise the
1233 efficient use of the disk space. Prior to the initiation of the data backup
1234 process by step (240), internally generated content hash may be
1235 checked for a match against hashes stored on the internet by step (250)
1236 or a list of previously backed up data (250). This will allow only one
1237 backed up copy of data to be kept. This reduces the network wide
1238 requirement to backup data which has the exact same contents.
1239 Notification of shared key existence is passed back to instigating node by
1240 step (260) to access authority check requested, which has to pass for
1241 signed result is to be passed back to storage node. The storage node
1242 passes shared key and database back to instigating node by step (270)

1243 Such data is backed up via a shared key which after proof of the file
1244 existing (260) on the instigating node, the shared key (270) is shared with

1245 this instigating node. The location of the data is then passed to the node
1246 for later retrieval if required.

1247 This maintains copyright as people can only backup what they prove to
1248 have on their systems and not publicly share copyright infringed data
1249 openly on the network.

1250 This data may be marked as protected or not protected by step (280)
1251 which has check carried out for protected or non-protected data content.
1252 The protected data ignores sharing process.

Perpetual Data (Figure 1 – PT1 and Figure 11)

1253 ***According to a related aspect of this invention***, a file is chunked or
1254 split into constituent parts (1) this process involves calculating the chunk
1255 size, preferably from known data such as the first few bytes of the hash
1256 of the file itself and preferably using a modulo division technique to
1257 resolve a figure between the optimum minimum and optimum maximum
1258 chunk sizes for network transmission and storage.

1259 Preferably each chunk is then encrypted and obfuscated in some manner
1260 to protect the data. Preferably a search of the network is carried out
1261 looking for values relating to the content hash of each of the chunks (2).

1262 If this is found (4) then the other chunks are identified too, failure to
1263 identify all chunks may mean there is a collision on the network of file
1264 names or some other machine is in the process of backing up the same
1265 file. A back-off time is calculated to check again for the other chunks. If
1266 all chunks are on the network the file is considered backed up and the
1267 user will add their MID signature to the file after preferably a challenge
1268 response to ensure there a valid user and have enough resources to do
1269 this.

1270 If no chunks are on the net the user preferably via another node (3) will
1271 request the saving of the first copy (preferably in distinct time zones or
1272 other geographically dispersing method).

1273 The chunk will be stored (5) on a storage node allowing us to see the
1274 PMID of the storing node and store this.

1275 Then preferably a Key.value pair of chunkid.public key of initiator is
1276 written to net creating a Chunk ID (CID) (6).

Storage and Retrieval (Figure 1- P4)

1277 ***According to a related aspect of this invention,*** the data is stored in
1278 multiple locations. Each location stores the locations of its peers that hold
1279 identical chunks (at least identical in content) and they all communicate
1280 regularly to ascertain the health of the data. The preferable method is as
1281 follows:

1282 Preferably the data is copied to at least three disparate locations.

1283 Preferably each copy is performed via many nodes to mask the initiator.

1284 Preferably each local copy is checked for validity and checks are made
1285 that the preferably other 2 copies are also still valid.

1286 Preferably any single node failure initiates a replacement copy being
1287 made in another disparate location and the other associated copies are
1288 updated to reflect this change.

1289 Preferably the steps of storing and retrieving are carried out via other
1290 network nodes to mask the initiator.

- 1291 Preferably, the method further comprises the step of renaming all files
1292 with a hash of their contents.
- 1293 Preferably each chunk may alter its name by a known process such as a
1294 binary shift left of a section of the data. This allows the same content to
1295 exist but also allows the chunks to appear as three different bits of data
1296 for the sake of not colliding on the network.
- 1297 Preferably each chunk has a counter attached to it that allows the
1298 network to understand easily just how many users are attached to the
1299 chunk – either by sharing or otherwise. A user requesting a 'chunk forget'
1300 will initiate a system question if they are the only user using the chunk
1301 and if so the chunk will be deleted and the user's required disk space
1302 reduced accordingly. This allows users to remove files no longer required
1303 and free up local disk space. Any file also being shared is preferably
1304 removed from the user's quota and the user's database record or data
1305 map (see later) is deleted.
- 1306 Preferably this counter is digitally signed by each node sharing the data
1307 and therefore will require a signed 'forget' or 'delete' command.
1308 Preferably even 'store', 'put', 'retrieve' and 'get' commands should also
1309 be either digitally signed or preferably go through a PKI challenge
1310 response mechanism.
- 1311 To ensure fairness preferably this method will be monitored by a
1312 supernode or similar to ensure the user has not simply copied the data
1313 map for later use without giving up the disk space for it. Therefore the
1314 user's private ID public key will be used to request the forget chunk
1315 statement. This will be used to indicate the user's acceptance of the
1316 'chunk forget' command and allow the user to recover the disk space.
1317 Any requests against the chunk will preferably be signed with this key

- 1318 and consequently rejected unless the user's system gives up the space
1319 required to access this file.
- 1320 Preferably each user storing a chunk will append their signed request to
1321 the end of the chunk in an identifiable manner i.e. prefixed with 80 – or
1322 similar.
- 1323 Forgetting the chunk means the signature is removed from the file. This
1324 again is done via a signed request from the storage node as with the
1325 original backup request.
- 1326 Preferably this signed request is another small chunk stored at the same
1327 location as the data chunk with an appended postfix to the chunk
1328 identifier to show a private ID is storing this chunk. Any attempt by
1329 somebody else to download the file is rejected unless they first subscribe
1330 to it, i.e. a chunk is called 12345 so a file is saved called 12345 <signed
1331 store request>. This will allow files to be forgotten when all signatories to
1332 the chunk are gone. A user will send a signed 'no store' or 'forget' and
1333 their ID chunk will be removed, and in addition if they are the last user
1334 storing that chunk, the chunk is removed. Preferably this will allow a
1335 private anonymous message to be sent upon chunk failure or damage
1336 allowing a proactive approach to maintaining clean data.
- 1337 Preferably as a node fails the other nodes can preferably send a
1338 ~~message to all sharers of the chunk to identify the new location of the~~
1339 replacement chunk.
- 1340 Preferably any node attaching to a file then downloading immediately
1341 should be considered an alert and the system may take steps to slow
1342 down this node's activity or even halt it to protect data theft.

Chunk Checks: (Figure 1 – P9 and Figure 12)

- 1343 1. Storage node containing chunk 1 checks its peers. As each peer is
1344 checked it reciprocates the check. These checks are split into preferably
1345 2 types:
- 1346 a. Availability check (i.e. simple network ping)
1347 b. Data integrity check – in this instance the checking node takes a chunk
1348 and appends random data to it and takes a hash of the result. It then
1349 sends the random data to the node being checked and requests the
1350 hash of the chunk with the random data appended. The result is
1351 compared with a known result and the chunk will be assessed as
1352 either healthy or not. If not, further checks with other nodes occur to
1353 find the bad node.
- 1354 2. There may be multiple storage nodes depending on the rating of
1355 machines and other factors. The above checking is carried out by all
1356 nodes from 1 to n (where n is total number of storage nodes selected for
1357 the chunk). Obviously a poorly rated node will require to give up disk
1358 space in relation to the number of chunks being stored to allow perpetual
1359 data to exist. This is a penalty paid by nodes that are switched off.
- 1360 3. The user who stored the chunk will check on a chunk from 1 storage
1361 node randomly selected. This check will ensure the integrity of the chunk
1362 ~~and also ensure there are at least 10 other signatures existing already for~~
1363 the chunk. If there are not and the user's ID is not listed, the user signs
1364 the chunk.
- 1365 4. This shows another example of another user checking the chunk. Note
1366 that the user checks X (40 days in this diagram) are always at least 75%
1367 of the forget time retention (Y) (i.e. when a chunk is forgotten by all
1368 signatories it is retained for a period of time Y). This is another algorithm
1369 that will continually develop.

Storage of Additional Chunks: (Figure 12)

- 1370 1. maidsafe.net program with user logged in (so MID exists) has chunked a
 1371 file. It has already stored a chunk and is now looking to store additional
 1372 chunks. Therefore a Chunk ID (CID) should exist on the net. This process
 1373 retrieves this CID.

- 1374 2. The CID as shown in storing initial chunk contains the chunk name and
 1375 any public keys that are sharing the chunk. In this instance it should only
 1376 be our key as we are first ones storing the chunks (others would be in a
 1377 back-off period to see if we back other chunks up). We shift the last bit
 1378 (could be any function on any bit as long as we can replicate it)

- 1379 3. We then check we won't collide with any other stored chunk on the net –
 1380 i.e. it does a CID search again.

- 1381 4. We then issue our broadcast to our supernodes (i.e. the supernodes we
 1382 are connected to) stating we need to store X bytes and any other
 1383 information about where we require to store it (geographically in our case
 1384 – time zone (TZ))

- 1385 5. The supernode network finds a storage location for us with the correct
 1386 - rank etc.

- 1387 6. The chunk is stored after a successful challenge response i.e. In the
 1388 maidsafe.net network. MIDs will require to ensure they are talking or
 1389 dealing with validated nodes, so to accomplish this a challenge process
 1390 is carried out as follows: sender [S] receiver [R]

- 1391 • [S] I wish to communicate (store retrieve forget data etc.) and I am MAID

WO 2008/065341

PCT/GB2007/004421

53

- 1392 • [R] retrieves MAID public key from DHT and encrypts a challenge
 - 1393 (possibly a very large number encrypted with the public key retrieved)
 - 1394 • [S] gets key and decrypts and encrypts [R] answer with his challenge
 - 1395 number also encrypted with [R]'s public key
 - 1396 • [R] receives response and decrypts his challenge and passes back
 - 1397 answer encrypted again with [S] public key
 - 1398 (Communication is now authenticated between these two nodes.)
- 1399 7. The CID is then updated with the second chunk name and the location it
- 1400 is stored at. This process is repeated for as many copies of a chunk that
- 1401 are required.
- 1402 8. Copies of chunks will be dependent on many factors including file
- 1403 popularity (popular files may require to be more dispersed closer to
- 1404 nodes and have more copies. Very poorly ranked machines may require
- 1405 an increased amount of chunks to ensure they can be retrieved at any
- 1406 time (poorly ranked machines will therefore have to give up more space.)

Security Availability (Figure 1 – P3)

- 1407 ***According to a related aspect of this invention***, each file is split into
- 1408 small chunks and encrypted to provide security for the data. Only the
- 1409 person or the group, to whom the overall data belongs, will know the
- 1410 ~~location of the other related but dissimilar chunks of data.~~
- 1411 Preferably, each of the above chunks does not contain location
- 1412 information for any other dissimilar chunks; which provides for security of
- 1413 data content, a basis for integrity checking and redundancy.
- 1414 Preferably, the method further comprises the step of only allowing the
- 1415 person (or group) to whom the data belongs to have access to it,

1416 preferably via a shared encryption technique which allows persistence of
1417 data.

1418 Preferably, the checking of data or chunks of data between machines is
1419 carried out via any presence type protocol such as a distributed hash
1420 table network.

1421 Preferably, on the occasion when all data chunks have been relocated,
1422 i.e. the user has not logged on for a while, a redirection record is created
1423 and stored in the super node network, (a three copy process – similar to
1424 data) therefore when a user requests a check, the redirection record is
1425 given to the user to update their database, which provides efficiency that
1426 in turn allows data resilience in cases where network churn is a problem
1427 as in peer to peer or distributed networks. This system message can be
1428 preferably passed via the messenger system described herein.

1429 Preferably the system may simply allow a user to search for his chunks
1430 and through a challenge response mechanism, locate and authenticate
1431 himself to have authority to get/forget this chunk.

1432 Further users can decide on various modes of operation preferably such
1433 as maintain a local copy of all files on their local machine, unencrypted or
1434 chunked or chunk and encrypt even local files to secure machine
1435 (preferably referred to as off line mode operation) or indeed users may
1436 decide to remove all local data and rely completely on preferably
1437 maidsafe.net or similar system to secure their data.

Self Healing (Figure 1 – P2)

1438 **According to a related aspect of this invention,** a self healing network
1439 method is provided via the following process;

- 1440 • As data or chunks become invalid – data is ignored from that location
- 1441 • Data or chunks are recreated in a new and safer location.
- 1442 • The original location is marked as bad.
- 1443 • Peers note this condition and add the bad location to a watch list.

1444 This will prevent the introduction of viruses; worms etc. will allow faulty
 1445 machines/ equipment to be identified automatically.

1446 Preferably, the network layer will use SSL or TLS channel encryption to
 1447 prevent unauthorised access or snooping.

Self Healing (Figure 13)

- 1448 1. A data element called a Chunk ID (CID) is created for each chunk. Added
 1449 to this is the 'also stored at' MID for the other identical chunks. The other
 1450 chunk names are also here as they may be renamed slightly (i.e. by bit
 1451 shifting a part of the name in a manner that calculable).
- 1452 2. All storing nodes (related to this chunk) have a copy of this CID file or
 1453 can access it at any stage from the DHT network, giving each node
 1454 knowledge of all others.
- 1455 3. Each of the storage nodes have their copy of the chunk.
- 1456 4. Each node queries its partner's availability at frequent intervals. On less
 1457 frequent intervals a chunk health check is requested. This involves a
 1458 node creating some random data and appending this to it's chunk and
 1459 taking the hash. The partner node will be requested to take the random
 1460 data and do likewise and return the hash result. This result is checked
 1461 against the result the initiator had and chunk is then deemed healthy or
 1462 not. Further tests can be done as each node knows the hash their chunk

WO 2008/065341

PCT/GB2007/004421

56

- 1463 should create and can self check in that manner on error and report a
1464 dirty node.
- 1465 5. Now we have a node fail (creating a dirty chunk)
- 1466 6. The first node to note this carries out a broadcast to other nodes to say it
1467 is requesting a move of the data.
- 1468 7. The other nodes agree to have CID updated (they may carry out their
1469 own check to confirm this).
- 1470 8. A broadcast is sent to the supernode network closest to the storage node
1471 that failed, to state a re-storage requirement.
- 1472 9. The supernode network picks up the request.
- 1473 10. The request is to the supernode network to store x amount of data at a
1474 rank of y.
- 1475 11. A supernode will reply with a location
- 1476 12. The storage node and new location carry out a challenge response
1477 request to validate each other.
- 1478 13. The chunk is stored and the CID is updated and signed by the three or
1479 more nodes storing the chunk.

Peer Ranking (Figure 1 – P1)

- 1480 **According to a related aspect of this invention,** there is the addition of
1481 a peer ranking mechanism, where each node (leaf node) monitors its

1482 own peer node's resources and availability in a scalable manner. Nodes
1483 constantly perform this monitoring function.

1484 Each data store (whether a network service, physical drive etc.) is
1485 monitored for availability. A ranking figure is appended and signed by the
1486 supply of a key from the monitoring super node, this being preferably
1487 agreed by more super nodes to establish a consensus before altering the
1488 ranking of the node. Preferably, the new rank will be appended to the
1489 node address or by a similar mechanism to allow the node to be
1490 managed in terms of what is stored there and how many copies there
1491 has to be of the data for it to be seen as perpetual.

1492 Each piece of data is checked via a content hashing mechanism. This is
1493 preferably carried out by the storage node itself or by its partner nodes
1494 via super nodes or by an instigating node via super nodes by retrieving
1495 and running the hashing algorithm against that piece of data.

1496 Preferably, as a peer (whether an instigating node or a partner peer (i.e.
1497 one that has same chunk)) checks the data, the super node querying the
1498 storage peer will respond with the result of the integrity check and update
1499 this status on the storage peer. The instigating node or partner peer will
1500 decide to forget this data and will replicate it in a more suitable location.

1501 If data fails the integrity check, the node itself will be marked as 'dirty' and
1502 this status will preferably be appended to the node's address for further
1503 checks on other data to take this into account. Preferably a certain
1504 percentage of dirty data being established may conclude that this node is
1505 compromised or otherwise damaged and the network would be informed
1506 of this. At that point the node will be removed from the network except for
1507 the purpose of sending it warning messages.

1508 In general, the node ranking figure will take into account at least;
1509 availability of the network connection, availability of resources, time on

1510 the network with a rank (later useful for effort based trust model), amount
1511 of resource (including network resources) and also the connectivity
1512 capabilities of any node (i.e. directly or indirectly contactable)

1513 This then allows data to be stored on nodes of equivalent availability and
1514 efficiency, and to determine the number of copies of data required to
1515 maintain reliability.

Aput: (Figure 15)

1516 Here the MID is the MID of the machine saving data to the net and the
1517 PMID is the ID of the storage node chunk server. The communication is
1518 therefore between a maidsafe.net application with a logged in user (to
1519 provide MID) and a chunking system on the net somewhere (storage
1520 node).

- 1521 1. A message signed with a user's MID (checked by getting the MAID
1522 packet from the net) is received requesting storage of a data chunk.
- 1523 2. This message is a specific message stating the storage node's ID (PMID)
1524 and the chunk name to be saved and signed (i.e. this is a unique
1525 message)
- 1526 3. The chunk server decides if it will store the chunk.
- 1527 4. A signed message is returned stating if PMID will store this chunk
1528 (chunkID).
- 1529 5. The chunk is stored and checked (SHA check)
- 1530 6. A message is sent back to state that the chunk is saved and is ok. This is
1531 signed by the PMID of the chunk server.

- 1532 7. The chunk server awaits the locations of the other identical chunks.
- 1533 8. Locations of the identical chunks returned to the chunk server signed with
1534 the MID.
- 1535 9. Each storage node is contacted and public keys exchanged (PMIDs)
- 1536 10. The chunk checking process is initiated.

Aforget (Figure 16)

- 1537 1. A user has requested that a file should be deleted from his backup
1538 (forgotten). The system signs a request using the user MID.
- 1539 2. The request is sent to a chunk server (storage node).
- 1540 3. The storage node picks up the request
- 1541 4. The storage node sends the signed request to the other storage nodes
1542 that have this chunk.
- 1543 5. The MID is checked as being on the list of MIDs that are watching the
1544 chunk-(remember only a few--20-in our case-are ever listed)
- 1545 6. The other storage nodes are notified of this.
- 1546 7. If this is the only MID listed then all owners are possibly gone.
- 1547 8. Chunk delete times begins, this timer will always be higher than a user
1548 check interval – i.e. timer of 60 days – user check interval 40 days.

1549 9. This information is also passed to other storage nodes.

Duplicate Removal (Figure 1 - P5)

1550 **According to a related aspect of this invention,** prior to data being
1551 backed up, the content hash may be checked against a list of previously
1552 backed up data. This will allow only one backed up copy of data to be
1553 kept, thereby reducing the network wide requirement to backup data that
1554 has the exact same content. Preferably this will be done via a simple
1555 search for existence on the net of all chunks of a particular file.

1556 Preferably such data is backed up via a shared key or mechanism of
1557 appending keys to chunks of data. After proof of the file existing on the
1558 instigating node, the shared key is shared with the instigating node and
1559 the storing node issues a challenge response to add their ID to the pool if
1560 it is capable of carrying out actions on the file such as get/ forget (delete).
1561 The location of the data is then passed to the node for later retrieval if
1562 required.

1563 This maintains copyright as people can only backup what they prove to
1564 have on their systems and not easily publicly share copyright infringed
1565 data openly on the network.

1566 Preferably, data may be marked as protected or not protected. Preferably
1567 protected data ignores sharing process.

Chunking (Figure 1 - P7)

1568 **According to a related aspect of this invention,** files are split
1569 preferably using an algorithm to work out the chunk size into several
1570 component parts. The size of the parts is preferably worked out from

1571 known information about the file as a whole, preferably the hash of the
 1572 complete file. This information is run through an algorithm such as adding
 1573 together the first x bits of the known information and using modulo
 1574 division to give a chunk size that allows the file to preferably split into at
 1575 least three parts.

1576 Preferably known information from each chunk is used as an encryption
 1577 key. This is preferably done by taking a hash of each chunk and using
 1578 this as the input to an encryption algorithm to encrypt another chunk in
 1579 the file. Preferably this is a symmetrical algorithm such as AES256.

1580 Preferably this key is input into a password creating algorithm such as
 1581 pbkdf and an initial vector and key calculated from that. Preferably the
 1582 iteration count for the pbkdf is calculated from another piece of known
 1583 information, preferably the sum of bits of another chunk or similar.

1584 Preferably each initial chunk hash and the final hash after encryption are
 1585 stored somewhere for later decryption.

