Secure transmission of data is provided between a plurality of computer systems over a public communication system, such as the Internet. Secure transmission of data is provided from a customer computer system to a merchant computer system, and for the further secure transmission of data from the merchant computer system to a payment gateway computer system. The payment gateway system evaluates the information and returns authorization or denial of credit via a secure transmission to the merchant which is communicated to the customer by the merchant.
**FIG. 1A**

**FIG. 1B**
FIG. 2
FIG. 3
CREATE AUTHORIZATION REQUEST

ADD MERCHANTS: ENCRYPTION CERT. SIGNATURE CERT. DIGITAL SIG.

GENERATE RANDOM KEY RK-0

ENCRYPT WITH RANDOM KEY RK-0

ENCRYPT RAND. KEY WITH P.G. PUBLIC KEY

CONCATENATE ENCRYPTED REQUEST AND ENCRYPTED RANDOM KEY RK-0

TRANSMIT

FIG. 4
FIG. 7A

FIG. 7B

FIG. 7C

FIG. 7D

FIG. 7E

FIG. 7F

FIG. 7G

FIG. 7H

FIG. 7I

FIG. 7J
FIG. 8

1. Decrypt RK-1
2. Decrypt combined block
3. Verify P.G. certificate
4. Validate P.G. signature
5. Store combined encrypted capture token & encrypted RK-2
6. Process customer purchase request

FIG. 9

1. Merchant capture request
2. Merchant capture response
3. Process capture request
CREATE CAPTURE REQUEST

ADD MERCHANTS: ENCRYPTION CERT SIGNATURE CERT DIGITAL SIG

GENERATE RANDOM KEY RK-3

ENCRYPT WITH RANDOM KEY RK-3

ENCRYPT RANDOM KEY RK-3 WITH P.G. PUBLIC KEY

CONCATENATE ENCRYPTED CAPTURE REQUEST; ENCRYPTED RK-3; SAVED ENCRYPTED CAPTURE TOKEN; ENCRYPTED RK-2

TRANSMIT

DECRYPT RK-4

DECRYPT COMBINED BLOCK

VERIFY P.G. CERTIFICATE

VALIDATE P.G. SIG

STORE CAPTURE RESPONSE

FIG. 14

FIG. 10
DECRYPT RANDOM KEY RK-3

DECRYPT BLOCK

VERIFY MERCHANT CERTIFICATES

VALIDATE DIGITAL SIGNATURE

DECRYPT RANDOM KEY RK-2

DECRYPT CAPTURE TOKEN

VERIFY CONSISTENCY BETWEEN CAPTURE REQUEST AND CAPTURE TOKEN

TRIGGER FUND TRANSFER

CREATE BASIC CAPTURE RESPONSE

ADD PG'S: SIG CERTIFICATE DIGITAL SIG

GENERATE RK-4

ENCRYPT WITH RK-4

ENCRYPT RK-4 WITH MERCHANT PUBLIC KEY

CONCATENATE

TRANSMIT

FIG. 12B

FIG. 12A
SYSTEM, METHOD AND ARTICLE OF MANUFACTURE FOR SECURE NETWORK ELECTRONIC PAYMENT AND CREDIT COLLECTION

FIELD OF THE INVENTION

[0001] The present invention relates to the electronic payment in exchange for goods and services purchased over a communication network, and more specifically, and more particularly, to a system, method and article of manufacture for securely transmitting payment information from a customer to a merchant to a payment gateway and returning appropriate, secure authorization to the merchant and the customer.

BACKGROUND OF THE INVENTION

[0002] It is desirable for a computer operated under the control of a merchant to obtain information offered by a customer and transmitted by a computer operating under the control of the customer over a publicly accessible packet-switched network (e.g., the Internet) to the computer operating under the control of the merchant, without risking the exposure of the information to interception by third parties that have access to the network, and to assure that the information is from an authentic source. It is further desirable to have the ability for the merchant to transmit information, including a subset of the information provided by the customer, over such a network to a payment gateway computer system that is authorized, by a bank or other financial institution that has the responsibility of providing payment on behalf of the customer, to authorize a commercial transaction on behalf of such a financial institution, without the risk of exposing that information to interception by third parties. Such institutions include, for example, financial institutions offering credit or debit card services.

[0003] One such attempt to provide such a secure transmission channel is a secure payment technology such as Secure Electronic Transaction (hereinafter “SET”), jointly developed by the Visa and MasterCard card associations, and described in Visa and MasterCard’s “Secure Electronic Transaction (SET) Specification,” Feb. 23, 1996, hereby incorporated by reference. Other such secure payment technologies include Secure Transaction Technology (“STT”), Secure Electronic Payments Protocol (“SEPP”), Internet Keyed Payments (“IKP”), Net Trust, and Cybercash Credit Payment Protocol. One of ordinary skill in the art will readily comprehend that any of the secure payment technologies can be substituted for the SET protocol without undue experimentation. Such secure payment technologies require the customer to operate software that is compliant with the secure payment technology, interacting with third-party certification authorities, thereby allowing the customer to transmit encoded information to a merchant, some of which may be decoded by the merchant, and some which can be decoded only by a payment gateway specified by the customer. A drawback to the secure payment technology approach is that it requires deployment of special-purpose software compliant with the particular secure payment technology to the customer, thereby limiting user acceptance of the secure payment technology to those customers willing to install that software. Customers are generally reluctant to install such specialized software in the absence of a general acceptance of merchant software and payment gateway software that incorporate the corresponding secure payment technology with which to interact. Similarly, merchants and payment gateways are reluctant to implement a secure payment technology in the absence of an installed customer base that is available to use that secure payment technology. This presents a “chicken-and-the-egg” problem in that no particular component of a secure payment technology is likely to achieve general acceptance until the other component also achieve general acceptance.

[0004] Another such attempt to provide such a secure transmission channel is a general-purpose secure communication protocol such as Netscape, Inc.’s Secure Sockets Layer (hereinafter “SSL”), as described in Freier, Karlton & Kocher (hereinafter “Freier”), The SSL Protocol Version 3.0, March 1996, and hereby incorporated by reference. SSL provides a means for secure transmission between two computers. SSL has the advantage that it does not require special-purpose software to be installed on the customer’s computer because it is already incorporated into widely available software that many people utilize as their standard Internet access medium, and does not require that the customer interact with any third-party certification authority. Instead, the support for SSL may be incorporated into software already in use by the customer, e.g., the Netscape Navigator World Wide Web browsing tool. However, although a computer on an SSL connection may initiate a second SSL connection to another computer, a drawback to the SSL approach is that SSL connection supports only a two-computer connection. Therefore, SSL does not provide a mechanism for transmitting encoded information to a merchant for retransmission to a payment gateway such that a subset of the information is readable to the payment gateway but not to the merchant. Although SSL allows for robustly secure two-party data transmission, it does not meet the ultimate need of the electronic commerce market for robustly secure three-party data transmission. Other examples of general-purpose secure communication protocols include Private Communications Technology (“PCT”) from Microsoft, Inc., Secure Hyper-Text Transport Protocol (“SHTTP”) from Theresa Systems, Shen, Kerberos, Pto- turis, Pretty Good Privacy (“PGP”) and IPv6 which meets the IPSEC criteria. One of ordinary skill in the art will readily comprehend that any of the general-purpose secure communication protocols can be substituted for the SSL transmission protocol without undue experimentation.

