EXHIBIT 11 - ORGANICK

The Multics System: An Examination of Its Structure
Author: Elliott I Organick
Publication Date: 1972
("*Organick*")

U.S. Patent No. 6,192,476 – Claim 1	
1. A method for providing security, the method comprising the steps of:	*Organick* discloses a method for providing security. Specifically, Organick discloses the "Multics" system, a sophisticated computer program with internal security control components. These components were disclosed as a means to implement "controlled sharing of data and procedures." *Organick*, 133. "**4.2 Access Control and Ring-Bracket Protection** In this section some basic details are provided on the two types of isolation techniques, access control and ring brackets,[] which, in proper combination, are fundamental to the system of protection and to the controlled sharing of data and procedures in Multics." *Organick*, 133.
detecting when a request for an action is made by a principal; and	*Organick* discloses detecting when a request for an action is made by a principal. For example, *Organick* discloses the use of a hardware device to "detect and trap a process whenever it attempts to make a cross-ring reference." *Organick*, 133. "It has already been suggested why segments within a process should be subdivided into rings and why there should be a separate stack segment for each ring. It is proper to remark here that ring compartmentalization is carried out with some hardware aid. Multics exploits special GE 645 fault-detection hardware to detect and trap a process whenever it attempts to make a cross-ring reference, in order to invoke the intervention of supervisory software." *Organick*, 133.
in response to detecting the request, determining whether said action is authorized based on permissions	*Organick* discloses in response to detecting the request, determining whether said action is authorized based on permissions associated with a plurality of routines in a calling hierarchy associated with said principal, wherein said permissions are associated with said plurality of

U.S. Patent No. 6,192,476 – Claim 1	
associated with a plurality of routines in a calling hierarchy associated with said principal, wherein said permissions are associated with said plurality of routines based on a first association between protection domains and permissions.	routines based on a first association between protection domains and permissions. *Organick* discloses the access controls along with the ring structure to allow for multi-level permission. "The segments of any one process are associated with a set of generally two, but possibly more, concentric rings." *Organick*, 130. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." The controlled entry via gates is into procedures that may reside in any one of several inner rings. " *Organick*, 130.

U.S. Patent No. 6,192,476 – Claim 3	
3. The method of claim 1, wherein:	
the calling hierarchy includes a first routine; and	*Organick* discloses the calling hierarchy includes a first routine. For example, *Organick* discloses "a separate stack segment . . . created for each ring." *Organick*, 153. This inherently includes a first segment in the hierarchy. "We are now ready to see how ring access control has been implemented in Multics. First, three important implementation concepts are amplified. (1) A process can have, if necessary, up to 64 rings; user rings are numbered 32 through 63. (2) For each ring in which a process executes there is actually a separate descriptor segment. Ring-O supervisor routines create and maintain these segments as needed.[] The per-ring descriptor segments differ only in the way fault-inducing bit patterns are placed in the descriptors. The bit patterns are set so as to trap during address formation on all inward data references and on all inward or outward procedure references. (3) There is also a separate stack segment, called <stack_n>, created for each ring in which the process executes. Here, n is one of the integers 0 through 63 (or, strictly speaking, 00, 01, . .. , 63). Supervisory routines are responsible for creating these stack segments,18 but once created they are to be treated as ordinary data segments." *Organick*, 153.

U.S. Patent No. 6,192,476 – Claim 3	
	"A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.
the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with said first routine.	*Organick* discloses the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with said first routine. For example, *Organick* discloses rings that limit the access of a procedure. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." " Organick, 130. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target

U.S. Patent No. 6,192,476 – Claim 3	
	procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.

U.S. Patent No. 6,192,476 – Claim 4	
4. The method of claim 1, wherein the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy.	*Organick* discloses a method wherein the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy. *Organick* discloses this limitation by way of its ring functionality, limiting access in a manner analogous to "rings" or levels of access, hierarchically arranged in terms of the permissions needed to access sensitive information. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.

