<u>Exhibit 9</u>

<u>The Gateway Security Model in the Java Electronic Commerce Framework</u>
Theodore Goldstein
Chief Java Commerce Officer, Sun Microsystems Laboratories / JavaSoft
November 29, 1996
("*Goldstein*")

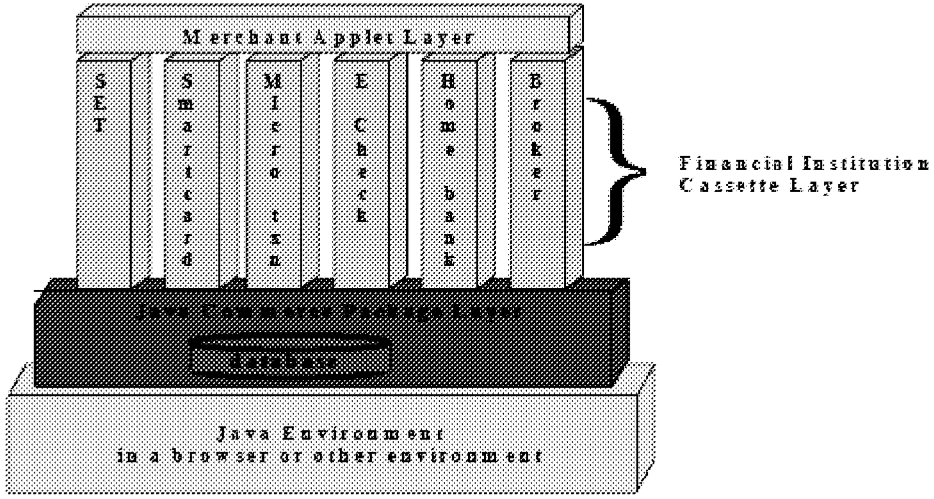
<u>Java APIs: Playing Monopoly with Java via the JECF</u>
Rawn Shah
JavaWorld.com
December 1, 1996
("*Shah*")

U.S. Patent No. 6,125,447 – Claim 1	***Goldstein in view of Shah***
1. A method for providing security, the method comprising the steps of:	The *Goldstein* reference discloses a method for providing security. For example, *Goldstein* discloses the "*Gateway*" security model for use in the Java Electronic Commerce Framework ("JECF"). *Goldstein* discloses that the *Gateway* security model is "an extension to the current Java security model": "This paper describes an extension to the current Java security model called the 'Gateway' and why it was necessary to create it. This model allows secure applications, such as those used in electronic commerce, to safely exchange data and interoperate without compromising each individual application's security. The Gateway uses digital signatures to enable application programming interfaces to authenticate their caller. JavaSoft is using the Gateway to create a new integrated open platform for financial applications called Java Electronic Commerce Framework. The JECF will be the foundation for electronic wallets, point of sale terminals, electronic merchant servers and other financial software. The Gateway model can also be used for access control in many multiple application environments that require trusted interaction between applications from multiple vendors. These applications include browsers, servers, operating systems,

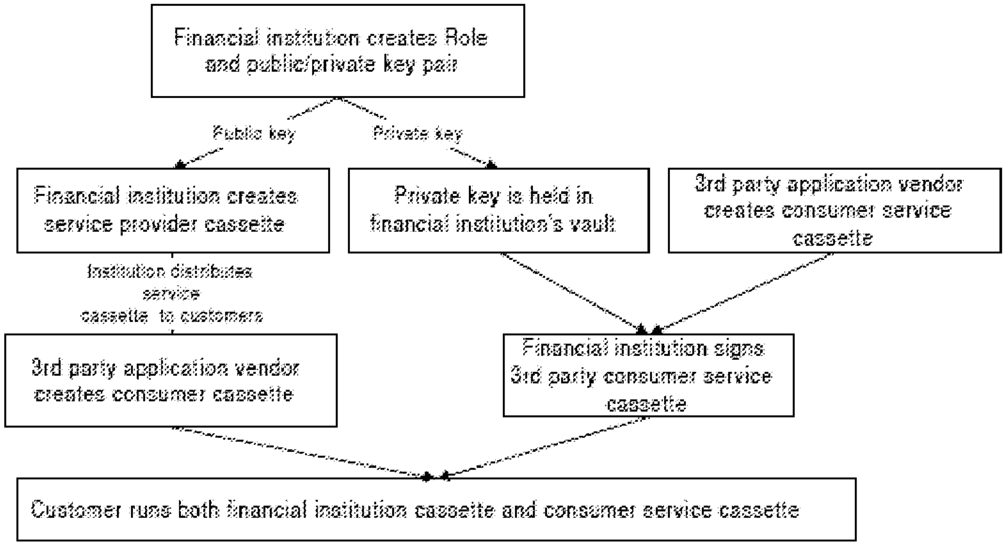
U.S. Patent No. 6,125,447 – Claim 1	***Goldstein in view of Shah***
	medical systems and smartcards." *Goldstein* at 1. "The limitations of Java motivate a new security model called the *Gateway* that extends the current Java security model. This new model is the core of an entire application framework for financial applications called the *Java Electronic Commerce Framework* (JECF)." *Goldstein* at 1. "The JECF provides a complementary model to the Sandbox security model called the *Gateway* security model. The Gateway security model provides the means to implement contractual trust relationships. This model uses the safety features of the Java language as well as the infrastructure of the Java Sandbox security model. It also provides the ability for developers to create arbitrary trust relationships." *Goldstein* at 9.
establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions;	The *Goldstein* reference discloses "establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions." The '447 Patent discloses that "A protection domain can be viewed as a set of permissions granted to one or more principals." '447 Patent, 8:42-43. ("A 'principal' is an entity in the computer system to which permissions are granted. Examples of principals include processes, objects and threads." '447 Patent, 2:25-27.) As discussed more thoroughly below, *Goldstein* discloses establishing a set of permissions that are granted to an application called a "cassette." A set of permissions (protection domain) is first established by a financial institution or third party company at the time a cassette is created. (*Goldstein* at 7-8, 10, 13-14 & Figs. 1, 5-6.) These permissions are represented by "Roles" that "reify [i.e., implement] the trust relationship between two business entities." (*Goldstein* at 13.) The Roles are objects that represent and/or contain the specific authorizations (e.g., permissions) for a particular cassette as well as a digital signature (based on a public/private key pair) corresponding to the creator of the cassette. (*Id.*) After the cassette application is created, it is distributed to end users. (*Goldstein* at 14.) A user installs the cassette on a computer, and subsequently the cassette (including the associated Roles objects that correspond to the set of permissions for the cassette) is loaded

U.S. Patent No. 6,125,447 – Claim 1	*Goldstein in view of Shah*
	by a Java applet class named “CassetteLoader” which communicates with a “CassetteInstallation” object to facilitate the load operation. (*Id.*) Accordingly, establishing the protection domain, or set of rights associated with the cassette, may be thought to occur at any of the following times, as disclosed in *Goldstein*: (1) by the financial institution when it defines the Roles or permissions for an institution cassette, (2) by a third party company when it defines the Roles or permissions for a third-party cassette, or (3) by the CassetteLoader applet when it loads a given cassette (including loading and creating the corresponding Roles) in the run-time environment. Further details of the above-described system are provided below. *Goldstein* discloses establishing a protection domain as a set of permissions (e.g., “rights”) that are used by a client application (e.g., a principal or “body of code”): “To completely understand what is occurring, it is necessary to define a few terms: *rights* and *principals*. In this paper, a *right* is an abstract privilege. A *principal* uses a right. A principal can be a person, a corporation, a program, or a body of code.” *Goldstein* at 10. “[The JECF] allows secure applications, such as those used in electronic commerce, to safely exchange data and interoperate without compromising each individual application’s security.” *Goldstein* at 1. The rights in the JECF are based on the Capabilities model, which provides for the association of rights with object-oriented structures so that rights can be transferred from one principal to another: “Electronic commerce needs an object-oriented security model where rights can be transferred from one principal to another. The *Capabilities model,* originally defined by Dennis and Van Horn, provides such a model. Although originally based on hardware protection, Wulf and a team at Carnegie-Mellon University[WCCJRPP74] adapted Capabilities to object-oriented technology. The object-oriented definition of Capabilities is very simple. The Capabilities model means that possession of an object confers the right to

U.S. Patent No. 6,125,447 – Claim 1	***Goldstein in view of Shah***
	use it. Capabilities need a gatekeeper service to decide whether to grant a right to the object. * * * Adding the Capabilities model to Java has proven to be simple because Java already has many of the necessary features. Java objects are unforgeable. Java's language visibility rules for private and package-private scope provide address space-like protection against unauthorized access. The only needed component is some form of authorization mechanism to allow or deny access. In the JECF, individual Capabilities are called *permits*. Permit objects require careful handling. Permit objects should never be stored in a public or protected variable. Permits are rarely the actual implementation of some right, but are typically implemented by another object. The permit forwards the call to the actual implementation." *Goldstein* at 10-11. *Goldstein* discloses client applications as "cassettes" that may reside on a user's system: "Many object-oriented practitioners use the term 'framework' to mean an 'application programming interface.' In this paper, the term 'framework' refers to a set of APIs that impose responsibilities amongst participating software packages. The JECF defines application responsibilities for merchants and financial institutions. JECF provides services, such as database services, to merchant and financial institution applications. These layers are depicted in Figure 1.

U.S. Patent No. 6,125,447 – Claim 1	*Goldstein in view of Shah*
	 Merchant Applet Layer SET Smartcard Micro txn E Check Home bank Broker Financial Institution Cassette Layer Java Commerce Package Layer database Java Environment in a browser or other environment **Figure 1. Simple Schema of JECF** * * * The *Cassette Layer* implements long term customer relationships such as credit cards, home banking and brokerages. Cassettes are a new feature that JECF adds to Java. ***Similar to applets, cassettes are downloaded from servers to client computers. Unlike applets which disappear when users quit the browser, cassettes are retained on the customer's system***. Cassettes store information in a database provided by the JECF. Cassettes may safely store valuable information such as public key certificates and transaction records since the entire database is encrypted. Cassettes provide long term customer-to-institution relationships. Examples of cassettes include SET certificates and protocols, home banking, brokerage accounts, financial analysis, and planning software. Cassettes contain code, digital certificates, GIF images and other resources. Financial institutions will use cassettes to deliver customer service features. Smartcard application developers can put smartcard reader device drivers and application user interfaces in cassettes." *Goldstein* at 7-8 (emphasis added). *Goldstein* discloses examples of the types of rights (permissions) that may be associated

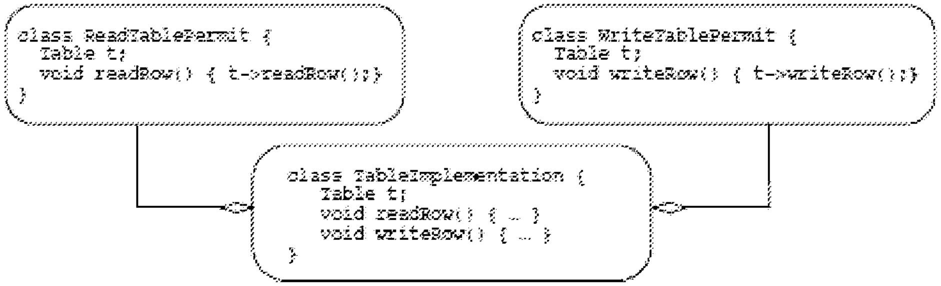
U.S. Patent No. 6,125,447 – Claim 1	*Goldstein in view of Shah*
	with a cassette application: “Potentially dangerous operations include opening a file or a network socket, changing a system variable, and setting the security manager.” *Goldstein* at 7. *Goldstein* discloses the use of objects in the “Roles” class that correspond to the protection domain, or set of rights, in a given cassette: “The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant.” *Goldstein* at 13. To the extent *Goldstein* fails to explicitly disclose the exact contents of Roles, a person of ordinary skill in the art would have understood that the Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application. For example, *Shah*, which relates to the subject matter of Goldstein provides: “When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program’s interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator.” *Shah* at 2. *Goldstein* discloses that cassettes, and the Roles associated with an individual cassette, may be created by the financial institution (in the case of a cassette provided by the institution) or by a third party company. As described in the *Shah*, *Goldstein* discloses that the Roles and

U.S. Patent No. 6,125,447 – Claim 1	***Goldstein in view of Shah***
	a digital signature (based on a public/private key pair) are created and stored in a cassette that is eventually distributed to the end user: “Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role’s public key is embedded in the home banking, brokerage software or payment cassette. The financial institution’s cassette is then distributed over the Internet, via floppy disks, etc. When third party companies want to gain access to the financial institution’s cassette, they first sign a contract with the financial institution. The institution will typically impose certain usage rules about the data and Capabilities of the cassette that are specified in the contract. The institution then signs the third party cassette which is then distributed over the Internet, via floppy disks, or any other means desired. This process of code signing is illustrated in Figure 6.”  Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes**

U.S. Patent No. 6,125,447 – Claim 1	***Goldstein in view of Shah***
	Goldstein at 13-14. In the distributed cassette application, *Goldstein* discloses that "an extensible set of rights" (e.g., a protection domain) may be established on a per-package basis or, "in a future version of Java, another security domain may emerge": "Electronic commerce requires an extensible set of rights. Electronic commerce needs to identify a principal more precisely than the applet thread. Possible candidates include *threads* and *packages*. The Sandbox security manager can only control the rights invented by the Java system developers on a per thread basis. Electronic commerce also needs the ability to delegate rights from one principal to another as long as the delegation follows the contract-specified business. This cannot be done by threads alone. In addition to threads, the Java language provides a construct called a *package*. Packages provide namespaces in the Java language. All classes in a package have access to package-private data members and methods. Package trees are primarily an organizational convention. Packages within a package tree have no special access. ***Packages are a natural choice for creating a security principal. Perhaps in a future version of Java, another security domain may emerge. Packages are a natural source of protection***." *Goldstein* at 10 (emphasis added). *Goldstein* discloses that a user may install the cassettes on a computer using the CassetteLocator applet in conjunction with the CassetteInstallation object in the JECF. At this time, the protection domain is established in the run-time environment: "Installation of cassettes is initiated by a Java applet class named CassetteLocator. CassetteLocator applets are dynamically loaded from financial institutions and other software suppliers. CassetteLocators communicate with a CassetteInstallation object in the JECF to negotiate loading of cassettes." *Goldstein* at 14.
establishing an association between said one or more protection domains and one	The *Goldstein* reference discloses "establishing an association between said one or more protection domains and one or more classes of one or more objects."

U.S. Patent No. 6,125,447 – Claim 1	*Goldstein in view of Shah*
or more classes of one or more objects; and	For example, as discussed above, *Goldstein* discloses that the protection domain (e.g., Roles or permissions) is defined on a per-package basis. Thus, any class that is part of the package (e.g., a cassette package) is necessarily associated with the protection domain: "Electronic commerce also needs the ability to delegate rights from one principal to another as long as the delegation follows the contract-specified business. This cannot be done by threads alone. In addition to threads, the Java language provides a construct called a *package*. Packages provide namespaces in the Java language. ***All classes in a package have access to package-private data members and methods.*** Package trees are primarily an organizational convention. Packages within a package tree have no special access. ***Packages are a natural choice for creating a security principal.*** Perhaps in a future version of Java, another security domain may emerge. Packages are a natural source of protection." *Goldstein* at 10 (emphasis added). *Goldstein* discloses additional methods of establishing associations between the protection domain and one or more classes of one or more objects. For example, *Goldstein* discloses establishing an association between the protection domain of a user's cassette with a class corresponding to a protected resource such as a JECF database: "Adding the Capabilities model to Java has proven to be simple because Java already has many of the necessary features. Java objects are unforgeable. Java's language visibility rules for private and package-private scope provide address space-like protection against unauthorized access. The only needed component is some form of authorization mechanism to allow or deny access. In the JECF, individual Capabilities are called *permits.* Permit objects require careful handling. Permit objects should never be stored in a public or protected variable. Permits are rarely the actual implementation of some right, but are typically implemented by another object. The permit forwards the call to the actual implementation. For example, in the JECF database, the ability to read a row in a table and the ability to write a row in the table are both implemented in a private class named

U.S. Patent No. 6,125,447 – Claim 1	***Goldstein in view of Shah***

	TableImplemenation. Access to the implementation for TableImplemenation from outside the implementation is granted by a pair of objects named ReadTablePermit and WriteTablePermit. Figure 4 illustrates the relationship between permits and the permit's underlying implementation in the Java language in the Java language."
	
	class ReadTablePermit {
	  Table t;
	  void readRow() { t->readRow();}
	}
	class WriteTablePermit {
	  Table t;
	  void writeRow() { t->writeRow();}
	}
	class TableImplementation {
	  Table t;
	  void readRow() { .. }
	  void writeRow() { .. }
	}
	Figure 4. Relationship of permit objects to implementation object
	Goldstein at 11-12.
	
	Goldstein discloses that a "Gate" is used to establish an association between the protection domain of the cassette application and the class corresponding to the protected resource (e.g., JECF database class). The Gate does this by sending the Permit to the client code:
	
	"Permits are Capabilities implemented with Java objects." *Goldstein* at 15.
	
	"A permit is a delegater pattern (as described in *Design Patterns Methodology*) to the actual implementation object. Both the permits and the implementation are in the same package. The implementation however will have only package private constructors. The permit objects need to be obtained by a client package to be useful.
	