OBJECTS OF THE INVENTION

[0005] It is desirable to provide a hybrid approach that encourages the deployment of a three-party secure channel such as SET by payment gateways in the absence of customer acceptance, thereby providing customers with an incentive to install SET-compliant software on their computer systems. It is further desirable to provide a means by which a merchant may communicate with a customer using a readily deployed secure channel such as SSL, or another general-purpose secure communication protocol, and communicate with a payment gateway using a modified SET-like protocol that is not dependent upon customer certification.

SUMMARY OF THE INVENTION

[0006] According to a broad aspect of a preferred embodiment of the invention, secure transmission of data is provided between a plurality of computer systems over a public
communication system, such as the Internet. Secure transmission of data is provided from a customer computer system to a merchant computer system, and for the further secure transmission of data from the merchant computer system to a payment gateway computer system. The payment gateway system evaluates the information and returns authorization or denial of credit via a secure transmission to the merchant which is communicated to the customer by the merchant.

DESCRIPTION OF THE DRAWINGS

[0007] The foregoing and other objects, aspects and advantages are better understood from the following detailed description of a preferred embodiment of the invention with reference to the drawings, in which:

[0008] FIG. 1A is a block diagram of a representative hardware environment in accordance with a preferred embodiment;

[0009] FIG. 1B depicts an overview in accordance with a preferred embodiment;

[0010] FIG. 2 depicts a more detailed view of a customer computer system in communication with the merchant system under the Secure Sockets Layer protocol in accordance with a preferred embodiment;

[0011] FIG. 3 depicts an overview of the method of securely supplying payment information to a payment gateway in order to obtain payment authorization in accordance with a preferred embodiment;

[0012] FIG. 4 depicts the detailed steps of generating and transmitting a payment authorization request in accordance with a preferred embodiment;

[0013] FIGS. 5A through 5F depict views of the payment authorization request and its component parts in accordance with a preferred embodiment;

[0014] FIGS. 6A and 6B depict the detailed steps of processing a payment authorization request and generating and transmitting a payment authorization request response in accordance with a preferred embodiment;

[0015] FIGS. 7A through 7J depict views of the payment authorization response and its component parts in accordance with a preferred embodiment;

[0016] FIG. 8 depicts the detailed steps of processing a payment authorization response in accordance with a preferred embodiment;

[0017] FIG. 9 depicts an overview of the method of securely supplying payment capture information to a payment gateway in accordance with a preferred embodiment;

[0018] FIG. 10 depicts the detailed steps of generating and transmitting a payment capture request in accordance with a preferred embodiment;

[0019] FIGS. 11A through 11F depict views of the payment capture request and its component parts in accordance with a preferred embodiment;

[0020] FIGS. 12A and 12B depict the detailed steps of processing a payment capture request and generating and transmitting a payment capture request response in accordance with a preferred embodiment;

[0021] FIGS. 13A through 13F depict views of the payment capture response and its component parts in accordance with a preferred embodiment; and

[0022] FIG. 14 depicts the detailed steps of processing a payment capture response in accordance with a preferred embodiment.

DETAILED DESCRIPTION

[0023] A preferred embodiment of a system in accordance with the present invention is preferably practiced in the context of a personal computer such as the IBM PS/2, Apple Macintosh computer or UNIX based workstation. A representative hardware environment is depicted in FIG. 1A, which illustrates a typical hardware configuration of a workstation in accordance with a preferred embodiment having a central processing unit 10, such as a microprocessor, and a number of other units interconnected via a system bus 12. The workstation shown in FIG. 1A includes a Random Access Memory (RAM) 14, Read Only Memory (ROM) 16, an I/O adapter 18 for connecting peripheral devices such as disk storage units 20 to the bus 12, a user interface adapter 22 for connecting a keyboard 24, a mouse 26, a speaker 28, a microphone 32, and/or other user interface devices such as a touch screen (not shown) to the bus 12, communication adapter 34 for connecting the workstation to a communication network (e.g., a data processing network) and a display adapter 36 for connecting the bus 12 to a display device 38. The workstation typically has resident thereon an operating system such as the Microsoft Windows Operating System (OS), the IBM OS/2 operating system, the MAC OS, or UNIX operating system. Those skilled in the art will appreciate that the present invention may also be implemented on platforms and operating systems other than those mentioned.

[0024] A preferred embodiment is written using JAVA, C, and the C++ language and utilizes object oriented programming methodology. Object oriented programming (OOP) has become increasingly used to develop complex applications. As OOP moves toward the mainstream of software design and development, various software solutions will need to be adapted to make use of the benefits of OOP. A need exists for these principles of OOP to be applied to a messaging interface of an electronic messaging system such that a set of OOP classes and objects for the messaging interface can be provided.

[0025] OOP is a process of developing computer software using objects, including the steps of analyzing the problem, designing the system, and constructing the program. An object is a software package that contains both data and a collection of related structures and procedures. Since it contains both data and a collection of structures and procedures, it can be visualized as a self-sufficient component that does not require other additional structures, procedures or data to perform its specific task. OOP, therefore, views a computer program as a collection of largely autonomous components, called objects, each of which is responsible for a specific task. This concept of packaging data, structures, and procedures together in one component or module is called encapsulation.

[0026] In general, OOP components are reusable software modules which present an interface that conforms to an object model and which are accessed at run-time through a
component integration architecture. A component integration architecture is a set of architecture mechanisms which allow software modules in different process spaces to utilize each others capabilities or functions. This is generally done by assuming a common component object model on which to build the architecture.

[0027] It is worthwhile to differentiate between an object and a class of objects at this point. An object is a single instance of the class of objects, which is often just called a class. A class of objects can be viewed as a blueprint, from which many objects can be formed.

[0028] OOP allows the programmer to create an object that is a part of another object. For example, the object representing a piston engine is said to have a composition-relationship with the object representing a piston. In reality, a piston engine comprises a piston, valves and many other components; the fact that a piston is an element of a piston engine can be logically and semantically represented in OOP by two objects.

[0029] OOP also allows creation of an object that "depends from" another object. If there are two objects, one representing a piston engine and the other representing a piston engine wherein the piston is made of ceramic, then the relationship between the two objects is not that of composition. A ceramic piston engine does not make up a piston engine. Rather it is merely one kind of piston engine that has one more limitation than the piston engine; its piston is made of ceramic. In this case, the object representing the ceramic piston engine is called a derived object, and it inherits all of the aspects of the object representing the piston engine and adds further limitation or detail to it. The object representing the ceramic piston engine "depends from" the object representing the piston engine. The relationship between these objects is called inheritance.

[0030] When the object or class representing the ceramic piston engine inherits all of the aspects of the objects representing the piston engine, it inherits the thermal characteristics of a standard piston defined in the piston engine class. However, the ceramic piston engine object overrides these ceramic specific thermal characteristics, which are typically different from those associated with a metal piston. It skips over the original and uses new functions related to ceramic pistons. Different kinds of piston engines will have different characteristics, but may have the same underlying functions associated with it (e.g., how many pistons in the engine, ignition sequences, lubrication, etc.). To access each of these functions in any piston engine object, a programmer would call the same functions with the same names, but each type of piston engine may have different/overriding implementations of functions behind the same name. This ability to hide different implementations of a function behind the same name is called polymorphism and it greatly simplifies communication among objects.

[0031] With the concepts of composition-relationship, encapsulation, inheritance and polymorphism, an object can represent just about anything in the real world. In fact, our logical perception of the reality is the only limit on determining the kinds of things that can become objects in object-oriented software. Some typical categories are as follows:

[0032] Objects can represent physical objects, such as automobiles in a traffic-flow simulation, electrical components in a circuit-design program, countries in an economics model, or aircraft in an air-traffic-control system.

