



(43) International Publication Date
3 February 2005 (03.02.2005)

PCT

(10) International Publication Number
WO 2005/010689 A2

(51) International Patent Classification⁷: **G06F**
(21) International Application Number: PCT/US2004/022821

(74) **Agent: ROSENBERG, Gerald**; NewTechLaw, 285 Hamilton Avenue, Suite 520, Palo Alto, California 94301 (US).

(22) International Filing Date: 15 July 2004 (15.07.2004)

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

(25) Filing Language: English

(26) Publication Language: English

(30) Priority Data:
10/622,596 18 July 2003 (18.07.2003) US

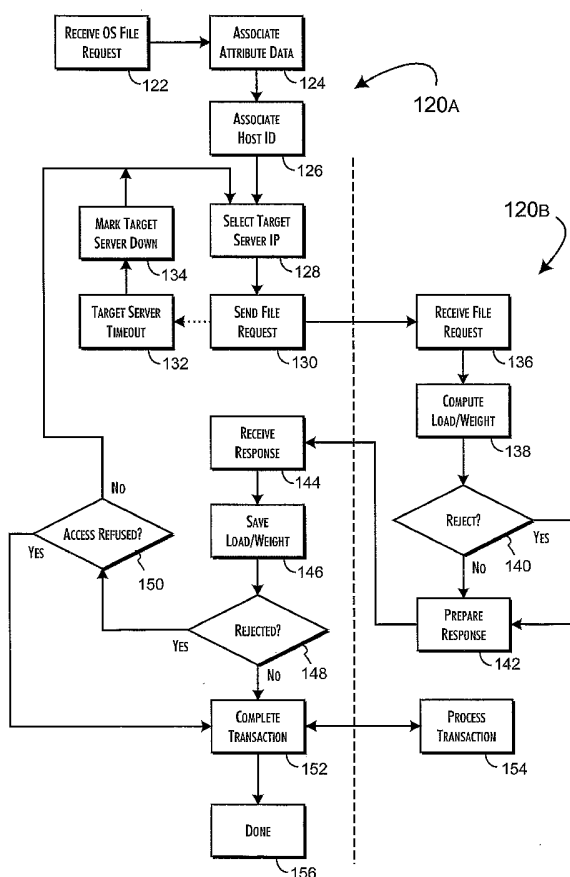
(71) **Applicant** (for all designated States except US): **VORMETRIC, INC.** [US/US]; 3131 Jay Street, Santa Clara, California 95054-3308 (US).

(72) **Inventors:** **ZHANG, Pu**; 6404 Mojave Drive, San Jose, 5 95120 (US). **PHAM, Duc**; 10412 Menhart Lane, Cupertino, 5 95014 (US). **NGUYEN, Tien**; 10105 Stern Ave, Cupertino, 5 95014 (US). **TSAI, Peter**; 5062 Rigatti Circle, Pleasanton, California 94588 (US).

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

[Continued on next page]

(54) Title: SECURE CLUSTER CONFIGURATION DATA SET TRANSFER PROTOCOL



(57) Abstract: Communications between server computer systems of a cluster routinely exchange notice of configuration status and, on demand, transmit updated configuration data sets. Each status message identifies any change in the local configuration of a servers and, further, includes encrypted validation data. Each of the servers stores respective configuration data including respective sets of data identifying the servers known to the respective servers as participating in the cluster. Each status message, as received, is validating against the respective configuration data stored by the receiving server. A status message is determined valid only when originating from a server as known by the receiving server, as determined from the configuration data held by the receiving server. Where a validated originating server identifies updated configuration data, the receiving server requests a copy of the updated configuration data set, which must also be validated, to equivalently modify the locally held configuration data. The configuration of the cluster thus converges on the updated configuration.



FR, GB, GR, HU, IE, IT, LU, MC, NL, PL, PT, RO, SE, SI, SK, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

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

Published:

— *without international search report and to be republished upon receipt of that report*

- 1 -

1 [0001] SECURE CLUSTER CONFIGURATION
2 DATA SET TRANSFER PROTOCOL
3

4 [0002] Inventors:
5 Pu Paul Zhang
6 Duc Pham
7 Tien Le Nguyen
8 Peter Tsai
9
10

11 [0003] Background of the Invention

12 [0004] Field of the Invention:

13 [0005] The present invention is generally related to the
14 coordinated control of server systems utilized to provide network services and,
15 in particular, to techniques for securely coordinating and distributing
16 configuration data among a cluster of network servers and coordinating the
17 implementation of the configuration data with respect to the cluster systems
18 and host computers systems that request execution of network services.
19

20 [0006] Description of the Related Art:

21 [0007] The concept and need for load-balancing arises in a
22 number of different computing circumstances, most often as a requirement for
23 increasing the reliability and scalability of information serving systems.
24 Particularly in the area of networked computing, load-balancing is commonly
25 encountered as a means for efficiently utilizing, in parallel, a large number of
26 information server systems to respond to various processing requests including
27 requests for data from typically remote client computer systems. A logically
28 parallel arrangement of servers adds an intrinsic redundant capability while

- 2 -

1 permitting performance to be scaled linearly, at least theoretically, through the
2 addition of further servers. Efficient distribution of requests and moreover the
3 resulting load then becomes an essential requirement to fully utilizing the
4 paralleled cluster of servers and maximizing performance.

5 [0008] Many different systems have been proposed and variously
6 implemented to perform load-balancing with distinctions typically dependent
7 on the particularities of the load-balancing application. Chung et al. (US
8 Patent 6,470,389) describes the use of a server-side central dispatcher that
9 arbitrates the selection of servers to respond to client domain name service
10 (DNS) requests. Clients direct requests to a defined static DNS cluster-server
11 address that corresponds to the central dispatcher. Each request is then
12 redirected by the dispatcher to an available server that can then return the
13 requested information directly to the client. Since each of the DNS requests
14 are atomic and require well-defined server operations, actual load is presumed
15 to be a function of the rate of requests made to each server. The dispatcher
16 therefore implements just a basic hashing function to distribute requests
17 uniformly to the servers participating in the DNS cluster.

18 [0009] The use of a centralized dispatcher for load-balancing control
19 is architecturally problematic. Since all requests flow through the dispatcher,
20 there is an immediate exposure to a single-point failure stopping the entire
21 operation of the server cluster. Further, there is no direct way to scale the
22 performance of the dispatcher. To handle larger request loads or more
23 complex load-balancing algorithms, the dispatcher must be replaced with
24 higher performance hardware at substantially higher cost.

25 [0010] As an alternative, Chung et al. proposes broadcasting all client
26 requests to all servers within the DNS cluster, thereby obviating the need for a
27 centralized dispatcher. The servers implement mutually exclusive hash
28 functions in individualized broadcast request filter routines to select requests

- 3 -

1 for unique local response. This approach has the unfortunate consequence
2 of requiring each server to initially process, to some degree, each DNS
3 request, reducing the effective level of server performance. Further, the
4 selection of requests to service based on a hash of the requesting client
5 address in effect locks individual DNS servers to statically defined groups of
6 clients. The assumption of equal load distribution will therefore be statistically
7 valid, if at all, only over large numbers of requests. The static nature of the
8 policy filter routines also means that all of the routines must be changed every
9 time a server is added or removed from the cluster to ensure that all requests
10 will be selected by a unique server. Given that in a large server cluster,
11 individual server failures are not uncommon and indeed must be planned for,
12 administrative maintenance of such a cluster is likely difficult if not impractical.

13 [0011] Other techniques have been advanced to load-balance networks
14 of servers under various operating conditions. Perhaps the most prevalent
15 load-balancing techniques take the approach of implementing a background
16 or out-of-channel load monitor that accumulates the information necessary to
17 determine when and where to shift resources among the servers dynamically
18 in response to the actual requests being received. For example, Jorden et al.
19 (US Patent 6,438,652) describes a cluster of network proxy cache servers
20 where each server further operates as a second level proxy cache for all of the
21 other servers within the cluster. A background load monitor observes the
22 server cluster for repeated second level cache requests for particular content
23 objects. Excessive requests for the same content satisfied from the same
24 second level cache is considered an indication that the responding server is
25 overburdened. Based on a balancing of the direct or first level cache request
26 frequency being served by a server and the second level cache request
27 frequency, the load monitor determines whether to copy the content object to

- 4 -

1 one or more other caches, thereby spreading the second level cache work-load
2 for broadly and repeatedly requested content objects.

3 [0012] Where resources, such as simple content objects, cannot be
4 readily shifted to effect load-balancing, alternate approaches have been
5 developed that characteristically operate by selectively transferring requests,
6 typically represented as tasks or processes, to other servers within a cluster
7 network of servers. Since a centralized load-balancing controller is preferably
8 to be avoided, each server is required to implement a monitoring and
9 communications mechanism to determine which other server can
10 accommodate a request and then actually provide for the corresponding
11 request transfer. The process transfer aspect of the mechanism is often
12 implementation specific in that the mechanism will be highly dependent on the
13 particular nature of the task to transfer and range in complexity from a transfer
14 of a discrete data packet representing the specification of a task to the
15 collection and transport of the entire state of an actively executing process.
16 Conversely, the related conventional load monitoring mechanisms can be
17 generally categorized as source or target oriented. Source oriented servers
18 actively monitor the load status of target servers by actively inquiring of and
19 retrieving the load status of at least some subset of target servers within the
20 cluster. Target oriented load monitoring operates on a publication principle
21 where individual target servers broadcast load status information reflecting, at
22 a minimum, a capacity to receive a task transfer.

23 [0013] In general, the source and target sharing of load status
24 information is performed at intervals to allow other servers within the cluster
25 to obtain on demand or aggregate over time some dynamic representation of
26 the available load capacity of the server cluster. For large server clusters,
27 however, the load determination operations are often restricted to local or
28 server relative network neighborhoods to minimize the number of discrete

- 5 -

1 communications operations imposed on the server cluster as a whole. The
2 trade-off is that more distant server load values must propagate through the
3 network over time and, consequently, result in inaccurate loading reports that
4 lead to uneven distribution of load.

5 [0014] A related problem is described in Allon et al. (US Patent
6 5,539,883). Server load values, collected into a server cluster load vector, are
7 incrementally requested or advertized by the various servers of the server
8 cluster. Before a server transfers a local copy of the vector, the load values for
9 the server are updated in the vector. Servers receiving the updated vector in
10 turn update the server local copy of the vector with the received load values
11 based on defined rules. Consequently, the redistribution of load values for
12 some given neighborhood may expose an initially lightly loaded server to a
13 protracted high demand for services. The resulting task overload and
14 consequential refusal of service will last at least until a new load vector
15 reflecting the higher server load values circulates among a sufficient number
16 of the servers to properly reflect the load. To alleviate this problem, Allon et
17 al. further describes a tree-structured distribution pattern for load value
18 information as part of the load-balancing mechanism. Based on the tree-
19 structured transfer of load information, low load values, identifying lightly
20 loaded servers, are aged through distribution to preclude lightly loaded servers
21 from being flooded with task transfers.

22 [0015] Whether source or target originated, load-balancing based on
23 the periodic sharing of load information between the servers of the server
24 cluster operates on the fundamental assumption that the load information is
25 reliable as finally delivered. Task transfer rejections are conventionally treated
26 as fundamental failures and, while often recoverable, require extensive
27 exception processing. Consequently, the performance of individual servers
28 may tend to degrade significantly under progressively increasing load, rather

- 6 -

1 than stabilize, as increasing numbers of task transfer recovery and retries
2 operations are required to ultimately achieve a balanced load distribution.

3 [0016] In circumstances where high load conditions are normally
4 incurred, specialized network protocols have been developed to accelerate the
5 exchange and certainty of loading information. Routers and other switch
6 devices are often clustered in various configurations to share network traffic
7 load. A linking network protocol is provided to provide fail-over monitoring
8 in local redundant router configurations and to share load information
9 between both local and remote routers. Current load information, among
10 other shared information, is propagated at high frequency between devices to
11 continuously reflect the individual load status of the clustered devices. As
12 described in Bare (US Patent 6,493,318) for example, protocol data packets
13 can be richly detailed with information to define and manage the propagation
14 of the load information and to further detail the load status of individual
15 devices within the cluster. Sequence numbers, hop counts, and various flag-
16 bits are used in support of spanning tree-type information distribution
17 algorithms to control protocol packet propagation and prevent loop-backs.
18 The published load values are defined in terms of internal throughput rate and
19 latency cost, which allows other clustered routers a more refined basis for
20 determining preferred routing paths. While effective, the custom protocol
21 utilized by the devices described in Bare essentially requires that substantial
22 parts of the load-balancing protocol be implemented in specialized, high-
23 speed hardware, such as network processors. The efficient handling of such
24 protocols is therefore limited to specialized, not general purpose computer
25 systems.

26 [0017] Ballard (US Patent 6,078,960) describes a client/server system
27 architecture that, among other features, effects a client-directed load-balanced
28 use of a server network. For circumstances where the various server computer

- 7 -

1 systems available for use by client computer systems may be provided by
2 independent service providers and where use of the different servers may
3 involve different cost structures, Ballard describes a client-based approach for
4 selectively distributing load from the clients to distinct individual servers within
5 the server network. By implementing client-based load-balancing, the client
6 computer systems in Ballard are essentially independent of the service provider
7 server network implementation.

8 [0018] To implement the Ballard load-balancing system, each client
9 computer system is provided with a server identification list from which servers
10 are progressively selected to receive client requests. The list specifies load
11 control parameters, such as the percentage load and maximum frequency of
12 client requests that are to be issued, for each server identified in the list. Server
13 loads are only roughly estimated by the clients based on the connection time
14 necessary for a request to complete or the amount of data transferred in
15 response to a request. Client requests are then issued by the individual clients
16 to the servers selected as necessary to statistically conform to the load-
17 balancing profile defined by the load control parameters. While the server
18 identification list and included load control parameters are static as held by a
19 client, the individual clients may nonetheless retrieve new server identification
20 lists at various intervals from dedicated storage locations on the servers.
21 Updated server identification lists are distributed to the servers as needed
22 under the manual direction of an administrator. Updating of the server
23 identification lists allows an administrator to manually adjust the load-balance
24 profiles as needed due to changing client requirements and to accommodate
25 the addition and removal of servers from the network.

26 [0019] The static nature of the server identification lists makes the client-
27 based load-balancing operation of the Ballard system fundamentally
28 unresponsive to the actual operation of the server network. While specific

- 8 -

1 server loading can be estimated by the various clients, only complete failures
2 to respond to client requests are detectable and then handled only by
3 excluding a non-responsive server from further participation in servicing client
4 requests. Consequently, under dynamically varying loading conditions, the
5 one sided load-balancing performed by the clients can seriously misapprehend
6 the actual loading of the server network and further exclude servers from
7 participation at least until re-enabled through manual administrative
8 intervention. Such blind exclusion of a server from the server network only
9 increases the load on the remaining servers and the likelihood that other
10 servers will, in turn, be excluded from the server network. Constant manual
11 administrative monitoring of the active server network, including the manual
12 updating of server identification lists to re-enable servers and to adjust the
13 collective client balancing of load on the server network, is therefore required.
14 Such administrative maintenance is quite slow, at least relative to how quickly
15 users will perceive occasions of poor performance, and costly to the point of
16 operational impracticality.

17 [0020] From the forgoing discussion, it is evident that an improved
18 system and methods for cooperatively load-balancing a cluster of servers is
19 needed. There is also a further need, not even discussed in the prior art, for
20 cooperatively managing the configuration of a server cluster, not only with
21 respect to the interoperation of the servers as part of the cluster, but further as
22 a server cluster providing a composite service to external client computer
23 systems. Also, unaddressed is any need for security over the information
24 exchanged between the servers within a cluster. As clustered systems become
25 more widely used for security sensitive purposes, diversion of any portion of the
26 cluster operation through the interception of shared information or
27 introduction of a compromised server into the cluster represents an
28 unacceptable risk.