	Gaining access to a permit is done via a pattern called a *Gate*. A Gate is a specialized form of *factory* pattern method. A Gate decides whether to grant an instance of a Permit to a caller or deny access by throwing an exception. The JECF uses the Java Sandbox digital signature mechanism for authentication. The general form of a gate

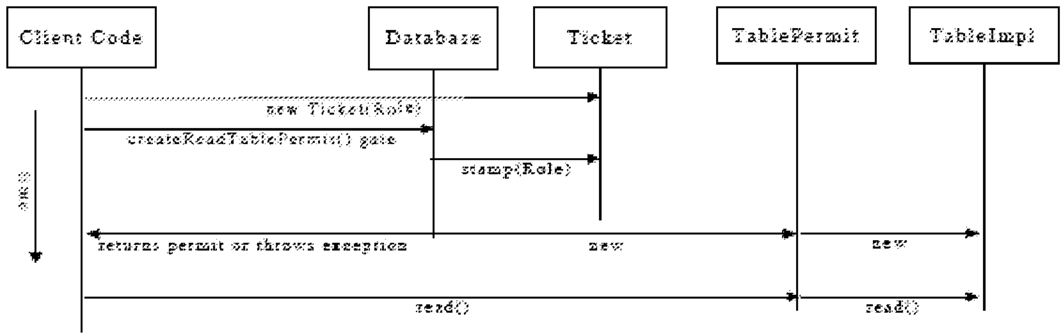
U.S. Patent No. 6,125,447 – Claim 1	*Goldstein in view of Shah*

method in Java is:

```
class Database {
   TablePermit createReadTablePermit() throws SecurityException {
      if ( caller is digitally signed with the ReadTableRole )
         return new TablePermit();
      else
         throw new SecurityException();
   }
}
```

A Java security manager class can look up the call stack. But often the immediate caller of the method is not the code that should be tested. The implementation itself will frequently invoke such a factory method. Instead of validating the eventual consumer method, the validation check is checking the implementation module itself. The JECF needs a token that represents the actual consumer of the permit. Figure 5 describes in more detail the interaction and call structure of creating the objects so the proper cassette will be validated. This token is passed from the client code into the gate method. In the JECF, this token is of class Ticket. Thus the final form of the Gateway check code is:

```
class Database {
   TablePermit createReadTablePermit(Ticket t) throws SecurityException {
      if ( t.stamp(ReadTableRole) )
         return new TablePermit();
      else
         throw new SecurityException();
   }
}”
```

U.S. Patent No. 6,125,447 – Claim 1	*Goldstein in view of Shah*
	 Client Code Database Ticket TablePermit TableImpl new Ticket(Role) createReadTablePermit() gate stamp(Role) time returns permit or throws exception new new read() read() **Figure 5. Interaction diagram for Table permits** *Goldstein* at 12-13 (citations omitted). Thus, according to *Goldstein*'s Figure 5, the client code (e.g., a cassette application) calls the "new Ticket(Role)" method, which instantiates a Ticket object corresponding to the Role (i.e., a set of permissions). The code then passes the Ticket to the Gate (i.e., the createReadTablePermit() method above) where the Gate verifies the authorization. "A Ticket is instantiated with a specific role. The gate function in the client code calls the stamp() method with the Role object as an argument to test and invalidate the Ticket. This technique limits the possibility of a Ticket being used for another possibly malicious purpose." *Goldstein* at 13. The *Shah* further describes this aspect of the *Goldstein* system: "Each cassette's identity is established at the time of signing. This Identity class itself is not directly transferred between objects; instead, a Ticket is generated as a representative for the identity. If the identity were to be directly transferred, the calling cassette could attempt to impersonate the originator and thus break security. A Ticket object is for one-time use only, and once any of its methods are invoked, the object is thrown away. This prevents illegal reuse of a Ticket." *Shah* at 3.

U.S. Patent No. 6,125,447 – Claim 1	*Goldstein in view of Shah*
	In other words, when the Gate receives the Ticket, it tests the Ticket to make sure the calling cassette is authorized for the Role, then the Gate invalidates the ticket to ensure it is used only once. In this manner, the Gate is used to establish an association between the protection domain of the cassette application and the class corresponding to the protected resource (e.g., one or more classes of one or more objects).
determining whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or more classes.	The *Goldstein* reference discloses "determining whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or more classes." For example, *Goldstein* discloses using the association between a cassette's class and the set of permissions (i.e., protection domain) of that class (as represented by the Role objects) to determine whether an action is permitted. In one particular disclosed example, the Gate checks a Role to determine if the cassette requesting authorization to read a JECF database is permitted: "For example, in the JECF database, the ability to read a row in a table and the ability to write a row in the table are both implemented in a private class named TableImplemenation. Access to the implementation for TableImplemenation from outside the implementation is granted by a pair of objects named ReadTablePermit and WriteTablePermit." *Goldstein* at 11. "Gaining access to a permit is done via a pattern called a *Gate*. A Gate is a specialized form of *factory* pattern method. A Gate decides whether to grant an instance of a Permit to a caller or deny access by throwing an exception." *Goldstein* at 12. "The JECF needs a token that represents the actual consumer of the permit. Figure 5 describes in more detail the interaction and call structure of creating the objects so the proper cassette will be validated. This token is passed from the client code into the gate method. In the JECF, this token is of class Ticket. Thus the final form of the Gateway check code is:

<table>
<tr><th>U.S. Patent No. 6,125,447 – Claim 1</th><th><em>Goldstein in view of Shah</em></th></tr>
<tr><td></td><td>

```
class Database {
   TablePermit createReadTablePermit(Ticket t) throws SecurityException {
      if ( t.stamp(ReadTableRole) )
         return new TablePermit();
      else
         throw new SecurityException();
   }
}”
```

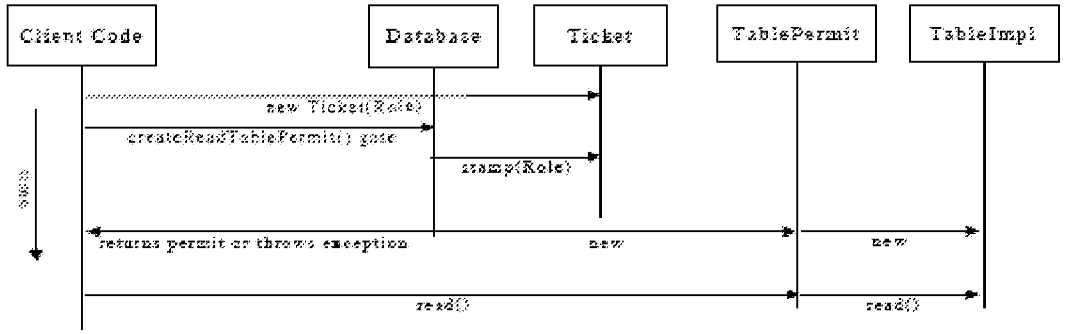


Figure 5. Interaction diagram for Table permits

Goldstein at 12-13 (citations omitted).

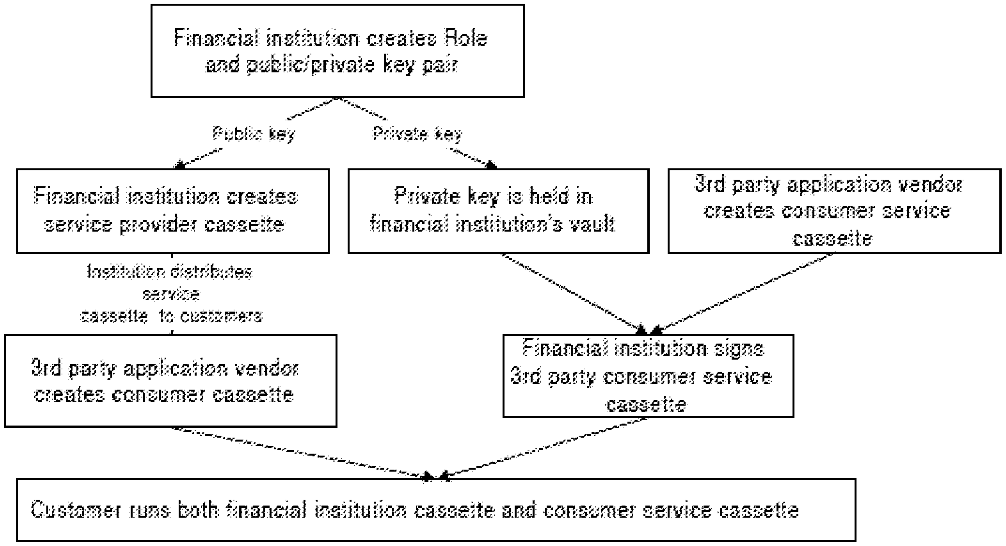
In other words, the Gate (e.g., the createReadTablePermit() method above) determines if the calling object (e.g. of the cassette application) is permitted to read the database table. This determination is made based on the association between the protection domain (e.g., a set of permissions represented by the Role that is passed to the Gate via the Ticket object) and the class of the calling object (e.g., of the cassette application).

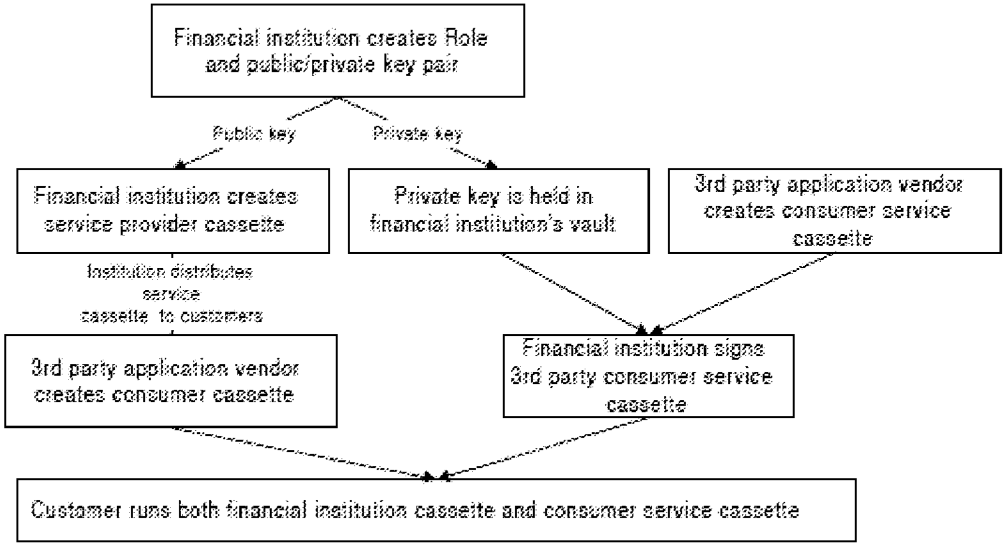
</td></tr>
</table>

U.S. Patent No. 6,125,447 – Claim 2	*Goldstein*
2. The method of claim 1, wherein:	The *Goldstein* reference discloses “the method of claim 1.” *See* claim chart above for further details.
at least one protection domain of said	The *Goldstein* reference discloses “at least one protection domain of said one or more

U.S. Patent No. 6,125,447 – Claim 2	*Goldstein*
one or more protection domains is associated with a code identifier;	protection domains is associated with a code identifier." For example, the '447 Patent discloses a code identifier as "describing the source of code that defines a class, a set of public cryptographic keys associated with the source of code, or other information which describes the source of code, or any combination thereof. A 'source of code' is an entity from which computer instructions are received." '447 Patent, 3:13-19. Figure 3 of the '447 Patent discloses a policy file that includes a URL (i.e., file://somesource) and a key name (i.e., "somekey"), and describes both as code identifiers. '447 Patent, 9:26-37 & Fig. 3. Similar to this disclosure of the "code identifier" in the '447 Patent, *Goldstein* discloses a Role (i.e., a set of permissions or protection domain) associated with a digital signature corresponding to a public key/private key pair (e.g., a code identifier). "The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. "Roles represent the signature and are used to check Tickets." *Goldstein* at 15. "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair." *Goldstein* at 13. As described above, the *Shah* further clarifies that Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness.

U.S. Patent No. 6,125,447 – Claim 2	*Goldstein*
	When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2.
at least one class of said one or more classes is associated with said code identifier; and	The *Goldstein* reference discloses "at least one class of said one or more classes is associated with said code identifier." For example, a cassette application, including all of the classes that make up the application, are associated with the code identifier because the cassette is digitally signed: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc. When third party companies want to gain access to the financial institution's cassette, they first sign a contract with the financial institution. The institution will typically impose certain usage rules about the data and Capabilities of the cassette that are specified in the contract. The institution then signs the third party cassette which is then distributed over the Internet, via floppy disks, or any other means desired. This process of code signing is illustrated in Figure 6."

U.S. Patent No. 6,125,447 – Claim 2	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 13-14 (illustrating association between public/private key pair and cassettes, which are signed by the originating financial institution).
the step of establishing an association between said one or more protection domains and said one or more classes of one or more objects further includes the step of associating said one or more protection domains and said one or more classes based on said code identifier.	The *Goldstein* reference discloses "the step of establishing an association between said one or more protection domains and said one or more classes of one or more objects further includes the step of associating said one or more protection domains and said one or more classes based on said code identifier." As described above with respect to claim 1, the financial institution may establish the association between the protection domain and the classes of the cassette (e.g., when defining or creating the Roles of a cassette and including those Roles in the cassette's package). As *Goldstein*'s Figure 6 illustrates, this process is done based on the public/private key pair (i.e., the code identifier).

U.S. Patent No. 6,125,447 – Claim 2	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes**

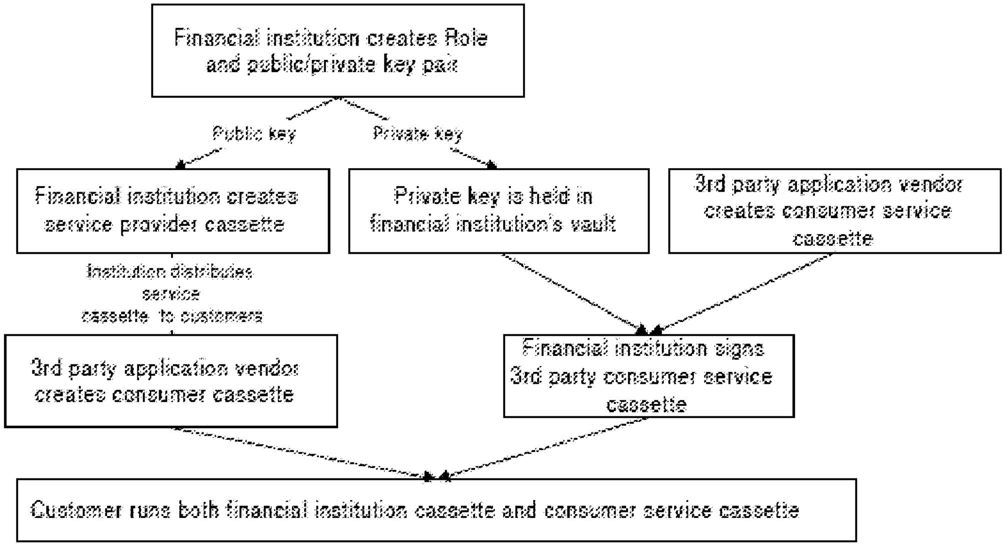
U.S. Patent No. 6,125,447 – Claim 3	*Goldstein*
3. The method of claim 2, wherein said code identifier indicates a source of code used to define each class of said one or more classes.	The *Goldstein* reference discloses "the method of claim 2, wherein said code identifier indicates a source of code used to define each class of said one or more classes." The '447 Patent discloses that "A 'source of code' is an entity from which computer instructions are received." '447 Patent, 3:15-17. As described above, *Goldstein* discloses a digital signature corresponding to a private/public key pair (e.g., a code identifier). This digital signature indicates the authority that signed the certificate, which *Goldstein* discloses as either a financial institution or a trust authority company: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial

U.S. Patent No. 6,125,447 – Claim 3	*Goldstein*
	institution’s cassette is then distributed over the Internet, via floppy disks, etc. * * * Some financial institutions will not want to do their own signing. These institutions will delegate Role signing to trust authority companies. These companies will make a business of validating the design and conformance of a cassette to the financial institutions specification.” *Goldstein* at 13-14.

U.S. Patent No. 6,125,447 – Claim 4	*Goldstein*
4. The method of claim 2, wherein said code identifier indicates a key associated with each class of said one or more classes.	The *Goldstein* reference discloses “the method of claim 2, wherein said code identifier indicates a key associated with each class of said one or more classes.” As described above, *Goldstein* discloses a digital signature corresponding to a private/public key pair (e.g., a code identifier). “Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role’s public key is embedded in the home banking, brokerage software or payment cassette. The financial institution’s cassette is then distributed over the Internet, via floppy disks, etc.” *Goldstein* at 13. The key in the cassette is associated with the classes in the cassette, as the JECF security policy is provided on a per-package basis: “Electronic commerce requires an extensible set of rights. Electronic commerce needs to identify a principal more precisely than the applet thread. Possible candidates include *threads* and *packages*. The Sandbox security manager can only control the rights invented by the Java system developers on a per thread basis. Electronic commerce also needs the ability to delegate rights from one principal to another as long as the delegation follows the contract-specified business. This cannot be done by threads alone.

U.S. Patent No. 6,125,447 – Claim 4	*Goldstein*
	In addition to threads, the Java language provides a construct called a *package*. Packages provide namespaces in the Java language. All classes in a package have access to package-private data members and methods. Package trees are primarily an organizational convention. Packages within a package tree have no special access. ***Packages are a natural choice for creating a security principal. Perhaps in a future version of Java, another security domain may emerge. Packages are a natural source of protection*.**" *Goldstein* at 10 (emphasis added).

U.S. Patent No. 6,125,447 – Claim 5	*Goldstein*
5. The method of claim 2, wherein said code identifier indicates a source of code used to define each class of said one or more classes and . . .	The *Goldstein* reference discloses "the method of claim 2, wherein said code identifier indicates a source of code used to define each class of said one or more classes." *Goldstein* expressly discloses that the digital signature (i.e., code identifier) may be associated with a financial institution (i.e., a source of code or "an entity from which computer instructions are received") that provides the cassette program: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc." *Goldstein* at 13.

U.S. Patent No. 6,125,447 – Claim 5	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 14 (showing digital signing that indicates the originating financial institution).
. . . [wherein said code identifier] indicates a key associated with each class of said one or more classes.	The *Goldstein* reference discloses ". . . [wherein said code identifier] indicates a key associated with each class of said one or more classes." For example, the digital signature includes a public key that is associated with all of the classes in the cassette application: "The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc." *Goldstein* at 13.