[0033] Objects can represent elements of the computer-user environment such as windows, menus or graphics objects.

[0034] An object can represent an inventory, such as a personnel file or a table of the latitudes and longitudes of cities.

[0035] An object can represent user-defined data types such as time, angles, and complex numbers, or points on the plane.

[0036] With this enormous capability of an object to represent just about any logically separable matters, OOP allows the software developer to design and implement a computer program that is a model of some aspects of reality, whether that reality is a physical entity, a process, a system, or a composition of matter. Since the object can represent anything, the software developer can create an object which can be used as a component in a larger software project in the future.

[0037] If 90% of a new OOP software program consists of proven, existing components made from preexisting reusable objects, then only the remaining 10% of the new software project has to be written and tested from scratch. Since 90% already came from an inventory of extensively tested reusable objects, the potential domain from which an error could originate is 10% of the program. As a result, OOP enables software developers to build objects out of other, previously built, objects.

[0038] This process closely resembles complex machinery being built out of assemblies and sub-assemblies. OOP technology, therefore, makes software engineering more like hardware engineering in that software is built from existing components, which are available to the developer as objects. All this adds up to an improved quality of the software as well as an increased speed of its development.

[0039] Programming languages are beginning to fully support the OOP principles, such as encapsulation, inheritance, polymorphism, and composition-relationship. With the advent of the C++ language, many commercial software developers have embraced OOP. C++ is an OOP language that offers a fast, machine-executable code. Furthermore, C++ is suitable for both commercial-application and systems-programming projects. For now, C++ appears to be the most popular choice among many OOP programmers, but there is a host of other OOP languages, such as Smalltalk, common lisp object system (CLOS), and Eiffel. Additionally, OOP capabilities are being added to more traditional popular computer programming languages such as Pascal.

[0040] The benefits of object classes can be summarized, as follows:

[0041] Objects and their corresponding classes break down complex programming problems into many smaller, simpler problems.

[0042] Encapsulation enforces data abstraction through the organization of data into small, independent objects that can communicate with each other. Encapsulation protects the data in an object from
accidental damage, but allows other objects to interact with that data by calling the object’s member functions and structures.

[0043] Subclassing and inheritance make it possible to extend and modify objects through deriving new kinds of objects from the standard classes available in the system. Thus, new capabilities are created without having to start from scratch.

[0044] Polymorphism and multiple inheritance make it possible for different programmers to mix and match characteristics of many different classes and create specialized objects that can still work with related objects in predictable ways.

[0045] Class hierarchies and containment hierarchies provide a flexible mechanism for modeling real-world objects and the relationships among them.

[0046] Libraries of reusable classes are useful in many situations, but they also have some limitations. For example:

[0047] Complexity. In a complex system, the class hierarchies for related classes can become extremely confusing, with many dozens or even hundreds of classes.

[0048] Flow of control. A program written with the aid of class libraries is still responsible for the flow of control (i.e., it must control the interactions among all the objects created from a particular library). The programmer has to decide which functions to call at what times for which kinds of objects.

[0049] Duplication of effort. Although class libraries allow programmers to use and reuse many small pieces of code, each programmer puts those pieces together in a different way. Two different programmers can use the same set of class libraries to write two programs that do exactly the same thing but whose internal structure (i.e., design) may be quite different, depending on hundreds of small decisions each programmer makes along the way. Inevitably, similar pieces of code end up doing similar things in slightly different ways and do not work as well together as they should.

[0050] Class libraries are very flexible. As programs grow more complex, more programmers are forced to reinvent basic solutions to basic problems over and over again. A relatively new extension of the class library concept is to have a framework of class libraries. This framework is more complex and consists of significant collections of collaborating classes that capture both the small scale patterns and major mechanisms that implement the common requirements and design in a specific application domain. They were first developed to free application programmers from the chores involved in displaying menus, windows, dialog boxes, and other standard user interface elements for personal computers.

[0051] Frameworks also represent a change in the way programmers think about the interaction between the code they write and code written by others. In the early days of procedural programming, the programmer called libraries provided by the operating system to perform certain tasks, but basically the program executed down the page from start to finish, and the programmer was solely responsible for the flow of control. This was appropriate for printing out paychecks, calculating a mathematical table, or solving other problems with a program that executed in just one way.

[0052] The development of graphical user interfaces began to turn this procedural programming arrangement inside out. These interfaces allow the user, rather than program logic, to drive the program and decide when certain actions should be performed. Today, most personal computer software accomplishes this by means of an event loop which monitors the mouse, keyboard, and other sources of external events and calls the appropriate parts of the programmer’s code according to actions that the user performs. The programmer no longer determines the order in which events occur. Instead, a program is divided into separate pieces that are called at unpredictable times and in an unpredictable order. By relinquishing control in this way to users, the developer creates a program that is much easier to use. Nevertheless, individual pieces of the program written by the developer still call libraries provided by the operating system to accomplish certain tasks, and the programmer must still determine the flow of control within each piece after it’s called by the event loop. Application code still “sits on top of” the system.

[0053] Even event loop programs require programmers to write a lot of code that should not need to be written separately for every application. The concept of an application framework carries the event loop concept further. Instead of dealing with all the nuts and bolts of constructing basic menus, windows, and dialog boxes and then making these things all work together, programmers using application frameworks start with working application code and basic user interface elements in place. Subsequently, they build from there by replacing some of the generic capabilities of the framework with the specific capabilities of the intended application.

[0054] Application frameworks reduce the total amount of code that a programmer has to write from scratch. However, because the framework is really a generic application that displays windows, supports copy and paste, and so on, the programmer can also relinquish control to a greater degree than event loop programs permit. The framework code takes care of almost all event handling and flow of control, and the programmer’s code is called only when the framework needs it (e.g., to create or manipulate a proprietary data structure).

[0055] A programmer writing a framework program not only relinquishes control to the user (as is also true for event loop programs), but also relinquishes the detailed flow of control within the program to the framework. This approach allows the creation of more complex systems that work together in interesting ways, as opposed to isolated programs, having custom code, being created over and over again for similar problems.

[0056] Thus, as is explained above, a framework basically is a collection of cooperating classes that make up a reusable design solution for a given problem domain. It typically includes objects that provide default behavior (e.g., for menus and windows), and programmers use it by inheriting some of that default behavior and overriding other behavior so that the framework calls application code at the appropriate times.
There are three main differences between frameworks and class libraries:

Behavior versus protocol. Class libraries are essentially collections of behaviors that you can call when you want those individual behaviors in your program. A framework, on the other hand, provides not only behavior but also the protocol or set of rules that govern the ways in which behaviors can be combined, including rules for what a programmer is supposed to provide versus what the framework provides.

Call versus override. With a class library, the code the programmer instantiates objects and calls their member functions. It’s possible to instantiate and call objects in the same way with a framework (i.e., to treat the framework as a class library), but to take full advantage of a framework’s reusable design, a programmer typically writes code that overrides and is called by the framework. The framework manages the flow of control among its objects. Writing a program involves dividing responsibilities among the various pieces of software that are called by the framework rather than specifying how the different pieces should work together.

Implementation versus design. With class libraries, programmers reuse only implementations, whereas with frameworks, they reuse design. A framework embodies the way a family of related programs or pieces of software work. It represents a generic design solution that can be adapted to a variety of specific problems in a given domain. For example, a single framework can embody the way a user interface works, even though two different user interfaces created with the same framework might solve quite different interface problems.