Self Encrypting Files (Figure 1 – PT2 and Figure 17)

- 1586 1. Take a content hash of a file or data element
- 1587 2. ~~Chunk a file with preferably a random calculable size i.e. based on an~~
 1588 algorithm of the content hash (to allow recovery of file). Also obfuscate
 1589 the file such as in 3
- 1590 3. Obfuscate the chunks to ensure safety even if encryption is eventually
 1591 broken (as with all encryption if given enough processing power and time)
- 1592 a. chunk 1 byte 1 swapped with byte1 of chunk 2
- 1593 b. chunk 2 byte 2 swapped with byte 1 chunk 3

WO 2008/065341

PCT/GB2007/004421

62

- 1594 c. chunk 3 byte 2 swapped with byte 2 of chunk 1
1595 d. This repeats until all bytes swapped and then repeats the same
1596 number of times as there are chunks with each iteration making next
1597 chunk first one
1598 e. - i.e. second time round chunk 2 is starting position
- 1599 4. Take hash of each chunk and rename chunk with its hash.
- 1600 5. Take h2 and first x bytes of h3 (6 in our example case) and either use
1601 modulo division or similar to get a random number between 2 fixed
1602 parameter (in our case 1000) to get a variable number. Use the above
1603 random number and h2 as the encryption key to encrypt h1 or use h2 and
1604 the random number as inputs to another algorithm (pdk2 in our case) to
1605 create a key and iv.(initialisation vector)
- 1606 6. This process may be repeated multiple times to dilute any keys
1607 throughout a series of chunks.
- 1608 7. Chunk name i.e. h1 (unencrypted) and h1c (and likewise for each chunk)
1609 written to a location for later recovery of the data. Added to this we can
1610 simply update such a location with new chunks if a file has been altered,
1611 thereby creating a revision control system where each file can be rebuilt
1612 to any previous state.
- 1613 8. -The existence of the chunk will be checked on the net to ensure it is not
1614 already backed up. All chunks may be checked at this time.
- 1615 9. If a chunk exists all chunks must be checked for existence.
- 1616 10. The chunk is saved
- 1617 11. The file is marked as backed up.

WO 2008/065341

PCT/GB2007/004421

63

1618 12. If a collision is detected the process is redone altering the original size
1619 algorithm (2) to create a new chunk set, each system will be aware of this
1620 technique and will do the exact same process till a series of chunks do
1621 not collide. There will be a back off period here to ensure the chunks are
1622 not completed due to the fact another system is backing up the same file.
1623 The original chunk set will be checked frequently in case there are false
1624 chunks or ones that have been forgotten. If the original names become
1625 available the file is reworked using these parameters.

Duplicate Removal (Figure 1 - P5)

1626 **According to a related aspect of this invention**, data chunked and
1627 ready for storing can be stored on a distributed network but a search
1628 should preferably be carried out for the existence of all associated
1629 chunks created. Preferably the locations of the chunks have the same
1630 ranking (From earlier ranking system) as user or better, otherwise the
1631 existing chunks on the net are promoted to a location of equivalent rank
1632 at least. If all chunks exist then the file is considered as already backed
1633 up. If less than all chunks exist then this will preferably be considered as
1634 a collision (after a time period) and the file will be re chunked using the
1635 secondary algorithms (preferably just adjusted file sizes). This allows
1636 duplicate files on any 2 or more machines to be only backed up once,
1637 although through perpetual data several copies will exist of each file, this
1638 is limited to an amount that will maintain perpetual data.

Encrypt – Decrypt (Figure 1 - P8)

1639 **According to a related aspect of this invention**, the actual encrypting
1640 and decrypting is carried out via knowledge of the file's content and this
1641 is somehow maintained (see next). Keys will be generated and preferably
1642 stored for decrypting. Actually encrypting the file will preferably include a

WO 2008/065341

PCT/GB2007/004421

64

1643 compression process and further obfuscation methods. Preferably the
1644 chunk will be stored with a known hash preferably based on the contents
1645 of that chunk.

1646 Decrypting the file will preferably require the collation of all chunks and
1647 rebuilding of the file itself. The file may preferably have its content mixed
1648 up by an obfuscation technique rendering each chunk useless on its own.

1649 Preferably every file will go through a process of byte (or preferably bit)
1650 swapping between its chunks to ensure the original file is rendered
1651 useless without all chunks.

1652 This process will preferably involve running an algorithm which preferably
1653 takes the chunk size and then distributes the bytes in a pseudo random
1654 manner preferably taking the number of chunks and using this as an
1655 iteration count for the process. This will preferably protect data even in
1656 event of somebody getting hold of the encryption keys – as the chunks
1657 data is rendered useless even if transmitted in the open without
1658 encryption.

1659 This defends against somebody copying all data and storing for many
1660 years until decryption of today's algorithms is possible, although this is
1661 many years away.

1662- ~~This also defends against somebody, instead of attempting to decrypt a~~
1663 ~~chunk by creating the enormous amount of keys possible, (in the region~~
1664 ~~of 2^{54}) rather instead creating the keys and presenting chunks to all~~
1665 ~~keys – if this were possible (which is unlikely) a chunk would decrypt.~~
1666 The process defined here makes this attempt useless.

1667 All data will now be considered to be diluted throughout the original
1668 chunks and preferably additions to this algorithm will only strengthen the
1669 process.

Identify Chunks (Figure 1 - P9)

1670 **According to a related aspect of this invention**, a chunk's original
1671 hash or other calculable unique identifier will be stored. This will be
1672 stored with preferably the final chunk name. This aspect defines that
1673 each file will have a separate map preferably a file or database entry to
1674 identify the file and the name of its constituent parts. Preferably this will
1675 include local information to users such as original location and rights
1676 (such as a read only system etc.). Preferably some of this information
1677 can be considered shareable with others such as filename, content hash
1678 and chunks names.

ID Data with Small File (Figure 1 - P11)

1679 **According to a related aspect of this invention**, these data maps may
1680 be very small in relation to the original data itself allowing transmission of
1681 files across networks such as the internet with extreme simplicity,
1682 security and bandwidth efficiency. Preferably the transmission of maps
1683 will be carried out in a very secure manner, but failure to do this is akin to
1684 currently emailing a file in its entirety.

1685 ~~This allows a very small file such as the data map or database record to~~
1686 be shared or maintained by a user in a location not normally large
1687 enough to fit a file system of any great size, such as on a PDA or mobile
1688 phone. The identification of the chunk names, original names and final
1689 names are all that is required to retrieve the chunks and rebuild the file
1690 with certainty.

1691 With data maps in place a user's whole machine, or all its data, can exist
1692 elsewhere. Simply retrieving the data maps of all data, is all that is

WO 2008/065341

PCT/GB2007/004421

66

1693 required to allow the user to have complete visibility and access to all
1694 their data as well as any shared files they have agreed to.

Revision Control (Figure 1 - P10)

1695 **According to a related aspect of this invention**, as data is updated
1696 and the map contents alter to reflect the new contents, this will preferably
1697 not require the deletion or removal of existing chunks, but instead allow
1698 the existing chunks to remain and the map appended to with an
1699 indication of a new revision existing. Preferably further access to the file
1700 will automatically open the last revision unless requested to open an
1701 earlier revision.

1702 Preferably revisions of any file can be forgotten or deleted (preferably
1703 after checking the file counter or access list of sharers as above). This
1704 will allow users to recover space from no longer required revisions.

Create Map of Maps (Figure 1 - P15)

1705 **According to a related aspect of this invention**, data identifiers,
1706 preferably data maps as mentioned earlier, can be appended to each
1707 other in a way that preferably allows a single file or database record to
1708 identify several files in one map. This is known as a share. Such a share
1709 can be private to the individual, thereby replacing the directory structure
1710 of files that users are normally used to, and replacing this with a new
1711 structure of shares very similar to volumes or filing cabinets as this is
1712 more in line with normal human nature and should make things simpler.

Share Maps (Figure 1 - P16)

1713 **According to a related aspect of this invention**, this map of maps will
 1714 preferably identify the users connected to it via some public ID that is
 1715 known to each other user, with the map itself will being passed to users
 1716 who agree to join the share. This will preferably be via an encrypted
 1717 channel such as ms messenger or similar. This map may then be
 1718 accessed at whatever rank level users have been assigned. Preferably
 1719 there will be access rights such as read / delete / add / edit as is typically
 1720 used today. As a map is altered, the user instigating this is checked
 1721 against the user list in the map to see if this is allowed. If not, the request
 1722 is ignored but preferably the users may then save the data themselves to
 1723 their own database or data maps as a private file or even copy the file to
 1724 a share they have access rights for. These shares will preferably also
 1725 exhibit the revision control mechanism described above.

1726 Preferably joining the share will mean that the users subscribe to a
 1727 shared amount of space and reduce the other subscription, i.e. a 10Gb
 1728 share is created then the individual gives up 10Gb (or equivalent
 1729 dependent on system requirements which may be a multiple or divisor of
 1730 10Gb). Another user joining means they both have a 5Gb space to give
 1731 up and 5 users would mean they all have a 2Gb or equivalent space to
 1732 give up. So with more people sharing, requirements on all users reduce.

Shared Access to Private Files (Figure 1 – PT5 and Figure 18)

- 1733 1. User 1 logs on to network
- 1734 2. Authenticates ID – i.e. gets access to his public and private keys to sign
 1735 messages. This should NOT be stored locally but should have been
 1736 retrieved from a secure location – anonymously and securely.
- 1737 3. User 1 saves a file as normal (encrypted, obfuscated, chunked, and
 1738 stored on the net via a signed and anonymous ID. This ID is a special

WO 2008/065341

PCT/GB2007/004421

68

- 1739 maidsafe.net Share ID (MSID) and is basically a new key pair created
1740 purely for interacting with the share users – to mask the user's MID (i.e.
1741 cannot be tied to MPID via a share). So again the MSID is a key pair and
1742 the ID is the hash of the public key – this public key is stored in a chunk
1743 called the hash and signed and put on the net for others to retrieve and
1744 confirm that the public key belongs to the hash.
- 1745 4. User creates a share – which is a data map with some extra elements to
1746 cover users and privileges.
- 1747 5. File data added to file map is created in the backup process, with one
1748 difference, this is a map of maps and may contain many files – see 14
- 1749 6. User 2 logs in
- 1750 7. User 2 has authentication details (i.e. their private MPID key) and can
1751 sign / decrypt with this MPID public key.
- 1752 8. User 1 sends a share join request to user 2 (shares are invisible on the
1753 net – i.e. nobody except the sharers to know they are there).
- 1754 9. User 1 signs the share request to state he will join share. He creates his
1755 MSID key pair at this time. The signed response includes User 2's MSID
1756 public key.
- 1757 10. Share map is encrypted or sent encrypted (possibly by secure
1758 messenger) to User 1 along with the MSID public keys of any users of the
1759 share that exist. Note the transmission of MSID public key may not be
1760 required as the MSID chunks are saved on the net as described in 3 so
1761 any user can check the public key at any time – this just saves the search
1762 operation on that chunk to speed the process up slightly.

- 1763 11. Each user has details added to the share these include public name
1764 (MPID) and rights (read / write / delete / admin etc.)
- 1765 12. A description of the share file
- 1766 Note that as each user saves new chunks he does so with the MSID
1767 keys. this means that if a shares is deleted or removed the chunks still
1768 exist in the users home database and he can have the option to keep the
1769 data maps and files as individual files or simply forget them all.
- 1770 Note also that as a user opens a file, a lock is transmitted to all other
1771 shares and they will only be allowed to open a file read only – they can
1772 request unlock (i.e. another user unlocks the file – meaning it becomes
1773 read only). Non-logged in users will have a message buffered for them –
1774 if the file is closed the buffered message is deleted (as there is no point
1775 in sending it to the user now) and logged in users are updated also.
- 1776 This will take place using the messenger component of the system to
1777 automatically receive messages from share users about shares (but
1778 being limited to that).

Provide Public ID (Figure 1 - P17)

- 1779 ~~According to a related aspect of this invention~~, a public and Private
1780 key pair is created for a network where preferably the user is
1781 anonymously logged on, and preferably has a changeable pseudo
1782 random private id which is only used for transmission and retrieval of ID
1783 blocks giving access to that network.
- 1784 Preferably this public private key pair will be associated with a public ID.
1785 This ID will be transmittable in a relatively harmless way using almost
1786 any method including in the open (email, ftp, www etc.) but preferably in

1787 an encrypted form. Preferably this ID should be simple enough to
1788 remember such as a phone number type length. Preferably this ID will be
1789 long enough however, to cope with all the world's population and more,
1790 therefore it would be preferably approx 11 characters long.

1791 This ID can be printed on business cards or stationary like a phone
1792 number or email address and cannot be linked to the users private ID by
1793 external sources. However the user's own private information makes this
1794 link by storing the data in the ID bit the user retrieves when logging in to
1795 the network or via another equally valid method of secure network
1796 authentication.

1797 This ID can then be used in data or resource sharing with others in a
1798 more open manner than with the private id. This keeps the private ID
1799 private and allows much improved inter-node or inter-person
1800 communications.

Secure Communications (Figure 1 - P18)

1801 **According to a related aspect of this invention**, the communications
1802 between nodes should be both private and validated. This is preferably
1803 irrefutable but there should be options for refutable communications if
1804 required. For irrefutable communications the user logs on to the network
1805 and retrieves their key pair and-ID. This is then used to start
1806 communications. Preferably the user's system will seek another node to
1807 transmit and receive from randomly – this adds to the masking of the
1808 user's private ID as the private ID is not used in any handshake with
1809 network resources apart from logging in to the network.

1810 As part of the initial handshake between users, a key may be passed.
1811 Preferably this is a code passed between users over another
1812 communications mechanism in a form such as a pin number known only

1813 to the users involved or it may be as simple as appending the user's
1814 name and other info to a communication request packet such as exists in
1815 some instant messaging clients today - i.e. David wants to communicate
1816 with you allow / deny / block.

1817 Unlike many communications systems today, this is carried out on a
1818 distributed server-less network. This however provides the problem of
1819 what to do when users are off line. Today messages are either, stopped
1820 or stored on a server, and in many cases not encrypted or secured. This
1821 invention allows users to have messages securely buffered whilst off line.
1822 This is preferably achieved by the node creating a unique identifier for
1823 only this session and passing that ID to all known nodes in the user's
1824 address book. Users on-line get this immediately, users off-line have this
1825 buffered to their last known random ID. This ensures that the ability to
1826 snoop on a user's messages is significantly reduced as there is no
1827 identifier to people outside the address book as to the name of the
1828 random ID bit the messages are stored to. The random ID bit is
1829 preferably used as the first part of the identified buffer file name and
1830 when more messages are stored then another file is saved with the
1831 random id and a number appended to it representing the next sequential
1832 available number. Therefore a user will log on and retrieve the messages
1833 sequentially. This allows buffered secured and distributed messaging to
1834 exist.

Document Signing (Figure 1 - P19)

1835 **According to a related aspect of this invention**, a by-product of
1836 securing communications between nodes using asymmetric encryption is
1837 as previously stated, introducing a non-refutable link. This allows for not
1838 only messages between nodes to be non-refutable but also for
1839 documents signed in the same manner as messages to be non refutable.
1840 Today somebody can easily steal a user's password or purposely attack

1841 users as they are not anonymous; this invention provides a great deal of
1842 anonymity and backs this up with access to resources.

1843 Documents may be signed and passed as legally enforceable between
1844 parties as a contract in many countries.

Contract Conversations (Figure 1 – P20)

1845 ***According to a related aspect of this invention,*** a conversation or
1846 topic can be requested under various contracted conditions. The system
1847 may have a non disclosure agreement as an example and both parties
1848 digitally sign this agreement automatically on acceptance of a contract
1849 conversation. In this case a non disclosure conversation. This will
1850 preferably speed up and protect commercial entities entering into
1851 agreements or where merely investigating a relationship. Preferably other
1852 conditions can be applied here such as preferably full disclosure
1853 conversations, Purchase order conversations, contract signing
1854 conversations etc. This is all carried out via a system preferably having
1855 ready made enforceable contracts for automatic signing. These contracts
1856 may preferably be country or legal domain specific and will require to be
1857 enforceable under the law of the countries where the conversation is
1858 happening. This will require the users to preferably automatically use a
1859 combination of geographic IP status and by selecting which is their home
1860 country and where are they are at that time located and having that
1861 conversation.

1862 Preferably only the discussion thread is under this contract, allowing any
1863 party to halt the contract but not the contents of the thread which is under
1864 contract.

1865 Preferably there can also be a very clear intent statement for a
1866 conversation that both parties agree to. This statement will form the basis

1867 of a contract in the event of any debate. The clearer the intent statement
 1868 is; the better for enforceability. These conversations are potentially not
 1869 enforceable but should lead to simplifying any resolution required at a
 1870 later date. Preferably this can be added together with an actual contract
 1871 conversation such as a non disclosure agreement to form a pack of
 1872 contracts per conversation. Contract conversations will be clearly
 1873 identified as such with copies of the contracts easily viewable by both
 1874 parties at any time, these contracts will preferably be data maps and be
 1875 very small in terms of storage space required.

ms_messenger (Figure 1 – PT6 and Figure 19)

- 1876 1. A non public ID preferably one which is used in some other autonomous
 1877 system is used as a sign in mechanism and creates a Public ID key pair.
- 1878 2. The user selects or creates their public ID by entering a name that can
 1879 easily be remembered (such as a nickname) the network is checked for a
 1880 data element existing with a hash of this and if not there, this name is
 1881 allowed. Otherwise the user is asked to choose again.
- 1882 3. This ID called the MPID (maidsafe.net public ID) can be passed freely
 1883 between friends or printed on business cards etc. as an email address is
 1884 today.
- 1885 4. To initiate communications a user enters the nickname of the person he
 1886 is trying to communicate with along with perhaps a short statement (like a
 1887 prearranged pin or other challenge). The receiver agrees or otherwise to
 1888 this request, disagreeing means a negative score starts to build with
 1889 initiator. This score may last for hours, days or even months depending
 1890 on regularity of refusals. A high score will accompany any communication
 1891 request messages. Users may set a limit on how many refusals a user
 1892 has prior to being automatically ignored.

- 1893 5. All messages now transmitted are done so encrypted with the receiving
1894 party's public key, making messages less refutable.
- 1895 6. These messages may go through a proxy system or additional nodes to
1896 mask the location of each user.
- 1897 7. This system also allows document signing (digital signatures) and
1898 interestingly, contract conversations. This is where a contract is signed
1899 and shared between the users. Preferably this signed contract is equally
1900 available to all in a signed (non changeable manner) and retrievable by
1901 all. Therefore a distributed environment suits this method. These
1902 contracts may be NDAs Tenders, Purchase Orders etc.
- 1903 8. This may in some cases require individuals to prove their identity and this
1904 can take many forms from dealing with drivers licenses to utility bills
1905 being signed off in person or by other electronic methods such as
1906 inputting passport numbers, driving license numbers etc.
- 1907 9. If the recipient is on line then messages are sent straight to them for
1908 decoding.
- 1909 10. If the recipient is not on line, messages are require to be buffered as
1910 required with email today.
- 1911 11. Unlike today's email though, this is a distributed system with no servers to
1912 buffer to. In maidsafe.net messages are stored on the net encrypted with
1913 the receiver's public key. Buffer nodes may be known trusted nodes or
1914 not.
- 1915 12. Messages will look like receivers id.message 1.message 2 or simply
1916 be appended to the users MPID chunk, in both cases messages are
1917 signed by the sender. This allows messages to be buffered in cases

WO 2008/065341

PCT/GB2007/004421

75

1918 where the user is offline. When the user comes on line he will check his
 1919 ID chunk and look for appended messages as above ID.message1 etc.
 1920 which is MPID.<message 1 data>.<message 2 data> etc.

1921 This system allows the ability for automatic system messages to be sent,
 1922 i.e... in the case of sharing the share, data maps can exist on everyone's
 1923 database and never be transmitted or stored in the open. File locks and
 1924 changes to the maps can automatically be routed between users using
 1925 the messenger system as described above. This is due to the distributed
 1926 nature of maidsafe.net and is a great, positive differentiator from other
 1927 messenger systems. These system commands will be strictly limited for
 1928 security reasons and will initially be used to send alerts from trusted
 1929 nodes and updates to share information by other shares of a private file
 1930 share (whether they are speaking with them or not).