U.S. Patent No. 6,192,476 – Claim 6	
6. A method for providing security, the method comprising the steps of:	*Organick* discloses a method for providing security. *Organick* discloses a method for providing security. Specifically, Organick discloses the "Multics" system, a sophisticated computer program with internal security control components. These components were disclosed as a means to implement "controlled sharing of data and procedures." *Organick*, 133. "**4.2 Access Control and Ring-Bracket Protection** In this section some basic details are provided on the two types of isolation techniques, access control and ring brackets,[] which, in proper combination, are fundamental to the system of protection and to the controlled sharing of data and procedures in Multics." *Organick*, 133.
detecting when a request for an action is made by a principal; and	*Organick* discloses detecting when a request for an action is made by a principal. For example, *Organick* discloses the use of a hardware device to "detect and trap a process whenever it attempts to make a cross-ring reference." *Organick*, 133. "It has already been suggested why segments within a process should be subdivided into rings and why there should be a separate stack segment for each ring. It is proper to remark here that ring compartmentalization is carried out with some hardware aid. Multics exploits special GE 645 fault-detection hardware to detect and trap a process whenever it attempts to make a cross-ring reference, in order to invoke the intervention of supervisory software." *Organick*, 133.
in response to detecting the request, determining whether said action is authorized based on permissions associated with a plurality of routines in a calling hierarchy associated with said principal, wherein a first routine in said calling hierarchy is privileged; and	*Organick* discloses in response to detecting the request, determining whether said action is authorized based on permissions associated with a plurality of routines in a calling hierarchy associated with said principal, wherein said permissions are associated with said plurality of routines based on a first association between protection domains and permissions. *Organick* discloses the access controls along with the ring structure to allow for multi-level permission. "The segments of any one process are associated with a set of generally two, but possibly more, concentric rings." *Organick*, 130. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any

U.S. Patent No. 6,192,476 – Claim 6	
	peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." The controlled entry via gates is into procedures that may reside in any one of several inner rings. " *Organick*, 130.
wherein the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy between and including said first routine and a second routine in said calling hierarchy, wherein said second routine is invoked after said first routine, wherein said second routine is a routine for performing said requested action.	*Organick* discloses wherein the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy between and including said first routine and a second routine in said calling hierarchy, wherein said second routine is invoked after said first routine, wherein said second routine is a routine for performing said requested action. *Organick* discloses a process whereby permission is granted to certain rings, wherein a first ring may be privileged and inaccessible but a second ring may be accessible. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." " Organick, 130. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket.

U.S. Patent No. 6,192,476 – Claim 6	
	(Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.

U.S. Patent No. 6,192,476 – Claim 10	
10. A computer-readable medium carrying one or more sequences of one or more instructions, the one or more sequences of the one or more instructions including instructions which, when executed by one or more processors, causes the one or more processors to perform the steps of:	*Organick* disclosed a computer-readable medium, e.g., "core memory," carrying one or more sequences of one or more instructions. "A segment of a process is a collection of information important enough to be given a name. A segment is a unit of sharing and has associated with it a collection of attributes including a unique identification. Segments are, generally speaking, blocks of code (procedures) or blocks of data ranging in size from zero to 2^{16} words.[] Each segment can be allowed to grow or shrink during execution of the process. A record of its size is kept in the "descriptor word" associated with the segment. . . . Unseen by the user, hardware mechanisms exist for subdividing a segment into smaller units called *pages*, each of which may be located in smaller discontinuous blocks of core memory." *Organick*, 5.
detecting when a request for an action is made by a principal; and	*Organick* discloses detecting when a request for an action is made by a principal. For example, *Organick* discloses the use of a hardware device to "detect and trap a process whenever it attempts to make a cross-ring reference." *Organick*, 133. "It has already been suggested why segments within a process should be subdivided into rings and why there should be a separate stack segment for each ring. It is proper to remark here that ring compartmentalization is carried out with some hardware aid. Multics exploits special GE 645 fault-detection hardware to detect and trap a process whenever it attempts to make a cross-ring reference, in order to invoke the intervention of supervisory software." *Organick*, 133.
in response to detecting the request, determining whether said action is authorized based on permissions	*Organick* discloses in response to detecting the request, determining whether said action is authorized based on permissions associated with a plurality of routines in a calling hierarchy associated with said principal, wherein said permissions are associated with said plurality of