Thus, through the development of frameworks for solutions to various problems and programming tasks, significant reductions in the design and development effort for software can be achieved. A preferred embodiment of the invention utilizes HyperText Markup Language (HTML) to implement documents on the Internet together with a general-purpose secure communication protocol for a transport medium between the client and the merchant. HTML is a simple data format used to create hypertext documents that are portable from one platform to another. HTML documents are SGML documents with generic semantics that are appropriate for representing information from a wide range of domains. HTML has been in use by the World-Wide Web global information initiative since 1990. HTML is an application of ISO Standard 8879:1986 Information Processing Text and Office Systems, Standard Generalized Markup Language (SGML).

To date, Web development tools have been limited in their ability to create dynamic Web applications which span from client to server and interoperate with existing computing resources. Until recently, HTML has been the dominant technology used in development of Web-based solutions. However, HTML has proven to be inadequate in the following areas:

- Poor performance;
- Restricted user interface capabilities;
- Can only produce static Web pages;
- Lack of interoperability with existing applications and data; and
- Inability to scale.

Sun Microsystems’s Java language solves many of the client-side problems by:

- Improving performance on the client side;
- Enabling the creation of dynamic, real-time Web applications; and
- Providing the ability to create a wide variety of user interface components.

With Java, developers can create robust User Interface (UI) components. Custom “widgets” (e.g., real-time stock tickers, animated icons, etc.) can be created, and client-side performance is improved. Unlike HTML, Java supports the notion of client-side validation, offloading appropriate processing onto the client for improved performance. Dynamic, real-time Web pages can be created. Using the above-mentioned custom UI components, dynamic Web pages can also be created. Sun’s Java language has emerged as an industry-recognized language for “programming the Internet.” Sun defines Java as: “a simple, object-oriented, distributed, interpreted, robust, secure, architecture-neutral, portable, high-performance, multithreaded, dynamic, buzzword-compliant, general-purpose programming language. Java supports programming for the Internet in the form of platform-independent Java applets.” Java applets are small, specialized applications that comply with Sun’s Java Application Programming Interface (API) allowing developers to add “interactive content” to Web documents (e.g., simple animations, page adornments, basic games, etc.). Applets execute within a Java-compatible browser (e.g., Netscape Navigator) by copying code from the server to the client. From a language standpoint, Java’s core feature set is based on C++. Sun’s Java literature states that Java is basically “C++ with extensions from Objective C for more dynamic method resolution”.

Another technology that provides similar function to JAVA is provided by Microsoft and ActiveX Technologies, to give developers and Web designers wherewithal to build dynamic content for the Internet and personal computers. ActiveX includes tools for developing animation, 3-D virtual reality, video and other multimedia content. The tools use Internet standards, work on multiple platforms, and are being supported by over 100 companies. The group’s building blocks are called ActiveX Controls, small, fast components that enable developers to embed parts of software in hypertext markup language (HTML) pages. ActiveX Controls work with a variety of programming languages including Microsoft Visual C++, Borland Delphi, Microsoft Visual Basic programming system and, in the future, Microsoft’s development tool for Java, code named “Jakarta.” ActiveX Technologies also includes ActiveX Server Framework, allowing developers to create server applications. One of ordinary skill in the art will readily recognize that ActiveX could be substituted for JAVA without undue experimentation to practice the invention.
chant session 150 operates under a general-purpose secure communication protocol such as the SSL protocol. Merchant computer system 130 is additionally in communication with payment gateway computer system 140. A payment gateway is a system that provides electronic commerce services in support of a bank or other financial institution, and that interfaces to the financial institution to support the authorization and capture of transactions. The customer-institution session 170 operates under a variant of a secure payment technology such as the SET protocol, as described herein, referred to as Merchant-Originated Secure Electronic Transactions ("MOSET"), as is more fully described herein.

[0075] Customer-to-Merchant Communication

[0076] FIG. 2 depicts a more detailed view of customer computer system 120 in communication with merchant system 130 using customer-merchant session 150 operating under the SSL protocol as documented in Freier and incorporated by reference.

[0077] Customer computer system 120 initiates communication with merchant computer system 130 using any well-known access protocol, e.g., Transmission Control Protocol/Internet Protocol ("TCP/IP"). In this implementation, customer computer system 120 acts as a client and merchant computer system 130 acts as a server. Customer computer system 120 initiates communication by sending "client hello" message 210 to the merchant computer system 130. When a client first connects to a server it is required to send the client hello message 210 as its first message. The client can also send a client hello message 210 in response to a hello request on its own initiative in order to renegotiate the security parameters in an existing connection. The client hello message includes a random structure, which is used later in the protocol. Specifically, the random structure includes the current time and date in standard UNIX 32-bit format according to the sender’s internal clock and twenty-eight bytes of data generated by a secure random number generator. The client hello message 210 further includes a variable length session identifier. If not empty, the session identifier value identifies a session between the same client and server whose security parameters the client wishes to reuse. The session identifier may be from an earlier connection, the current connection, or another currently active connection. It is useful to specify the current connection if the client only wishes to update the random structures and derived values of a connection. It is useful to specify another currently active connection if the client wishes to establish several simultaneous independent secure connections to the same server without repeating the full handshake protocol. Client hello message 210 further includes an indicator of the cryptographic algorithms supported by the client in order of the client’s preference, ordered according to client preference.

[0078] In response to client hello message 210, if merchant computer system 130 wishes to correspond with customer computer system 120, it responds with server hello message 215. If merchant computer system 130 does not wish to communicate with customer computer system 120, it responds with a message, not shown, indicating refusal to communicate.

[0079] Server hello message 215 includes a random structure, which is used later in the protocol. The random structure in server hello message 215 is in the same format as, but has contents independent of, the random structure in client hello message 210. Specifically, the random structure includes the current time and date in standard UNIX 32-bit format according to the sender’s internal clock and twenty-eight bytes of data generated by a secure random number generator. Server hello message 215 further includes a variable length session identifier. The session identifier value identifies a new or existing session between the same client and server. Server hello message 215 further includes an indicator of the cryptographic algorithms selected from among the algorithms specified by client hello message 210, which will be used in further encrypted communications.

[0080] Optionally, Merchant computer system 130 transmits a server certificate 220. If transmitted, server certificate 130 enables customer computer system 120 to authenticate the identity of merchant computer system 130.

[0081] If merchant computer system 130 does not transmit a server certificate 220, or if server certificate 220 is suitable only for authentication, it may optionally transmit a server key exchange message 225. Server key exchange message 225 identifies a key that may be used by customer computer system 120 to decrypt further messages sent by merchant computer system 130.

[0082] After transmitting server hello message 215, and optionally transmitting server certificate 220 or server key exchange message 225, merchant computer system 130 transmits a server hello done message 230 and waits for a further response from customer computer system 120.

[0083] Customer computer system 120 optionally transmits client certificate 240 to merchant computer system 130. If transmitted, client certificate 240 enables merchant computer system 130 to authenticate the identity of customer computer system 120. Alternatively, customer computer system 120 may transmit a no-client-certificate alert 245, to indicate that the customer has not registered with any certification authority.

[0084] If customer computer system 130 does not transmit a client certificate 240, or if client certificate 240 is suitable only for authentication, customer computer system 130 may optionally transmit a server key exchange message 250. Client key exchange message 250 identifies a key that may be used by merchant computer system 130 to decrypt further messages sent by customer computer system 120.