- 9 -

1

2

3 [0021] Summary of the Invention

4 [0022] Thus, a general purpose of the present invention is to provide
5 an efficient system and methods of securely coordinating and distributing
6 configuration data among a cluster of network servers to effectively provide a
7 secure system of managing the common configuration of a scalable network
8 service.

9 [0023] This is achieved in the present invention by securely managing
10 communications between server computer systems within a cluster to maintain
11 control over the identification of configuration updates and the distribution of
12 updated configuration data sets. Configuration status messages are routinely
13 exchanged among the servers over a communications network. Each status
14 message identifies any change in the local configuration of a servers and,
15 further, includes encrypted validation data. Each of the servers stores
16 respective configuration data including respective sets of data identifying the
17 servers known to the respective servers as participating in the cluster. Each
18 status message, as received, is validating against the respective configuration
19 data stored by the receiving server. A status message is determined valid when
20 originating from a server as known by the receiving server, as determined from
21 the configuration data held by the receiving server. Where a validated
22 originating server identifies updated configuration data, the receiving server
23 equivalently modifies the locally held configuration data. The configuration of
24 the cluster thus converges on the updated configuration.

25 [0024] Thus, an advantage of the present invention is that acceptance
26 of notice of any update to the configuration data and, further, the acceptance
27 of any subsequently received updated configuration data set is constrained to
28 the set of servers that are mutually known to one another. A receiving server

- 10 -

1 will only accept as valid a message that originates from a server that is already
2 known to the receiving server. Thus, the cluster is a securely closed set of
3 server systems.

4 [0025] Another advantage of the present invention is that, since only
5 light-weight status messages are routinely transmitted among the servers of the
6 cluster, there is minimal processing overhead imposed on the servers to
7 maintain a consistent overall configuration. Updated configuration data sets
8 are transmitted only on demand and generally only occurring after an
9 administrative update is performed.

10 [0026] A further advantage of the present invention is that status
11 messages can serve to both identify the availability of updated configuration
12 sets and to subsequently coordinate the mutual installation of the new
13 configuration data, thereby ensuring consistent operation of the cluster.
14 Configuration version control is also asserted against the host computers to
15 ensure consistent operation.

16 [0027] Still another advantage of the present invention is that both
17 status messages and the transmitted configuration data sets are validated to
18 ensure that they originate from a known participant of the cluster. Receipt
19 validation ensures that rogues and trojans cannot infiltrate the cluster.

20 [0028] Yet another advantage of the present invention is that updated
21 configuration data sets, as encrypted for transmission on demand between
22 servers of the cluster, are further structured to ensure decryption only by a
23 server of the cluster already known to the server that prepares the encrypted;
24 updated configuration data set for transmission.

25

26

- 11 -

1 [0029] Brief Description of the Drawings

2 [0030] Figure 1A is a network diagram illustrating a system environment
3 within which host computer systems directly access network services provided
4 by a server cluster in accordance with a preferred embodiment of the present
5 invention.

6 [0031] Figure 1B is a network diagram illustrating a system environment
7 within which a preferred core network gateway embodiment of the present
8 invention is implemented.

9 [0032] Figure 2 is a detailed block diagram showing the network
10 interconnection between an array of hosts and a cluster of security processor
11 servers constructed in accordance with a preferred embodiment of the present
12 invention.

13 [0033] Figure 3 is a detailed block diagram of a security processor
14 server as constructed in accordance with a preferred embodiment of the
15 present invention.

16 [0034] Figure 4 is a block diagram of a policy enforcement module
17 control process as implemented in a host computer system in accordance with
18 a preferred embodiment of the present invention.

19 [0035] Figure 5 is a simplified block diagram of a security processor
20 server illustrating the load-balancing and policy update functions shared by a
21 server cluster service provider in accordance with a preferred embodiment of
22 the present invention.

23 [0036] Figure 6 is a flow diagram of a transaction process cooperatively
24 performed between a policy enforcement module process and a selected
25 cluster server in accordance with a preferred embodiment of the present
26 invention.

- 12 -

1 [0037] Figure 7A is a flow diagram of a secure cluster server policy
2 update process as performed between the members of a server cluster in
3 accordance with a preferred embodiment of the present invention.

4 [0038] Figure 7B is a block illustration of a secure cluster server policy
5 synchronization message as defined in accordance with a preferred
6 embodiment of the present invention.

7 [0039] Figure 7C is a block illustration of a secure cluster server policy
8 data set transfer message data structure as defined in accordance with a
9 preferred embodiment of the present invention.

10 [0040] Figure 8 is a flow diagram of a process to regenerate a secure
11 cluster server policy data set transfer message in accordance with a preferred
12 embodiment of the present invention.

13 [0041] Figure 9 is a flow diagram illustrating an extended transaction
14 process performed by a host policy enforcement process to account for a
15 version change in the reported secure cluster server policy data set of a cluster
16 server in accordance with a preferred embodiment of the present invention.

17

18

19 [0042] Detailed Description of the Invention

20 [0043] While system architectures have generally followed a
21 client/server paradigm, actual implementations are typically complex and
22 encompass a wide variety of layered network assets. Although architectural
23 generalities are difficult, in all there are fundamentally common requirements
24 of reliability, scalability, and security. As recognized in connection with the
25 present invention, a specific requirement for security commonly exists for at
26 least the core assets, including the server systems and data, of a networked
27 computer system enterprise. The present invention provides for a system and
28 methods of providing a cluster of servers that provide a security service to a

- 13 -

1 variety of hosts established within an enterprise without degrading access to
2 the core assets while maximizing, through efficient load balancing, the
3 utilization of the security server cluster. Those of skill in the art will appreciate
4 that the present invention, while particularly applicable to the implementation
5 of a core network security service, provides fundamentally enables the efficient,
6 load-balanced utilization of a server cluster and, further, enables the efficient
7 and secure administration of the server cluster. As will also be appreciated,
8 in the following detailed description of the preferred embodiments of the
9 present invention, like reference numerals are used to designate like parts as
10 depicted in one or more of the figures.

11 [0044] A basic and preferred system embodiment 10 of the present
12 invention is shown in Figure 1A. Any number of independent host computer
13 systems 12_{1-N} are redundantly connected through a high-speed switch 16 to
14 a security processor cluster 18. The connections between the host computer
15 systems 12_{1-N} , the switch 16 and cluster 18 may use dedicated or shared
16 media and may extend directly or through LAN or WAN connections variously
17 between the host computer systems 12_{1-N} , the switch 16 and cluster 18. In
18 accordance with the preferred embodiments of the present invention, a policy
19 enforcement module (PEM) is implemented on and executed separately by
20 each of the host computer systems 12_{1-N} . Each PEM, as executed, is
21 responsible for selectively routing security related information to the security
22 processor cluster 18 to discretely qualify requested operations by or on behalf
23 of the host computer systems 12_{1-N} . For the preferred embodiments of the
24 present invention, these requests represent a comprehensive combination of
25 authentication, authorization, policy-based permissions and common
26 filesystem related operations. Thus, as appropriate, file data read or written
27 with respect to a data store, generically shown as data store 14, is also routed
28 through the security processor cluster 18 by the PEM executed by the

- 14 -

1 corresponding host computer systems 12_{1-N}. Since all of the operations of the
2 PEMs are, in turn, controlled or qualified by the security processor cluster 18,
3 various operations of the host computer systems 12_{1-N} can be securely
4 monitored and qualified.

5 [0045] An alternate enterprise system embodiment 20 of the present
6 invention implementation of the present invention is shown in Figure 1B. An
7 enterprise network system 20 may include a perimeter network 22
8 interconnecting client computer systems 24_{1-N} through LAN or WAN
9 connections to at least one and, more typically, multiple gateway servers 26_{1-M}
10 that provide access to a core network 28. Core network assets, such as
11 various back-end servers (not shown), SAN and NAS data stores 30, are
12 accessible by the client computer systems 24_{1-N} through the gateway servers
13 26_{1-M} and core network 28.

14 [0046] In accordance with the preferred embodiments of the present
15 invention, the gateway servers 26_{1-M} may implement both perimeter security
16 with respect to the client computer systems 14_{1-N} and core asset security with
17 respect to the core network 28 and attached network assets 30 within the
18 perimeter established by the gateway servers 26_{1-M}. Furthermore, the gateway
19 servers 26_{1-M} may operate as application servers executing data processing
20 programs on behalf of the client computer systems 24_{1-N}. Nominally, the
21 gateway servers 26_{1-M} are provided in the direct path for the processing of
22 network file requests directed to core network assets. Consequently, the overall
23 performance of the network computer system 10 will directly depend, at least
24 in part, on the operational performance, reliability, and scalability of the
25 gateway servers 26_{1-M}.

26 [0047] In implementing the security service of the gateway servers 26_{1-M},
27 client requests are intercepted by each of the gateway servers 26_{1-M} and
28 redirected through a switch 16 to a security processor cluster 18. The switch

- 15 -

1 16 may be a high-speed router fabric where the security processor cluster 18
2 is local to the gateway servers 26_{1-M}. Alternatively, conventional routers may
3 be employed in a redundant configuration to establish backup network
4 connections between the gateway servers 26_{1-M} and security processor cluster
5 18 through the switch 16.

6 [0048] For both embodiments 10, 20, shown in Figures 1A and 1B, the
7 security processor cluster 18 is preferably implemented as a parallel organized
8 array of server computer systems, each configured to provide a common
9 network service. In the preferred embodiments of the present invention, the
10 provided network service includes a firewall-based filtering of network data
11 packets, including network file data transfer requests, and the selective
12 bidirectional encryption and compression of file data, which is performed in
13 response to qualified network file requests. These network requests may
14 originate directly with the host computer systems 12_{1-N}, client computer systems
15 14_{1-N}, and gateway servers 16_{1-M} operating as, for example, application
16 servers or in response to requests received by these systems. The detailed
17 implementation and processes carried out by the individual servers of the
18 security processor cluster 18 are described in copending applications Secure
19 Network File Access Control System, Serial Number 10/201,406, Filed July 22,
20 2002, Logical Access Block Processing Protocol for Transparent Secure File
21 Storage, Serial Number 10/201,409, Filed July 22, 2002, Secure Network File
22 Access Controller Implementing Access Control and Auditing, Serial Number
23 10/201,358, Filed July 22, 2002, and Secure File System Server Architecture
24 and Methods, Serial Number 10/271,050, Filed October 16, 2002, all of
25 which are assigned to the assignee of the present invention and hereby
26 expressly incorporated by reference.

27 [0049] The interoperation 40 of an array of host computers 12_{1-X} and
28 the security processor cluster 18 is shown in greater detail in Figure 2. For the

- 16 -

1 preferred embodiments of the present invention, the host computers 12_{1-X} are
2 otherwise conventional computer systems variously operating as ordinary host
3 computer systems, whether specifically tasked as client computer systems,
4 network proxies, application servers, and database servers. A PEM component
5 42_{1-X} is preferably installed and executed on each of the host computers 12_{1-X}
6 to functionally intercept and selectively process network requests directed to
7 any local and core data stores 14, 30. In summary, the PEM components
8 42_{1-X} selectively forward specific requests in individual transactions to target
9 servers 44_{1-Y} within the security processor cluster 18 for policy evaluation and,
10 as appropriate, further servicing to enable completion of the network requests.
11 In forwarding the requests, the PEM components 42_{1-X} preferably operate
12 autonomously. Information regarding the occurrence of a request or the
13 selection of a target server 44_{1-Y} within the security processor cluster 18 is not
14 required to be shared between the PEM components 42_{1-X}, particularly on any
15 time-critical basis. Indeed, the PEM components 42_{1-X} have no required notice
16 of the presence or operation of other host computers 12_{1-X} throughout
17 operation of the PEM components 42_{1-X} with respect to the security processor
18 cluster 18.

19 [0050] Preferably, each PEM component 42_{1-X} is initially provided with
20 a list identification of the individual target servers 44_{1-Y} within the security
21 processor cluster 18. In response to a network request, a PEM component
22 42_{1-X} selects a discrete target server 44 for the processing of the request and
23 transmits the request through the IP switch 16 to the selected target server 44.
24 Particularly where the PEM component 42_{1-X} executes in response to a local
25 client process, as occurs in the case of application server and similar
26 embodiments, session and process identifier access attributes associated with
27 the client process are collected and provided with the network request. This
28 operation of the PEM component 42_{1-X} is particularly autonomous in that the

- 17 -

1 forwarded network request is preemptively issued to a selected target server 44
2 with the presumption that the request will be accepted and handled by the
3 designated target server 44.

4 [0051] In accordance with the present invention, a target servers 44_{1-Y}
5 will conditionally accept a network request depending on the current resources
6 available to the target server 44_{1-Y} and a policy evaluation of the access
7 attributes provided with the network request. Lack of adequate processing
8 resources or a policy violation, typically reflecting a policy determined
9 unavailability of a local or core asset against which the request was issued, will
10 result in the refusal of the network request by a target server 44_{1-Y}. Otherwise,
11 the target server 44_{1-Y} accepts the request and performs the required network
12 service.

13 [0052] In response to a network request, irrespective of whether the
14 request is ultimately accepted or rejected, a target server 44_{1-Y} returns load
15 and, optionally, weight information as part of the response to the PEM
16 component 42_{1-X} that originated the network request. The load information
17 provides the requesting PEM component 42_{1-X} with a representation of the
18 current data processing load on the target server 44_{1-Y}. The weight
19 information similarly provides the requesting PEM component 42_{1-X} with a
20 current evaluation of the policy determined prioritizing weight for a particular
21 network request, the originating host 12 or gateway server 26 associated with
22 the request, set of access attributes, and the responding target server 44_{1-Y}.
23 Preferably, over the course of numerous network request transactions with the
24 security processor cluster 18, the individual PEM components 42_{1-X} will develop
25 preference profiles for use in identifying the likely best target server 44_{1-Y} to use
26 for handling network requests from specific client computer systems 12_{1-N} and
27 gateway servers 26_{1-M}. While load and weight values reported in individual
28 transactions will age with time and may further vary based on the intricacies

- 18 -

1 of individual policy evaluations, the ongoing active utilization of the host
2 computer systems 12_{1-N} permits the PEM components 42_{1-X} to develop and
3 maintain substantially accurate preference profiles that tend to minimize the
4 occurrence of request rejections by individual target servers 44_{1-Y}. The load
5 distribution of network requests is thereby balanced to the degree necessary
6 to maximize the acceptance rate of network request transactions.

7 [0053] As with the operation of the PEM components 42_{1-X}, the
8 operation of the target servers 44_{1-Y} are essentially autonomous with respect
9 to the receipt and processing of individual network requests. In accordance
10 with the preferred embodiments of the present invention, load information is
11 not required to be shared between the target servers 44_{1-Y} within the cluster 18,
12 particularly in the critical time path of responding to network requests.
13 Preferably, the target servers 44_{1-Y} uniformly operate to receive any network
14 requests presented and, in acknowledgment of the presented request, identify
15 whether the request is accepted, provide load and optional weight information,
16 and specify at least implicitly the reason for rejecting the request.

17 [0054] While not particularly provided to share load information, a
18 communications link between the individual target servers 44_{1-Y} within the
19 security processor cluster 18 is preferably provided. In the preferred
20 embodiments of the present invention, a cluster local area network 46 is
21 established in the preferred embodiments to allow communication of select
22 cluster management information, specifically presence, configuration, and
23 policy information, to be securely shared among the target servers 44_{1-Y}. The
24 cluster local area network 46 communications are protected by using secure
25 sockets layer (SSL) connections and further by use of secure proprietary
26 protocols for the transmission of the management information. Thus, while a
27 separate, physically secure cluster local area network 46 is preferred, the
28 cluster management information may be routed over shared physical networks

- 19 -

1 as necessary to interconnect the target servers 44_{1-y} of the security processor
2 cluster 18.