1931 The best way within our current power to get rid of email spam is to get
 1932 rid of email servers.

Anonymous Transactions (Figure 1 - P24)

1933 **According to a related aspect of this invention**, the ability to transact
 1934 in a global digital medium is made available with this invention. This is
 1935 achieved by passing signed credits to sellers in return for goods. The
 1936 ~~credits are data chunks with a given worth preferably 1, 5, 10, 20, 50,~~
 1937 100 etc. units (called cybers in this case). These cybers are a digital
 1938 representation of a monetary value and can be purchased as described
 1939 below or earned for giving up machine resources such as disk space of
 1940 cpu time etc. There should be preferably many ways to earn cybers.

1941 A cyber is actually a digitally signed piece of data containing the value
 1942 statement i.e. 10 cybers and preferably a serial number. During a
 1943 transaction the seller's serial number database is checked for validity of

WO 2008/065341

PCT/GB2007/004421

76

1944 the cyber alone. The record of the ID used to transact is preferably not
 1945 transmitted or recorded. This cyber will have been signed by the issuing
 1946 authority as having a value. This value will have been proven and
 1947 preferably initially will actually equate to a single currency for instance
 1948 linked to a Euro. This will preferably alter through time as the system
 1949 increases in capability.

1950 Some sellers may request non anonymous transactions and if the user
 1951 agrees he will then use the public ID creation process to authenticate the
 1952 transaction and may have to supply more data. However there may be
 1953 other sellers who will sell anonymously. This has a dramatic effect on
 1954 marketing and demographic analysis etc. as some goods will sell
 1955 anywhere and some will not. It is assumed this system allows privacy
 1956 and freedom to purchase goods without being analysed.

1957 The process of transacting the cybers will preferably involve a signing
 1958 system such that two people in a transaction will actually pass the cyber
 1959 from the buyer to the seller. This process will preferably alter the
 1960 signature on the cyber to the seller's signature. This new signature is
 1961 reported back to the issuing authority.

Interface with Non-Anonymous Systems (Figure 1 - P23)

1962 ~~According to a related aspect of this invention,~~ people may purchase
 1963 digital cash or credits from any seller of the cash. The seller will
 1964 preferably create actual cash data chunks which are signed and
 1965 serialised to prevent forgery. This is preferably accountable as with
 1966 today's actual cash to prevent fraud and counterfeiting. Sellers will
 1967 preferably be registered centrally in some cases. The users can then
 1968 purchase cybers for cash and store these in their database of files in a
 1969 system preferably such as maidsafe.net.

WO 2008/065341

PCT/GB2007/004421

77

- 1970 As a cyber is purchased it is preferably unusable and in fact simply a
1971 reference number used to claim the cyber's monetary value by the
1972 purchaser's system. This reference number will preferably be valid for a
1973 period of time. The purchaser then logs in to their system such as
1974 maidsafe.net and inputs the reference number in a secure
1975 communications medium as a cyber request. This request is analysed by
1976 the issuing authority and the transaction process begins. Preferably the
1977 cyber is signed by the issuing authority that then preferably encrypts it
1978 with the purchaser's public key and issues a signing request. The cyber
1979 is not valid at this point. Only when a signed copy of the cyber is received
1980 by the issuing authority is the serial number made valid and the cyber is
1981 live.
- 1982 This cyber now belongs to the purchaser and validated by the issuer. To
1983 carry out a transaction this process is preferably carried out again i.e. the
1984 seller asks for payment and a cyber signed by the buyer is presented –
1985 this is validated by checking with the issuer that the serial code is valid
1986 and that the buyer is the actual owner of the cyber. Preferably the buyer
1987 issues a digitally signed transaction record to the issuing authority to
1988 state he is about to alter that cyber's owner. This is then passed to the
1989 seller who is requested to sign it. The seller then signs the cyber and
1990 requests the issuing authority to accept him as new owner via a signed
1991 request. The authority then simply updates the current owner of the cyber
1992 in their records.
- 1993 These transactions are preferably anonymous, as users should be using
1994 a private id to accomplish this process. This private ID can be altered at
1995 any time but the old id should be saved to allow cyber transactions to
1996 take place with the old id.

Anonymity (Figure 1 - P25)

- 1997 ***According to a related aspect of this invention***, a system of voting
 1998 which is non refutable and also anonymous is to be considered. This is a
 1999 requirement to allow free speech and thinking to take place on a global
 2000 scale without recrimination and negative feedback as is often the case.
- 2001 To partake in a vote the user will have to be authenticated as above then
 2002 preferably be presented with the issue to be voted on. The user will then
 2003 use a private ID key to sign their vote anonymously. Preferably non
 2004 anonymous irrefutable voting may also take place in the system by
 2005 simply switching from a private ID to a public one. This will preferably
 2006 form the basis of a petition based system as an add-on to the voting
 2007 system.
- 2008 The system will require that a block of data can be published (preferably
 2009 broadcast to each user via messenger) and picked up by each user of
 2010 the system and presented as a poll. This poll will then be signed by the
 2011 user and sent back to the poll issuer whose system will count the votes
 2012 and preferably show a constant indication of the votes to date.
- 2013 As there are public and private IDs available, then each vote will require
 2014 preferably only ONE ID to be used to prevent double voting. Preferably
 2015 geographic IP may be used to establish geographic analysis of the voting
 2016 community, particularly on local issues.

Voting System (Figure 1 – PT8 and Figure 20)

- 2017 1. A vote is created in a normal fashion; it could be a list of candidates or a
 2018 list of choices that users have to select. Preferably this list will always
 2019 have an "I do not have enough information" option appended to the
 2020 bottom of the list -- to ensure voters have sufficient knowledge to make a

WO 2008/065341

PCT/GB2007/004421

79

- 2021 decision. A limit on the last option should be stipulated as a limit to void
2022 the vote and redo with more information.
- 2023 2. This vote is stored on the system with the ID of the voting authority. This
2024 may be a chunk of data called with a specific name and digitally signed
2025 for authenticity. All storage nodes may be allowed to ensure certain
2026 authorities are allowed to store votes, and only store votes digitally
2027 signed with the correct ID.
- 2028 3. A system broadcast may be used to let everyone interested know that
2029 there is a new vote to be retrieved. This is an optional step to reduce
2030 network congestion with constant checking for votes; other similar
2031 systems may be used for the same ends.
- 2032 4. A non anonymous user logged into the net will pick up the vote. This is a
2033 user with a public ID known at least to the authority. The vote may in fact
2034 be a shared chunk that only certain IDs have access to or know of its
2035 location (i.e. split onto several component parts and a messaging system
2036 used to alert when votes are ready.)
- 2037 5. An anonymous user may be logged onto the net and may in fact use a
2038 random ID to pick up the vote.
- 2039 6. The vote is retrieved.
- 2040 7. The system will send back a signed (with the ID used to pick up the vote)
2041 "I accept the vote".
- 2042 8. The voting authority will transmit a ballot paper – i.e. a digitally signed
2043 (and perhaps encrypted / chunked) ballot paper. This may be a digitally
2044 signed "authorisation to vote" slip which may or may not be sequentially
2045 numbered or perhaps a batch of x number of the same serial numbers (to

WO 2008/065341

PCT/GB2007/004421

80

- 2046 prevent fraud by multiple voting from one source - i.e. issue 5 same
2047 numbers randomly and only accept 5 votes with that number).
- 2048 9. User machine decrypts this ballot paper.
- 2049 10. The users system creates a one time ID + key pair to vote. This public
2050 key can be hashed and stored on the net as with a MAID or PMID so as
2051 to allow checking of any signed or encrypted votes sent back.
- 2052 11. The vote is sent back to the authority signed and preferably encrypted
2053 with the authority's public key.
- 2054 12. In the case of anonymous or non anonymous voting this may be further
2055 masqueraded by passing the vote through proxy machines en route.
- 2056 13. The vote is received and a receipt chunk put on the net. This is a chunk
2057 called with the user's temp (or voting) ID hash with the last bit shifted or
2058 otherwise knowingly mangled – so as not to collide with the voting ID bit
2059 the user stores for authentication of their public key.
- 2060 14. The authority can then publish a list of who voted for what (i.e. a list of
2061 votes and the voting ID's)
- 2062 15. The user's system checks the list for the ID that was used being present
2063 in the list and validates that the vote was cast properly.
- 2064 If this is not the case.
- 2065 16. The users system issues an alert. This alert may take many forms and
2066 may include signing a vote alert packet; this can be a packed similarly (as
2067 in 13,) altered to be a known form of the vote chunk itself. There are
2068 many forms of raising alerts including simply transmitting an electronic

2069 message through messenger or similar and possibly to a vote
2070 authentication party and not necessarily the voting authority themselves.

2071 17. The user has all the information to show the party investigating voting
2072 authenticity, accuracy, legality or some other aspect, thereby allowing
2073 faults and deliberately introduced issues to be tracked down.

2074 18. The user has the option to remove all traces of the vote from his system
2075 at this time.

Proven Individual (Figure 1 - P26)

2076 ***According to a related aspect of this invention***, using a system of
2077 anonymous authentication preferably as in maidsafe.net, the first stage is
2078 partially complete and individual accounts are authentic but this does not
2079 answer the question of anonymous individuals, this is described here.

2080 Access to a system can be made with information that we possess
2081 (passwords etc.) or something that we physically have (iris/ fingerprint or
2082 other biometric test). To prove an individual's identity the system will
2083 preferably use a biometric test. This is a key to the voting system as it
2084 becomes more broadly adopted. It is inherent in this system that any
2085 personally identifying data must be kept secret, and also that any
2086 passwords or access control information is never transmitted.

2087 When a user authenticates, the system can recognise if they have done
2088 so biometrically. In this case, the account is regarded as a unique
2089 individual rather than an individual account. This is possible as
2090 maidsafe.net can authenticate without accessing servers or database
2091 records of a biometric nature for example.

WO 2008/065341

PCT/GB2007/004421

82

2092 As a user logs into maidsafe.net through a biometric mechanism then the
 2093 state of login is known so no login box is presented for typing information
 2094 in to access the system. This allows the system to guarantee that the
 2095 user has logged in biometrically. The system on each machine is always
 2096 validated by maidsafe.net on login to ensure this process cannot be
 2097 compromised.

2098 Preferably some votes will exist only for biometrically authenticated
 2099 users.

Distributed Controlled Voting (Figure 1 - P29)

2100 **According to a related aspect of this invention**, to further manage the
 2101 system there has to be a level of control as well as distribution to enable
 2102 all users to access it at any time. The distribution of the votes is
 2103 controlled as system messages and stored for users using the
 2104 messenger system described earlier.

2105 The main issue with a system such as this would be 'what' is voted on
 2106 and 'who' poses the votes and words polls. This is key to the fairness
 2107 and clarity of the system and process. This voting system will preferably
 2108 always have a 'not enough information' selection to provide a route by
 2109 which users are able to access information so that they are well informed
 2110- before making any decision.

2111 The system will require a group of individuals, who are preferably voted
 2112 into office by the public as the policyholders/ trustees of the voting
 2113 system. This group will be known by their public ID and use their public
 2114 ID to authenticate and publish a poll. This group will preferably be voted
 2115 into office for a term and may be removed at any time via a consensus of
 2116 the voting public. For this reason there will be continual polls on line

2117 which reflect how well the policyholders are doing as a group and
2118 preferably individually as well.

2119 ***According to a related aspect of this invention***, users of the system
2120 will input to the larger issues on the system. Macro management should
2121 be carried out via the policyholders of the system, whom as mentioned
2122 previously may be voted in or out at any time, however larger issues
2123 should be left to the users. These issues can preferably be what licenses
2124 are used, costs of systems, dissemination of charitable contributions,
2125 provision to humanitarian and scientific projects of virtual computing
2126 resources on large scales etc.

2127 To achieve this, preferably a system message will be sent out, where it is
2128 not presented as a message but as a vote. This should show up in the
2129 users voting section of the system. User private IDs will be required to
2130 act on this vote and they can make their decision.

2131 There will be appeals on these votes when it would be apparent that
2132 conclusion of the vote is dangerous to either a small community or the
2133 system as a whole. Users will have an option of continuing with the vote
2134 and potential damage but essentially the user will decide and that will be
2135 final. Preferably this system does not have a block vote or any other
2136 system which rates one individual over another at any time or provides
2137 an advantage in any other way. This requires no ability to allow veto on
2138 any decision or casting of votes by proxy so that the authenticated user's
2139 decision is seen as properly recorded and final.

2140 ***According to a related aspect of this invention***, a system of perpetual
2141 data, self encrypting files and data mapping will allow a global
2142 anonymous backup and restore system for data to exist. This system can
2143 be constructed from the previous discussions where data may be made
2144 perpetual on a network and anonymously shared to prevent duplication.
2145 This together with the ability to check, manipulate and maintain revision

2146 control over files adds the capability of a 'time machine' type environment
2147 where data may be time stamped on backup.

2148 This allows a system to rebuild a user's data set as it was at any time in
2149 history since using maidsafe.net or similar technologies. This may form a
2150 defence at times where in cases like prior art enquiries, insider dealing
2151 etc. is being considered, as the system is secure and validated by many
2152 other nodes etc. It can therefore be shown what knowledge (at least from
2153 the point of view of owning the data pertaining to a subject,) anyone had
2154 of certain circumstances.

2155 ***According to a related aspect of this invention,*** preferably using
2156 aspect(s) previously defined or any that may improve this situation.
2157 Taking distributed authentication, backup and restore along with data
2158 map sharing; the system can add to this the ability for granular access
2159 controls. In this case a node entering the network will request an
2160 authenticator to authorise its access. In this case the authenticator will be
2161 a manager or equivalent in an organisation (whether matrix managed or
2162 traditional pyramid). This authorisation will tie the public ID of the
2163 authoriser to the system as having access to this node's data and any
2164 other authorisations they make (in an authorisation chain).

2165 This allows an environment of distributed secure backup, restore and
2166 sharing in a corporate or otherwise private environment.

2167 ***According to a related aspect of this invention,*** all of the capabilities
2168 described here with the exception of the above will ensure that a network
2169 of nodes can be created, in which users have security privacy and
2170 freedom to operate.

2171 These nodes will have refutable IDs (MAID, PMID etc.) as well as non
2172 refutable IDs (MPID) for different purposes, just as in human life in

WO 2008/065341

PCT/GB2007/004421

85

2173 general there is time to be identified and times when it is just best not to
2174 be.

2175 **According to a related aspect of this invention,** adding the ability of
2176 non refutable messaging allows users to not only communicate genuinely
2177 and securely but also the ability to communicate under contracted terms.
2178 This allows for the implementation of legally kept trade secrets (as
2179 implied with NDA agreements etc.) plus many more contracted
2180 communications. This will hopefully lessen the burden on legal issues
2181 such as litigation etc.

2182 **According to a related aspect of this invention,** adding the ability to
2183 create two voting systems, anonymous and non-anonymous, allows the
2184 system to provide a mechanism for instant democracy. This is achieved
2185 by allowing a voting panel in a user's account that is constantly updated
2186 with issues regarding the system and it's improvements initially. These
2187 votes will be anonymous.

2188 In another anonymous voting scenario users may continually vote on
2189 certain subjects (as in a running poll) these subjects could be the leaders
2190 of boards etc.

2191 In a non anonymous voting scenario it may be there's groups of identified
2192 people (via their MPID) who have a common grouping such as a charity
2193 or similar and they may require certain people to vote on certain matters
2194 and be recognised. This is where the MPID is used for voting.

2195 **According to a related aspect of this invention,** adding to this the
2196 ability to collect and trade credits anonymously allows users to sell
2197 machine resources they are not using, trade on a network with a cash
2198 equivalent and go about there business on a network as they do in real
2199 life.