[0085] After optionally transmitting client certificate 240, no-client-certificate alert 245, and/or client key exchange message 250, customer computer system 120 transmits a finished message 260.

[0086] At this point, customer computer system 120 and merchant computer system 130 have:

[0087] 1) negotiated an encryption scheme that may be commonly employed in further communications, and

[0088] 2) have communicated to each other a set of encryption keys that may be used to decrypt further communications between the two computer systems.

[0089] Customer computer system 120 and merchant computer system 130 may thereafter engage in secure communications 270 with less risk of interception by third parties.
Among the messages communicated by customer computer system 120 to merchant computer system 130 may be messages that specify goods or services to be ordered and payment information, such as a credit card number and related information, collectively referred to as “payment information,” that may be used to pay for the goods and/or services ordered. In order to obtain payment, the merchant must supply this information to the bank or other payment gateway responsible for the proffered payment method. This enables the merchant to perform payment authorization and payment capture. Payment authorization is the process by which permission is granted by a payment gateway operating on behalf of a financial institution to authorize payment on behalf of the financial institution. This is a process that assesses transaction risk, confirms that a given transaction does not raise the account holder’s debt above the account’s credit limit, and reserves the specified amount of credit. Payment capture is the process that triggers the movement of funds from the financial institution to the merchant’s account.

Payment Authorization

FIG. 3 depicts an overview of the method of securely supplying payment information to a payment gateway in order to obtain payment authorization. In function block 310, merchant computer system 130 generates a payment authorization request 315 and transmits it to payment gateway computer system 140. In function block 330, payment gateway system 140 processes the payment authorization request, generates a payment authorization response 325 and transmits it to merchant computer system 130. In function block 320, merchant computer system 130 processes payment authorization response 325 and determines whether payment for the goods or services sought to be obtained by the customer has been authorized.

Payment Authorization Request Generation

FIG. 4 depicts the detailed steps of generating and transmitting a payment authorization request. FIGS. 5A through 5F depict views of the payment authorization request and its component parts. In function block 410, merchant computer system 130 creates a basic authorization request 510. The basic authorization request is a data area that includes all the information for determining whether a request should be granted or denied. Specifically, it includes such information as the party who is being charged, the amount to be charged, the account number of the account to be charged, and any additional data, such as passwords, needed to validate the charge.

This information is either calculated based upon prior customer merchandise selection, or provided by the customer over the secure link 270 established in the customer-merchant general-purpose secure communication protocol session. FIG. 5A depicts a basic authorization request 510.

In function block 420, merchant computer system 130 combines basic authorization request 510, a copy of its encryption public key certificate 515 and a copy of its signature public key certificate 520. Merchant computer system 130 calculates a digital signature 525 for the combined contents of the combined block 530 comprising basic authorization request 510, the encryption public key certificate 515 and the signature public key certificate 520, and appends it to the combination of the combined basic authorization request 510, the encryption public key certificate 515 and the signature public key certificate 520. The merchant computer system calculates digital signature 525 by first calculating a “message digest” based upon the contents of the combined basic authorization request 510, the encryption public key certificate 515 and the signature public key certificate 520. A message digest is the fixed-length result that is generated when a variable length message is fed into a one-way hashing function. Message digests help verify that a message has not been altered because altering the message would change the digest. The message digest is then encrypted using the merchant computer system’s 130 digital signature private key, thus forming a digital signature.

FIG. 5B depicts the combined block 530 formed by function block 420 and containing basic authorization request 510, the encryption public key certificate 515, the signature public key certificate 520, and digital signature 525.

In function block 430, merchant computer system 130 generates a random encryption key RK-0 540, denoted as RK-0. Random encryption key RK-0 540 is a symmetric encryption key. A symmetric encryption key is a key characterized by the property that a message encrypted with a symmetric key can be decrypted with that same key. This is contrasted with an asymmetric key pair, such as a public-key/private-key key pair, where a message encrypted with one key of the key pair may only be decrypted with the other key of the same key pair. FIG. 5C depicts random encryption key RK-0 540.

In function block 440, merchant computer system 130 encrypts combined block 530 using random encryption key RK-0 540 to form encrypted combined block 550. FIG. 5D depicts encrypted combined block 550. The encryption state of encrypted combined block 550 is graphically shown by random key lock 555, which indicates that encrypted combined block 550 is encrypted using random key RK-0 540.

In function block 450, merchant computer system 130 encrypts random encryption key RK-0 540 using the public key of payment gateway system 140 to form encrypted random key 560. FIG. 5E depicts encrypted random key 560. The encryption state of encrypted random key 560 is graphically shown by payment gateway public key lock 565, which indicates that encrypted random key 560 is encrypted using the payment gateway public key.

In function block 460, merchant computer system 130 concatenates encrypted combined block 550 and encrypted random key 560 to form merchant authorization request 315. FIG. 5F depicts merchant authorization request 315 comprising encrypted combined block 550 and encrypted random key 560. In function block 470, merchant computer system 130 transmits merchant authorization request 315 to payment gateway system 140.

Payment Authorization Request Processing

FIG. 6 depicts the detailed steps of processing a payment authorization request and generating and transmitting a payment authorization request response. Function blocks 610 through 630 depict the steps of processing a payment authorization request, while function blocks 635
through 685 depict the steps of generating and transmitting a payment authorization request response.

[0104] In function block 610, payment gateway computer system 140 applies its private key to encrypted random key 560 contained within received merchant authorization request 315, thereby decrypting it and obtaining a cleartext version of random key RK-0 540. In function block 615, payment gateway computer system 140 applies random key RK-0 540 to encrypted combined block 550, thereby decrypting it and obtaining a cleartext version of combined block 530. It will be recalled that combined block 530 comprises basic authorization request 510, a copy of merchant computer system’s 130 encryption public key certificate 515 and a copy of merchant computer system’s 130 signature public key certificate 520, as well as merchant digital signature 525.

[0105] In function block 620, payment gateway computer system 140 verifies merchant computer system’s 130 encryption public key certificate 515 and merchant computer system’s 130 signature public key certificate 520. Payment gateway computer system 140 performs this verification by making a call to the certification authorities associated with each certificate. If verification of either certificate fails, payment gateway computer system 140 rejects the authorization request.

[0106] In function block 625, payment gateway computer system 140 validates merchant digital signature 525. Payment gateway computer system 140 performs this validation by calculating a message digest over the contents of the combined basic authorization request 510, the encryption public key certificate 515 and the signature public key certificate 520. Payment gateway computer system 140 then decrypts digital signature 525 to obtain a copy of the equivalent message digest calculated by merchant computer system 130 in function block 420. If the two message digests are equal, the digital signature 525 is validated. If validation fails, payment gateway computer system 140 rejects the authorization request.

[0107] In function block 630, payment gateway computer system 140 determines the financial institution for which authorization is required by inspection of basic authorization request 510. Payment gateway computer system 140 contacts the appropriate financial institution using a secure means, e.g., a direct-dial modem-to-modem connection, or a proprietary internal network that is not accessible to third parties, and using prior art means, obtains a response indicating whether the requested payment is authorized.

[0108] Payment Authorization Response Generation

[0109] Function blocks 635 through 685 depict the steps of generating and transmitting a payment authorization request response. FIGS. 7A through 7H depict views of the payment authorization response and its component parts.

[0110] In function block 635, payment gateway computer system 140 creates a basic authorization response 710. The basic authorization request is a data area that includes all the information to determine whether a request was granted or denied. FIG. 7A depicts basic authorization response 710.