3 [0055] Preferably, presence information is transmitted by a broadcast
4 protocol periodically identifying, using encrypted identifiers, the participating
5 target servers 44_{1-y} of the security processor cluster 18. The security
6 information is preferably transmitted using a lightweight protocol that operates
7 to ensure the integrity of the security processor cluster 18 by precluding rogue
8 or Trojan devices from joining the cluster 18 or compromising the secure
9 configuration of the target servers 44_{1-y}. A set of configuration policy
10 information is communicated using an additional lightweight protocol that
11 supports controlled propagation of configuration information, including a
12 synchronous update of the policy rules utilized by the individual target servers
13 44_{1-y} within the security processor cluster 18. Given that the presence
14 information is transmitted at a low frequency relative to the nominal rate of
15 network request processing, and the security and configuration policy
16 information protocols execute only on the administrative reconfiguration of the
17 security processor cluster 18, such as through the addition of target servers
18 44_{1-y} and entry of administrative updates to the policy rule sets, the processing
19 overhead imposed on the individual target servers 44_{1-y} to support intra-cluster
20 communications is negligible and independent of the cluster loading.

21 [0056] A block diagram and flow representation of the software
22 architecture 50 utilized in a preferred embodiment of the present invention is
23 shown in Figure 3. Generally inbound network request transactions are
24 processed through a hardware-based network interface controller that
25 supports routable communications sessions through the switch 16. These
26 inbound transactions are processed through a first network interface 52, a
27 protocol processor 54, and a second network interface 54, resulting in
28 outbound transactions redirected through the host computers 12_{1-x} to local and

- 20 -

1 core data processing and storage assets 14, 30. The same, separate, or
2 multiple redundant hardware network interface controllers can be
3 implemented in each target server 44_{1-Y} and correspondingly used to carry the
4 inbound and outbound transactions through the switch 16.

5 [0057] Network request data packets variously received by a target
6 server 44 from PEM components 42_{1-X}, each operating to initiate
7 corresponding network transactions against local and core network assets 14,
8 30, are processed through the protocol processor 54 to initially extract selected
9 network and application data packet control information. Preferably, this
10 control information is wrapped in a conventional TCP data packet by the
11 originating PEM component 42_{1-X} for conventional routed transfer to the target
12 server 44_{1-Y}. Alternately, the control information can be encoded as a
13 proprietary RPC data packet. The extracted network control information
14 includes the TCP, IP, and similar networking protocol layer information, while
15 the extracted application information includes access attributes generated or
16 determined by operation of the originating PEM component 42_{1-X} with respect
17 to the particular client processes and context within which the network request
18 is generated. In the preferred embodiments of the present invention, the
19 application information is a collection of access attributes that directly or
20 indirectly identifies the originating host computer, user and domain,
21 application signature or security credentials, and client session and process
22 identifiers, as available, for the host computer 12_{1-N} that originates the network
23 request. The application information preferably further identifies, as available,
24 the status or level of authentication performed to verify the user. Preferably,
25 a PEM component 42_{1-X} automatically collects the application information into
26 a defined data structure that is then encapsulated as a TCP network data
27 packet for transmission to a target server 44_{1-Y}.

- 21 -

1 [0058] Preferably, the network information exposed by operation of the
2 protocol processor 54 is provided to a transaction control processor 58 and
3 both the network and application control information is provided to a policy
4 parser 60. The transaction control processor 58 operates as a state machine
5 that controls the processing of network data packets through the protocol
6 processor 54 and further coordinates the operation of the policy parser in
7 receiving and evaluating the network and application information. The
8 transaction control processor 58 state machine operation controls the detailed
9 examination of individual network data packets to locate the network and
10 application control information and, in accordance with the preferred
11 embodiments of the present invention, selectively control the encryption and
12 compression processing of an enclosed data payload. Network transaction
13 state is also maintained through operation of the transaction control processor
14 58 state machine. Specifically, the sequences of the network data packets
15 exchanged to implement network file data read and write operations, and
16 other similar transactional operations, are tracked as necessary to maintain the
17 integrity of the transactions while being processed through the protocol
18 processor 54.

19 [0059] In evaluating a network data packet identified by the transaction
20 control processor 58 as an initial network request, the policy parser 60
21 examines selected elements of the available network and application control
22 information. The policy parser 60 is preferably implemented as a rule-based
23 evaluation engine operating against a configuration policy/key data set stored
24 in a policy/key store 62. The rules evaluation preferably implements decision
25 tree logic to determine the level of host computer 12_{1-N} authentication required
26 to enable processing the network file request represented by the network file
27 data packet received, whether that level of authentication has been met,
28 whether the user of a request initiating host computer 12_{1-N} is authorized to

- 22 -

1 access the requested core network assets, and further whether the process and
2 access attributes provided with the network request are adequate to enable
3 access to the specific local or core network resource 14, 30 identified in the
4 network request.

5 [0060] In a preferred embodiment of the present invention, the decision
6 tree logic evaluated in response to a network request to access file data
7 considers user authentication status, user access authorization, and access
8 permissions. Authentication of the user is considered relative to a minimum
9 required authentication level defined in the configuration policy/key data set
10 against a combination of the identified network request core network asset,
11 mount point, target directory and file specification. Authorization of the user
12 against the configuration policy/key data set is considered relative to a
13 combination of the particular network file request, user name and domain,
14 client IP, and client session and client process identifier access attributes.
15 Finally, access permissions are determined by evaluating the user name and
16 domains, mount point, target directory and file specification access attributes
17 with correspondingly specified read/modify/write permission data and other
18 available file related function and access permission constraints as specified
19 in the configuration policy/key data set.

20 [0061] Where PEM components 42_{1-X} function as filesystem proxies,
21 useful to map and redirect filesystem requests for virtually specified data stores
22 to particular local and core network file system data stores 14, 30, data is also
23 stored in the policy/key store 62 to define the set identity of virtual file system
24 mount points accessible to host computer systems 12_{1-N} and the mapping of
25 virtual mount points to real mount points. The policy data can also variously
26 define permitted host computer source IP ranges, whether application
27 authentication is to be enforced as a prerequisite for client access, a limited,
28 permitted set of authenticated digital signatures of authorized applications,

- 23 -

whether user session authentication extends to spawned processes or processes with different user name and domain specifications, and other attribute data that can be used to match or otherwise discriminate, in operation of the policy parser 60, against application information that can be marshaled on demand by the PEM components 42_{1-x} and network information.

[0062] In the preferred embodiments of the present invention, encryption keys are also stored in the policy/key store 62. Preferably, individual encryption keys, as well as applicable compression specifications, are maintained in a logically hierarchical policy set rule structure parseable as a decision tree. Each policy rule provides an specification of some combination of network and application attributes, including the access attributed defined combination of mount point, target directory and file specification, by which permissions constraints on the further processing of the corresponding request can be discriminated. Based on a pending request, a corresponding encryption key is parsed by operation of the policy parser 60 from the policy rule set as needed by the transaction control processor 58 to support the encryption and decryption operations implemented by the protocol processor subject. For the preferred embodiments of the present invention, policy rules and related key data are stored in a hash table permitting rapid evaluation against the network and application information.

[0063] Manual administration of the policy data set data is performed through an administration interface 64, preferably accessed over a private network and through a dedicated administration network interface 66. Updates to the policy data set are preferably exchanged autonomously among the target servers 44_{1-y} of the security processor cluster 18 through the cluster network 46 accessible through a separate cluster network interface 68. A cluster policy protocol controller 70 implements the secure protocols for

- 24 -

1 handling presence broadcast messages, ensuring the security of the cluster 46
2 communications, and exchanging updates to the configuration policy/key data
3 set data.

4 [0064] On receipt of a network request, the transaction control
5 processor 58 determines whether to accept or reject the network request
6 dependent on the evaluation performed by the policy parser 60 and the
7 current processing load values determined for the target server 44. A policy
8 parser 60 based rejection will occur where the request fails authentication,
9 authorization, or permissions policy evaluation. For the initially preferred
10 embodiments of the present invention, rejections are not issued for requests
11 received in excess of the current processing capacity of a target server 44.
12 Received requests are buffered and processed in order of receipt with an
13 acceptable increase in the request response latency. The load value
14 immediately returned in response to a request that is buffered will effectively
15 redirect subsequent network requests from the host computers 12_{1-N} to other
16 target servers 44_{1-Y}. Alternately, any returned load value can be biased
17 upward by a small amount to minimize the receipt of network requests that are
18 actually in excess of the current processing capacity of a target server 44. In
19 an alternate embodiment of the present invention, an actual rejection of a
20 network request may be issued by a target server 44_{1-Y} to expressly preclude
21 exceeding the processing capacity of a target server 44_{1-Y}. A threshold of, for
22 example, 95% load capacity can be set to define when subsequent network
23 requests are to be refused.

24 [0065] To provide the returned load value, a combined load value is
25 preferably computed based on a combination of individual load values
26 determined for the network interface controllers connected to the primary
27 network interfaces 52, 56, main processors, and hardware-based
28 encryption/compression coprocessors employed by a target server 44. This

- 25 -

1 combined load value and, optionally, the individual component load values
2 are returned to the request originating host computer 12_{1-N} in response to the
3 network request. Preferably, at least the combined load value is preferably
4 projected to include handling of the current network request. Depending then
5 on the applicable load policy rules governing the operation of the target server
6 44_{1-Y}, the response returned signals either an acceptance or rejection of the
7 current network request.

8 [0066] In combination with authorization, authentication and
9 permissions evaluation against the network request, the policy parser 60
10 optionally determines a policy set weighting value for the current transaction,
11 preferably irrespective of whether the network request is to be rejected. This
12 policy determined weighting value represents a numerically-based
13 representation of the appropriateness for use of a particular target server 44
14 relative to a particular a network request and associated access attributes. For
15 a preferred embodiment of the present invention, a relative low value in a
16 normalized range of 1 to 100, indicating preferred use, is associated with
17 desired combinations of acceptable network and application information.
18 Higher values are returned to identify generally backup or alternative
19 acceptable use. A preclusive value, defined as any value above a defined
20 threshold such as 90, is returned as an implicit signal to a PEM component
21 42_{1-X} that corresponding network requests are not to be directed to the specific
22 target server 44 except under exigent circumstances.

23 [0067] In response to a network request, a target server 44 returns the
24 reply network data packet including the optional policy determined weighting
25 value, the set of one or more load values, and an identifier indicating the
26 acceptance or rejection of the network request. In accordance with the
27 preferred embodiments of the present invention, the reply network data packet
28 may further specify whether subsequent data packet transfers within the current

- 26 -

1 transaction need be transferred through the security processor cluster 18.
2 Nominally, the data packets of an entire transaction are routed through a
3 corresponding target server 44 to allow for encryption and compression
4 processing. However, where the underlying transported file data is not
5 encrypted or compressed, or where any such encryption or compression is not
6 to be modified, or where the network request does not involve a file data
7 transfer, the current transaction transfer of data need not route the balance of
8 the transaction data packets through the security processor cluster 18. Thus,
9 once the network request of the current transaction has been evaluated and
10 approved by the policy parser 60 of a target server 44, and an acceptance
11 reply packet returned to the host computer 12_{1-N}, the corresponding PEM
12 component 42_{1-X} can selectively bypass use of the security processor cluster 18
13 for the completion of the current transaction.

14 [0068] An exemplary representation of a PEM component 42, as
15 executed, is shown 80 in Figure 4. A PEM control layer 82, executed to
16 implement the control function of the PEM component 42, is preferably
17 installed on a host system 12 as a kernel component under the operating
18 system virtual file system switch or equivalent operating system control
19 structure. In addition to supporting a conventional virtual file system switch
20 interface to the operating system kernel, the PEM control layer 82 preferably
21 implements some combination of a native or network file system or an
22 interface equivalent to the operating system virtual file system switch interface
23 through which to support internal or operating system provided file systems 84.
24 Externally provided file systems 84 preferably include block-oriented interfaces
25 enabling connection to direct access (DAS) and storage network (SAN) data
26 storage assets and file-oriented interfaces permitting access to network
27 attached storage (NAS) network data storage assets.

- 27 -

1 [0069] The PEM control layer 82 preferably also implements an
2 operating system interface that allows the PEM control layer 82 to obtain the
3 hostname or other unique identifier of the host computer system 12, the source
4 session and process identifiers corresponding to the process originating a
5 network file request as received through the virtual file system switch, and any
6 authentication information associated with the user name and domain for the
7 process originating the network file request. In the preferred embodiments of
8 the present invention, these access attributes and the network file request as
9 received by the PEM control layer 82 are placed in a data structure that is
10 wrapped by a conventional TCP data packet. This effectively proprietary TCP
11 data packet is then transmitted through the IP switch 16 to present the network
12 request to a selected target server 44. Alternately, a conventional RPC
13 structure could be used in place of the proprietary data structure.

14 [0070] The selection of the target server 44 is performed by the PEM
15 control layer 82 based on configuration and dynamically collected
16 performance information. A security processor IP address list 86 provides the
17 necessary configuration information to identify each of the target servers 44_{1-Y}
18 within the security processor cluster 18. The IP address list 86 can be provided
19 manually through a static initialization of the PEM component 42 or,
20 preferably, is retrieved as part of an initial configuration data set on an initial
21 execution of the PEM control layer 82 from a designated or default target
22 server 44_{1-Y} of the security processor cluster 18. In the preferred embodiment
23 of the present invention, each PEM component 42_{1-X}, in initial execution,
24 implements an authentication transaction against the security processor cluster
25 18 through which the integrity of the executing PEM control layer 82 is verified
26 and the initial configuration data, including an IP address list 86, is provided
27 to the PEM component 42_{1-X}.

- 28 -

1 [0071] Dynamic information, such as the server load and weight values,
2 is progressively collected by an executing PEM component 42_{1-X} into a SP
3 loads/weights table 88. The load values are timestamped and indexed relative
4 to the reporting target server 44. The weight values are similarly timestamped
5 and indexed. For an initial preferred embodiment, PEM component 42_{1-X}
6 utilizes a round-robin target server 44_{1-Y} selection algorithm, where selection
7 of a next target server 44_{1-Y} occurs whenever the loading of a current target
8 server 44_{1-Y} reaches 100%. Alternately, the load and weight values may be
9 further inversely indexed by any available combination of access attributes
10 including requesting host identifier, user name, domain, session and process
11 identifiers, application identifiers, network file operation requested, core
12 network asset reference, and any mount point, target directory and file
13 specification. Using a hierarchical nearest match algorithm, this stored
14 dynamic information allows a PEM component 42_{1-X} to rapidly establish an
15 ordered list several target servers 44_{1-Y} that are both least loaded and most
16 likely to accept a particular network request. Should the first identified target
17 server 44_{1-Y} reject the request, the next listed target server 44_{1-Y} is tried.

18 [0072] A network latency table 90 is preferably utilized to store dynamic
19 evaluations of network conditions between the PEM control layer 82 and each
20 of the target servers 44_{1-Y}. Minimally, the network latency table 90 is used to
21 identify those target servers 44_{1-Y} that no longer respond to network requests
22 or are otherwise deemed inaccessible. Such unavailable target servers 44_{1-Y}
23 are automatically excluded from the target servers selection process performed
24 by the PEM control layer 82. The network latency table 90 may also be utilized
25 to store timestamped values representing the response latency times and
26 communications cost of the various target servers 44_{1-Y}. These values may be
27 evaluated in conjunction with the weight values as part of the process of

- 29 -

1 determining and ordering of the target servers 44_{1-Y} for receipt of new network
2 requests.

3 [0073] Finally, a preferences table 92 may be implemented to provide
4 a default traffic shaping profile individualized for the PEM component 42_{1-X}.
5 For an alternate embodiment of the present invention, a preferences profile
6 may be assigned to each of the PEM components 42_{1-X} to establish a default
7 allocation or partitioning of the target servers 44_{1-Y} within a security processor
8 cluster 18. By assigning target servers 44_{1-Y} different preference values among
9 the PEM components 42_{1-X} and further evaluating these preference values in
10 conjunction with the weight values, the network traffic between the various host
11 computers 12_{1-N} and individual target servers 44_{1-Y} can be used to flexibly
12 define use of particular target servers 44_{1-Y}. As with the IP address list 86, the
13 contents of the preferences table may be provided by manual initialization of
14 the PEM control layer 82 or retrieved as configuration data from the security
15 processor cluster 18.