U.S. Patent No. 6,192,476 – Claim 10	
associated with a plurality of routines in a calling hierarchy associated with said principal, wherein said permissions are associated with said plurality of routines based on a first association between protection domains and permissions.	routines based on a first association between protection domains and permissions. *Organick* discloses the access controls along with the ring structure to allow for multi-level permission. "The segments of any one process are associated with a set of generally two, but possibly more, concentric rings." *Organick*, 130. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." The controlled entry via gates is into procedures that may reside in any one of several inner rings. " *Organick*, 130.

U.S. Patent No. 6,192,476 – Claim 12	
12. The computer readable medium of claim 10, wherein:	
the calling hierarchy includes a first routine; and	*Organick* discloses the calling hierarchy includes a first routine. For example, *Organick* discloses "a separate stack segment . . . created for each ring." *Organick*, 153. This inherently includes a first segment in the hierarchy. "We are now ready to see how ring access control has been implemented in Multics. First, three important implementation concepts are amplified. (1) A process can have, if necessary, up to 64 rings; user rings are numbered 32 through 63. (2) For each ring in which a process executes there is actually a separate descriptor segment. Ring-O supervisor routines create and maintain these segments as needed.[] The per-ring descriptor segments differ only in the way fault-inducing bit patterns are placed in the descriptors. The bit patterns are set so as to trap during address formation on all inward data references and on all inward or outward procedure references. (3) There is also a separate stack segment, called <stack_n>, created for each ring in which the process executes. Here, n is one of the integers 0 through 63 (or, strictly speaking, 00, 01, . .. , 63). Supervisory routines are responsible for creating these stack segments,18 but once created they are to be treated as ordinary data segments."

U.S. Patent No. 6,192,476 – Claim 12	
	Organick, 153. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)." *Organick*, 157–58.
the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with said first routine.	*Organick* discloses the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with said first routine. For example, *Organick* describes three situations, explained below, that show the multi-leveled protection scheme in place. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." *Organick*, 130. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;.

U.S. Patent No. 6,192,476 – Claim 12	
	r <;'1. (The ring of the calling procedure lies within the access bracket of the target procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.

U.S. Patent No. 6,192,476 – Claim 13	
13. The computer readable medium of claim 10, wherein the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy.	*Organick* discloses a computer readable medium wherein the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy. For example, *Organick* describes three situations, explained below, that show the multi-leveled protection scheme in place. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.

U.S. Patent No. 6,192,476 – Claim 15	

U.S. Patent No. 6,192,476 – Claim 15	
15. A computer-readable medium carrying one or more sequences of one or more instructions, the one or more sequences of the one or more instructions including instructions which, when executed by one or more processors, causes the one or more processors to perform the steps of:	*Organick* disclosed a computer-readable medium carrying one or more sequences of one or more instructions. *Organick* discloses a method for providing security. Specifically, Organick discloses the "Multics" system, a sophisticated computer program with internal security control components. These components were disclosed as a means to implement "controlled sharing of data and procedures." *Organick*, 133. "A segment of a process is a collection of information important enough to be given a name. A segment is a unit of sharing and has associated with it a collection of attributes including a unique identification. Segments are, generally speaking, blocks of code (procedures) or blocks of data ranging in size from zero to 2^{16} words.[] Each segment can be allowed to grow or shrink during execution of the process. A record of its size is kept in the "descriptor word" associated with the segment. . . . Unseen by the user, hardware mechanisms exist for subdividing a segment into smaller units called *pages*, each of which may be located in smaller discontinuous blocks of core memory." *Organick*, 5.
detecting when a request for an action is made by a principal; and	*Organick* discloses detecting when a request for an action is made by a principal. For example, *Organick* discloses the use of a hardware device to "detect and trap a process whenever it attempts to make a cross-ring reference." *Organick*, 133. "It has already been suggested why segments within a process should be subdivided into rings and why there should be a separate stack segment for each ring. It is proper to remark here that ring compartmentalization is carried out with some hardware aid. Multics exploits special GE 645 fault-detection hardware to detect and trap a process whenever it attempts to make a cross-ring reference, in order to invoke the intervention of supervisory software." *Organick*, 133.
in response to detecting the request, determining whether said action is authorized based on permissions associated with a plurality of routines in	*Organick* discloses in response to detecting the request, determining whether said action is authorized based on permissions associated with a plurality of routines in a calling hierarchy associated with said principal, wherein a first routine in said calling hierarchy is privileged. *Organick* discloses the access controls along with the ring structure to allow for multi-level