[0111] In function block 640, payment gateway computer system 140 combines basic authorization response 710, and a copy of its signature public key certificate 720. Payment computer system 140 calculates a digital signature 725 for the combined contents of the combined block 730 comprising basic authorization response 710 and the signature public key certificate 720, and appends the signature to the combination of the basic authorization response 710 and the signature public key certificate 720. The payment gateway computer system calculates digital signature 725 by first calculating a message digest based on the contents of the combined basic authorization response 710 and signature public key certificate 720. The message digest is then encrypted using the merchant computer system’s 140 digital signature private key, thus forming a digital signature.

[0112] FIG. 7B depicts the combined block 730 formed in function block 640 and containing basic authorization response 710, the signature public key certificate 720, and digital signature 725.

[0113] In function block 645, payment gateway computer system 150 generates a first symmetric random encryption key 740, denoted as RK-1. FIG. 7C depicts first random encryption key RK-1 740.

[0114] In function block 650, payment gateway computer system 140 encrypts combined block 730 using random encryption key RK-1 740 to form encrypted combined block 750. FIG. 7D depicts encrypted combined block 750. The encrypted state of encrypted combined block 750 is graphically shown by random key lock 755, which indicates that encrypted combined block 750 is encrypted using random key RK-1 740.

[0115] In function block 655, payment gateway computer system 140 encrypts random encryption key RK-1 740 using the public key of merchant computer system 130 to form encrypted random key RK 760. FIG. 7E depicts encrypted random key RK-1 760. The encrypted state of encrypted random key 760 is graphically shown by merchant public key lock 765, which indicates that encrypted random key 760 is encrypted using the merchant public key.

[0116] In function block 660, payment gateway computer system 140 generates a random capture token 770. Random capture token 770 will be used in subsequent payment capture processing to associate the payment capture request with the payment authorization request being processed. FIG. 7F depicts capture token 775.

[0117] In function block 665, payment gateway computer system 140 generates a second symmetric random encryption key 775, denoted as RK-2. FIG. 7G depicts second random encryption key RK-2 775.

[0118] In function block 670, payment gateway computer system 140 encrypts capture token 770 using random encryption key RK-2 775 to form encrypted capture token 780. FIG. 7H depicts encrypted capture token 780. The encrypted state of encrypted capture token 780 is graphically shown by random key lock 785, which indicates that encrypted capture token 780 is encrypted using random key RK-2 770.

[0119] In function block 675, payment gateway computer system 140 encrypts second random encryption key RK-2 775 using its own public key to form encrypted random key RK-2 790. FIG. 7I depicts encrypted random key RK-2 790. The encrypted state of encrypted random key 790 is
graphically shown by payment gateway public key lock 795, which indicates that encrypted random key 790 is encrypted using the payment gateway public key.

[0120] In function block 680, payment gateway computer system 140 concatenates encrypted combined block 750, encrypted random key RK-1 760, encrypted capture token 780 and encrypted random key RK-2 790 to form merchant authorization response 325. FIG. 7J depicts merchant authorization response 325 comprising encrypted combined block 750, encrypted random key RK-1 760, encrypted capture token 780 and encrypted random key RK-2 790. In function block 685, payment gateway computer system 140 transmits merchant authorization response 325 to merchant system 130.

[0121] Payment Authorization Response Processing

[0122] FIG. 8 depicts the detailed steps of processing a payment authorization response. In function block 810, merchant computer system 130 applies its private key to encrypted random key RK-1 760 contained within received merchant authorization response 325, thereby decrypting it and obtaining a cleartext version of random key RK-1 740.

[0123] In function block 820, merchant computer system 130 applies random key RK-1 740 to encrypted combined block 750, thereby decrypting it and obtaining a cleartext version of combined block 730. It will be recalled that combined block 730 comprises basic authorization response 710, a copy of payment gateway computer system’s 140 signature public key certificate 720, as well as payment gateway digital signature 725.

[0124] In function block 830, merchant computer system 130 verifies payment gateway computer system’s 140 signature public key certificate 720. Merchant computer system 130 performs this verification by making a call to the certification authority associated with the certificate. If verification of the certificate fails, merchant computer system 130 concludes that the authorization response is counterfeit and treats it though the authorization request had been rejected.

[0125] In function block 840, merchant computer system 130 validates payment gateway digital signature 725. Merchant computer system 130 performs this validation by calculating a message digest over the contents of the combined basic authorization request 710 and the signature public key certificate 720. Merchant computer system 130 then decrypts digital signature 725 to obtain a copy of the equivalent message digest calculated by payment gateway computer system 140 in function block 640. If the two message digests are equal, the digital signature 725 is validated. If validation fails, concludes that the authorization response is counterfeit and treats it though the authorization request had been rejected.

[0126] In function block 850, merchant computer system 130 stores encrypted capture token 780 and encrypted random key RK-2 790 for later use in payment capture. In function block 860, merchant computer system 130 processes the customer purchase request in accordance with the authorization response 710. If the authorization response indicates that payment in authorized, merchant computer system 130 fills the requested order. If the authorization response indicates that payment is not authorized, or if merchant computer system 130 determined in function block 830 or 840 that the authorization response is counterfeit, merchant computer system 130 indicates to the customer that the order cannot be filled.

[0127] Payment Capture

[0128] FIG. 9 depicts an overview of the method of securely supplying payment capture information to payment gateway 140 in order to obtain payment capture. In function block 910, merchant computer system 130 generates a merchant payment capture request 915 and transmits it to payment gateway computer system 140. In function block 930, payment gateway system 140 processes the payment capture request 915, generates a payment capture response 925 and transmits it to merchant computer system 130. In function block 920, merchant computer system 130 processes payment capture response 925 and verifies that payment for the goods or services sought to be obtained by the customer have been captured.

[0129] Payment Capture Request Generation

[0130] FIG. 10 depicts the detailed steps of generating and transmitting a payment capture request. FIGS. 11A through 11F depict views of the payment capture request and its components part. In function block 1010, merchant computer system 130 creates a basic capture request 510. The basic capture request is a data area that includes all the information needed by payment gateway computer system 140 to trigger a transfer of funds to the merchant operating merchant computer system 130.

[0131] Specifically, a capture request includes a capture request amount, a capture token, a date, summary information of the purchased items and a Merchant ID (MID) for the particular merchant. FIG. 11A depicts basic authorization request 1110.

[0132] In function block 1020, merchant computer system 130 combines basic capture request 1110, a copy of its encryption public key certificate 1115 and a copy of its signature public key certificate 1120. Merchant computer system 130 calculates a digital signature 1125 for the combined contents of the combined block 1130 comprising basic capture request 1110, the encryption public key certificate 1115 and the signature public key certificate 1120, and appends it to the combination of the basic capture request 1110, the encryption public key certificate 1115 and the signature public key certificate 1120. The merchant computer system calculates digital signature 1125 by first calculating a message digest over the contents of the combined basic capture request 1110, the encryption public key certificate 1115 and the signature public key certificate 1120. The message digest is then encrypted using the merchant computer system’s 130 digital signature private key, thus forming a digital signature.

[0133] FIG. 11B depicts the combined block 1130 formed by function block 1020 and containing basic capture request 1110, the encryption public key certificate 1115, the signature public key certificate 1120, and digital signature 1125.