U.S. Patent No. 6,125,447 – Claim 6	*Goldstein*
6. The method of claim 2, wherein the	The *Goldstein* reference discloses "the method of claim 2, wherein the step of associating

U.S. Patent No. 6,125,447 – Claim 6	*Goldstein*
step of associating said one or more protection domains and said one or more classes based on said code identifier further includes associating said one or more protection domains and said one or more classes based on data persistently stored,	said one or more protection domains and said one or more classes based on said code identifier further includes associating said one or more protection domains and said one or more classes based on data persistently stored." For example, the financial institution may establish the association between the protection domain and the classes of the cassette (e.g., when defining or creating the Roles of a cassette and including those Roles in the cassette's package). This association is made based on a public/private key pair (i.e., the code identifier) that is persistently stored within the code of the cassette: "But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. The *Shah* further clarifies that Roles are stored (e.g., persistently) in a local database of the cassette: "When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2.
wherein said data associates code identifiers with a set of one or more permissions.	The *Goldstein* reference discloses "wherein said data associates code identifiers with a set of one or more permissions." As described above, the persistently stored data in *Goldstein* includes digital signatures (i.e., code identifiers) and Roles (i.e., objects that correspond to the protection domain, or set of rights, in a given cassette). *Goldstein* discloses that the Roles are implementations of the trust relationship: "The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship

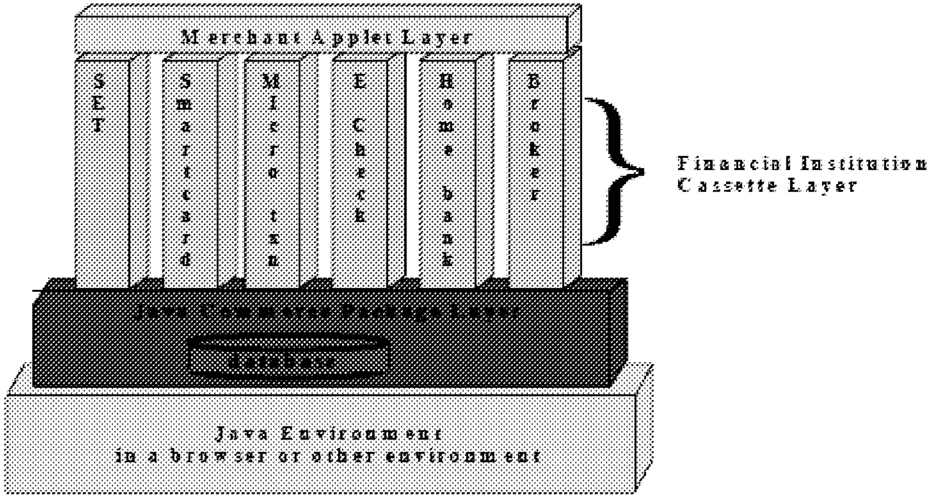
U.S. Patent No. 6,125,447 – Claim 6	*Goldstein*
	between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. *Shah* clarifies that Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2. Accordingly, the stored data associates the digital signatures (i.e., code identifier) and Roles which represent one ore more permissions.

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
7. A method of providing security, the method comprising the steps of:	The *Goldstein* reference discloses "a method of providing security." For example, *Goldstein* discloses the "*Gateway*" security model for use in the Java Electronic Commerce Framework ("JECF"). *Goldstein* discloses that the *Gateway* security model is "an extension to the current Java security model": "This paper describes an extension to the current Java security model called the 'Gateway' and why it was necessary to create it. This model allows secure applications, such as those used in electronic commerce, to safely exchange data and interoperate without compromising each individual application's security. The Gateway uses digital signatures to enable application programming interfaces to authenticate their caller.

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	JavaSoft is using the Gateway to create a new integrated open platform for financial applications called Java Electronic Commerce Framework. The JECF will be the foundation for electronic wallets, point of sale terminals, electronic merchant servers and other financial software. The Gateway model can also be used for access control in many multiple application environments that require trusted interaction between applications from multiple vendors. These applications include browsers, servers, operating systems, medical systems and smartcards." *Goldstein* at 1. "The limitations of Java motivate a new security model called the *Gateway* that extends the current Java security model. This new model is the core of an entire application framework for financial applications called the *Java Electronic Commerce Framework* (JECF)." *Goldstein* at 1. "The JECF provides a complementary model to the Sandbox security model called the *Gateway* security model. The Gateway security model provides the means to implement contractual trust relationships. This model uses the safety features of the Java language as well as the infrastructure of the Java Sandbox security model. It also provides the ability for developers to create arbitrary trust relationships." *Goldstein* at 9.
establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions;	The *Goldstein* reference discloses "establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions." The '447 Patent discloses that "A protection domain can be viewed as a set of permissions granted to one or more principals." '447 Patent, 8:42-43. ("A 'principal' is an entity in the computer system to which permissions are granted. Examples of principals include processes, objects and threads." '447 Patent, 2:25-27.) As discussed more thoroughly below, *Goldstein* discloses establishing a set of permissions that are granted to an application called a "cassette." A set of permissions (protection domain) is first established by a financial institution or third party company at the time a cassette is created. (*Goldstein* at 7-8, 10, 13-14 & Figs. 1, 5-6.) These permissions are represented by "Roles" that "reify [i.e., implement] the trust relationship between two

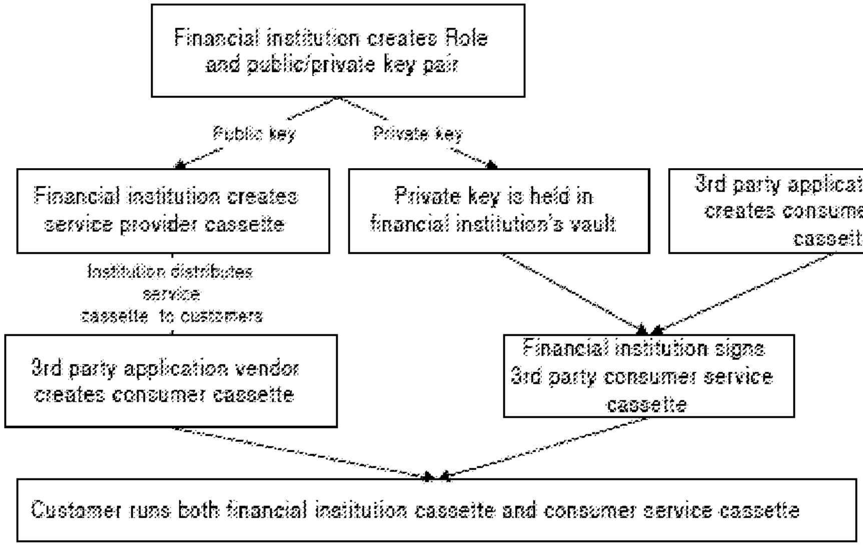
U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	business entities." (*Goldstein* at 13.) The Roles are objects that represent and/or contain the specific authorizations (e.g., permissions) for a particular cassette as well as a digital signature (based on a public/private key pair) corresponding to the creator of the cassette. (*Id.*) After the cassette application is created, it is distributed to end users. (*Goldstein* at 14.) A user installs the cassette on a computer, and subsequently the cassette (including the associated Roles objects that correspond to the set of permissions for the cassette) is loaded by a Java applet class named "CassetteLoader" which communicates with a "CassetteInstallation" object to facilitate the load operation. (*Id.*) Accordingly, establishing the protection domain, or set of rights associated with the cassette, may be thought to occur at any of the following times, as disclosed in *Goldstein*: (1) by the financial institution when it defines the Roles or permissions for an institution cassette, (2) by a third party company when it defines the Roles or permissions for a third-party cassette, or (3) by the CassetteLoader applet when it loads a given cassette (including loading and creating the corresponding Roles) in the run-time environment. Further details of the above-described system are provided below. *Goldstein* discloses establishing a protection domain as a set of permissions (e.g., "rights") that are used by a client application (e.g., a principal or "body of code"): "To completely understand what is occurring, it is necessary to define a few terms: *rights* and *principals*. In this paper, a *right* is an abstract privilege. A *principal* uses a right. A principal can be a person, a corporation, a program, or a body of code." *Goldstein* at 10. "[The JECF] allows secure applications, such as those used in electronic commerce, to safely exchange data and interoperate without compromising each individual application's security." *Goldstein* at 1. The rights in the JECF are based on the Capabilities model, which provides for the association of rights with object-oriented structures so that rights can be transferred from one principal to another:

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	“Electronic commerce needs an object-oriented security model where rights can be transferred from one principal to another. The *Capabilities model,* originally defined by Dennis and Van Horn, provides such a model. Although originally based on hardware protection, Wulf and a team at Carnegie-Mellon University[WCCJRPP74] adapted Capabilities to object-oriented technology. The object-oriented definition of Capabilities is very simple. The Capabilities model means that possession of an object confers the right to use it. Capabilities need a gatekeeper service to decide whether to grant a right to the object. * * * Adding the Capabilities model to Java has proven to be simple because Java already has many of the necessary features. Java objects are unforgeable. Java’s language visibility rules for private and package-private scope provide address space-like protection against unauthorized access. The only needed component is some form of authorization mechanism to allow or deny access. In the JECF, individual Capabilities are called *permits*. Permit objects require careful handling. Permit objects should never be stored in a public or protected variable. Permits are rarely the actual implementation of some right, but are typically implemented by another object. The permit forwards the call to the actual implementation.” *Goldstein* at 10-11. *Goldstein* discloses client applications as “cassettes” that may reside on a user’s system: “Many object-oriented practitioners use the term ‘framework’ to mean an ‘application programming interface.’ In this paper, the term ‘framework’ refers to a set of APIs that impose responsibilities amongst participating software packages. The JECF defines application responsibilities for merchants and financial institutions. JECF provides services, such as database services, to merchant and financial institution applications. These layers are depicted in Figure 1.

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	 Merchant Applet Layer SET Smartcard Micro tran E Check Home bank Broker Financial Institution Cassette Layer Java Commerce Package Layer Database Java Environment in a browser or other environment **Figure 1. Simple Schema of JECF** * * * The *Cassette Layer* implements long term customer relationships such as credit cards, home banking and brokerages. Cassettes are a new feature that JECF adds to Java. ***Similar to applets, cassettes are downloaded from servers to client computers. Unlike applets which disappear when users quit the browser, cassettes are retained on the customer's system***. Cassettes store information in a database provided by the JECF. Cassettes may safely store valuable information such as public key certificates and transaction records since the entire database is encrypted. Cassettes provide long term customer-to-institution relationships. Examples of cassettes include SET certificates and protocols, home banking, brokerage accounts, financial analysis, and planning software. Cassettes contain code, digital certificates, GIF images and other resources. Financial institutions will use cassettes to deliver customer service features. Smartcard application developers can put smartcard reader device drivers and application user interfaces in cassettes." *Goldstein* at 7-8 (emphasis added). *Goldstein* discloses examples of the types of rights (permissions) that may be associated

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	with a cassette application: “Potentially dangerous operations include opening a file or a network socket, changing a system variable, and setting the security manager.” *Goldstein* at 7. *Goldstein* discloses the use of objects in the “Roles” class that correspond to the protection domain, or set of rights, in a given cassette: “The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant.” *Goldstein* at 13. To the extent *Goldstein* fails to explicitly disclose the exact contents of Roles, a person of ordinary skill in the art would have understood that the Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application. For example, *Shah* provides: “When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program’s interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator.” *Shah* at 2. Accordingly, the *Shah* discloses features that are necessarily present in the JECF system disclosed by *Goldstein*. *Goldstein* discloses that cassettes, and the Roles associated with an individual cassette, may

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	be created by the financial institution (in the case of a cassette provided by the institution) or by a third party company. As described in the *Shah*, *Goldstein* discloses that the Roles and a digital signature (based on a public/private key pair) are created and stored in a cassette that is eventually distributed to the end user: “Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role’s public key is embedded in the home banking, brokerage software or payment cassette. The financial institution’s cassette is then distributed over the Internet, via floppy disks, etc. When third party companies want to gain access to the financial institution’s cassette, they first sign a contract with the financial institution. The institution will typically impose certain usage rules about the data and Capabilities of the cassette that are specified in the contract. The institution then signs the third party cassette which is then distributed over the Internet, via floppy disks, or any other means desired. This process of code signing is illustrated in Figure 6.”

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 13-14. In the distributed cassette application, *Goldstein* discloses that "an extensible set of rights" (e.g., a protection domain) may be established on a per-package basis or, "in a future version of Java, another security domain may emerge": "Electronic commerce requires an extensible set of rights. Electronic commerce needs to identify a principal more precisely than the applet thread. Possible candidates include *threads* and *packages*. The Sandbox security manager can only control the rights invented by the Java system developers on a per thread basis. Electronic commerce also needs the ability to delegate rights from one principal to another as long as the delegation follows the contract-specified business. This cannot be done by threads alone. In addition to threads, the Java language provides a construct called a *package*. Packages provide namespaces in the Java language. All classes in a package have access to

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	package-private data members and methods. Package trees are primarily an organizational convention. Packages within a package tree have no special access. ***Packages are a natural choice for creating a security principal. Perhaps in a future version of Java, another security domain may emerge. Packages are a natural source of protection*.” *Goldstein* at 10 (emphasis added). *Goldstein* discloses that a user may install the cassettes on a computer using the CassetteLocator applet in conjunction with the CassetteInstallation object in the JECF. At this time, the protection domain is established in the run-time environment: “Installation of cassettes is initiated by a Java applet class named CassetteLocator. CassetteLocator applets are dynamically loaded from financial institutions and other software suppliers. CassetteLocators communicate with a CassetteInstallation object in the JECF to negotiate loading of cassettes.” *Goldstein* at 14.
establishing an association between said one or more protection domains and one or more sources of code; and	The *Goldstein* reference discloses “establishing an association between said one or more protection domains and one or more sources of code.” For example, the ‘447 Patent discloses a “source of code” as “an entity from which computer instructions are received.” ‘447 Patent, 3:15-19. Similar to this disclosure of the “source of code” in the ‘447 Patent, *Goldstein* discloses the Roles objects (i.e., protection domain data structure), which explicitly associates the protection domain with a “source of code” such as the signer of a digital certificate: “The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant.” *Goldstein* at 13.

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	"Roles represent the signature and are used to check Tickets." *Goldstein* at 15. "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair." *Goldstein* at 13. As described above, the *Shah* further clarifies that Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2.
in response to executing code making a request to perform an action, determining whether said request is permitted based on a source of said code making said request and said association between said one or more protection domains and said one or more sources of code.	The *Goldstein* reference discloses "in response to executing code making a request to perform an action, determining whether said request is permitted based on a source of said code making said request and said association between said one or more protection domains and said one or more sources of code." For example, *Goldstein* discloses using the digital signature of a cassette (i.e., indicating the source of code) and an associated Roles object (i.e., protection domain data structure) to determine whether an action is permitted. In one particular disclosed example, the Gate checks a Role to determine if the cassette requesting authorization to read a JECF database is permitted: "For example, in the JECF database, the ability to read a row in a table and the ability to write a row in the table are both implemented in a private class named TableImplemenation. Access to the implementation for TableImplemenation from outside the implementation is granted by a pair of objects named ReadTablePermit and WriteTablePermit." *Goldstein* at

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	11.

"Gaining access to a permit is done via a pattern called a *Gate*. A Gate is a specialized form of *factory* pattern method. A Gate decides whether to grant an instance of a Permit to a caller or deny access by throwing an exception." *Goldstein* at 12.

"The JECF needs a token that represents the actual consumer of the permit. Figure 5 describes in more detail the interaction and call structure of creating the objects so the proper cassette will be validated. This token is passed from the client code into the gate method. In the JECF, this token is of class Ticket. Thus the final form of the Gateway check code is:

```
class Database {
   TablePermit createReadTablePermit(Ticket t) throws SecurityException {
      if ( t.stamp(ReadTableRole) )
         return new TablePermit();
      else
         throw new SecurityException();
   }
}"
```

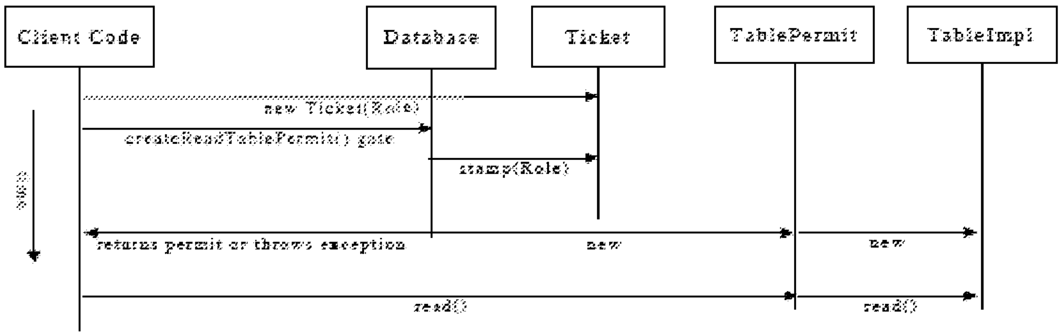


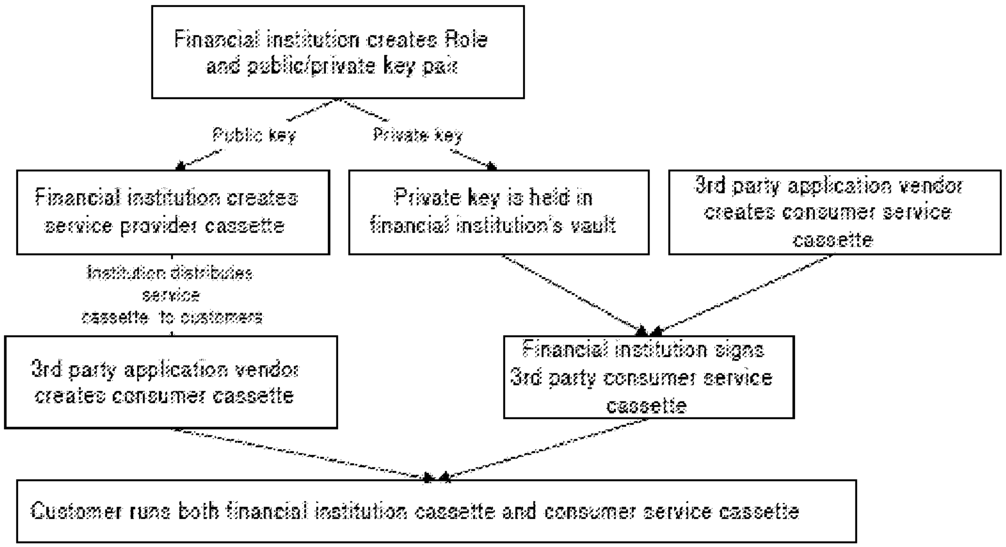
Figure 5. Interaction diagram for Table permits

Goldstein at 12-13 (citations omitted).