16 [0074] A preferred hardware server system 100 for the target servers
17 44_{1-Y} is shown in Figure 5. In the preferred embodiments of the present
18 invention, the software architecture 50, as shown in Figure 3, is substantially
19 executed by one or more main processors 102 with support from one or more
20 peripheral, hardware-based encryption/compression engines 104. One or
21 more primary network interface controllers (NICs) 106 provide a hardware
22 interface to the IP switch 16. Other network interface controllers, such as the
23 controller 108, preferably provide separate, redundant network connections
24 to the secure cluster network 46 and to an administrator console (not shown).
25 A heartbeat timer 110 preferably provides a one second interval interrupt to
26 the main processors to support maintenance operations including, in
27 particular, the secure cluster network management protocols.

- 30 -

1 [0075] The software architecture 50 is preferably implemented as a
2 server control program 112 loaded in and executed by the main processors
3 102 from the main memory of the hardware server system 100. In executing
4 the server control program 112, the main processors 102 preferably perform
5 on-demand acquisition of load values for the primary network interface
6 controller 106, main processors 102, and the encryption/compression engines
7 104. Depending on the specific hardware implementation of the network
8 interface controller 106 and encryption/compression engines 104, individual
9 load values may be read 114 from corresponding hardware registers.
10 Alternately, software-based usage accumulators may be implemented through
11 the execution of the server control program 112 by the main processors 102
12 to track throughput use of the network interface controller 106 and current
13 percentage capacity processing utilization of the encryption/compression
14 engines 104. In the initially preferred embodiments of the present invention,
15 each of the load values represents the percentage utilization of the
16 corresponding hardware resource. The execution of the server control
17 program 112, also provides for establishment of a configuration policy/key
18 data set 116 table also preferably within the main memory of the hardware
19 server system 100 and accessible to the main processors 102. A second table
20 118 is similarly maintained to receive an updated configuration policy/key
21 data set through operation of the secure cluster network 46 protocols.

22 [0076] Figure 6 provides a process flow diagram illustrating the load-
23 balancing operation 120A implemented by a PEM component 42_{1-X} as
24 executed on a host computer 12_{1-N} cooperatively 120B with a selected target
25 server 44 of the security processor cluster 18. On receipt 122 of a network
26 request from a client 14, typically presented through the virtual filesystem
27 switch to the PEM component 42_{1-X} as a filesystem request, the network
28 request is evaluated by the PEM component 42_{1-X} to associate available access

- 31 -

1 attributes 124, including the unique host identifier 126, with the network
2 request. The PEM component 42_{1-X} then selects 128 the IP address of a target
3 server 44 from the security processor cluster 18.

4 [0077] The proprietary TCP-based network request data packet is then
5 constructed to include the corresponding network request and access
6 attributes. This network request is then transmitted 130 through the IP switch
7 16 to the target server 44. A target server response timeout period is set
8 concurrently with the transmission 130 of the network request. On the
9 occurrence of a response timeout 132, the specific target server 44 is marked
10 in the network latency table 90 as down or otherwise non-responsive 134.
11 Another target server 44 is then selected 128 to receive the network request.
12 Preferably, the selection process is reexecuted subject to the unavailability of
13 the non-responsive target server 44. Alternately, the ordered succession of
14 target servers identified upon initial receipt of the network request may be
15 transiently preserved to support retries in the operation of the PEM component
16 42_{1-X}. Preservation of the selection list at least until the corresponding network
17 request is accepted by a target server 44 allows a rejected network request to
18 be immediately retried to the next successive target server without incurring the
19 overhead of reexecuting the target server 44 selection process 128.
20 Depending on the duration of the response timeout 132 period, however, re-
21 use of a selection list may be undesirable since any intervening dynamic
22 updates to the security processor loads and weights table 88 and network
23 latency table 90 will not be considered, potentially leading to a higher rate of
24 rejection on retries. Consequently, reexecution of the target server 44 selection
25 process 128 taking into account all data in the security processor loads and
26 weights table 88 and network latency table 90 is generally preferred.

27 [0078] On receipt 120B of the TCP-based network request 136, a target
28 server 44 initially examines the network request to access to the request and

- 32 -

1 access attribute information. The policy parser 60 is invoked 138 to produce
2 a policy determined weight value for the request. The load values for the
3 relevant hardware components of the target server 44 are also collected. A
4 determination is then made of whether to accept or reject 140 the network
5 request. If the access rights under the policy evaluated network and
6 application information precludes the requested operation, the network request
7 is rejected. For embodiments of the present invention that do not
8 automatically accept and buffer in all permitted network requests, the network
9 request is rejected if the current load or weight values exceed the configuration
10 established threshold load and weight limits applicable to the target server
11 44_{1-Y}. In either event, a corresponding request reply data packet is generated
12 142 and returned.

13 [0079] The network request reply is received 144 by the request
14 originating host computer 12_{1-N} and passed directly to the locally executing
15 PEM component 42_{1-X}. The load and any returned weight values are
16 timestamped and saved to the security processor loads and weights table 88.
17 Optionally, the network latency between the target server 44 and host
18 computer 12_{1-N}, determined from the network request response data packet,
19 is stored in the network latency table 90. If the network request is rejected 148
20 based on insufficient access attributes 150, the transaction is correspondingly
21 completed 152 with respect to the host computer 12_{1-N}. If rejected for other
22 reasons, a next target server 44 is selected 128. Otherwise, the transaction
23 confirmed by the network request reply is processed through the PEM
24 component 42_{1-X} and, as appropriate, transferring network data packets to the
25 target server 44 as necessary for data payload encryption and compression
26 processing 154. On completion of the client requested network file
27 operation 152, the network request transaction is complete 156.

- 33 -

1 [0080] The preferred secure process 160A/160B for distributing
2 presence information and responsively transferring configuration data sets,
3 including the configuration policy/key data, among the target servers 44_{1-Y} of
4 a security processor cluster 18 is generally shown in Figure 7A. In accordance
5 with the preferred embodiments of the present invention, each target server 44
6 transmits various cluster messages on the secure cluster network 46.
7 Preferably, a cluster message 170, generally structured as shown in Figure 7B,
8 includes a cluster message header 172 that defines a message type, header
9 version number, target server 44_{1-Y} identifier or simply source IP address,
10 sequence number, authentication type, and a checksum. The cluster message
11 header 172 further includes a status value 174 and a current policy version
12 number 146, representing the assigned version number of the most current
13 configuration and configuration policy/key data set held by the target server
14 44 transmitting the cluster message 170. The status value 174 is preferably
15 used to define the function of the cluster message. The status types include
16 discovery of the set of target servers 44_{1-Y} within the cluster, the joining, leaving
17 and removal of target servers 44_{1-Y} from the cluster, synchronization of the
18 configuration and configuration policy/key data sets held by the target servers
19 44_{1-Y}, and, where redundant secure cluster networks 46 are available, the
20 switch to a secondary secure cluster network 46.

21 [0081] The cluster message 170, also includes a PK digest 178 that
22 contains a structured list including a secure hash of the public key, the
23 corresponding network IP, and a status field for each target server 44_{1-Y} of the
24 security processor cluster 18, as known by the particular target server 44
25 originating a cluster message 170. Preferably, a secure hash algorithm, such
26 as SHA-1, is used to generate the secure public key hashes. The included
27 status field reflects the known operating state of each target server 44,

- 34 -

1 including synchronization in progress, synchronization done, cluster join, and
2 cluster leave states.

3 [0082] Preferably, the cluster message header 172 also includes a
4 digitally signed copy of the source target server 44 identifier as a basis for
5 assuring the validity of a received cluster message 170. Alternately, a digital
6 signature generated from the cluster message header 172 can be appended
7 to the cluster message 170. In either case, a successful decryption and
8 comparison of the source target server 44 identifier or secure hash of the
9 cluster message header 172 enables a receiving target server 44 to verify that
10 the cluster message 170 is from a known source target server 44 and, where
11 digitally signed, has not been tampered with.

12 [0083] For the preferred embodiments of the present invention, the
13 target servers 44_{1-Y} of a cluster 18 maintain essentially a common
14 configuration to ensure a consistent operating response to any network request
15 made by any host computer 12_{1-X} . To ensure synchronization the configuration
16 of the target servers 44_{1-Y} , cluster synchronization messages are periodically
17 broadcast 160A on the secure cluster network 46 by each of the target servers
18 44_{1-Y} , preferably in response to a hardware interrupt generated by the local
19 heartbeat timer 162. Each cluster synchronization message is sent 164 in a
20 cluster message 170 with a synchronization status 174 value, the current policy
21 version level 176 of the cluster 18, and the securely recognizable set of target
22 servers 44_{1-Y} permitted to participate in the security processor cluster 18,
23 specifically from the frame of reference of the target server 44 originating the
24 cluster synchronization message 170.

25 [0084] Each target server 44 concurrently processes 160B broadcast
26 cluster synchronization messages 170 as received 180 from each of the other
27 active target servers 44_{1-Y} on the secure cluster network 46. As each cluster
28 synchronization message 170 is received 180 and validated as originating

- 35 -

1 from a target server 44 known to validly exist in the security processor cluster
2 18, the receiving target server 44 will search 182 the digests of public keys
3 176 to determine whether the public key of the receiving target server is
4 contained within the digest list 176. If the secure hash equivalent of the public
5 key of a receiving target server 44 is not found 184, the cluster synchronization
6 message 170 is ignored 186. Where the secure hashed public key of the
7 receiving target server 44 is found in a received cluster synchronization
8 message 170, the policy version number 174 is compared to the version
9 number of the local configuration policy/key data set held by the receiving
10 target server 44. If the policy version number 174 is the same or less than that
11 of the local configuration policy/key data set, the cluster synchronization
12 message 170 is again ignored 186.

13 [0085] Where the policy version number 174 identified in a cluster
14 synchronization message 170 is greater than that of the current active
15 configuration policy/key data set, the target server 44 issues a retrieval request
16 190, preferably using an HTTPs protocol, to the target server 44 identified
17 within the corresponding network data packet as the source of the cluster
18 synchronization message 170. The comparatively newer configuration
19 policy/key data set held by the identified source target server 44 is retrieved to
20 update the configuration policy/key data set held by the receiving target server
21 44. The identified source target server 44 responds 192 by returning a source
22 encrypted policy set 200.

23 [0086] As generally detailed in Figure 7C, a source encrypted policy set
24 200 is preferably a defined data structure containing an index 202, a series
25 of encrypted access keys 204_{1-Z} , where Z is the number of target servers 44_{1-Y}
26 known by the identified source target server 44 to be validly participating in
27 security processor cluster 18, an encrypted configuration policy/key data set
28 206, and a policy set digital signature 208. Since the distribution of

- 36 -

1 configuration policy/key data sets 206 may occur successively among the
2 target servers 44_{1-Y}, the number of valid participating target servers 44_{1-Y} may
3 vary from the viewpoint of different target servers 44_{1-Y} of the security
4 processor cluster 18 while a new configuration policy/key data set version is
5 being distributed.

6 [0087] The index 202 preferably contains a record entry for each of the
7 known validly participating target servers 44_{1-Y}. Each record entry preferably
8 stores a secure hash of the public key and an administratively assigned
9 identifier of a corresponding target server 44_{1-Y}. By convention, the first listed
10 record entry corresponds to the source target server 44 that generated the
11 encrypted policy set 200. The encrypted access keys 204_{1-Z} each contain the
12 same triple-DES key, through encrypted with the respective public keys of the
13 known validly participating target servers 44_{1-Y}. The source of the public keys
14 used in encrypting the triple-DES key is the locally held configuration policy/key
15 data set. Consequently, only those target servers 44_{1-Y} that are validly known
16 to the target server 44 that sources an encrypted policy set 200 will be able to
17 first decrypt a corresponding triple-DES encryption key 204_{1-Z} and then
18 successfully decrypt the included configuration policy/key data set 206.

19 [0088] A new triple-DES key is preferably generated using a random
20 function for each policy version of an encrypted policy set 200 constructed by
21 a particular target servers 44_{1-Y}. Alternately, new encrypted policy sets 200 can
22 be reconstructed, each with a different triple-DES key, in response to each
23 HTTPs request received by a particular target servers 44_{1-Y}. The locally held
24 configuration policy/key data set 206 is triple-DES encrypted using the current
25 generated triple-DES key. Finally, a digital signature 208, generated based on
26 a secure hash of the index 202 and list of encrypted access keys 204_{1-Z}, is
27 appended to complete the encrypted policy set 200 structure. The digital
28 signature 208 thus ensures that the source target server 44 identified by the

- 37 -

1 initial secure hash/identifier pair record is in fact the valid source of the
2 encrypted policy set 200.

3 [0089] Referring again to Figure 7A, on retrieval 190 of a source
4 encrypted policy set 200 and further validation as secure and originating from
5 a target server 44 known to validly exist in the security processor cluster 18, the
6 receiving target server 44 searches the public key digest index 202 for digest
7 value matching the public key of the receiving target server 44. Preferably, the
8 index offset location of the matching digest value is used as a pointer to the
9 data structure row containing the corresponding public key encrypted triple-
10 DES key 206 and triple-DES encrypted configuration policy/key data set 204.
11 The private key of the receiving target server 44 is then utilized 210 to recover
12 the triple-DES key 206 that is then used to decrypt the configuration policy/key
13 data set 204. As decrypted, the relatively updated configuration policy/key
14 data set 204 is transferred to and held in the update configuration policy/key
15 data set memory 118 of the receiving target server 44. Pending installation
16 of the updated configuration policy/key data set 204, a target server 44
17 holding a pending updated configuration policy/key data set resumes periodic
18 issuance of cluster synchronization messages 170, though using the updated
19 configuration policy/key data set version number 174.

20 [0090] In accordance with the preferred embodiments of the present
21 invention, updated configuration policy/key data sets 204 are relatively
22 synchronously installed as current configuration policy/key data sets 116 to
23 ensure that the active target servers 44_{1-y} of the security processor cluster 18
24 are concurrently utilizing the same version of the configuration policy/key data
25 set. Effectively synchronized installation is preferably obtained by having each
26 target server 44 wait 212 to install an updated configuration policy/key data
27 set 204 by monitoring cluster synchronization messages 170 until all such
28 messages contain the same updated configuration policy/key data set version

- 38 -

1 number 174. Preferably, a threshold number of cluster synchronization
2 messages 170 must be received from each active target server 44, defined as
3 those valid target servers 44_{1-z} that have issued a cluster synchronization
4 message 170 within a defined time period, for a target server 44 to conclude
5 to install an updated configuration policy/key data set. For the preferred
6 embodiments of the present invention, the threshold number of cluster
7 synchronization messages 170 is two. From the perspective of each target
8 server 44, as soon as all known active target servers 44_{1-y} are recognized as
9 having the same version configuration policy/key data set, the updated
10 configuration policy/key data set 118 is installed 214 as the current
11 configuration policy/key data set 116. The process 160B of updating of a
12 local configuration policy/key data set is then complete 216.

13 [0091] Referring to Figure 8, an updated configuration policy/key data
14 set is generated 220 ultimately as a result of administrative changes made to
15 any of the information stored as the local configuration policy/key set data.
16 Administrative changes 222 may be made to modify access rights and similar
17 data principally considered in the policy evaluation of network requests.
18 Changes may also be made as a consequence of administrative
19 reconfiguration 224 of the security processor cluster 18, typically due to the
20 addition or removal of a target server 44. In accordance with the preferred
21 embodiments of the present invention, administrative changes 222 are made
22 by an administrator by access through the administration interface 64 on any
23 of the target servers 44_{1-y} . The administrative changes 222, such as adding,
24 modifying, and deleting policy rules, changing encryption keys for select policy
25 rule sets, adding and removing public keys for known target servers 44, and
26 modifying the target server 44 IP address lists to be distributed to the client
27 computers 12, when made and confirmed by the administrator, are committed
28 to the local copy of the configuration policy/key data set. On committing the

- 39 -

1 changes 222, the version number of the resulting updated configuration
2 policy/key data set is also automatically incremented 226. For the preferred
3 embodiments, the source encrypted configuration policy/key data set 200 is
4 then regenerated 228 and held pending transfer requests from other target
5 servers 44_{1-Y}. The cluster synchronization message 170 is also preferably
6 regenerated to contain the new policy version number 174 and corresponding
7 digest set of public keys 176 for broadcast in nominal response to the local
8 heartbeat timer 162. Consequently, the newly updated configuration
9 policy/key data set will be automatically distributed and relatively
10 synchronously installed on all other active target servers 44_{1-Y} of the security
11 processor cluster 18.