[0134] In function block 1030, merchant computer system 130 generates a random encryption key 1140, denoted as RK-3. Random encryption key RK-3 1140 is a symmetric encryption key. FIG. 11C depicts random encryption key RK-3 1140.
In function block 1040, merchant computer system 130 encrypts combined block 1130 using random encryption key RK-3 1140 to form encrypted combined block 1150. FIG. 11D depicts encrypted combined block 1150. The encryption state of encrypted combined block 1150 is graphically shown by random key lock 1155, which indicates that encrypted combined block 1150 is encrypted using random key RK-3 1140.

In function block 1050, merchant computer system 130 encrypts random encryption key RK-3 1140 using the public key of payment gateway system 140 to form encrypted random key 1160. FIG. 11E depicts encrypted random key 1160. The encryption state of encrypted random key 1160 is graphically shown by payment gateway public key lock 1165, which indicates that encrypted random key RK-3 1160 is encrypted using the payment gateway public key.

In function block 1060, merchant computer system 130 concatenates encrypted combined block 1150, encrypted random key 1160, and the encrypted capture token 780 and encrypted random key RK-2 790 that were stored in function block 850 to form merchant capture request 915. FIG. 11F depicts merchant capture request 915, comprising encrypted combined block 1150, encrypted random key 1160, encrypted capture token 780 and encrypted random key RK-2 790. In function block 1070, merchant computer system 130 transmits merchant capture request 915 to payment gateway system 140.

Payment Capture Request Processing

FIG. 12 depicts the detailed steps of processing a payment capture request and generating and transmitting a payment capture request response. Function blocks 1210 through 1245 depict the steps of processing a payment capture request, while function blocks 1250 through 1285 depict the steps of generating and transmitting a payment capture request response.

In function block 1210, payment gateway computer system 140 applies its private key to encrypted random key 1160 contained within received merchant capture request 915, thereby decrypting it and obtaining a cleartext version of random key RK-3 1140. In function block 1212, payment gateway computer system 140 applies random key RK-3 1140 to encrypted combined block 1150, thereby decrypting it and obtaining a cleartext version of combined block 1130. It will be recalled that combined block 1130 comprises basic capture request 1110, a copy of merchant computer system’s 130 encryption public key certificate 1115 and a copy of merchant computer system’s 130 signature public key certificate 1120, as well as merchant digital signature 1125.

In function block 1220, payment gateway computer system 140 verifies merchant computer system’s 130 encryption public key certificate 1115 and merchant computer system’s 130 signature public key certificate 1120.

Payment gateway computer system 140 performs this verification by making a call to the certification authorities associated with each certificate. If verification of either certificate fails, payment gateway computer system 140 rejects the capture request.

In function block 1225, payment gateway computer system 140 validates merchant digital signature 1125. Payment gateway computer system 140 performs this validation by calculating a message digest over the contents of the combined basic capture request 1110, the encryption public key certificate 1115 and the signature public key certificate 1120. Payment gateway computer system 140 then decrypts digital signature 1125 to obtain a copy of the equivalent message digest calculated by merchant computer system 130 in function block 1020. If the two message digests are equal, the digital signature 1125 is validated. If validation fails, payment gateway computer system 140 rejects the capture request.

In function block 1230, payment gateway computer system 140 applies its private key to encrypted random key RK-2 790 contained within received merchant capture request 915, thereby decrypting it and obtaining a cleartext version of random key RK-2 775. In function block 1235, payment gateway computer system 140 applies random key RK-2 775 to encrypted capture token 780, thereby decrypting it and obtaining a cleartext version of capture token 770.

In function block 1240, payment gateway computer system 140 verifies that a proper transaction is being transmitted between capture token 780 and capture request 1110. A capture token contains data that the gateway generates at the time of authorization. When the authorization is approved, the encrypted capture token is given to the merchant for storage. At the time of capture, the merchant returns the capture token to the gateway along with other information required for capture. Upon receipt of the capture token, the gateway compares a message made of the capture request data and the capture token data and transmits this information over a traditional credit/debit network. If an improperly formatted transaction is detected, payment gateway computer system 140 rejects the capture request.

In function block 1245, payment gateway computer system 140 determines the financial institution for which capture is requested by inspection of basic capture request 1110. Payment gateway computer system 140 contacts the appropriate financial institution using a secure means, e.g., a direct-dial modem-to-modem connection, or a proprietary internal network that is not accessible to third parties, and using prior art means, instructs a computer at the financial institution to perform the requested funds transfer.

Payment Capture Response Generation

Function blocks 1250 through 1285 depict the steps of generating and transmitting a payment capture request response. FIGS. 13A through 13F depict views of the payment capture response and its component parts.

In function block 1250, payment gateway computer system 140 creates a basic capture response 710. The basic capture request is a data area that includes all the information to indicate whether a capture request was granted or denied. FIG. 13A depicts basic authorization request 1310.

In function block 1255, payment gateway computer system 140 combines basic capture response 1310, and a copy of its signature public key certificate 1320. Payment computer system 140 calculates a digital signature 1325 for the combined contents of the combined block 1330 comprising basic capture response 1310 and the signature public key certificate 1320, and appends the signature to the combination of the combined basic authorization request
1310 and the signature public key certificate 1320. The payment gateway computer system calculates digital signature 1325 by first calculating a message digest over the contents of the combined basic capture response 1310 and signature public key certificate 1320. The message digest is then encrypted using the merchant computer system’s 140 digital signature private key, thus forming a digital signature.

[0151] FIG. 13B depicts the combined block 1330 formed by function block 1255 and containing basic capture request 1310, the signature public key certificate 1320, and digital signature 1325.

[0152] In function block 1260, payment gateway computer system 140 generates a symmetric random encryption key 1340, denoted as RK-4. FIG. 13C depicts random encryption key RK-4 1340.

[0153] In function block 1275, payment gateway computer system 140 encrypts combined block 1330 using random encryption key RK-4 1340 to form encrypted combined block 1350. FIG. 13D depicts encrypted combined block 1350. The encryption state of encrypted combined block 1350 is graphically shown by random key lock 1355, which indicates that encrypted combined block 1350 is encrypted using random key RK-4 1340.

[0154] In function block 1275, payment gateway computer system 140 encrypts random encryption key RK-4 1340 using the public key of merchant computer system 130 to form encrypted random key RK-4 1360. FIG. 13E depicts encrypted random key RK-4 1360. The encryption state of encrypted random key 1360 is graphically shown by merchant public key lock 1365, which indicates that encrypted random key 1360 is encrypted using the merchant public key.

[0155] In function block 1280, payment gateway computer system 140 concatenates encrypted combined block 1350 and encrypted random key RK-4 1360 to form merchant capture response 925. FIG. 13F depicts merchant capture response 925 comprising encrypted combined block 1350 and encrypted random key RK-4 1360. In function block 1285, payment gateway computer system 140 transmits merchant capture response 925 to merchant system 130.

[0156] Payment Capture Response Processing

[0157] FIG. 14 depicts the detailed steps of processing a payment capture response. In function block 1410, merchant computer system 130 applies its private key to encrypted random key RK-4 1340 contained within received merchant capture response 925, thereby decrypting it and obtaining a cleartext version of random key RK-4 1340.

[0158] In function block 1420, merchant computer system 130 applies random key RK-4 1340 to encrypted combined block 1350, thereby decrypting it and obtaining a cleartext version of combined block 1330. It will be recalled that combined block 1330 comprises basic capture response 1310, a copy of payment gateway computer system’s 140 signature public key certificate 1320, as well as payment gateway digital signature 1325.