In other words, the Gate (e.g., the createReadTablePermit() method above) determines if the

U.S. Patent No. 6,125,447 – Claim 7	*Goldstein*
	calling object (e.g. of the cassette application) is permitted to read the database table. This determination is made based on the digital signature (i.e., indicating the source of code) and an associated Roles object (i.e., a set of permissions, or protection domain data structure). As shown in Figure 5 (and accompanying text at 12-13), the Role is passed to the Gate via the Ticket object. The Gate verifies the permission before sending a Permit object back to the calling object in the cassette application.

U.S. Patent No. 6,125,447 – Claim 8	*Goldstein*
8. The method of claim 7, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code further includes establishing an association between said one or more protection domains and said one or more sources of code and one or more keys associated with said one or more sources of code.	The *Goldstein* reference discloses "the method of claim 7, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code further includes establishing an association between said one or more protection domains and said one or more sources of code and one or more keys associated with said one or more sources of code." As described above with respect to claim 7, the financial institution may establish the association between the protection domain (e.g., set of permissions implemented as Roles objects) and the source of code (e.g., the digital certificate), for example, when defining or creating the Roles for a cassette and including those Roles in the cassette's package. As *Goldstein*'s Figure 6 illustrates, this process uses a the public/private key pair, which identify the financial institution as the source of code:

U.S. Patent No. 6,125,447 – Claim 8	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 14 (top box showing association of protection domain (e.g., set of permissions implemented as Roles) and source of code (e.g., keys that identify financial institution as originator of code for cassette)).

U.S. Patent No. 6,125,447 – Claim 9	*Goldstein*
9. The method of claim 8, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code and said one or more keys associated with said one or more sources of code further includes establishing said association between said one or more protection domains and said one or more sources of code	The *Goldstein* reference discloses "the method of claim 8, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code and said one or more keys associated with said one or more sources of code further includes establishing said association between said one or more protection domains and said one or more sources of code and said one or more keys associated with said one or more sources of code based on data persistently stored." For example, the financial institution may establish the association between the protection domain and the source of code (e.g., the digital certificate), for example, when defining or creating the Roles for a cassette and including those Roles in the cassette's package. This

U.S. Patent No. 6,125,447 – Claim 9	*Goldstein*
and said one or more keys associated with said one or more sources of code based on data persistently stored,	association is made based on a public/private key pair (i.e., the code identifier) that is persistently stored within the code of the cassette: “But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant.” *Goldstein* at 13. The *Shah* further clarifies that Roles are stored (e.g., persistently) in a local database of the cassette: “When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program’s interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator.” *Shah* at 2.
wherein said data associates particular sources of code and particular keys with a set of one or more permissions.	The *Goldstein* reference discloses “wherein said data associates particular sources of code and particular keys with a set of one or more permissions.” As described above, the persistently stored data in *Goldstein* includes digital signatures that include a public key (i.e., sources of code) and Roles (i.e., objects that correspond to the set of rights in a given cassette). Again, *Goldstein* discloses that the Roles are implementations of the trust relationship: “The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant.” *Goldstein* at 13. *Shah* clarifies that Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the

U.S. Patent No. 6,125,447 – Claim 9	*Goldstein*
	cassette application: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2. Accordingly, the stored data associates the digital signatures (i.e., sources of code) and associated public/private key pair and Roles which represent one ore more permissions.

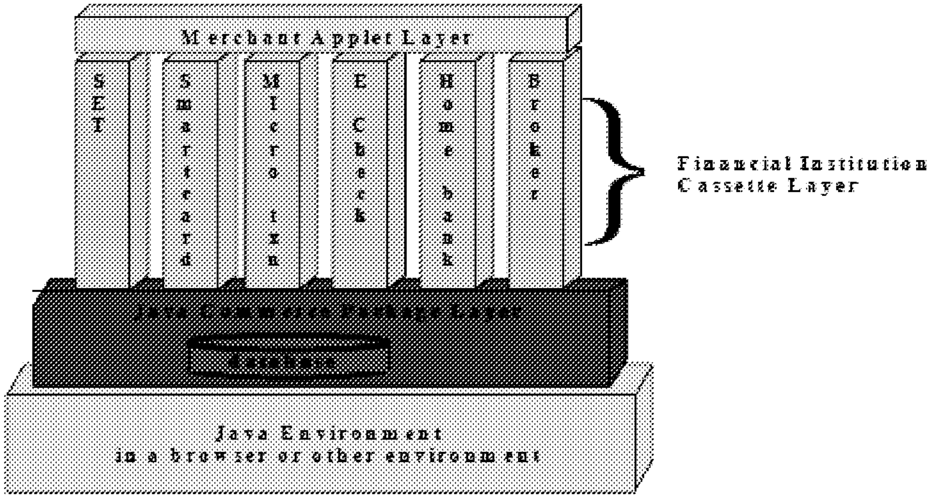
NOTE: Claims 10-18 are exact replicas of claims 1-9, except that claims 10-18 are written as apparatus claims (instructions embodied in a computer readable medium), whereas claims 1-9 are written as method claims.

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
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:	The *Goldstein* reference discloses "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." For example, the '447 defines computer-readable medium as: "The term 'computer-readable medium' as used herein refers to any medium that participates in providing instructions to processor 104 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks, such as storage device 110. Volatile media includes dynamic memory, such as main memory 106. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 102. Transmission media can also take the form of acoustic or light

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
	waves, such as those generated during radio-wave and infra-red data communications. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read." '447 Patent at 5:4-25. Consistent with this definition of "computer-readable media" in the '447 patent, *Goldstein* discloses software that is run on personal computers: "Sun Microsystems Inc.'s Java programming language and APIs provide a reliable, secure platform to deliver applications on Internet and Intranet networks. Without Java, users who downloaded applications from these networks could be vulnerable to dangerous software viruses. All of the popular personal computer operating systems are vulnerable to such attacks. Of course, Java executing on conventional personal computers cannot totally prevent virus attacks." *Goldstein* at 1. *Goldstein* discloses specific cassette applications software as part of the JECF that are implemented in software and distributed to end user's over the internet (i.e., computer-readable medium carrier wave): "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc." *Goldstein* at 13.
establishing one or more protection domains, wherein a protection domain	The *Goldstein* reference discloses "establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions."

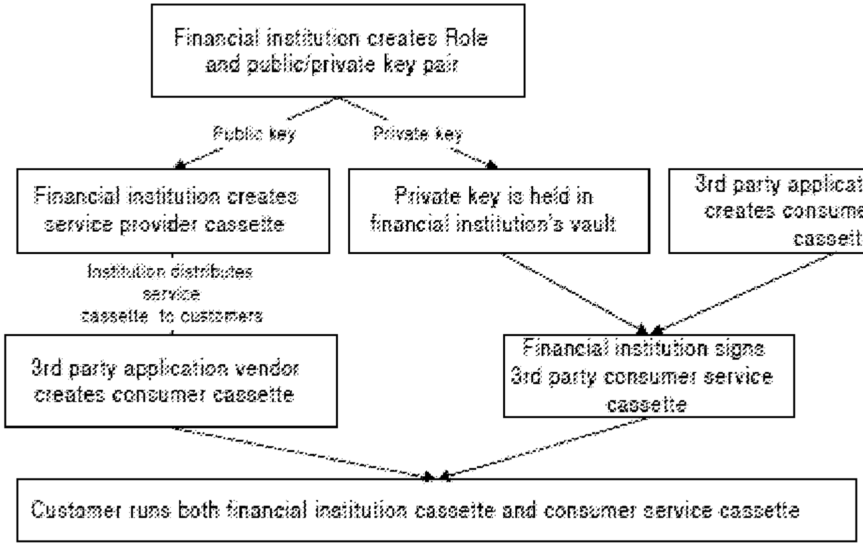
U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
is associated with zero or more permissions;	The ‘447 Patent discloses that “A protection domain can be viewed as a set of permissions granted to one or more principals.” ‘447 Patent, 8:42-43. (“A ‘principal’ is an entity in the computer system to which permissions are granted. Examples of principals include processes, objects and threads.” ‘447 Patent, 2:25-27.) As discussed more thoroughly below, *Goldstein* discloses establishing a set of permissions that are granted to an application called a “cassette.” A set of permissions (protection domain) is first established by a financial institution or third party company at the time a cassette is created. (*Goldstein* at 7-8, 10, 13-14 & Figs. 1, 5-6.) These permissions are represented by “Roles” that “reify [i.e., implement] the trust relationship between two business entities.” (*Goldstein* at 13.) The Roles are objects that represent and/or contain the specific authorizations (e.g., permissions) for a particular cassette as well as a digital signature (based on a public/private key pair) corresponding to the creator of the cassette. (*Id.*) After the cassette application is created, it is distributed to end users. (*Goldstein* at 14.) A user installs the cassette on a computer, and subsequently the cassette (including the associated Roles objects that correspond to the set of permissions for the cassette) is loaded by a Java applet class named “CassetteLoader” which communicates with a “CassetteInstallation” object to facilitate the load operation. (*Id.*) Accordingly, establishing the protection domain, or set of rights associated with the cassette, may be thought to occur at any of the following times, as disclosed in *Goldstein*: (1) by the financial institution when it defines the Roles or permissions for an institution cassette, (2) by a third party company when it defines the Roles or permissions for a third-party cassette, or (3) by the CassetteLoader applet when it loads a given cassette (including loading and creating the corresponding Roles) in the run-time environment. Further details of the above-described system are provided below. *Goldstein* discloses establishing a protection domain as a set of permissions (e.g., “rights”) that are used by a client application (e.g., a principal or “body of code”):

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
	“To completely understand what is occurring, it is necessary to define a few terms: *rights* and *principals*. In this paper, a *right* is an abstract privilege. A *principal* uses a right. A principal can be a person, a corporation, a program, or a body of code.” *Goldstein* at 10. “[The JECF] allows secure applications, such as those used in electronic commerce, to safely exchange data and interoperate without compromising each individual application’s security.” *Goldstein* at 1. The rights in the JECF are based on the Capabilities model, which provides for the association of rights with object-oriented structures so that rights can be transferred from one principal to another: “Electronic commerce needs an object-oriented security model where rights can be transferred from one principal to another. The *Capabilities model,* originally defined by Dennis and Van Horn, provides such a model. Although originally based on hardware protection, Wulf and a team at Carnegie-Mellon University[WCCJRPP74] adapted Capabilities to object-oriented technology. The object-oriented definition of Capabilities is very simple. The Capabilities model means that possession of an object confers the right to use it. Capabilities need a gatekeeper service to decide whether to grant a right to the object. * * * Adding the Capabilities model to Java has proven to be simple because Java already has many of the necessary features. Java objects are unforgeable. Java’s language visibility rules for private and package-private scope provide address space-like protection against unauthorized access. The only needed component is some form of authorization mechanism to allow or deny access. In the JECF, individual Capabilities are called *permits*. Permit objects require careful handling. Permit objects should never be stored in a public or protected variable. Permits are rarely the actual implementation of some right, but are typically implemented by another object. The permit forwards the call to the actual implementation.” *Goldstein* at 10-11.

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
	Goldstein discloses client applications as "cassettes" that may reside on a user's system: "Many object-oriented practitioners use the term 'framework' to mean an 'application programming interface.' In this paper, the term 'framework' refers to a set of APIs that impose responsibilities amongst participating software packages. The JECF defines application responsibilities for merchants and financial institutions. JECF provides services, such as database services, to merchant and financial institution applications. These layers are depicted in Figure 1.  Merchant Applet Layer SET Smartcard MIcro txn E Check Home bank Broker Financial Institution Cassette Layer Java Commerce Package Layer database Java Environment in a browser or other environment **Figure 1. Simple Schema of JECF** * * * The *Cassette Layer* implements long term customer relationships such as credit cards, home banking and brokerages. Cassettes are a new feature that JECF adds to Java. ***Similar to applets, cassettes are downloaded from servers to client computers. Unlike applets which disappear when users quit the browser, cassettes are retained on the customer's system***. Cassettes store information in a database provided by the JECF. Cassettes may safely store valuable information such as public key certificates and transaction records since the entire database is encrypted. Cassettes provide long term

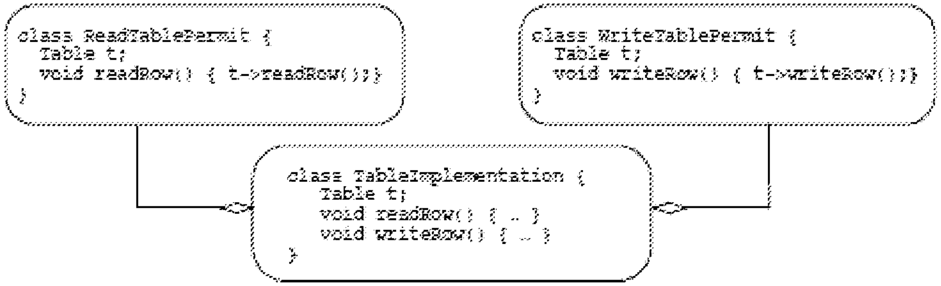
U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
	customer-to-institution relationships. Examples of cassettes include SET certificates and protocols, home banking, brokerage accounts, financial analysis, and planning software. Cassettes contain code, digital certificates, GIF images and other resources. Financial institutions will use cassettes to deliver customer service features. Smartcard application developers can put smartcard reader device drivers and application user interfaces in cassettes." *Goldstein* at 7-8 (emphasis added). *Goldstein* discloses examples of the types of rights (permissions) that may be associated with a cassette application: "Potentially dangerous operations include opening a file or a network socket, changing a system variable, and setting the security manager." *Goldstein* at 7. *Goldstein* discloses the use of objects in the "Roles" class that correspond to the protection domain, or set of rights, in a given cassette: "The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. To the extent *Goldstein* fails to explicitly disclose the exact contents of Roles, a person of ordinary skill in the art would have understood that the Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application. For example, *Shah* provides: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
	interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2. Accordingly, the *Shah* discloses features that are necessarily present in the JECF system disclosed by *Goldstein*. *Goldstein* discloses that cassettes, and the Roles associated with an individual cassette, may be created by the financial institution (in the case of a cassette provided by the institution) or by a third party company. As described in the *Shah*, *Goldstein* discloses that the Roles and a digital signature (based on a public/private key pair) are created and stored in a cassette that is eventually distributed to the end user: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc. When third party companies want to gain access to the financial institution's cassette, they first sign a contract with the financial institution. The institution will typically impose certain usage rules about the data and Capabilities of the cassette that are specified in the contract. The institution then signs the third party cassette which is then distributed over the Internet, via floppy disks, or any other means desired. This process of code signing is illustrated in Figure 6."

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 13-14. In the distributed cassette application, *Goldstein* discloses that "an extensible set of rights" (e.g., a protection domain) may be established on a per-package basis or, "in a future version of Java, another security domain may emerge": "Electronic commerce requires an extensible set of rights. Electronic commerce needs to identify a principal more precisely than the applet thread. Possible candidates include *threads* and *packages*. The Sandbox security manager can only control the rights invented by the Java system developers on a per thread basis. Electronic commerce also needs the ability to delegate rights from one principal to another as long as the delegation follows the contract-specified business. This cannot be done by threads alone. In addition to threads, the Java language provides a construct called a *package*. Packages provide namespaces in the Java language. All classes in a package have access to

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
	package-private data members and methods. Package trees are primarily an organizational convention. Packages within a package tree have no special access. ***Packages are a natural choice for creating a security principal. Perhaps in a future version of Java, another security domain may emerge. Packages are a natural source of protection***." *Goldstein* at 10 (emphasis added). *Goldstein* discloses that a user may install the cassettes on a computer using the CassetteLocator applet in conjunction with the CassetteInstallation object in the JECF. At this time, the protection domain is established in the run-time environment: "Installation of cassettes is initiated by a Java applet class named CassetteLocator. CassetteLocator applets are dynamically loaded from financial institutions and other software suppliers. CassetteLocators communicate with a CassetteInstallation object in the JECF to negotiate loading of cassettes." *Goldstein* at 14.
establishing an association between said one or more protection domains and one or more classes of one or more objects; and	The *Goldstein* reference discloses "establishing an association between said one or more protection domains and one or more classes of one or more objects." For example, as discussed above, *Goldstein* discloses that the protection domain (e.g., Roles or permissions) is defined on a per-package basis. Thus, any class that is part of the package (e.g., a cassette package) is necessarily associated with the protection domain: "Electronic commerce also needs the ability to delegate rights from one principal to another as long as the delegation follows the contract-specified business. This cannot be done by threads alone. In addition to threads, the Java language provides a construct called a *package*. Packages provide namespaces in the Java language. ***All classes in a package have access to package-private data members and methods.*** Package trees are primarily an organizational convention. Packages within a package tree have no special access. ***Packages are a natural choice for creating a security principal.*** Perhaps in a future version of Java, another security domain may emerge. Packages are a natural source of protection." *Goldstein* at 10

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*

	(emphasis added).
	