12 [0092] A reconfiguration of the security processor cluster 18 requires a
13 corresponding administrative change to the configuration policy/key data set
14 to add or remove a corresponding public key 232. In accordance with the
15 preferred embodiments of the present invention, the integrity of the security
16 processor cluster 18 is preserved as against rogue or Trojan target servers
17 44_{1-Y} by requiring the addition of a public key to a configuration policy/key
18 data set to be made only by a locally authenticated system administrator or
19 through communications with a locally known valid and active target server 44
20 of the security processor cluster 18. Specifically, cluster messages 170 from
21 target servers 44 not already identified by a corresponding public key in the
22 installed configuration policy/key data set of a receiving target server 44_{1-Y} are
23 ignored. The public key of a new target server 44 must be administratively
24 entered 232 on another known and valid target server 44 to be, in effect,
25 securely sponsored by that existing member of the security processor cluster 18
26 in order for the new target server 44 to be recognized.

27 [0093] Consequently, the present invention effectively precludes a rogue
28 target server from self-identifying a new public key to enable the rogue to join

- 40 -

1 the security processor cluster 18. The administration interface 64 on each
2 target server 44 preferably requires a unique, secure administrative login in
3 order to make administrative changes 222, 232 to a local configuration
4 policy/key data set. An intruder attempting to install a rogue or Trojan target
5 server 44 must have both access to and specific security pass codes for an
6 existing active target server 44 of the security processor cluster 18 in order to
7 be possibly successful. Since the administrative interface 64 is preferably not
8 physically accessible from the perimeter network 12, core network 18, or
9 cluster network 46, an external breach of the security over the configuration
10 policy/key data set of the security processor cluster 18 is fundamentally
11 precluded.

12 [0094] In accordance with the preferred embodiments of the present
13 intention, the operation of the PEM components 42_{1-x}, on behalf of the host
14 computer systems 12_{1-x}, is also maintained consistent with the version of the
15 configuration policy/key data set installed on each of the target servers 44_{1-y}
16 of the security processor cluster 18. This consistency is maintained to ensure
17 that the policy evaluation of each host computer 12 network request is handled
18 seamlessly irrespective of the particular target server 44 selected to handle the
19 request. As generally shown in Figure 9, the preferred execution 240A of the
20 PEM components 42_{1-x} operates to track the current configuration policy/key
21 data set version number. Generally consistent with the PEM component 42_{1-x}
22 execution 120A, following receipt of a network request 122, the last used
23 policy version number held by the PEM component 42_{1-x} is set 242 with the IP
24 address of the selected target server 44, as determined through the target
25 server selection algorithm 128, in the network request data packet. The last
26 used policy version number is set to zero, as is by default the case on
27 initialization of the PEM component 42_{1-x}, to a value based on initializing
28 configuration data provided by a target server 44 of the security processor

- 41 -

1 cluster 18, or to a value developed by the PEM component 42_{1-x} through the
2 cooperative interaction with the target servers 44 of the security processor
3 cluster 18. The network request data packet is then sent 130 to the chosen
4 target server 44.

5 [0095] The target server 44 process execution 240B is similarly
6 consistent with the process execution 120B nominally executed by the target
7 servers 44_{1-y}. Following receipt of the network request data packet 136, an
8 additional check 244 is executed to compare the policy version number
9 provided in the network request with that of the currently installed
10 configuration policy/key data set. If the version number presented by the
11 network request is less than the installed version number, a bad version
12 number flag is set 246 to force generation of a rejection response 142 further
13 identifying the version number mismatch as a reason for the rejection.
14 Otherwise, the network request is processed consistent with the procedure
15 120B. Preferably, the target server process execution 240B also provides the
16 policy version number of the locally held configuration policy/key data set in
17 the request reply data packet irrespective of whether a bad version number
18 rejection response 142 is generated.

19 [0096] On receipt 144 specifically of a version number mismatch
20 rejection response, a PEM component 42_{1-x} preferably updates the network
21 latency table 90 to mark 248 the corresponding target server 44 as down due
22 to a version number mismatch. Preferably, the reported policy version number
23 is also stored in the network latency table 90. A retry selection 128 of a next
24 target server 44 is then performed unless 250 all target servers 44_{1-y} are then
25 determined unavailable based on the combined information stored by the
26 security processor IP address list 86 and network latency table 90. The PEM
27 component 42_{1-x} then assumes 252 the next higher policy version number as
28 received in a bad version number rejection response 142. Subsequent

- 42 -

1 network requests 122 will also be identified 242 with this new policy version
2 number. The target servers 44_{1-y} previously marked down due to version
3 number mismatches are then marked up 254 in the network latency table 90.
4 A new target server 44 selection is then made 128 to again retry the network
5 request utilizing the updated policy version number. Consequently, each of the
6 PEM components 42_{1-x} will consistently track changes made to the
7 configuration policy/key data set in use by the security processor cluster 18
8 and thereby obtain consistent results independent of the particular target server
9 44 chosen to service any particular network request.

10 [0097] Thus, a system and methods for cooperatively load-balancing
11 a cluster of servers to effectively provide a reliable, scalable network service
12 has been described. While the present invention has been described
13 particularly with reference to a host-based, policy enforcement module inter-
14 operating with a server cluster, the present invention is equally applicable to
15 other specific architectures by employing a host computer system or host proxy
16 to distribute network requests to the servers of a server cluster through
17 cooperative interoperation between the clients and individual servers.
18 Furthermore, while the server cluster service has been described as a security,
19 encryption, and compression service, the system and methods of the present
20 invention are generally applicable to server clusters providing other network
21 services. Also, while the server cluster has been describes as implementing a
22 single, common service, such is only the preferred mode of the present
23 invention. The server cluster may implement multiple independent services that
24 are all cooperatively load-balanced based on the type of network request
25 initially received by a PEM component.

26 [0098] In view of the above description of the preferred embodiments
27 of the present invention, many modifications and variations of the disclosed
28 embodiments will be readily appreciated by those of skill in the art. It is

- 43 -

- 1 therefore to be understood that, within the scope of the appended claims, the
- 2 invention may be practiced otherwise than as specifically described above.

- 44 -

Claims

1 1. A method of managing the secure mutual configuration of a plurality
2 of servers interconnected by a communications network, said method
3 comprising the steps of:

4 a) routinely exchanging status messages between said plurality of
5 servers wherein said status messages identify changes in the mutual
6 configuration of said plurality of servers, wherein each said status message
7 includes encrypted validation data and wherein said plurality of servers stores
8 respective configuration data including respective sets of data identifying the
9 servers known to the respective servers as constituting said plurality of servers;

10 b) validating status messages as respectively received by said plurality
11 of servers against the respective configuration data stored by said plurality of
12 servers wherein status messages are determined valid when originating from
13 a first server as determined known relative to the respective configuration data
14 of a second server; and

15 c) selectively modifying the respective configuration data of said second
16 server.

1 2. The method of Claim 1 wherein said step of selectively modifying
2 includes the steps of:

3 a) retrieving the respective configuration data of said first server; and

4 b) incorporating the respective configuration data of said first server as
5 the respective configuration data of said second server.

1 3. The method of Claim 2 wherein the respective configuration data of
2 said first server includes encrypted validation data and wherein said step of
3 retrieving includes the step of validating the respective configuration data of

- 45 -

4 said first server as originating from a known server of said plurality of servers
5 as determined relative to the respective configuration data of a second server.

1 4. The method of Claim 3 wherein said known server is said first server.

1 5. The method of Claim 4 wherein the encrypted validation data included
2 in the respective configuration data is a digital signature of the respective
3 configuration data.

1 6. The method of Claim 5 wherein the encrypted validation data included
2 in the status messages includes an encrypted identifier of the respective servers
3 originating the status messages.

1 7. The method of Claim 6 the method of Claim 6 wherein the respective
2 configuration data includes respective sets of the public keys corresponding to
3 the servers known to the respective servers as constituting said plurality of
4 servers.

1 8. The method of Claim 7 wherein the status messages include respective
2 configuration data version identifiers and wherein said step of retrieving is
3 responsive to configuration data version identifiers that are more current the
4 configuration data version identifier of the respective configuration data held
5 by a respective server of said plurality of servers.

1 9. The method of Claim 8 wherein a predetermined server of said plurality
2 of servers includes an administrative interface through which the respective
3 configuration data may be modified, said method further comprising the step

- 46 -

4 of revising the respective configuration data version identifier of
5 administratively modified configuration data.

1 10. A method of securely distributing configuration data over a
2 communications network to a plurality of computer systems, each computer
3 system operating to evaluate configuration data, as stored in respective
4 configuration data stores, in response to service requests to determine
5 respective responses, said method comprising:

6 a) receiving, by a computer system, a version message from said
7 communications network;

8 b) verifying said version message using verification encryption data
9 securely held by said computer system;

10 c) determining, based on said version message, to retrieve updated
11 configuration data from a configuration data source server identified relative
12 to said a version number message; and

13 d) installing updated configuration data to the configuration data store
14 of said computer system as retrieved from said configuration data source
15 server, wherein said updated configuration data is retrieved as an encrypted
16 data block and wherein said step of installing includes locating predetermined
17 configuration data within said encrypted data block and decrypting said
18 predetermined configuration data.

1 11. The method of Claim 10 wherein said locating step determines the
2 location of said predetermined configuration data using location encryption
3 data securely held by said computer system.

- 47 -

1 12. The method of Claim 11 wherein said verification encryption data and
2 said location encryption data are related to a private encryption key securely
3 held by said computer system.

1 13. The method of Claim 12 wherein said verification encryption data and
2 said location encryption data are related to respective private encryption keys
3 securely held by said computer system.

1 14. The method of Claim 10 wherein said plurality of computer systems
2 use a common version of said configuration data, wherein said step of
3 installing finally installs said updated configuration data for use by said
4 computer system and wherein said method further comprises the steps of:

5 a) staging said updated configuration data pending completion of the
6 installation of said updated configuration data; and

7 b) waiting for said plurality of computer systems to signal use of a
8 common version of said configuration data whereupon said step of installing
9 completes the installation of said updated configuration data.

1 15. The method of Claim 14 wherein said computer system receives version
2 message from each of the other ones of said plurality of computer systems,
3 wherein said step of determining determines the latest version of configuration
4 data in use or staged for use by each of the other ones of said plurality of
5 computer systems, and wherein said step of waiting waits for each of the other
6 ones of said plurality of computer systems to signal use of a common latest
7 version of said configuration data.

- 48 -

1 16. The method of Claim 15 wherein said plurality of computer systems to
2 signal use of a common latest version of said configuration data through
3 respective version messages.

1 17. A method of securely distributing configuration information through a
2 communications network among a cluster of computer systems providing a
3 network service, wherein configuration information modifications are
4 distributed from a computer system participating in the cluster and mutually
5 coordinated in installation in the participating cluster computer systems to
6 enable a consistent configuration information versioned operation of said
7 cluster of computer systems, said method comprising:

8 a) receiving a modified configuration data set having a predetermined
9 configuration version;

10 b) preparing an encrypted configuration data set by encrypting said
11 modified configuration data set using predetermined encryption keys
12 corresponding to encryption key data included in said modified configuration
13 data set;

14 c) sending a configuration version message, referencing said
15 predetermined configuration version, over the communications network
16 connecting the cluster of computer systems;

17 d) servicing requests to retrieve a copy of said encrypted configuration
18 data set; and

19 e) coordinating, among the cluster of computer systems, installation of
20 said modified configuration data set as the operative configuration data set of
21 the computer systems of the cluster.

1 18. The method of Claim 17 wherein the computer systems of said cluster
2 respectively execute a common network service application, wherein execution

- 49 -

3 of said common network service application is dependent on a respective
4 installed configuration data set, and wherein said step of coordinating
5 provides for the mutually corresponding installation of said modified
6 configuration data set by said computer systems whereby execution of said
7 common network service application is consistent across the cluster of
8 computer systems.

1 19. The method of Claim 18 wherein said modified configuration data set
2 includes individual configuration data sets identified with respective computer
3 systems of the cluster, wherein said modified configuration data set includes
4 a plurality of private encryption keys and wherein said step of preparing
5 provides for the encryption of said modified configuration data using said
6 plurality of private encryption keys.

1 20. The method of Claim 19 wherein said individual data sets are
2 encrypted relative to respective ones of said plurality of private encryption keys
3 and wherein a predetermined computer system of said cluster must have a
4 respective one of said plurality of private encryption keys to decrypt a
5 corresponding one of said individual data sets from said encrypted
6 configuration data set.

1 21. A method of securely distributing configuration data sets among server
2 computer systems of a server cluster, wherein an operative configuration data
3 set is used by an individual server computer system to define the parameters
4 for executing a network service by that server computer system, said method
5 comprising the steps of:

- 50 -

- 6 a) identifying, by a first server computer system of said server cluster, a
7 revised configuration data set held by a second server computer system of said
8 server cluster;
- 9 b) retrieving, by said first server computer system, said revised
10 configuration data set from said second server computer system;
- 11 c) decrypting said revised configuration data set for installation as a
12 current configuration data set for said first server computer system, said revised
13 configuration data set having been uniquely encrypted for decryption by said
14 first server computer system;
- 15 d) verifying, by said first server computer system, that each server
16 computer system of said server cluster has said current configuration data set;
17 and
- 18 e) installing said current configuration data set on said first server
19 computer system as the operative configuration data for said first server
20 computer system.

1 22. The method of Claim 21 wherein said revised configuration data set
2 includes a plurality of respectively encrypted current configuration data sets
3 and wherein said step of decrypting includes the steps of:

- 4 a) locating said current configuration data set, as encrypted uniquely
5 for said first server computer system, from among said plurality of respectively
6 encrypted configuration data sets; and
- 7 b) discretely decrypting said current configuration data set from said
8 revised configuration data set.

1 23. The method of Claim 22 wherein said first server computer system has
2 a unique private decryption key and wherein said step of locating depends on
3 the identification, by said first server computer system, of a representation of

- 51 -

4 said unique private decryption key in said revised configuration data set,
5 whereby location and decryption of said plurality of respectively encrypted
6 current configuration data sets is locked to the respective server computer
7 systems of said server cluster.

1 24. The method of Claim 23 wherein said revised configuration data set
2 includes representations of the unique private decryption keys of the respective
3 server computer systems of said server cluster and wherein said step of
4 locating includes:

5 a) matching a predetermined representation of said unique private
6 decryption key of said first server computer system with a corresponding one
7 of said representations of said revised configuration data set; and

8 b) determining, based on the matched representation of said unique
9 private decryption key of said first server computer system, the location of said
10 current configuration data set, as encrypted, from among said plurality of
11 respectively encrypted configuration data sets.

1 25. The method of Claim 24 wherein said representations of the unique
2 private decryption keys are secure digests of the unique private decryption keys
3 of the respective server computer systems of said server cluster.

1 26. The method of Claim 21 wherein said operative configuration data set
2 includes a first version identifier and wherein said step of identifying includes:

3 a) receiving, by said first server computer system, version messages
4 from the other server computer systems of said server cluster, wherein each
5 version message includes a second version identifier and identifies a version
6 message source server computer system; and

- 52 -

7 b) determining, with respect to a predetermined version message,
8 whether said second version identifier, relative to said first version identifier,
9 corresponds to said revised configuration data set.