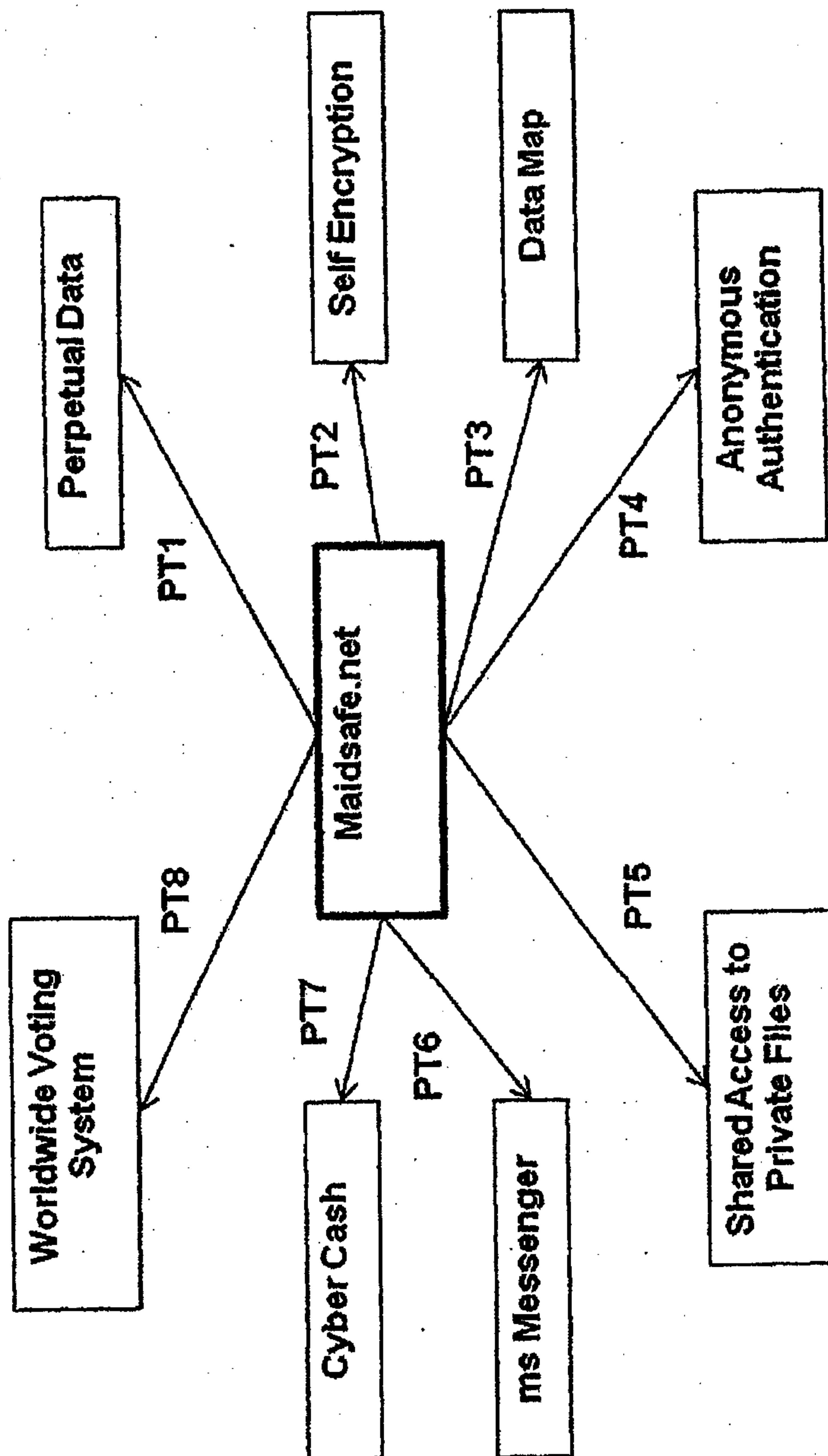
CLAIMS

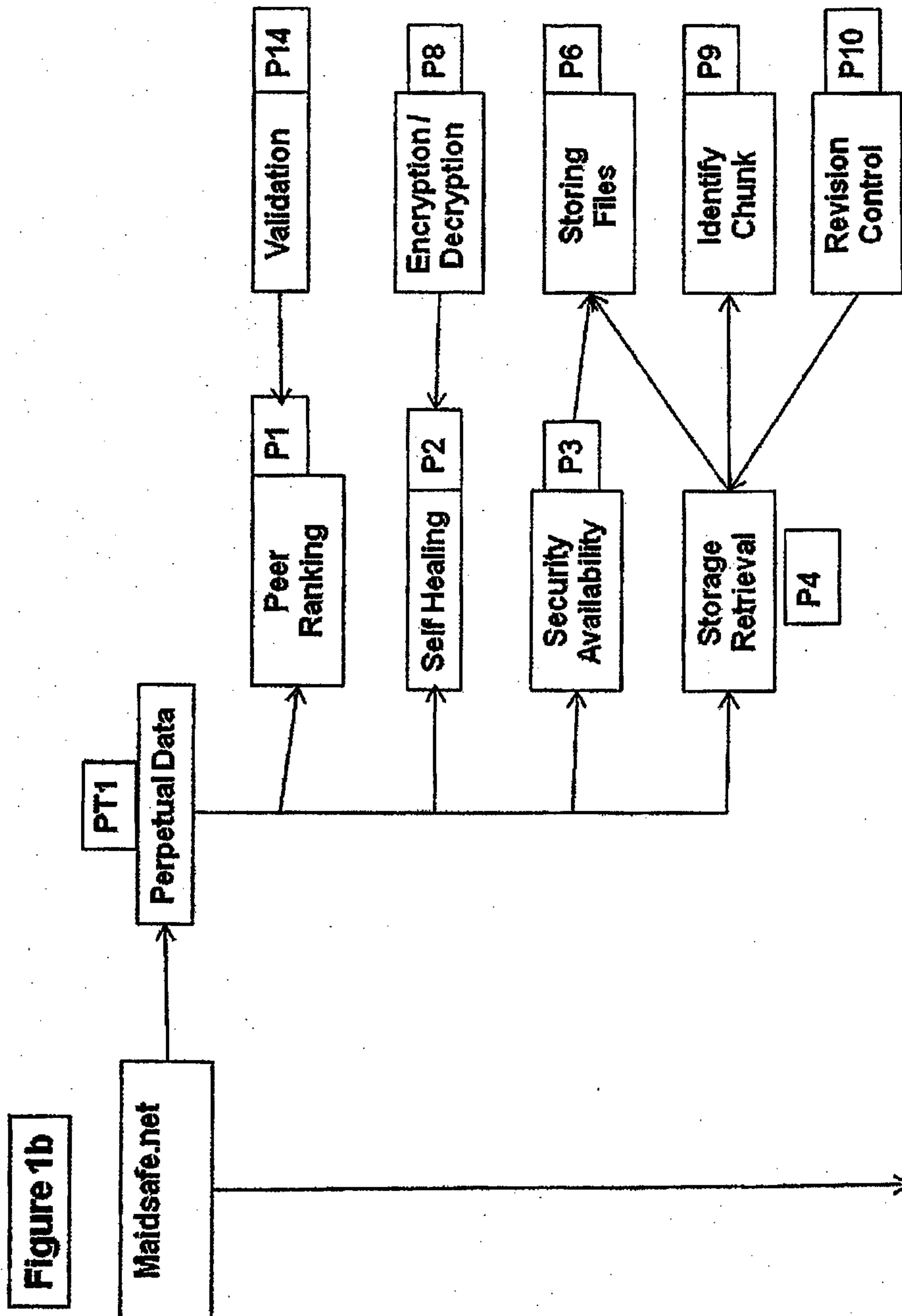
1. A method of protecting data comprising:
obfuscating the data, comprising splitting the data into a plurality of chunks, and
5 swapping data between the chunks; and
encrypting the data, using an encryption algorithm.
2. A method as claimed in Claim 1, wherein known information from the data is used
as the encryption key.
- 10 3. A method as claimed in Claim 2, wherein each chunk is encrypted separately,
and wherein, for each chunk, known information from another chunk is used as the
encryption key.
- 15 4. A method as claimed in any one of claims 1 to 3, including determining a hash
value for the data, and using the determined hash value to determine the chunk size
and/or the number of chunks.
- 20 5. A method as claimed in any any one of claims 1 to 4, wherein the encryption
algorithm is a symmetric encryption algorithm.
6. A method as claimed in any one of claims 1 to 5, wherein swapping data between
the chunks comprises swapping a byte of a first chunk with a byte of a second chunk.
- 25 7. A method as claimed in any one of claims 1 to 6, including determining a hash
value of each chunk and renaming the chunk using the determined hash value of the
chunk.

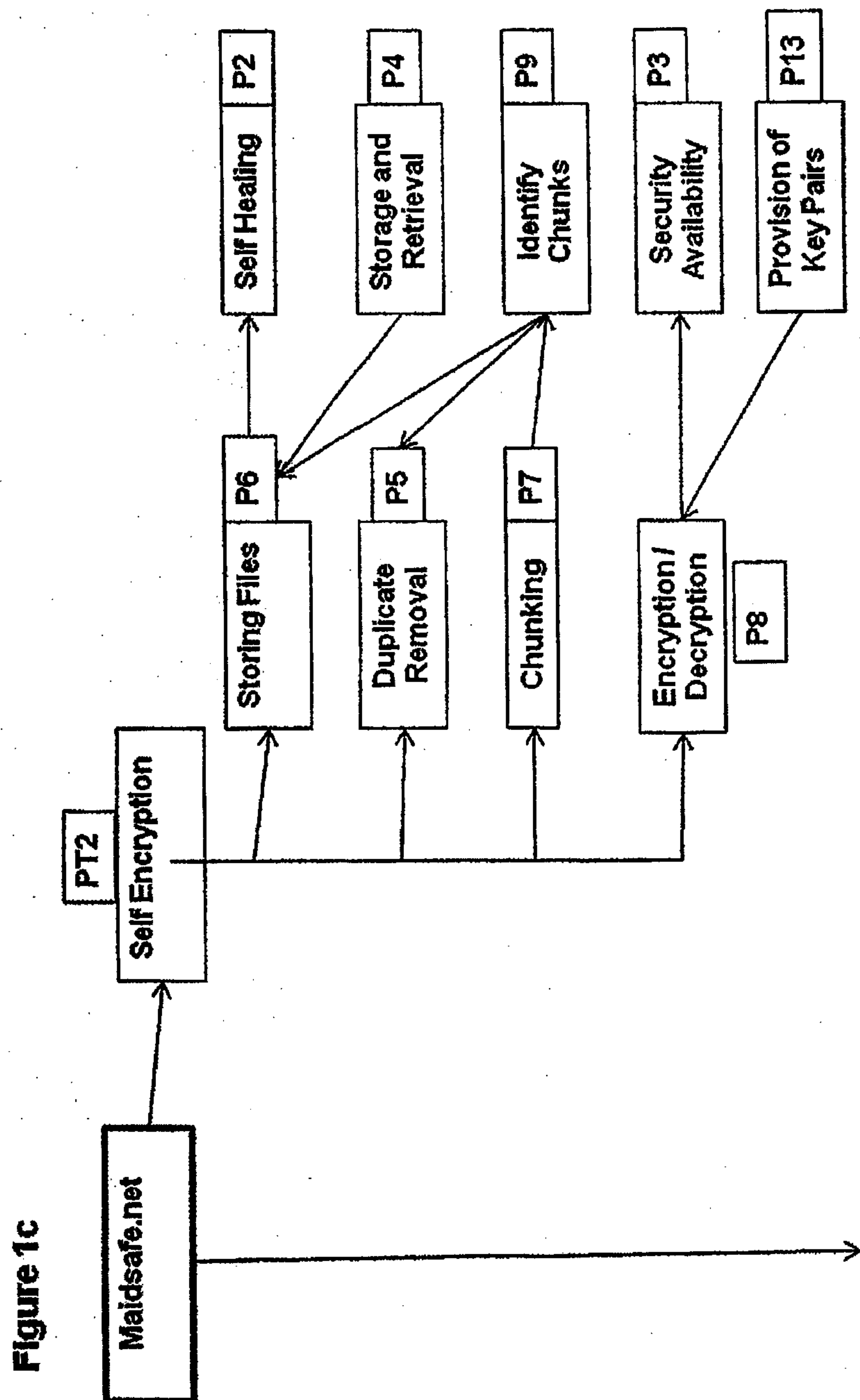
8. A method as claimed in any one of claims 1 to 7, including storing the protected data on a distributed network.
9. A method as claimed in any one of claims 1 to 8, including determining if each
- 5 chunk of the data already exists on the network and, if each chunk of the data already exists, not storing the protected data.