	Goldstein discloses additional methods of establishing associations between the protection domain and one or more classes of one or more objects. For example, *Goldstein* discloses establishing an association between the protection domain of a user's cassette with a class corresponding to a protected resource such as a JECF database:
	
	"Adding the Capabilities model to Java has proven to be simple because Java already has many of the necessary features. Java objects are unforgeable. Java's language visibility rules for private and package-private scope provide address space-like protection against unauthorized access. The only needed component is some form of authorization mechanism to allow or deny access. In the JECF, individual Capabilities are called *permits*. Permit objects require careful handling. Permit objects should never be stored in a public or protected variable. Permits are rarely the actual implementation of some right, but are typically implemented by another object. The permit forwards the call to the actual implementation. For example, in the JECF database, the ability to read a row in a table and the ability to write a row in the table are both implemented in a private class named TableImplemenation. Access to the implementation for TableImplemenation from outside the implementation is granted by a pair of objects named ReadTablePermit and WriteTablePermit. Figure 4 illustrates the relationship between permits and the permit's underlying implementation in the Java language in the Java language."
	
	
	`class ReadTablePermit {`
	`  Table t;`
	`  void readRow() { t->readRow();}`
	`}`
	`class WriteTablePermit {`
	`  Table t;`
	`  void writeRow() { t->writeRow();}`
	`}`
	`class TableImplementation {`
	`  Table t;`
	`  void readRow() { .. }`
	`  void writeRow() { .. }`
	`}`
	Figure 4. Relationship of permit objects to implementation object

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*

Goldstein at 11-12.

Goldstein discloses that a “Gate” is used to establish an association between the protection domain of the cassette application and the class corresponding to the protected resource (e.g., JECF database class). The Gate does this by sending the Permit to the client code:

“Permits are Capabilities implemented with Java objects.” *Goldstein* at 15.

“A permit is a delegater pattern (as described in *Design Patterns Methodology*) to the actual implementation object. Both the permits and the implementation are in the same package. The implementation however will have only package private constructors. The permit objects need to be obtained by a client package to be useful.

Gaining access to a permit is done via a pattern called a *Gate*. A Gate is a specialized form of *factory* pattern method. A Gate decides whether to grant an instance of a Permit to a caller or deny access by throwing an exception. The JECF uses the Java Sandbox digital signature mechanism for authentication. The general form of a gate method in Java is:

```
class Database {
  TablePermit createReadTablePermit() throws SecurityException {
    if ( caller is digitally signed with the ReadTableRole )
      return new TablePermit();
    else
      throw new SecurityException();
  }
}
```

A Java security manager class can look up the call stack. But often the immediate caller of the method is not the code that should be tested. The implementation itself will frequently invoke such a factory method. Instead of validating the eventual consumer method, the validation check is checking the implementation module itself. The JECF needs a token that represents the actual consumer of the permit. Figure 5 describes in more detail the interaction and call structure of creating the objects so the proper cassette will be

<table>
<tr><th>U.S. Patent No. 6,125,447 – Claim 10</th><th><em>Goldstein</em></th></tr>
<tr><td></td><td>

validated. This token is passed from the client code into the gate method. In the JECF, this token is of class Ticket. Thus the final form of the Gateway check code is:

```
class Database {
   TablePermit createReadTablePermit(Ticket t) throws SecurityException {
      if ( t.stamp(ReadTableRole) )
         return new TablePermit();
      else
         throw new SecurityException();
   }
}”
```

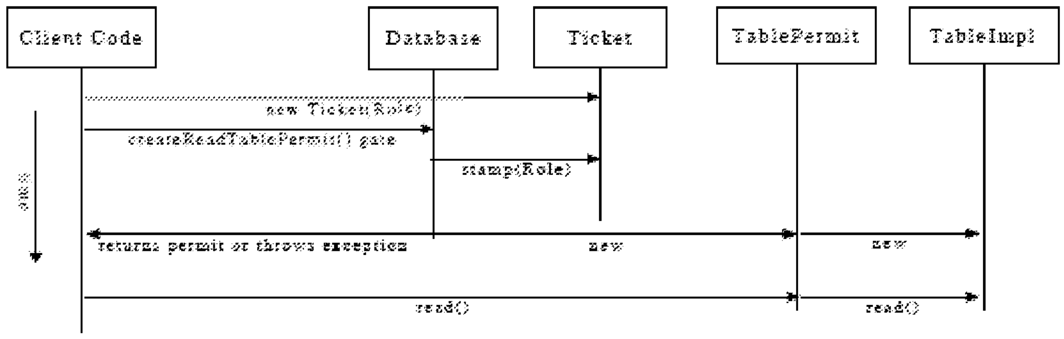


Figure 5. Interaction diagram for Table permits

Goldstein at 12-13 (citations omitted).

Thus, according to *Goldstein*’s Figure 5, the client code (e.g., a cassette application) calls the “new Ticket(Role)” method, which instantiates a Ticket object corresponding to the Role (i.e., a set of permissions). The code then passes the Ticket to the Gate (i.e., the createReadTablePermit() method above) where the Gate verifies the authorization.

“A Ticket is instantiated with a specific role. The gate function in the client code calls the stamp() method with the Role object as an argument to test and invalidate the Ticket. This technique limits the possibility of a Ticket being used for another possibly malicious purpose.” *Goldstein* at 13.

</td></tr>
</table>

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
	The *Shah* further describes this aspect of the *Goldstein* system: "Each cassette's identity is established at the time of signing. This Identity class itself is not directly transferred between objects; instead, a Ticket is generated as a representative for the identity. If the identity were to be directly transferred, the calling cassette could attempt to impersonate the originator and thus break security. A Ticket object is for one-time use only, and once any of its methods are invoked, the object is thrown away. This prevents illegal reuse of a Ticket." *Shah* at 3. In other words, when the Gate receives the Ticket, it tests the Ticket to make sure the calling cassette is authorized for the Role, then the Gate invalidates the ticket to ensure it is used only once. In this manner, the Gate is used to establish an association between the protection domain of the cassette application and the class corresponding to the protected resource (e.g., one or more classes of one or more objects).
determining whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or more classes.	The *Goldstein* reference discloses "determining whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or more classes." For example, *Goldstein* discloses using the association between a cassette's class and the set of permissions (i.e., protection domain) of that class (as represented by the Role objects) to determine whether an action is permitted. In one particular disclosed example, the Gate checks a Role to determine if the cassette requesting authorization to read a JECF database is permitted: "For example, in the JECF database, the ability to read a row in a table and the ability to write a row in the table are both implemented in a private class named TableImplemenation. Access to the implementation for TableImplemenation from outside the implementation is granted by a pair of objects named ReadTablePermit and WriteTablePermit." *Goldstein* at 11.

<table>
<tr><th>U.S. Patent No. 6,125,447 – Claim 10</th><th><i>Goldstein</i></th></tr>
<tr><td></td><td>

“Gaining access to a permit is done via a pattern called a *Gate*. A Gate is a specialized form of *factory* pattern method. A Gate decides whether to grant an instance of a Permit to a caller or deny access by throwing an exception.” *Goldstein* at 12.

“The JECF needs a token that represents the actual consumer of the permit. Figure 5 describes in more detail the interaction and call structure of creating the objects so the proper cassette will be validated. This token is passed from the client code into the gate method. In the JECF, this token is of class Ticket. Thus the final form of the Gateway check code is:

```
class Database {
   TablePermit createReadTablePermit(Ticket t) throws SecurityException {
      if ( t.stamp(ReadTableRole) )
         return new TablePermit();
      else
         throw new SecurityException();
   }
}”
```

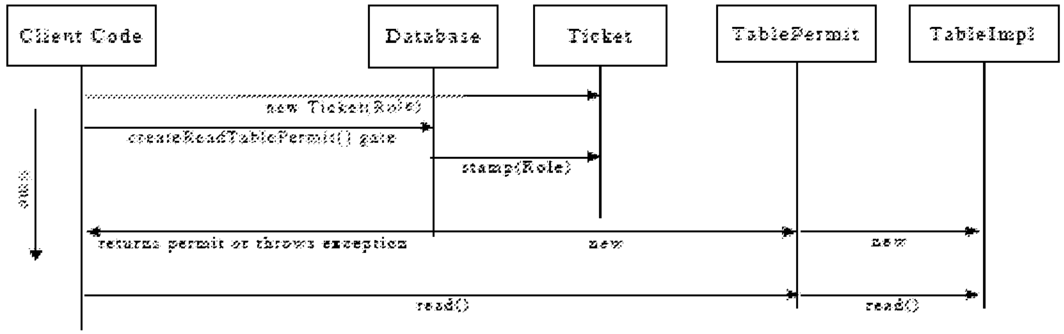


Figure 5. Interaction diagram for Table permits

Goldstein at 12-13 (citations omitted).

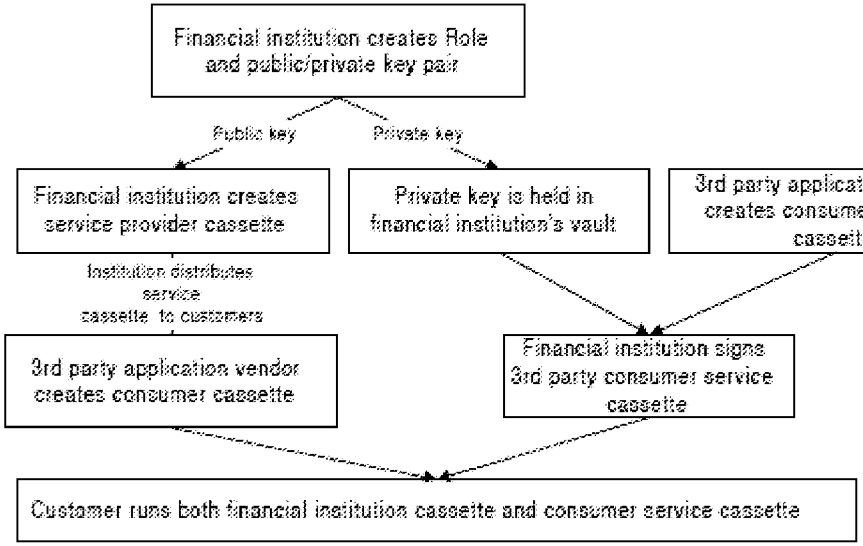
In other words, the Gate (e.g., the createReadTablePermit() method above) determines if the calling object (e.g. of the cassette application) is permitted to read the database table. This determination is made based on the association between the protection domain (e.g., a set of

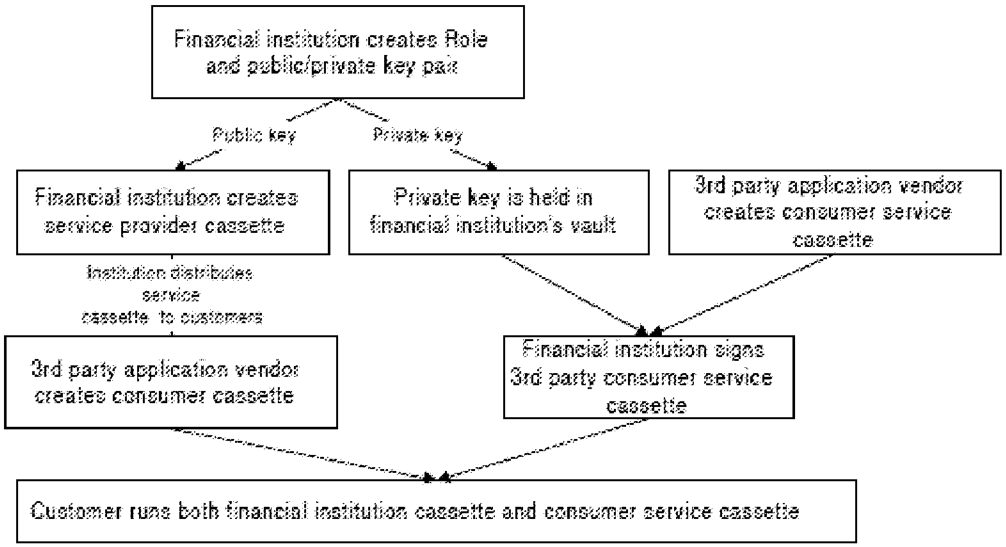
</td></tr>
</table>

U.S. Patent No. 6,125,447 – Claim 10	*Goldstein*
	permissions represented by the Role that is passed to the Gate via the Ticket object) and the class of the calling object (e.g., of the cassette application).

U.S. Patent No. 6,125,447 – Claim 11	*Goldstein*
11. The computer readable medium of claim 10, wherein:	The *Goldstein* reference discloses "the computer readable medium of claim 10." *See* claim chart for claim 10 above.
at least one protection domain of said one or more protection domains is associated with a code identifier;	The *Goldstein* reference discloses "at least one protection domain of said one or more protection domains is associated with a code identifier." For example, the '447 Patent discloses a code identifier as "describing the source of code that defines a class, a set of public cryptographic keys associated with the source of code, or other information which describes the source of code, or any combination thereof. A 'source of code' is an entity from which computer instructions are received." '447 Patent, 3:13-19. Figure 3 of the '447 Patent discloses a policy file that includes a URL (i.e., file://somesource) and a key name (i.e., "somekey"), and describes both as code identifiers. '447 Patent, 9:26-37 & Fig. 3. Similar to this disclosure of the "code identifier" in the '447 Patent, *Goldstein* discloses a Role (i.e., a set of permissions or protection domain) associated with a digital signature corresponding to a public key/private key pair (e.g., a code identifier). "The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. "Roles represent the signature and are used to check Tickets." *Goldstein* at 15.

U.S. Patent No. 6,125,447 – Claim 11	*Goldstein*
	"Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair." *Goldstein* at 13. As described above, the *Shah* further clarifies that Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2.
at least one class of said one or more classes is associated with said code identifier; and	The *Goldstein* reference discloses "at least one class of said one or more classes is associated with said code identifier." For example, a cassette application, including all of the classes that make up the application, are associated with the code identifier because the cassette is digitally signed: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc. When third party companies want to gain access to the financial institution's cassette, they first sign a contract with the financial institution. The institution will typically impose certain usage rules about the data and Capabilities of the cassette that are specified in the contract. The institution then signs the third party cassette which is then distributed over the Internet, via floppy disks, or any other means desired. This process of code signing is illustrated in Figure 6."

U.S. Patent No. 6,125,447 – Claim 11	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 13-14 (illustrating association between public/private key pair and cassettes, which are signed by the originating financial institution).
the step of establishing an association between said one or more protection domains and said one or more classes of one or more objects further includes the step of associating said one or more protection domains and said one or more classes based on said code identifier.	The *Goldstein* reference discloses the step of establishing an association between said one or more protection domains and said one or more classes of one or more objects further includes the step of associating said one or more protection domains and said one or more classes based on said code identifier. As described above with respect to claim 1, the financial institution may establish the association between the protection domain and the classes of the cassette (e.g., when defining or creating the Roles of a cassette and including those Roles in the cassette's package). As *Goldstein*'s Figure 6 illustrates, this process is done based on the public/private key pair (i.e., the code identifier).

U.S. Patent No. 6,125,447 – Claim 11	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes**

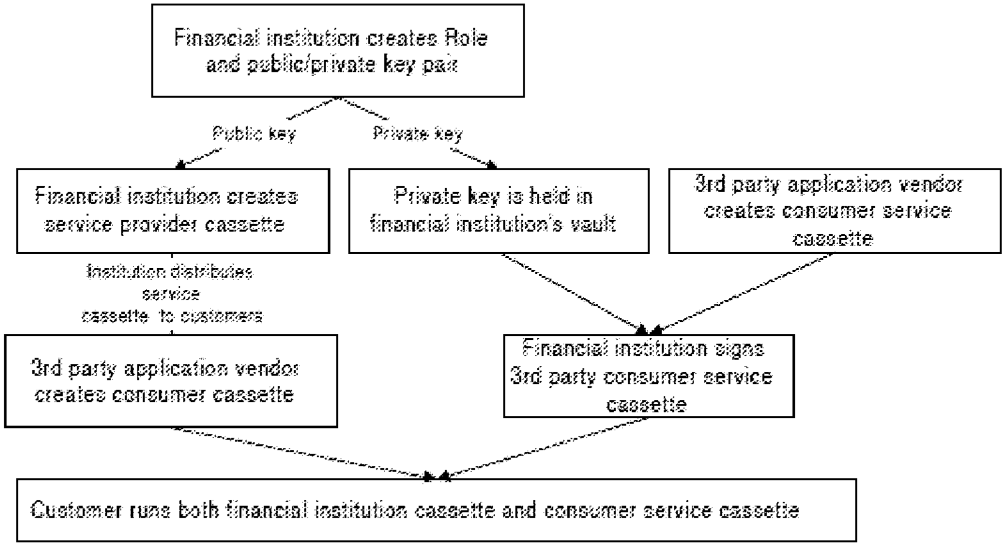
U.S. Patent No. 6,125,447 – Claim 12	*Goldstein*
12. The computer readable medium of claim 11, wherein said code identifier indicates a source of code used to define each class of said one or more classes.	The *Goldstein* reference discloses "the computer readable medium of claim 11, wherein said code identifier indicates a source of code used to define each class of said one or more classes." The '447 Patent discloses that "A 'source of code' is an entity from which computer instructions are received." '447 Patent, 3:15-17. As described above, *Goldstein* discloses a digital signature corresponding to a private/public key pair (e.g., a code identifier). This digital signature indicates the authority that signed the certificate, which *Goldstein* discloses as either a financial institution or a trust authority company: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is

U.S. Patent No. 6,125,447 – Claim 12	*Goldstein*
	embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc. * * * Some financial institutions will not want to do their own signing. These institutions will delegate Role signing to trust authority companies. These companies will make a business of validating the design and conformance of a cassette to the financial institutions specification." *Goldstein* at 13-14.