1 27. The method of Claim 26 wherein said step of identifying further
2 includes a step of validating said version messages with respect to said first
3 server computer system.

1 28. The method of Claim 27 wherein said first server computer system has
2 a unique private decryption key and wherein said step of validating depends
3 on the identification, by said first server computer system, of a representation
4 of said unique private decryption key in said version messages such that
5 version messages lacking said representation are discarded by said first
6 computer server computer system.

1 29. The method of Claim 28 wherein said version message includes
2 representations of the unique private decryption keys of the respective server
3 computer systems of said server cluster and wherein said step of validating
4 includes matching said representation of said unique private decryption key of
5 said first server computer system with a corresponding one of said
6 representations of said revised configuration data set.

1 30. The method of Claim 29 wherein said representations of the unique
2 private decryption keys are secure digests of the unique private decryption keys
3 of the respective server computer systems of said server cluster.

1 31. The method of Claim 30 wherein said revised configuration data set
2 includes said representations of the unique private decryption keys and a

- 53 -

3 plurality of respectively encrypted current configuration data sets, wherein said
4 step of decrypting includes the steps of:
5 a) matching, by said first server computer system, of said predetermined
6 representation of said unique private decryption key of said first server
7 computer system in said revised configuration data set;
8 b) determining, based on the matched representation of said unique
9 private decryption key of said first server computer system, the location of said
10 current configuration data set, as encrypted, from among said plurality of
11 respectively encrypted configuration data sets; and
12 c) discretely decrypting said current configuration data set from said
13 revised configuration data set, whereby the location and decryption of said
14 plurality of respectively encrypted current configuration data sets is locked to
15 the respective server computer systems of said server cluster.

1 32. A server computer system coupleable through a communications
2 network as part of a computer system cluster to support performance of a
3 network service on behalf of a client computer system, said server computer
4 system comprising:
5 a) a processor operative to execute control programs; and
6 b) a service program operative, through execution by said processor as
7 a control program,
8 to generate responses to predetermined client requests,
9 wherein responses are generated based on an evaluation of an
10 installed configuration data set,
11 said service program being further operative to implement a
12 secure network protocol, interactive with said computer system cluster, to
13 identify and receive an updated configuration data set for installation as said
14 installed configuration data set,

- 54 -

15 said service program including a unique private encryption key,
16 wherein said secure network protocol provides for the transfer
17 of an encrypted configuration data block including a plurality of encrypted
18 updated configuration data sets, a respective one of said plurality of encrypted
19 updated data sets being decryptable using said unique private encryption key.

1 33. The server computer system of Claim 32 wherein said service program
2 is operative to first locate and second decrypt said respective one of said
3 plurality of encrypted updated data sets based on said unique private
4 encryption key.

1 34. The server computer system of Claim 33 wherein said encrypted
2 configuration data block includes a plurality of location references, wherein
3 said service program is operative to associate said unique private encryption
4 key with a corresponding one of said plurality of location references, said
5 service program being further operative to locate said respective one of said
6 plurality of encrypted updated data sets based on said corresponding one of
7 said plurality of location references.

1 35. The server computer system of Claim 34 wherein said corresponding
2 one of said plurality of location references is a secure digest of said unique
3 private encryption key.

1 36. The server computer system of Claim 34 wherein said service program
2 is operative to determine whether to receive said updated configuration data
3 set.

- 55 -

1 37. The server computer system of Claim 36 wherein said installed
2 configuration data set has a first version identifier, wherein said updated
3 configuration data set has a second version identifier, said service program
4 being responsive to said first and second version identifiers to determine
5 whether to receive said updated configuration data set.

1 38. The server computer system of Claim 37 wherein said service program
2 is operative, in response to administrative modifications that produce said
3 updated configuration data set, to generate said encrypted configuration data
4 block with said second version identifier, said service program being further
5 operative to provide a version message containing said second version
6 identifier to said computer system cluster and responsive to requests to transfer
7 said encrypted configuration data block.

1 39. The server computer system of Claim 37 wherein said service program
2 is operative to successively broadcast said version message.

1 40. The server computer system of Claim 39 further comprising a first
2 network connection coupleable to said client computer system and a second
3 network connection coupleable to said computer system cluster, wherein said
4 second network connection is utilized to successively transfer said version
5 message and to transfer said encrypted configuration data block.

1 41. The server computer system of Claim 40 further comprising a third
2 network connection accessible by an administrator for applying administrative
3 modifications that produce said updated configuration data set.

- 56 -

1 42. A method of securely constraining participation of select computer
2 systems in the cooperative operation of a server cluster to insure the integrity
3 of the information transactions among the computer systems of said server
4 cluster, said method comprising the steps of:

5 a) receiving, by a first computer system of a server cluster, a request for
6 the transfer of first specified data, held in a first secure memory store of said
7 first computer system, to a second computer system of said server cluster;

8 b) transmitting, by said first computer system, encrypted information
9 including said first specified data to said second computer system, wherein
10 said encrypted information, as prepared by said first computer system, is
11 further encoded to include a first secure discrete reference;

12 c) verifying said first secure discrete reference against a second secure
13 discrete reference determinable from second specified data stored in a second
14 secure memory store of said second computer system;

15 d) locating, by said second computer system with respect to said first
16 secure discrete reference, a predetermined subset of said encrypted
17 information decryptable by said second computer system to recover said first
18 specified data; and

19 e) installing said first specified data in said second secure memory store
20 of said second computer system.

1 43. The method of Claim 42 wherein said request and said encrypted
2 information are transferred over a communication network interconnecting the
3 computer systems of said server cluster.

1 44. The method of Claim 43 wherein said first and second secure discrete
2 references are secure digests of a secure private encryption key assigned to
3 said second computer systems.

- 57 -

1 45. The method of Claim 44 wherein said first and second specified data
2 respectively includes said first and second secure discrete references.

1 46. The method of Claim 44 wherein said first and second specified data
2 includes said secure private encryption key.

1 47. The method of Claim 44 wherein each of the computer systems of said
2 server cluster are assigned unique secure private encryption keys and wherein
3 a set of secure discrete references, including said first and second secure
4 discrete references, corresponding to the set of unique secure private
5 encryption keys are determinable from the respective specified data stored by
6 said computer systems of said server cluster.

1 48. The method of Claim 47 wherein the set of unique secure private
2 encryption keys are stored in each of the respective specified data stored by
3 said computer systems of said server cluster.

1 49. The method of Claim 47 wherein the respective specified data stored
2 by said computer systems of said server cluster are versioned, said method
3 further comprising the steps of:

4 a) identifying, by said second computer system, that said first specified
5 data is a later version relative to the version of said second specified data; and

6 b) requesting, by said second computer system, said first specified data
7 from said first computer system.

- 58 -

1 50. The method of Claim 49 further comprising the step of coordinating,
2 relative to others of said computer systems of said server cluster, the
3 installation of said first specified data by said second computer system.

1 51. The method of Claim 50 wherein said step of identifying further
2 identifies said first specified data as the latest available version of the
3 respective specified data.

1 52. The method of Claim 50 wherein said second computer system
2 nominally responds to predetermined client requests, said method further
3 comprising the step of declining, by said second computer system, said
4 predetermined client requests in the interim between said step of identifying
5 and said step of installing.

1 53. A method of distributing configuration control data among a cluster of
2 computer systems to ensure consistent operation of the cluster in response to
3 network requests received from host computers, wherein each computer system
4 maintains a local control data set that, in active use, determines the functional
5 operation of the respective computer system, and wherein said cluster of
6 computer systems and said host computers are interconnected by a
7 communications network, said method comprising the steps of:

8 a) storing, in a first computer system of a cluster of computer systems,
9 a first local control data set having a predetermined version number;

10 b) transmitting a cluster message including said predetermined version
11 number from said first computer system to said cluster of computer systems;

12 c) transferring said first local control data set to requesting computer
13 systems of said cluster of computer systems; and

- 59 -

14 d) synchronizing with said requesting computer systems the installation
15 of said first local control data set for active use by said requesting computer
16 systems.

1 54. The method of Claim 53 further comprising the step of changing said
2 predetermined version number in connection with a predetermined
3 modification of said first local control data set, wherein said requesting
4 computer systems are responsive to the change in said predetermined version
5 number.

1 55. The method of Claim 54 wherein said step of synchronization includes
2 the steps of:

3 a) providing, respectively by said requesting computer systems,
4 readiness signals to said cluster to establish a readiness to install said first local
5 control data set; and

6 b) establishing, respectively by said requesting computer systems,
7 receipt of said readiness signals from all of said requesting computer systems
8 to enable respective installation of said first local control data set for active
9 use.

1 56. The method of Claim 55 wherein each actively participating computer
2 system of said cluster routinely performs said step of transmitting, wherein
3 each actively participating computer system may be a requesting computer
4 system relative to any other participating computer system that transmits a
5 cluster message with a relatively more recently predetermined version number,
6 wherein said readiness signals include respective predetermined version
7 numbers, and wherein said step of establishing converges on a common
8 predetermined version number.

- 60 -

1 57. The method of Claim 56 wherein said cluster messages securely
2 circumscribe said actively participating computer systems relative to said
3 respectively transmitting computer systems.

1 58. The method of Claim 57 wherein said readiness signals include
2 respective cluster messages.

1 59. A method of enabling the secure, consistent, single-point management
2 of the individual computer system configurations within a cluster of computer
3 systems provided to perform a common network service in response to network
4 requests provided by host computers, wherein said cluster of computer systems
5 and said host computers are interconnected by a communications network,
6 said method comprising the steps of:
7 a) providing each of said computer systems of said cluster with a active
8 configuration data set that operatively defines the respective operation of said
9 computer system with respect to network requests received from host
10 computers, wherein said active configuration data sets are represented by
11 predefined version values;
12 b) transmitting, mutually among said computer systems of said cluster,
13 cluster messages including respective representations of said predefined
14 version values;
15 c) supporting, with respect to a predetermined computer system of said
16 cluster, provision of a updated configuration data set for installation as said
17 active configuration data set, said updated configuration data set having an
18 updated version value, and wherein said cluster messages transmitted by said
19 predetermined computer system reflect said updated version value;

- 61 -

20 d) propagating said updated configuration data set from said
21 predetermined computer system among said computer systems of said cluster;
22 and

23 e) coordinating the installation of said updated configuration data set
24 as said active configuration data set in each of said computer systems of said
25 cluster.

1 60. The method of Claim 59 wherein said step of coordinating provides for
2 the installation of said updated configuration data set determined respectively
3 based on a convergence of the version values received in said cluster
4 messages from the computer systems of said cluster.

1 61. The method of Claim 60 wherein said provision of said updated
2 configuration data set includes administrative provision.

1 62. The method of Claim 61 wherein said provision of said updated
2 configuration data set includes a modification of the local active configuration
3 data set.

1 63. The method of Claim 62 wherein said step of coordinating provides for
2 the respective installation of said updated configuration data set by the
3 computer systems of the cluster once a respective computer system receives a
4 predetermined number of said cluster messages from each of the other
5 computer systems of said cluster each including said updated version value.

1 64. The method of Claim 63 wherein said predetermined number is two.

- 62 -

2 65. A method of securely establishing consistent operation of a networked
3 cluster of computer systems to provide a network service on behalf of host
4 computer systems, said method comprising the steps of:

5 a) enabling distribution of an updated configuration data set among a
6 cluster of computer systems, wherein each said computer system is operative
7 against a respective active configuration data set, and wherein respective
8 instances of said updated configuration data set are received and held by said
9 cluster of computer systems pending installation as said respective active
10 configuration data sets;

11 b) determining, by each said computer system, when a predetermined
12 installation criteria is met with respect to each said computer system; and

13 c) installing said respective instances of said updated configuration data
14 set as said respective active configuration data sets.

1 66. The method of Claim 66 wherein said step of enabling distribution
2 includes a step of transmitting a message to said cluster of computer systems
3 and wherein said message is encrypted so as to be readable only by those
4 computer systems of said cluster that are predetermined valid participants in
5 said cluster.

1 67. The method of Claim 67 wherein said message, as prepared by a first
2 computer systems of said cluster, is encrypted so as to be readable only by
3 second computer systems of said cluster that are preestablished to said first
4 computer system as valid participants of said cluster.

1 68. The method of Claim 68 further comprising the step of preparing, by
2 said first computer system, said message utilizing encryption codes respectively
3 corresponding to said second computer systems.

- 63 -

1 69. The method of Claim 69 wherein said first computer system securely
2 stores a preestablished set of public key encryption codes corresponding to
3 said second computer systems and wherein said second computer systems are
4 only responsive to said message where said message stores a secure
5 representation of the respective public key encryption code of a corresponding
6 one of said second computer systems that receives said message.

1 70. The method of Claim 65 further comprising the steps of:
2 a) storing, by a first computer system of said cluster, a set of encryption
3 codes corresponding to second computer systems of said cluster, said set of
4 encryption codes being preestablished, representing predetermined valid
5 participants in said cluster, and securely stored by said first computer system;
6 b) preparing, by said first computer system, a message for distribution
7 to said second computer systems, said message including secure
8 representations of said encryption codes;
9 c) transmitting said message to said cluster; and
10 d) validating said message by said second computer systems upon
11 recognizing respective instances of said representations of said encryption
12 codes.

1 71. The method of Claim 70 further comprising the step of retrieving, by
2 a predetermined second computer system, a copy of said set of encryption
3 codes from said first computer system for use by said predetermined second
4 computer system subsequently in the role of said first computer system.

1 72. The method of Claim 71 wherein said secure representations are secure
2 hashes of the public keys of said second computer systems.

- 64 -

1 73. A method of securely constraining participation of select computer
2 systems in the operation of a server cluster, interconnected by a
3 communications network, to insure the security of configuration information
4 distributed among the computer systems of said server cluster, said method
5 comprising the steps of:

6 a) storing, by a first computer system of a server cluster, first specified
7 data within a secure memory store, said first specified data being identified by
8 a first version identifier;

9 b) receiving, by said first computer system, a cluster message including
10 a second version identifier and verification data from a second computer
11 system of said server cluster;

12 c) verifying, by said first computer system, said cluster message by
13 evaluation of said verification data relative to said first specified data;

14 d) obtaining, from said second computer system, dependent on a
15 successful verification of said verification data, encrypted information including
16 second specified data corresponding to said second version identifier;

17 e) decrypting said second specified data from said encrypted
18 information; and

19 f) incorporating said second specified data into said secure memory
20 store of said first computer system, whereby the operation configuration of said
21 first computer system is securely made consistent with that of said second
22 computer system.

1 74. The method of Claim 73 wherein said first and second specified data
2 is configuration information that, as incorporated in said secure memory store,
3 determines the operational performance of said first computer system of said

- 65 -

4 server cluster in responding to network requests issued by any of a plurality of
5 host computer systems to obtain a performance of a network service.

1 75. The method of Claim 74 wherein said first specified data includes a
2 predetermined set of encryption codes corresponding to the validly
3 participating computer systems of said server cluster as known to said first
4 computer system, and wherein said step of validating said cluster message
5 includes requiring the valid decryption of said verification data using an
6 encryption code of said predetermined set identified with said second
7 computer system, whereby cluster messages are accepted as valid by said first
8 computer system only from preestablished validly participating computer
9 system of said server cluster.

1 76. The method of Claim 75 wherein changes to said predetermined set of
2 encryption codes are constrained to secure update procedures including
3 incorporation of said second specified data into said secure memory store and
4 administrative operations securely executed against a computer system of said
5 server cluster, whereby configuration data is securely maintained and
6 distributed only among the computer systems of said server cluster of
7 administratively preestablished identity.

1 77. The method of Claim 76 wherein said verification data includes a
2 predetermined cipher text encrypted to ensure secure identification of said
3 request as originating from said second computer system.

1 78. The method of Claim 77 wherein said verification data includes a
2 predetermined cipher text encrypted with an encryption key pair corresponding
3 to said second computer system.