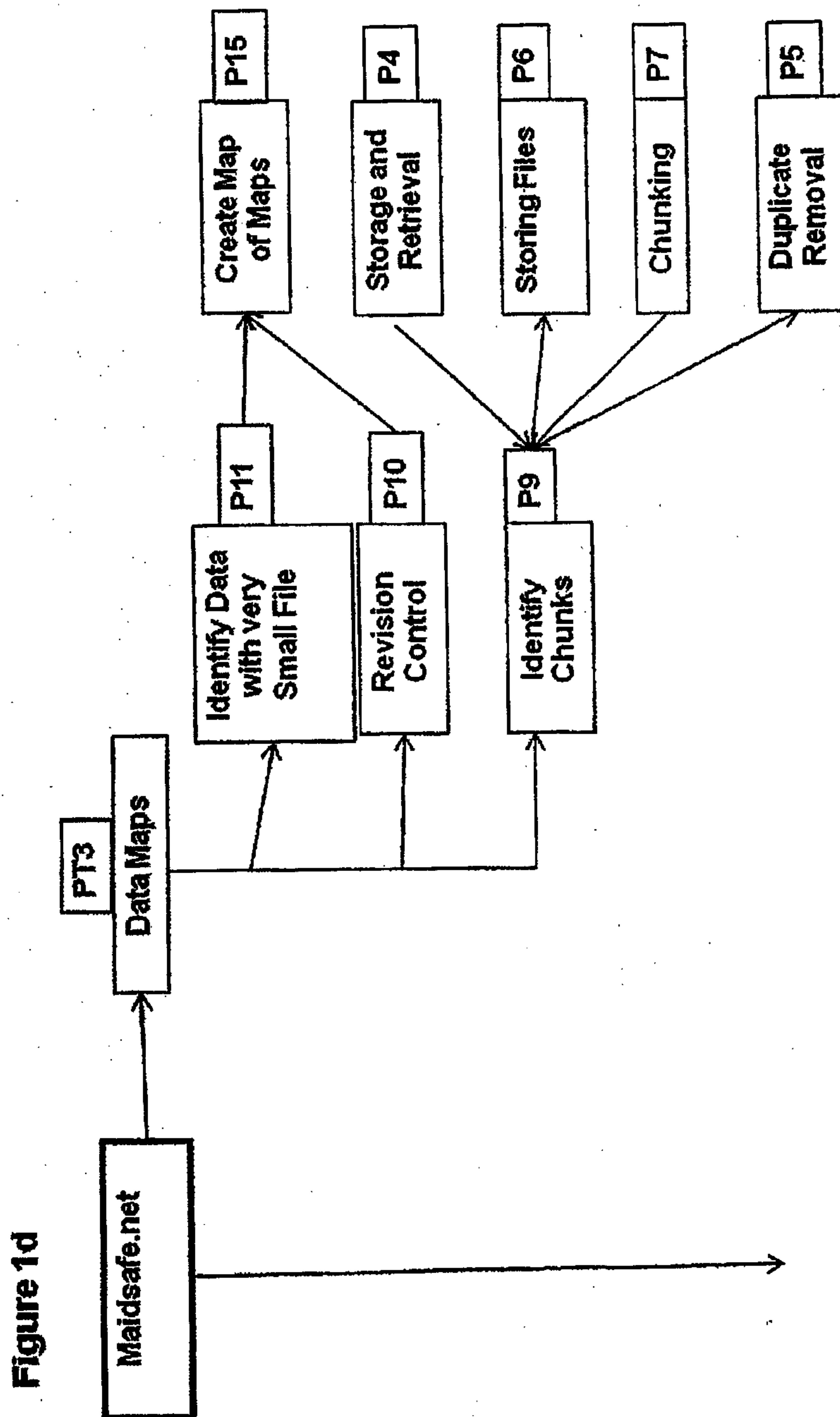
Figure 1 – maidsafe.net associations

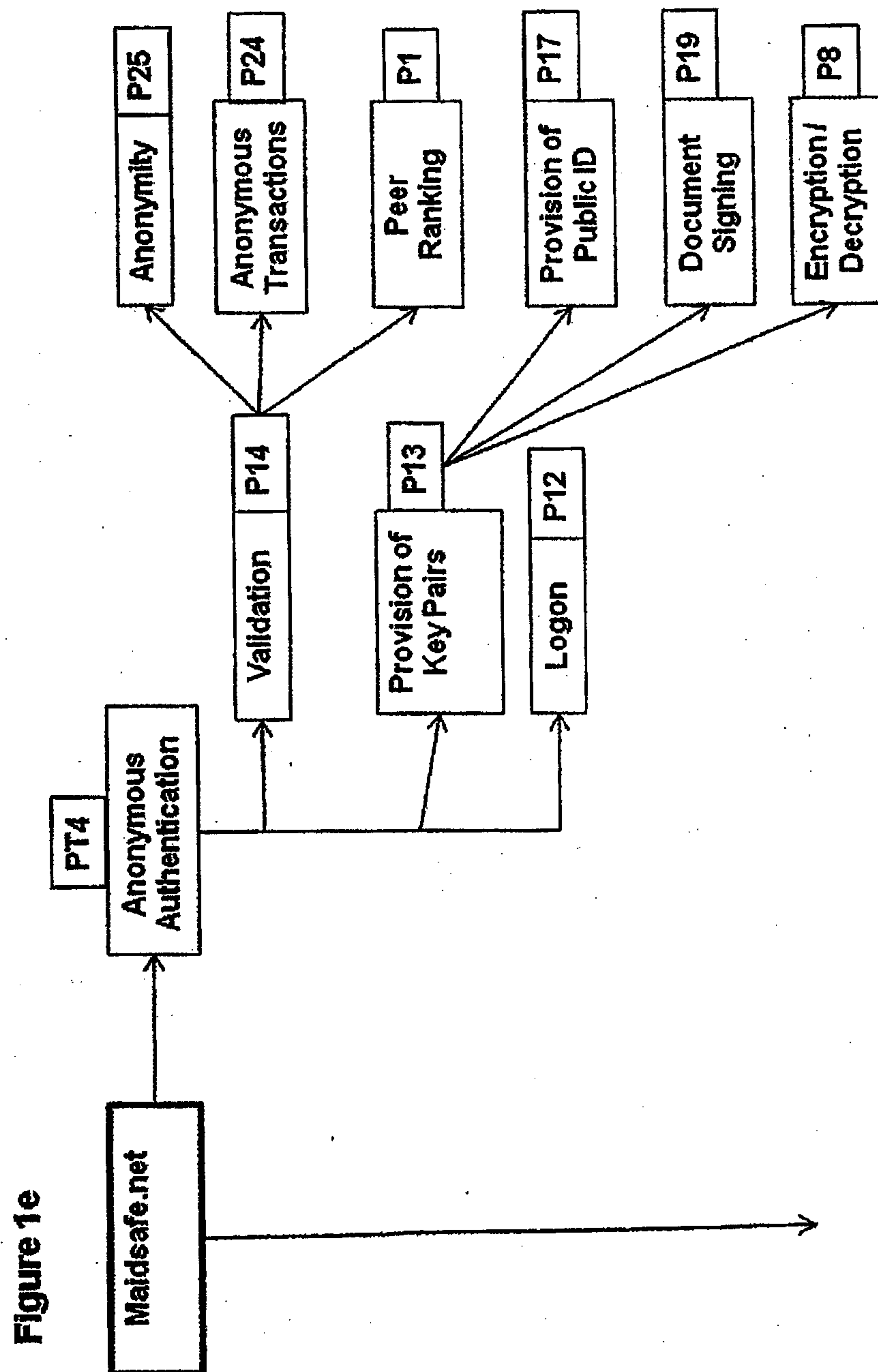
Figure 1a

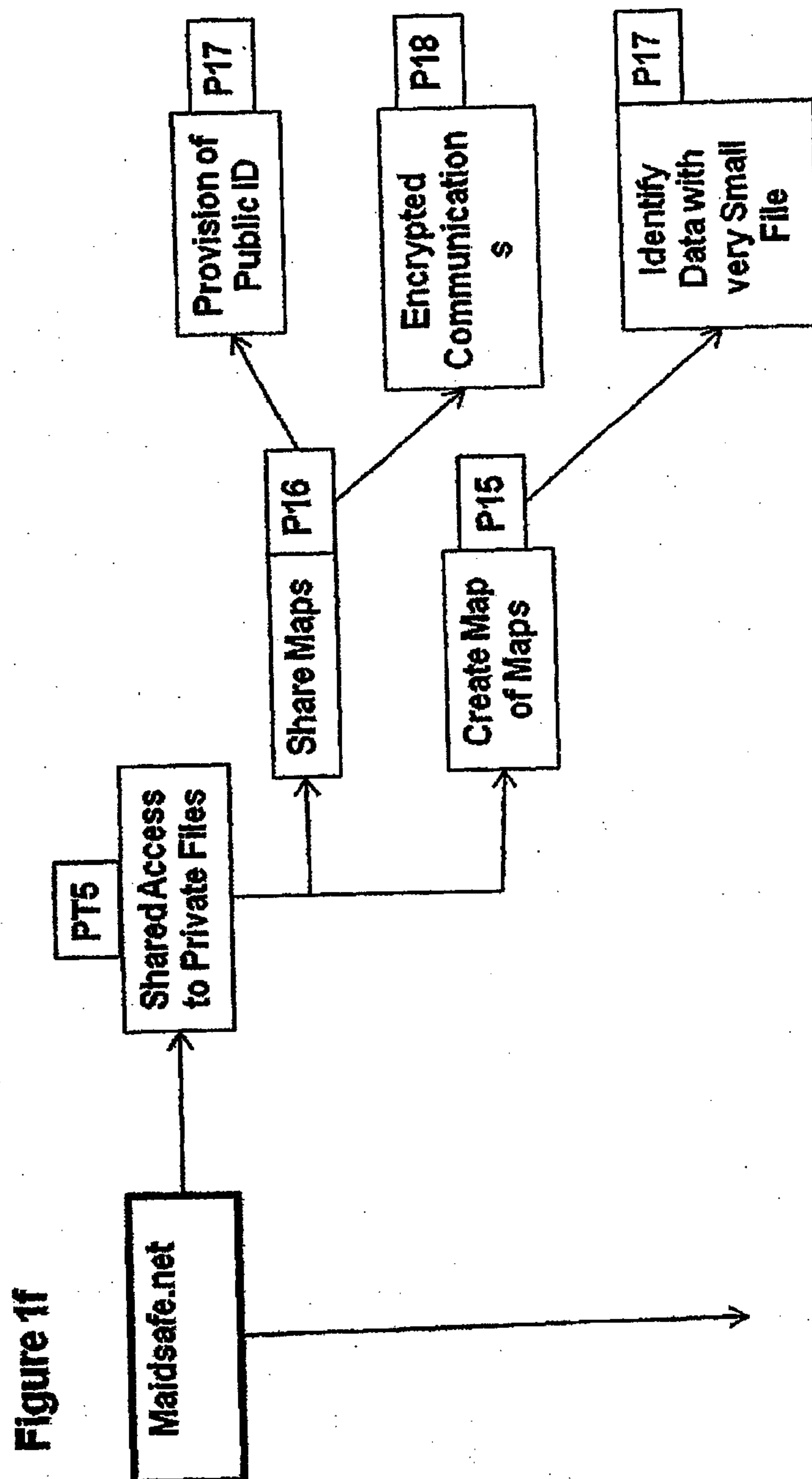


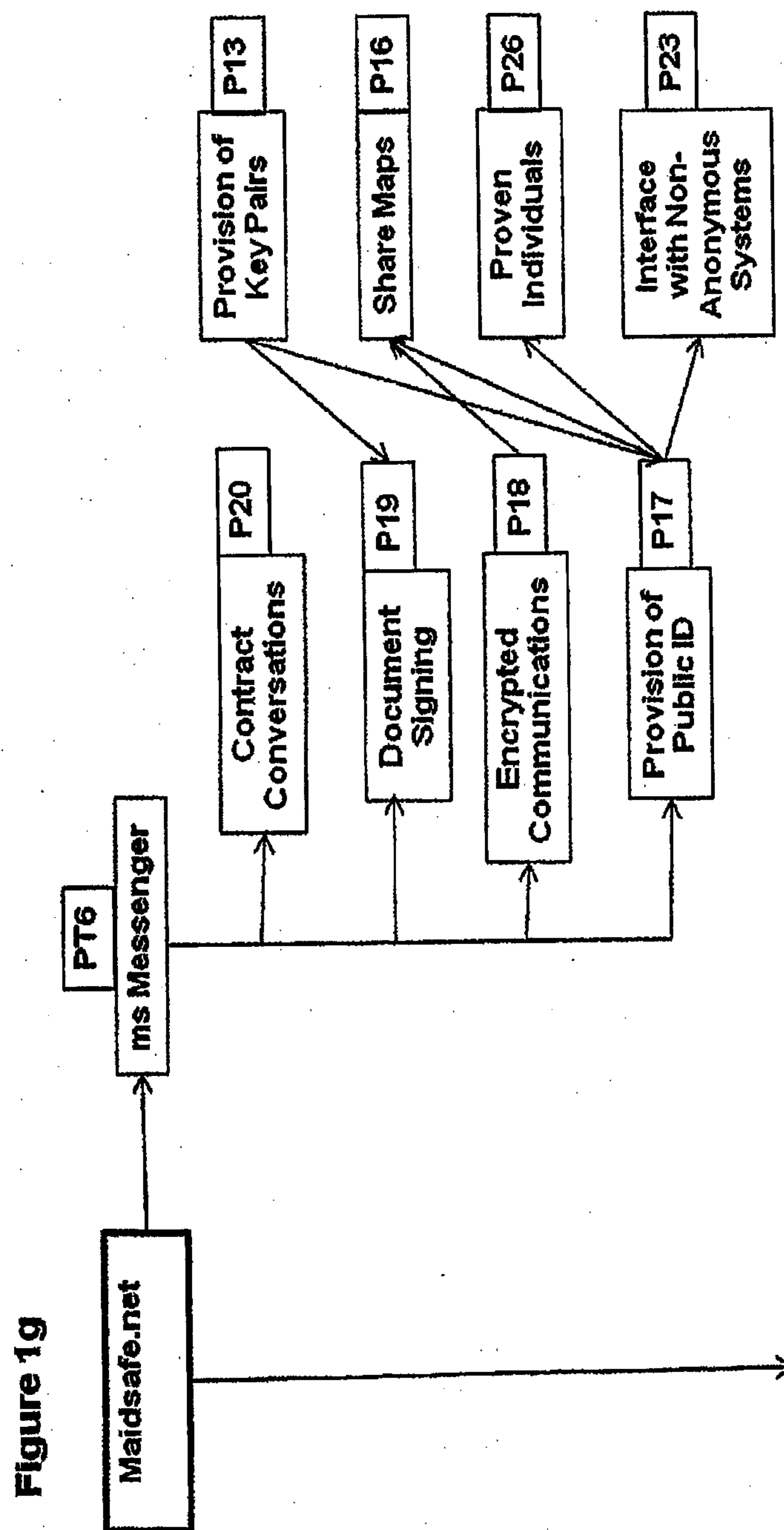


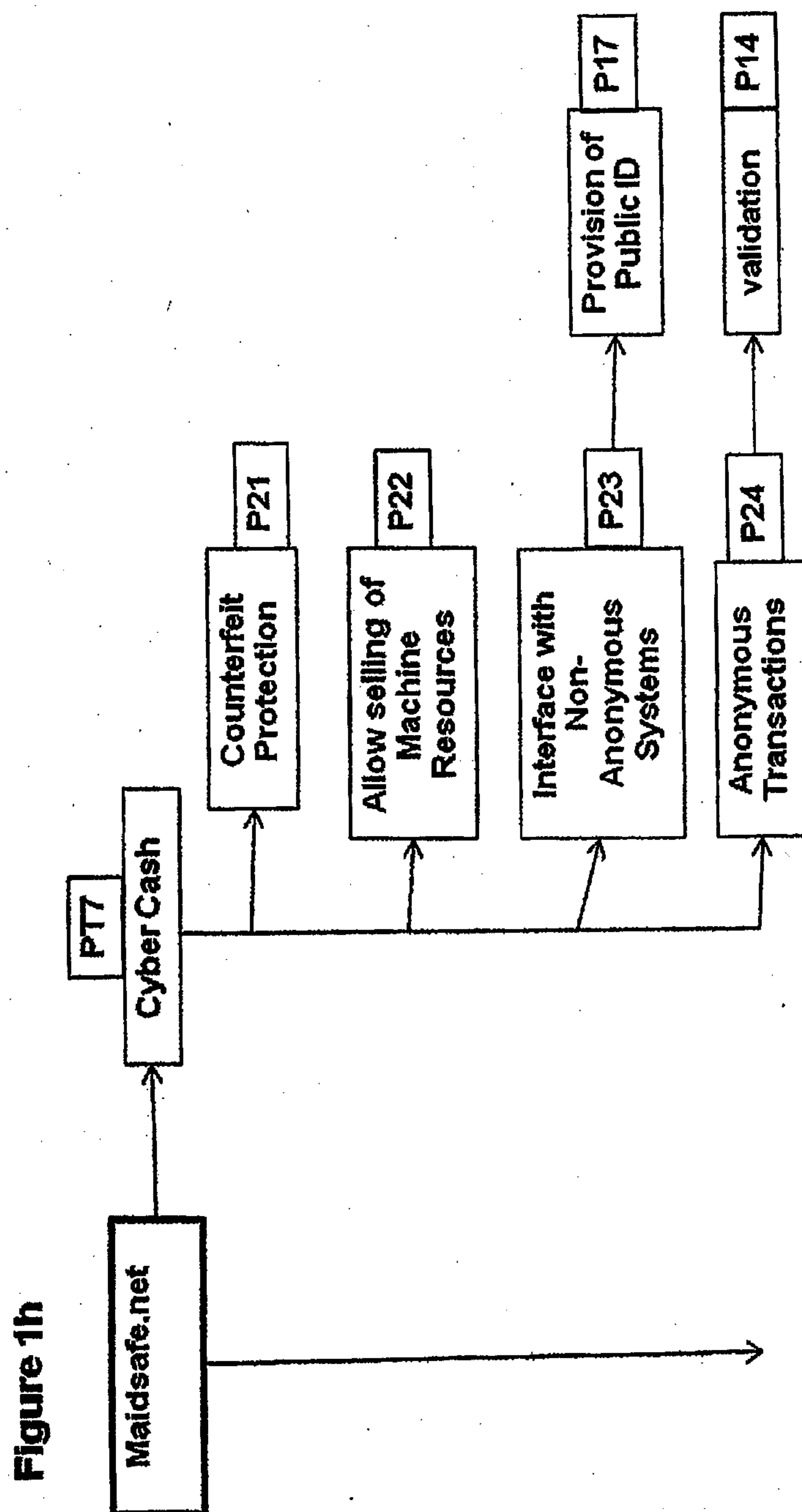












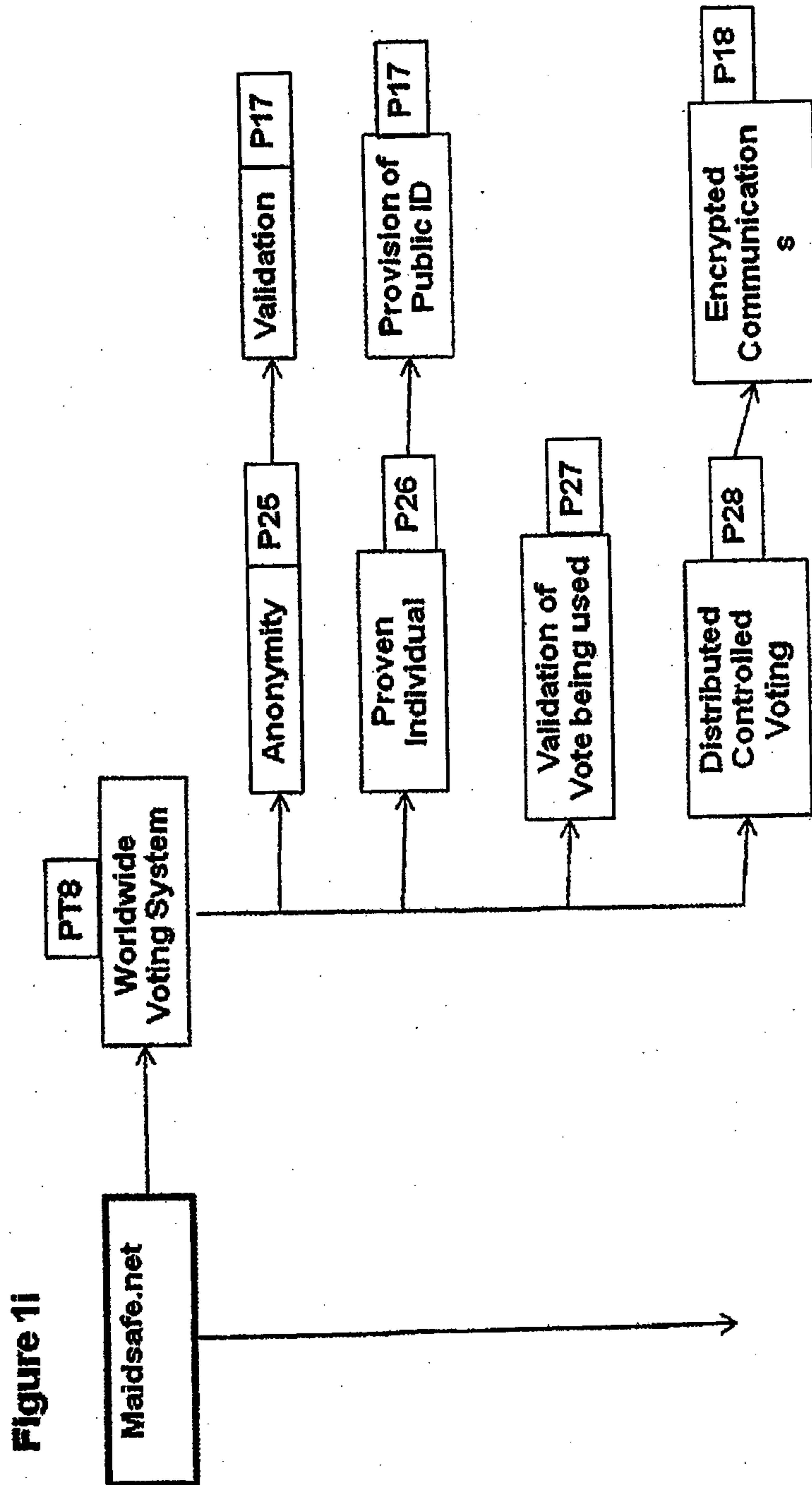


Figure 2 – Self Authentication Detail

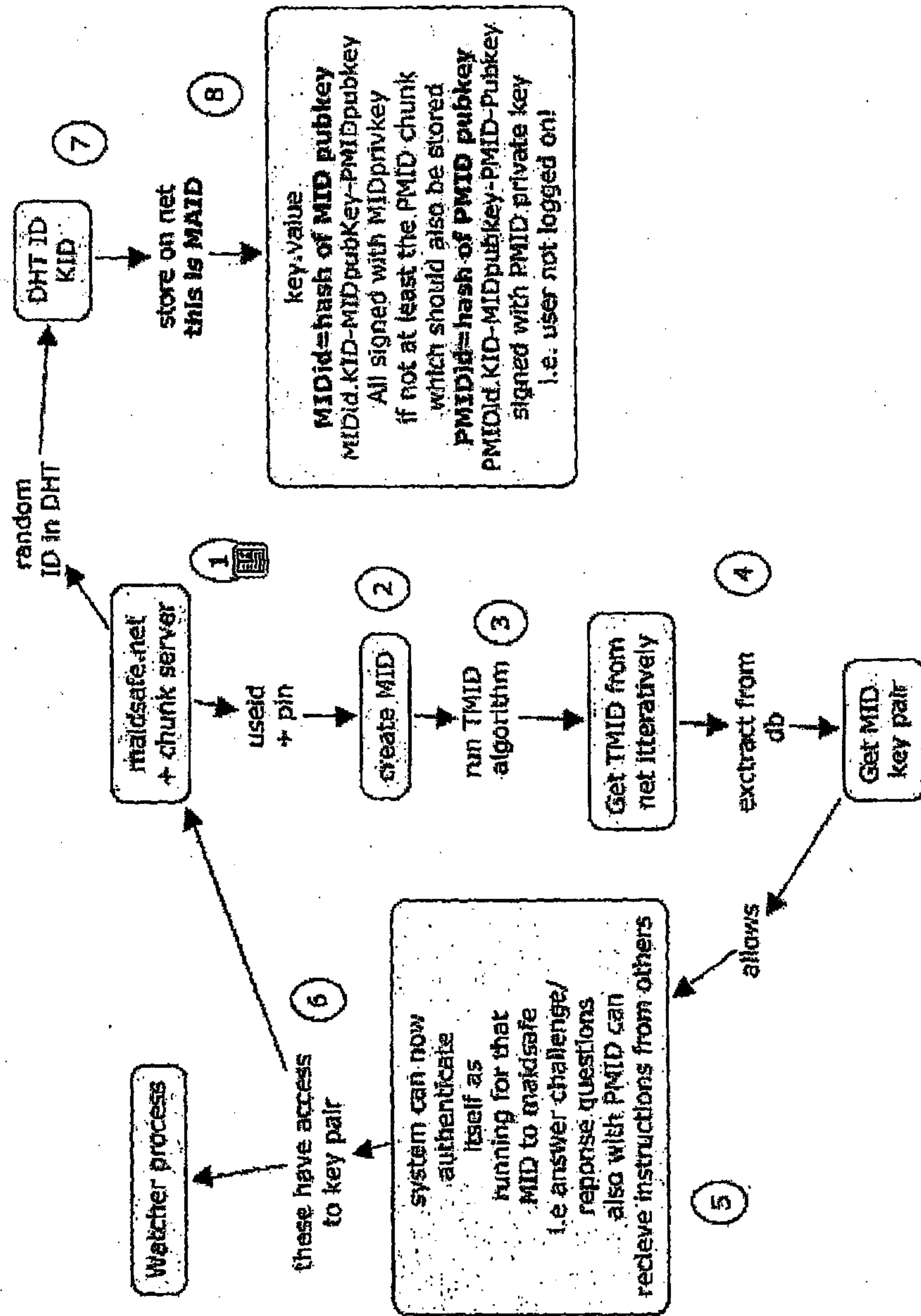


Figure 3 – Peer to Peer Schematic

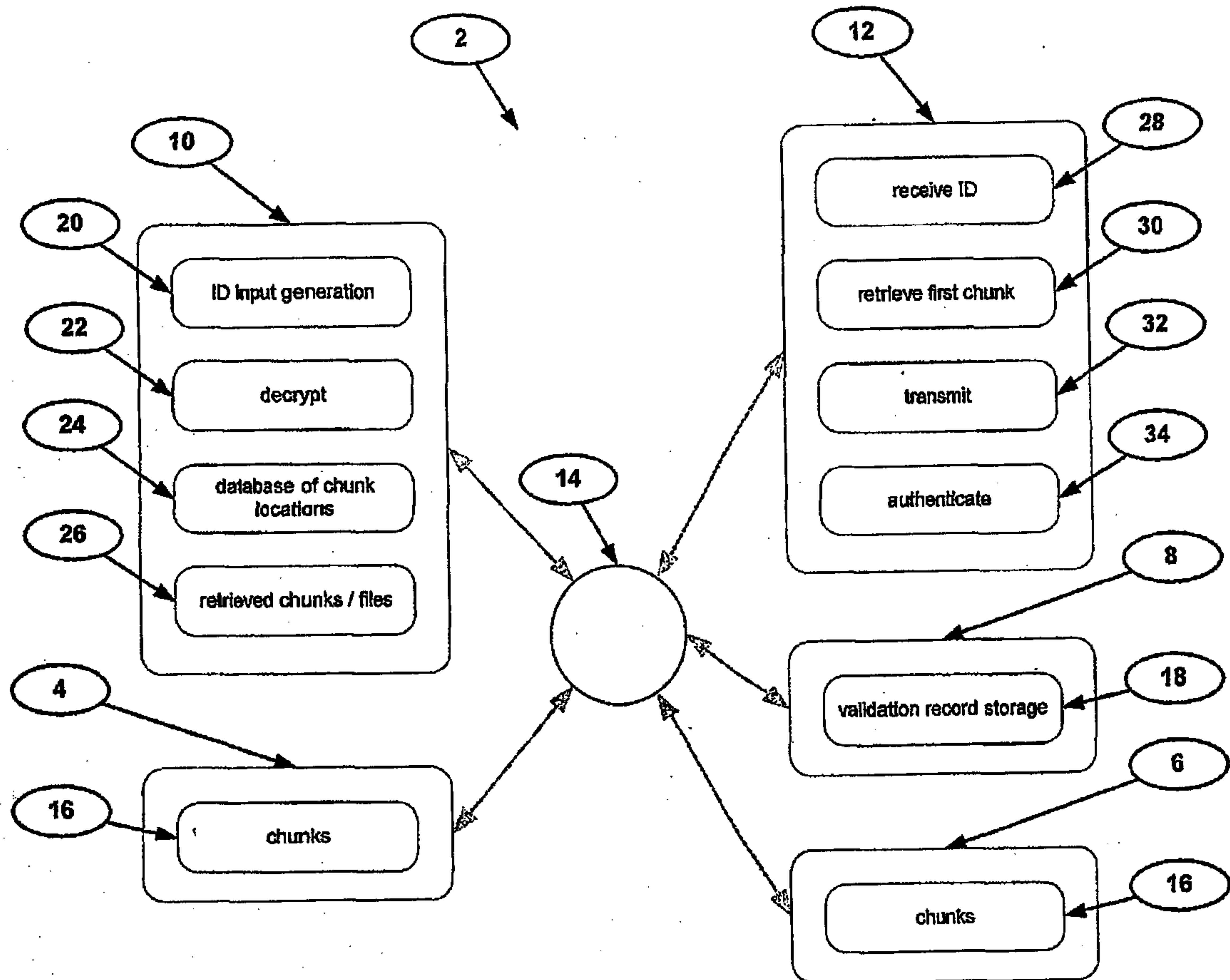