[0159] In function block 1430, merchant computer system 130 verifies payment gateway computer system’s 140 signature public key certificate 1320. Merchant computer system 130 performs this verification by making a call to the certification authority associated with the certificate. If verification of the certificate fails, merchant computer system 130 concludes that the capture response is counterfeit and raises an error condition.

[0160] In function block 1440, merchant computer system 130 validates payment gateway digital signature 1325. Merchant computer system 130 performs this validation by calculating a message digest over the contents of the combined basic authorization request 1310 and the signature public key certificate 1320. Merchant computer system 130 then decrypts digital signature 1325 to obtain a copy of the equivalent message digest calculated by payment gateway computer system 140 in function block 1255. If the two message digests are equal, the digital signature 1325 is validated. If validation fails, merchant computer system 130 concludes that the authorization response is counterfeit and raises an error condition.

[0161] In function block 1450, merchant computer system 130 stores capture response for later use in by legacy system accounting programs, e.g., to perform reconciliation between the merchant operating merchant computer system 130 and the financial institution from whom payment was requested, thereby completing the transaction.

[0162] The system of the present invention permits immediate deployment of a secure payment technology architecture such as the SET architecture without first establishing a public-key encryption infrastructure for use by consumers. It thereby permits immediate use of SET-compliant transaction processing without the need for consumers to migrate to SET-compliant application software.

[0163] All publications and existing subsystems mentioned in this specification are hereby incorporated by reference to the same extent as if each individual publication or existing subsystem were specifically and individually indicated to be incorporated by reference.

[0164] While various embodiments of a preferred embodiment have been described above, it should be understood that they have been presented by way of example only, and not limitation. Thus, the breadth and scope of a preferred embodiment should not be limited by any of the above described exemplary embodiments, but should be defined only in accordance with the following claims and their equivalents.

What is claimed is:

1. A method for initiating secure communication between a first and a second computer connected to a network for receiving and transmitting payment information, comprising the steps of:

(a) establishing a communication between said first and said second computer via said network;

(b) identifying an encryption procedure and a decryption procedure utilized by said first and said second computer;

(c) transmitting encrypted payment information from said first computer to said second computer;

(d) receiving said encrypted payment information at said second computer and decrypting the payment information utilizing the decryption procedure; and
(e) repackaging said payment information to comply with a third party secure protocol for further payment processing.

2. The method as recited in claim 1, including the step of utilizing the Internet for transmitting information between said first and said second computer systems.

3. The method as recited in claim 1, including the step of transmitting from the second computer system to a third computer system for authorizing or denying credit in the payment processing.

4. The method as recited in claim 1, wherein the secure third party protocol is a Secure Electronic Transaction protocol.

5. A method for initiating secure communication between a first and a second computer connected to a network for receiving and transmitting payment information, comprising the steps of:

   (a) obtaining client information for use in said secure communication between a first and a second computer;

   (b) establishing a communication between said first and said second computer via said network; and

   (c) repackaging said payment information to comply with a third party secure protocol for further payment processing.

6. The method as recited in claim 5, including:

   (d) transmitting encrypted payment information from said first computer to said second computer;

   (e) receiving said encrypted payment information at said second computer and decrypting the payment information utilizing the decryption procedure; and

   (f) performing further payment processing on said decrypted information.

7. The method as recited in claim 5, wherein said client information is obtained via a telephone, fax machine or electronic mail.

8. The method as recited in claim 5, wherein an electronic signature is utilized to authenticate payment processing.

9. The method as recited in claim 5, wherein said client information is obtained via a secure general purpose protocol.

10. The method as recited in claim 5, further comprises reversing previous payment transactions.

11. Apparatus for initiating payment in a computer under the control of software with an attached display and an input device connected to a network for receiving and transmitting network information, comprising:

    (a) means for establishing a communication between said first and said second computer via said network;

    (b) means for identifying an encryption procedure and a decryption procedure utilized by said first and said second computer;

    (c) means for transmitting encrypted payment information from said first computer to said second computer;

    (d) means for receiving said encrypted payment information at said second computer and decrypting the payment information utilizing the decryption procedure; and

    (e) means for repackaging said payment information to comply with a Secure Electronic Transaction protocol for further payment processing.

12. The apparatus as recited in claim 11, including means for utilizing the Internet for transmitting information between said first and said second computer systems.

13. The apparatus as recited in claim 12, including means for transmitting from the second computer system to a third computer system for authorizing or denying credit in the payment processing.

14. An apparatus for initiating secure communication between a first and a second computer connected to a network for receiving and transmitting payment information, comprising:

    (a) communication hardware utilized by a client to communicate information for use in said secured communication between a first and a second computer;

    (b) a computer under the control of software which establishes secure communication between said first and said second computer via said network; and

    (c) a computer under the control of software which repackages said payment information to comply with a third party secure protocol for further payment processing.

15. A computer program embodied on a computer-readable medium for effecting purchase transactions by a customer system at a merchant system and effecting payment for said transactions by a payment system, comprising:

    (a) a code segment for controlling secure communication of a purchase request from said customer system to said merchant system, including providing payment information from said customer system to said merchant system;

    (b) a code segment for controlling secure processing of said payment information by said merchant system to generate a payment authorization request and securely transmit said payment authorization request from said merchant system to said payment system;

    (c) a code segment for controlling the secure processing of said payment authorization request by said payment system to generate a payment authorization response authorizing said purchase and securely transmit said payment authorization response to said merchant system pursuant to which said merchant system fills said purchase request;

    (d) a code segment for controlling the secure processing of said payment authorization response by said merchant system to generate a payment capture request and securely transmit said payment capture request to said payment system;

    (e) a code segment for controlling the secure processing of said payment capture request by said payment system to generate a payment capture response authorizing payment and transmit said payment capture response to said merchant system; and

    (f) a code segment for controlling the secure processing of said payment capture response by said merchant system to effect payment to the merchant system for filling said purchase request.
16. A computer program embodied on a computer-readable medium as recited in claim 15, wherein said secure protocol is a Secure Electronic Transaction protocol.

17. A computer program embodied on a computer-readable medium as recited in claim 15, wherein said secure protocol is a Secure Transaction Technology protocol.

18. A computer program embodied on a computer-readable medium as recited in claim 15, wherein said secure protocol is a Secure Electronic Payments Protocol.

19. A computer program embodied on a computer-readable medium as recited in claim 15, wherein said information includes payment administration information.

20. A computer program embodied on a computer-readable medium for initiating payment in a computer under the control of software with an attached display and an input device connected to a network for receiving and transmitting network information, comprising:

(a) a code segment for establishing a communication between said first and said second computer via said network;

(b) a code segment for identifying an encryption procedure and a decryption procedure utilized by said first and said second computer;

(c) a code segment for transmitting encrypted payment information from said first computer to said second computer;

(d) a code segment for receiving said encrypted payment information at said second computer and decrypting the payment information utilizing the decryption procedure; and

(e) a code segment for repackaging said payment information to comply with a secure protocol for further payment processing.

21. A computer program embodied on a computer-readable medium for effecting purchase transactions by a merchant computer system and payment for said transactions by a payment computer system, comprising:

(a) a code segment controlling communication between said customer and said merchant computer system for effecting purchase requests, including providing payment information from said customer to said merchant computer system; and

(b) a code segment controlling secure communication between said merchant computer system and said payment computer system for effecting operation on said payment information by said merchant computer systems to obtain administration information from said payment computer system to said merchant computer system pursuant to which said merchant computer system completes said purchase request.