- 66 -

1 79. A computer system operable as a participant in a cluster of computer
2 systems providing a network service in response to network requests received
3 from host computer systems through a communications network, the cluster
4 computer systems interoperating to ensure the security and secure exchange
5 of configuration data in response to a single point secure administrative
6 modification of configuration data on any computer system of the cluster, said
7 computer system comprising:

8 a) a computer memory providing for the storage of configuration data
9 including a set of identifications of computer systems participating in a
10 preestablished computer system cluster;

11 b) a processor coupled to said computer memory and operative to
12 execute a control program that defines performance of a predetermined
13 network service in response to network requests as received from host
14 computers, wherein performance of said predetermined network service is
15 controlled by said configuration data;

16 c) an administrative interface coupled to said processor, wherein said
17 control program further enables secure performance of local administrative
18 modifications to said configuration data through said administrative interface;
19 and

20 d) a communications interface coupled to said processor and
21 coupleable to said preestablished computer system cluster, wherein said
22 control program further enables secure synchronization of said configuration
23 data among said computer system and said preestablished computer system
24 cluster, said control program limiting the transfer of said configuration data
25 between said computer system, as a first computer system, and a second
26 computer system securely matching a identification preexisting in said set of
27 identifications.

- 67 -

1 80. The computer system of Claim 79 wherein said configuration data is
2 transferred encrypted between said first computer system and said second
3 computer system and wherein the encryption of said configuration data is
4 specific to said first computer system and said second computer system.

1 81. The computer system of Claim 80 wherein said processor is responsive
2 to a network synchronization message identifying a configuration data set
3 more recently modified relative to the configuration data stored in said
4 computer memory, said processor being operative to identify said second
5 computer system as the source of said network synchronization message, send
6 a network configuration data request message to said second computer
7 system. receive, and decrypt said configuration data set, and incorporate said
8 configuration data set into said computer memory.

1 82. The computer system of Claim 81 wherein said processor is further
2 operative to validate said network synchronization message relative to said set
3 of identifications, prepare said network configuration data request to be
4 validateable against said set of identifications, and validate said configuration
5 data set as received against said set of identifications, whereby the transfer of
6 said configuration data set is secure and constrained with respect to said first
7 computer system to those computer systems of said preestablished computer
8 system cluster that are preexistingly identified by said set of identifications.

1/7

FIG. 1A

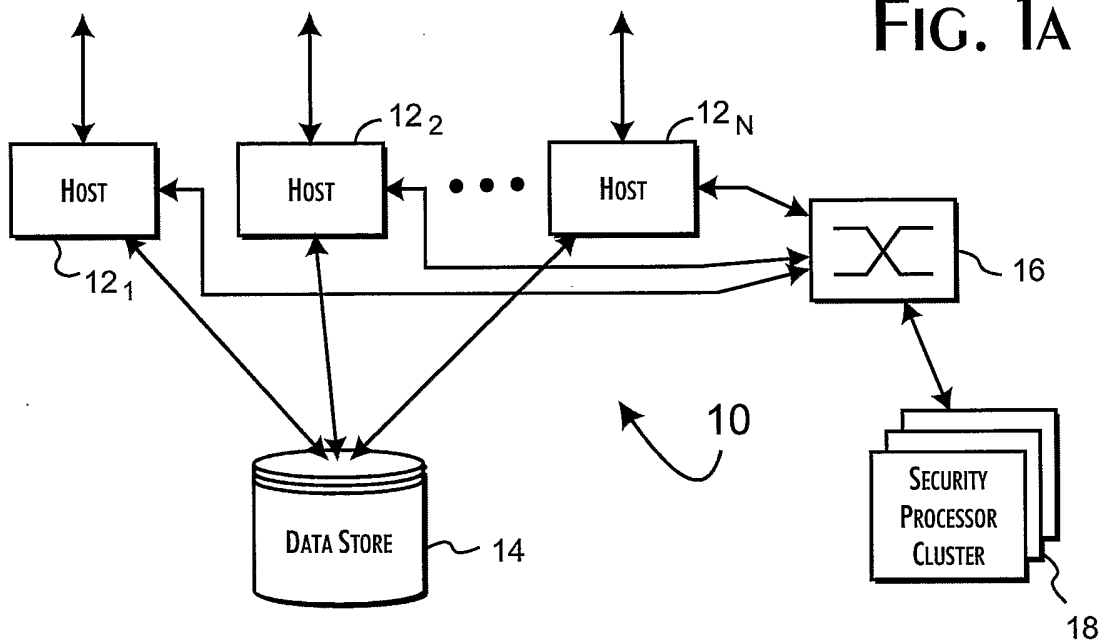
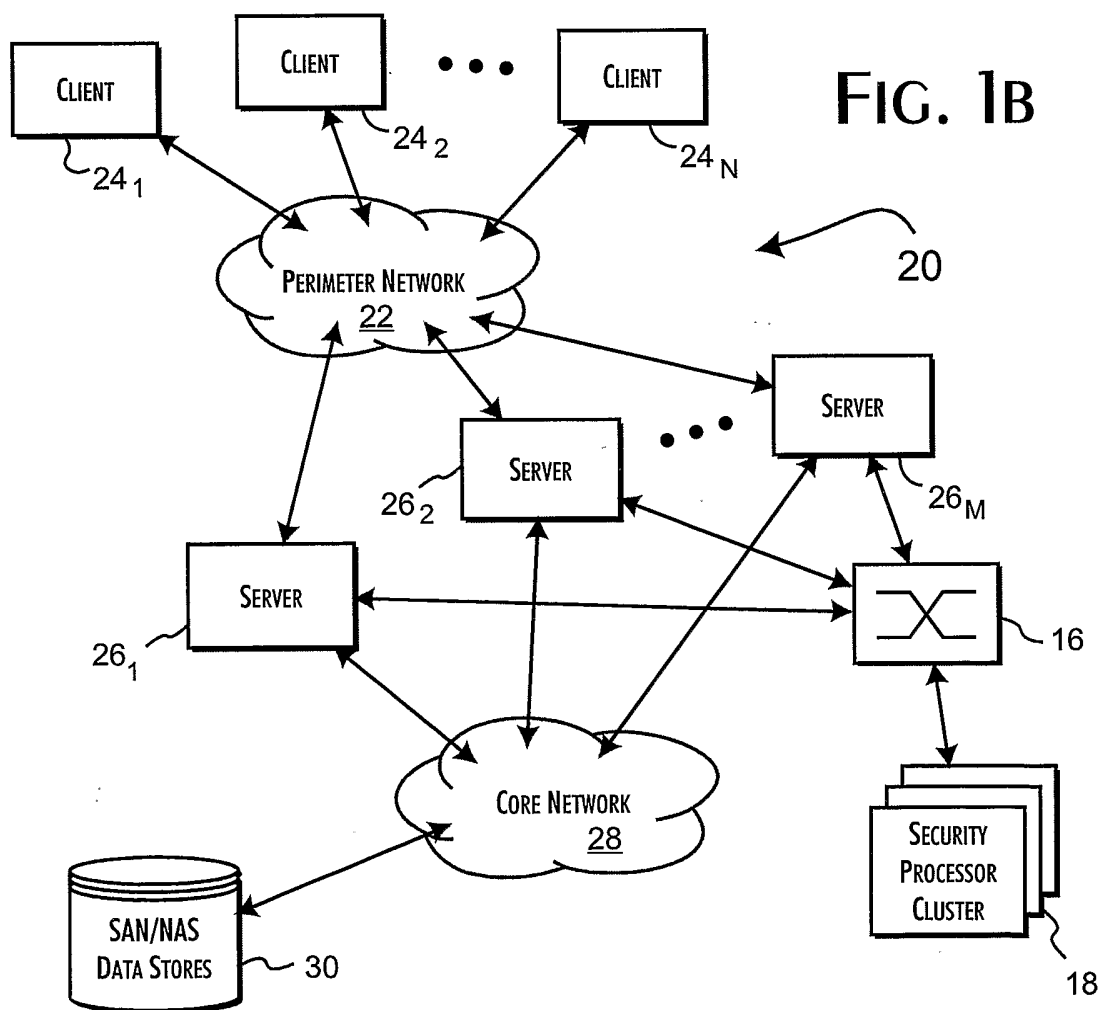
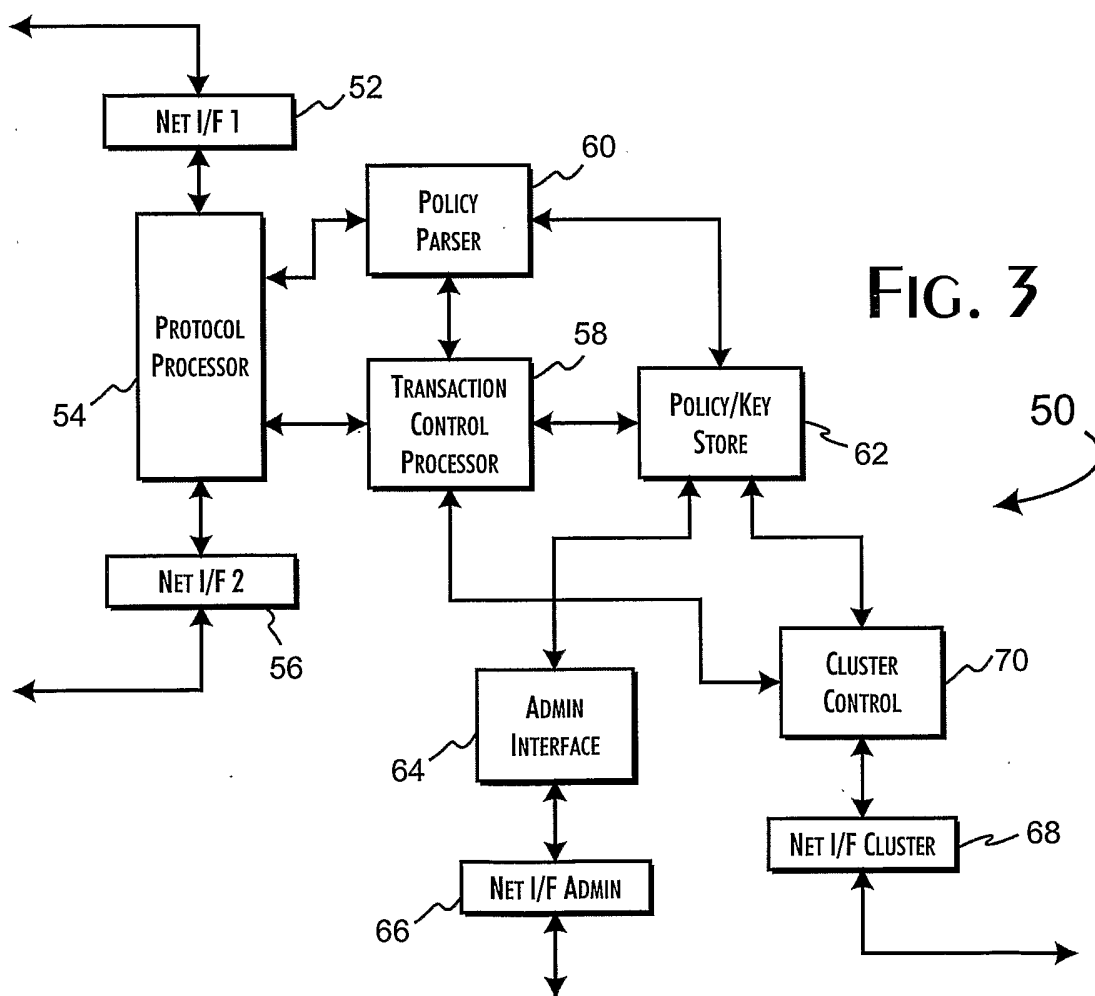
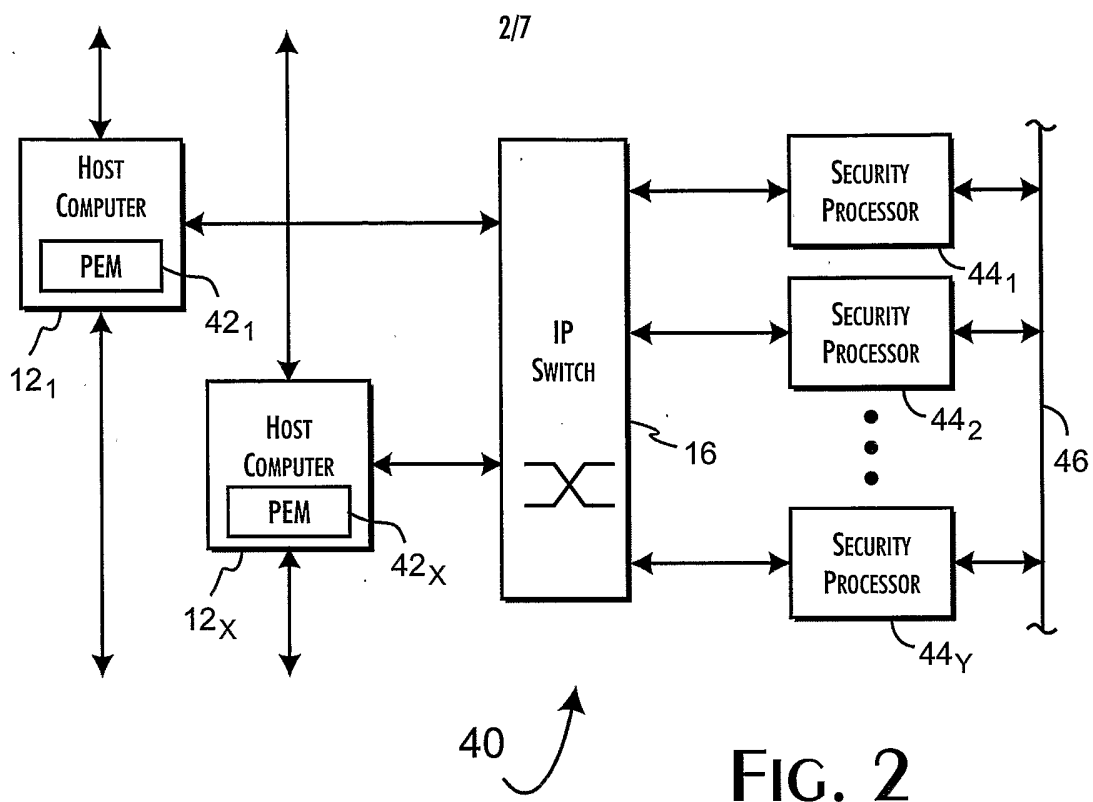