Figure 4 – Authentication Flowchart

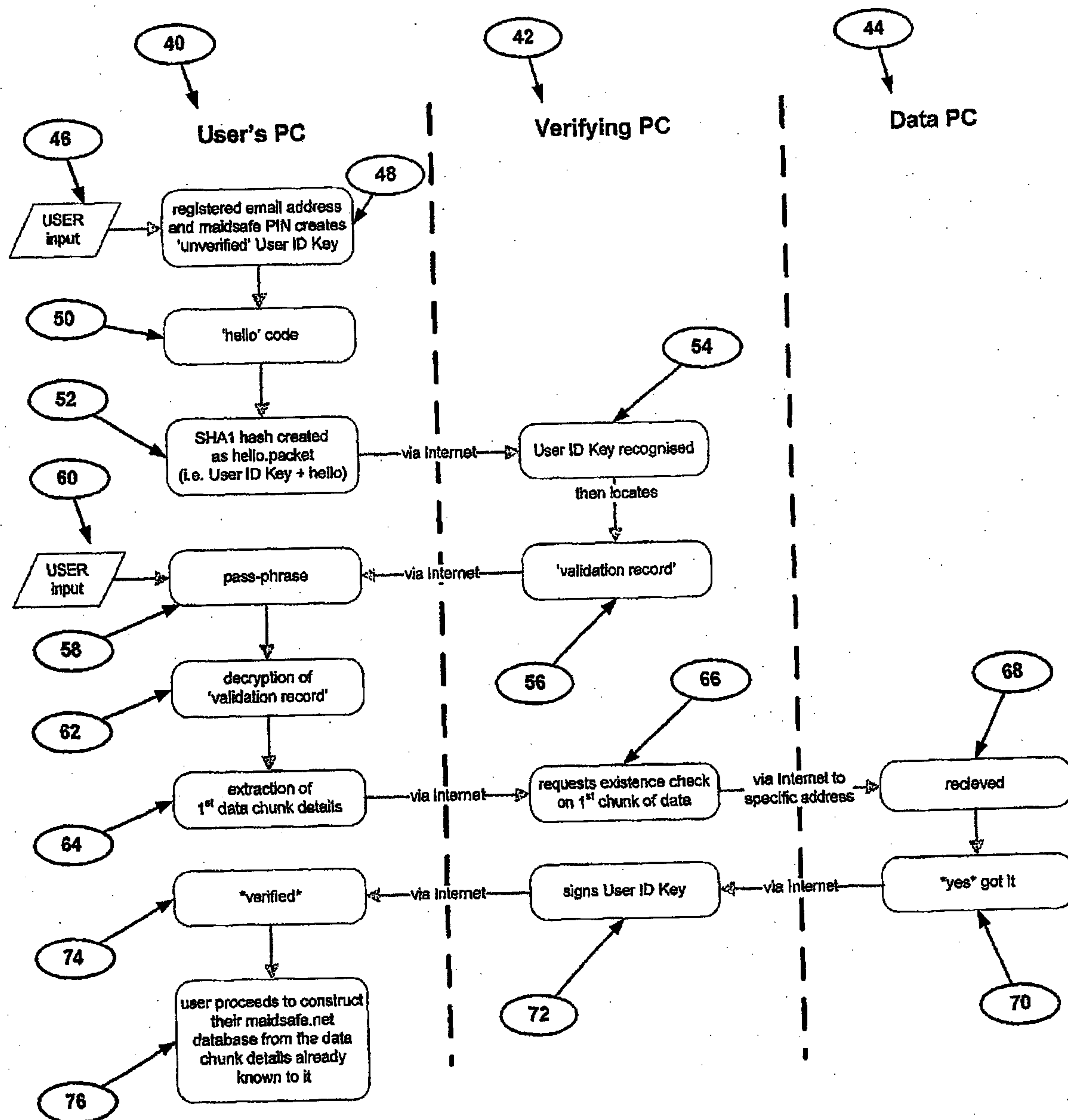


Figure 5 - Data Assurance Event Sequence

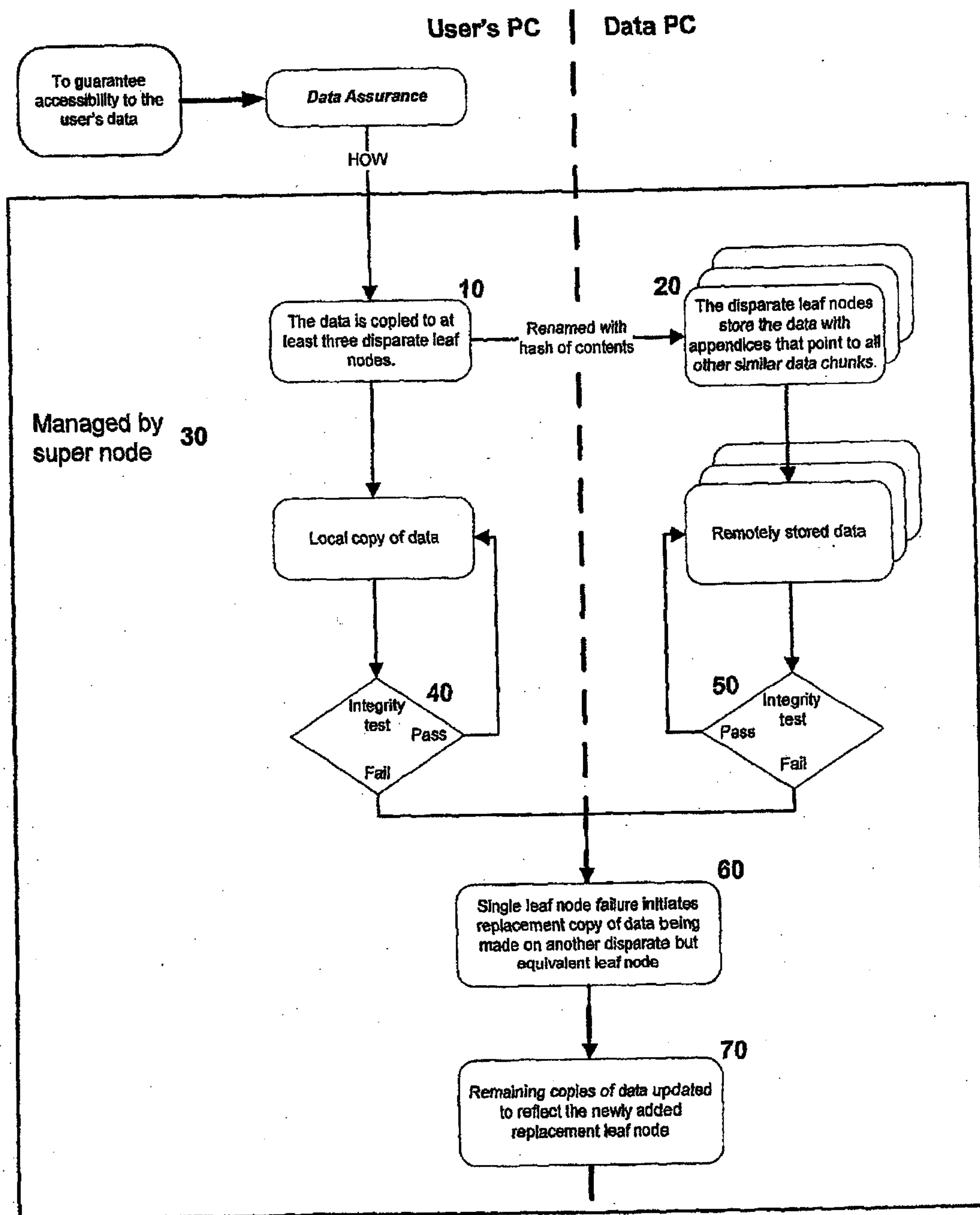


Figure 6 – Chunking Event Sequence

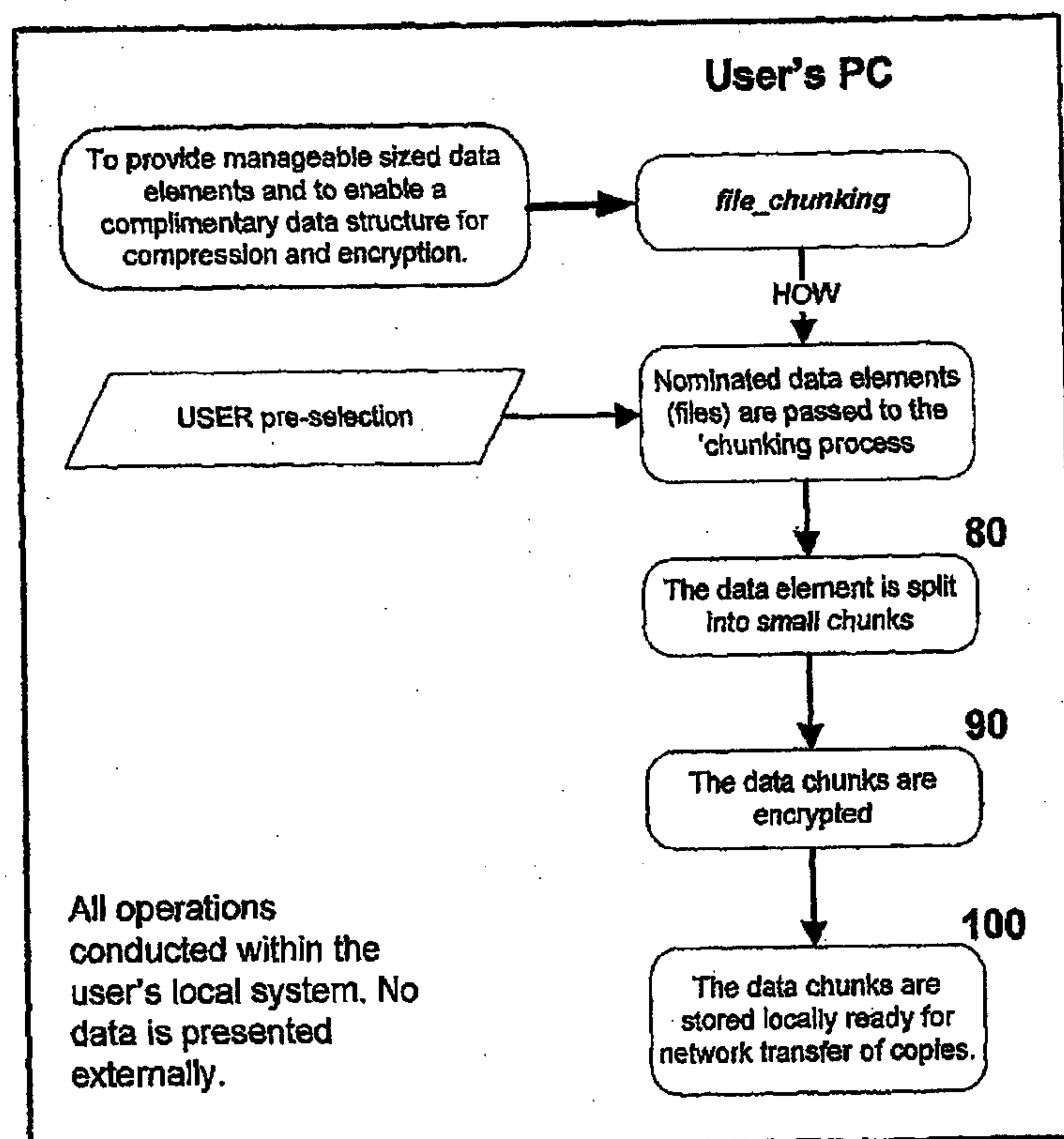


Figure 7 – Chunking Example

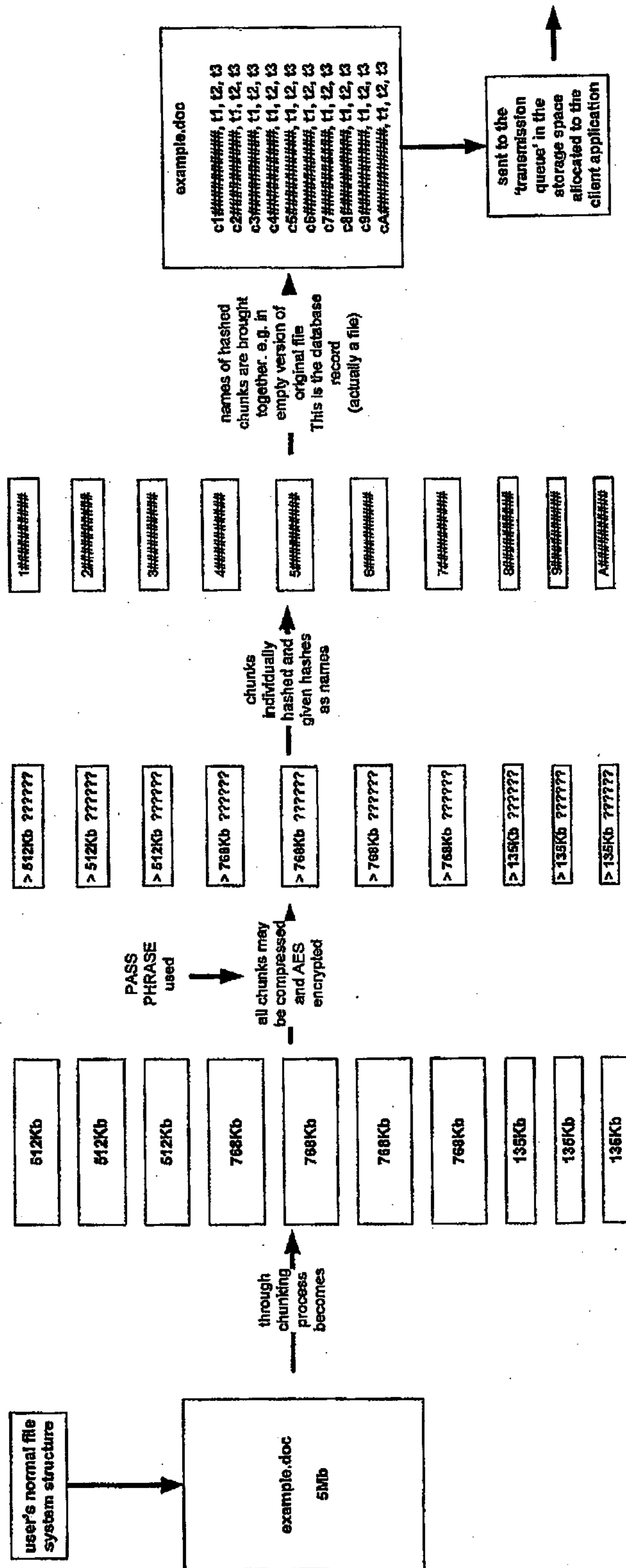


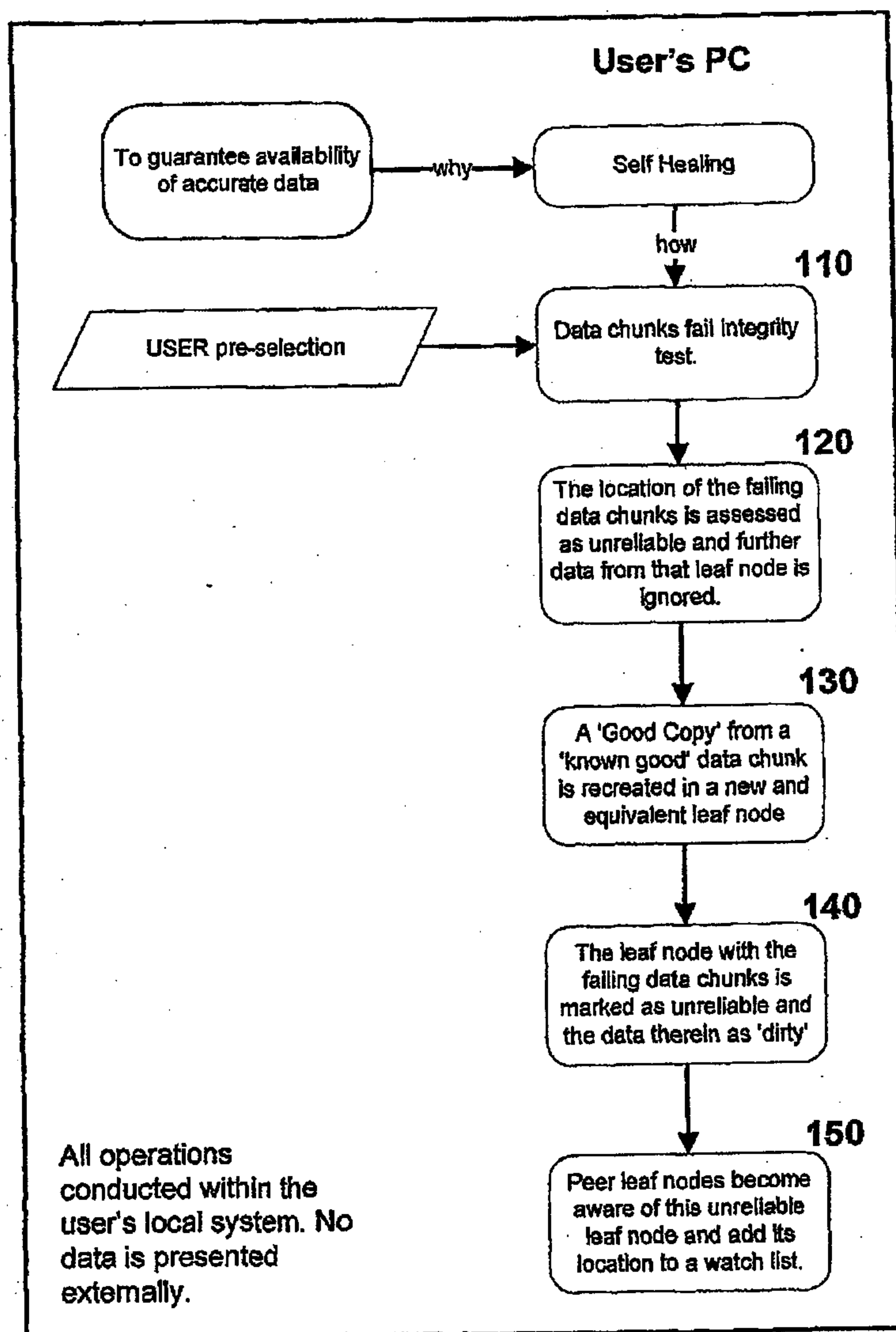
Figure 8 – Self Healing Event Sequence

Figure 9 – Peer Ranking Event Sequence