U.S. Patent No. 6,192,476 – Claim 15	
a calling hierarchy associated with said principal, wherein a first routine in said calling hierarchy is privileged; and	permission. "The segments of any one process are associated with a set of generally two, but possibly more, concentric rings." *Organick*, 130. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." The controlled entry via gates is into procedures that may reside in any one of several inner rings. " *Organick*, 130.
wherein the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy between and including said first routine and a second routine in said calling hierarchy, wherein said second routine is invoked after said first routine, wherein said second routine is a routine for performing said requested action.	*Organick* discloses wherein the step of determining whether said action is authorized further includes determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy between and including said first routine and a second routine in said calling hierarchy, wherein said second routine is invoked after said first routine, wherein said second routine is a routine for performing said requested action. *Organick* discloses a process whereby permission is granted to certain rings, wherein a first ring may be privileged and inaccessible but a second ring may be accessible. For example, *Organick* describes three situations, explained below, that show the multi-leveled protection scheme in place. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." " Organick, 130. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target

U.S. Patent No. 6,192,476 – Claim 15	
	procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.

U.S. Patent No. 6,192,476 – Claim 19	
19. A computer system comprising:	
a processor;	*Organick* discloses a processor, i.e., "a computer processing unit." "1.2.1 Processor A computer processing unit as found on any familiar computer." *Organick*, 5.
a memory coupled to said processor;	*Organick* disclosed a memory, e.g., "core memory," coupled to said processor. "A segment of a process is a collection of information important enough to be given a name. A segment is a unit of sharing and has associated with it a collection of attributes including a unique identification. Segments are, generally speaking, blocks of code (procedures) or blocks of data ranging in size from zero to 2^{16} words.[] Each segment can be allowed to grow or shrink during execution of the process. A record of its size is kept in the "descriptor word" associated with the segment. . . . Unseen by the user, hardware mechanisms exist for subdividing a segment into smaller units called *pages*, each of which may be located in smaller discontinuous blocks of core memory." *Organick*, 5.
said processor being configured to detect when a request for an action is	*Organick* discloses a processor being configured to detect when a request for an action is made by a principal. For example, *Organick* discloses the use of a hardware device to

U.S. Patent No. 6,192,476 – Claim 19	
made by a principal; and	"detect and trap a process whenever it attempts to make a cross-ring reference." *Organick*, 133. "It has already been suggested why segments within a process should be subdivided into rings and why there should be a separate stack segment for each ring. It is proper to remark here that ring compartmentalization is carried out with some hardware aid. Multics exploits special GE 645 fault-detection hardware to detect and trap a process whenever it attempts to make a cross-ring reference, in order to invoke the intervention of supervisory software." *Organick*, 133.
said processor being configured to respond to detecting the request by determining whether said action is authorized based on permissions associated with a plurality of routines in a calling hierarchy associated with said principal, wherein said permissions are associated with said plurality of routines based on a first association between protection domains and permissions.	*Organick* discloses a processor being configured to respond to detecting the request by determining whether said action is authorized based on permissions associated with a plurality of routines in a calling hierarchy associated with said principal, wherein said permissions are associated with said plurality of routines based on a first association between protection domains and permissions. *Organick* discloses the access controls along with the ring structure to allow for multi-level permission. "The segments of any one process are associated with a set of generally two, but possibly more, concentric rings." *Organick*, 130. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." The controlled entry via gates is into procedures that may reside in any one of several inner rings. " *Organick*, 130.