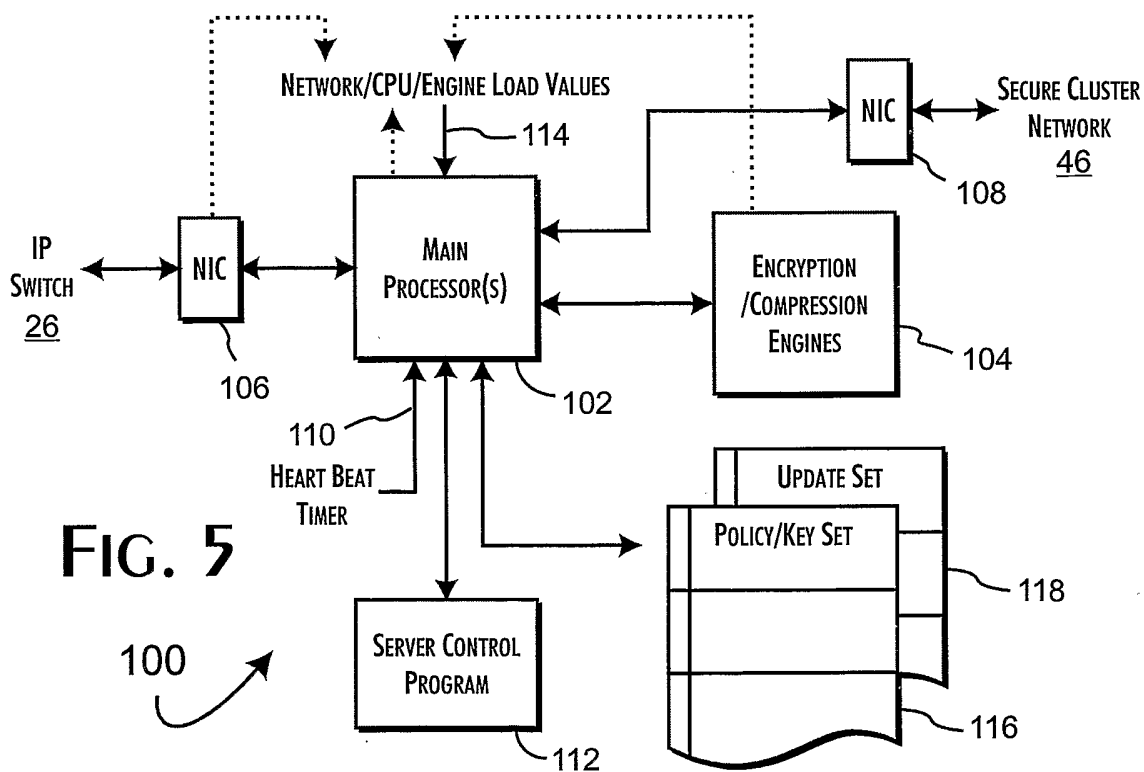
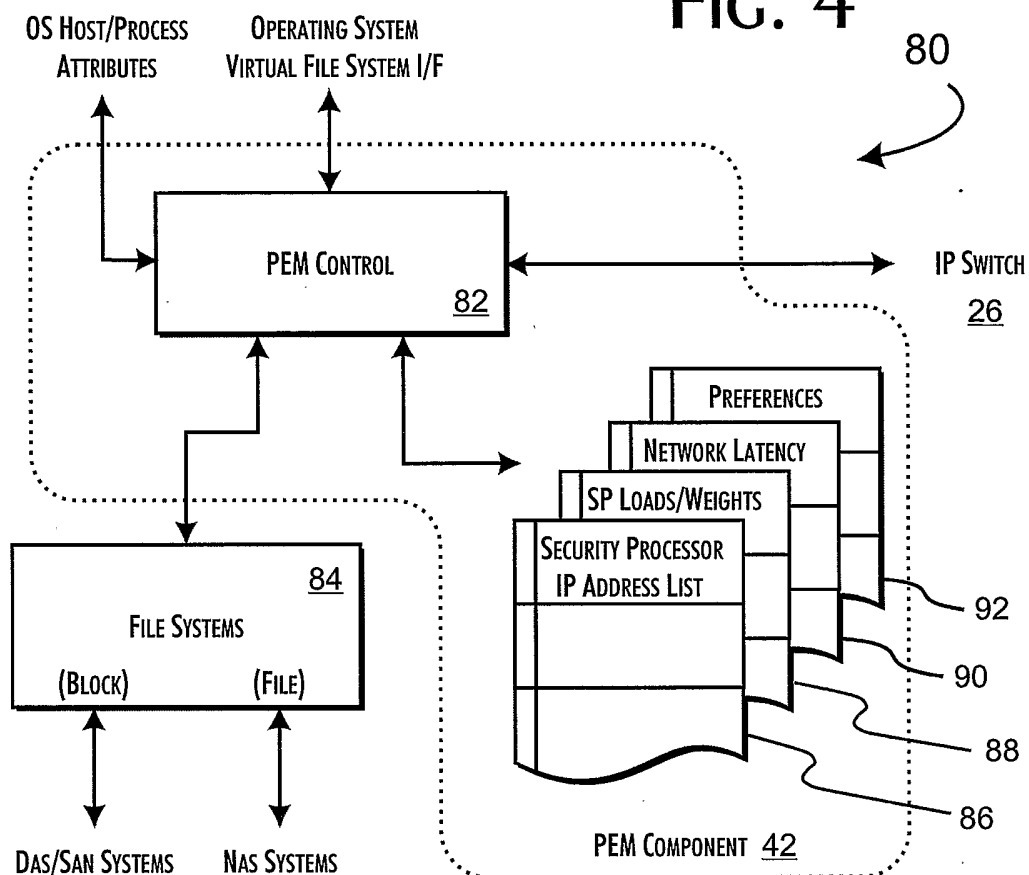
FIG. 1B



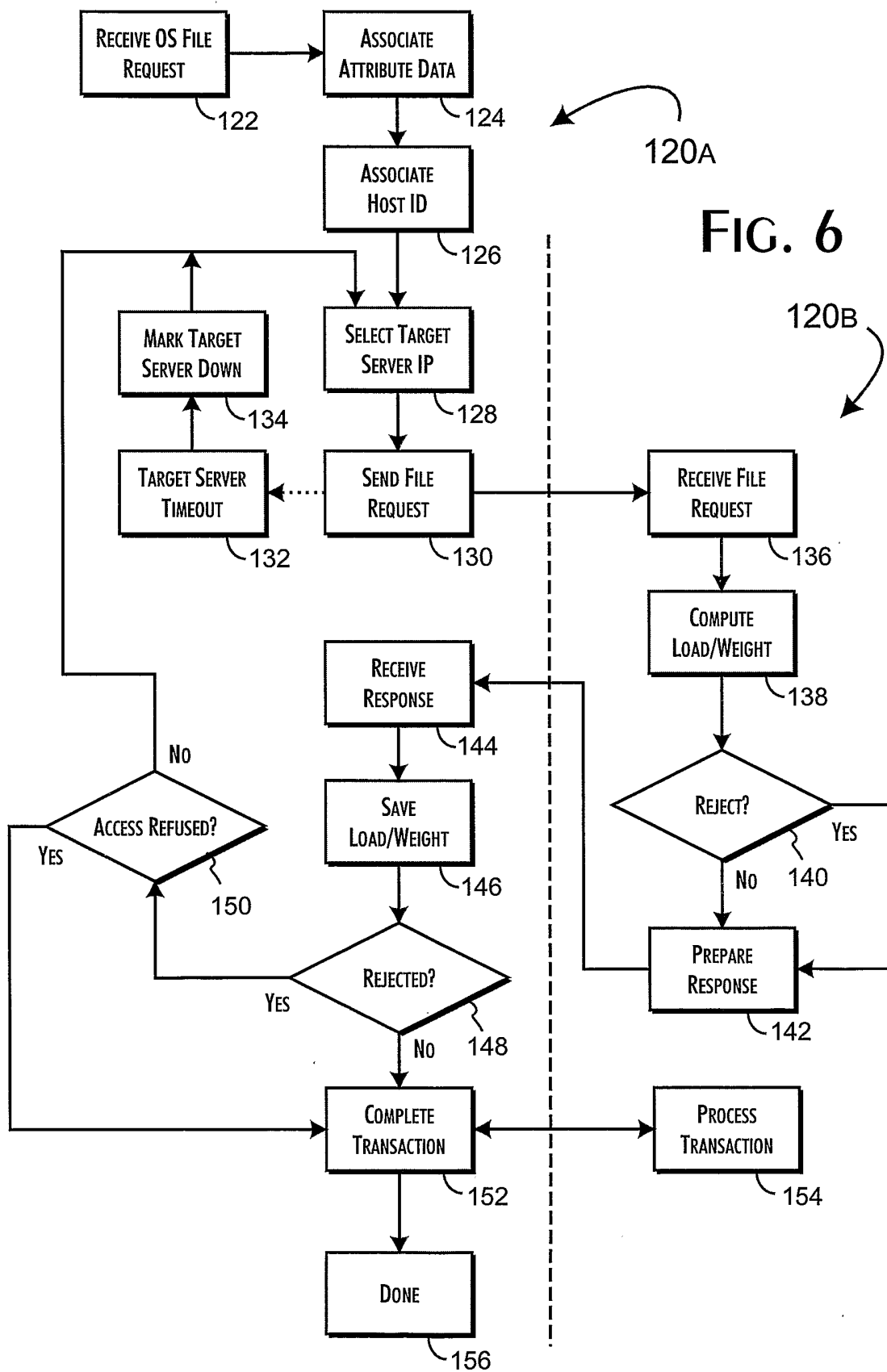


3/7

FIG. 4

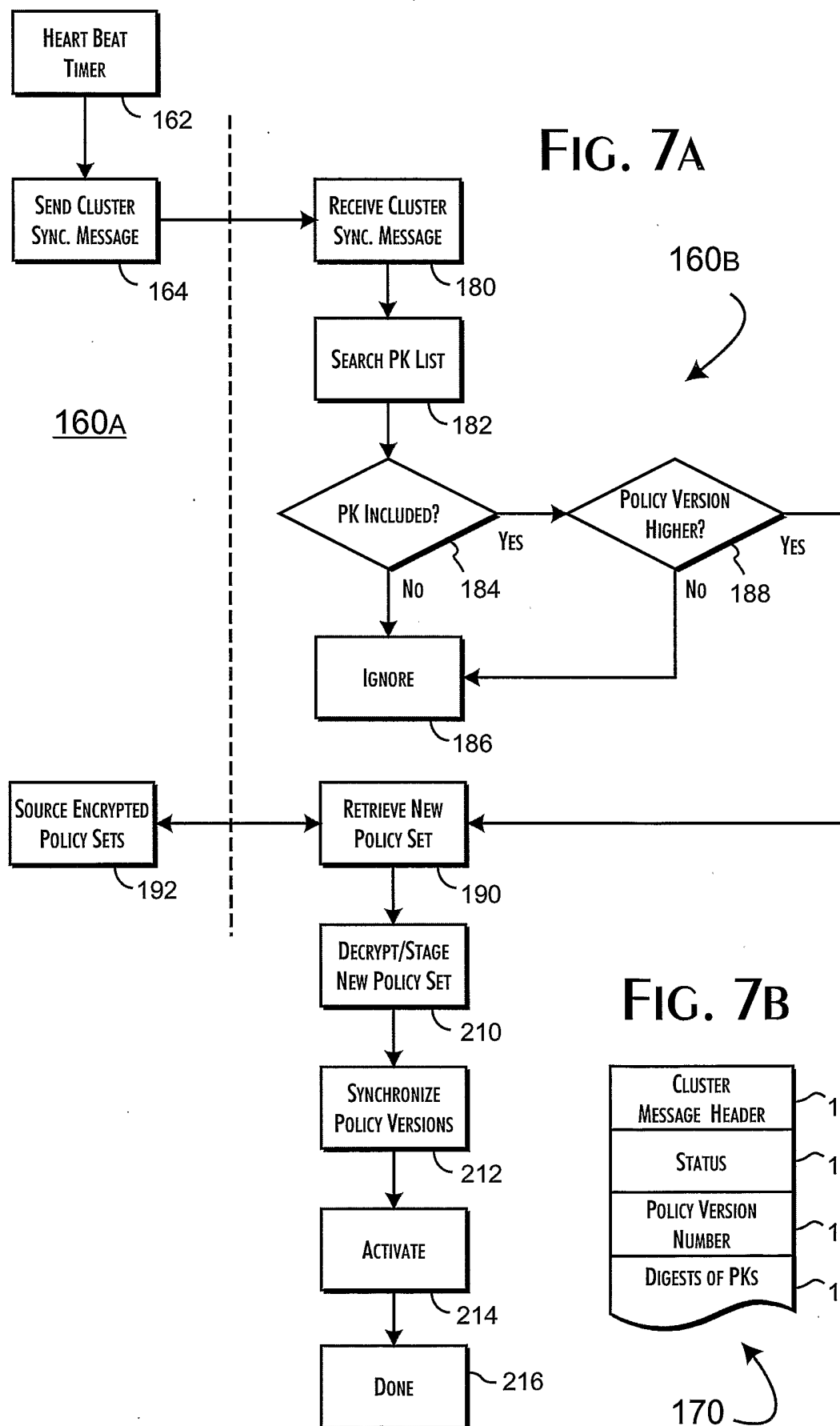


4/7



5/7

FIG. 7A



6/7

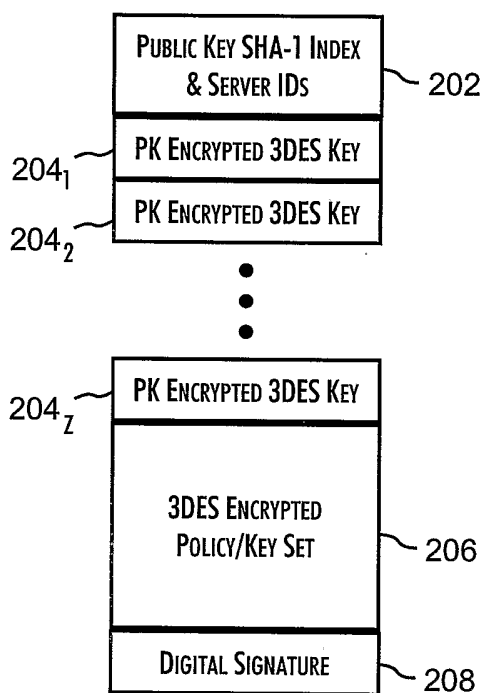


FIG. 7C

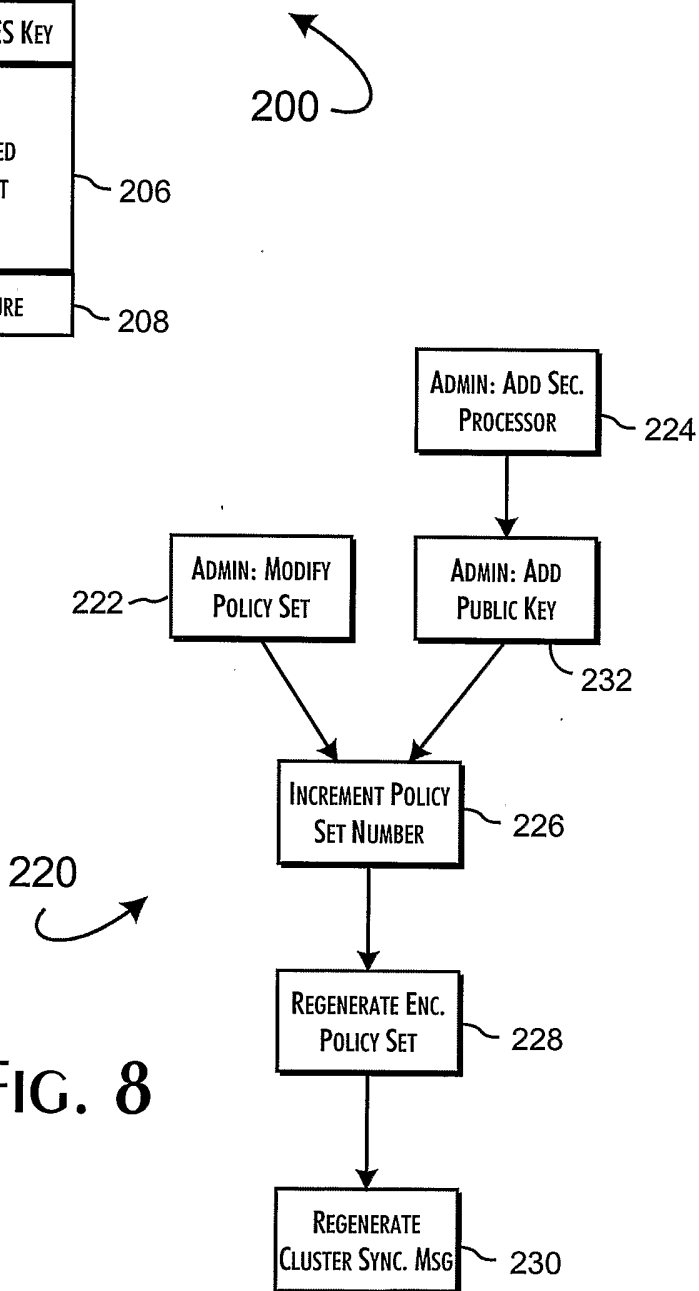


FIG. 8

7/7

FIG. 9