U.S. Patent No. 6,125,447 – Claim 13	*Goldstein*
13. The computer readable medium of claim 11, wherein said code identifier indicates a key associated with each class of said one or more classes.	The *Goldstein* reference discloses "the computer readable medium of claim 11, wherein said code identifier indicates a key associated with each class of said one or more classes." As described above, *Goldstein* discloses a digital signature corresponding to a private/public key pair (e.g., a code identifier). "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc." *Goldstein* at 13. The key in the cassette is associated with the classes in the cassette, as the JECF security policy is provided on a per-package basis: "Electronic commerce requires an extensible set of rights. Electronic commerce needs to identify a principal more precisely than the applet thread. Possible candidates include *threads* and *packages*. The Sandbox security manager can only control the rights invented by the Java system developers on a per thread basis. Electronic commerce also needs the

U.S. Patent No. 6,125,447 – Claim 13	*Goldstein*
	ability to delegate rights from one principal to another as long as the delegation follows the contract-specified business. This cannot be done by threads alone. In addition to threads, the Java language provides a construct called a *package*. Packages provide namespaces in the Java language. All classes in a package have access to package-private data members and methods. Package trees are primarily an organizational convention. Packages within a package tree have no special access. ***Packages are a natural choice for creating a security principal. Perhaps in a future version of Java, another security domain may emerge. Packages are a natural source of protection*.**" *Goldstein* at 10 (emphasis added).

U.S. Patent No. 6,125,447 – Claim 14	Goldstein
14. The computer readable medium of claim 11, wherein said code identifier indicates a source of code used to define each class of said one or more classes and . . .	The *Goldstein* reference discloses "the computer readable medium of claim 11, wherein said code identifier indicates a source of code used to define each class of said one or more classes and . . .." *Goldstein* expressly discloses that the digital signature (i.e., code identifier) may be associated with a financial institution (i.e., a source of code or "an entity from which computer instructions are received") that provides the cassette program: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc." *Goldstein* at 13.

U.S. Patent No. 6,125,447 – Claim 14	Goldstein
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 14 (showing digital signing that indicates the originating financial institution).
. . . [wherein said code identifier] indicates a key associated with each class of said one or more classes.	The *Goldstein* reference discloses “. . . [wherein said code identifier] indicates a key associated with each class of said one or more classes.” For example, the digital signature includes a public key that is associated with all of the classes in the cassette application: “The Role’s public key is embedded in the home banking, brokerage software or payment cassette. The financial institution’s cassette is then distributed over the Internet, via floppy disks, etc.” *Goldstein* at 13.

U.S. Patent No. 6,125,447 – Claim 15	*Goldstein*
15. The computer readable medium of	The *Goldstein* reference discloses “the computer readable medium of claim 14, wherein the

U.S. Patent No. 6,125,447 – Claim 15	*Goldstein*
claim 14, wherein the step of associating said one or more protection domains and said one or more classes based on said code identifier further includes associating said one or more protection domains and said one or more classes based on data persistently stored,	step of associating said one or more protection domains and said one or more classes based on said code identifier further includes associating said one or more protection domains and said one or more classes based on data persistently stored." For example, the financial institution may establish the association between the protection domain and the classes of the cassette (e.g., when defining or creating the Roles of a cassette and including those Roles in the cassette's package). This association is made based on a public/private key pair (i.e., the code identifier) that is persistently stored within the code of the cassette: "But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. The *Shah* further clarifies that Roles are stored (e.g., persistently) in a local database of the cassette: "When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2.
wherein said data associates code identifiers with a set of one or more permissions.	The *Goldstein* reference discloses "wherein said data associates code identifiers with a set of one or more permissions." As described above, the persistently stored data in *Goldstein* includes digital signatures (i.e., code identifiers) and Roles (i.e., objects that correspond to the protection domain, or set of rights, in a given cassette). *Goldstein* discloses that the Roles are implementations of the trust relationship: "The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship

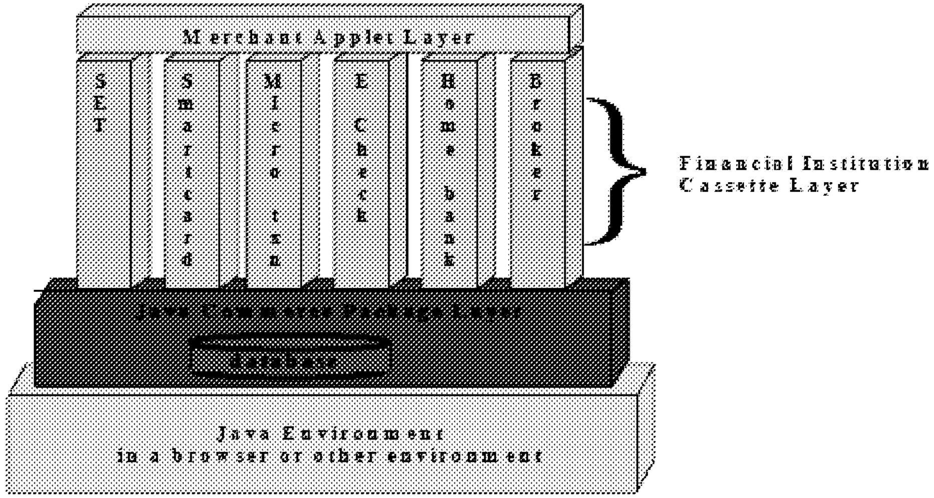
U.S. Patent No. 6,125,447 – Claim 15	*Goldstein*
	between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. *Shah* clarifies that Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2. Accordingly, the stored data associates the digital signatures (i.e., code identifier) and Roles which represent one ore more permissions.

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
16. A computer-readable medium carrying one or more sequences of one or more instructions, wherein the execution of the one or more sequences of the one or more instructions causes the one or more processors to perform the steps of:	The *Goldstein* reference discloses "a computer-readable medium carrying one or more sequences of one or more instructions, wherein the execution of the one or more sequences of the one or more instructions causes the one or more processors to perform the steps of." For example, the '447 defines computer-readable medium as: "The term 'computer-readable medium' as used herein refers to any medium that participates in providing instructions to processor 104 for execution. Such a medium may take many forms, including but not limited to, non-volatile media, volatile media, and transmission media. Non-volatile media includes, for example, optical or magnetic disks,

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	such as storage device 110. Volatile media includes dynamic memory, such as main memory 106. Transmission media includes coaxial cables, copper wire and fiber optics, including the wires that comprise bus 102. Transmission media can also take the form of acoustic or light waves, such as those generated during radio-wave and infra-red data communications. Common forms of computer-readable media include, for example, a floppy disk, a flexible disk, hard disk, magnetic tape, or any other magnetic medium, a CD-ROM, any other optical medium, punchcards, papertape, any other physical medium with patterns of holes, a RAM, a PROM, and EPROM, a FLASH-EPROM, any other memory chip or cartridge, a carrier wave as described hereinafter, or any other medium from which a computer can read." '447 Patent at 5:4-25. Consistent with this definition of "computer-readable media" in the '447 patent, *Goldstein* discloses software that is run on personal computers: "Sun Microsystems Inc.'s Java programming language and APIs provide a reliable, secure platform to deliver applications on Internet and Intranet networks. Without Java, users who downloaded applications from these networks could be vulnerable to dangerous software viruses. All of the popular personal computer operating systems are vulnerable to such attacks. Of course, Java executing on conventional personal computers cannot totally prevent virus attacks." *Goldstein* at 1. *Goldstein* discloses specific cassette applications software as part of the JECF that are implemented in software and distributed to end user's over the internet (i.e., computer-readable medium carrier wave): "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc." *Goldstein* at 13.

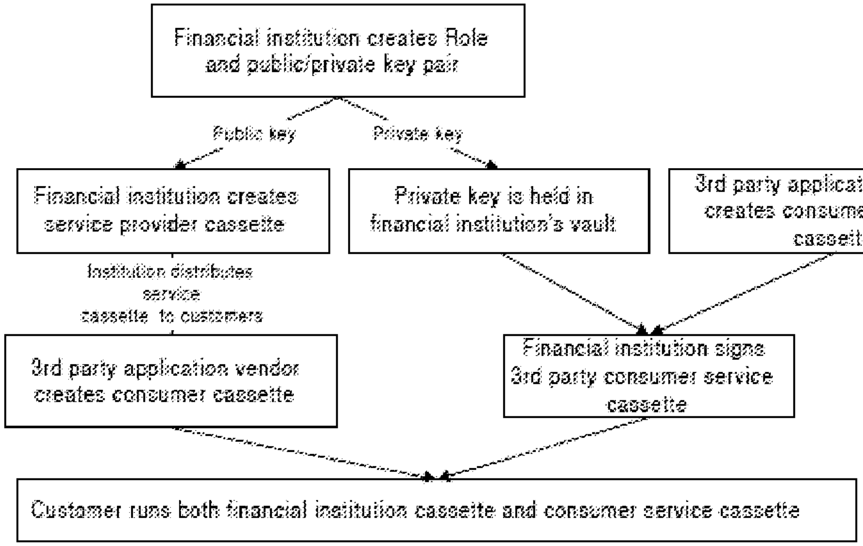
U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions;	The *Goldstein* reference discloses "establishing one or more protection domains, wherein a protection domain is associated with zero or more permissions." The '447 Patent discloses that "A protection domain can be viewed as a set of permissions granted to one or more principals." '447 Patent, 8:42-43. ("A 'principal' is an entity in the computer system to which permissions are granted. Examples of principals include processes, objects and threads." '447 Patent, 2:25-27.) As discussed more thoroughly below, *Goldstein* discloses establishing a set of permissions that are granted to an application called a "cassette." A set of permissions (protection domain) is first established by a financial institution or third party company at the time a cassette is created. (*Goldstein* at 7-8, 10, 13-14 & Figs. 1, 5-6.) These permissions are represented by "Roles" that "reify [i.e., implement] the trust relationship between two business entities." (*Goldstein* at 13.) The Roles are objects that represent and/or contain the specific authorizations (e.g., permissions) for a particular cassette as well as a digital signature (based on a public/private key pair) corresponding to the creator of the cassette. (*Id.*) After the cassette application is created, it is distributed to end users. (*Goldstein* at 14.) A user installs the cassette on a computer, and subsequently the cassette (including the associated Roles objects that correspond to the set of permissions for the cassette) is loaded by a Java applet class named "CassetteLoader" which communicates with a "CassetteInstallation" object to facilitate the load operation. (*Id.*) Accordingly, establishing the protection domain, or set of rights associated with the cassette, may be thought to occur at any of the following times, as disclosed in *Goldstein*: (1) by the financial institution when it defines the Roles or permissions for an institution cassette, (2) by a third party company when it defines the Roles or permissions for a third-party cassette, or (3) by the CassetteLoader applet when it loads a given cassette (including loading and creating the corresponding Roles) in the run-time environment. Further details of the above-described system are provided below.

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	Goldstein discloses establishing a protection domain as a set of permissions (e.g., “rights”) that are used by a client application (e.g., a principal or “body of code”): “To completely understand what is occurring, it is necessary to define a few terms: *rights* and *principals*. In this paper, a *right* is an abstract privilege. A *principal* uses a right. A principal can be a person, a corporation, a program, or a body of code.” *Goldstein* at 10. “[The JECF] allows secure applications, such as those used in electronic commerce, to safely exchange data and interoperate without compromising each individual application’s security.” *Goldstein* at 1. The rights in the JECF are based on the Capabilities model, which provides for the association of rights with object-oriented structures so that rights can be transferred from one principal to another: “Electronic commerce needs an object-oriented security model where rights can be transferred from one principal to another. The *Capabilities model,* originally defined by Dennis and Van Horn, provides such a model. Although originally based on hardware protection, Wulf and a team at Carnegie-Mellon University[WCCJRPP74] adapted Capabilities to object-oriented technology. The object-oriented definition of Capabilities is very simple. The Capabilities model means that possession of an object confers the right to use it. Capabilities need a gatekeeper service to decide whether to grant a right to the object. * * * Adding the Capabilities model to Java has proven to be simple because Java already has many of the necessary features. Java objects are unforgeable. Java’s language visibility rules for private and package-private scope provide address space-like protection against unauthorized access. The only needed component is some form of authorization mechanism to allow or deny access. In the JECF, individual Capabilities are called *permits*. Permit objects require careful handling. Permit objects should never be stored in a public or protected variable. Permits are rarely the actual implementation of some right, but are

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	typically implemented by another object. The permit forwards the call to the actual implementation." *Goldstein* at 10-11. *Goldstein* discloses client applications as "cassettes" that may reside on a user's system: "Many object-oriented practitioners use the term 'framework' to mean an 'application programming interface.' In this paper, the term 'framework' refers to a set of APIs that impose responsibilities amongst participating software packages. The JECF defines application responsibilities for merchants and financial institutions. JECF provides services, such as database services, to merchant and financial institution applications. These layers are depicted in Figure 1.  Merchant Applet Layer SET Smartcard Micro txn E Check Home bank Broker Financial Institution Cassette Layer Java Commerce Package Layer Database Java Environment in a browser or other environment **Figure 1. Simple Schema of JECF** * * * The *Cassette Layer* implements long term customer relationships such as credit cards, home banking and brokerages. Cassettes are a new feature that JECF adds to Java. ***Similar to applets, cassettes are downloaded from servers to client computers. Unlike applets which disappear when users quit the browser, cassettes are retained on the***

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	customer's system. Cassettes store information in a database provided by the JECF. Cassettes may safely store valuable information such as public key certificates and transaction records since the entire database is encrypted. Cassettes provide long term customer-to-institution relationships. Examples of cassettes include SET certificates and protocols, home banking, brokerage accounts, financial analysis, and planning software. Cassettes contain code, digital certificates, GIF images and other resources. Financial institutions will use cassettes to deliver customer service features. Smartcard application developers can put smartcard reader device drivers and application user interfaces in cassettes." *Goldstein* at 7-8 (emphasis added). *Goldstein* discloses examples of the types of rights (permissions) that may be associated with a cassette application: "Potentially dangerous operations include opening a file or a network socket, changing a system variable, and setting the security manager." *Goldstein* at 7. *Goldstein* discloses the use of objects in the "Roles" class that correspond to the protection domain, or set of rights, in a given cassette: "The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. To the extent *Goldstein* fails to explicitly disclose the exact contents of Roles, a person of ordinary skill in the art would have understood that the Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application. For example, *Shah* provides: "When the applet is loaded, the Class Loader object is set to execute in a limited

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2. Accordingly, the *Shah* discloses features that are necessarily present in the JECF system disclosed by *Goldstein*. *Goldstein* discloses that cassettes, and the Roles associated with an individual cassette, may be created by the financial institution (in the case of a cassette provided by the institution) or by a third party company. As described in the *Shah*, *Goldstein* discloses that the Roles and a digital signature (based on a public/private key pair) are created and stored in a cassette that is eventually distributed to the end user: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc. When third party companies want to gain access to the financial institution's cassette, they first sign a contract with the financial institution. The institution will typically impose certain usage rules about the data and Capabilities of the cassette that are specified in the contract. The institution then signs the third party cassette which is then distributed over the Internet, via floppy disks, or any other means desired. This process of code signing is illustrated in Figure 6."

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 13-14. In the distributed cassette application, *Goldstein* discloses that "an extensible set of rights" (e.g., a protection domain) may be established on a per-package basis or, "in a future version of Java, another security domain may emerge": "Electronic commerce requires an extensible set of rights. Electronic commerce needs to identify a principal more precisely than the applet thread. Possible candidates include *threads* and *packages*. The Sandbox security manager can only control the rights invented by the Java system developers on a per thread basis. Electronic commerce also needs the ability to delegate rights from one principal to another as long as the delegation follows the contract-specified business. This cannot be done by threads alone. In addition to threads, the Java language provides a construct called a *package*. Packages provide namespaces in the Java language. All classes in a package have access to

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	package-private data members and methods. Package trees are primarily an organizational convention. Packages within a package tree have no special access. ***Packages are a natural choice for creating a security principal. Perhaps in a future version of Java, another security domain may emerge. Packages are a natural source of protection*.” *Goldstein* at 10 (emphasis added). *Goldstein* discloses that a user may install the cassettes on a computer using the CassetteLocator applet in conjunction with the CassetteInstallation object in the JECF. At this time, the protection domain is established in the run-time environment: “Installation of cassettes is initiated by a Java applet class named CassetteLocator. CassetteLocator applets are dynamically loaded from financial institutions and other software suppliers. CassetteLocators communicate with a CassetteInstallation object in the JECF to negotiate loading of cassettes.” *Goldstein* at 14.
establishing an association between said one or more protection domains and one or more sources of code; and	The *Goldstein* reference discloses “establishing an association between said one or more protection domains and one or more sources of code.” For example, the ‘447 Patent discloses a “source of code” as “an entity from which computer instructions are received.” ‘447 Patent, 3:15-19. Similar to this disclosure of the “source of code” in the ‘447 Patent, *Goldstein* discloses the Roles objects (i.e., protection domain data structure), which explicitly associates the protection domain with a “source of code” such as the signer of a digital certificate: “The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant.” *Goldstein* at 13.

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	"Roles represent the signature and are used to check Tickets." *Goldstein* at 15. "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair." *Goldstein* at 13. As described above, the *Shah* further clarifies that Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2.
in response to executing code making a request to perform an action, determining whether said request is permitted based on a source of said code making said request and said association between said one or more protection domains and said one or more sources of code.	The *Goldstein* reference discloses "in response to executing code making a request to perform an action, determining whether said request is permitted based on a source of said code making said request and said association between said one or more protection domains and said one or more sources of code." For example, *Goldstein* discloses using the digital signature of a cassette (i.e., indicating the source of code) and an associated Roles object (i.e., protection domain data structure) to determine whether an action is permitted. In one particular disclosed example, the Gate checks a Role to determine if the cassette requesting authorization to read a JECF database is permitted: "For example, in the JECF database, the ability to read a row in a table and the ability to write a row in the table are both implemented in a private class named TableImplemenation. Access to the implementation for TableImplemenation from outside the implementation is granted by a pair of objects named ReadTablePermit and WriteTablePermit." *Goldstein* at

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	11.

"Gaining access to a permit is done via a pattern called a *Gate*. A Gate is a specialized form of *factory* pattern method. A Gate decides whether to grant an instance of a Permit to a caller or deny access by throwing an exception." *Goldstein* at 12.