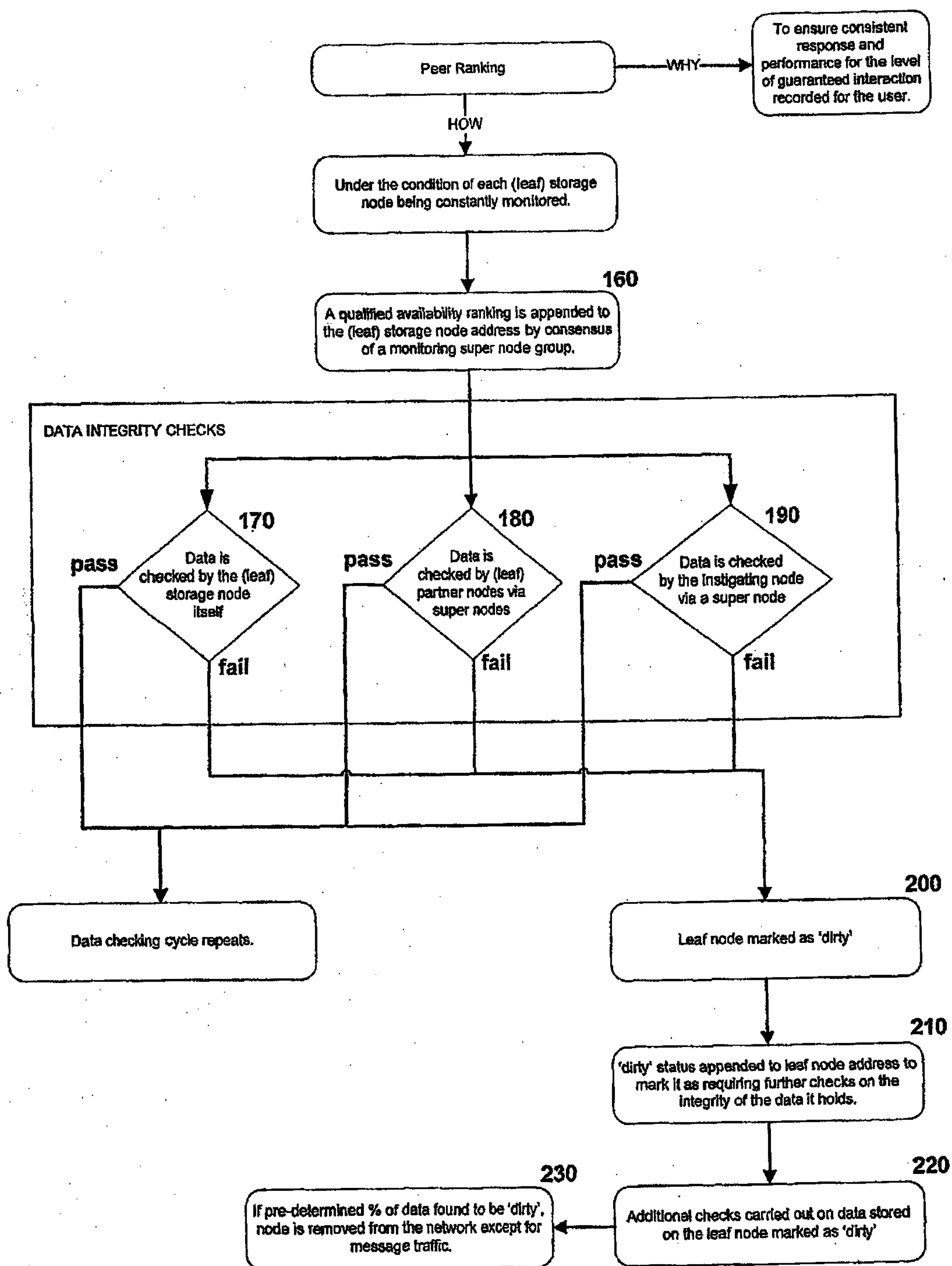


Figure 10 – Duplicate Removal Event Sequence

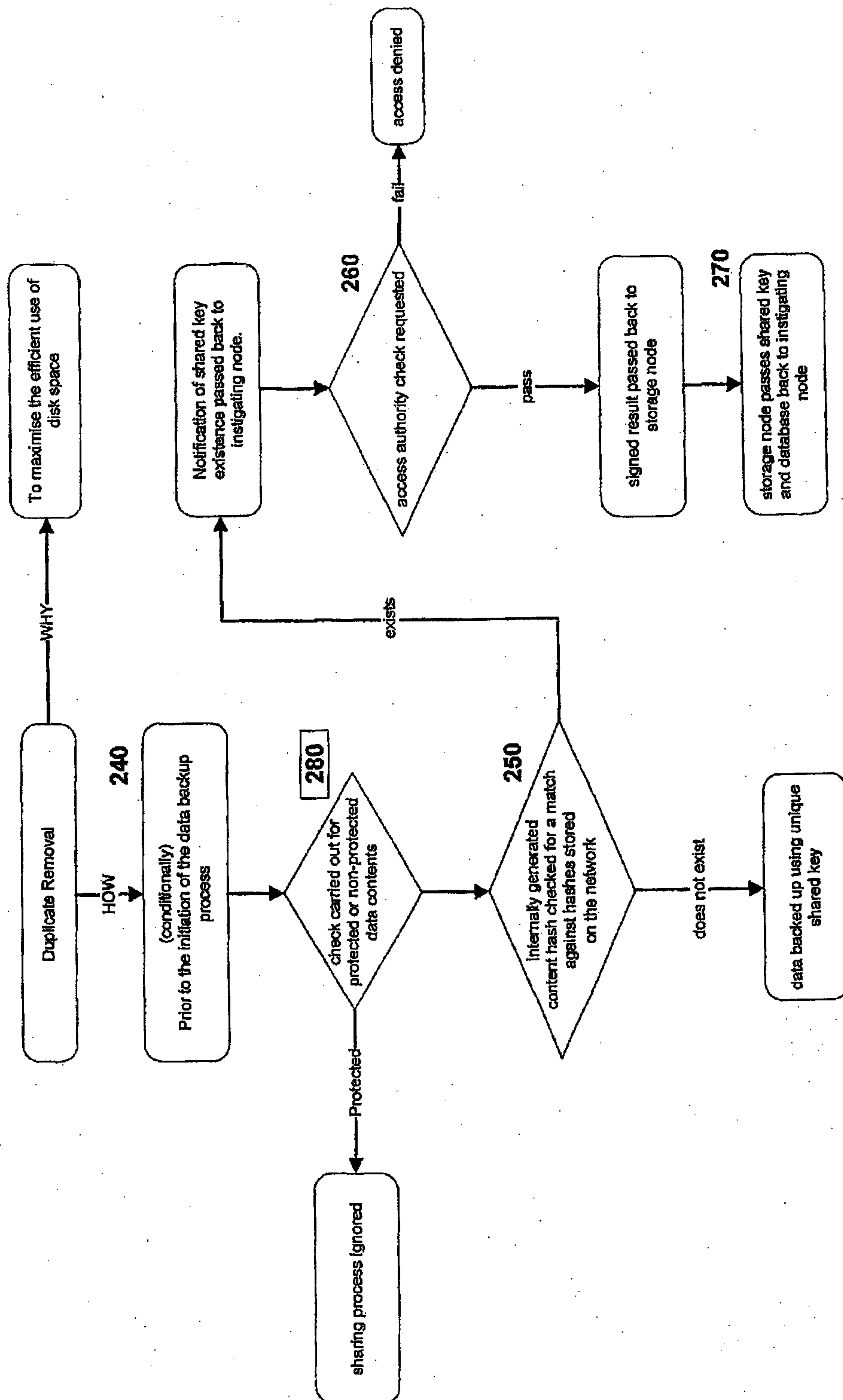
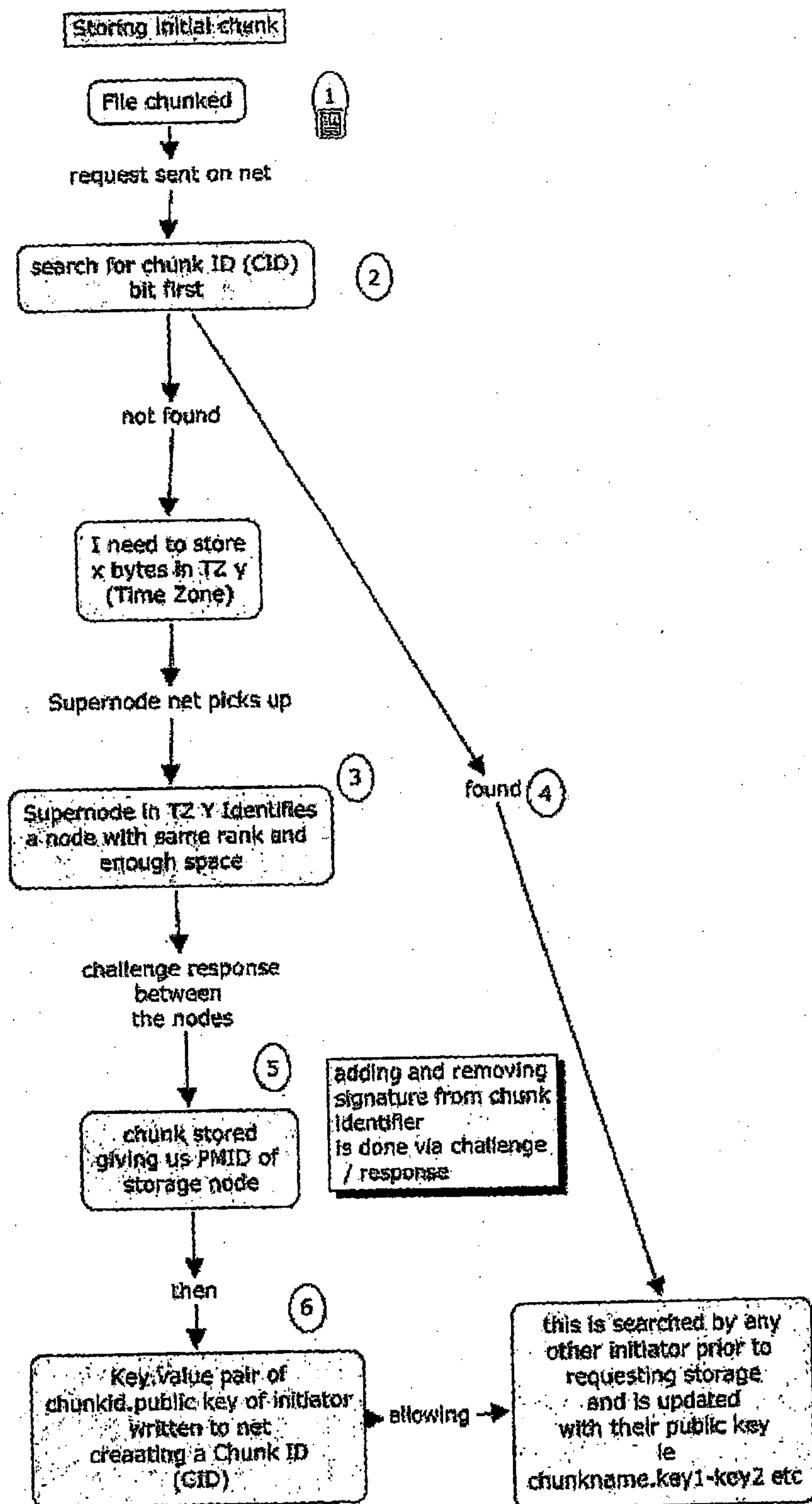


Figure 11 – Perpetual Data

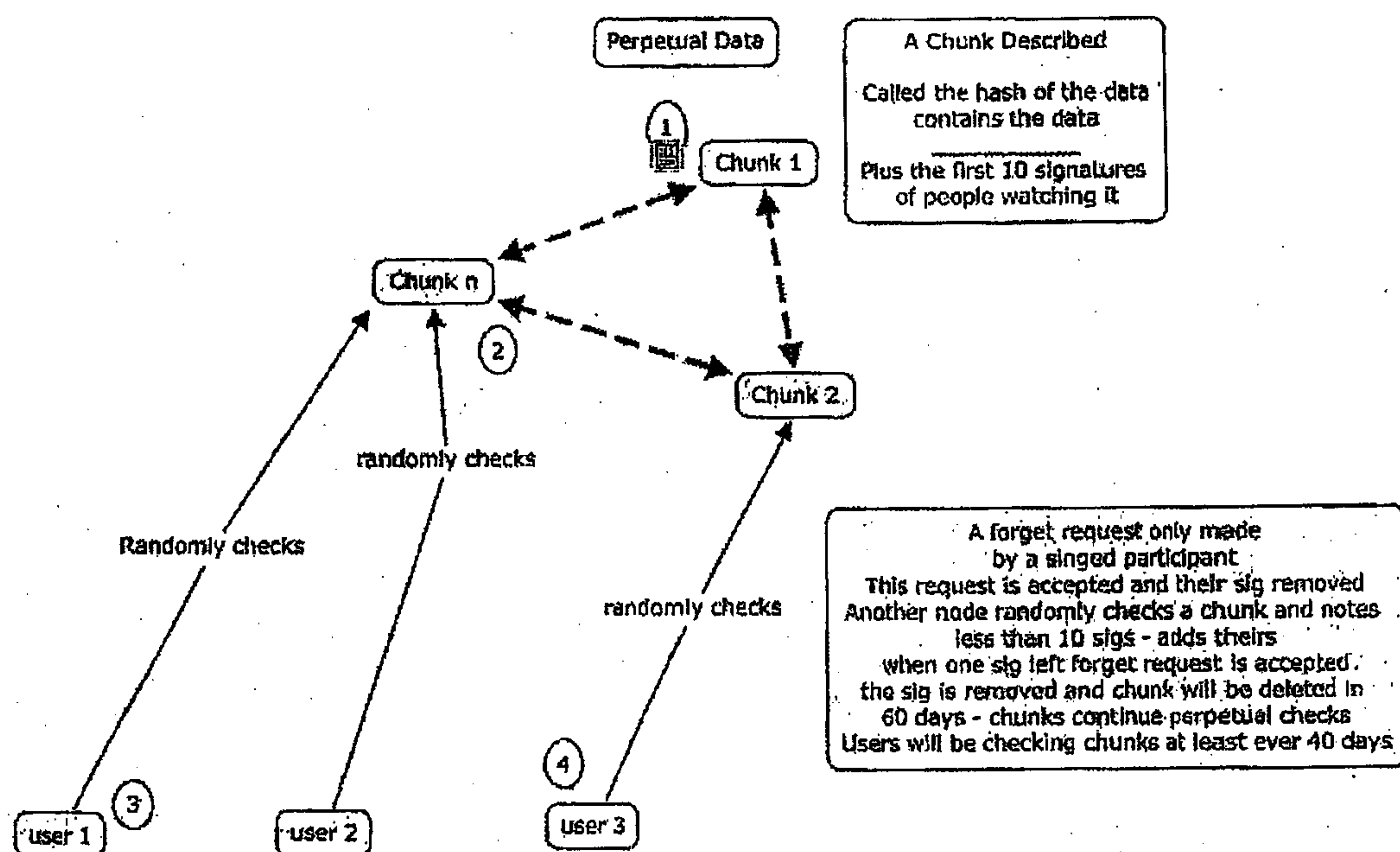


WO 2008/065341

PCT/GB2007/004421

20/28

Figure 12 – Chunk Checks



WO 2008/065341

21/28

PCT/GB2007/004421

Figure 13 – Storage of Additional Chunks

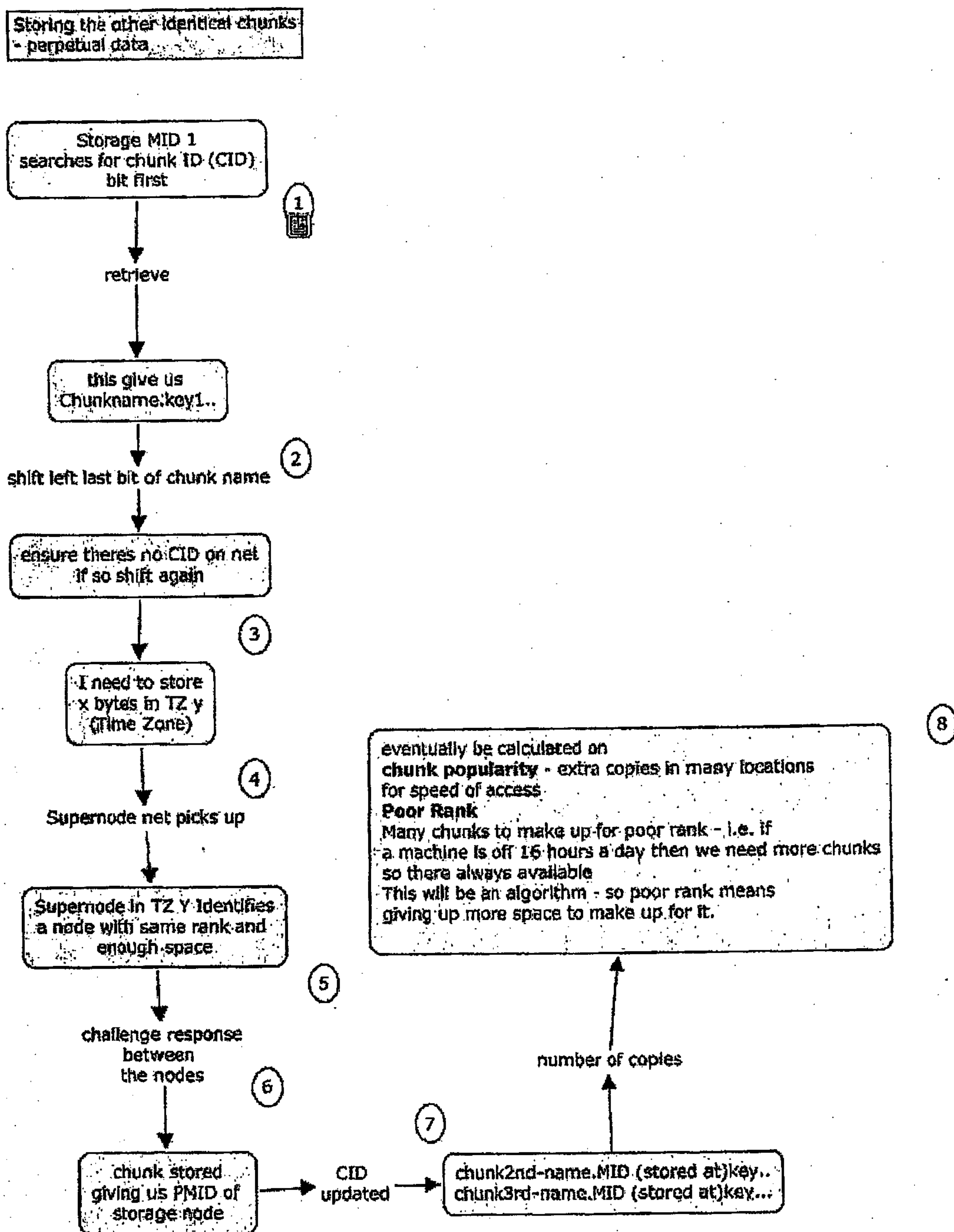
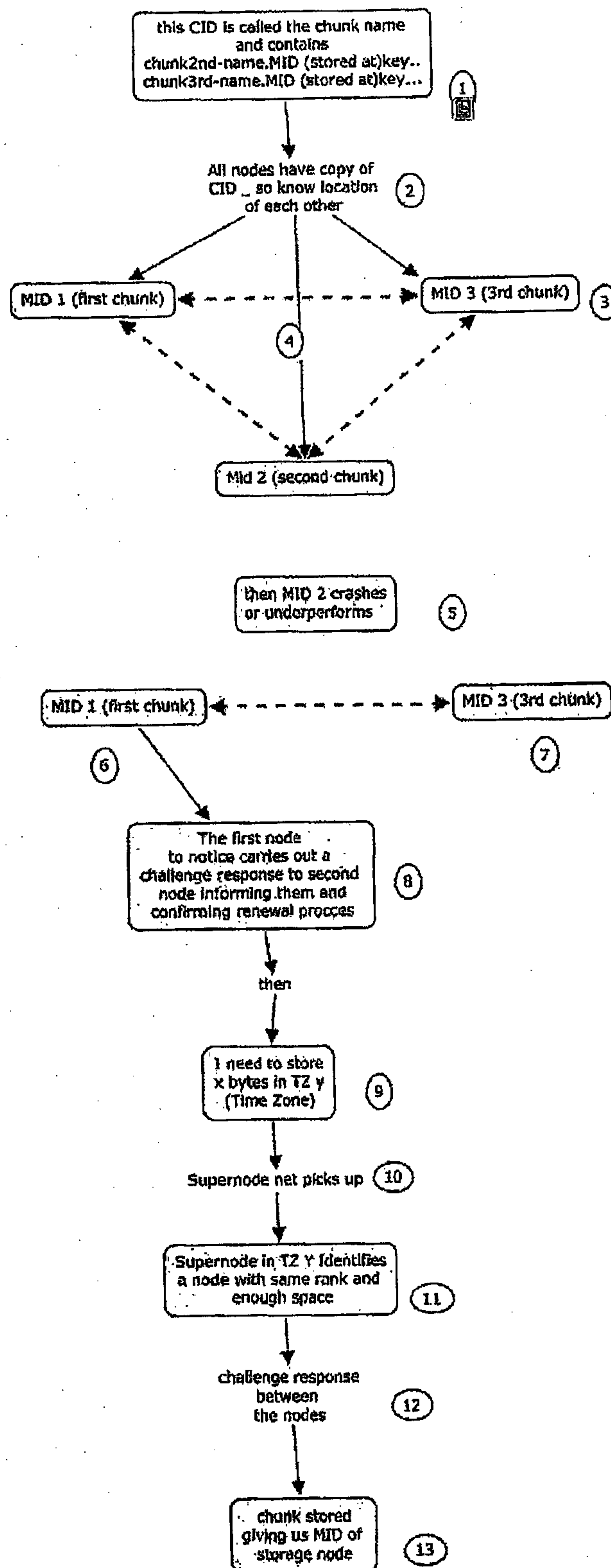