U.S. Patent No. 6,192,476 – Claim 20	
20. The computer system of claim 19, wherein:	
the calling hierarchy includes a first routine; and	*Organick* discloses the calling hierarchy includes a first routine. For example, *Organick* discloses "a separate stack segment . . . created for each ring." *Organick*, 153. This

U.S. Patent No. 6,192,476 – Claim 20	
	inherently includes a first segment in the hierarchy. "We are now ready to see how ring access control has been implemented in Multics. First, three important implementation concepts are amplified. (1) A process can have, if necessary, up to 64 rings; user rings are numbered 32 through 63. (2) For each ring in which a process executes there is actually a separate descriptor segment. Ring-O supervisor routines create and maintain these segments as needed.[] The per-ring descriptor segments differ only in the way fault-inducing bit patterns are placed in the descriptors. The bit patterns are set so as to trap during address formation on all inward data references and on all inward or outward procedure references. (3) There is also a separate stack segment, called <stack_n>, created for each ring in which the process executes. Here, n is one of the integers 0 through 63 (or, strictly speaking, 00, 01, . .. , 63). Supervisory routines are responsible for creating these stack segments,18 but once created they are to be treated as ordinary data segments." *Organick*, 153. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.
said processor is configured to determine whether said action is authorized by determining whether a	*Organick* discloses a processor that is configured to determine whether said action is authorized by determining whether a permission required to perform said action is encompassed by at least one permission associated with said first routine. For example,

U.S. Patent No. 6,192,476 – Claim 20	
permission required to perform said action is encompassed by at least one permission associated with said first routine.	*Organick* discloses that "a procedure that is assigned the category of ring *r* [such that] a procedure of ring *r* is prevented from referencing data segments in a more "privileged," that is, "inner" ring and is permitted call access to more privileged procedures only through specially controlled entry points called "gates." Organick at 130. "Basically, a procedure that is assigned the category of ring r is privileged during its execution to call (or to reference) any procedure (or data) segment in ring r or in any peripheral to, that is "outside of," ring r. Conversely, a procedure of ring r is prevented from referencing data segments in a more "privileged," that is "inner," ring and is permitted call access to more privileged procedure only through specially controlled entry points called "gates." " Organick, 130. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.

U.S. Patent No. 6,192,476 – Claim 21	
21. The computer system of claim 19, wherein	
said processor is configured to determine whether said action is	*Organick* discloses a processor that is configured to determine whether said action is authorized by determining whether a permission required to perform said action is

U.S. Patent No. 6,192,476 – Claim 21	
authorized by determining whether a permission required to perform said action is encompassed by at least one permission associated with each routine in said calling hierarchy.	encompassed by at least one permission associated with each routine in said calling hierarchy. For example, *Organick* describes three situations, explained below, that show the multi-leveled protection scheme in place. "A good question to ask is, In which of the rings within a segment's access bracket will a particular segment execute when it is called? There are three cases to be considered. Assume that <a> is the calling procedure now executing in ring r, and that <b> is to be the called or target procedure whose ring bracket is (k, I, m) such that 0 <;. k < I < m <;. 63. Case 1. k <;. r <;'1. (The ring of the calling procedure lies within the access bracket of the target procedure.) Procedure <b> will execute in ring r. No ring-crossing fault will be triggered. Case 2. r < k. (Outward fault. The ring of the faulting procedure is less than k.) Procedure <b> will execute in ring k, the innermost ring of the target's access bracket. The design rationale for this choice is necessarily arbitrary: Pick the ring "nearest to the caller." Case 3. 1< r <;. m. (Inward fault. The ring of the faulting procedure lies within the call bracket of <b>.) Procedure <b> will execute in ring I, the outermost ring of the target's access bracket. (Of course, the desired entry point must also be found to have the format of a gate.)" *Organick*, 157–58.