"The JECF needs a token that represents the actual consumer of the permit. Figure 5 describes in more detail the interaction and call structure of creating the objects so the proper cassette will be validated. This token is passed from the client code into the gate method. In the JECF, this token is of class Ticket. Thus the final form of the Gateway check code is:

```
class Database {
   TablePermit createReadTablePermit(Ticket t) throws SecurityException {
      if ( t.stamp(ReadTableRole) )
         return new TablePermit();
      else
         throw new SecurityException();
   }
}"
```

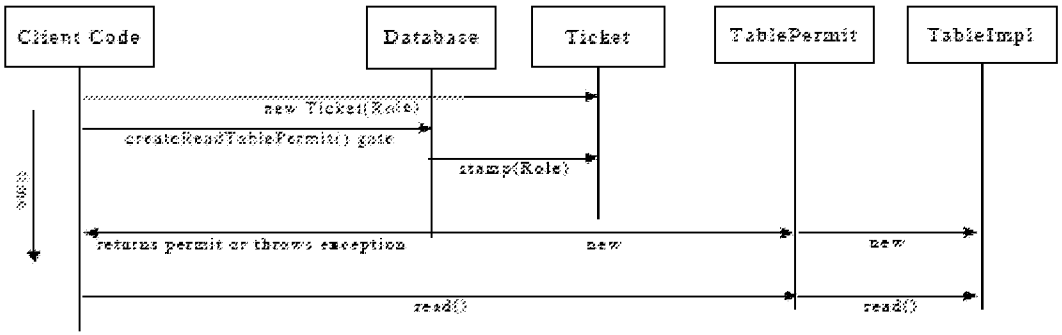


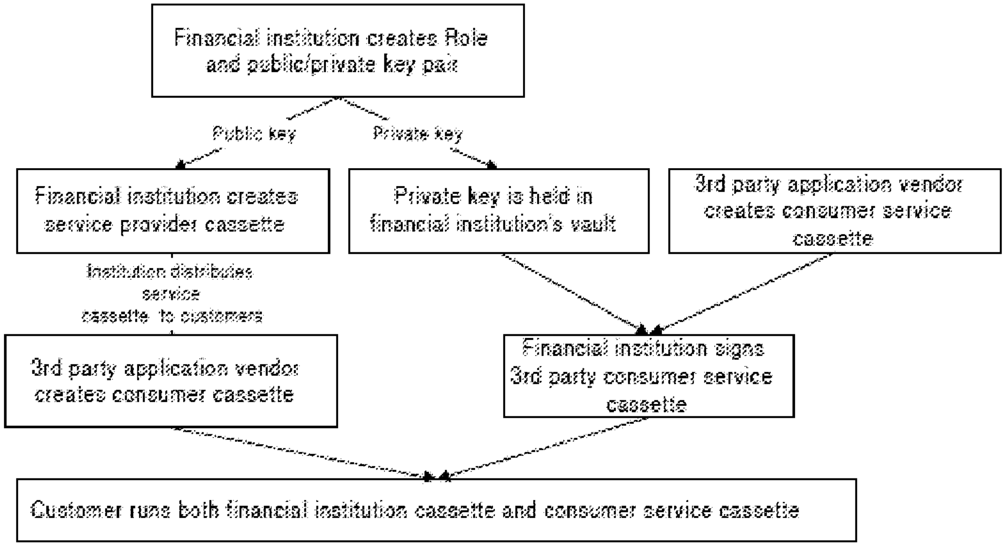
Figure 5. Interaction diagram for Table permits

Goldstein at 12-13 (citations omitted).

In other words, the Gate (e.g., the createReadTablePermit() method above) determines if the

U.S. Patent No. 6,125,447 – Claim 16	*Goldstein*
	calling object (e.g. of the cassette application) is permitted to read the database table. This determination is made based on the digital signature (i.e., indicating the source of code) and an associated Roles object (i.e., a set of permissions, or protection domain data structure). As shown in Figure 5 (and accompanying text at 12-13), the Role is passed to the Gate via the Ticket object. The Gate verifies the permission before sending a Permit object back to the calling object in the cassette application.

U.S. Patent No. 6,125,447 – Claim 17	*Goldstein*
17. The computer readable medium of claim 16, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code further includes establishing an association between said one or more protection domains and said one or more sources of code and one or more keys associated with said one or more sources of code.	The *Goldstein* reference discloses "the computer readable medium of claim 16, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code further includes establishing an association between said one or more protection domains and said one or more sources of code and one or more keys associated with said one or more sources of code." As described above with respect to claim 7, the financial institution may establish the association between the protection domain (e.g., set of permissions implemented as Roles objects) and the source of code (e.g., the digital certificate), for example, when defining or creating the Roles for a cassette and including those Roles in the cassette's package. As *Goldstein*'s Figure 6 illustrates, this process uses a the public/private key pair, which identify the financial institution as the source of code:

U.S. Patent No. 6,125,447 – Claim 17	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 14 (top box showing association of protection domain (e.g., set of permissions implemented as Roles) and source of code (e.g., keys that identify financial institution as originator of code for cassette)).

U.S. Patent No. 6,125,447 – Claim 18	*Goldstein*
18. The computer readable medium of claim 17, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code and said one or more keys associated with said one or more sources of code further includes establishing said association between said one or more protection domains and said one or more sources	The *Goldstein* reference discloses "the computer readable medium of claim 17, wherein the step of establishing an association between said one or more protection domains and said one or more sources of code and said one or more keys associated with said one or more sources of code further includes establishing said association between said one or more protection domains and said one or more sources of code and said one or more keys associated with said one or more sources of code based on data persistently stored, wherein said data associates particular sources of code and particular keys with a set of one or more permissions." For example, the financial institution may establish the association between the protection

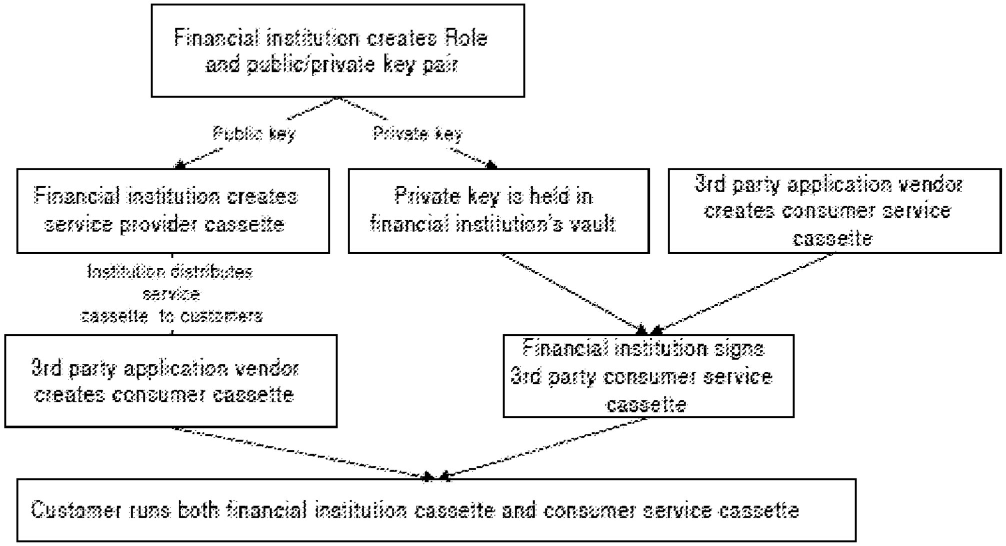
U.S. Patent No. 6,125,447 – Claim 18	*Goldstein*
of code and said one or more keys associated with said one or more sources of code based on data persistently stored, wherein said data associates particular sources of code and particular keys with a set of one or more permissions.	domain and the source of code (e.g., the digital certificate), for example, when defining or creating the Roles for a cassette and including those Roles in the cassette's package. This association is made based on a public/private key pair (i.e., the code identifier) that is persistently stored within the code of the cassette: "But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. The *Shah* further clarifies that Roles are stored (e.g., persistently) in a local database of the cassette: "When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2.

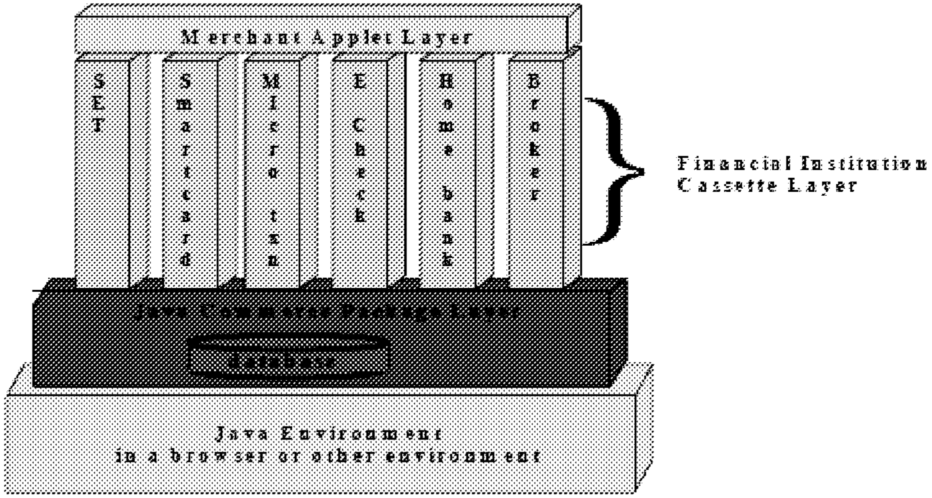
U.S. Patent No. 6,125,447 – Claim 19	*Goldstein*
19. A computer system comprising:	The *Goldstein* reference discloses "a computer system." *Goldstein* discloses software that is run on personal computers: "Sun Microsystems Inc.'s Java programming language and APIs provide a reliable, secure platform to deliver applications on Internet and Intranet networks. Without Java, users who downloaded applications from these networks could be vulnerable to dangerous software viruses. All of the popular personal computer operating systems are vulnerable to such attacks. Of course, Java executing on conventional personal computers cannot totally prevent virus attacks." *Goldstein* at 1.
a processor;	The *Goldstein* reference discloses "a processor."

U.S. Patent No. 6,125,447 – Claim 19	*Goldstein*
	A processor is inherent in a computer system. In addition, *Goldstein* discloses running Java directly on a processor: "The ultimate solution is an entirely Java operating system running directly on a processor." *Goldstein* at 1.
one or more protection domains stored as objects in said memory,	The *Goldstein* reference discloses "one or more protection domains stored as objects in said memory." The '447 Patent discloses that "A protection domain can be viewed as a set of permissions granted to one or more principals." '447 Patent, 8:42-43. ("A 'principal' is an entity in the computer system to which permissions are granted. Examples of principals include processes, objects and threads." '447 Patent, 2:25-27.) As discussed more thoroughly below, *Goldstein* discloses establishing a set of permissions that are granted to an application called a "cassette." A set of permissions (protection domain) is first established by a financial institution or third party company at the time a cassette is created. (*Goldstein* at 7-8, 10, 13-14 & Figs. 1, 5-6.) These permissions are represented by "Roles" that "reify [i.e., implement] the trust relationship between two business entities." (*Goldstein* at 13.) The Roles are objects that represent and/or contain the specific authorizations (e.g., permissions) for a particular cassette as well as a digital signature (based on a public/private key pair) corresponding to the creator of the cassette. (*Id.*) After the cassette application is created, it is distributed to end users. (*Goldstein* at 14.) A user installs the cassette on a computer, and subsequently the cassette (including the associated Roles objects that correspond to the set of permissions for the cassette) is loaded by a Java applet class named "CassetteLoader" which communicates with a "CassetteInstallation" object to facilitate the load operation. (*Id.*) Accordingly, establishing the protection domain, or set of rights associated with the cassette, may be thought to occur at any of the following times, as disclosed in *Goldstein*: (1) by the financial institution when it defines the Roles or permissions for an institution cassette, (2)

U.S. Patent No. 6,125,447 – Claim 19	*Goldstein*
	by a third party company when it defines the Roles or permissions for a third-party cassette, or (3) by the CassetteLoader applet when it loads a given cassette (including loading and creating the corresponding Roles) in the run-time environment. Further details of the above-described system are provided below. *Goldstein* discloses the use of objects in the "Roles" class that correspond to the protection domain, or set of rights, in a given cassette: "The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. To the extent *Goldstein* fails to explicitly disclose the exact contents of Roles, a person of ordinary skill in the art would have understood that the Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application. For example, *Shah* provides: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2. Accordingly, the *Shah* discloses features that are necessarily present in the JECF system disclosed by *Goldstein*. *Goldstein* discloses that cassettes, and the Roles associated with an individual cassette, may

U.S. Patent No. 6,125,447 – Claim 19	*Goldstein*
	be created by the financial institution (in the case of a cassette provided by the institution) or by a third party company. As described in the *Shah*, *Goldstein* discloses that the Roles and a digital signature (based on a public/private key pair) are created and stored in a cassette that is eventually distributed to the end user: “Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role’s public key is embedded in the home banking, brokerage software or payment cassette. The financial institution’s cassette is then distributed over the Internet, via floppy disks, etc. When third party companies want to gain access to the financial institution’s cassette, they first sign a contract with the financial institution. The institution will typically impose certain usage rules about the data and Capabilities of the cassette that are specified in the contract. The institution then signs the third party cassette which is then distributed over the Internet, via floppy disks, or any other means desired. This process of code signing is illustrated in Figure 6.”

U.S. Patent No. 6,125,447 – Claim 19	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 13-14. *Goldstein* discloses client applications as "cassettes" that may reside on a user's system: "Many object-oriented practitioners use the term 'framework' to mean an 'application programming interface.' In this paper, the term 'framework' refers to a set of APIs that impose responsibilities amongst participating software packages. The JECF defines application responsibilities for merchants and financial institutions. JECF provides services, such as database services, to merchant and financial institution applications. These layers are depicted in Figure 1.

U.S. Patent No. 6,125,447 – Claim 19	*Goldstein*
	 Merchant Applet Layer SET Smartcard Micro txn E Check Home bank Broker Financial Institution Cassette Layer Java Commerce Package Layer Database Java Environment in a browser or other environment **Figure 1. Simple Schema of JECF** * * * The *Cassette Layer* implements long term customer relationships such as credit cards, home banking and brokerages. Cassettes are a new feature that JECF adds to Java. ***Similar to applets, cassettes are downloaded from servers to client computers. Unlike applets which disappear when users quit the browser, cassettes are retained on the customer's system***. Cassettes store information in a database provided by the JECF. Cassettes may safely store valuable information such as public key certificates and transaction records since the entire database is encrypted. Cassettes provide long term customer-to-institution relationships. Examples of cassettes include SET certificates and protocols, home banking, brokerage accounts, financial analysis, and planning software. Cassettes contain code, digital certificates, GIF images and other resources. Financial institutions will use cassettes to deliver customer service features. Smartcard application developers can put smartcard reader device drivers and application user interfaces in cassettes." *Goldstein* at 7-8 (emphasis added). *Goldstein* discloses that a user may install the cassettes on a computer using the

U.S. Patent No. 6,125,447 – Claim 19	*Goldstein*
	CassetteLocator applet in conjunction with the CassetteInstallation object in the JECF. At this time, the protection domain is established in the run-time environment: "Installation of cassettes is initiated by a Java applet class named CassetteLocator. CassetteLocator applets are dynamically loaded from financial institutions and other software suppliers. CassetteLocators communicate with a CassetteInstallation object in the JECF to negotiate loading of cassettes." *Goldstein* at 14.
wherein each protection domain is associated with zero or more permissions;	The *Goldstein* reference discloses "wherein each protection domain is associated with zero or more permissions." *Goldstein* discloses establishing a protection domain as a set of permissions (e.g., "rights") that are used by a client application (e.g., a principal or "body of code"): "To completely understand what is occurring, it is necessary to define a few terms: *rights* and *principals*. In this paper, a *right* is an abstract privilege. A *principal* uses a right. A principal can be a person, a corporation, a program, or a body of code." *Goldstein* at 10. "[The JECF] allows secure applications, such as those used in electronic commerce, to safely exchange data and interoperate without compromising each individual application's security." *Goldstein* at 1. The rights in the JECF are based on the Capabilities model, which provides for the association of rights with object-oriented structures so that rights can be transferred from one principal to another: "Electronic commerce needs an object-oriented security model where rights can be transferred from one principal to another. The *Capabilities model,* originally defined by Dennis and Van Horn, provides such a model. Although originally based on hardware protection, Wulf and a team at Carnegie-Mellon University[WCCJRPP74] adapted Capabilities to object-oriented technology. The object-oriented definition of Capabilities is very simple. The Capabilities model means that possession of an object confers the right to

U.S. Patent No. 6,125,447 – Claim 19	*Goldstein*
	use it. Capabilities need a gatekeeper service to decide whether to grant a right to the object. * * * Adding the Capabilities model to Java has proven to be simple because Java already has many of the necessary features. Java objects are unforgeable. Java's language visibility rules for private and package-private scope provide address space-like protection against unauthorized access. The only needed component is some form of authorization mechanism to allow or deny access. In the JECF, individual Capabilities are called *permits*. Permit objects require careful handling. Permit objects should never be stored in a public or protected variable. Permits are rarely the actual implementation of some right, but are typically implemented by another object. The permit forwards the call to the actual implementation." *Goldstein* at 10-11. *Goldstein* discloses examples of the types of rights (permissions) that may be associated with a cassette application: "Potentially dangerous operations include opening a file or a network socket, changing a system variable, and setting the security manager." *Goldstein* at 7.
a domain mapping object stored in said memory, said domain mapping object establishing an association between said one or more protection domains and one or more classes of one or more objects; and	The *Goldstein* reference discloses "a domain mapping object stored in said memory, said domain mapping object establishing an association between said one or more protection domains and one or more classes of one or more objects." *Goldstein* discloses that a CassetteInstallation object in the JECF is used to load a cassette, thus establishing an association between a protection domain and one or more classes of one or more objects (i.e., classes that define the cassette) in the run-time environment: "Installation of cassettes is initiated by a Java applet class named CassetteLocator. CassetteLocator applets are dynamically loaded from financial institutions and other software suppliers. CassetteLocators communicate with a CassetteInstallation object in the JECF to negotiate loading of cassettes." *Goldstein* at 14.