Figure 14 – Self Healing



WO 2008/065341

PCT/GB2007/004421

23/28

Figure 15 – Aput

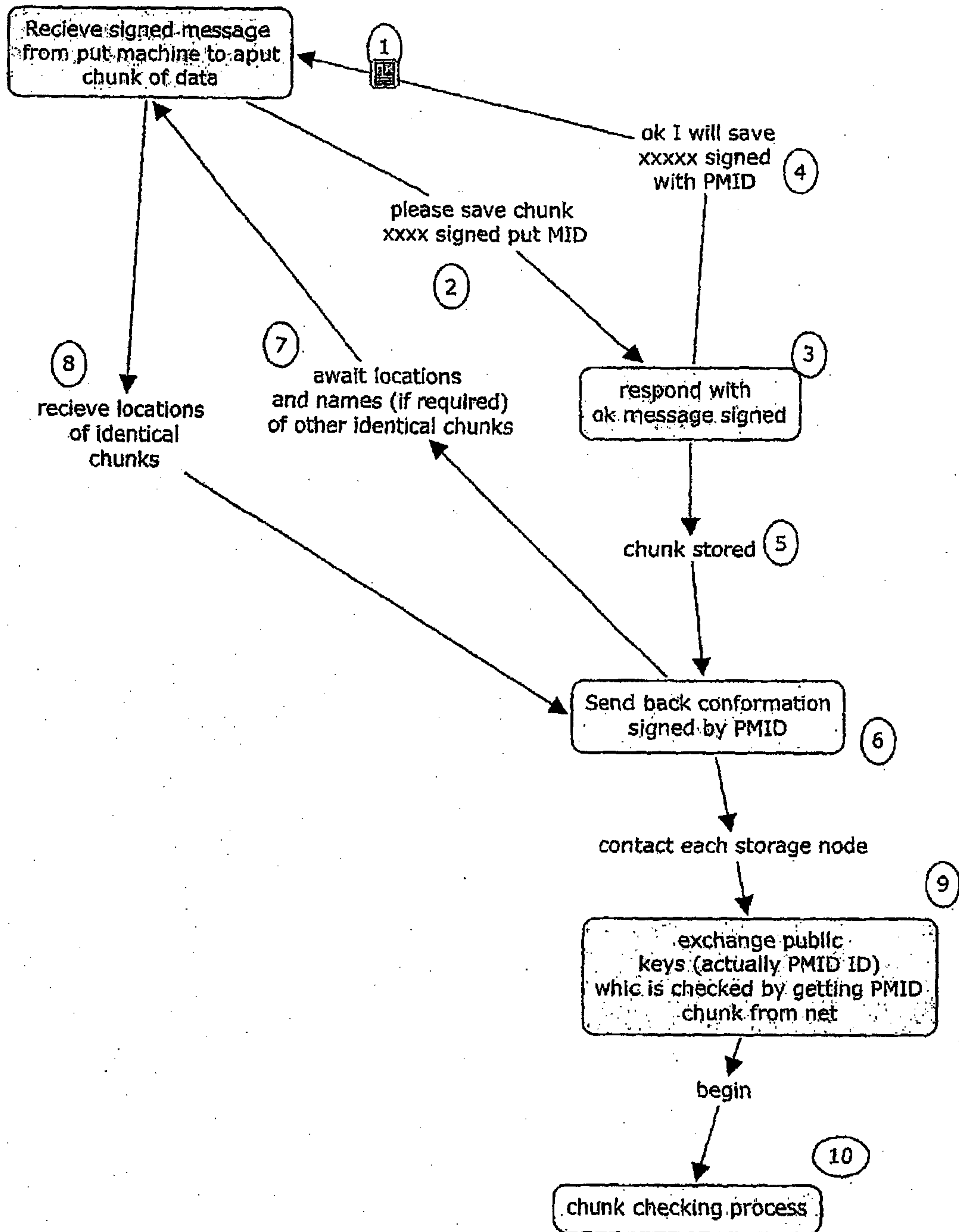
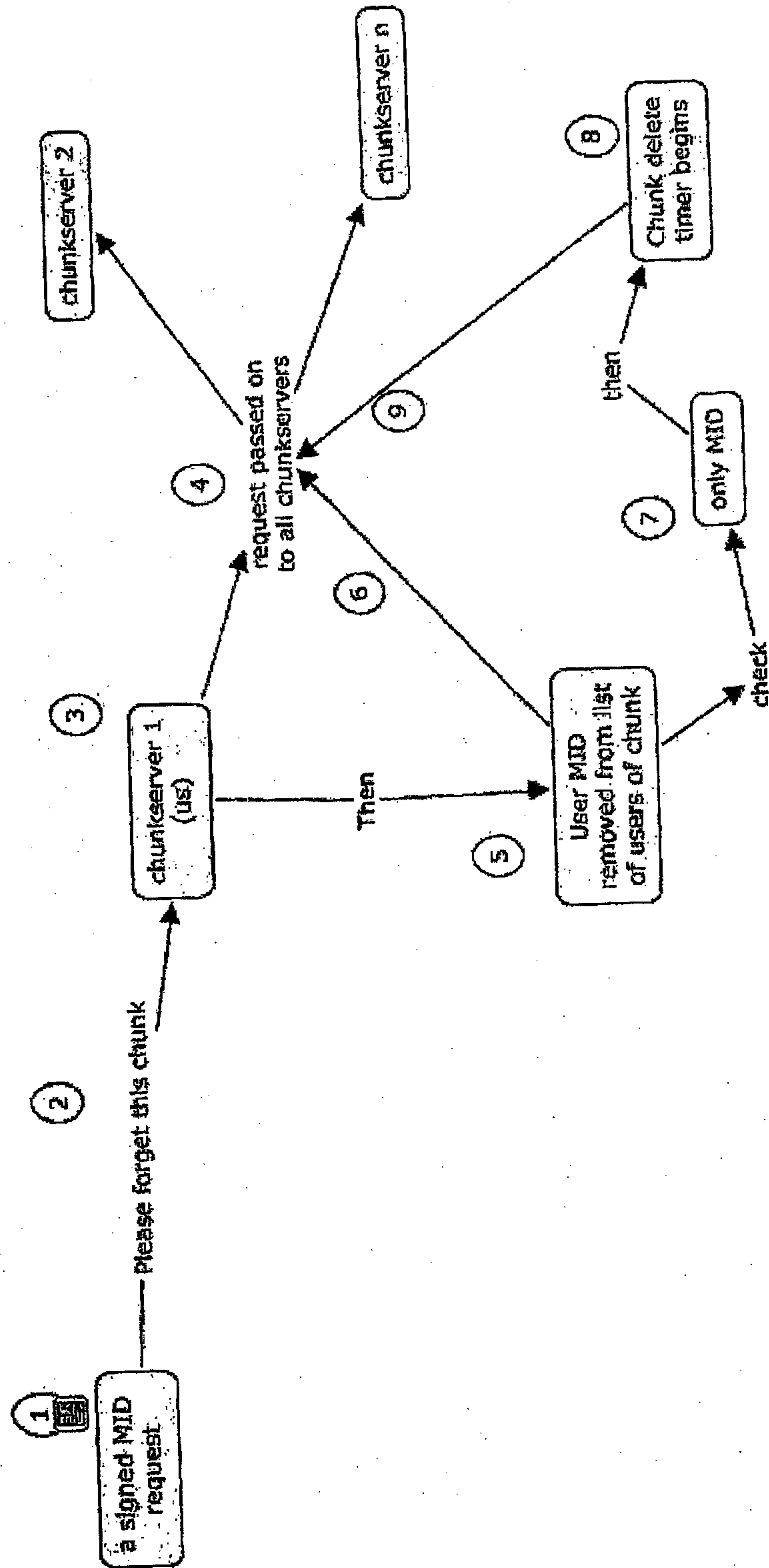


Figure 16 - Aforget



WO 2008/065341

25/28

PCT/GB2007/004421

Figure 17 – Self Encrypting Files

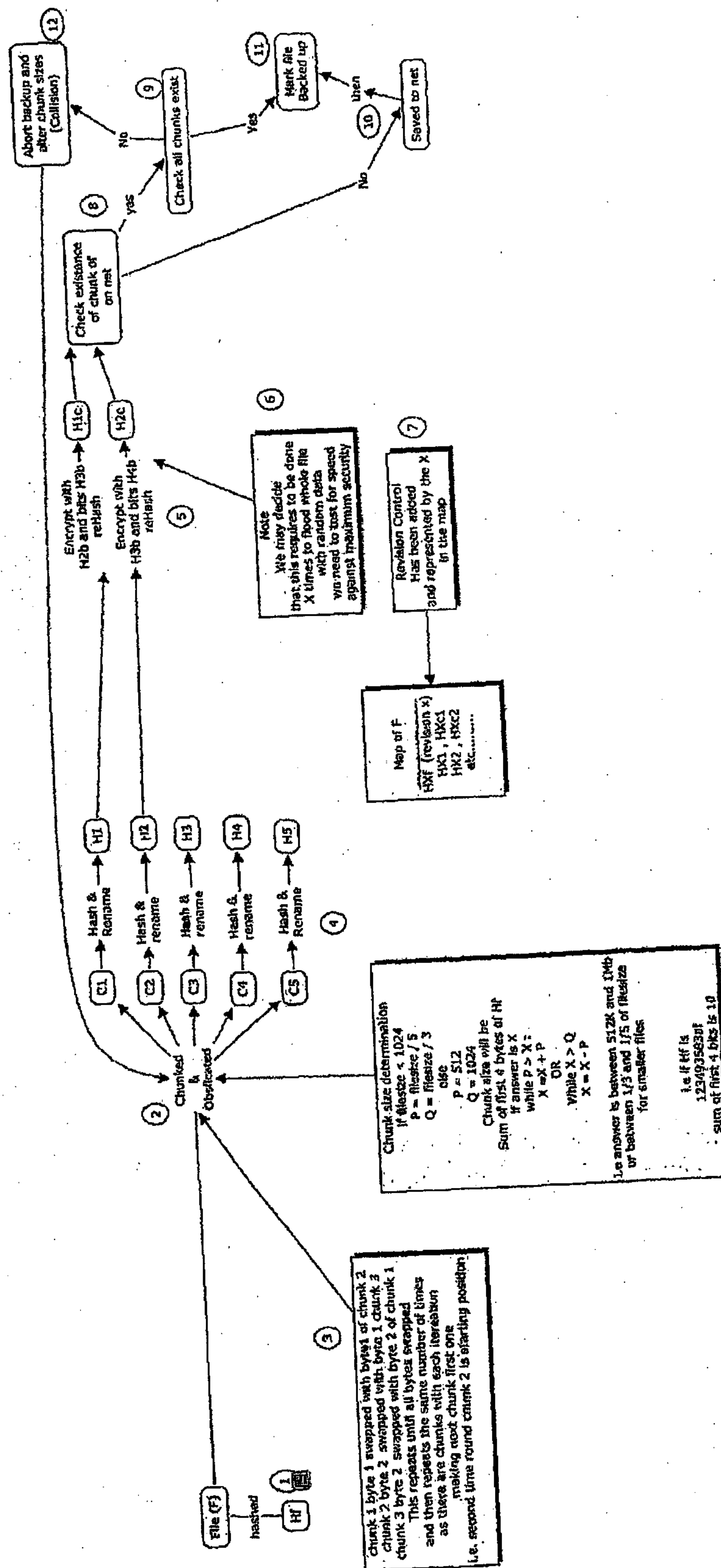


Figure 18 – Shared Access to Private Files

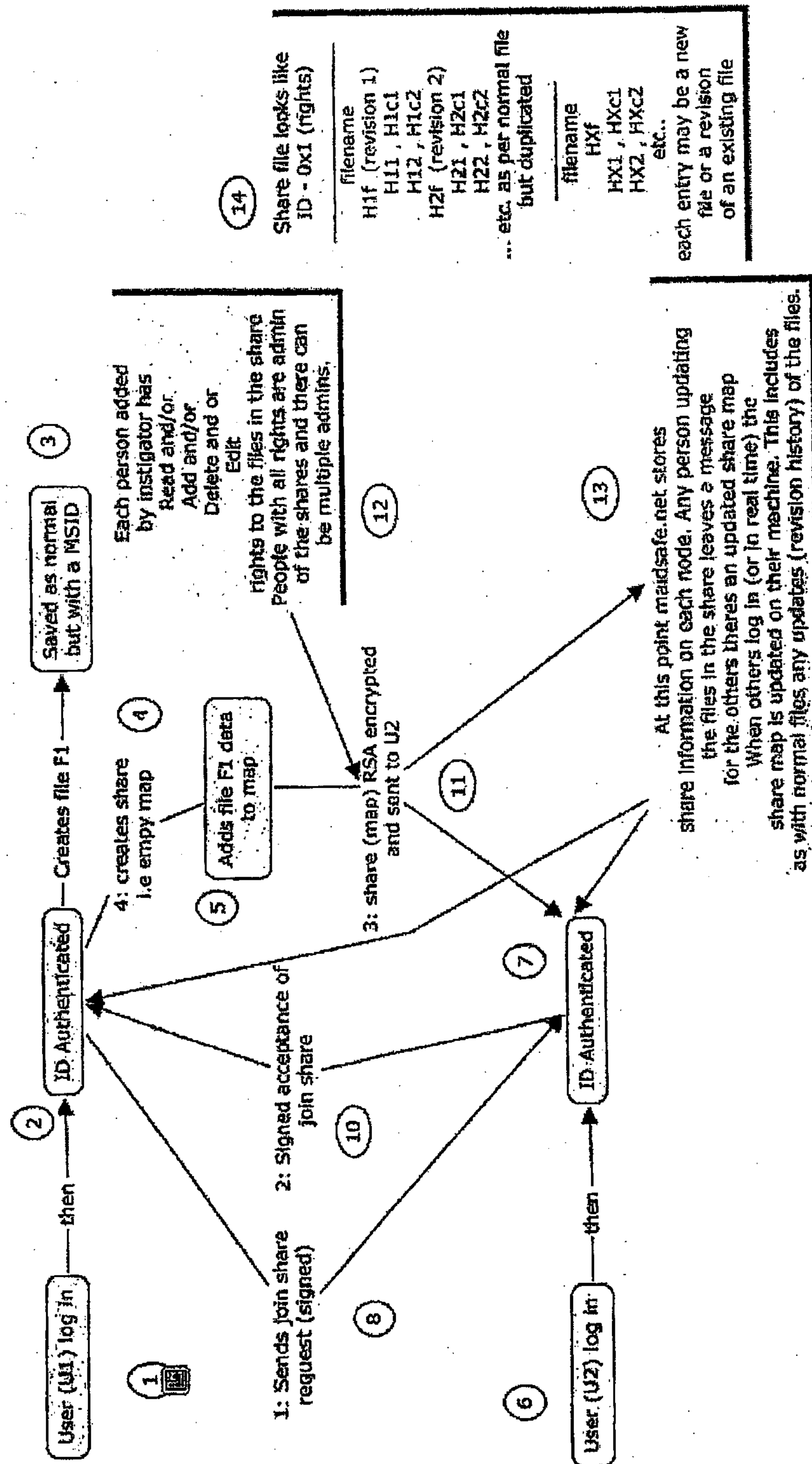


Figure 19 – ms Messenger

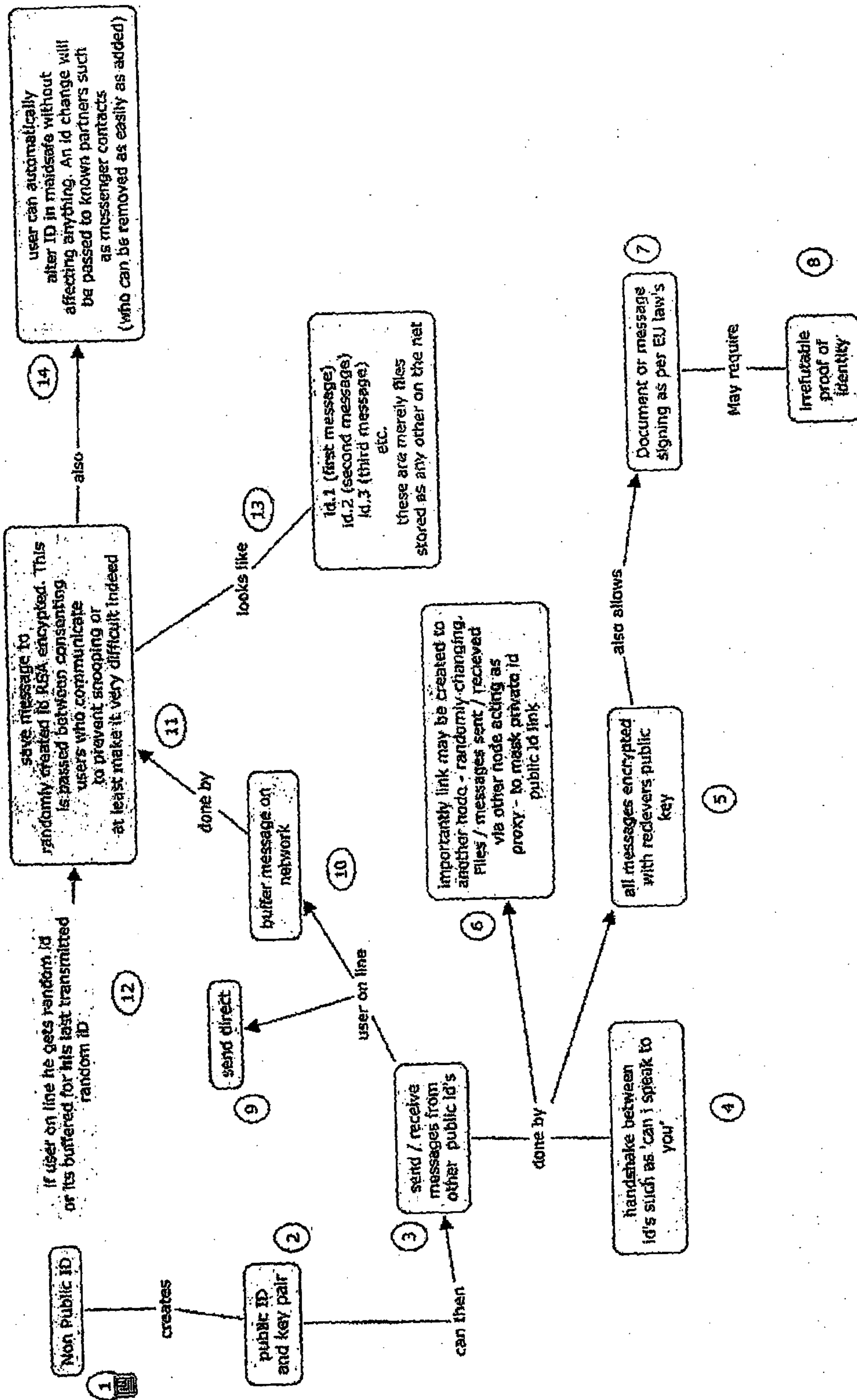
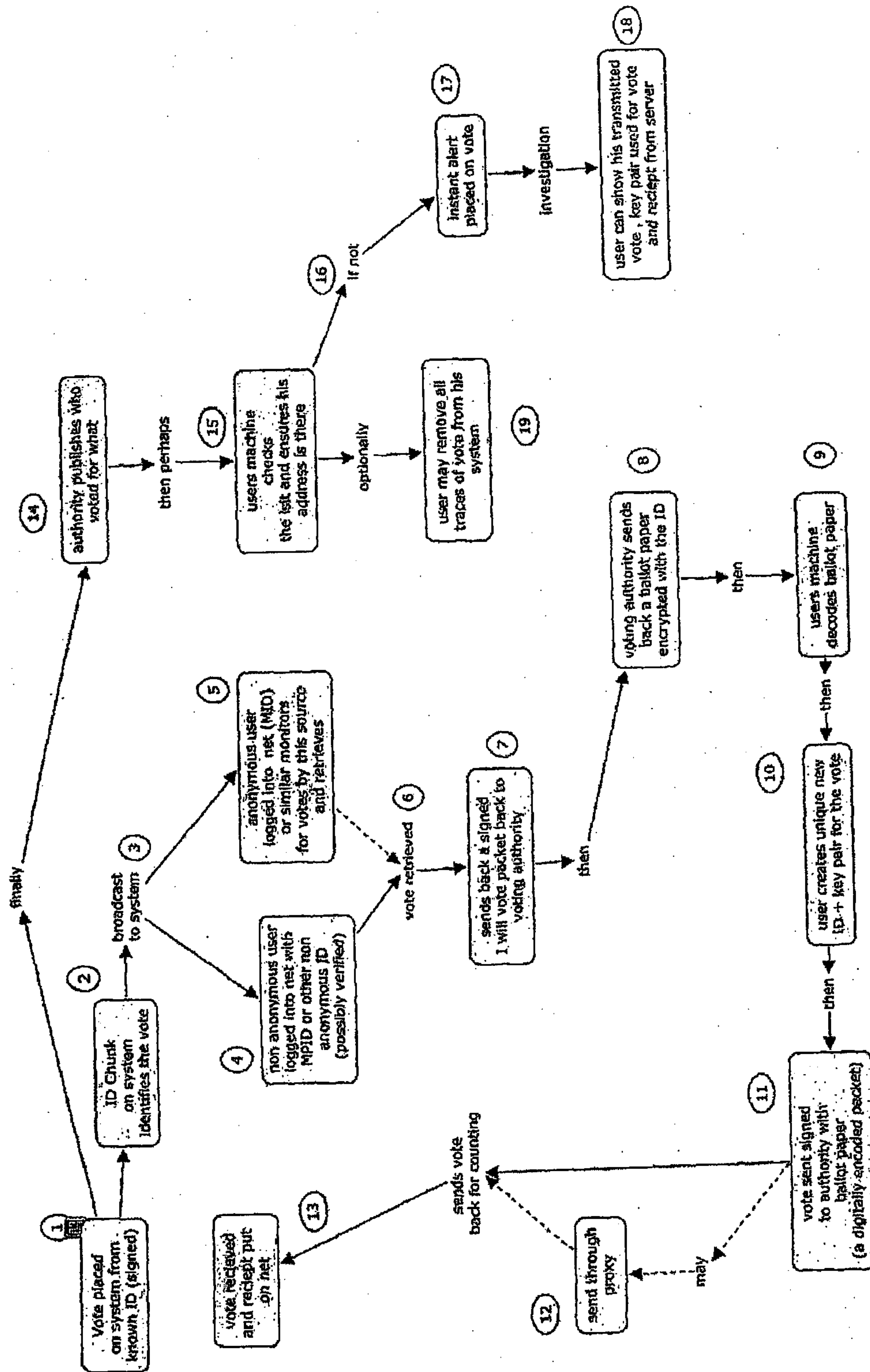


Figure 20 – Worldwide Voting



Authentication Flowchart