<table>
<tr><th>U.S. Patent No. 6,125,447 – Claim 19</th><th><em>Goldstein</em></th></tr>
<tr><td></td><td></td></tr>
<tr><td>said processor being configured to determine whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or more classes.</td><td>

The *Goldstein* reference discloses “said processor being configured to determine whether an action requested by a particular object is permitted based on said association between said one or more protection domains and said one or more classes.”

For example, *Goldstein* discloses using the association between a cassette’s class and the set of permissions (i.e., protection domain) of that class (as represented by the Role objects) to determine whether an action is permitted. In one particular disclosed example, the Gate checks a Role to determine if the cassette requesting authorization to read a JECF database is permitted:

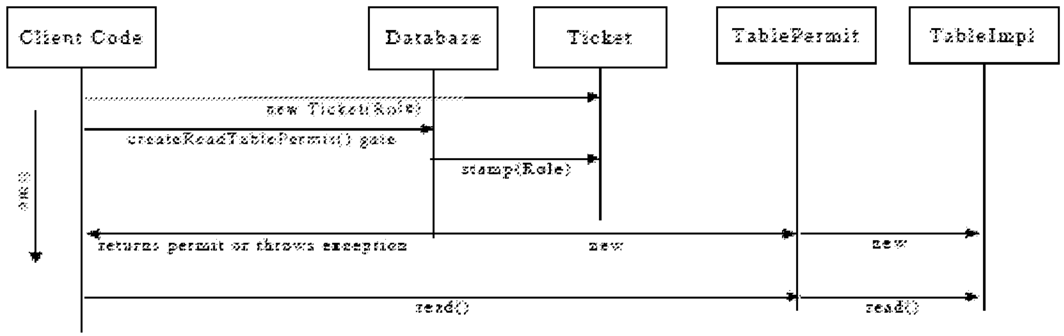
“For example, in the JECF database, the ability to read a row in a table and the ability to write a row in the table are both implemented in a private class named TableImplemenation. Access to the implementation for TableImplemenation from outside the implementation is granted by a pair of objects named ReadTablePermit and WriteTablePermit.” *Goldstein* at 11.

“Gaining access to a permit is done via a pattern called a *Gate*. A Gate is a specialized form of *factory* pattern method. A Gate decides whether to grant an instance of a Permit to a caller or deny access by throwing an exception.” *Goldstein* at 12.

“The JECF needs a token that represents the actual consumer of the permit. Figure 5 describes in more detail the interaction and call structure of creating the objects so the proper cassette will be validated. This token is passed from the client code into the gate method. In the JECF, this token is of class Ticket. Thus the final form of the Gateway check code is:

```
class Database {
   TablePermit createReadTablePermit(Ticket t) throws SecurityException {
      if ( t.stamp(ReadTableRole) )
         return new TablePermit();
      else
         throw new SecurityException();
   }
}”
```

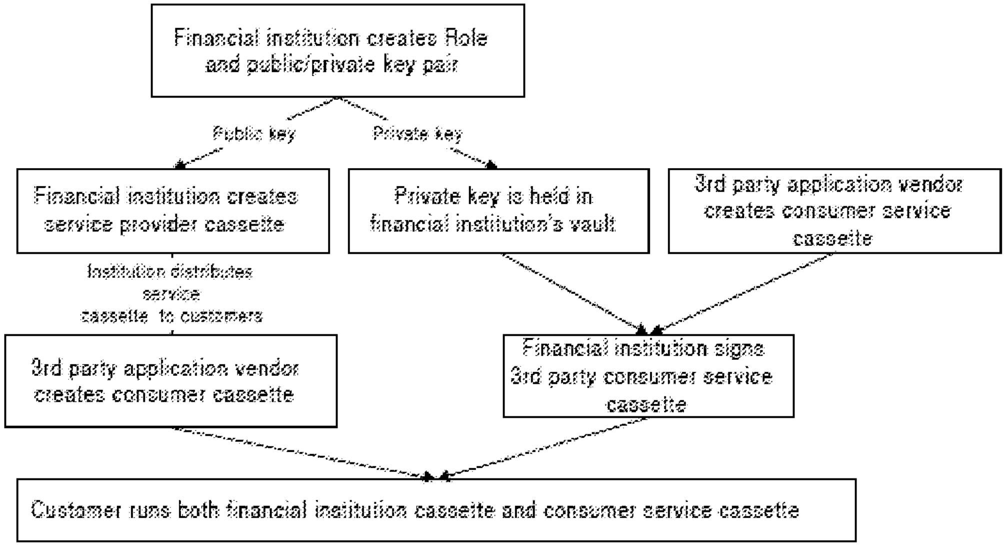
</td></tr>
</table>

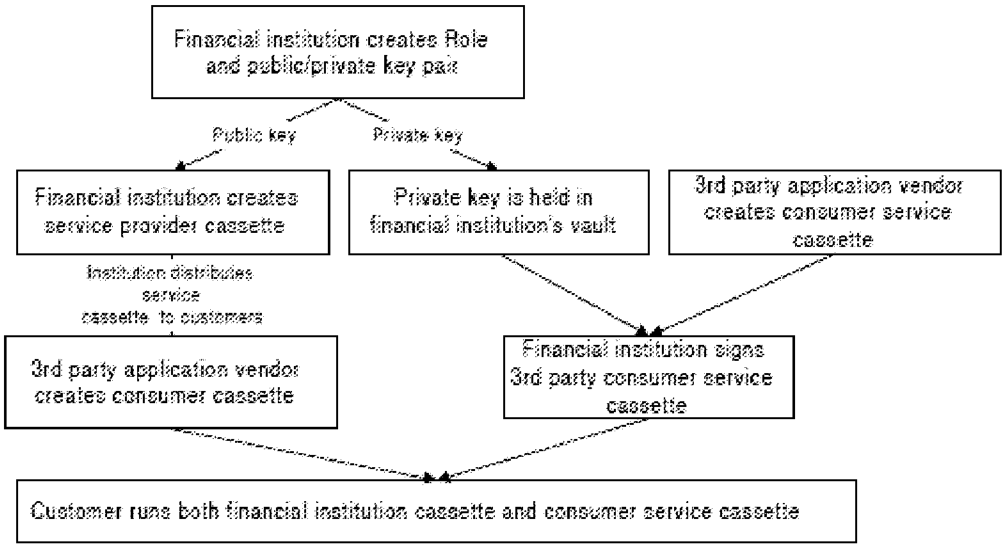
U.S. Patent No. 6,125,447 – Claim 19	*Goldstein*
	 Client Code Database Ticket TablePermit TableImpl time new Ticket(Role) createReadTablePermit() gate stamp(Role) returns permit or throws exception new new read() read() **Figure 5. Interaction diagram for Table permits** *Goldstein* at 12-13 (citations omitted). In other words, the Gate (e.g., the createReadTablePermit() method above) determines if the calling object (e.g. of the cassette application) is permitted to read the database table. This determination is made based on the association between the protection domain (e.g., a set of permissions represented by the Role that is passed to the Gate via the Ticket object) and the class of the calling object (e.g., of the cassette application).

U.S. Patent No. 6,125,447 – Claim 20	*Goldstein*
20. The computer system of claim 19, wherein:	The *Goldstein* reference discloses “the computer system of claim 19.” *See* chart for claim 19 above.
at least one protection domain of said one or more protection domains is associated with a code identifier;	The *Goldstein* reference discloses “at least one protection domain of said one or more protection domains is associated with a code identifier.” For example, the ‘447 Patent discloses a code identifier as “describing the source of code that defines a class, a set of public cryptographic keys associated with the source of code, or other information which describes the source of code, or any combination thereof. A ‘source of code’ is an entity from which computer instructions are received.” ‘447 Patent,

U.S. Patent No. 6,125,447 – Claim 20	*Goldstein*
	3:13-19. Figure 3 of the '447 Patent discloses a policy file that includes a URL (i.e., file://somesource) and a key name (i.e., "somekey"), and describes both as code identifiers. '447 Patent, 9:26-37 & Fig. 3. Similar to this disclosure of the "code identifier" in the '447 Patent, *Goldstein* discloses a Role (i.e., a set of permissions or protection domain) associated with a digital signature corresponding to a public key/private key pair (e.g., a code identifier). "The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. "Roles represent the signature and are used to check Tickets." *Goldstein* at 15. "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair." *Goldstein* at 13. As described above, the *Shah* further clarifies that Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2.

U.S. Patent No. 6,125,447 – Claim 20	*Goldstein*
at least one class of said one or more classes is associated with said code identifier; and	*Goldstein* discloses at least one class of said one or more classes is associated with said code identifier. For example, as discussed above, Figures 2 and 3C of *Goldstein* disclose that the PAI data structure may contain a manufacturer's signature (i.e., code identifier) (*see* Fig. 2), and that the PAI with the code identifier is associated with the object/class data structure because it is expressly included as part of the object/class data structure. For example, a cassette application, including all of the classes that make up the application, are associated with the code identifier because the cassette is digitally signed: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc. When third party companies want to gain access to the financial institution's cassette, they first sign a contract with the financial institution. The institution will typically impose certain usage rules about the data and Capabilities of the cassette that are specified in the contract. The institution then signs the third party cassette which is then distributed over the Internet, via floppy disks, or any other means desired. This process of code signing is illustrated in Figure 6."

U.S. Patent No. 6,125,447 – Claim 20	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 13-14 (illustrating association between public/private key pair and cassettes, which are signed by the originating financial institution).
said computer system further comprises said processor configured to establish an association between said one or more protection domains and said one or more classes of one or more objects by associating said one or more protection domains and said one or more classes based on said code identifier.	The *Goldstein* reference discloses "said computer system further comprises said processor configured to establish an association between said one or more protection domains and said one or more classes of one or more objects by associating said one or more protection domains and said one or more classes based on said code identifier." As described above with respect to claim 1, the financial institution may establish the association between the protection domain and the classes of the cassette (e.g., when defining or creating the Roles of a cassette and including those Roles in the cassette's package). As *Goldstein*'s Figure 6 illustrates, this process is done based on the public/private key pair (i.e., the code identifier).

U.S. Patent No. 6,125,447 – Claim 20	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes**

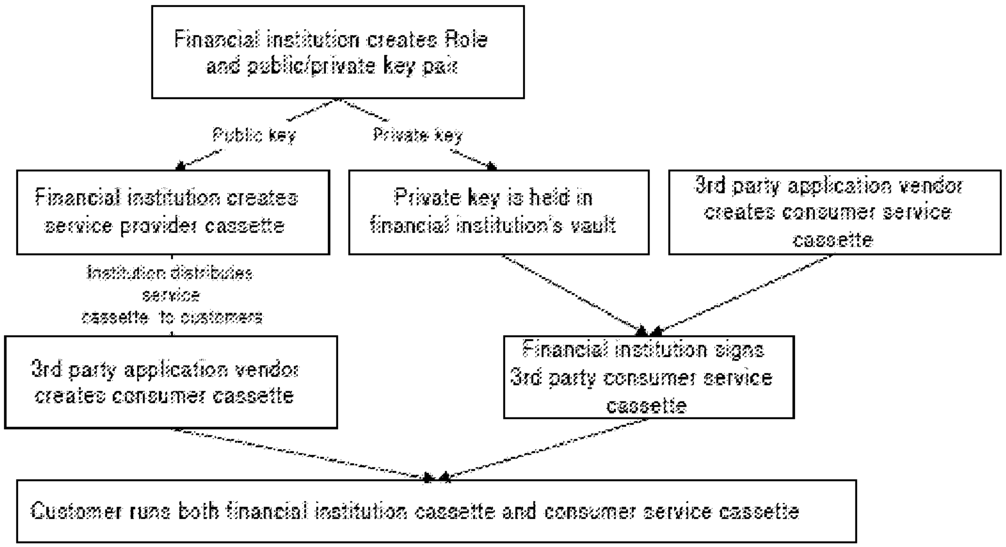
U.S. Patent No. 6,125,447 – Claim 21	*Goldstein*
21. The computer system of claim 20, wherein said code identifier indicates a source of code used to define each class of said one or more classes.	The *Goldstein* reference discloses "the computer system of claim 20, wherein said code identifier indicates a source of code used to define each class of said one or more classes." The '447 Patent discloses that "A 'source of code' is an entity from which computer instructions are received." '447 Patent, 3:15-17. As described above, *Goldstein* discloses a digital signature corresponding to a private/public key pair (e.g., a code identifier). This digital signature indicates the authority that signed the certificate, which *Goldstein* discloses as either a financial institution or a trust authority company: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial

U.S. Patent No. 6,125,447 – Claim 21	*Goldstein*
	institution's cassette is then distributed over the Internet, via floppy disks, etc. * * * Some financial institutions will not want to do their own signing. These institutions will delegate Role signing to trust authority companies. These companies will make a business of validating the design and conformance of a cassette to the financial institutions specification." *Goldstein* at 13-14.

U.S. Patent No. 6,125,447 – Claim 22	*Goldstein*
22. The computer system of claim 20, wherein said code identifier indicates a key associated with each class of said one or more classes.	The *Goldstein* reference discloses "the computer system of claim 20, wherein said code identifier indicates a key associated with each class of said one or more classes." As described above, *Goldstein* discloses a digital signature corresponding to a private/public key pair (e.g., a code identifier). "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc." *Goldstein* at 13. The key in the cassette is associated with the classes in the cassette, as the JECF security policy is provided on a per-package basis: "Electronic commerce requires an extensible set of rights. Electronic commerce needs to identify a principal more precisely than the applet thread. Possible candidates include *threads* and *packages*. The Sandbox security manager can only control the rights invented by the Java system developers on a per thread basis. Electronic commerce also needs the ability to delegate rights from one principal to another as long as the delegation follows

U.S. Patent No. 6,125,447 – Claim 22	*Goldstein*
	the contract-specified business. This cannot be done by threads alone. In addition to threads, the Java language provides a construct called a *package*. Packages provide namespaces in the Java language. All classes in a package have access to package-private data members and methods. Package trees are primarily an organizational convention. Packages within a package tree have no special access. ***Packages are a natural choice for creating a security principal. Perhaps in a future version of Java, another security domain may emerge. Packages are a natural source of protection*.**" *Goldstein* at 10 (emphasis added).

U.S. Patent No. 6,125,447 – Claim 23	*Goldstein*
23. The computer system of claim 20, wherein said code identifier indicates a source of code used to define each class of said one or more classes and	The *Goldstein* reference discloses the computer system of claim 20, wherein said code identifier indicates a source of code used to define each class of said one or more classes. *Goldstein* expressly discloses that the digital signature (i.e., code identifier) may be associated with a financial institution (i.e., a source of code or "an entity from which computer instructions are received") that provides the cassette program: "Anyone (with the appropriate tool) can create a Role. It is essentially a public and private key pair. Every Role creator is the top of its own certificate authority. All the usual procedures about keeping the private key a secret apply. The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc." *Goldstein* at 13.

U.S. Patent No. 6,125,447 – Claim 23	*Goldstein*
	 Financial institution creates Role and public/private key pair Public key Private key Financial institution creates service provider cassette Private key is held in financial institution's vault 3rd party application vendor creates consumer service cassette Institution distributes service cassette to customers 3rd party application vendor creates consumer cassette Financial institution signs 3rd party consumer service cassette Customer runs both financial institution cassette and consumer service cassette **Figure 6. Distribution of signatures among cassettes** *Goldstein* at 14 (showing digital signing that indicates the originating financial institution).
indicates a key associated with each class of said one or more classes.	The *Goldstein* reference discloses a code identifier that indicates a key associated with each class of said one or more classes. For example, the digital signature includes a public key that is associated with all of the classes in the cassette application: "The Role's public key is embedded in the home banking, brokerage software or payment cassette. The financial institution's cassette is then distributed over the Internet, via floppy disks, etc." *Goldstein* at 13.

U.S. Patent No. 6,125,447 – Claim 24	*Goldstein*
24. The computer system of claim 20,	The *Goldstein* reference discloses "the computer system of claim 20, further comprising said

U.S. Patent No. 6,125,447 – Claim 24	*Goldstein*
further comprising said processor configured to associate said one or more protection domains and said one or more classes based on said code identifier by associating said one or more protection domains and said one or more classes based on data persistently stored in said computer system,	processor configured to associate said one or more protection domains and said one or more classes based on said code identifier by associating said one or more protection domains and said one or more classes based on data persistently stored in said computer system." For example, the financial institution may establish the association between the protection domain and the classes of the cassette (e.g., when defining or creating the Roles of a cassette and including those Roles in the cassette's package). This association is made based on a public/private key pair (i.e., the code identifier) that is persistently stored within the code of the cassette: "But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. The *Shah* further clarifies that Roles are stored (e.g., persistently) in a local database of the cassette: "When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2.
wherein said data associates code identifiers with a set of one or more permissions.	The *Goldstein* reference discloses "wherein said data associates code identifiers with a set of one or more permissions." As described above, the persistently stored data in *Goldstein* includes digital signatures (i.e., code identifiers) and Roles (i.e., objects that correspond to the protection domain, or set of rights, in a given cassette). *Goldstein* discloses that the Roles are implementations of the trust relationship: "The new element in this example is class Role. The class ReadTableRole provides a representation of the business relationship for this object. Roles reify the trust relationship

U.S. Patent No. 6,125,447 – Claim 24	*Goldstein*
	between two business entities. The JECF uses digital signatures to represent roles. This is the same mechanism that the Java sandbox security model uses. But instead of installing the digital signature into the Sandbox database, the digital signature is embedded in the code as a constant." *Goldstein* at 13. *Shah* clarifies that Roles contain information regarding the permissions (or authorization) of the cassette (invoking program) and that Roles are digitally signed by the originator of the cassette application: "When the applet is loaded, the Class Loader object is set to execute in a limited environment and calling programs are checked for their digital signature for uniqueness. When a JECF object is invoked, the invoking program or application is checked for its role. These roles dictate the available resources and security levels and control that program's interface to the JECF code. A local database contains these access control lists and role information. Each Payment and Service cassette with its specific roles must be signed by a trusted authority before use to guarantee the identity of the originator." *Shah* at 2. Accordingly, the stored data associates the digital signatures (i.e., code identifier) and Roles which represent one ore more permissions.